Model-Based Calibration Toolbox

For Use with MATLAB® and Simulink®

- Computation
- Visualization
- Programming
- Simulation

Getting Started

Version 3

The MathWorks

How to Contact The MathWorks:

	`www.mathworks.com`	Web
	`comp.soft-sys.matlab`	Newsgroup
@	`support@mathworks.com`	Technical Support
	`suggest@mathworks.com`	Product enhancement suggestions
	`bugs@mathworks.com`	Bug reports
	`doc@mathworks.com`	Documentation error reports
	`service@mathworks.com`	Order status, license renewals, passcodes
	`info@mathworks.com`	Sales, pricing, and general information
☎	508-647-7000	Phone
	508-647-7001	Fax
✉	The MathWorks, Inc. 3 Apple Hill Drive Natick, MA 01760-2098	Mail

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

Getting Started with the Model-Based Calibration Toolbox

Trademarks

Patents

Revision History

November 2005	Online only	New for Version 3.0 (Release 14SP3+)

Contents

1 Introduction

2 Gasoline Case Study

3 Diesel Case Study

4 Command-Line Interface to the Model-Based Calibration Toolbox

5 Tutorial: Model Quickstart

6 Tutorial: Design of Experiment

7 Tutorial: Data Editor

8 Tutorial: Feature Calibration

9 Tutorial: Tradeoff Calibration

10 Tutorial: Data Sets

11 Tutorial: Filling Tables from Data

Tutorial: Optimization and Automated Tradeoff

12

1

Introduction

The following sections introduce the Model-Based Calibration toolbox.

What Is the Model-Based Calibration Toolbox?

The Model-Based Calibration Toolbox contains tools for design of experiment, statistical modeling, and calibration of complex systems. It has two main user interfaces:

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

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

High accuracy engine models are a key component for reducing calibration effort and engine development time.

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

About the Model Browser

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

- Design of experiment tools can drastically reduce expensive data collection time.
- You can create and evaluate optimal, space-filling, and classical designs, and constraints can be designed or imported.

- Hierarchical statistical models can capture the nature of variability inherent in engine data, accounting for variation both within and between tests.
- The Model Browser has powerful, flexible tools for building, comparing, and evaluating statistical models and experimental designs.
- There is an extensive library of prebuilt model types and the capability to build user-defined models.
- You can export models to MATLAB®, Simulink®, or CAGE.

Starting the Model Browser

To start the application, type

```
mbcmodel
```

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 Model Browser models. From these models, CAGE builds steady-state ECU calibrations.

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

Starting the CAGE Browser

To start the application, type

```
cage
```

at the MATLAB command prompt.

How to Use This Manual

Learning the Model-Based Calibration Toolbox

The case studies chapters contain step-by-step examples that guide you through using the whole toolbox to solve engine modeling and calibration problems:

- Chapter 2, “Gasoline Case Study”
- Chapter 3, “Diesel Case Study”

The following tutorial chapter explains how to get started using the command-line interface to the toolbox:

- Chapter 4, “Command-Line Interface to the Model-Based Calibration Toolbox”

Learning the Model Browser

These tutorials guide you through using specific Model Browser tools:

- Chapter 5, “Tutorial: Model Quickstart ” provides a quick introduction to modeling with the toolbox. The tutorial describes how to set up and view a two-stage model using engine data.
- Chapter 6, “Tutorial: Design of Experiment” covers the tools for Design of Experiments with a step-by-step guide to setting up, viewing, and comparing one of each of the design types: classical, space-filling, and optimal. The tutorial also describes how to define and apply constraints and export designs.
- Chapter 7, “Tutorial: Data Editor” is a guide to using the Data Editor to load and manipulate data for modeling. You can load data from files or the workspace or custom Excel sheets. You can view plots of the data and define new variables and filters. You can store and import user-defined variables and filters, and define test groupings.

Learning CAGE

These tutorials guide you through using specific tools in CAGE:

- Chapter 8, “Tutorial: Feature Calibration” describes how to set up and calibrate lookup tables by reference to a model.
- Chapter 9, “Tutorial: Tradeoff Calibration” describes how to calibrate lookup tables using tradeoff calibrations.
- Chapter 10, “Tutorial: Data Sets” describes how to validate calibrations using experimental data.
- Chapter 11, “Tutorial: Filling Tables from Data” describes how to fill lookup tables using experimental data.
- Chapter 12, “Tutorial: Optimization and Automated Tradeoff” describes how to set up and run optimizations, including single objective, multiobjective, sum and user-defined optimization, and automated tradeoff.

Training Material

The files for the tutorial chapters are in the *matlabroot*`/toolbox/mbc/mbctraining` directory.

System Requirements

This section lists the following:

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

Operating System Requirements

The Model-Based Calibration Toolbox is a Windows 32-bit PC only product.

You can see the system requirements for MATLAB online at `http://www.mathworks.com/products/system.shtml/Windows`

Required MathWorks Products

The Model-Based Calibration Toolbox 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

Note If you want to import Excel files or use the custom Excel file facility of the toolbox, you must also have the Excel application.

2

Gasoline Case Study

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

Optimized Calibration (p. 2-56)	You use CAGE to create an optimized calibration based on the models exported from the Model Browser. This section is an overview of the required steps and the benefits of automated calibration.
Importing Models into CAGE (p. 2-59)	Importing models into CAGE.
Setting Up Calibration Tables to Fill (p. 2-61)	Creating and duplicating tables to fill with optimization results.
Setting Up the Optimization (p. 2-63)	Using the Optimization Wizard to set up the optimization.
Defining Variable Values (p. 2-67)	Defining the operating points where you want the optimization to run.
Running the Optimization (p. 2-70)	Running the optimization and viewing the results.
Filling Tables with Optimization Results (p. 2-72)	Filling tables with the results of the optimization.
MBT Spark Estimator Problem (p. 2-74)	Using the Feature Fill Wizard to fill the tables of a spark estimator feature.

Gasoline Case Study Overview

This case study is an example of a gasoline engine control calibration, for a 2.2 L naturally aspirated 4-valve overhead-cam spark ignition engine with twin-independent variable valve timing. This example takes you through the following steps:

1. Create a design for your experiment — see “Designing the Experiment” on page 2-7.
2. Import the resulting data (taken using the design) to examine and filter it in preparation for modeling — see “Importing and Filtering Data” on page 2-17.
3. Make statistical models based on the data.
   a. “How Is a Two-Stage Model Constructed?” on page 2-26
   b. “Building the Models” on page 2-30
   c. “Selecting Models” on page 2-37
   d. “Adding a New Response Model” on page 2-47
   e. “Creating Boundary Models” on page 2-50
4. Export these models to the CAGE part of the toolbox to generate optimal calibration tables — see “Exporting the Models” on page 2-54.
5. The Model Browser section of the case study involves design of experiment, data handling, and model construction and export. In the CAGE browser section of the case study you use the models to complete the optimization of the calibration tables, see “Optimized Calibration” on page 2-56.

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

- “Why Use Design of Experiment and Engine Modeling?” on page 2-4
- “Problem Definition” on page 2-4
- “Introduction to Two-Stage Modeling” on page 2-5

Why Use Design of Experiment and Engine Modeling?

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

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

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

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

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

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

Problem Definition

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

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

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

- Constrain Exhaust Temperature <=1200°C

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

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

Introduction to Two-Stage Modeling

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

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

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

Designing the Experiment

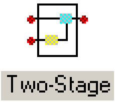
Creating a design in the Model Browser comprises several steps. You need to open the tool and create a new two-stage test plan. Then you need to enter the ranges and names of the input variables being used and choose a default model. Then you can create an initial design and set up the constraints on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

Creating a Test Plan

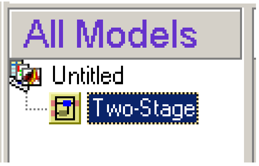
1. Start the Model Browser part of the toolbox by typing `mbcmodel` at the MATLAB command line.

2. From the startup project view, to create a new test plan, click **New** in the **Test Plans** list pane at the bottom.

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

3. Click the `Two-Stage` test plan icon  and click **OK**.

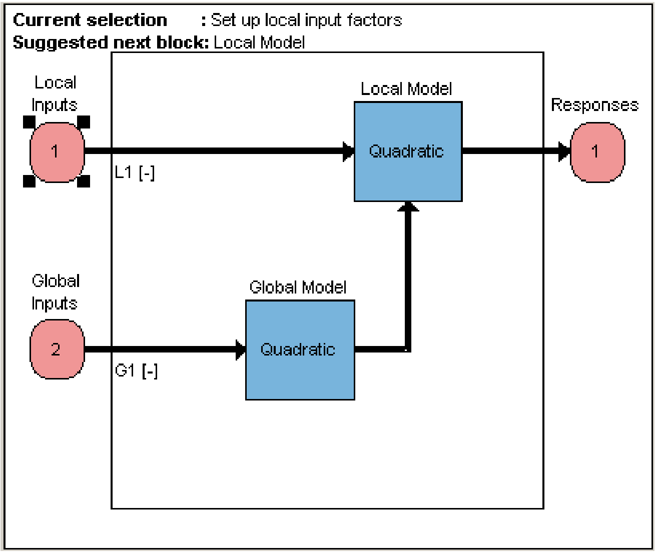
   The default name of the new test plan, `Two-Stage`, appears in the Model Browser tree, in the **All Models** pane.

4. Highlight this node of the tree , `Two-Stage`, by clicking it. The **Model Browser** window displays a diagram representing the two-stage model.

Specifying the Model Inputs

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

1 To specify spark angle as the local input, double-click the Local Inputs icon on the model diagram.



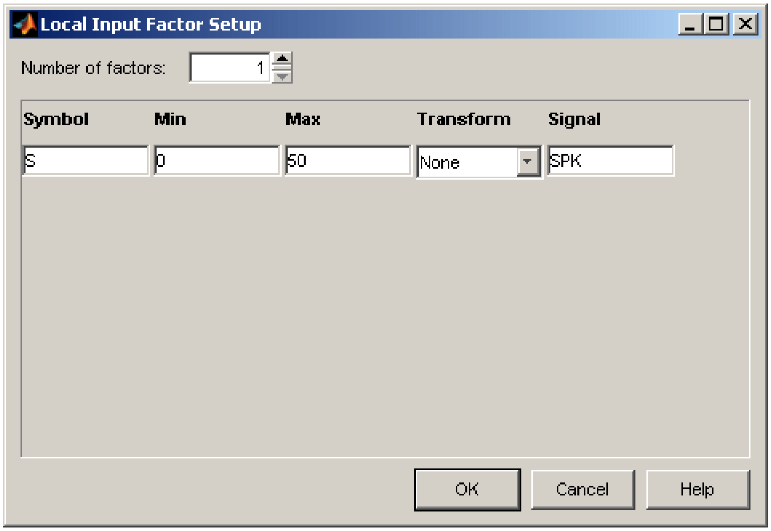
Current selection : Set up local input factors
Suggested next block: Local Model

The Local Input Factor Setup dialog box appears.

a Set **Symbol** to S.

b Set **Signal** to SPK. This is optional and matches the raw data.

c Set the range you want to model by changing **Max** to 50 (and leave **Min** at 0).



Local Input Factor Setup

Number of factors: 1

Symbol	Min	Max	Transform	Signal
S	0	50	None	SPK

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

Notice that the new name of the local model input now appears on the two-stage model diagram.

The global inputs are the variables that are held constant at each operating point while spark is swept. In this case, these global variables are engine speed, scaled throttle area, intake cam angle, and exhaust cam angle.

3 To specify the global inputs, double-click the Global Inputs icon on the model diagram.

The Global Input Factor Setup dialog box appears.

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

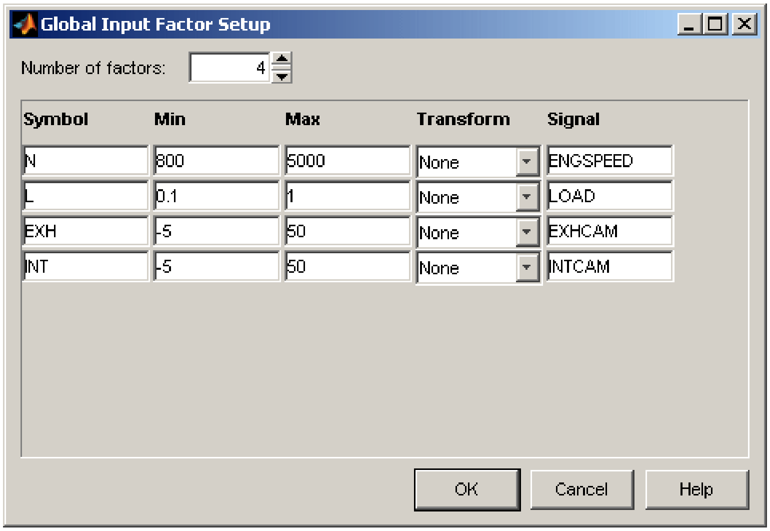
a Click the up arrow button to increase the number of factors to four.

b Edit the four factors to create the engine model input. In each case, change the symbols, signal names, and ranges to the following:

Symbol	Signal	Min	Max
N	ENGSPEED	800	5000
L	LOAD	0.1	1
EXH	EXHCAM	-5	50
INT	INTCAM	-5	50

Load = aircharge/maximum aircharge.

Cam angles are in units of degrees crankshaft, with positive values indicating retard from base timing.



Global Input Factor Setup

Number of factors: 4

Symbol	Min	Max	Transform	Signal
N	800	5000	None	ENGSPEED
L	0.1	1	None	LOAD
EXH	-5	50	None	EXHCAM
INT	-5	50	None	INTCAM

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

4 To change the global model type, double-click the Global Model block in the two-stage model diagram. The Global Model Setup dialog box appears.

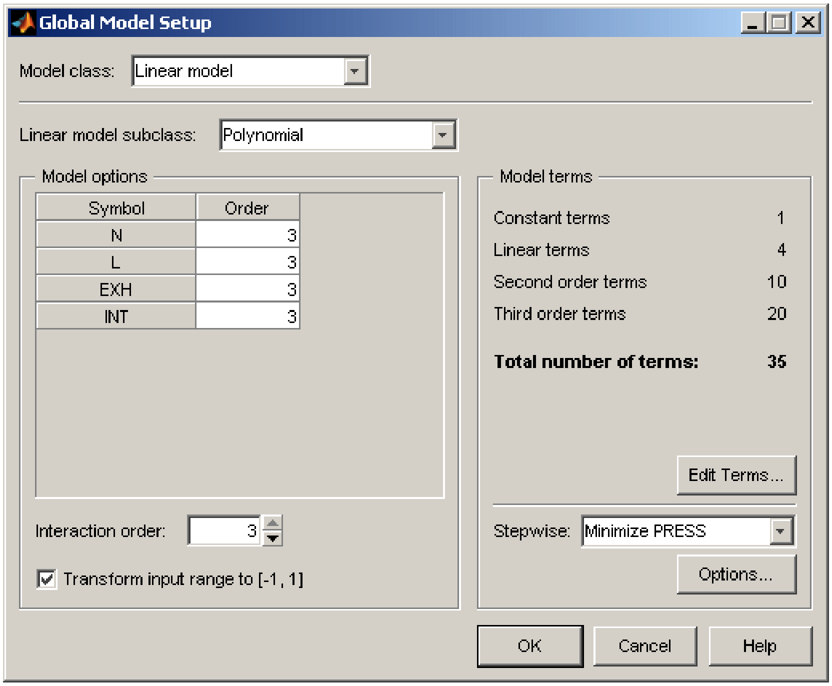
Deciding on the model to design for is vital for optimal designs only, when you already have some knowledge of how you expect the system to behave. In these cases, optimal designs can help you find the most efficient points for fitting the most robust models. In this case, you will create a space-filling design, which is best for exploring a new system where prior knowledge is low and you want to spread the available points to capture as much information as possible. These do not depend on model type; however, for this example you set a new model type now.

Remember that the statistical usefulness of different designs depends on the model type. For example, if you think you need cubic instead of quadratic in a factor, the number of points required rises dramatically and this has a highly adverse effect on the statistical quality of the designs. However, you do need to bear in mind that the final model will not be either of the possibilities listed here, because some terms will have been removed, or it might even be an RBF. You choose the most suitable model you can to construct a design, then when you have collected the data, you might find that a different model type produces the best fit.

5 `Polynomial` should already be selected from the **Linear model subclass** pop-up menu. Under **Model options**, set the order for each of the four variables to three, to fit cubic curves in each case.

6 Set **Stepwise** to `Minimize PRESS` (PREdicted Sum Square error).

This option will be important when you are fitting models to the data. You use the **Stepwise** feature to avoid overfitting the data; that is, you do not want to use unnecessarily complex models that “chase points” in an attempt to model random effects. Predicted error sum of squares (PRESS) is a measure of the predictive quality of a model. `Minimize PRESS` throws away terms in the model to improve its predictive quality, removing those terms that reduce the PRESS of the model. See “PRESS statistic” in the Model Browser documentation. You can also open the **Stepwise** window after model fitting to try to improve the fit with the **Stepwise** tools.



Model class: Linear model

Linear model subclass: Polynomial

Symbol	Order
N	3
L	3
EXH	3
INT	3

Interaction order: 3

☑ Transform input range to [-1, 1]

Constant terms	1
Linear terms	4
Second order terms	10
Third order terms	20
Total number of terms:	**35**

Stepwise: Minimize PRESS

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

Creating Designs

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

1 Right-click the global model in the diagram and choose **Design Experiment**.

The Design Editor appears.

2 Click the 📄 button in the toolbar or select **File > New Design**. A new node called `Linear Model Design` appears.

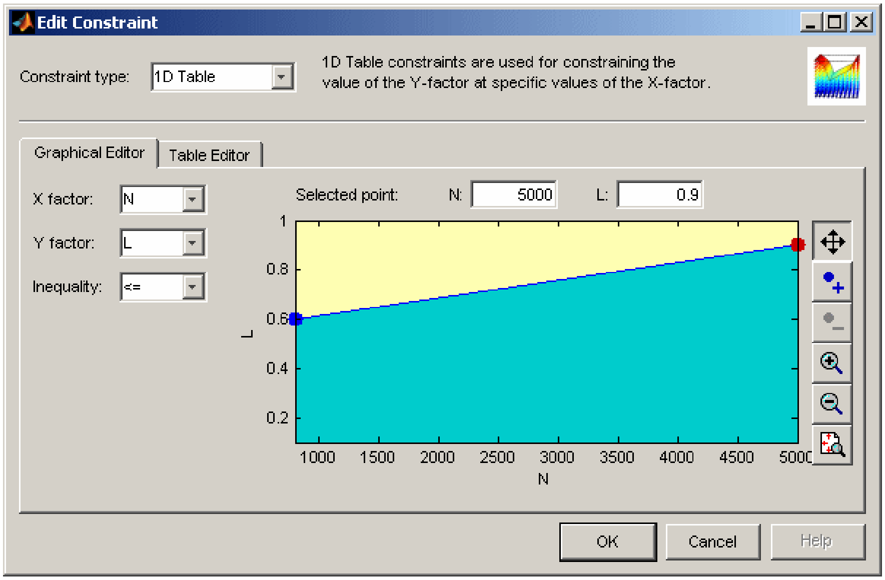
The new `Linear Model Design` node is automatically selected. An empty Design Table appears because you have not yet chosen a design.

3 Constrain the design space. Select **Edit > Constraints**.

4 The Constraints Manager dialog appears. Click **Add** to create a new constraint. The Constraint Editor appears.

5 You will add the default constraint type, a 1-D table constraint.

- **a** Select `N` and `L` from the `X` and `Y factor` drop-down menus.
- **b** Select the **Table Editor** tab and edit the **Number of breakpoints** to `2`, and click **Span range**.
- **c** On the **Graphical Editor** tab, click Move Points ✥, then click and drag the left point (where N=800) down to L=0.6. You can also enter the value in the **L** edit box, or in the table on the **Table Editor** tab.
- **d** Click and drag the right point (where N=5000) to L=0.9 (or enter 0.9 in the L edit box). The Constraint Editor should look like the following.



Edit Constraint

Constraint type: 1D Table

1D Table constraints are used for constraining the value of the Y-factor at specific values of the X-factor.

X factor: N

Y factor: L

Inequality: <=

Selected point: N: 5000 L: 0.9

e Click **OK** to add the new constraint, then click **OK** to dismiss the Constraint Manager.

6 Examine the constrained design space by right-clicking the title bar of a Design Table view and selecting **Current View > 3D Constraints**.

7 Select **Design > Space Filling > Design Browser**, or click the Space Filling Design button on the toolbar.

The Space Filling Design Browser appears.

Space-filling designs are best when there is little or no information about the underlying effects of factors on responses. For example, they are most useful when you are faced with a new type of engine, with little knowledge of the operating envelope. These designs do not assume a particular model form. The aim is to spread the points as evenly as possible around the operating space. You can use a mix-and-match approach: start with a space-filling design to survey the space, then continue testing with an optimal design once you have more understanding of the response and constraints. Once you have an idea of what model type will fit the response best, you can optimally add points in the most efficient places for the most robust model fit.

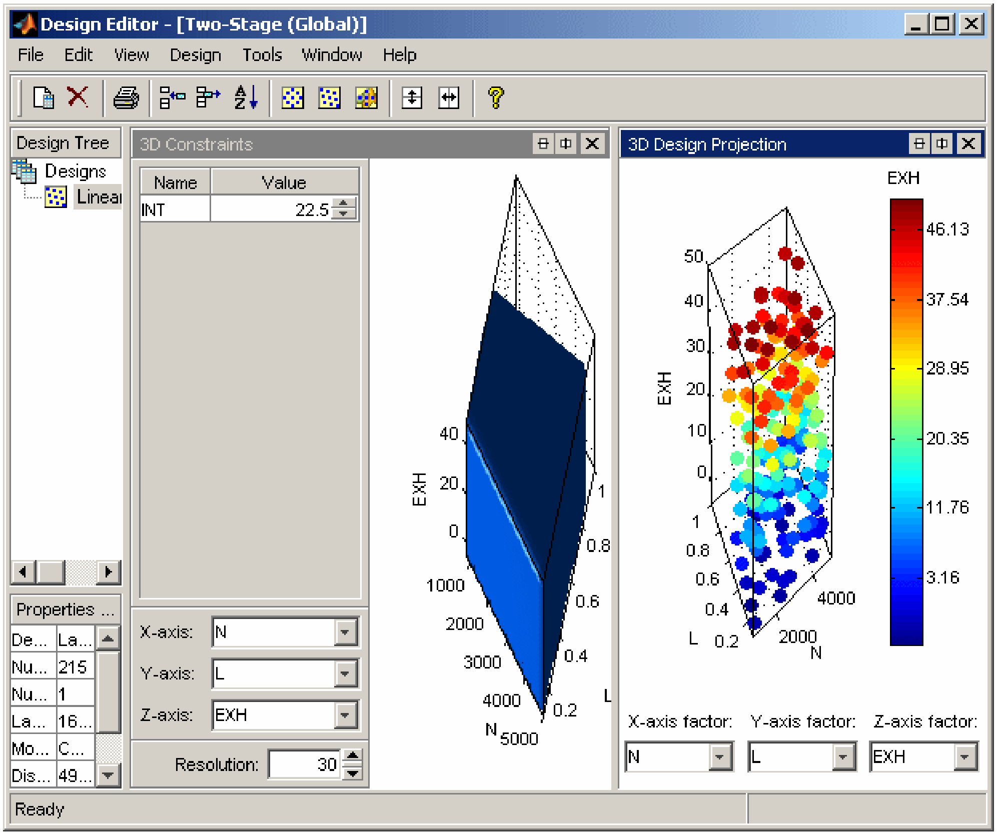
The most important thing to decide is how many design points you want. Testing is expensive and time-consuming, so you need to bear in mind how many points you have time for. When you consider the number of points, you also need to remember that a sweep will be done at each point and this will take some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

8 Enter 250 for the number of points and press **Enter**.

9 Click the 2-D and 3-D tabs to examine the plots of new design distribution. Note that the number of points will not be exactly the number you entered, because some of the points will be removed if they fall outside the constraint. You can add more points now or later if required to compensate for this.

10 Click **OK**. A space-filling design is constructed, using the latin hypercube sampling method.

11 Right-click a view and select **Split View > 3D Design Projection** to view the design points.



Design Editor - [Two-Stage (Global)]

Design Tree

Designs

Linear

3D Constraints

Name	Value
INT	22.5

X-axis: N

Y-axis: L

Z-axis: EXH

Resolution: 30

Properties ...

De...	La...
Nu...	215
Nu...	1
La...	16...
Mo...	C...
Dis...	49...

3D Design Projection

X-axis factor: N

Y-axis factor: L

Z-axis factor: EXH

Ready

You can use the Design Editor to make a selection of child designs to compare. When you have chosen the best design you can export it to file.

In this case, the example design for this case study is `ConstrainedVVTDesign` 253 point latin hypercube. Import this design to the project as follows:

1 Select **File > Import Design**.

2 Select `Comma separated format file (.csv)` in the **Import from** drop-down menu.

3 Browse to the design file `ConstrainedVVTDesign.csv` in the `mbctraining` directory, select it and click **Open**, then click **OK** to import the design. A new design node appears.

4 Rename the imported design (click, and press **F2**) `Constrained_VVT`. You can right-click to change the Current View to examine the design points in 2D and 3D.

5 Choose `Constrained_VVT` as the preferred design for future reference by selecting **Edit > Select As Best**.

6 Close the Design Editor.

Data Source

The data was collected using the constrained space-filling design. The points specified in the design were run using a GT-Power simulation from Gamma Technologies (see `http://www.gtisoft.com/broch_gtpower.html`). Simulation tools in MATLAB and Simulink controlled the GT Power model, running on a cluster of three dual processor PIII 800-MHz machines. The simulation time per point was 2.88 hours. The GT-Power model used predictive combustion, which gives good realistic results but is computationally expensive. Ten cycles were run at each point after attaining steady-state conditions.

Importing and Filtering Data

We provide an example data file, resulting from the example experimental design.

Import and View Data

1 In the Model Browser, click the top project node in the **All Models** tree to go to the project view.

2 Click New Data 📄 in the toolbar or select **Data > New Data**.

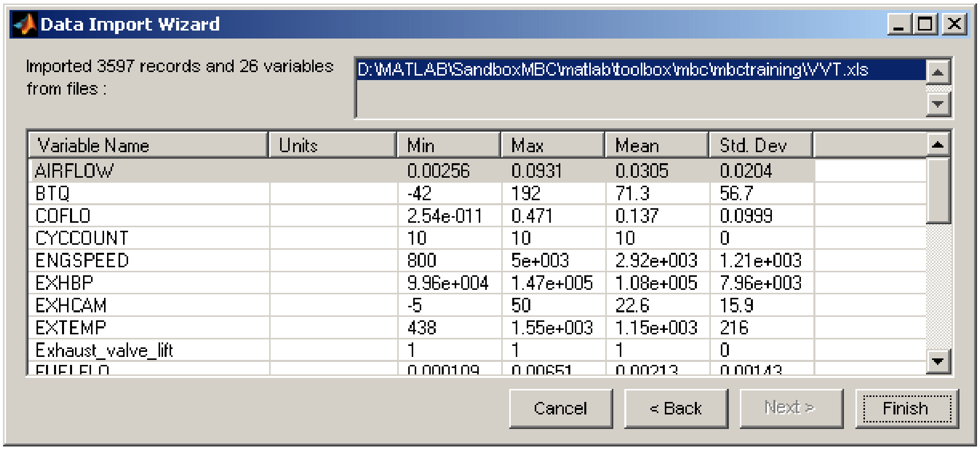
The Data Editor appears.

3 Click the Open File icon in the toolbar 📂 to load data from a file.

The Data Import Wizard appears to select a file.

4 Use the Browse button to find and select the `VVT.xls` data file in the `mbctraining` folder. Double-click to load the file, and click **Next**.

5 The Data Import Wizard displays a summary screen showing the total number of records and variables imported, and you can view each variable's range, mean, standard deviation, and units in the list box. Click **Finish** to accept the data.



Imported 3597 records and 26 variables from files :

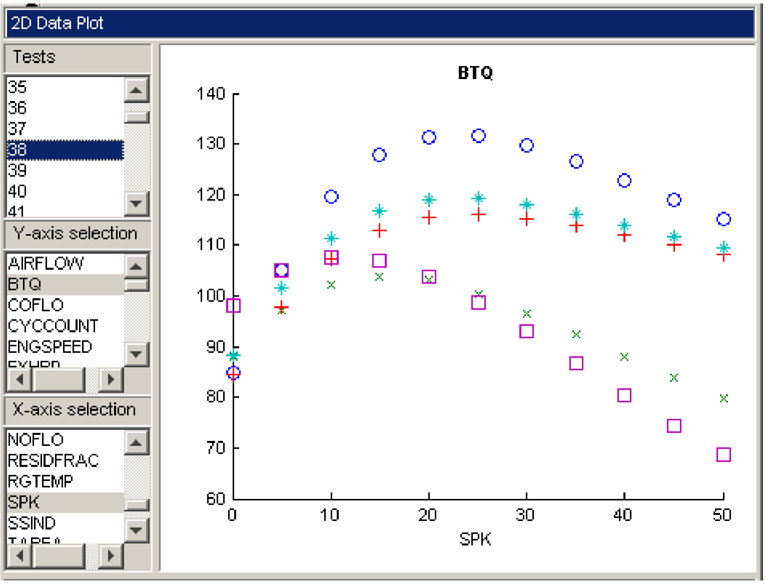
D:\MATLAB\SandboxMBC\matlab\toolbox\mbc\mbctraining\VVT.xls

Variable Name	Units	Min	Max	Mean	Std. Dev
AIRFLOW		0.00256	0.0931	0.0305	0.0204
BTQ		-42	192	71.3	56.7
COFLO		2.54e-011	0.471	0.137	0.0999
CYCCOUNT		10	10	10	0
ENGSPEED		800	5e+003	2.92e+003	1.21e+003
EXHBP		9.96e+004	1.47e+005	1.08e+005	7.96e+003
EXHCAM		-5	50	22.6	15.9
EXTEMP		438	1.55e+003	1.15e+003	216
Exhaust_valve_lift		1	1	1	0
FUELFLO		0.000109	0.00651	0.00213	0.00143

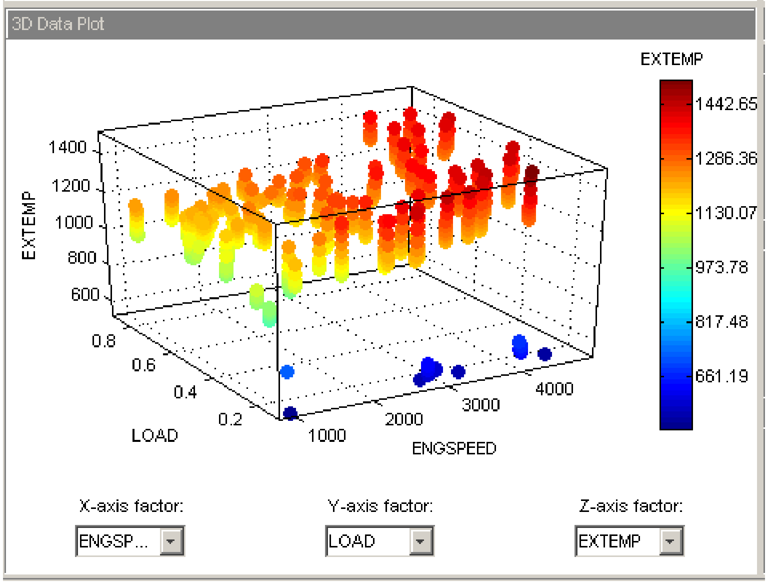
The Data Editor is a powerful tool for displaying and sorting your data. You can use the right-click context menu to split the views, or use the toolbar buttons or the **View** menu. You can choose 2-D plots, 3-D plots, multiple data plots, data tables, and list views of filters, variables, test filters,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

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

7 Click in the left lists to plot torque (`BTQ`) against spark (`SPK`), then select one or more tests to display.



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



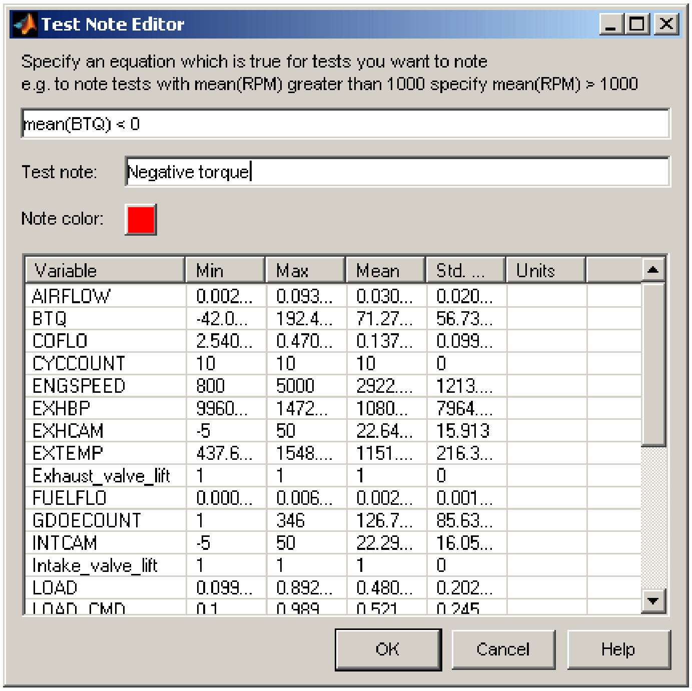
9 Right-click and select **Split View > Notes View** to split the currently selected view and add a test notes list view. This is empty until you add any test notes.

10 Select **Tools > Test Notes > Add**.

The Test Note editor appears.

a Enter an expression that defines the tests you want to note; for example `mean(BTQ) < 0` will evaluate the mean torque for each test and note those tests where the value is less than zero.

b Enter the text for this note in the edit box, e.g., `Negative Torque`.



Specify an equation which is true for tests you want to note
e.g. to note tests with mean(RPM) greater than 1000 specify mean(RPM) > 1000

mean(BTQ) < 0

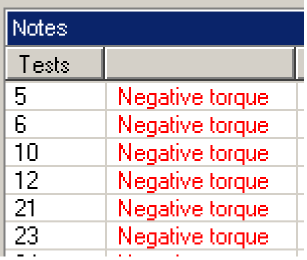
Test note: Negative torque

Note color: 🟥

Variable	Min	Max	Mean	Std. ...	Units
AIRFLOW	0.002...	0.093...	0.030...	0.020...	
BTQ	-42.0...	192.4...	71.27...	56.73...	
COFLO	2.540...	0.470...	0.137...	0.099...	
CYCCOUNT	10	10	10	0	
ENGSPEED	800	5000	2922....	1213....	
EXHBP	9960...	1472...	1080...	7964....	
EXHCAM	-5	50	22.64...	15.913	
EXTEMP	437.6...	1548....	1151....	216.3...	
Exhaust_valve_lift	1	1	1	0	
FUELFLO	0.000...	0.006...	0.002...	0.001...	
GDOECOUNT	1	346	126.7...	85.63...	
INTCAM	-5	50	22.29...	16.05...	
Intake_valve_lift	1	1	1	0	
LOAD	0.099...	0.892...	0.480...	0.202...	
LOAD_CMD	0.1	0.989	0.521	0.245	

c Click **OK** to apply the test note.

11 Note that the new test note appears in the Notes list view.



Tests	
5	Negative torque
6	Negative torque
10	Negative torque
12	Negative torque
21	Negative torque
23	Negative torque

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

Filter Data

1 If you decide certain operating points are unsuitable for modeling, for instance, unstable points on the edge of the engine's operating envelope, you can use the Data Editor tools to help you identify and remove them. You can remove suspect data in the following ways:

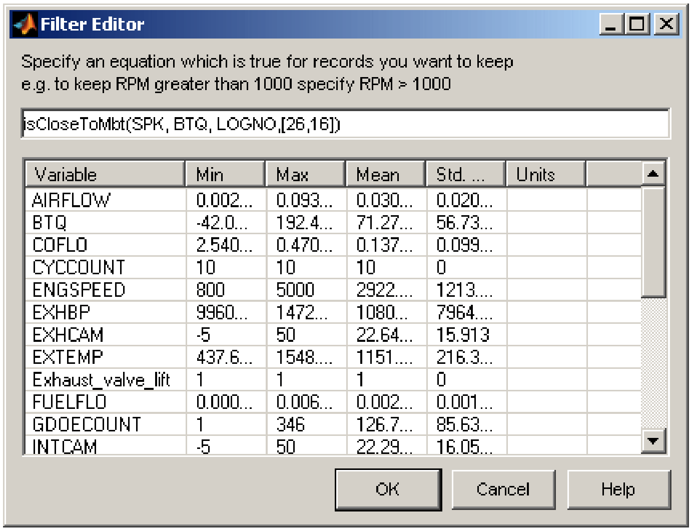
   a You can remove individual points by selecting them as outliers in 2D and multiple data plots by clicking, then selecting **Tools > Filters > Remove Outliers**. You can always replace them again with the **Tools** menu.

   b You can define filters to remove all data points that do not fulfil a certain expression. First, locate `isCloseToMbt.m` in the `mbctraining` directory and copy it to your working folder. This filter can only be applied successfully if the function `isCloseToMbt` is on your MATLAB path. You can use this function to remove points that are unrealistically far from maximum brake torque (MBT) — recall that this is simulation data.

   c Select **Tools > Filters > Add**.

   The Filter Editor appears. You can enter an expression that defines the records you want to keep. You can use any MATLAB function for filtering. Enter the following expression, which keeps records 26 degrees before maximum torque and 16 degrees after:

```
isCloseToMbt(SPK, BTQ, LOGNO,[26,16])
```

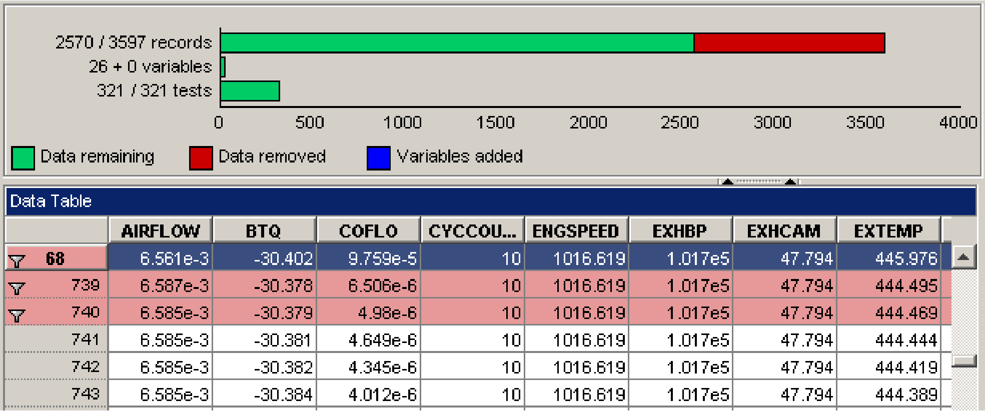


Specify an equation which is true for records you want to keep
e.g. to keep RPM greater than 1000 specify RPM > 1000

```
isCloseToMbt(SPK, BTQ, LOGNO,[26,16])
```

Variable	Min	Max	Mean	Std. ...	Units
AIRFLOW	0.002...	0.093...	0.030...	0.020...	
BTQ	-42.0...	192.4...	71.27...	56.73...	
COFLO	2.540...	0.470...	0.137...	0.099...	
CYCCOUNT	10	10	10	0	
ENGSPEED	800	5000	2922....	1213....	
EXHBP	9960...	1472...	1080...	7964....	
EXHCAM	-5	50	22.64...	15.913	
EXTEMP	437.6...	1548....	1151....	216.3...	
Exhaust_valve_lift	1	1	1	0	
FUELFLO	0.000...	0.006...	0.002...	0.001...	
GDOECOUNT	1	346	126.7...	85.63...	
INTCAM	-5	50	22.29...	16.05...	

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



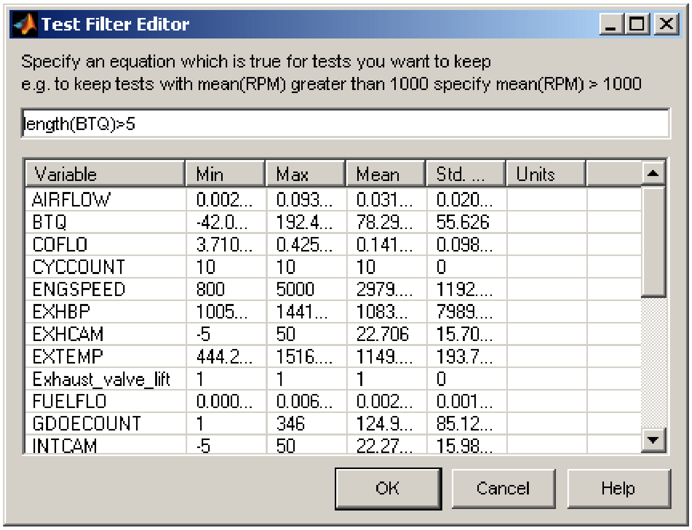
	AIRFLOW	BTQ	COFLO	CYCCOU...	ENGSPEED	EXHBP	EXHCAM	EXTEMP
▽ 68	6.561e-3	-30.402	9.759e-5	10	1016.619	1.017e5	47.794	445.976
▽ 739	6.587e-3	-30.378	6.506e-6	10	1016.619	1.017e5	47.794	444.495
▽ 740	6.585e-3	-30.379	4.98e-6	10	1016.619	1.017e5	47.794	444.469
741	6.585e-3	-30.381	4.649e-6	10	1016.619	1.017e5	47.794	444.444
742	6.585e-3	-30.382	4.345e-6	10	1016.619	1.017e5	47.794	444.419
743	6.585e-3	-30.384	4.012e-6	10	1016.619	1.017e5	47.794	444.389

d If you decide that whole tests should be excluded, you can define a test filter to remove them. Select **Tools > Test Filter > Add**. The Test Filter Editor appears.

To keep tests with positive mean torque, enter `mean(BTQ)>0` and click **OK**.

e You want to keep only those tests with sufficient points to fit the model (at least 5 points). Select **Tools > Test Filter > Add**.

Enter `length(BTQ)>5` and click **OK**.



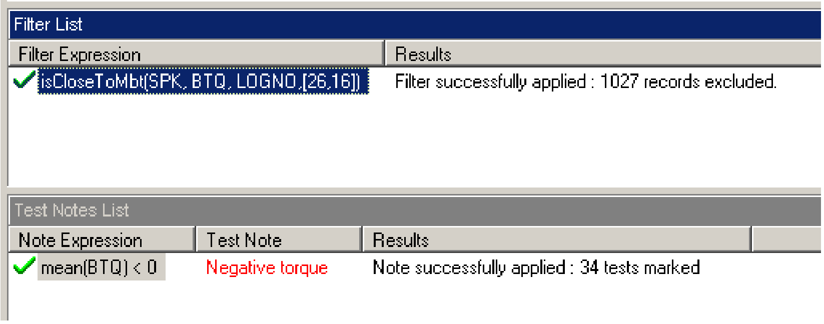
Test Filter Editor

Specify an equation which is true for tests you want to keep
e.g. to keep tests with mean(RPM) greater than 1000 specify mean(RPM) > 1000

length(BTQ)>5

Variable	Min	Max	Mean	Std. ...	Units
AIRFLOW	0.002...	0.093...	0.031...	0.020...	
BTQ	-42.0...	192.4...	78.29...	55.626	
COFLO	3.710...	0.425...	0.141...	0.098...	
CYCCOUNT	10	10	10	0	
ENGSPEED	800	5000	2979....	1192....	
EXHBP	1005...	1441...	1083...	7989....	
EXHCAM	-5	50	22.706	15.70...	
EXTEMP	444.2...	1516....	1149....	193.7...	
Exhaust_valve_lift	1	1	1	0	
FUELFLO	0.000...	0.006...	0.002...	0.001...	
GDOECOUNT	1	346	124.9...	85.12...	
INTCAM	-5	50	22.27...	15.98...	

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



Filter List

Filter Expression	Results
✓ isCloseToMbt(SPK, BTQ, LOGNO,[26,16])	Filter successfully applied : 1027 records excluded.

Test Notes List

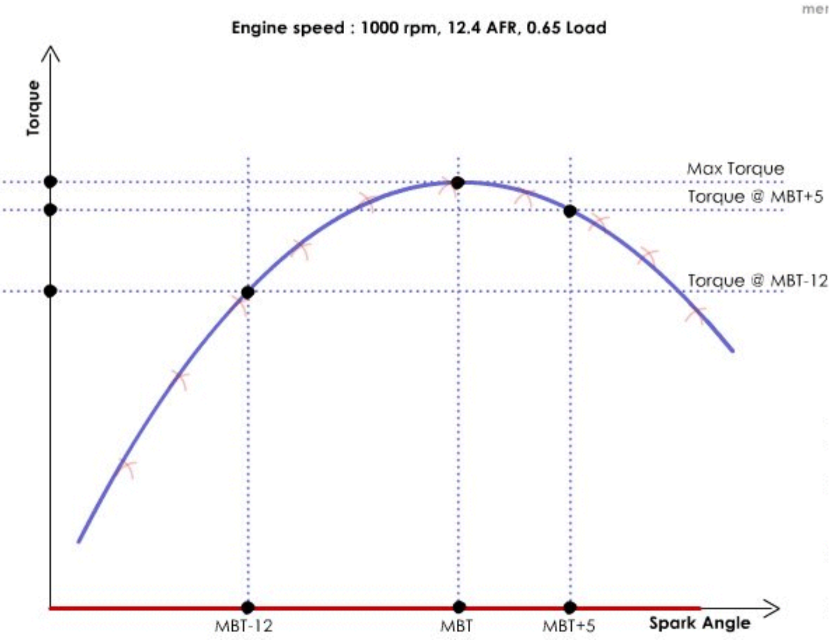
Note Expression	Test Note	Results
✓ mean(BTQ) < 0	Negative torque	Note successfully applied : 34 tests marked

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

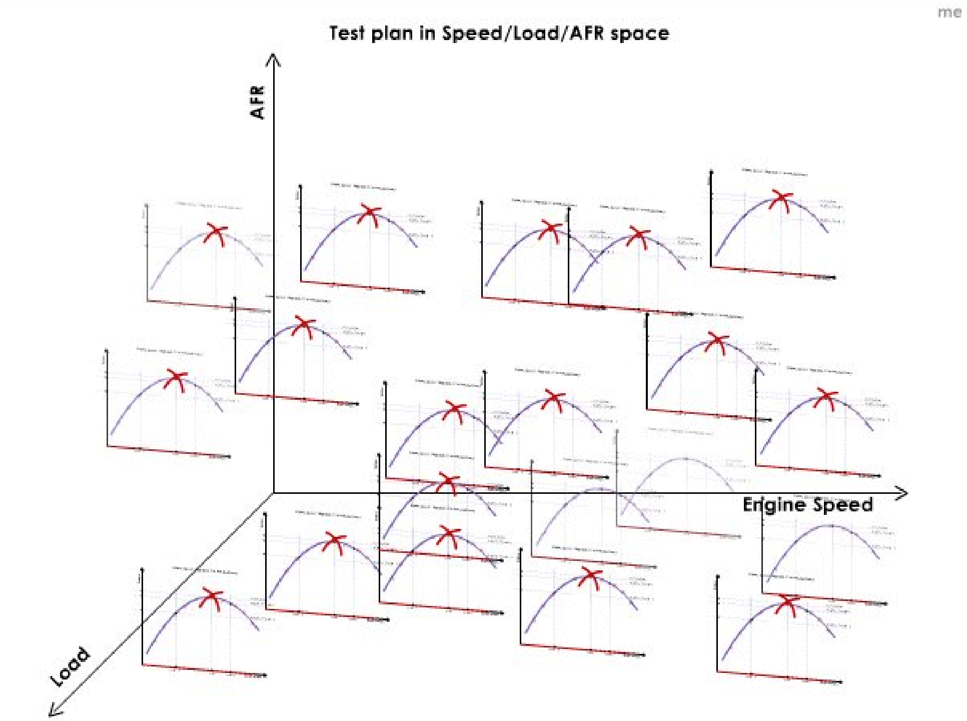
3 When you are satisfied with the data, close the Data Editor and return to the Model Browser.

How Is a Two-Stage Model Constructed?

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

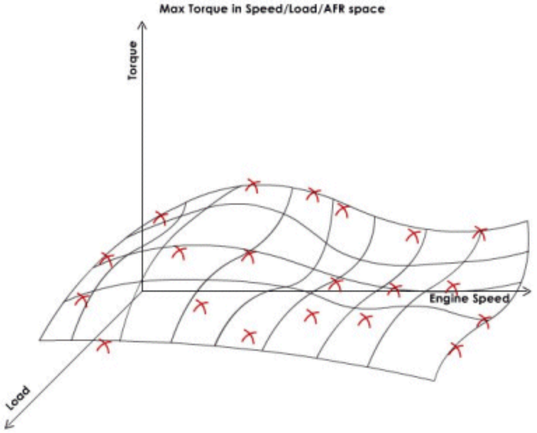


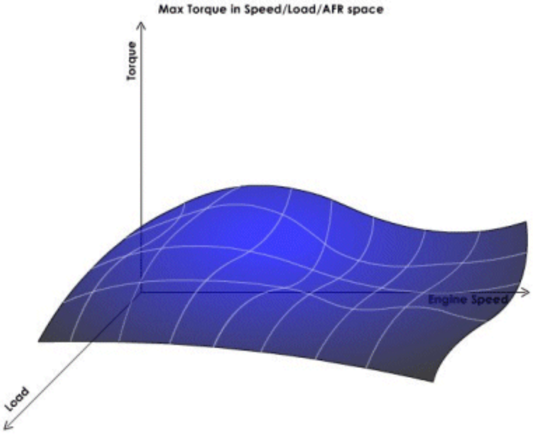
The local models provide the coefficients to generate global models. The equations describing those local model curves have certain coefficients such as max and knot, which for this data are peak torque and MBT spark (the spark angle that generates maximum brake torque).



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

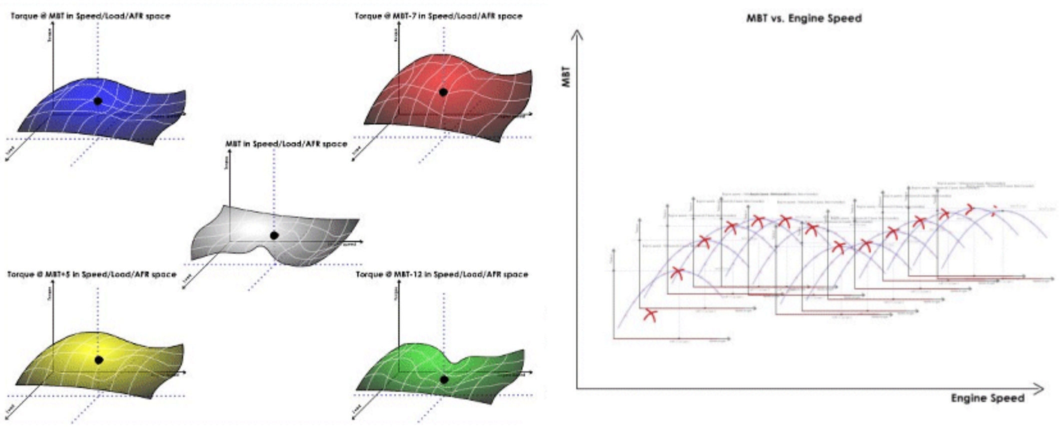
Global models are the best fit of a curve to the values of, for example, MBT for each test. This is repeated for each coefficient, producing several global models fitted to different coefficients of the local models. These coefficients are referred to as response features of the local models. The following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





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

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


Torque @ MBT in Speed/Load/AFR space
Torque @ MBT-7 in Speed/Load/AFR space
MBT in Speed/Load/AFR space
Torque @ MBT+5 in Speed/Load/AFR space
Torque @ MBT-12 in Speed/Load/AFR space
MBT vs. Engine Speed
MBT
Engine Speed

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

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

For more details on two-stage modeling, see “Two-Stage Models” on page 5-2, and see “Two-Stage Models for Engines” in the Model Browser documentation for more statistical depth. (In the Help Browser you can right-click and select **Back** to return to previous pages).

Building the Models

You have already set up the local and global inputs and the global model type before constructing the design. Now you have imported, examined, and filtered the data in preparation for modeling. To complete your model setup, you need to specify the local model type, select the data to model, and choose the responses (model outputs) you want to model.

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

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

  — as functions of load and rpm, subject to constraints of exhaust temperature.

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

- Torque
- Exhaust Temperature

Specifying the Local Model Type

The first response you want to model is torque against spark. The shape of torque/spark curves is well understood and you have examined some in the Data Editor. Polynomial spline curves are useful for fitting these shapes, where different curvature is required above and below the maximum. Therefore, you should set the local model type to polynomial spline. A spline is a curve made up of pieces of polynomial, joined smoothly together. The points of the joins are called knots. In this case, there is only one knot, at the maximum. The location of the knot in this case marks MBT.

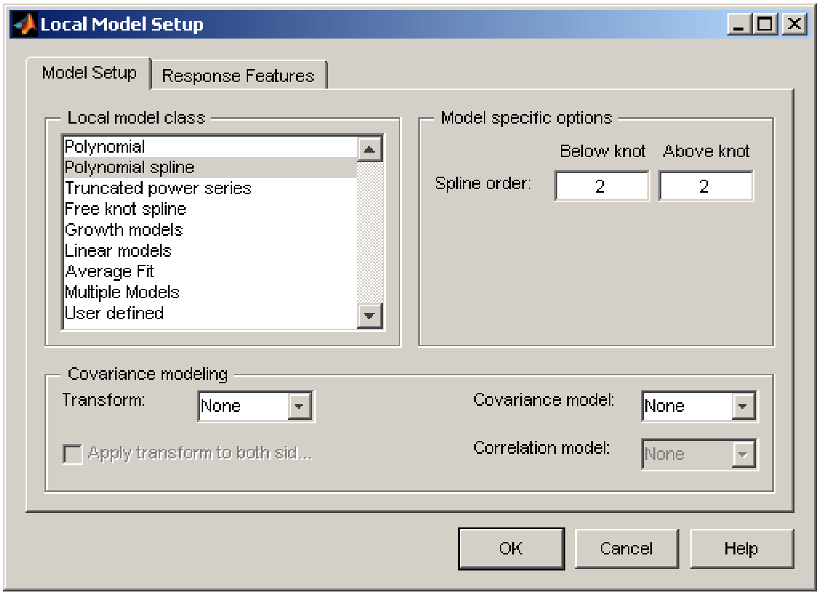
To specify polyspline as the local model type, 1 First select the `Two-Stage` test plan node in the model tree, so you can see the test plan diagram.

2 Double-click the local model icon in the test plan diagram.

   The Local Model Setup dialog box appears.

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

   b Set **Spline Order** to `2` below and `2` above knot.



Spline order: 2 2

Transform: None

Covariance model: None

Correlation model: None

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

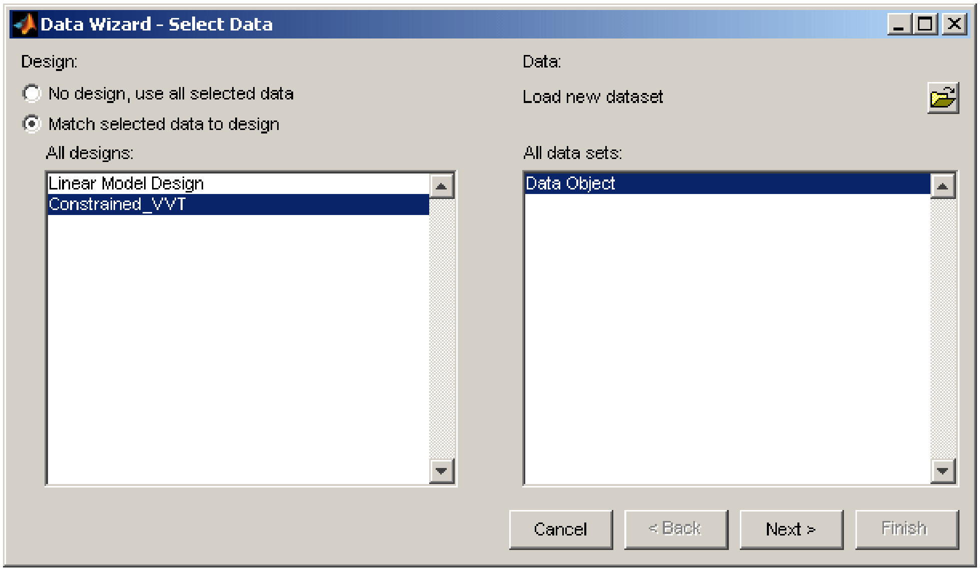
Notice that the new name of the local model class, `PS` (for polyspline) `2,2` (for spline order above and below the knot) now appears on the two-stage model diagram.

Selecting Data and Responses to Model

You have set up model types and model inputs. Now you can select the data for modeling and the responses (model outputs) you want to model.

1 Double-click the Response block in the test plan diagram. The Data Wizard appears.

2 Because the test plan contains a design, the radio button **Match selected data to design** is selected. The `Constrained_VVT` design appears in the left list, and the `VVT` data in the right list.



Data Wizard - Select Data

Design:

○ No design, use all selected data

● Match selected data to design

All designs:

Linear Model Design

Constrained_VVT

Data:

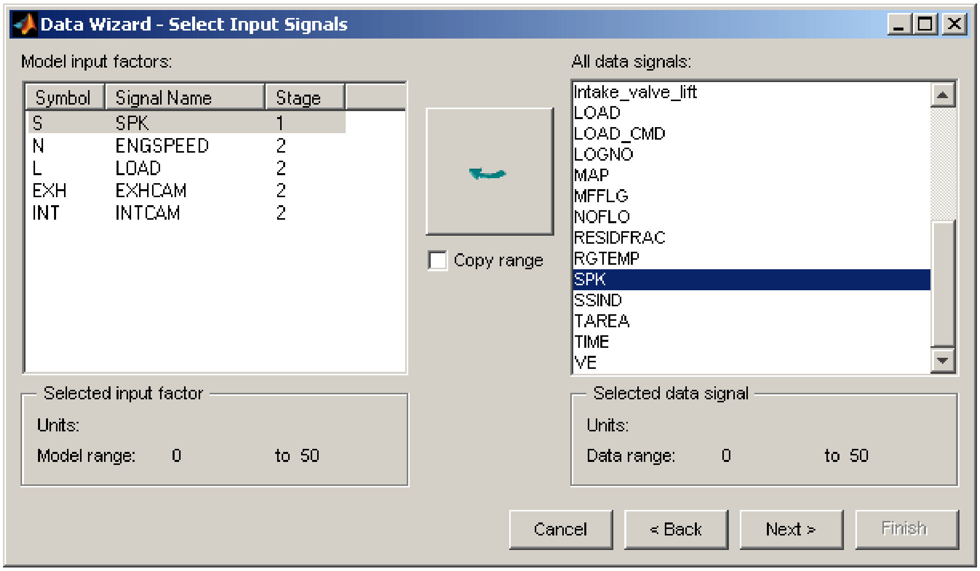
Load new dataset

All data sets:

Data Object

Click **Next**.

3 The wizard tried to match each symbol to a data signal (shown as **Signal Name**). If any are incorrect, select the corresponding entries in the input and data lists and click the green arrow to select. Do not select the copy range check box.



Model input factors:

Symbol	Signal Name	Stage
S	SPK	1
N	ENGSPEED	2
L	LOAD	2
EXH	EXHCAM	2
INT	INTCAM	2

□ Copy range

All data signals:

Intake_valve_lift
LOAD
LOAD_CMD
LOGNO
MAP
MFFLG
NOFLO
RESIDFRAC
RGTEMP
SPK
SSIND
TAREA
TIME
VE

Selected input factor

Units:

Model range: 0 to 50

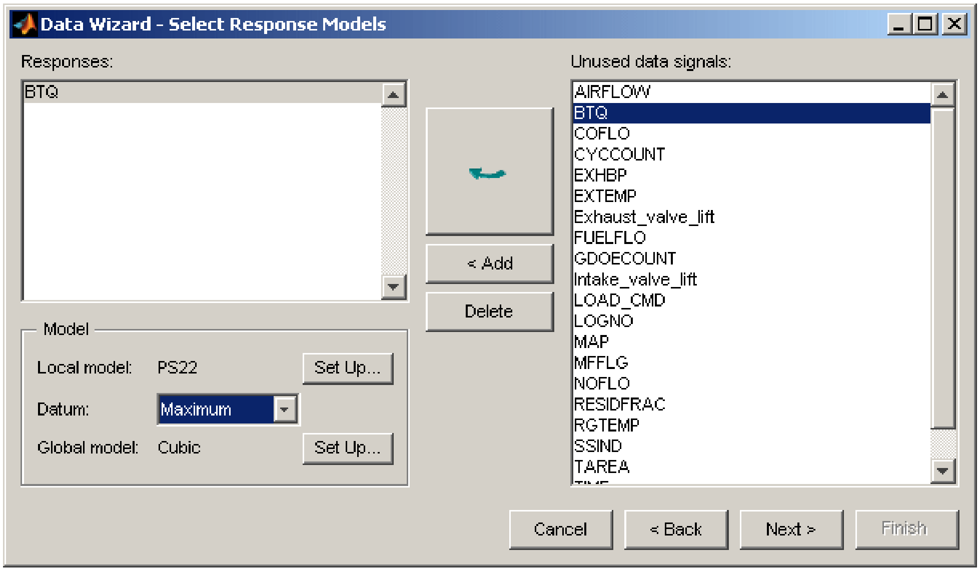
Selected data signal

Units:

Data range: 0 to 50

Click **Next**.

4 On this screen you select the response to model.



Responses:

BTQ

Model

Local model: PS22

Datum: Maximum

Global model: Cubic

Unused data signals:

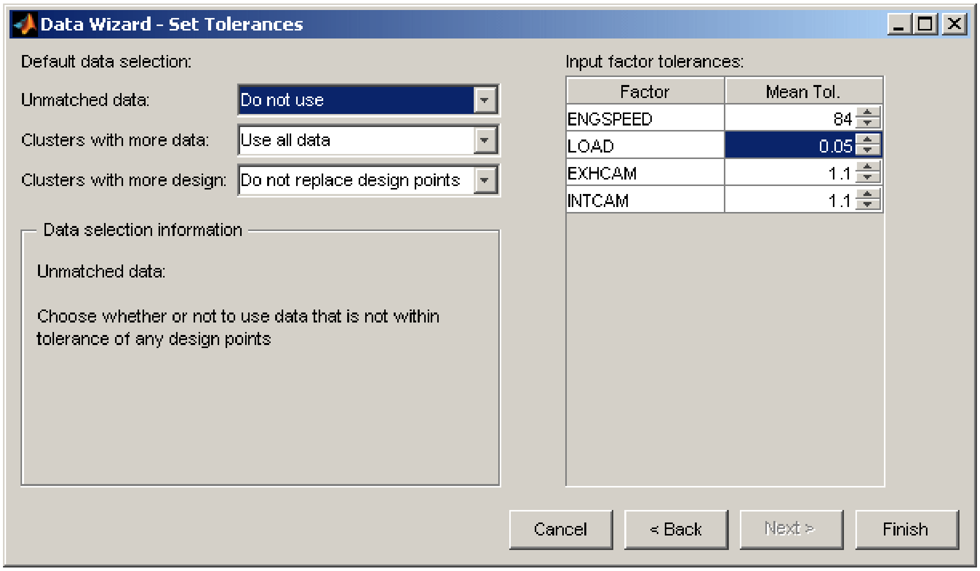
AIRFLOW
BTQ
COFLO
CYCCOUNT
EXHBP
EXTEMP
Exhaust_valve_lift
FUELFLO
GDOECOUNT
Intake_valve_lift
LOAD_CMD
LOGNO
MAP
MFFLG
NOFLO
RESIDFRAC
RGTEMP
SSIND
TAREA

a Select BTQ as the response you want to model and click **Add**. Notice the local model settings you set earlier (PS22).

b Select `Maximum` from the **Datum** drop-down menu. In this case, the maximum of the torque/spark curves is MBT (spark angle at maximum brake torque), so this can be a useful feature to model.

c Click **Next**.

5 On this screen, you see the settings for matching data to designs.



Data Wizard - Set Tolerances

Default data selection:

Unmatched data: Do not use

Clusters with more data: Use all data

Clusters with more design: Do not replace design points

Data selection information

Unmatched data:

Choose whether or not to use data that is not within tolerance of any design points

Input factor tolerances:

Factor	Mean Tol.
ENGSPEED	84
LOAD	0.05
EXHCAM	1.1
INTCAM	1.1

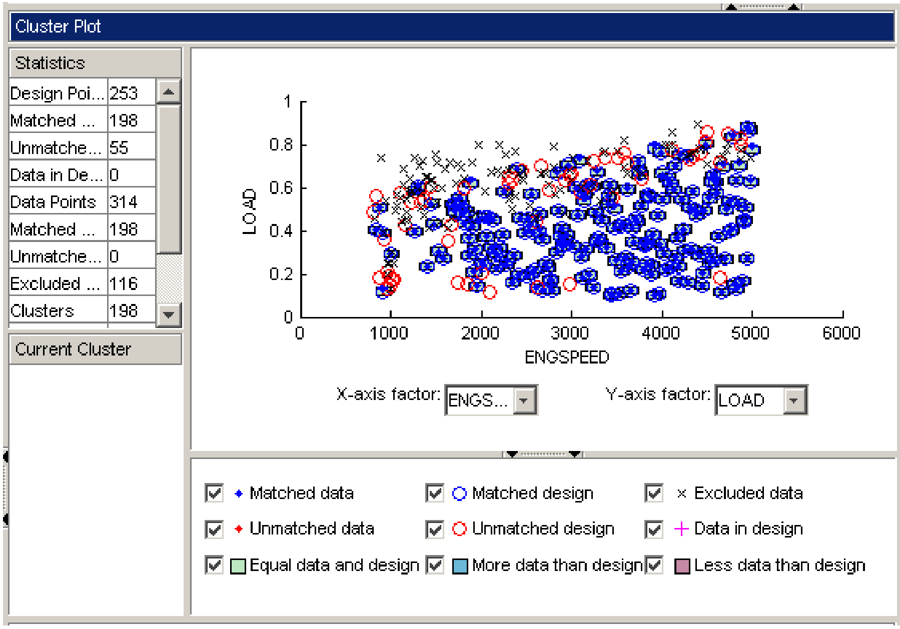
a Enter `0.05` in the LOAD tolerance edit box.

b Select `Do Not Use` from the drop-down menu for **Unmatched Data**. Data points that do not fall within tolerance of the design points will not be used for modeling.

c Leave the other settings at the defaults and click **Finish**.

The Data Editor appears so you can select data for modeling. The response model is built when you close the Data Editor.

6 Right-click a view and select **Current View > Cluster View**. Use the drop-down menus to select `ENGSPEED` and `LOAD` for plotting.



Design Poi...	253
Matched ...	198
Unmatche...	55
Data in De...	0
Data Points	314
Matched ...	198
Unmatche...	0
Excluded ...	116
Clusters	198

X-axis factor: ENGS...

Y-axis factor: LOAD

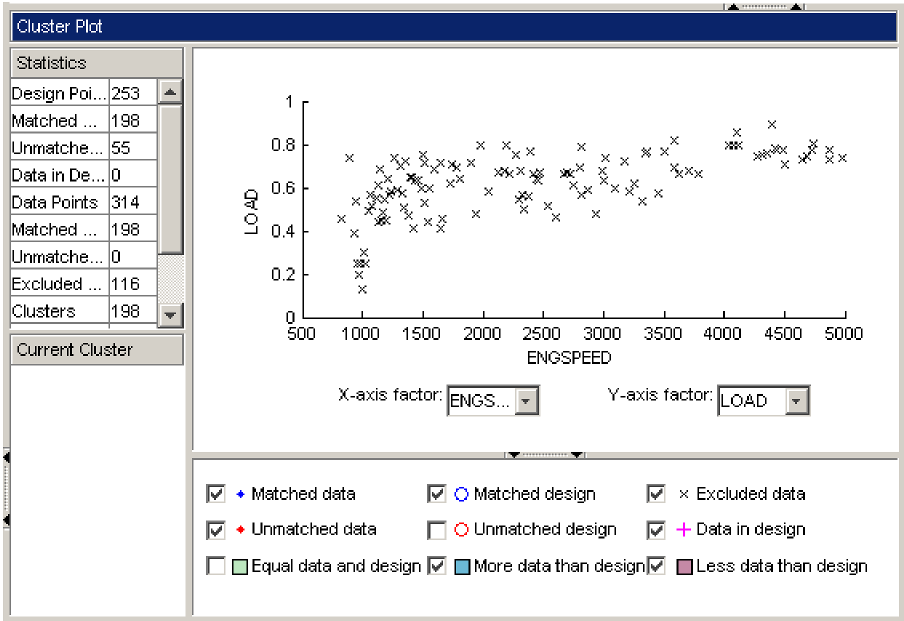
☑ ● Matched data ☑ ○ Matched design ☑ × Excluded data

☑ ● Unmatched data ☑ ○ Unmatched design ☑ + Data in design

☑ 🟩 Equal data and design ☑ 🟦 More data than design ☑ 🟪 Less data than design

7 Inspect the data and design points by using the check boxes in the cluster plot. Clear the check box for `Equal data and design` to remove matched (green) clusters from the display. These data points fall within tolerance of the design point, so these points are selected for modeling. When you remove them from the display, you can see other points more clearly.

8 Clear the box for **Unmatched design** points, and look at the remaining excluded data points (plotted as crosses).



Cluster Plot

Statistics

Design Poi...	253
Matched ...	198
Unmatche...	55
Data in De...	0
Data Points	314
Matched ...	198
Unmatche...	0
Excluded ...	116
Clusters	198

Current Cluster

X-axis factor: ENGS... Y-axis factor: LOAD

☑ ◆ Matched data ☑ ○ Matched design ☑ × Excluded data

☑ ◆ Unmatched data □ ○ Unmatched design ☑ + Data in design

□ 🟩 Equal data and design ☑ 🟦 More data than design ☑ 🟪 Less data than design

These points have not been matched to any design points because they do not lie within tolerance. The value of load achieved at these points was not close enough to the desired value of load, indicating a problem with these operating points. Notice that these points lie near the edge of the constrained area. In the Data Wizard, you selected `Do not use` unmatched points, so these points have not been selected for modeling. You can always change the tolerances and decide to include unmatched data points later; the choice in the Data Wizard is not irrevocable.

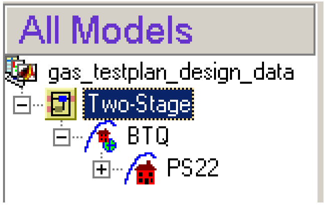
9 To accept the matched data for modeling, close the Data Editor. A dialog appears to check that you want to build the response model for torque and update the `Actual Design` to include all data selected for modeling. Click **Yes**, and the models are created.

Selecting Models

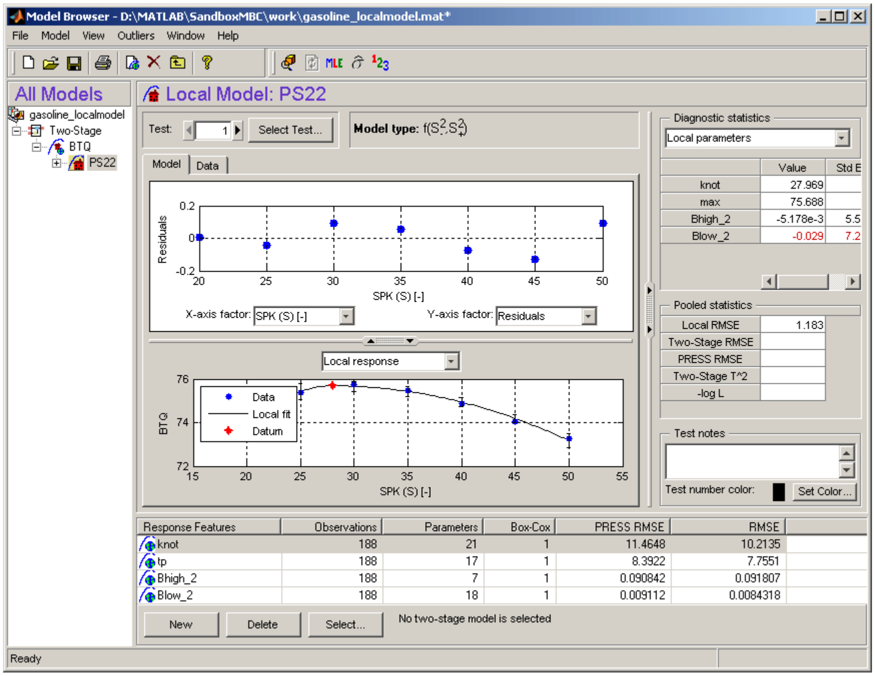
You should inspect the local and global models in turn, removing outliers if appropriate and trying different model types, before creating a two stage model. If the fit is good at the local and global levels you have the best chance of creating a two-stage model that accurately predicts the engine behavior.

Inspect the Local Models

1 Select the new local model node (`PS22`) in the model tree, in the **All Models** pane.



The local model view appears.



Model type: $f(S^2,S^2)$

X-axis factor: SPK (S) [-]

Y-axis factor: Residuals

Diagnostic statistics: Local parameters

	Value	Std E
knot	27.969	
max	75.688	
Bhigh_2	-5.178e-3	5.5
Blow_2	-0.029	7.2

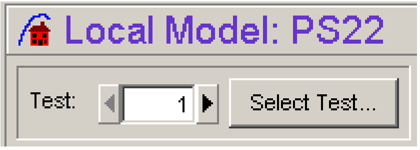
Pooled statistics

Local RMSE	1.183
Two-Stage RMSE	
PRESS RMSE	
Two-Stage T^2	
-log L	

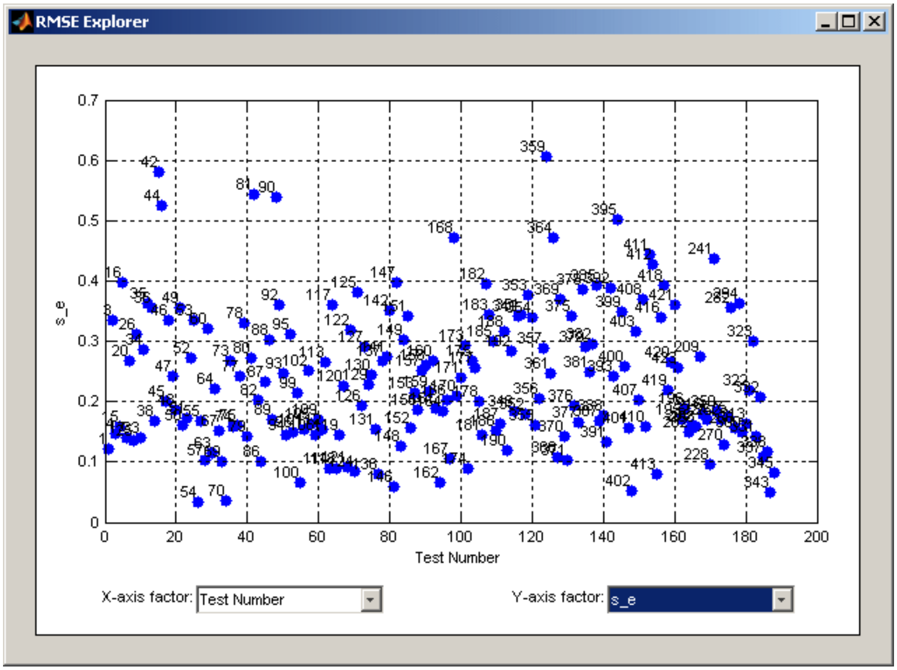
Response Features	Observations	Parameters	Box-Cox	PRESS RMSE	RMSE
knot	188	21	1	11.4648	10.2135
tp	188	17	1	8.3922	7.7551
Bhigh_2	188	7	1	0.090842	0.091807
Blow_2	188	18	1	0.009112	0.0084318

No two-stage model is selected

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



3 To quickly identify problem tests, click RMSE Plots in the toolbar (or **View** menu). Right-click to turn the test number display on, then inspect tests with high error values in the local model view.



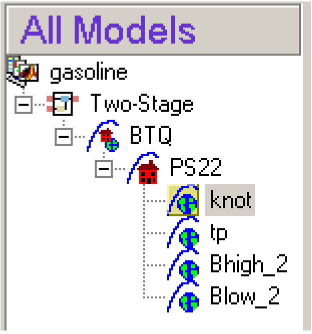
X-axis factor: Test Number

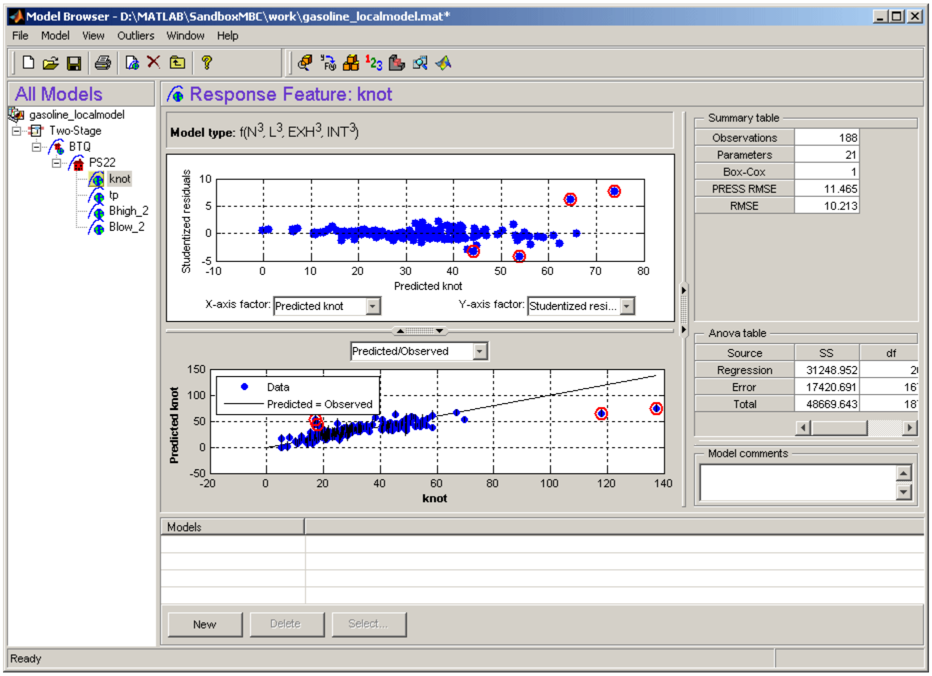
Y-axis factor: s_e

4 Consider removing outliers to improve fits if some points are badly distorting the torque spark curve. For example, for tests where the majority of points are at higher spark angles than the maximum (at MBT), it can improve the fit to remove some of these long “tails”. It can be useful to remove outliers in this region, because there is likely to be knock at spark values much higher than MBT where the engine is less stable.

Inspect the Global Models

1 When you are satisfied with the local fits, inspect the global models in turn. Expand the local model node (`PS22`) in the model tree and click `knot`.





Model type: $f(N^3, L^3, EXH^3, INT^3)$

X-axis factor: Predicted knot

Y-axis factor: Studentized resi...

Summary table

Observations	188
Parameters	21
Box-Cox	1
PRESS RMSE	11.465
RMSE	10.213

Anova table

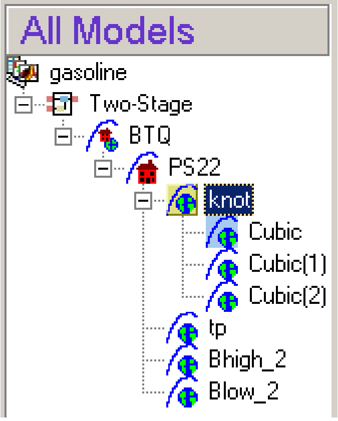
Source	SS	df
Regression	31248.952	2
Error	17420.691	16
Total	48669.643	18

Model comments

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

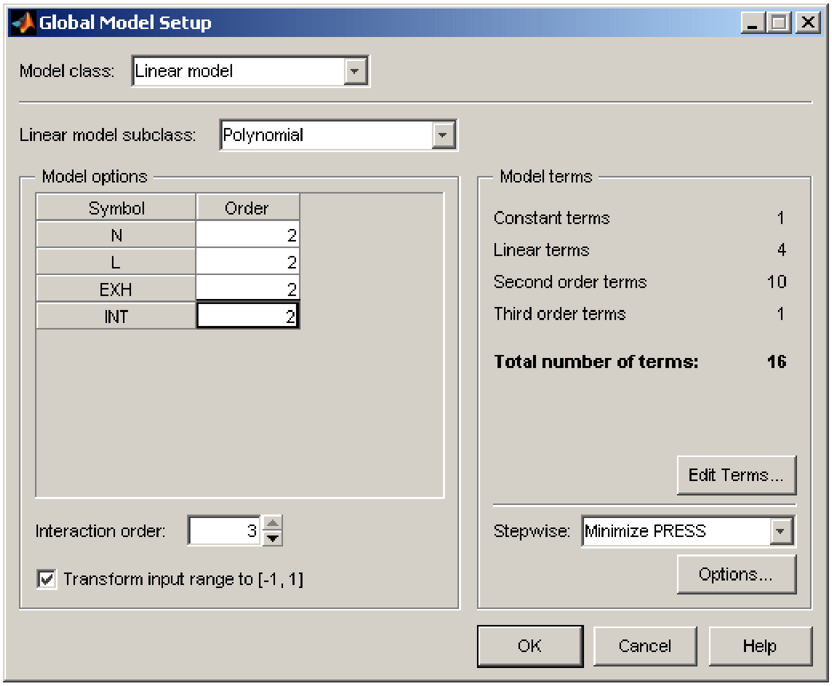
Create Multiple Models to Compare

1 Click the **New** button in the **Models** pane at the bottom. Click **New** again twice more, to create three child models of knot.



2 Select the second child model of knot in the tree (or double-click in the **Models** list at the bottom to select that model) and select **Model > Set Up**.

3 The Global Model Setup dialog appears. The model is currently a cubic because it is a copy of the parent model. Change the order for each factor to 2 and click **OK** to build a quadratic polynomial model.



Global Model Setup

Model class: Linear model

Linear model subclass: Polynomial

Model options

Symbol	Order
N	2
L	2
EXH	2
INT	2

Interaction order: 3

☑ Transform input range to [-1, 1]

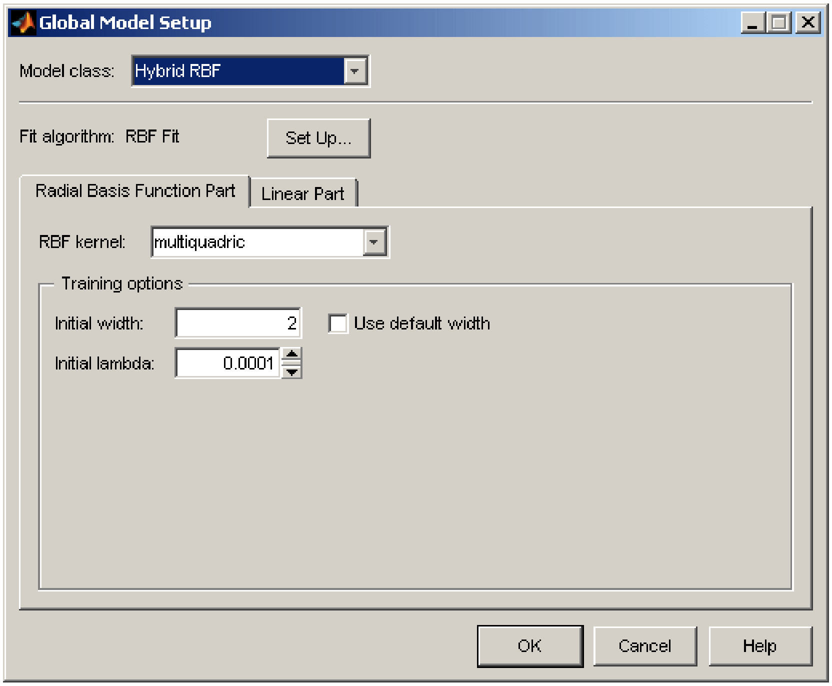
Model terms

Constant terms	1
Linear terms	4
Second order terms	10
Third order terms	1
Total number of terms:	**16**

Stepwise: Minimize PRESS

4 Select the third child model of `knot` in the tree and select **Model > Set Up**.

5 The Global Model Setup dialog appears. Select `Hybrid RBF` from the **Model class** drop-down menu. Leave the default settings and click **OK**.



Model class: Hybrid RBF

Fit algorithm: RBF Fit

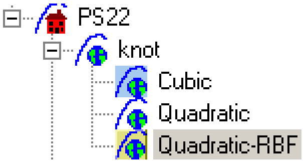
RBF kernel: multiquadric

Initial width: 2

□ Use default width

Initial lambda: 0.0001

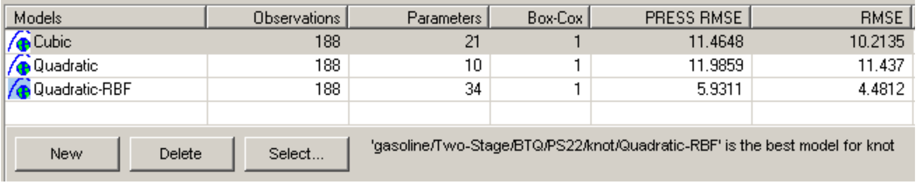
6 Now knot has three child models. Inspect each model in turn.



PS22
knot
Cubic
Quadratic
Quadratic-RBF

If you remove outliers that have an RBF center (marked with a star) be sure to refit the model (click the toolbar button Update Model Fit) to reselect widths and centers.

7 Return to the knot model node and look at the statistics reported in the list of child models at the bottom. From any parent model node you can see a list of statistical comparisons for all the child nodes in the lower list pane, along with information such as the number of parameters. Use this information to help you decide which model is best.



Models	Observations	Parameters	Box-Cox	PRESS RMSE	RMSE
Cubic	188	21	1	11.4648	10.2135
Quadratic	188	10	1	11.9859	11.437
Quadratic-RBF	188	34	1	5.9311	4.4812

'gasoline/Two-Stage/BTQ/PS22/knot/Quadratic-RBF' is the best model for knot

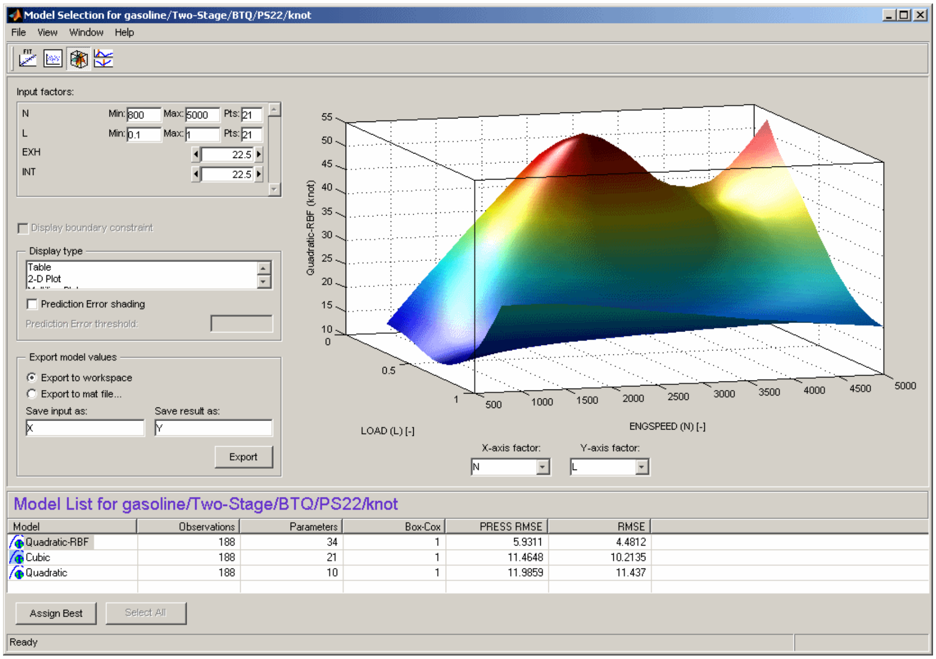
- Look for lower RMSE values to indicate better fits.
- Look for lower PRESS RMSE values to indicate better fits without overfitting.

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

  If the value of PRESS RMSE is much bigger than the RMSE then you are overfitting - the model is unnecessarily complex. As a rule of thumb, if you have about 100 data points, you should aim for a PRESS RMSE no more than 5% larger than the RMSE.

  PRESS RMSE can be the most helpful single statistic you can use to search for the best fit relative to the number of terms in the model. However you should not rely on any single statistic, but use a variety of criteria and especially the graphical tools available for comparison of models in the Model Evaluation tool when you click Select. You can also use other diagnostic statistics to help you select models. For detailed guidance on how to understand the selection tools see “Creating Multiple Models To Compare” and the “Model Selection Guide” in the Model Browser documentation. (In the Help Browser you can right-click and select **Back** to return to previous pages).

8 From the knot model node, click **Select** in the **Models** pane at the bottom to open the Model Selection window. Try the different views and compare the child models by selecting them in the **Model List** at the bottom.



Model Selection for gasoline/Two-Stage/BTQ/PS22/knot

Input factors:

N	Min: 800	Max: 5000	Pts: 21
L	Min: 0.1	Max: 1	Pts: 21
EXH		22.5	
INT		22.5	

□ Display boundary constraint

Display type: Table, 2-D Plot

□ Prediction Error shading

Prediction Error threshold:

Export model values

● Export to workspace

○ Export to mat file...

Save input as: X

Save result as: Y

X-axis factor: N

Y-axis factor: L

Model List for gasoline/Two-Stage/BTQ/PS22/knot

Model	Observations	Parameters	Box-Cox	PRESS RMSE	RMSE
Quadratic-RBF	188	34	1	5.9311	4.4812
Cubic	188	21	1	11.4648	10.2135
Quadratic	188	10	1	11.9859	11.437

Ready

9 Select one of the child models as best (click the button **Assign Best**) and close the Model Selection window to return to the Model Browser. Try other model types if you are not satisfied with the quality of the fit. You could work through the modeling tutorial for more guided examples of how to select models, see Chapter 5, “Tutorial: Model Quickstart ”.

10 From the parent model node (knot) select **Model > Make Template**. Save the template to a suitable directory.

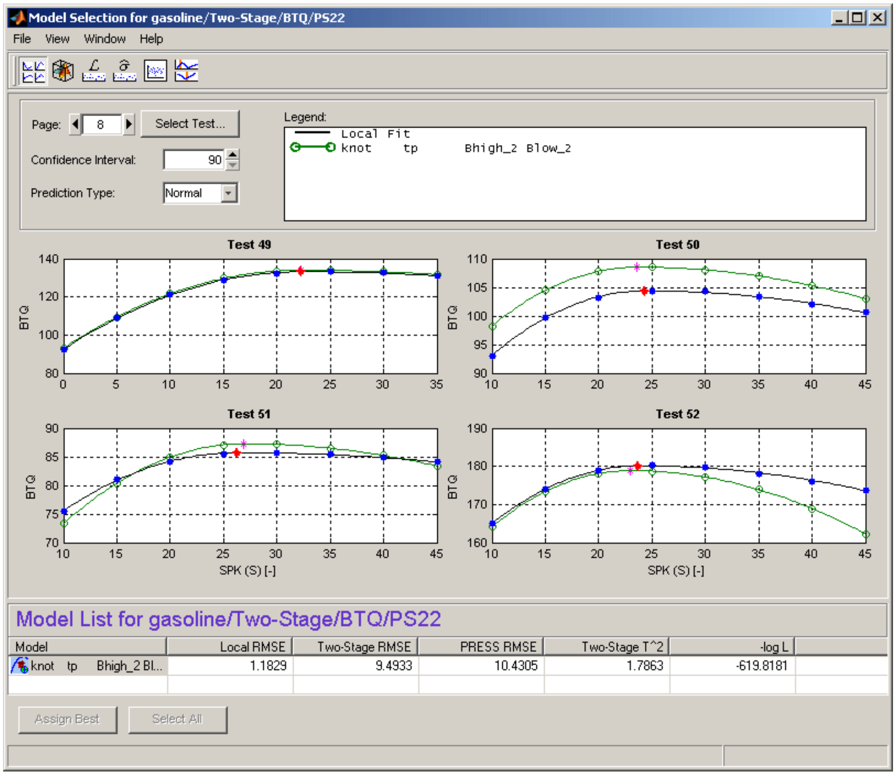
11 Select another global model such as Bhigh_2 and click Build Models 🧱 in the toolbar. Select your template to automatically build the same selection of child model types (cubic, quadratic, and hybrid-RBF) for Bhigh_2. This can help you quickly build multiple models to compare. When the child nodes are built the Model Selection dialog appears, where you can choose

a criterion such as `PRESS RMSE` for automatically selecting the best model out of the child nodes.

12 Repeat this process of searching for good global model fits for the other three global models.

Create a Two-Stage Model

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



Page: 8

Confidence Interval: 90

Prediction Type: Normal

Legend:

```
———    Local Fit
-○—○-  knot     tp       Bhigh_2 Blow_2
```

Model List for gasoline/Two-Stage/BTQ/PS22

Model	Local RMSE	Two-Stage RMSE	PRESS RMSE	Two-Stage T^2	-log L
knot tp Bhigh_2 Bl...	1.1829	9.4933	10.4305	1.7863	-619.8181

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

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

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

For more guidance on creating multiple local, global and two-stage models to search for the best fit, work through the step-by-step examples in Chapter 5, “Tutorial: Model Quickstart ”, especially the section “Creating Multiple Models to Compare” on page 5-39.

Adding a New Response Model

1 Select the test plan node in the model tree to return to the test plan view.

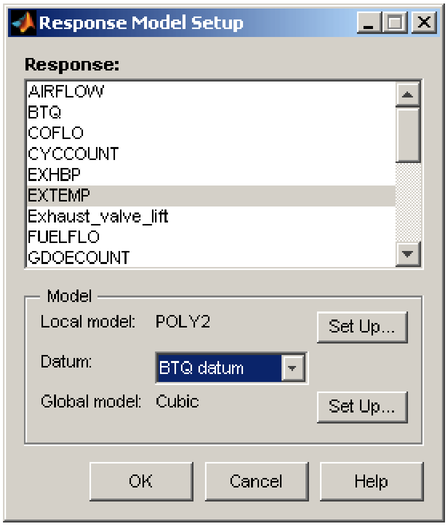
2 Double-click the **Responses** icon in the block diagram.

3 The Response Model Setup dialog appears.

  a Select `EXTEMP` from the list of signals.

  b Click the Local Model **Set Up** button. Select `Polynomial` from the **Model class** drop-down menu and click **OK**.

  c Select `BTQ datum` from the **Datum** drop-down menu. It can be useful to plot the position of MBT on other models.



Response Model Setup

Response:

AIRFLOW
BTQ
COFLO
CYCCOUNT
EXHBP
EXTEMP
Exhaust_valve_lift
FUELFLO
GDOECOUNT

Model

Local model: POLY2

Datum: BTQ datum

Global model: Cubic

d Click **OK**.

A new set of local and global models is calculated for the exhaust temperature response.

1. Expand the new model nodes in the tree and examine the fit in the same way as you did the torque fits.

   - Try different local models. Click **New** at the `EXTEMP` response node.

   - Try different global models as you did for the torque response features. Click **New** to add child nodes and try different model types.

   - Try a new response feature. Click **New** at one of the local nodes you have created and enter -10 for the value to use MBT minus 10 degrees of spark as a new response feature.

     See Chapter 5, "Tutorial: Model Quickstart " for simple worked examples of adding new local, global, response feature and two-stage models.

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

3 Try calculating MLE and return to the Model Selection window to compare the MLE model with the univariate model and select the best.

Look at the example finished project, `Gasoline_project.mat`, in the mbctraining directory, to see how the example models have been constructed.

Creating Boundary Models

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

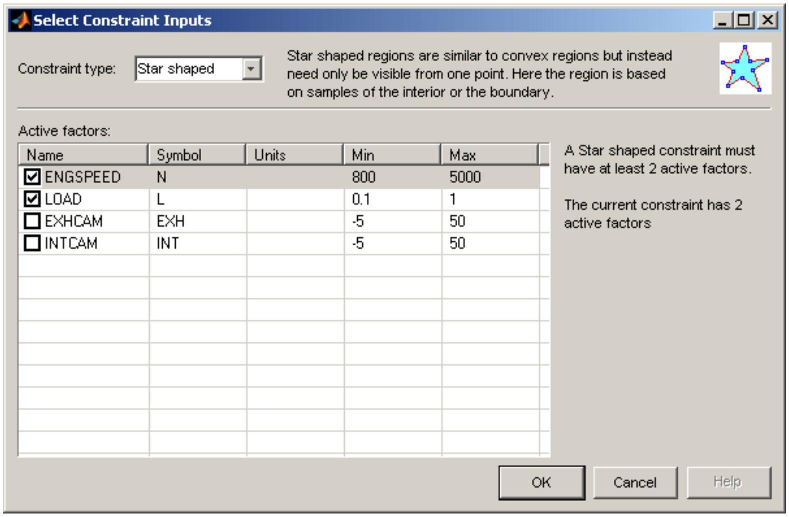
1 Select the test plan node in the model tree.

2 Select **TestPlan > Boundary Constraints**.

   The Boundary Constraint Editor opens.

3 Click Make Boundary Constraint in the toolbar. A dialog opens where you can choose to build a boundary model of the response, local, or global inputs. In this case, you are interested only in making a model of the global boundary. You are not interested in the `Local` boundary (you know the range of spark already). Select **Global** and click **OK**.

   A dialog opens where you can select constraint inputs.



Select Constraint Inputs

Constraint type: Star shaped

Star shaped regions are similar to convex regions but instead need only be visible from one point. Here the region is based on samples of the interior or the boundary.

Active factors:

Name	Symbol	Units	Min	Max
☑ ENGSPEED	N		800	5000
☑ LOAD	L		0.1	1
□ EXHCAM	EXH		-5	50
□ INTCAM	INT		-5	50

A Star shaped constraint must have at least 2 active factors.

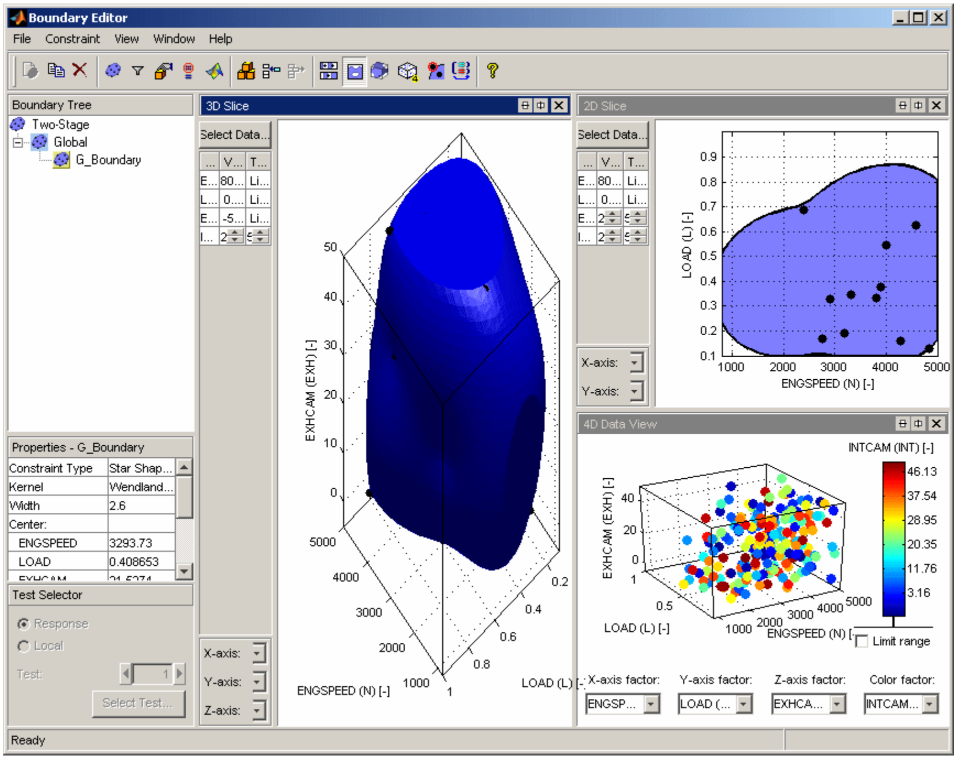
The current constraint has 2 active factors

4 Leave the **Constraint type** set to `Star shaped` and leave only the `ENGSPEED` and `LOAD` check boxes selected. Click **OK**.

5 A dialog appears where you can set up the star-shaped constraint parameters. Click **OK** to use the defaults and the boundary constraint is calculated.

A `G_Boundary` child node appears under the `Global` node.

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



Properties - G_Boundary

Constraint Type	Star Shap...
Kernel	Wendland...
Width	2.6
Center:	
ENGSPEED	3293.73
LOAD	0.408653

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

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

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

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

11 Click Make Boundary Constraint in the toolbar. You can build other boundary models to compare and combine for the most useful model.

In the dialog select **Global** and click **OK**.

12 This time select only `LOAD` and `INTCAM` as inputs and click **OK**. Click **OK** in the following dialog to accept the defaults and a new global boundary constraint is calculated.

13 Compare the two global boundary models by selecting the Pairwise view, and selecting each model in the tree in turn.

14 Select **Constraint > Assign Best** (also in the toolbar).

15 Select the other global model and select **Constraint > Add to Best** (also in the toolbar).

16 Now select the parent `Global` node to see the combination of the two constraints. Compare with both the child global nodes in the pairwise view.

17 Build another global boundary model using all four inputs at once, and compare it in the pairwise view with the speed and load model, the load and intake model, and the combination of the two. Select one of the models as best.

18 Close the Boundary Constraint Editor to return to the Model Browser. Once calculated, the boundary constraints remain part of the test plan unless you delete them.

Exporting the Models

Export the models so that they can be used in CAGE for optimizing calibration tables. You can:

- Export to CAGE
- "Export to File" on page 2-54

Export to CAGE

If you already have CAGE open you can export models directly to CAGE as follows:

1. At the MATLAB command prompt type `cage` to open the CAGE Browser part of the toolbox.
2. When CAGE is open, return to the Model Browser and select **File > Export Models** from the test plan node. Note that you cannot do this until you have calculated two-stage models for every response in the test plan.
3. Choose to export to `CAGE` from the top drop-down menu and click **OK**.
4. A dialog appears, listing the models to be exported to CAGE. You can double-click to edit the setting for any model: here you can edit the name for the new model, choose to replace an existing CAGE model, or skip the selected model. Click **OK** to export.
5. To see the models in CAGE, click the **Models** button in the **Data Objects** pane. To use these models to produce optimized calibration tables, proceed to the next section for instructions.

Export to File

1. Select the test plan node in the model tree. The models exported depend on the model selected in the tree, and in this case you want to export both response two-stage models and the boundary model from this test plan, so export from the test plan node.
2. Select **File > Export Models**. Note that you cannot do this until you have calculated two-stage models for every response in the test plan.

The Export Models dialog appears.

3 Choose to export to `File` from the top drop-down menu.

4 Select a destination and name for the file.

5 **Export constraints** is already selected because you have constructed a boundary constraint model. Leave this check box selected.

6 Select the check box to `Export datum models`. The datum model for the torque response is MBT, which can be useful.

7 You can click **Export Preview** to see a list of the models that will be exported. Click **OK** to dismiss the dialog.

8 Click **OK** in the Export Models dialog to export the models.

You now use these models in the CAGE part of the Model-Based Calibration Toolbox to produce optimized calibration tables.

Optimized Calibration

This section describes the creation and optimization of calibration tables for the gasoline case study. The Model Browser section of this case study covers creating the design for the experiment and creating and evaluating models from the resulting data. You can export your models directly to CAGE; or to Simulink 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
- Spark timing schedules

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

- Constrain exhaust temperature <=1200°C (to protect the catalyst)

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

1. Load models of engine responses, decide on optimization strategy and define additional models — “Importing Models into CAGE” on page 2-59.
2. Set up tables — “Setting Up Calibration Tables to Fill” on page 2-61.
3. Define optimization objective and constraints — “Setting Up the Optimization” on page 2-63.
4. Set up an operating point set for the optimization— “Defining Variable Values” on page 2-67.
5. Run the optimization and view the results — “Running the Optimization” on page 2-70

6. Fill tables from optimization results — “Filling Tables with Optimization Results” on page 2-72.

7. Use models and optimized tables to fill a spark estimator table — “MBT Spark Estimator Problem” on page 2-74.

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

Benefits of Automated Calibration

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

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

- Control problems
  - Injection timing and duration
  - EGR valve
  - Spark timing
  - Dual-independent variable valve timing
  - Emissions-constrained BSFC optimization over drive cycles
- Estimation problems
  - Torque
  - Emissions

- Air flow and manifold pressure
- Intake valve temperature
- Borderline spark

Importing Models into CAGE

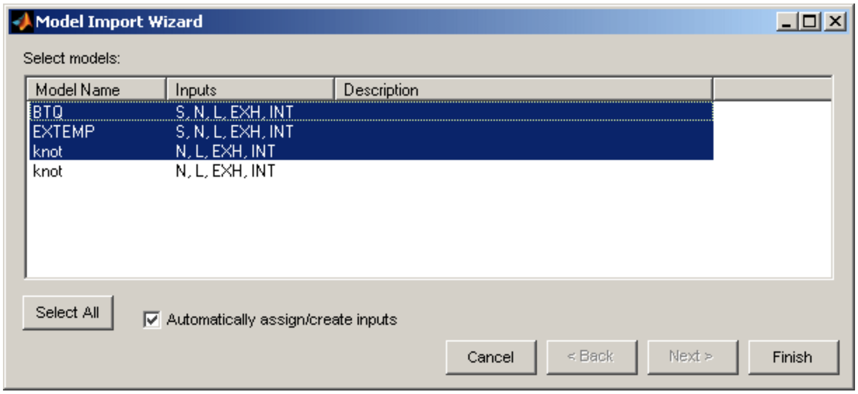
1. Start CAGE by typing `cage` at the MATLAB command line.

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 need only one copy of the `knot` model.



Select models:

Model Name	Inputs	Description
BTQ	S, N, L, EXH, INT	
EXTEMP	S, N, L, EXH, INT	
knot	N, L, EXH, INT	
knot	N, L, EXH, INT	

☑ Automatically assign/create inputs

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.

   Note you can also:

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

Setting Up Calibration Tables to Fill

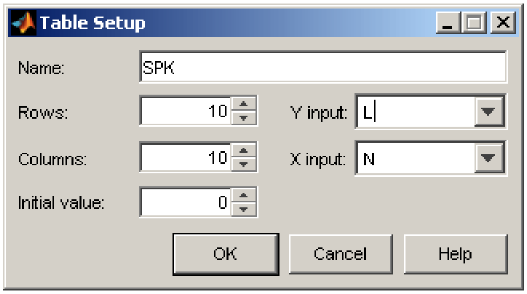
1 Set up a new 2-D table. Select **File > New > 2D Table.**

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.



Name: SPK

Rows: 10 Y input: L

Columns: 10 X input: N

Initial value: 0

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.

Setting Up the Optimization

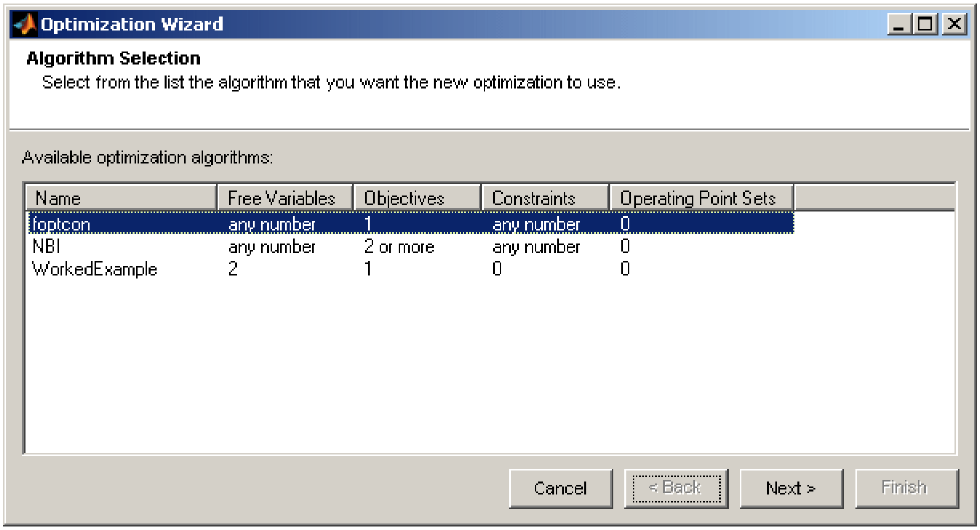
CAGE provides a flexible optimization environment. You can define the objectives, the constraints, and the points where the optimization is carried out.

The objective is to maximize 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 `foptcon` is selected by default, and 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**.



Algorithm Selection

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

Available optimization algorithms:

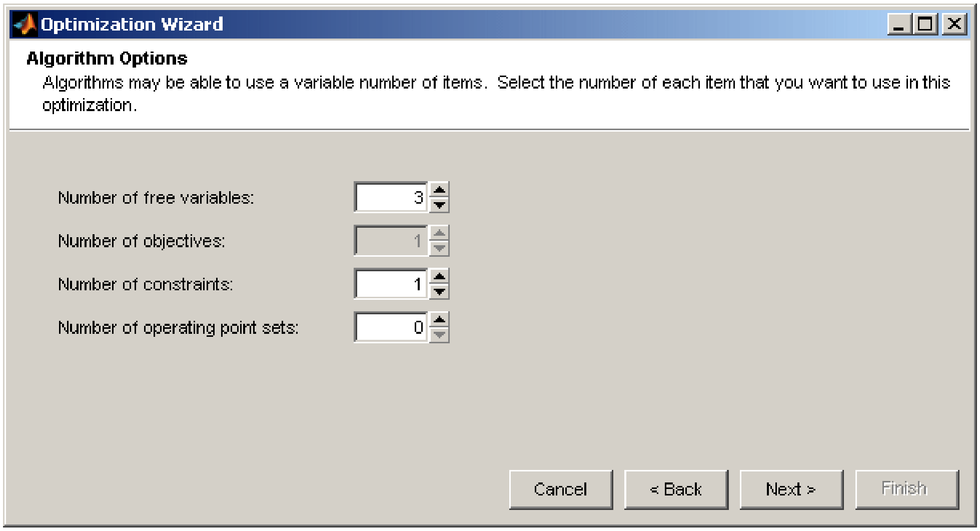
Name	Free Variables	Objectives	Constraints	Operating Point Sets
foptcon	any number	1	any number	0
NBI	any number	2 or more	any number	0
WorkedExample	2	1	0	0

3 Increase

a The number of free variables to 3

b The number of constraints to 1

Click **Next**.



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.

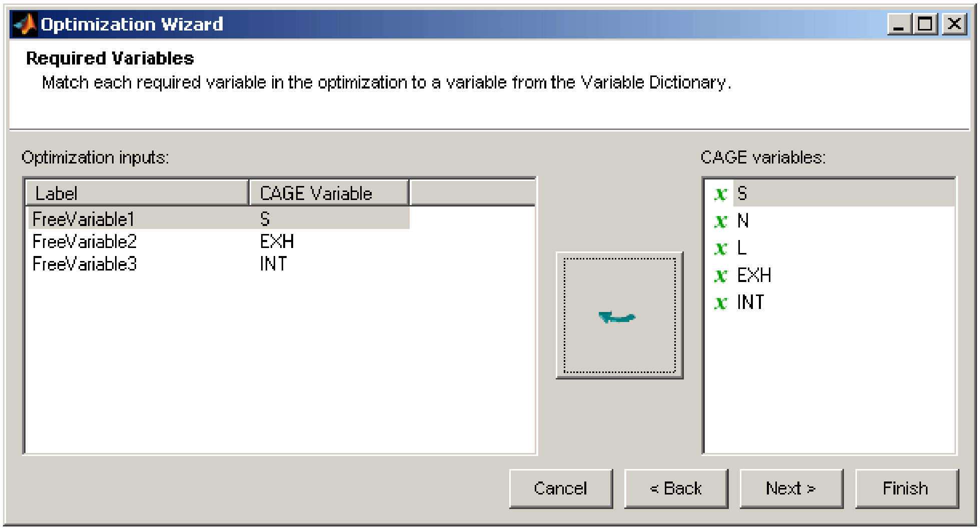
Number of free variables: 3

Number of objectives: 1

Number of constraints: 1

Number of operating point sets: 0

4 Select `S`, `EXH` and `INT` in turn for the free variables, click the button to select them, and click **Next**.



Optimization Wizard

Required Variables

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

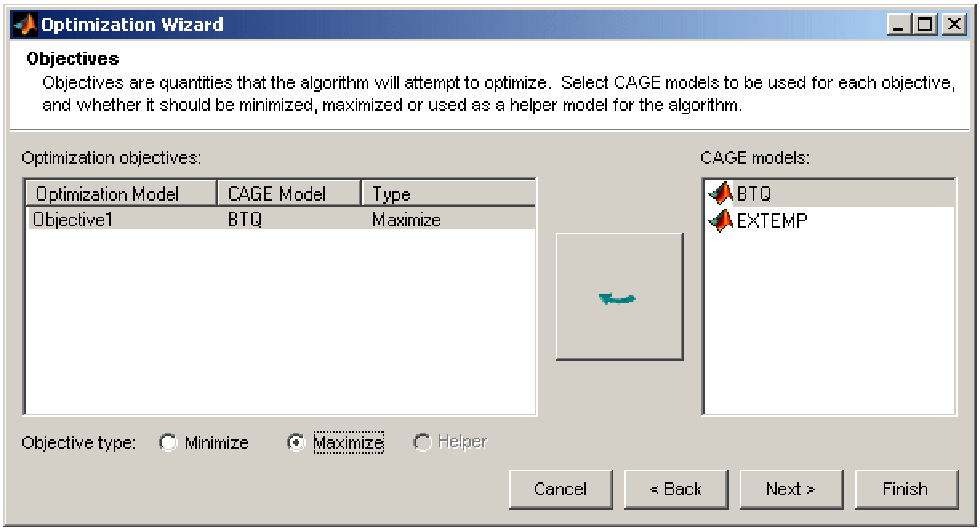
Optimization inputs:

Label	CAGE Variable
FreeVariable1	S
FreeVariable2	EXH
FreeVariable3	INT

CAGE variables:

- S
- N
- L
- EXH
- INT

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**.



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:

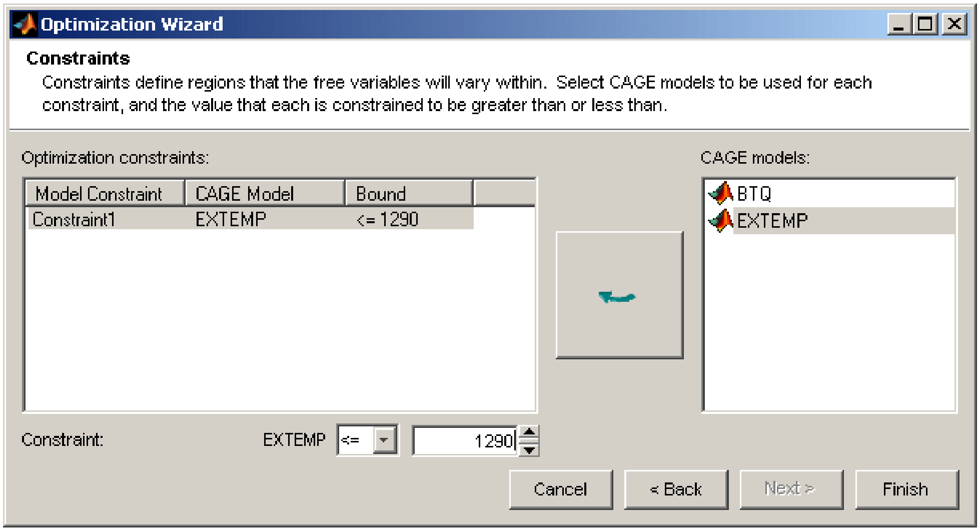
Optimization Model	CAGE Model	Type
Objective1	BTQ	Maximize

CAGE models:

- BTQ
- EXTEMP

Objective type: ○ Minimize ● Maximize ○ Helper

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 **Finish**.



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	EXTEMP	<= 1290

CAGE models:

- BTQ
- EXTEMP

Constraint: EXTEMP <= 1290

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

Defining Variable Values

You need to define the set of points where you want the optimization to run. To do this you use the **Free Variable Initial Values** and **Fixed Variable Values** panes in the Optimization view.

1 Increase the **Number of runs** to 15. Click the buttons or enter the value in the box. The number of rows in both variable value panes increases to 15. The default values are the set point of each variable. Leave the initial values for the free variables at the defaults.

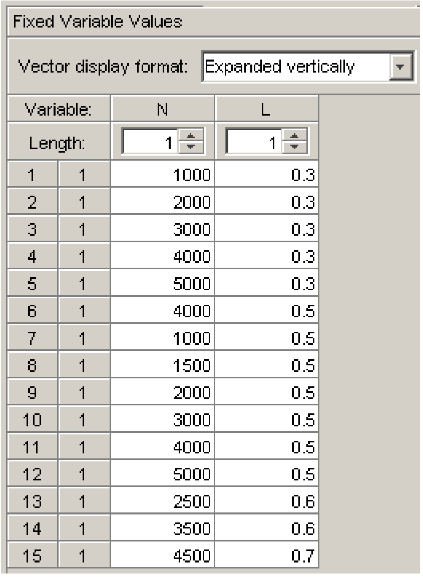
2 You can enter or copy values into the **Fixed Variable Values** pane to define the fixed variable values at each point where you want the optimization to run. You can copy all the variable values from a text file, or from the Help Browser copy each column in turn. Cut and paste the following points into the speed (N) column in the Fixed Points pane.

N
1000
2000
3000
4000
5000
4000
1000
1500
2000
3000
4000
5000
2500
3500
4500

3 Cut and paste the following points into the load (L) column in the Fixed Points pane.

L
0.3
0.3
0.3
0.3
0.3
0.5
0.5
0.5
0.5
0.5
0.5
0.5
0.6
0.6
0.7

The **Fixed Variable Values** pane should look as shown.



Fixed Variable Values

Vector display format: Expanded vertically

Variable:		N	L
Length:		1	1
1	1	1000	0.3
2	1	2000	0.3
3	1	3000	0.3
4	1	4000	0.3
5	1	5000	0.3
6	1	4000	0.5
7	1	1000	0.5
8	1	1500	0.5
9	1	2000	0.5
10	1	3000	0.5
11	1	4000	0.5
12	1	5000	0.5
13	1	2500	0.6
14	1	3500	0.6
15	1	4500	0.7

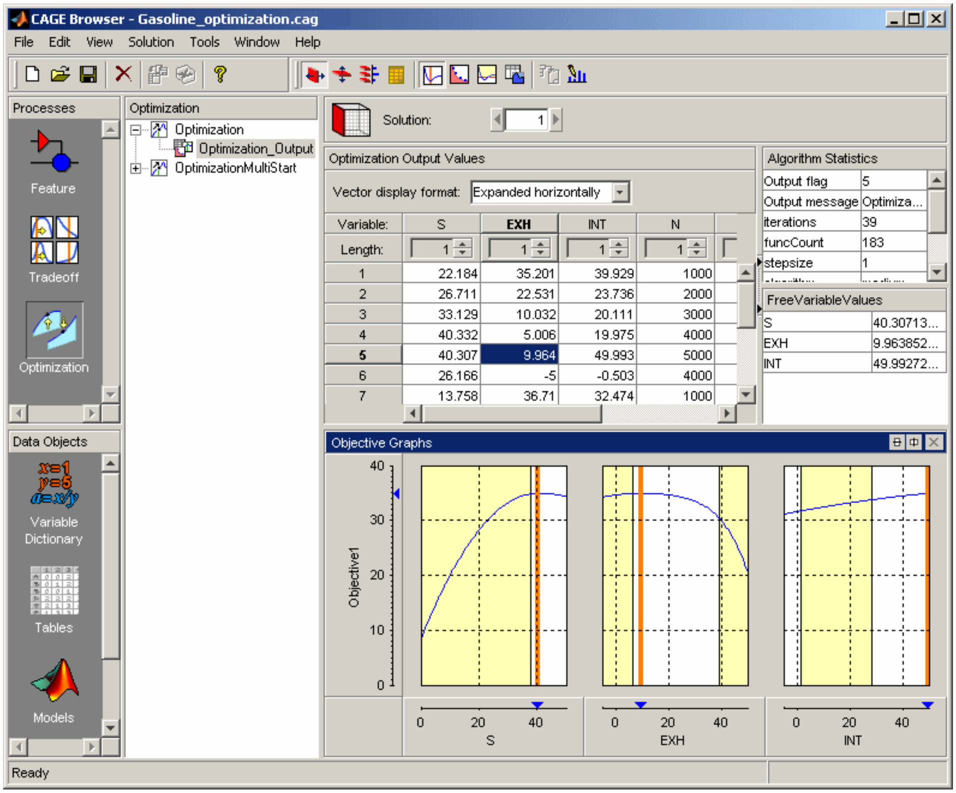
Running the Optimization

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

1 Click Run Optimization in the toolbar. The optimization runs.

2 When the optimization is complete, the view switches to the new child node, `Optimization_Output`, in the Optimization tree under the `Optimization` node. View the results.

3 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 objective 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.



Solution: 1

Optimization Output Values

Vector display format: Expanded horizontally

Variable:	S	EXH	INT	N
Length:	1	1	1	1
1	22.184	35.201	39.929	1000
2	26.711	22.531	23.736	2000
3	33.129	10.032	20.111	3000
4	40.332	5.006	19.975	4000
5	40.307	9.964	49.993	5000
6	26.166	-5	-0.503	4000
7	13.758	36.71	32.474	1000

Algorithm Statistics

Output flag	5
Output message	Optimiza...
iterations	39
funcCount	183
stepsize	1

FreeVariableValues

S	40.30713...
EXH	9.963852...
INT	49.99272...

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 **Shift**-click to multi-select the SPK, INTCAM, and EXHCAM tables and click the button to add the tables to the filling list. Click **Next**.

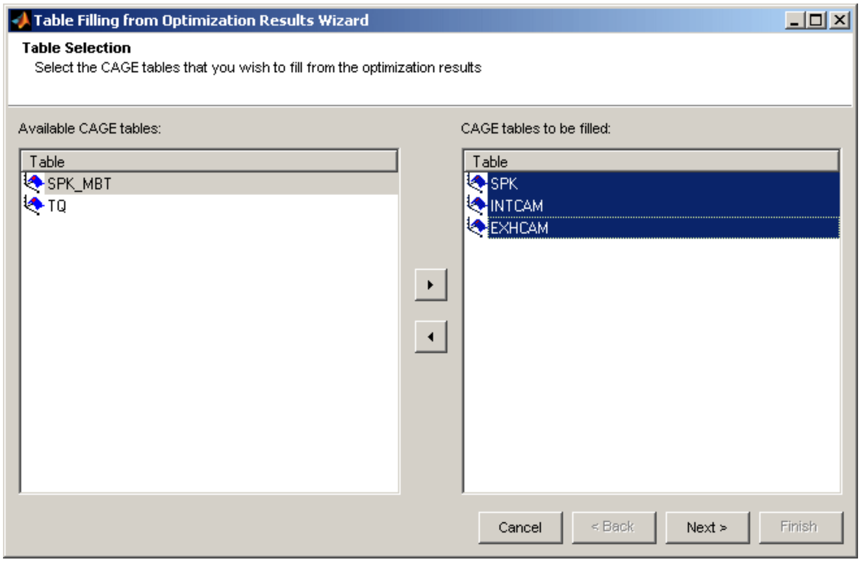


Table Filling from Optimization Results Wizard

Table Selection

Select the CAGE tables that you wish to fill from the optimization results

Available CAGE tables:

Table
SPK_MBT
TQ

CAGE tables to be filled:

Table
SPK
INTCAM
EXHCAM

3 You need to select factors for table filling.

a Select S from the right list of optimization results, select the SPK table on the left, and click the button to associate the two.

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

b Repeat to select INT and INTCAM.

c Repeat for EXH and EXHCAM.

d There is only one solution to fill the tables with, so you can click **Finish**.

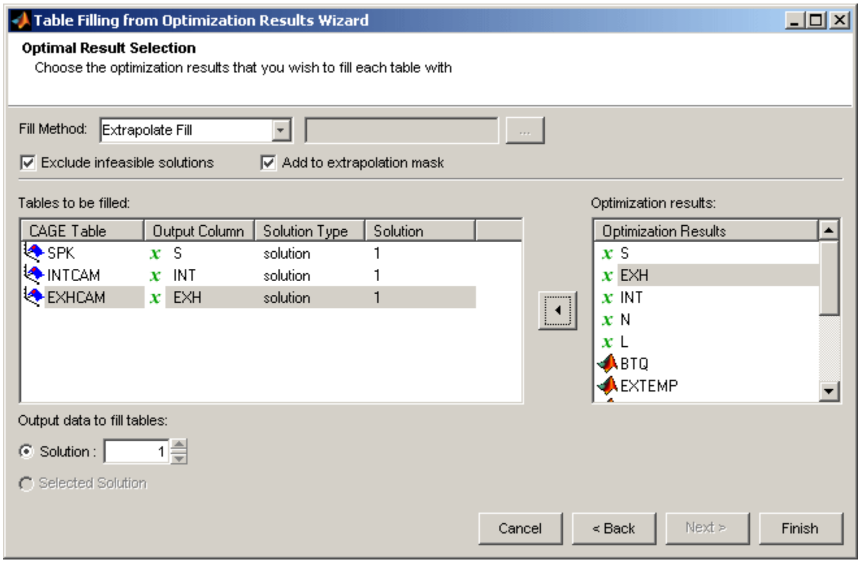


Table Filling from Optimization Results Wizard

Optimal Result Selection

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

Fill Method: Extrapolate Fill

☑ Exclude infeasible solutions ☑ Add to extrapolation mask

Tables to be filled:

CAGE Table	Output Column	Solution Type	Solution
SPK	*x* S	solution	1
INTCAM	*x* INT	solution	1
EXHCAM	*x* EXH	solution	1

Optimization results:

Optimization Results
x S
x EXH
x INT
x N
x L
BTQ
EXTEMP

Output data to fill tables:

● Solution : 1

○ Selected Solution

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

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

MBT Spark Estimator Problem

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

View the Feature

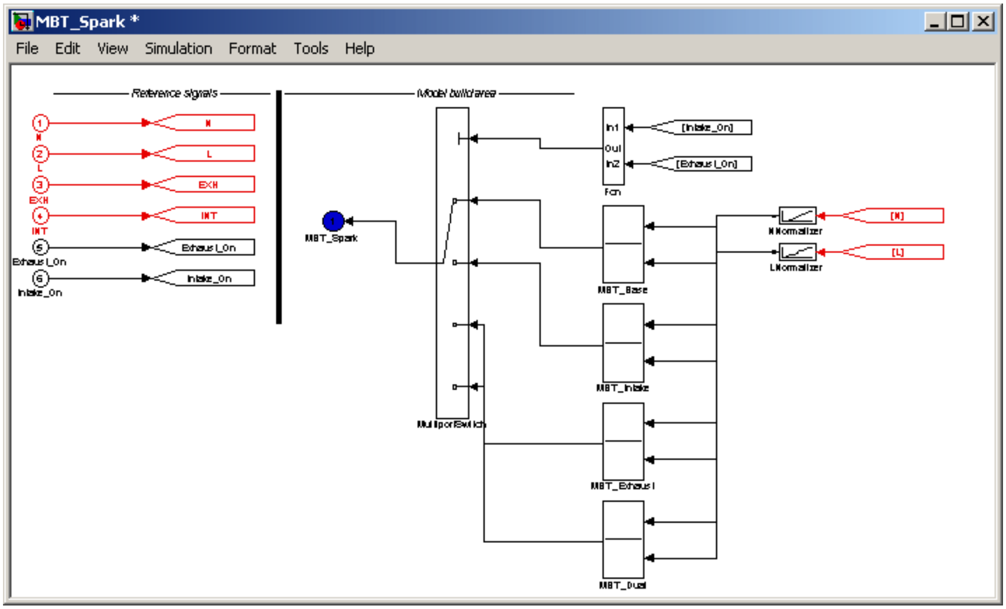
1. Select **File > Open Project** and load the example project, `Gasoline_optimization.cag` from the `mbctraining` directory.

2. Click **Feature** in the **Processes** pane to go to the Feature view.

   The features `Exhaust_CAM`, `Intake_CAM`, and `MBT_Spark` are in the Feature tree.

3. Select `MBT_Spark` and view the strategy by selecting **Feature > Graphical Strategy Editor**.

4. Examine the strategy model. Observe the Multiport Switch block which switches between MBT tables depending on whether each cam is on or off. The constants `Intake_On` and `Exhaust_On` are used to define whether cams are parked or not. All the MBT tables (with and without cams) share the same speed (`N`) and load (`L`) normalizers.



If you wanted to import this strategy model, you would double-click the `MBC_Spark` blue outport to parse the strategy into CAGE. This strategy has already been imported into this project so just close the model.

5 Expand the `MBT_Spark` feature in the tree to see the MBT tables that have been created by importing the strategy: `MBT_Base`, `MBT_Intake`, `MBT_Exhaust`, and `MBT_Dual`. These tables share the same normalizers in speed and load.

6 Similarly view the strategies for the Exhaust and Intake CAM features. These features switch between parked cams and active cams depending on the threshold value. The CAM features define the cam inputs and switch between the optimal CAM tables (from the optimization results) and the parked values. You can use the linking functionality in the Feature Filling Wizard to connect features and tables to models.

View Variables

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

View how this is done:

1 Click **Variable Dictionary** in the **Data Objects** pane to switch to the Variable Dictionary view.

2 Click to select the new variable `Exhaust_On`.

3 Observe the **Minimum** is 0 and the **Maximum** is 1.

4 The variable `Intake_On` has the same values.

Also two constants define the parked cam positions:

1 Select `Exhaust_Parked`.

2 Observe the **Set Point** of this constant is `40`.

3 Select `Intake_Parked`; this has a **Set Point** of `1`.

View Boundary Constraints

You can create a boundary model so you only fill over speed/load points where the data was collected. View how this is done: you can duplicate MBT, rename it and change it to a boundary model through the Model Properties dialog. You can use this to make a constraint function model.

1 Click **Models** to go to the models view.

2 Select `MBT_Boundary`.

3 Select **Model > Properties**. On the **General** tab, observe the selected radio button **Constraint boundary of model** for the **Output quantity**. Click **OK**.

4 You can use the `MBT_Boundary` model to create a constraint function that you can use to fill only speed/load points where the data was collected. Select the function model `MBT_Constraint`.

5 View the **Description** column or select **Model > Edit Function** to see the model is defined as `MBT_Boundary < 0`.

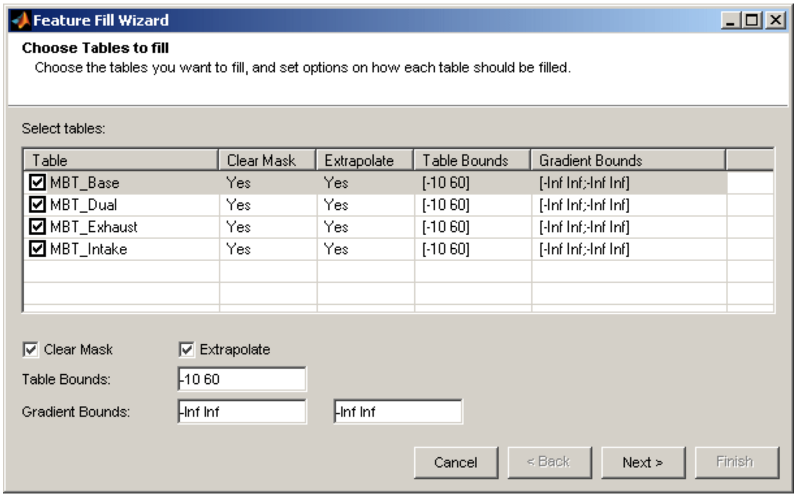
Use the Feature Fill Wizard

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

1 Click **Feature** in the **Processes** pane to return to the Feature view, then select the feature node `MBT_Spark`.

2 Click or select **Feature > Fill**. This opens the Feature Fill Wizard.

3 Select all four table check boxes to fill all tables. For each table in turn, enter `10 60` for the **Table Bounds** and press **Enter**.



Feature Fill Wizard

Choose Tables to fill

Choose the tables you want to fill, and set options on how each table should be filled.

Select tables:

Table	Clear Mask	Extrapolate	Table Bounds	Gradient Bounds
☑ MBT_Base	Yes	Yes	[-10 60]	[-Inf Inf;-Inf Inf]
☑ MBT_Dual	Yes	Yes	[-10 60]	[-Inf Inf;-Inf Inf]
☑ MBT_Exhaust	Yes	Yes	[-10 60]	[-Inf Inf;-Inf Inf]
☑ MBT_Intake	Yes	Yes	[-10 60]	[-Inf Inf;-Inf Inf]

☑ Clear Mask ☑ Extrapolate

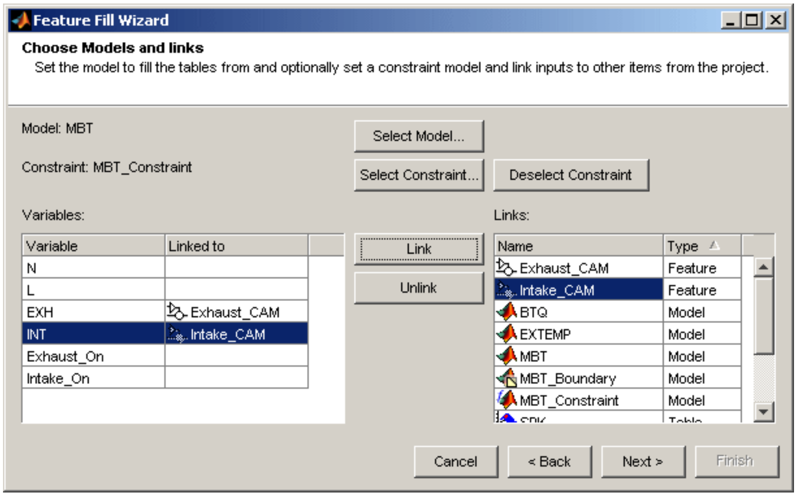
Table Bounds: -10 60

Gradient Bounds: -Inf Inf -Inf Inf

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

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

4 Choose filling model, constraint, and links.



Choose Models and links

Set the model to fill the tables from and optionally set a constraint model and link inputs to other items from the project.

Model: MBT

Constraint: MBT_Constraint

Variables:

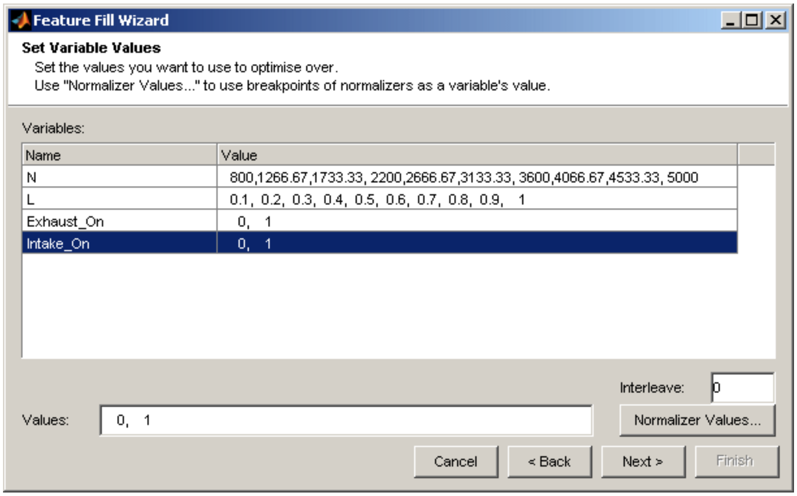
Variable	Linked to
N	
L	
EXH	Exhaust_CAM
INT	Intake_CAM
Exhaust_On	
Intake_On	

Links:

Name	Type
Exhaust_CAM	Feature
Intake_CAM	Feature
BTQ	Model
EXTEMP	Model
MBT	Model
MBT_Boundary	Model
MBT_Constraint	Model
SPK	Table

- Make sure `MBT` is the **Model** to fill the tables.
- Click **Select Constraint**, select `MBT_Constraint` in the dialog that opens, and click **OK**.
- Select `EXH` in the left **Variables** list and the `Exhaust_CAM` feature in the right **Links** list and click **Link**.
- Select `INT` in the left list and `Intake_CAM` in the right list and click **Link**. Click **Next**.

5 Set values to optimize over.



Feature Fill Wizard

Set Variable Values

Set the values you want to use to optimise over.
Use "Normalizer Values..." to use breakpoints of normalizers as a variable's value.

Variables:

Name	Value
N	800,1266.67,1733.33, 2200,2666.67,3133.33, 3600,4066.67,4533.33, 5000
L	0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1
Exhaust_On	0, 1
Intake_On	0, 1

Interleave: 0

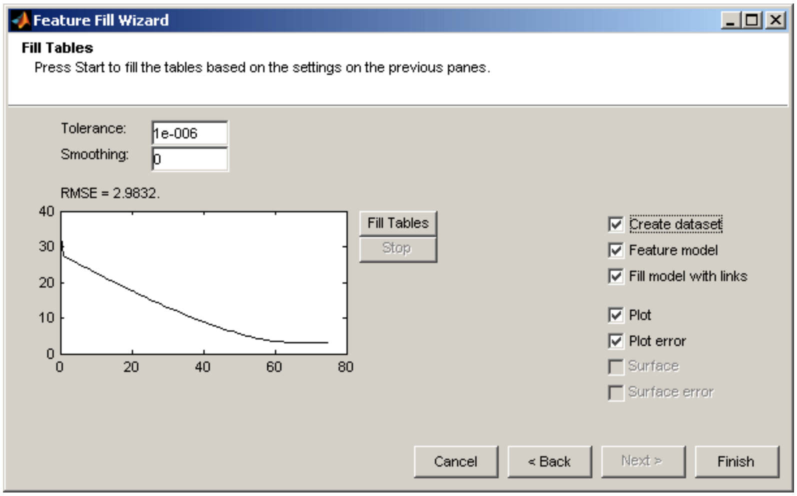
Values: 0, 1

- Select `Exhaust_On`, enter `0 1` in the **Values** edit box and press **Enter**.
- Select `Intake_On`, enter `0 1` in the **Values** edit box and press **Enter**.

The `N` and `L` normalizer values are automatically selected. You can use the **Interleave** setting here to minimize interpolation error by adding values between each normalizer value. In this way you can create a grid of more points than table cells to optimize over. Leave the setting alone for now.

Click **Next**.

6 Fill tables and generate plots.



Feature Fill Wizard

Fill Tables

Press Start to fill the tables based on the settings on the previous panes.

Tolerance: 1e-006

Smoothing: 0

RMSE = 2.9832.

☑ Create dataset
☑ Feature model
☑ Fill model with links
☑ Plot
☑ Plot error
□ Surface
□ Surface error

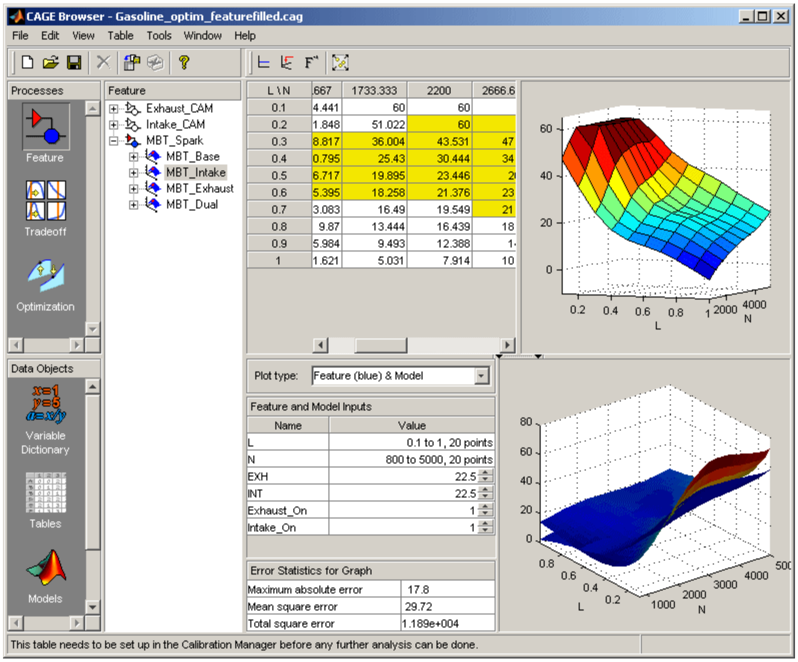
Click the **Fill Tables** button. Watch the graph as the optimization progresses.

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

Inspect Results

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

1 In the Feature view, select in turn the tables `MBT_Base`, `MBT_Intake`, `MBT_Exhaust`, and `MBT_Dual` in the feature tree.



L \ N	.667	1733.333	2200	2666.6
0.1	4.441	60	60	
0.2	1.848	51.022	60	
0.3	8.817	36.004	43.531	47
0.4	0.795	25.43	30.444	34
0.5	6.717	19.895	23.446	2
0.6	5.395	18.258	21.376	23
0.7	3.083	16.49	19.549	21
0.8	9.87	13.444	16.439	18
0.9	5.984	9.493	12.388	1
1	1.621	5.031	7.914	10

Plot type: Feature (blue) & Model

Feature and Model Inputs

Name	Value
L	0.1 to 1, 20 points
N	800 to 5000, 20 points
EXH	22.5
INT	22.5
Exhaust_On	1
Intake_On	1

Error Statistics for Graph

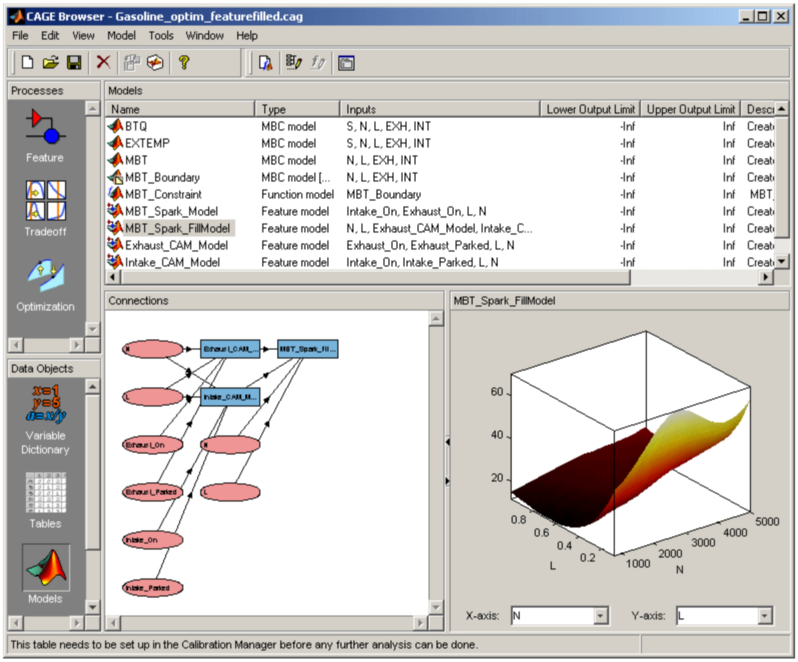
Maximum absolute error	17.8
Mean square error	29.72
Total square error	1.189e+004

This table needs to be set up in the Calibration Manager before any further analysis can be done.

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

The lower comparison pane (if you have it open) does not change as you change table, because it displays a comparison between the whole feature and the model, not individual tables. Note if you use links the comparison pane is not showing a true comparison as the cam inputs are not constant. The comparison pane is showing the comparison using constant values for the cam timings.

2 Click **Models** to select the Models view. Look at the feature model and fill model `MBT_SPARK_Model` and `MBT_SPARK_FillModel`. If you can't view the whole Connections diagram of `MBT_SPARK_FillModel`, right-click select **Zoom To Fit**.



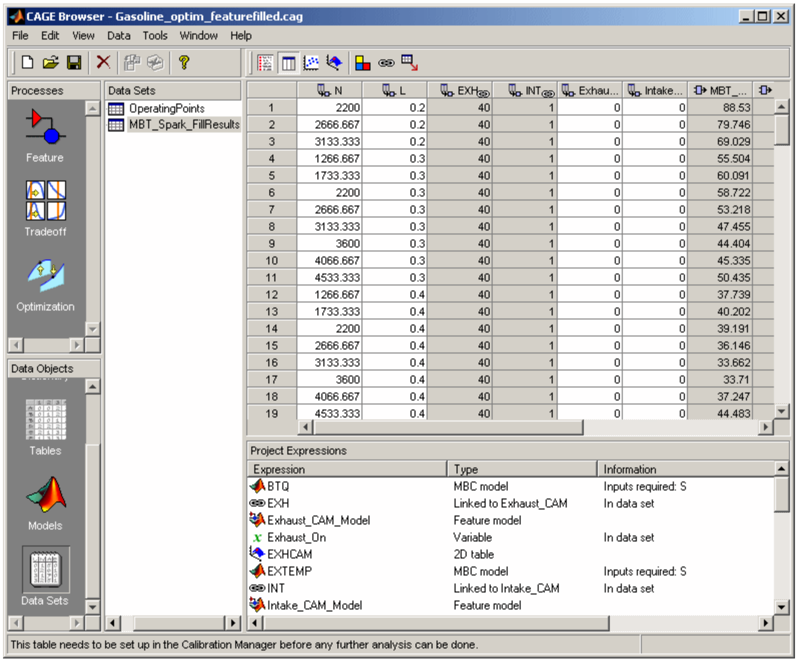
Name	Type	Inputs	Lower Output Limit	Upper Output Limit	Descr
BTQ	MBC model	S, N, L, EXH, INT	-Inf	Inf	Creat
EXTEMP	MBC model	S, N, L, EXH, INT	-Inf	Inf	Creat
MBT	MBC model	N, L, EXH, INT	-Inf	Inf	Creat
MBT_Boundary	MBC model [...	N, L, EXH, INT	-Inf	Inf	Creat
MBT_Constraint	Function model	MBT_Boundary	-Inf	Inf	MBT_
MBT_Spark_Model	Feature model	Intake_On, Exhaust_On, L, N	-Inf	Inf	Creat
MBT_Spark_FillModel	Feature model	N, L, Exhaust_CAM_Model, Intake_C...	-Inf	Inf	Creat
Exhaust_CAM_Model	Feature model	Exhaust_On, Exhaust_Parked, L, N	-Inf	Inf	Creat
Intake_CAM_Model	Feature model	Intake_On, Intake_Parked, L, N	-Inf	Inf	Creat

This table needs to be set up in the Calibration Manager before any further analysis can be done.

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

Notice that the CAM features are converted to feature models (`Exhaust_CAM_Model` and `Intake_CAM_Model` are new feature models) and they are connected to `MBT_SPARK_FillModel`.

3 Click **Data Sets** to select the Data Sets view. Select the dataset `MBT_SPARK_FillResults` to study the gridded data, the model and feature values for the feature fill. Click View Data ▥ in the toolbar to see the data table view. All the links in the feature fill process are defined in this dataset.



	N	L	EXH	INT	Exhau...	Intake...	MBT_...
1	2200	0.2	40	1	0	0	88.53
2	2666.667	0.2	40	1	0	0	79.746
3	3133.333	0.2	40	1	0	0	69.029
4	1266.667	0.3	40	1	0	0	55.504
5	1733.333	0.3	40	1	0	0	60.091
6	2200	0.3	40	1	0	0	58.722
7	2666.667	0.3	40	1	0	0	53.218
8	3133.333	0.3	40	1	0	0	47.455
9	3600	0.3	40	1	0	0	44.404
10	4066.667	0.3	40	1	0	0	45.335
11	4533.333	0.3	40	1	0	0	50.435
12	1266.667	0.4	40	1	0	0	37.739
13	1733.333	0.4	40	1	0	0	40.202
14	2200	0.4	40	1	0	0	39.191
15	2666.667	0.4	40	1	0	0	36.146
16	3133.333	0.4	40	1	0	0	33.662
17	3600	0.4	40	1	0	0	33.71
18	4066.667	0.4	40	1	0	0	37.247
19	4533.333	0.4	40	1	0	0	44.483

Project Expressions

Expression	Type	Information
BTQ	MBC model	Inputs required: S
EXH	Linked to Exhaust_CAM	In data set
Exhaust_CAM_Model	Feature model	
Exhaust_On	Variable	In data set
EXHCAM	2D table	
EXTEMP	MBC model	Inputs required: S
INT	Linked to Intake_CAM	In data set
Intake_CAM_Model	Feature model	

This table needs to be set up in the Calibration Manager before any further analysis can be done.

CAGE Import Tool

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

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

   The CAGE Import Tool appears.

2. You can choose a project file or import directly from the Model Browser if it is open. Use the toolbar buttons, or select **File > Select Project File**, or **File > Import From Model Browser**.

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

3 The CAGE Import Tool displays the available items. Select the items you want to import from the list: BTQ, MBT and EXTEMP.

4 Click the Import Selected Items toolbar button ( ) or select **File > Import Selected Items**.

5 The Import dialog opens displaying the items you selected for import.

- Double-click the **CAGE Item Name** column cells to edit item names.
- Choose to replace BTQ, BTQ_MBT, MBT and MBT_Boundary. When replacing items, double-click the **CAGE Item Name** column cells to open a dialog to select the correct item to replace.
- Click **OK** to import the items.

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

3

Diesel Case Study

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

Setting Up Calibration Tables to Fill (p. 3-33)	Creating and duplicating tables to fill with optimization results.
Setting Up the Optimization (p. 3-34)	Using the Optimization Wizard to set up the optimization.
Setting Up Constraints (p. 3-37)	Setting up the required constraints for this optimization.
Defining Variable Values (p. 3-49)	Defining the operating points where you want the optimization to run.
Running the Optimization (p. 3-51)	Running the optimization and viewing the results.
Filling Tables with Optimization Results (p. 3-53)	Filling tables with the results of the optimization.

Diesel Case Study Overview

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

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

1 Create a design for your experiment“Design of Experiment” on page 3-5

2 Create models from the collected data“Modeling” on page 3-17

The Model Browser section of the case study covers design of experiment, data handling, and model construction and export. In the later CAGE browser section of the case study you use the models to complete the optimization of the calibration tables, see “Optimized Calibration” on page 3-27.

Problem Definition

Produce optimal calibration tables in speed and torque for

Best injection timing	`soi`
Best fuel quantity	`basefuelmass`
Best fuel pressure	`fuelpress`
Best VGT	`grackmea`
Best EGR	`egrpos, egrmf`

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

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

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

Design of Experiment

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

Variables are

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

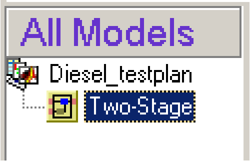
You need to set up a test plan before you can make designs. This experiment is set up as a two-stage test plan with start-of-injection (SOI) sweeps at the local level and the other five variables at the global level.

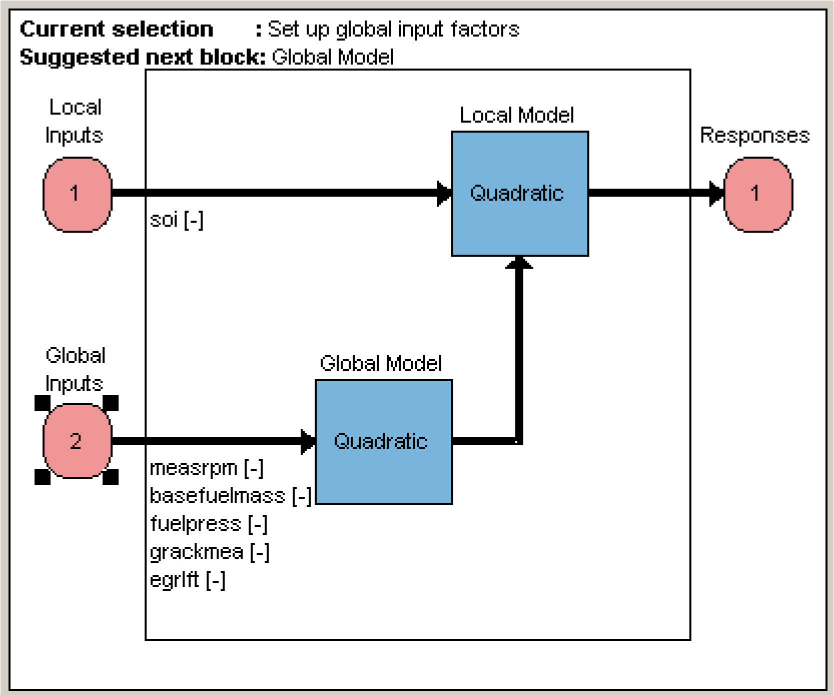
Open the example session with the test plan set up as follows:

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

2. Select **File > Open Project**. Locate the example session with the test plan set up, `Diesel_testplan.mat`, in the `mbctraining` directory and double click to load the project.

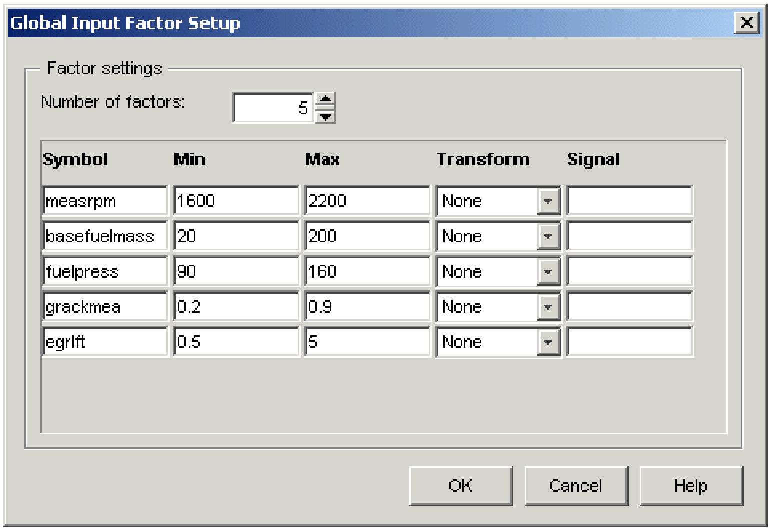
3. Click the Two-Stage test plan node in the model tree to see the test plan diagram.





Current selection : Set up global input factors
Suggested next block: Global Model

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



Global Input Factor Setup

Factor settings

Number of factors: 5

Symbol	Min	Max	Transform	Signal
measrpm	1600	2200	None	
basefuelmass	20	200	None	
fuelpress	90	160	None	
grackmea	0.2	0.9	None	
egrlft	0.5	5	None	

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

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

Some possible models are

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

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

Constraining the Design

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

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

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

To add a constraint to a design,

1. First open the Design Editor by right-clicking the Global Model block in the test plan diagram and selecting **Design Experiment**.

2. Click the New Design button in the toolbar or select **File > New Design**. A new node called `Linear Model Design` appears.

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

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

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

5. The Edit Constraint dialog with available constraints appears. Make sure the default `1D Table` is selected from the **Constraint Type** drop-down menu. These are easier to set up than linear constraints, although working

out the linear constraint numbers might be worthwhile for larger problems as it is faster.

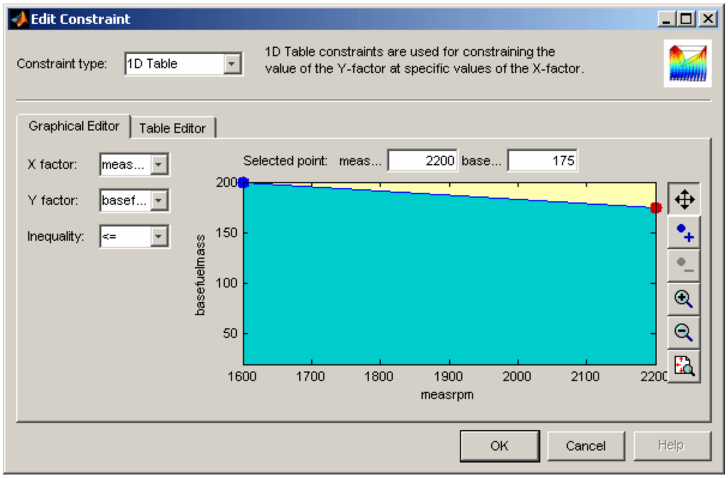
6 You can select the appropriate factors to use. For the first constraint, choose `measrpm`, `basefuelmass`, and the inequality <= from the menus.

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

7 Select the **Table Editor** tab and edit the **Number of breakpoints** to `2`, and click **Span range**.

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

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



Edit Constraint

Constraint type: 1D Table

1D Table constraints are used for constraining the value of the Y-factor at specific values of the X-factor.

Graphical Editor | Table Editor

X factor: meas...

Y factor: basef...

Inequality: <=

Selected point: meas... 2200 base... 175

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

10 Click **OK** to return to the Constraints Manager.

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

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

`fuelpress`

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

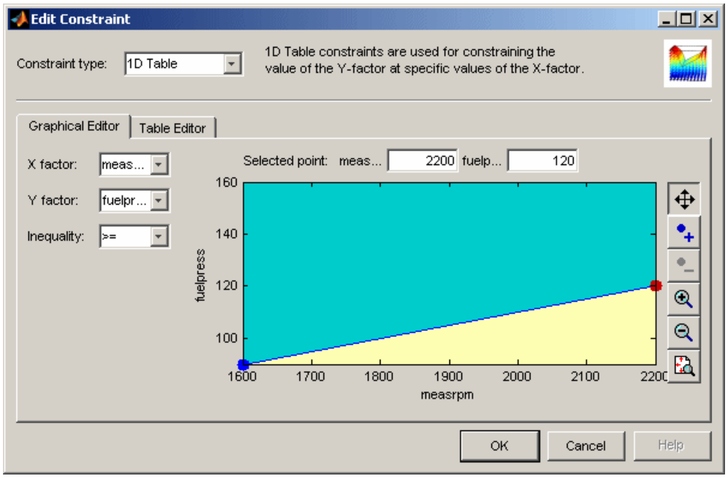
You achieve this by defining two constraints. In the first, the two table points define a `fuelpress` minimum of 90 at 1600 rpm and a minimum of 120 at 2200 rpm. In the second, the two table points define a `fuelpress` maximum of 110 at 1600 rpm and a maximum of 160 at 2200 rpm.

12 In the Edit Constraint dialog, change the Y factor to `fuelpress` and leave the X factor as `measrpm`.

13 Change the Inequality to `>=`.

14 Select the left point (where measrpm = 1600) and enter `90` in the **fuelpress** edit box.

15 Select the right point (where measrpm = 2200) and enter `120` in the **fuelpress** edit box. The dialog should look like the following.



Edit Constraint

Constraint type: 1D Table

1D Table constraints are used for constraining the value of the Y-factor at specific values of the X-factor.

X factor: meas...

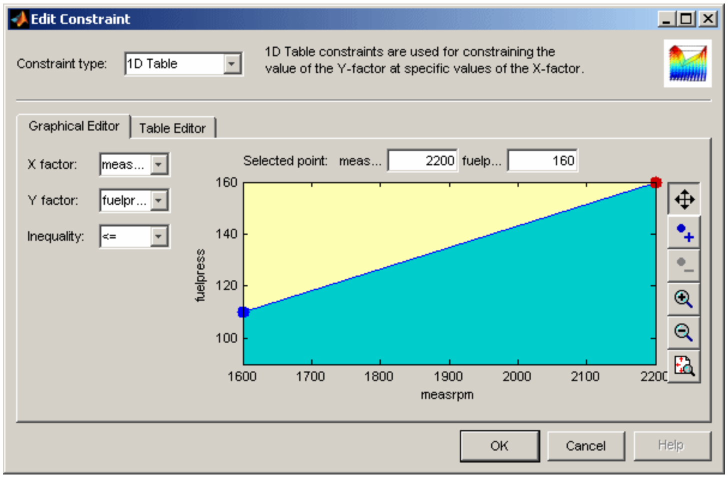
Y factor: fuelpr...

Inequality: >=

Selected point: meas... 2200 fuelp... 120

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

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



Edit Constraint

Constraint type: 1D Table

1D Table constraints are used for constraining the value of the Y-factor at specific values of the X-factor.

X factor: meas...

Y factor: fuelpr...

Inequality: <=

Selected point: meas... 2200 fuelp... 160

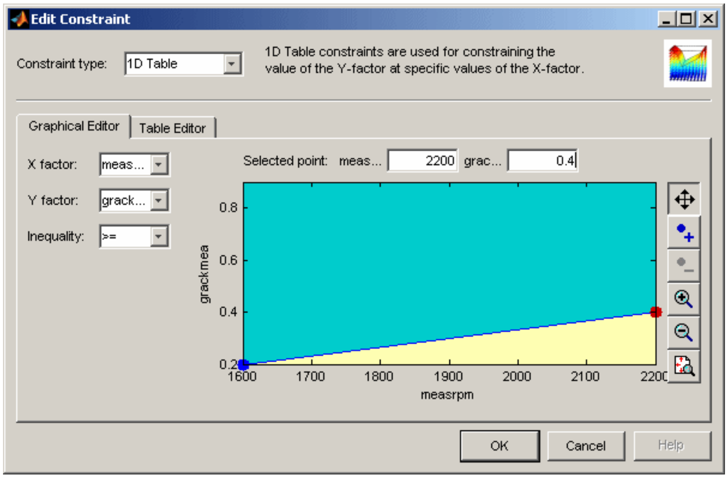
Click **OK** to return to the Constraint Manager.

17 Complete the other constraints in a similar way.

`grackmea`

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

This is achieved as shown in the following two constraints.



Edit Constraint

Constraint type: 1D Table

1D Table constraints are used for constraining the value of the Y-factor at specific values of the X-factor.

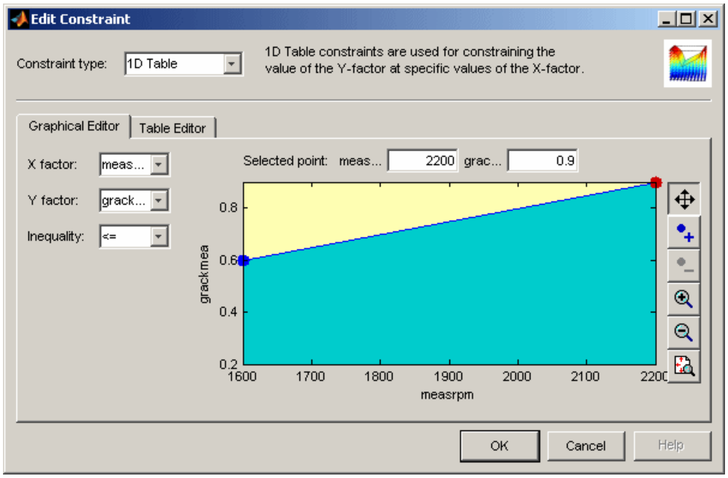
Graphical Editor | Table Editor

X factor: meas...

Y factor: grack...

Inequality: >=

Selected point: meas... 2200 grac... 0.4



Edit Constraint

Constraint type: 1D Table

1D Table constraints are used for constraining the value of the Y-factor at specific values of the X-factor.

Graphical Editor | Table Editor

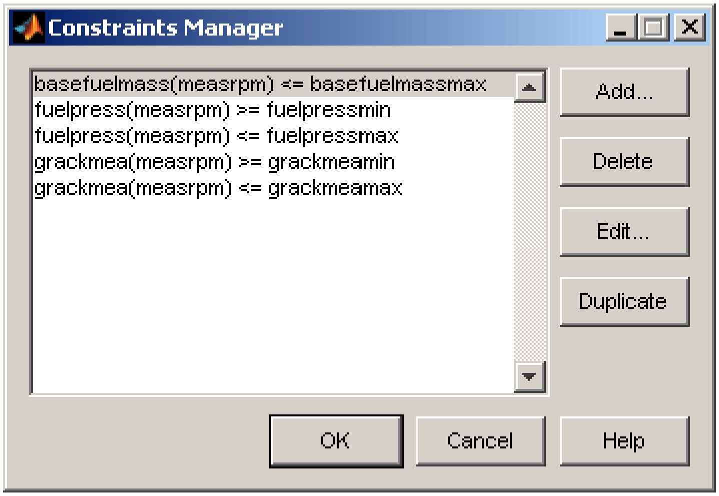
X factor: meas...

Y factor: grack...

Inequality: <=

Selected point: meas... 2200 grac... 0.9

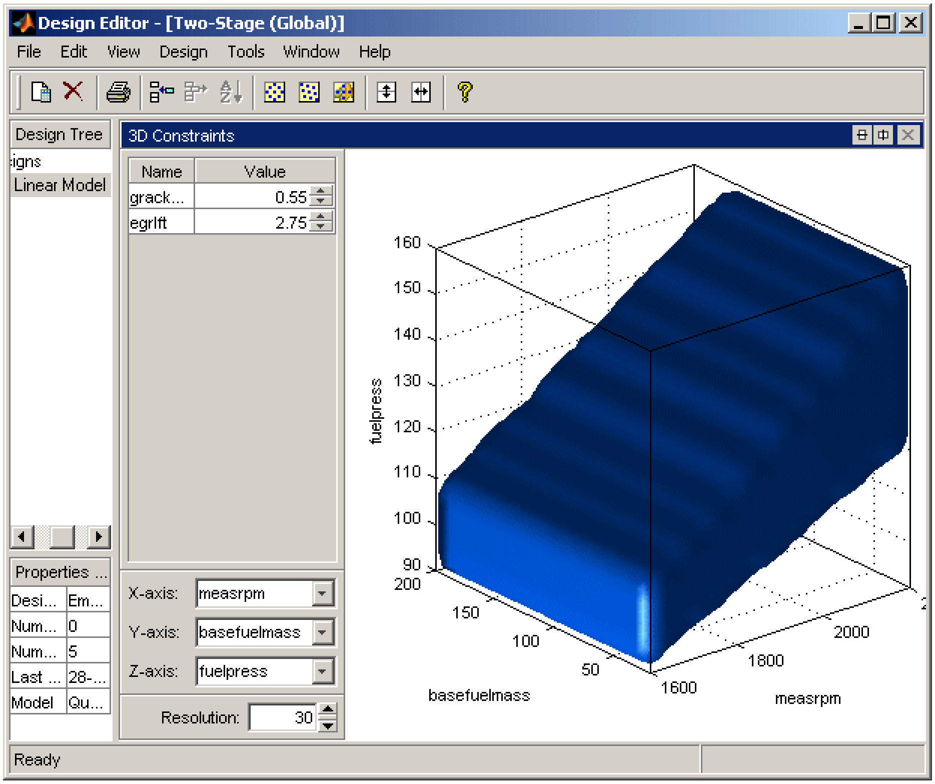
18 The Constraints Manager should contain all five constraints as shown.



basefuelmass(measrpm) <= basefuelmassmax
fuelpress(measrpm) >= fuelpressmin
fuelpress(measrpm) <= fuelpressmax
grackmea(measrpm) >= grackmeamin
grackmea(measrpm) <= grackmeamax

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

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



Name	Value
grack...	0.55
egrlft	2.75

X-axis: measrpm

Y-axis: basefuelmass

Z-axis: fuelpress

Resolution: 30

Desi...	Em...
Num...	0
Num...	5
Last ...	28-...
Model	Qu...

Creating Candidate Designs

Number of points

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

Design type

- V-optimal: reduces average prediction error

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

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

1 Click Optimal Design in the toolbar. The Optimal Design dialog opens.

2 Enter `65` in the **Total number of points** edit box.

3 Select `V-Optimal` from the **Optimality criteria** drop-down menu and click **OK**.

4 The Optimizing Design dialog appears. Click **Accept** when iterations are not producing noticeable improvements; that is, the graph becomes very flat.

5 Examine the design points and compare to the constraint space by right-clicking the 3D Constraints view and selecting **Split View > 3D Design Projection**.

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

Data Collection

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

You can use the toolbox to import test data, view it, sort it into tests, verify ranges, filter out unwanted points, and select data for modeling. For details on any of these processes, see the examples in the gasoline case study section “Importing and Filtering Data” on page 2-17, and for comprehensive information on data handling in the toolbox, see “Data” in the Model Browser documentation. See also the examples in the Chapter 7, “Tutorial: Data Editor”.

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

Modeling

See “Introduction to Two-Stage Modeling” on page 2-5 and “How Is a Two-Stage Model Constructed?” on page 2-26 for information about two-stage models (in the Help Browser you can right-click and select **Back** to return to this page).

Outline of modeling steps you will cover in this example:

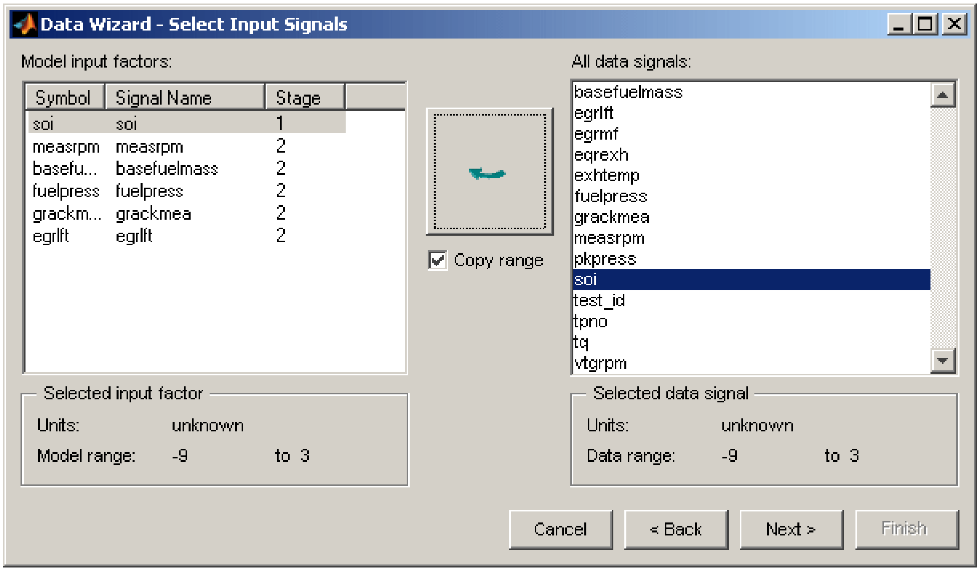
1. Build the models specified in the test plan.
2. Review local fits. Is the local model flexible enough?
3. Eliminate outliers.
4. Try alternative local models.
5. Review global fits. Is the global model flexible enough?
6. Eliminate outliers.
7. Try alternative global models.
8. Select best models.
9. Clean up tree and export models.

Building Models

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

1. Double-click the Responses block to select data signals for modeling and build response models.
2. The Data Wizard appears. You want to use all selected data, so click **Next** on the Select Data screen.
3. Select the check box to **Copy range** of the signals, then match up model input names with data signal names one pair at a time and click the button to associate each pair. As the input signal names are set up to match

the data signal names, each model input factor you select is matched automatically in the data signals list, so you can then click the button to select the data signal and copy the range. If the names did not match you would have to select data signals manually.



Model input factors:

Symbol	Signal Name	Stage
soi	soi	1
measrpm	measrpm	2
basefu...	basefuelmass	2
fuelpress	fuelpress	2
grackm...	grackmea	2
egrlft	egrlft	2

☑ Copy range

All data signals:

basefuelmass
egrlft
egrmf
eqrexh
exhtemp
fuelpress
grackmea
measrpm
pkpress
soi
test_id
tpno
tq
vtgrpm

Selected input factor

Units: unknown

Model range: -9 to 3

Selected data signal

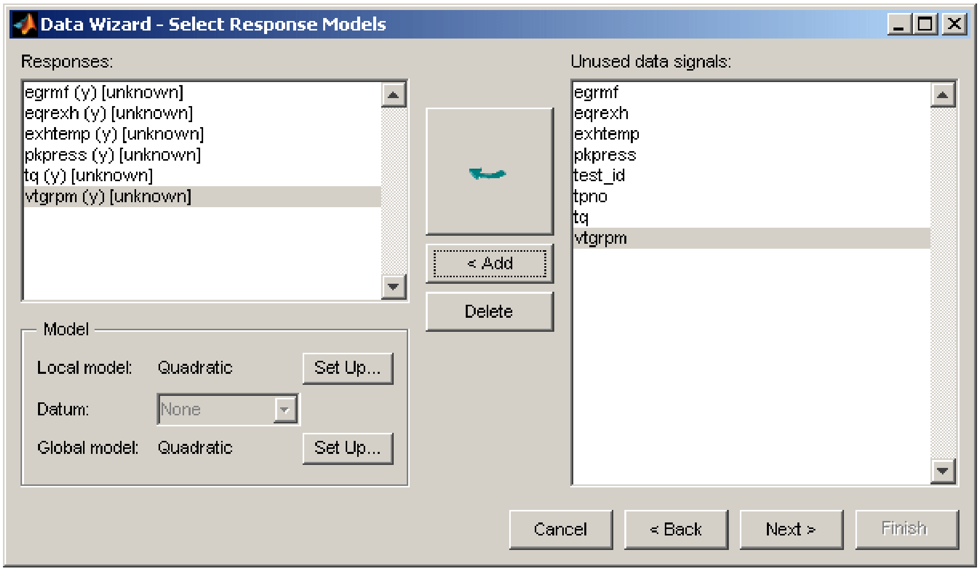
Units: unknown

Data range: -9 to 3

Click **Next**.

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

- `egrmf`
- `eqrexh`
- `pkpress`
- `tq`
- `vtgrpm`



Responses:

egrmf (y) [unknown]
eqrexh (y) [unknown]
exhtemp (y) [unknown]
pkpress (y) [unknown]
tq (y) [unknown]
vtgrpm (y) [unknown]

Unused data signals:

egrmf
eqrexh
exhtemp
pkpress
test_id
tpno
tq
vtgrpm

Local model: Quadratic

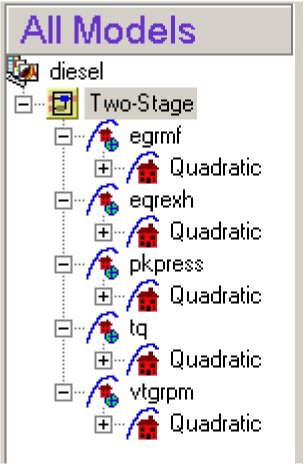
Datum: None

Global model: Quadratic

5 Click **Next**, and you are not matching data to a design so you can click **Finish**. The Data Editor appears so you can check the data selected for modeling.

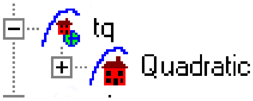
6 To accept the data for modeling, close the Data Editor. Click **Yes** in the dialog asking if you want to build models and update the design, then the models are calculated.

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

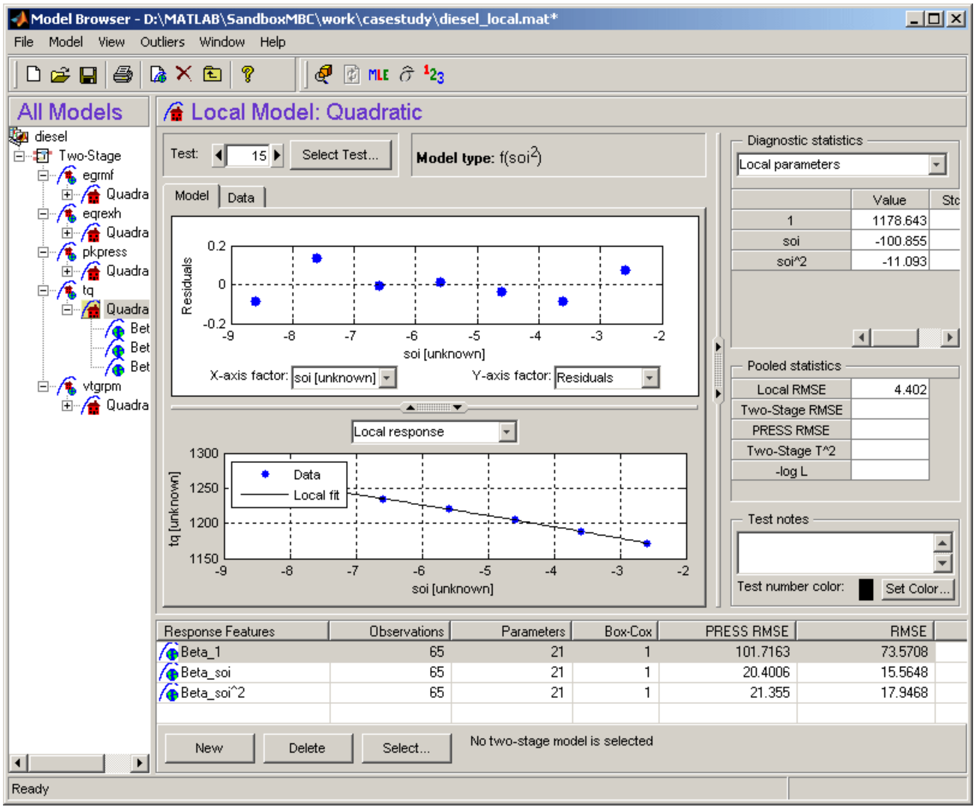


Building and Evaluating Alternative Models

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



The local model view appears.



Test: 15

Model type: $f(soi^2)$

X-axis factor: soi [unknown]

Y-axis factor: Residuals

Diagnostic statistics: Local parameters

	Value	Stc
1	1178.643	
soi	-100.855	
soi^2	-11.093	

Pooled statistics

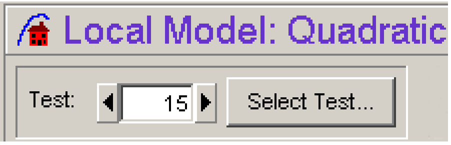
Local RMSE	4.402
Two-Stage RMSE	
PRESS RMSE	
Two-Stage T^2	
-log L	

Test number color: ■

Response Features	Observations	Parameters	Box-Cox	PRESS RMSE	RMSE
Beta_1	65	21	1	101.7163	73.5708
Beta_soi	65	21	1	20.4006	15.5648
Beta_soi^2	65	21	1	21.355	17.9468

No two-stage model is selected

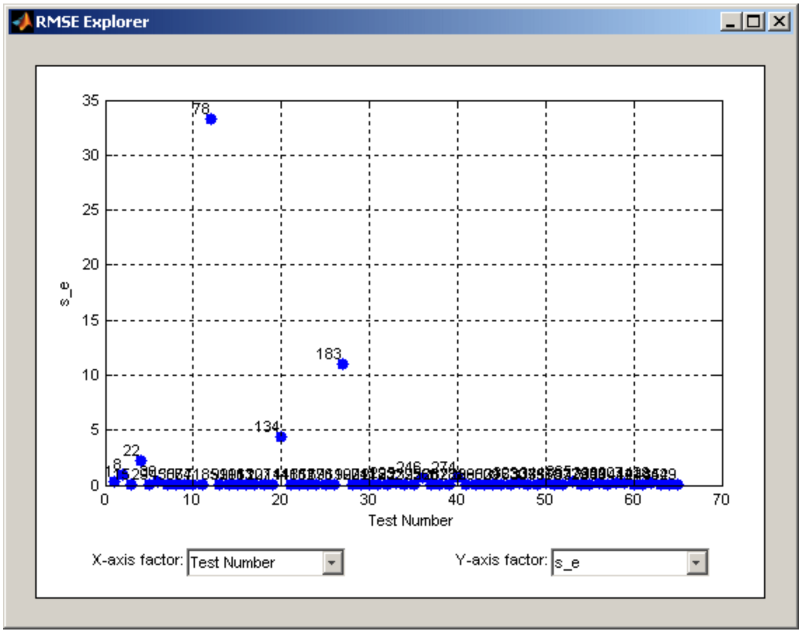
2 Look through the tests to inspect the fits. Use the **Test** controls.



Test: 15

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

3 To quickly identify problem tests with high errors, click RMSE Plots $\hat{\sigma}$ in the toolbar (or **View** menu). Right-click to turn the test number display on, then inspect tests with high error values in the local model view. Strongly outlying residuals should be investigated.

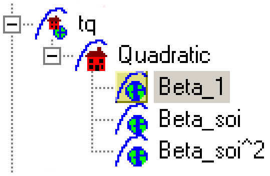


X-axis factor: Test Number

Y-axis factor: s_e

4 Consider removing outliers to improve fits if some points are badly distorting the fit. Engineering judgment is required to judge whether suspect data should be removed. Be careful not to remove outliers without good reason. If you keep removing points you can always get a better fit, but your aim is to achieve a model that predicts the data well. You can click the **Data** tab to inspect data variables for suspect tests.

5 Use the same principles to review the global fits. For example, expand the local model node (`Quadratic`) in the model tree and click `Beta_1`.



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

6 At both local and global level, create alternative model types to compare.

a From the response node, create alternative local models to compare.

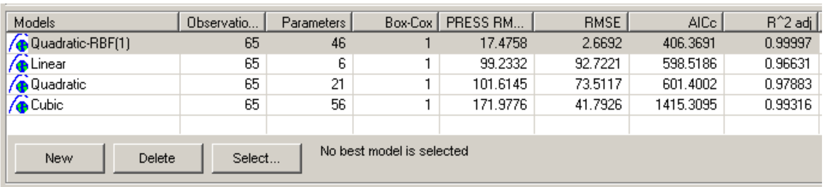
b From the local nodes, create alternative response feature models to compare.

c From the global nodes, create alternative child global models to compare.

To create models one at a time, click **New** from the response node to create a new local model node, or from the global node to create new child global model nodes, etc. Choose different model types from the Model Setup dialog, or you can change any existing model by selecting **Model > Set Up**. There are step-by-step instructions for doing this in the gasoline case study, see "Create Multiple Models to Compare" on page 2-41. You could also work through the modeling tutorial for more guided examples of how to select models, see Chapter 5, "Tutorial: Model Quickstart ".

When you have built a selection of different model types as child nodes of, say, a global model node, you can click the parent model node and select **Model > Make Template**. Save the template to a suitable directory, then you can use **Build Models** (in the toolbar) to automatically build the same selection of child model types for any other model.

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



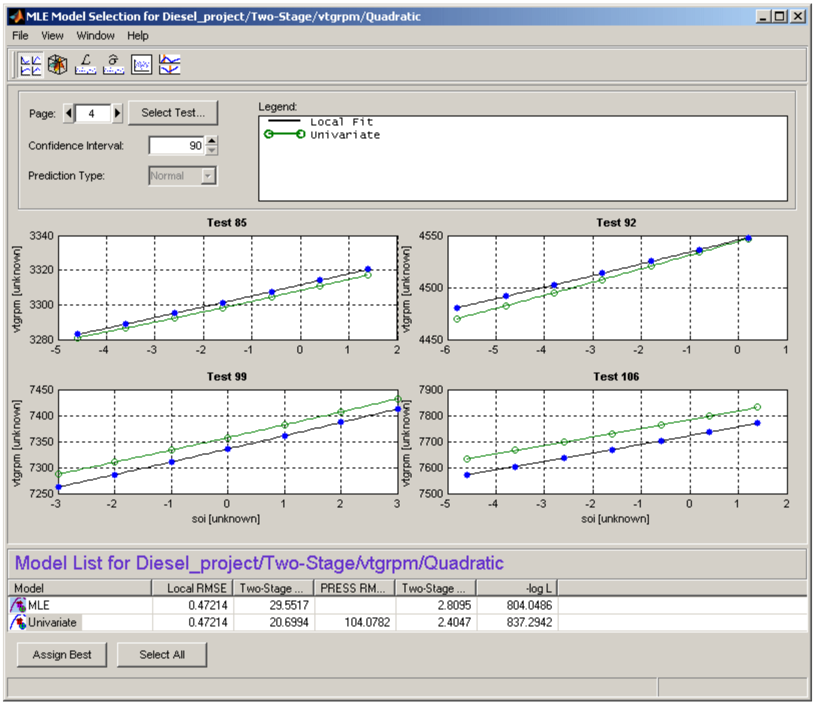
Models	Observatio...	Parameters	Box-Cox	PRESS RM...	RMSE	AICc	R^2 adj
Quadratic-RBF(1)	65	46	1	17.4758	2.6692	406.3691	0.99997
Linear	65	6	1	99.2332	92.7221	598.5186	0.96631
Quadratic	65	21	1	101.6145	73.5117	601.4002	0.97883
Cubic	65	56	1	171.9776	41.7926	1415.3095	0.99316

No best model is selected

Review the fits graphically as well. Search for the best fit but be careful not to overfit. You can increase the number of terms in a model until every point is exactly on the line, but the predictive power of that model will be very low. PRESS RMSE can be the most helpful single statistic you can use to search for the best fit relative to the number of terms in the model. However, you should not rely on any single statistic, but use a variety of criteria and especially the graphical tools available for comparison of models in the Model Evaluation tool when you click **Select**. For detailed guidance see “Creating Multiple Models To Compare” in the Model Browser documentation.

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

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



MLE Model Selection for Diesel_project/Two-Stage/vtgrpm/Quadratic

Page: 4

Confidence Interval: 90

Prediction Type: Normal

Model List for Diesel_project/Two-Stage/vtgrpm/Quadratic

Model	Local RMSE	Two-Stage ...	PRESS RM...	Two-Stage ...	-log L
MLE	0.47214	29.5517		2.8095	804.0486
Univariate	0.47214	20.6994	104.0782	2.4047	837.2942

8 Try MLE (Maximum Likelihood Estimation). You can choose to calculate MLE in the dialog that appears immediately after building a two-stage model, or you can click **Cancel** and choose to try MLE later. You can use the MLE toolbar button to calculate an MLE model any time. This process refits, taking proper account of the correlation between different response features. Once you have an MLE model, you can click **Select** from the local node to compare it with the univariate two-stage model. You can choose the univariate model as best here if you want; this is the way to "go back" from MLE.

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

When you are satisfied with the models, select **File > Export**. Note that the models exported depend on the model node you have selected in the tree. To export all models in the test plan, select the test plan node. You can export your models to file, to CAGE, to the workspace or to Simulink, which makes them easy to share across engineering groups. To discard all but the models you chose as best, select **File > Clean-up Tree**.

Try exporting the models directly to the CAGE browser part of the Model-Based Calibration toolbox for optimal calibration (CAGE must be open first). Alternatively you can export the models to file in order to import them into CAGE later. See “Exporting the Models” on page 2-54 in the gasoline case study for instructions.

You can now use these models in the CAGE part of the Model-Based Calibration Toolbox to produce optimized calibration tables — see the next section.

Optimized Calibration

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

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`)

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

1. Load models of engine responses — “Importing Models of Engine Responses into CAGE” on page 3-30.

2. Define additional models and variables required by optimization strategy — “Defining Additional Variables and Models” on page 3-31.

3. Set up tables — “Setting Up Calibration Tables to Fill” on page 3-33.

4. Set up the optimization — “Setting Up the Optimization” on page 3-34.

5. Define constraints — “Setting Up Constraints” on page 3-37.
6. Define optimization operating points — “Defining Variable Values” on page 3-49.
7. Run the optimization and view results — “Running the Optimization” on page 3-51
8. Fill tables from optimization results – “Filling Tables with Optimization Results” on page 3-53.

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

Benefits of Automated Calibration

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

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

- Example control applications

  Emissions-constrained BSFC optimization over 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

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.

Note you can also:

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

Defining Additional Variables and Models

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

Follow these steps:

1. Select **File > New > Variable Item > Variable**. The Variable Dictionary view appears.
   a. Rename the new variable `tq_desired`.
   b. Set the range of this variable to be minimum `0` and maximum `1500`.
   c. Edit the set point to `600` Nm
2. 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**. It is important to check for typos or this step can create unintended new inputs. Click **Finish**.

      The Models view appears.
3. This optimization requires a constraint on the air/fuel ratio (AFR). However, as you could not model AFR directly you need to create a function model that relates equivalence ratio to air/fuel ratio. Then you can constrain AFR. To do this, select **File > New > Function Model**.
   a. Enter `afr = 14.46/eqrexh` and click **Next**.
   b. Select **Automatically assign/create inputs** and click **Finish**.
4. Select **File > New > Variable Item > Variable**. The Variable Dictionary view appears.

a Rename the new variable `afr_min`.

b Set the range of this variable to be minimum `0` and maximum `100`

c Edit the set point to `50`

Setting Up Calibration Tables to Fill

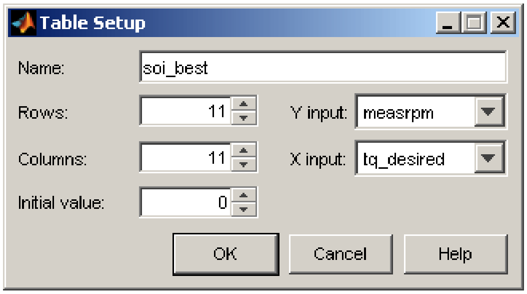
1 Set up a new 2-D table. Select **File > New > 2D Table**.

2 Name the table `soi_best`.

3 Select `measrpm` and `tq_desired` from the **Y Input** and **X Input** drop-down menus.

4 Enter 11 rows and 11 columns.

5 Enter 0 for the initial value.



Name: soi_best

Rows: 11

Columns: 11

Initial value: 0

Y input: measrpm

X input: tq_desired

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

Look at the **Tables** tree on the left. Note that the normalizers appear as calibrateable 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`

Setting Up the Optimization

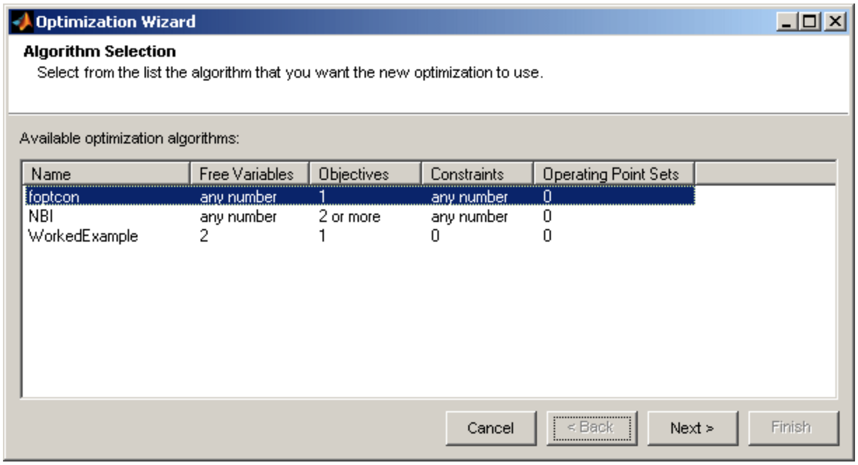
CAGE provides a flexible optimization environment. You can define the objectives, the constraints, and the points where the optimization is carried out.

The objective is to 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 `foptcon` is selected by default, and this is the optimization algorithm you will use for this example. Note that this algorithm specifies a single objective in the **Objectives** column.



Optimization Wizard

Algorithm Selection

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

Available optimization algorithms:

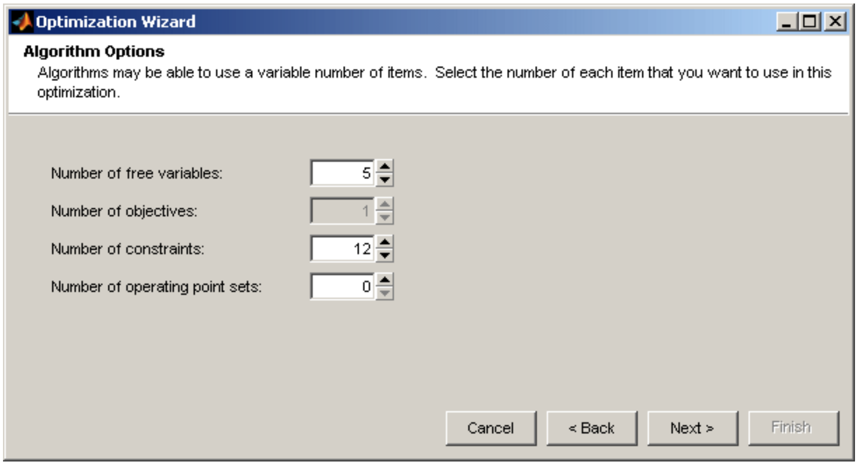
Name	Free Variables	Objectives	Constraints	Operating Point Sets
foptcon	any number	1	any number	0
NBI	any number	2 or more	any number	0
WorkedExample	2	1	0	0

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



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: 5

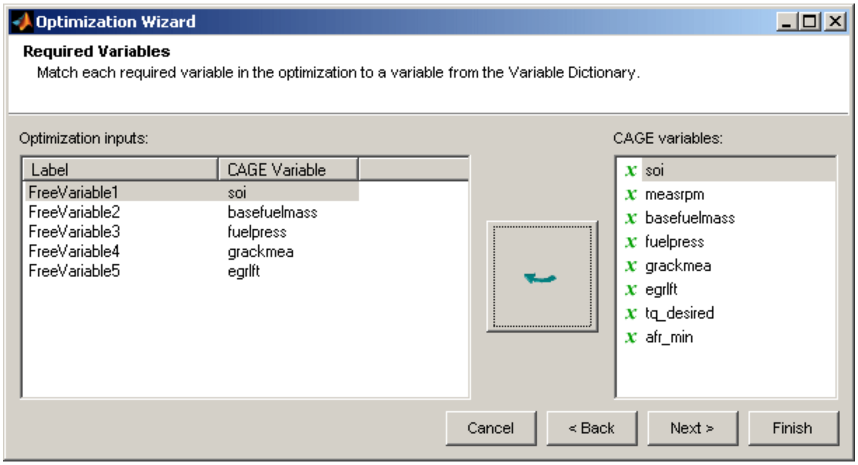
Number of objectives: 1

Number of constraints: 12

Number of operating point sets: 0

Click **Next**.

4 Select the free variables.



Optimization Wizard

Required Variables

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

Optimization inputs:

Label	CAGE Variable
FreeVariable1	soi
FreeVariable2	basefuelmass
FreeVariable3	fuelpress
FreeVariable4	grackmea
FreeVariable5	egrlft

CAGE variables:

- soi
- measrpm
- basefuelmass
- fuelpress
- grackmea
- egrlft
- tq_desired
- afr_min

Select in turn each of the following free variables and click the button to select them:

a `soi`

b `basefuelmass`

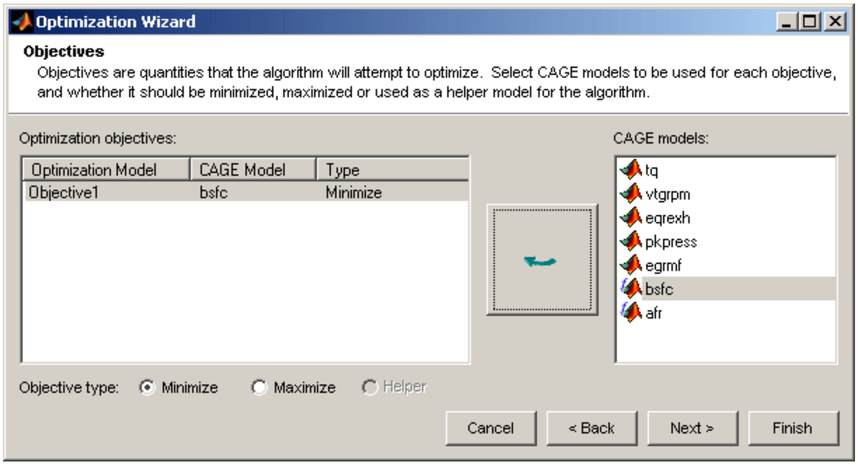
c `fuelpress`

d `grackmea`

e `egrlft`

Click **Next**.

5 Select `bsfc` from the list of models on the right and click the button to select this model for the objective. Leave the objective type set to **Minimize**.



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	bsfc	Minimize

CAGE models:

- tq
- vtgrpm
- eqrexh
- pkpress
- egrmf
- bsfc
- afr

Objective type: ● Minimize ○ Maximize ○ Helper

Click **Finish**.

The **Optimization** view appears. You have not yet set up your constraints, so you can add them here.

Setting Up Constraints

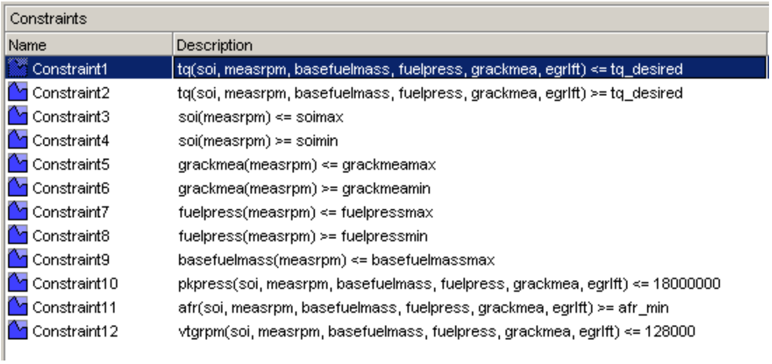
You need to set up two types of constraint, as described in the following sections:

- "Model Constraints" on page 3-38
- "1–D Table Constraints" on page 3-40

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

- `tq_desired`
- `vtgrpm`
- `pkpress`
- `afr`
- `soi`
- `grackmea`
- `fuelpress`
- `basefuelmass`

In the Optimization view these constraints are shown as follows.



Constraints

Name	Description
Constraint1	tq(soi, measrpm, basefuelmass, fuelpress, grackmea, egrlft) <= tq_desired
Constraint2	tq(soi, measrpm, basefuelmass, fuelpress, grackmea, egrlft) >= tq_desired
Constraint3	soi(measrpm) <= soimax
Constraint4	soi(measrpm) >= soimin
Constraint5	grackmea(measrpm) <= grackmeamax
Constraint6	grackmea(measrpm) >= grackmeamin
Constraint7	fuelpress(measrpm) <= fuelpressmax
Constraint8	fuelpress(measrpm) >= fuelpressmin
Constraint9	basefuelmass(measrpm) <= basefuelmassmax
Constraint10	pkpress(soi, measrpm, basefuelmass, fuelpress, grackmea, egrlft) <= 18000000
Constraint11	afr(soi, measrpm, basefuelmass, fuelpress, grackmea, egrlft) >= afr_min
Constraint12	vtgrpm(soi, measrpm, basefuelmass, fuelpress, grackmea, egrlft) <= 128000

Model Constraints

Set up these constraints as follows:

1. Double-click to edit the first constraint to `tq <= tq_desired`. The Constraint Editor appears.

   a. Select Constraint type `Model` (select `Model` from the drop-down menu)

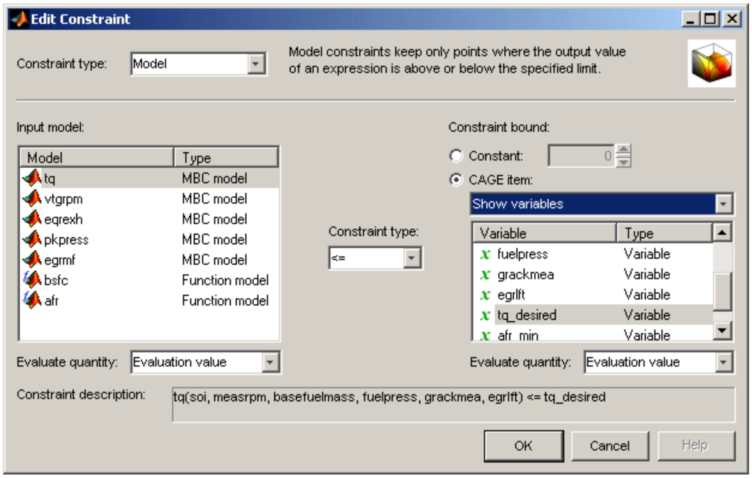
   b. Select `tq` in the **Input model** list

   c. Select <= as the **Constraint type**

   d. Select the **CAGE item** radio button

   e. Select `Show variables` from the drop-down list.

   f. Select `tq_desired` in the variable list. The dialog should look as shown.



Constraint type: Model

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

Input model:

Model	Type
tq	MBC model
vtgrpm	MBC model
eqrexh	MBC model
pkpress	MBC model
egrmf	MBC model
bsfc	Function model
afr	Function model

Evaluate quantity: Evaluation value

Constraint type: <=

Constraint bound:

○ Constant: 0

● CAGE item:

Show variables

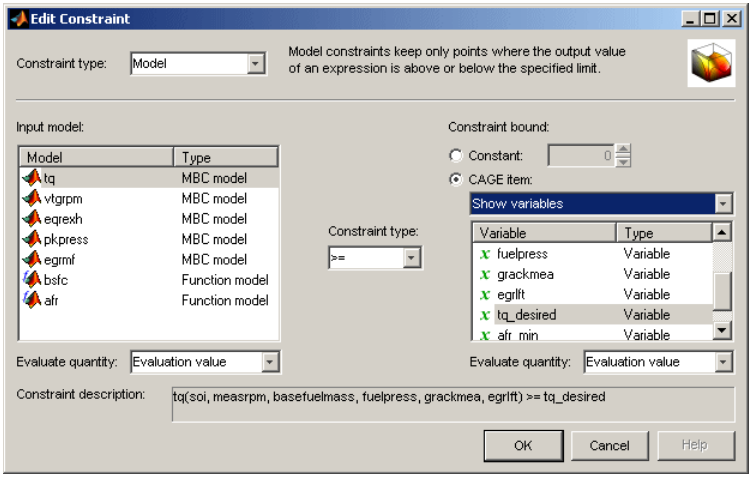
Variable	Type
fuelpress	Variable
grackmea	Variable
egrlft	Variable
tq_desired	Variable
afr_min	Variable

Evaluate quantity: Evaluation value

Constraint description: tq(soi, measrpm, basefuelmass, fuelpress, grackmea, egrlft) <= tq_desired

   g. Click **OK** to return to the Optimization view.

2. Similarly, double-click to edit the next constraint as shown: `tq >= tq_desired`.



Edit Constraint

Constraint type: Model

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

Input model:

Model	Type
tq	MBC model
vtgrpm	MBC model
eqrexh	MBC model
pkpress	MBC model
egrmf	MBC model
bsfc	Function model
afr	Function model

Evaluate quantity: Evaluation value

Constraint type: >=

Constraint bound:

○ Constant: 0

● CAGE item:

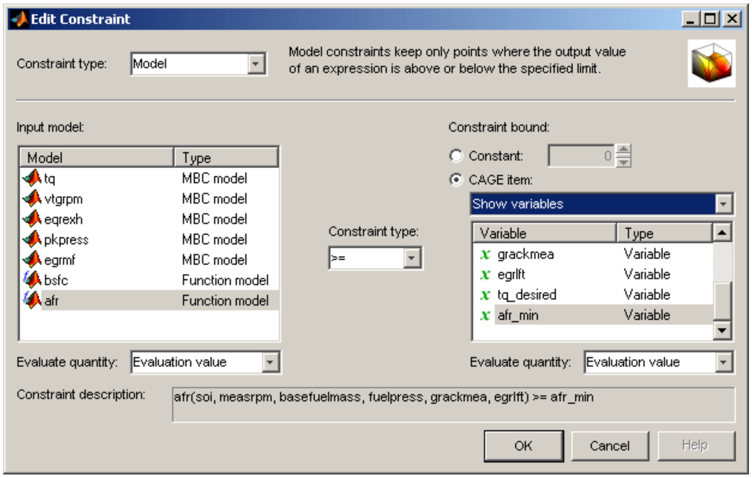
Show variables

Variable	Type
fuelpress	Variable
grackmea	Variable
egrlft	Variable
tq_desired	Variable
afr_min	Variable

Evaluate quantity: Evaluation value

Constraint description: tq(soi, measrpm, basefuelmass, fuelpress, grackmea, egrlft) >= tq_desired

3 Repeat to set up the constraint `afr >= afr_min`, as shown.



Edit Constraint

Constraint type: Model

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

Input model:

Model	Type
tq	MBC model
vtgrpm	MBC model
eqrexh	MBC model
pkpress	MBC model
egrmf	MBC model
bsfc	Function model
afr	Function model

Evaluate quantity: Evaluation value

Constraint type: >=

Constraint bound:

○ Constant: 0

● CAGE item:

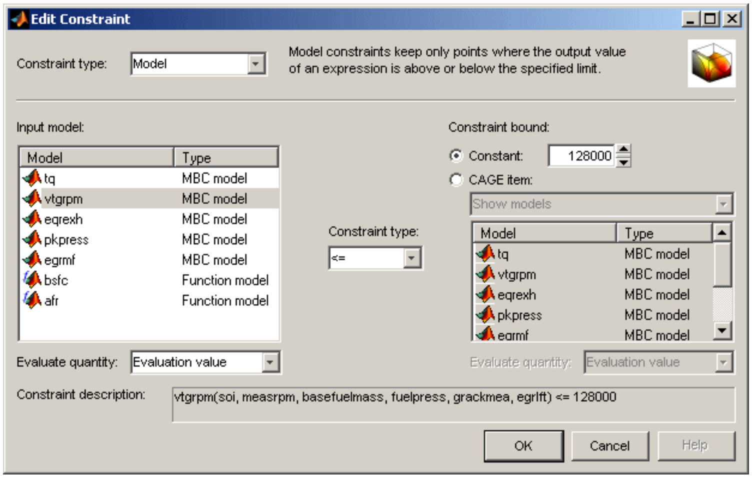
Show variables

Variable	Type
grackmea	Variable
egrlft	Variable
tq_desired	Variable
afr_min	Variable

Evaluate quantity: Evaluation value

Constraint description: afr(soi, measrpm, basefuelmass, fuelpress, grackmea, egrlft) >= afr_min

4 Repeat to set up the constraint `vtgrpm <= 128000`, as shown. Note, here you enter a value in the **Constant** edit box, rather than select a variable from the CAGE item list.



Edit Constraint

Constraint type: Model

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

Input model:

Model	Type
tq	MBC model
vtgrpm	MBC model
eqrexh	MBC model
pkpress	MBC model
egrmf	MBC model
bsfc	Function model
afr	Function model

Constraint type: <=

Constraint bound:

● Constant: 128000

○ CAGE item:

Show models

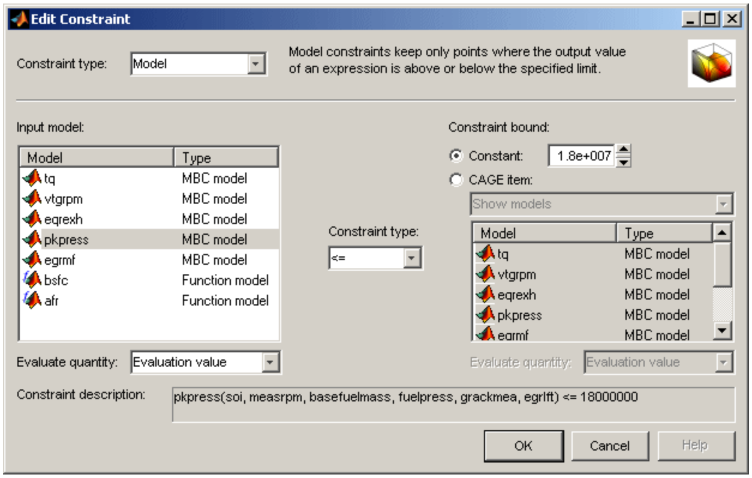
Model	Type
tq	MBC model
vtgrpm	MBC model
eqrexh	MBC model
pkpress	MBC model
egrmf	MBC model

Evaluate quantity: Evaluation value

Evaluate quantity: Evaluation value

Constraint description: vtgrpm(soi, measrpm, basefuelmass, fuelpress, grackmea, egrlft) <= 128000

5 Repeat to set up the constraint `pkpress <= 18000000`, as shown. Note, here you enter a value in the **Constant** edit box.



Edit Constraint

Constraint type: Model

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

Input model:

Model	Type
tq	MBC model
vtgrpm	MBC model
eqrexh	MBC model
pkpress	MBC model
egrmf	MBC model
bsfc	Function model
afr	Function model

Constraint type: <=

Constraint bound:

● Constant: 1.8e+007

○ CAGE item:

Show models

Model	Type
tq	MBC model
vtgrpm	MBC model
eqrexh	MBC model
pkpress	MBC model
egrmf	MBC model

Evaluate quantity: Evaluation value

Evaluate quantity: Evaluation value

Constraint description: pkpress(soi, measrpm, basefuelmass, fuelpress, grackmea, egrlft) <= 18000000

1–D Table Constraints

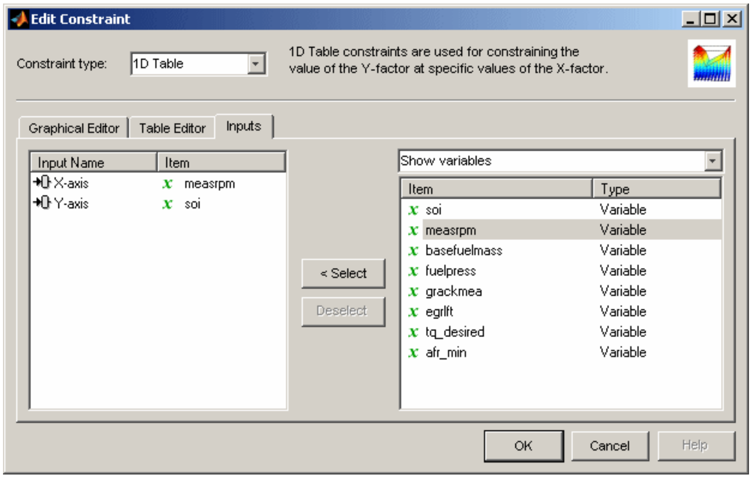
1 Double-click to edit the next constraint to be a 1D table defining `soi(measrpm)<=soimax` as follows.

a Select `1D Table` from the **Constraint type** drop-down menu

b Click the **Inputs** tab

   i Click `measrpm` in the variable item list and `X-axis` in the input name list and click **Select**. `measrpm` is selected for the `X-axis`.

   ii Similarly select `soi` for the `Y-axis`.



Constraint type: 1D Table

1D Table constraints are used for constraining the value of the Y-factor at specific values of the X-factor.

Input Name	Item
X-axis	measrpm
Y-axis	soi

Show variables

Item	Type
soi	Variable
measrpm	Variable
basefuelmass	Variable
fuelpress	Variable
grackmea	Variable
egrlft	Variable
tq_desired	Variable
afr_min	Variable

c Click the **Table Editor** tab.

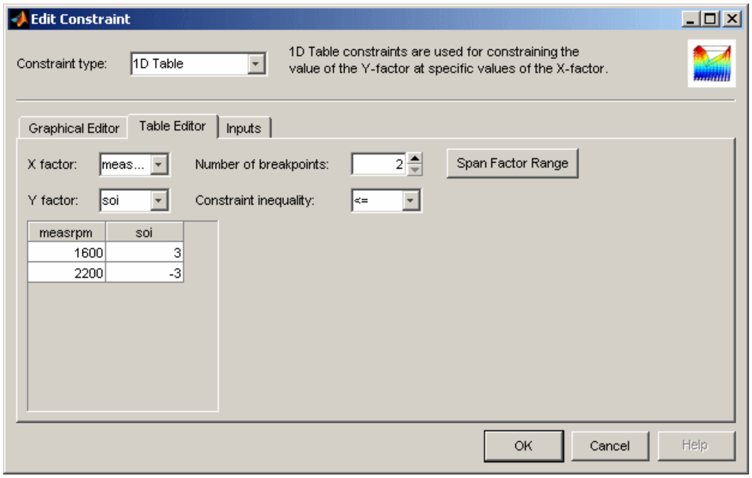
   iii Enter 2 for the **Number of breakpoints** and press **Enter**.

   iv Select <= as the **Constraint inequality**

   v Enter the following values in the table cells.

`measrpm`	`soi`
1600	3
2200	-3

The dialog should look as shown.



Constraint type: 1D Table

1D Table constraints are used for constraining the value of the Y-factor at specific values of the X-factor.

X factor: meas...

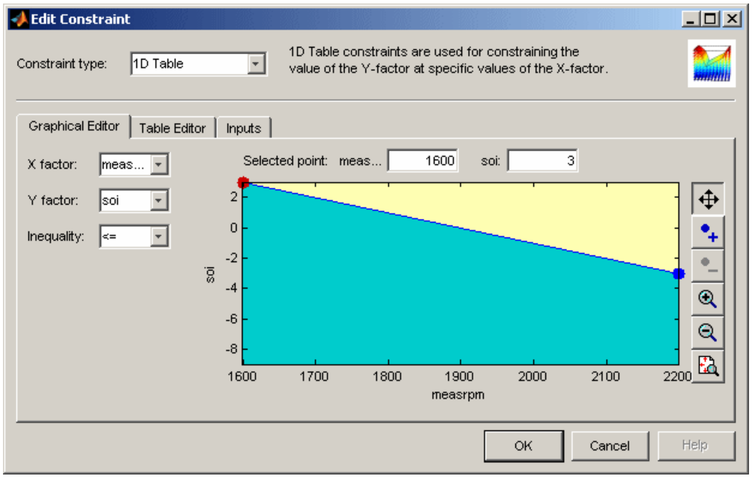
Y factor: soi

Number of breakpoints: 2

Constraint inequality: <=

measrpm	soi
1600	3
2200	-3

d Click the **Graphical editor** tab to view the constraint and check you have the correct inequality, as shown.



Constraint type: 1D Table

1D Table constraints are used for constraining the value of the Y-factor at specific values of the X-factor.

X factor: meas...

Y factor: soi

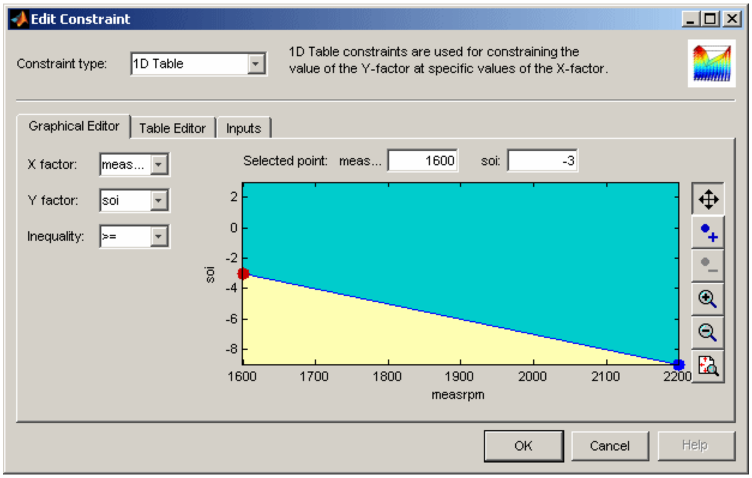
Inequality: <=

Selected point: meas... 1600 soi: 3

e Click **OK** to return to the Optimization view.

2 Similarly, double-click to edit the next constraint and repeat the steps to produce a 1D table defining `soi(measrpm)>=soimin` as shown.

`measrpm`	**`soi`**
1600	-3
2200	-9



Edit Constraint

Constraint type: 1D Table

1D Table constraints are used for constraining the value of the Y-factor at specific values of the X-factor.

Graphical Editor | Table Editor | Inputs

X factor: meas...

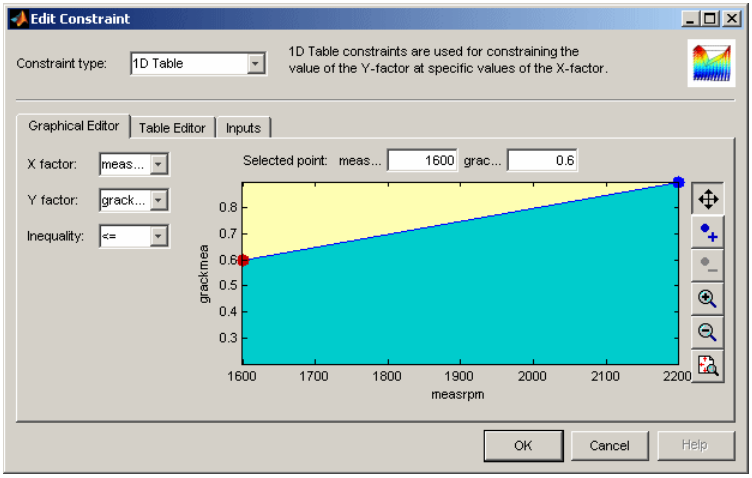
Y factor: soi

Inequality: >=

Selected point: meas... 1600 soi: -3

3 Repeat the steps to define `grackmea(measrpm)<=grackmeamax` as shown.

`measrpm`	`grackmea`
1600	0.6
2200	0.9



Constraint type: 1D Table

1D Table constraints are used for constraining the value of the Y-factor at specific values of the X-factor.

X factor: meas...

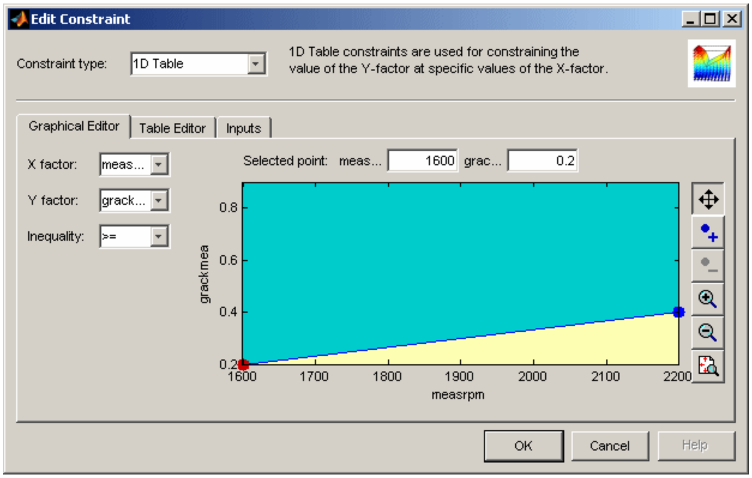
Y factor: grack...

Inequality: <=

Selected point: meas... 1600 grac... 0.6

4 Repeat the steps to define `grackmea(measrpm)>=grackmeamin` as shown.

`measrpm`	`grackmea`
1600	0.2
2200	0.4



Edit Constraint

Constraint type: 1D Table

1D Table constraints are used for constraining the value of the Y-factor at specific values of the X-factor.

X factor: meas...

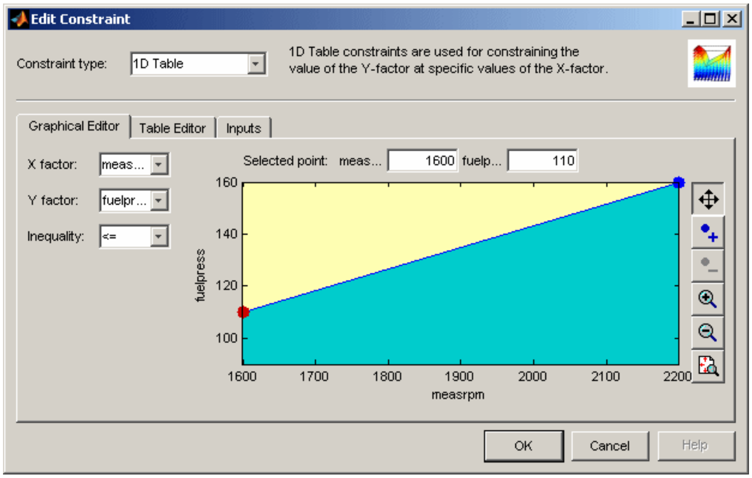
Y factor: grack...

Inequality: >=

Selected point: meas... 1600 grac... 0.2

5 Repeat the steps to define `fuelpress(measrpm)<=fuelpressmax` as shown.

`measrpm`	`fuelpress`
1600	110
2200	160



Edit Constraint

Constraint type: 1D Table

1D Table constraints are used for constraining the value of the Y-factor at specific values of the X-factor.

X factor: meas...

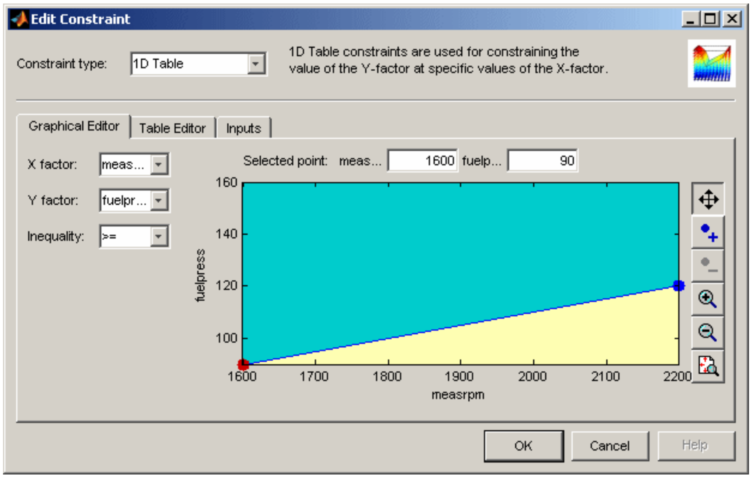
Y factor: fuelpr...

Inequality: <=

Selected point: meas... 1600 fuelp... 110

6 Repeat the steps to define `fuelpress(measrpm)>=fuelpressmin` as shown.

`measrpm`	`fuelpress`
1600	90
2200	120



Constraint type: 1D Table

1D Table constraints are used for constraining the value of the Y-factor at specific values of the X-factor.

X factor: meas...

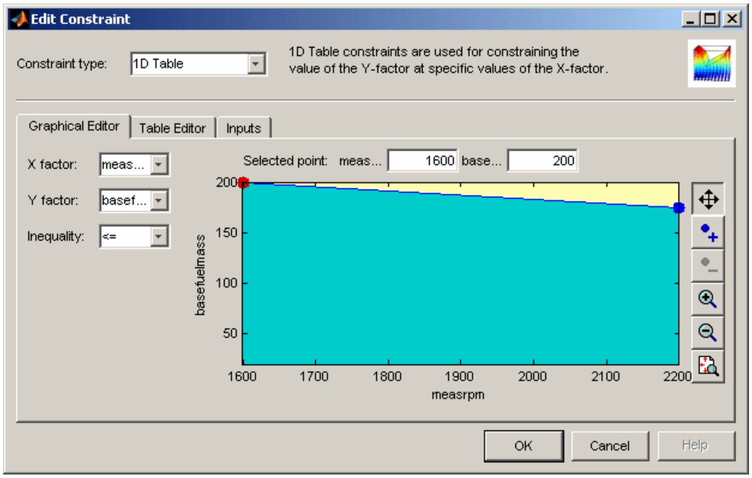
Y factor: fuelpr...

Inequality: >=

Selected point: meas... 1600 fuelp... 90

7 Repeat the steps to define `basefuelmass(measrpm)<=basefuelmassmax` as shown.

measrpm	basefuelmass
1600	200
2200	175



Edit Constraint

Constraint type: 1D Table

1D Table constraints are used for constraining the value of the Y-factor at specific values of the X-factor.

X factor: meas...

Y factor: basef...

Inequality: <=

Selected point: meas... 1600 base... 200

Defining Variable Values

You need to define the set of points where you want the optimization to run. To do this you use the **Free Variable Initial Values** and **Fixed Variable Values** panes in the Optimization view.

1 Increase the **Number of runs** to `7`. Click the buttons or enter the value in the box. The number of rows in both variable value panes increases to 7. The default values are the set point of each variable. Leave the initial values for the free variables at the defaults.

2 You can enter or copy values into the **Fixed Variable Values** pane to define the fixed variable values at each point where you want the optimization to run. You can copy all the variable values from a text file, or from the Help Browser copy each column in turn. Copy and paste the following values into the `measrpm` column in the **Fixed Variable Values** pane.

`measrpm`
2200
2200
2200
2200
1600
1600
1600

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

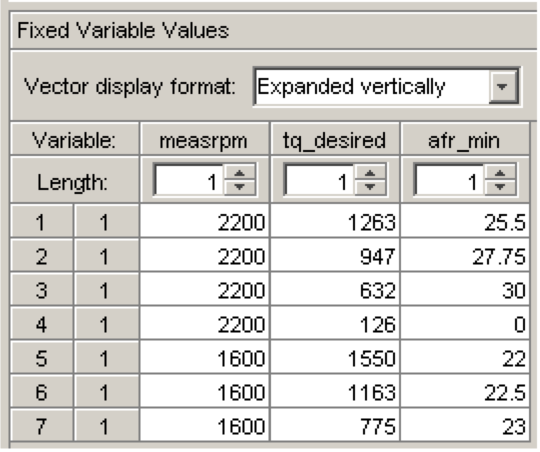
`tq_desired`
1263
947
632
126
1550

1163
775

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

`afr_min`
25.5
27.75
30.0
0
22
22.5
23

The **Fixed Variable Values** pane should look as shown. You have specified the speed and torque values you want for each run, and also the minimum air/fuel ratio (AFR) at each N, TQ point.



Fixed Variable Values

Vector display format: Expanded vertically

Variable:		measrpm	tq_desired	afr_min
Length:		1	1	1
1	1	2200	1263	25.5
2	1	2200	947	27.75
3	1	2200	632	30
4	1	2200	126	0
5	1	1600	1550	22
6	1	1600	1163	22.5
7	1	1600	775	23

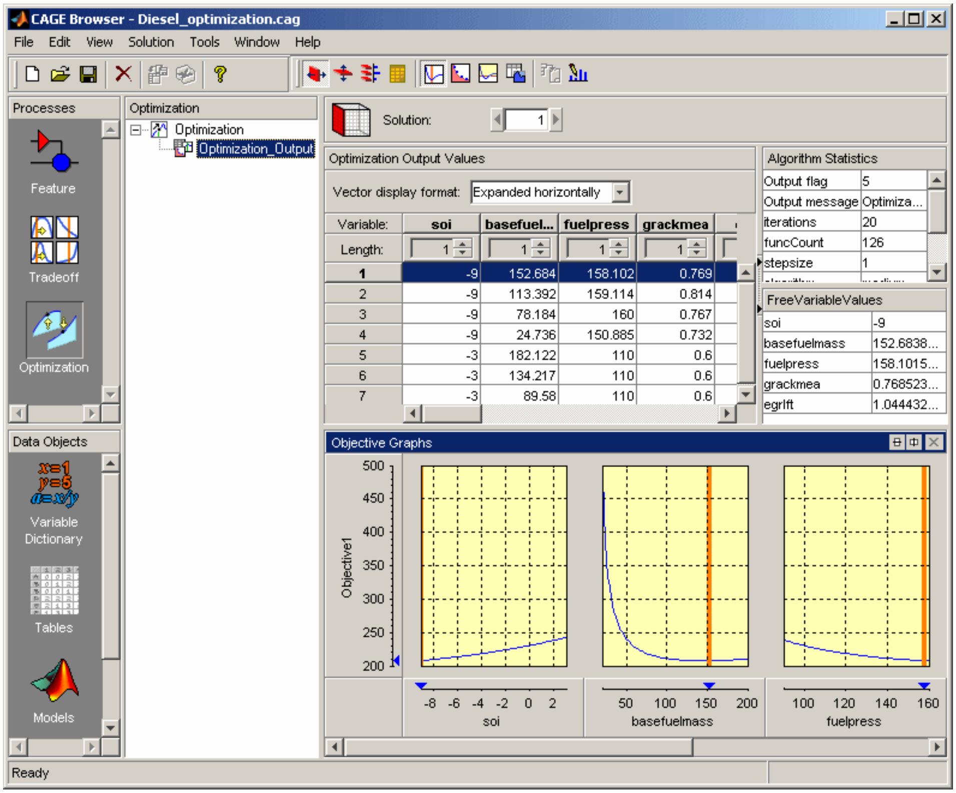
If you wished to constrain the free variables more than the range defined in the variable dictionary, you could select **Optimization > Edit Free Variable Ranges**. Leave the defaults for this problem.

Running the Optimization

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

1 Click Run Optimization in the toolbar. The optimization results are calculated.

2 When the optimization is complete, the view switches to the new child node, `Optimization_Output`, in the Optimization tree under the `Optimization` node. View the results.



Optimization Output Values

Vector display format: Expanded horizontally

Variable:	soi	basefuel...	fuelpress	grackmea
Length:	1	1	1	1
1	-9	152.684	158.102	0.769
2	-9	113.392	159.114	0.814
3	-9	78.184	160	0.767
4	-9	24.736	150.885	0.732
5	-3	182.122	110	0.6
6	-3	134.217	110	0.6
7	-3	89.58	110	0.6

Algorithm Statistics

Output flag	5
Output message	Optimiza...
iterations	20
funcCount	126
stepsize	1

FreeVariableValues

soi	-9
basefuelmass	152.6838...
fuelpress	158.1015...
grackmea	0.768523...
egrlft	1.044432...

3 Look through the solutions at different operating points by clicking cells in the output table. Compare with the gasoline example, where regions

that do not meet the constraint are yellow in the graphs — in this case the equality constraints cause all the graphs to be yellow, because there is only a solution at a single point.

4 Select **View > Display Constraints** and scroll through the lower graphs to see each constraint in relation to the solutions.

Filling Tables with Optimization Results

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

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

   The Table Filling Wizard appears.

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

   - `soi_best`
   - `basefuelmass_best`
   - `fuelpress_best`
   - `grackmea_best`
   - `egrlft_best`

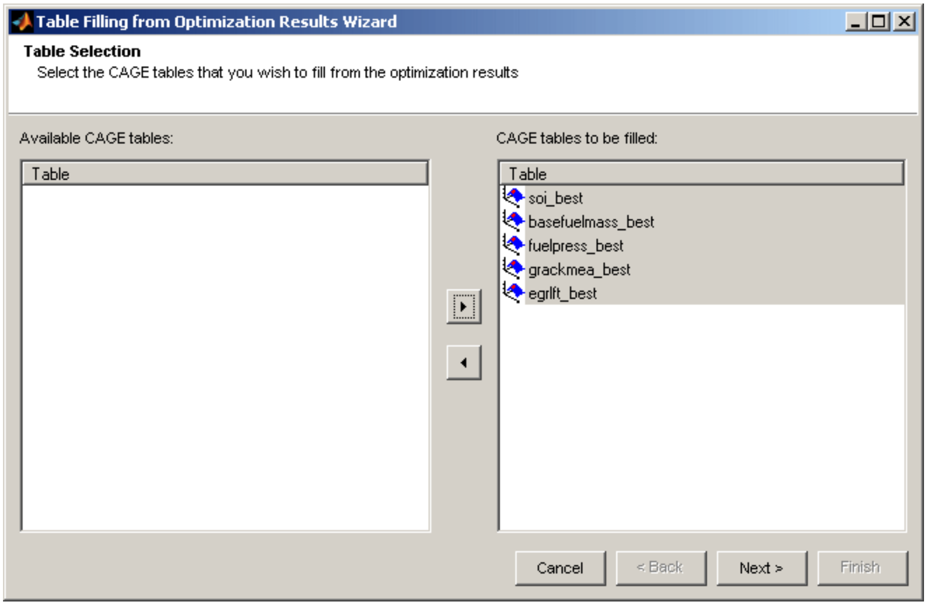


Table Filling from Optimization Results Wizard

Table Selection

Select the CAGE tables that you wish to fill from the optimization results

Available CAGE tables:

Table

CAGE tables to be filled:

Table
soi_best
basefuelmass_best
fuelpress_best
grackmea_best
egrlft_best

Click **Next**.

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

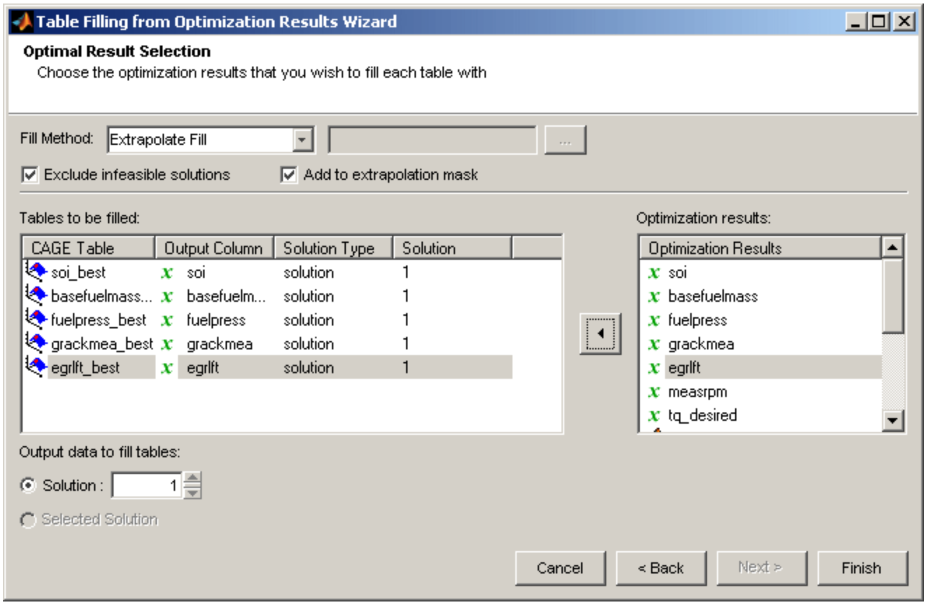


Table Filling from Optimization Results Wizard

Optimal Result Selection

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

Fill Method: Extrapolate Fill

☑ Exclude infeasible solutions ☑ Add to extrapolation mask

Tables to be filled:

CAGE Table	Output Column	Solution Type	Solution
soi_best	soi	solution	1
basefuelmass...	basefuelm...	solution	1
fuelpress_best	fuelpress	solution	1
grackmea_best	grackmea	solution	1
egrlft_best	egrlft	solution	1

Optimization results:

Optimization Results
soi
basefuelmass
fuelpress
grackmea
egrlft
measrpm
tq_desired

Output data to fill tables:

● Solution : 1

○ Selected Solution

4 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**.

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

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

Command-Line Interface to the Model-Based Calibration Toolbox

This section contains the following topics:

Introduction to the Command-Line Interface

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

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

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

Processes You Can Automate

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

1. Create or load a project — CreateProject; Load

2. Extract the points from an existing design that define data points to collect on the test bed — GetDesignMatrix

3. Create or load a data object for the project and make it editable — CreateData; BeginEdit

4. Load data from a file or the workspace — ImportFromFile; ImportFromMBCDataStructure

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

5. Work with the data:

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

6. Save your changes to the data, or discard them — CommitEdit; RollbackEdit

7. Create a new test plan for the project using a template set up in the Model Browser — CreateTestplan

8 Designate which project data object to use for modeling in the new test plan — AttachData

9 Create models for the data; these can be one- or two-stage models and can include datum models — CreateResponse

10 Work with your models:

- Examine input data and response data — DoubleInputData; DoubleResponseData
- Examine predicted values at specified inputs — PredictedValue; PredictedValueForTest
- Examine Predicted Error Variance (PEV) at specified inputs — PEV; PEVForTest
- Examine and remove outliers — OutlierIndices; OutlierIndicesForTest; RemoveOutliers; RemoveOutliersForTest
- Create a selection of alternative models — CreateAlternativeModels
- Choose the best model by using the diagnostic statistics —- AlternativeModelStatistics; DiagnosticStatistics; SummaryStatistics
- Extract a model object from any response object (Response), then:
  - Fit to new data (Fit)
  - Change model type and settings (ModelDialog)
  - Include and exclude terms to improve the model (StepwiseRegression)
  - Examine regression matrices and coefficient values (Jacobian; ParameterStatistics)
  - If you change the model you need to use UpdateResponse to replace the new model back into the response in the project.

- For two-stage test plans, once you are satisfied with the fit of the local and response feature models (and have selected best models from any alternatives you created), you can calculate the two-stage model — MakeHierarchicalResponse.
- Now you can also examine the predicted values and PEV of the two-stage model.

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

This overview is not an exhaustive list of the commands available. For that, see “Commands – Categorical List” and “Commands — Alphabetical List” in the Model-Based Calibration Toolbox Reference documentation.

Understanding Model Structure

To use the Model Browser in the Model-Based Calibration Toolbox, you must understand the structure and functions of the model tree to navigate the views. To use the command-line version of the Toolbox, you must understand the same structure and functions, that is, how projects, test plans, and models fit together. The following describes the relationship between the different models that you can construct. The diagrams in the following section, “How the Model Tree Relates to Command-Line Objects” on page 4-8, illustrate these relationships.

Projects

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

Test Plans

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

Responses

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

- Hierarchical Response (Level 0)

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

  A hierarchical response has one or more different local responses (accessible via the property `LocalResponses`) that provide different possible models of the `ResponseSignalName`. One of these must be chosen as the best, and that will then be the local response used subsequently. The response features of each of the local responses are available directly from those local response objects.

- Local Response (Level 1)

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

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

- Response (Level 1 or 2)

  - For two-stage test plans, response objects model the response features of local responses (`ResponseSignalName` corresponds to the name of the response feature). In this case, the response has a level value of 2.
  - For one-stage test plans, response objects simply model the `ResponseSignalName` as a function of the input factors. In this case, the response will have a level value of 1.

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

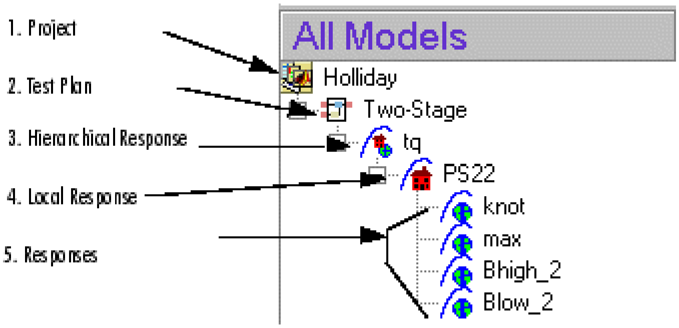
See the illustrations in the following section, “How the Model Tree Relates to Command-Line Objects” on page 4-8, for examples of different responses and how they relate to each other.

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

How the Model Tree Relates to Command-Line Objects

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

The following is an example of a two-stage model tree.



The elements of the tree correspond to the following objects in the command-line interface:

1 Project

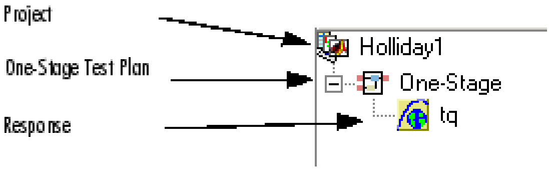
2 Test Plan

3 Hierarchical Response

4 Local Response

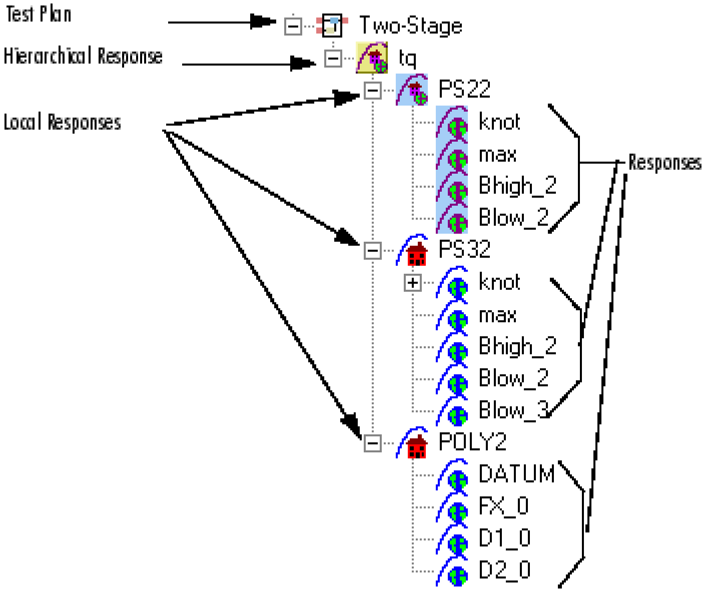
5 Responses

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

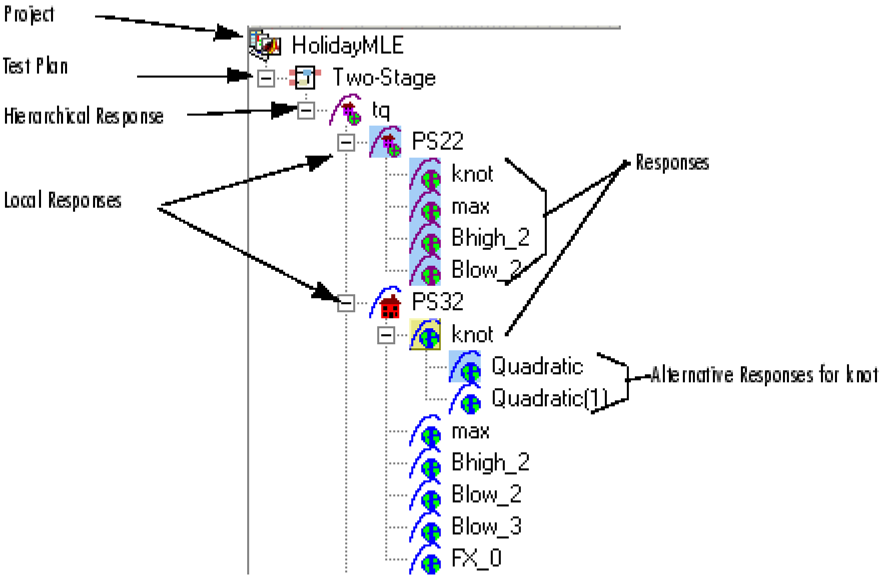


Hierarchical responses can have multiple local responses, as shown in the following example from the Model Browser. In the command-line interface these are accessible via the property LocalResponses for a hierarchical response object (`mbcmodel.hierarchicalresponse`). In this example, the local responses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

Only one of these local responses can be chosen as best (in this example, `PS22`, indicated by the blue icon) and used to construct the hierarchical response, together with the associated response features of the local response. Each local response object has a set of responses, accessible by the property ResponseFeatures.

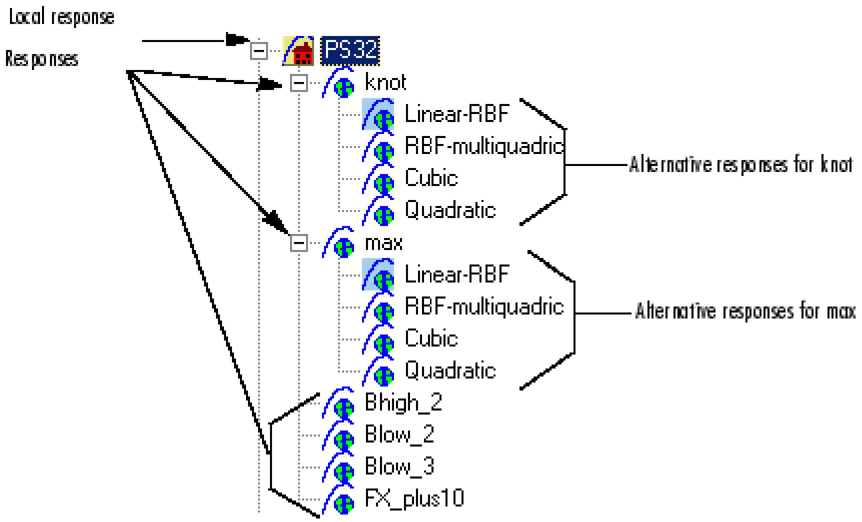


Responses can have zero or more alternative responses, as shown in the following model tree. You call the method CreateAlternativeModels on the command line to do the same.



In this example, the alternative responses for the `knot` response are accessible via the property AlternativeResponses. You can create alternative responses for any response (including all one-stage responses).

You can use model templates to try alternative model types for several responses. The following example shows the results of using a model template for four alternative responses (`Linear-RBF`, `RBF-multiquadric`, `Cubic`, and `Quadratic`). The model template has been used to create alternative responses for the responses `knot` and `max`. You can call the method CreateAlternativeModels to do this in the command-line interface.



One of the alternative responses must be chosen as best for each response (call the method ChooseAsBest). In this example, when `Linear-RBF` is chosen as best from the alternatives for the `knot` response, then it is copied to `knot`.

Worked Example

We provide an example M-file (commandlineMBCExample) demonstrating the use of the command-line interface to construct, examine, and export a two-stage model using the Holliday data (used for the Model Browser tutorials). To examine the file, change directory to the location of commandlineMBCExample.m, then type the following at the command line:

```
edit commandlineMBCExample
```

This example creates a project, loads and filters some data, and sets up a two-stage model test plan. Potential outliers are identified using the studentised residual statistic. Those tests with potential outliers are plotted for the user to decide whether to remove the outlier, remove the test, or keep all points. The resulting response feature models are also plotted where potential outliers are found, when the user can decide whether to remove them. Then the two stage model is constructed, and the models for torque and MBT (the spark angle for maximum brake torque) are exported to file.

To run the example type commandlineMBCExample at the command line.

5

Tutorial: Model Quickstart

This section discusses the following topics:

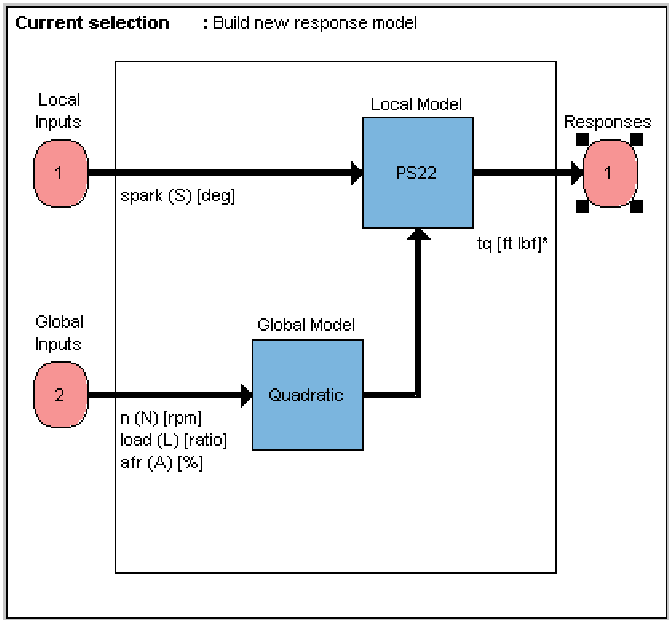
Two-Stage Models

This tutorial gives you a quick introduction to the modeling end of the Model-Based Calibration Toolbox. The instructions show you how to use the toolbox to create a statistical model of an automobile engine that predicts the torque generated by the engine as a function of spark angle and other variables. We provide example engine data and show you how to fit a statistical model to the data, examine and verify the fit, and export the model. In the normal modeling process, you would create many different models for one project and compare them to find the best solution. The tutorial also provides a quick guide to fitting and comparing multiple models.

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

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

The usual way for collecting brake torque data is to fix engine speed, load, and air/fuel ratio within each test and sweep the spark angle across a range of angles. For this experimental setup, there are two sources of variation. The first source is variation within tests when the spark angle is changed. The second source of variation is between tests when the engine speed, load, and air/fuel ratio are changed. The variation within a test is called local, and the variation between tests, global. Two-stage modeling estimates the local and global variation separately by fitting local and global models in two stages. A local model is fitted to each test independently. The results from all the local models are used to fit global models across all the global variables. Once the global models have been estimated, they can be used to estimate the local models' coefficients for any speed, load, and air/fuel ratio. The relationship between the local and global models is shown in the following block diagram, as you will see in the Model Browser.



Current selection : Build new response model

Starting the Toolbox and Loading Data

1 Double-click the MATLAB icon to start MATLAB.

2 To start the Model-Based Calibration Toolbox, enter `mbcmodel` at the command prompt in MATLAB.

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

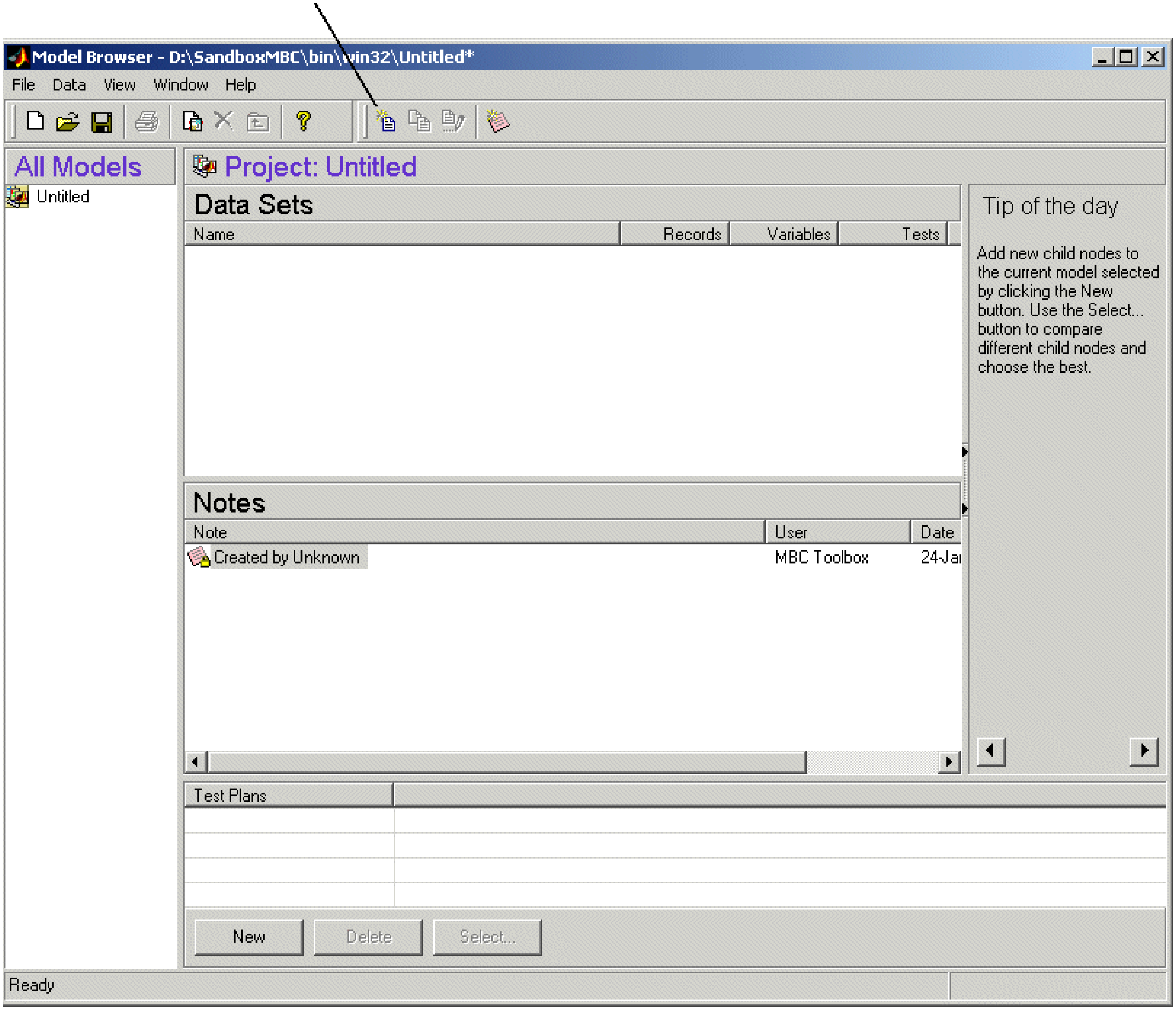
Note You can edit your user information at any time by selecting **File > Preferences**.

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

The Model Browser window appears.

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

Click here to load some new data.



Add new child nodes to the current model selected by clicking the New button. Use the Select... button to compare different child nodes and choose the best.

Note	User	Date
Created by Unknown	MBC Toolbox	24-Jai

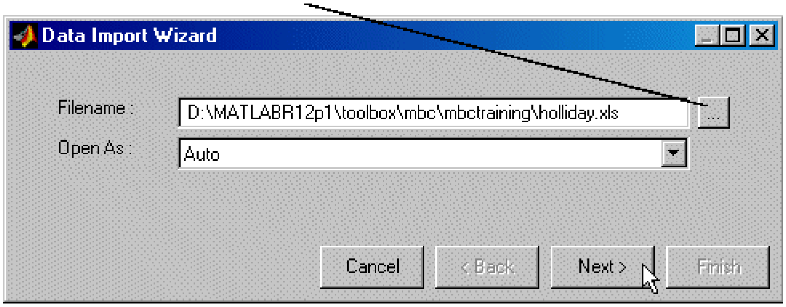
Load the example data file `holliday.xls`:

1 Click the button on the toolbar, or select **Data > New Data**.

This opens the Data Editor window.

2 Click the Import File icon on the toolbar ( ) or select **File > Import > File**.

3 Use the browse button to the right of the edit box in the Data Import Wizard to open a file browser and find the file `holliday.xls` in the `mbctraining` directory. Click **Open** or double-click the file.



Click here to browse for a file.

Filename : D:\MATLABR12p1\toolbox\mbc\mbctraining\holliday.xls

Open As : Auto

4 The file pathname appears in the Data Import Wizard. Click **Next**.

5 A summary screen displays information about the data. Click **Finish** to close the Data Import Wizard and return to the Data Editor.

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

For more details on functionality available within the Data Editor, see "Data".

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

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

Setting Up the Model

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

Note It does not matter in which order you set up local and global models, as both must be completed before you set up the response model.

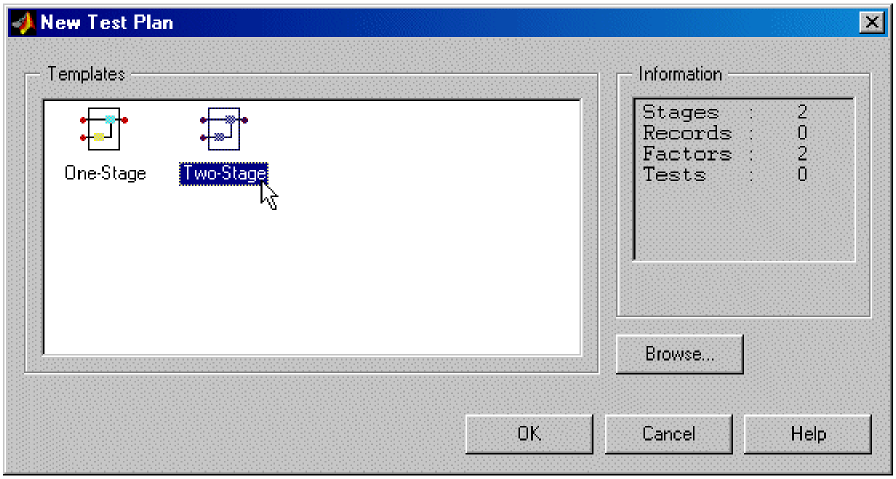
1 To create a new test plan, do one of the following:

- In the **Test Plans** list pane at the bottom, click **New**.

  Alternatively, click the New Test Plan button ( ) in the toolbar. Note that this button changes depending on which node is selected in the model tree. It always creates a child node (not necessarily a test plan node), as does the **New** button at the bottom.

  Or select **File > New Test Plan**.

The New Test Plan dialog box appears.



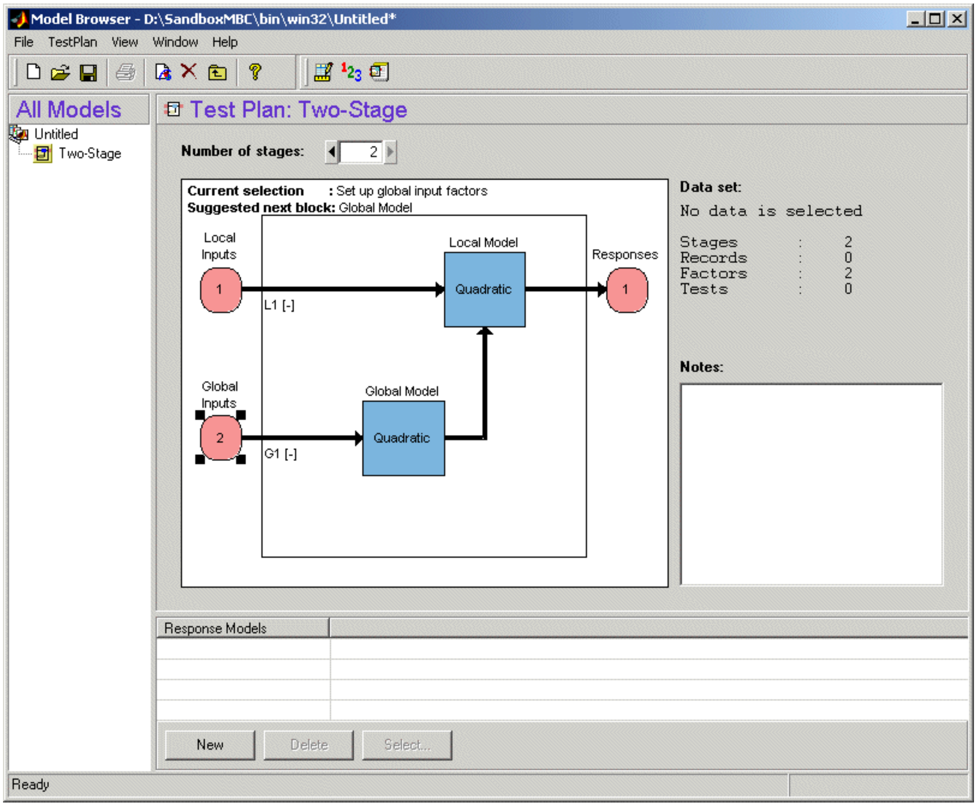
```
Stages   :   2
Records  :   0
Factors  :   2
Tests    :   0
```

2 Click the two-stage test plan icon and click **OK**.

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

3 Highlight this node of the tree , `Two-Stage`, by clicking it. The Model Browser window displays a diagram representing the two-stage model.

See also “Test Plan Level”.



Number of stages: 2

Current selection : Set up global input factors
Suggested next block: Global Model

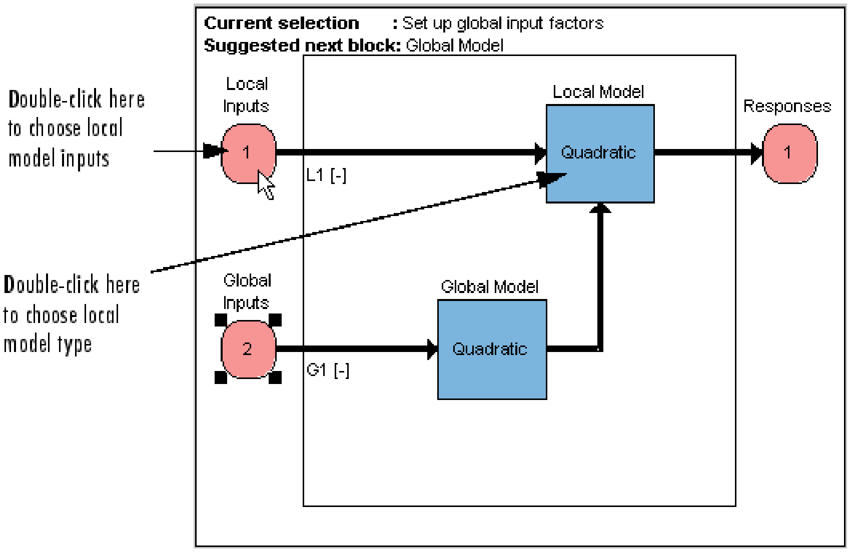
Data set:

```
No data is selected

Stages    :   2
Records   :   0
Factors   :   2
Tests     :   0
```

Notes:

Setting Up the Local Model

Setting up the local model requires that you specify the model’s inputs and type.



Double-click here to choose local model inputs

Double-click here to choose local model type

Current selection : Set up global input factors
Suggested next block: Global Model

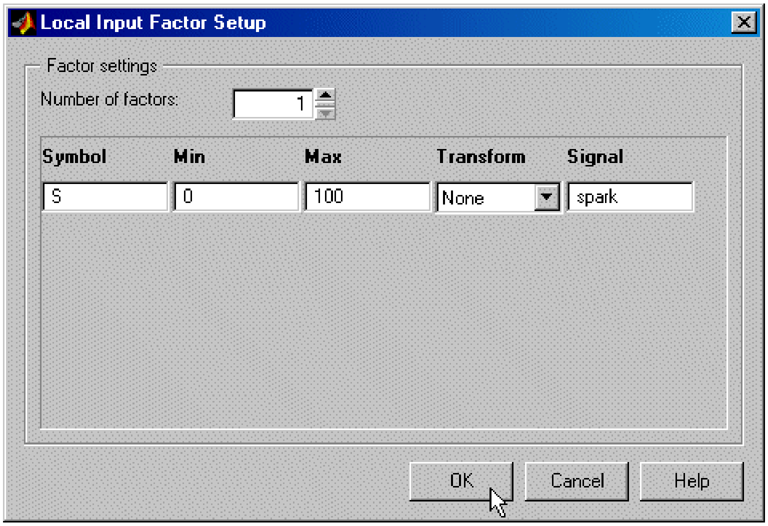
Specifying the Local Model Input

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

To specify spark angle as the input,

1 Double-click (or right-click) the Local Inputs icon on the model diagram to specify the local model input.

   The Local Input Factor Setup dialog box appears.



Factor settings

Number of factors: 1

Symbol	Min	Max	Transform	Signal
S	0	100	None	spark

a Set **Symbol** to `S`.

b Set **Signal** to `spark`. This is optional and matches the raw data.

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

Notice that the new name of the local model input now appears on the two-stage model diagram.

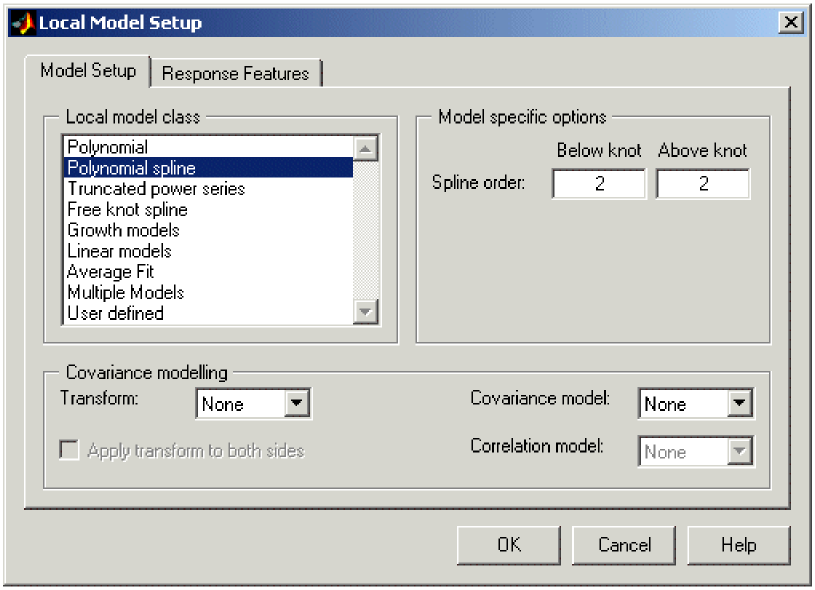
Specifying the Local Model Type

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

To specify polyspline as the model type,

1 Double-click the local model icon in the model diagram.

The Local Model Setup dialog box appears.



Below knot Above knot

Spline order: 2 2

Transform: None

□ Apply transform to both sides

Covariance model: None

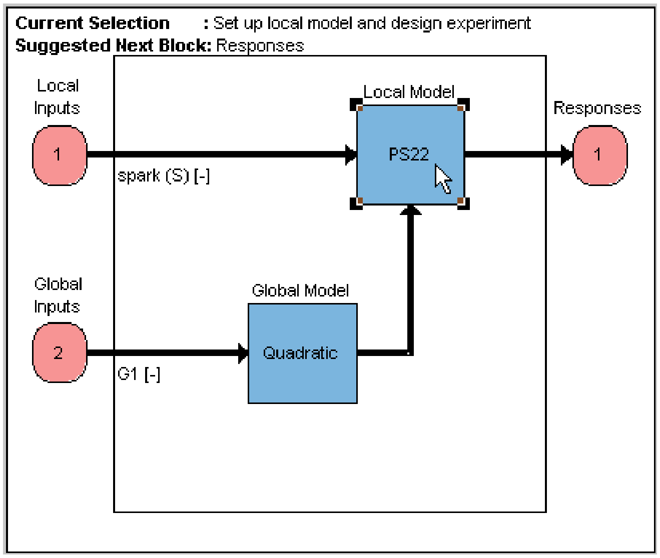
Correlation model: None

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

b Set **Spline Order** to `2` below and `2` above knot.

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

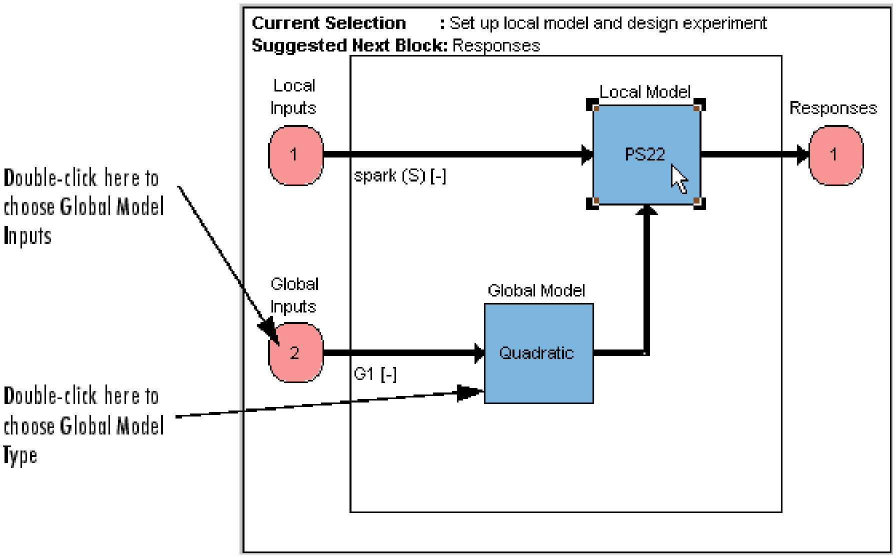
Notice that the new name of the local model class, `PS` (for polyspline) `2,2` (for spline order above and below knot) now appears on the two-stage model diagram.



Current Selection : Set up local model and design experiment
Suggested Next Block: Responses

Setting Up the Global Model

Setting up the global model is similar to setting up the local model. You must specify the model (or curve) type and the inputs used to create the model.



Current Selection : Set up local model and design experiment
Suggested Next Block: Responses

Double-click here to choose Global Model Inputs

Double-click here to choose Global Model Type

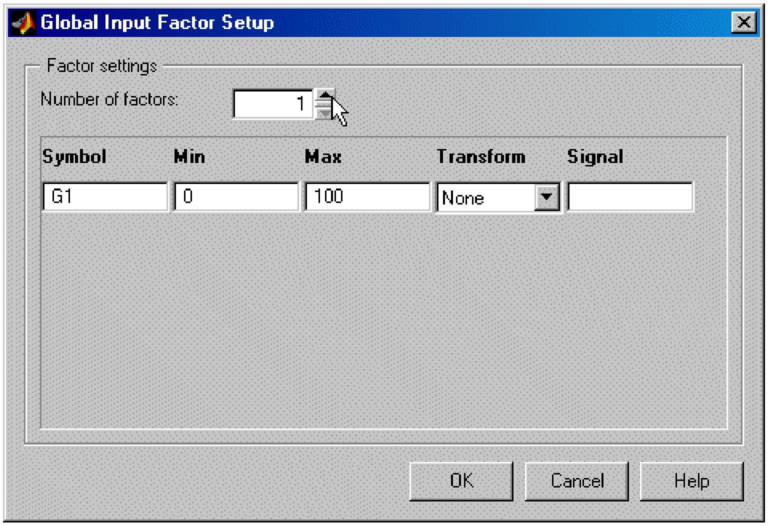
Specifying the Global Model Inputs

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

To specify these inputs,

1 Double-click the Global Inputs icon on the model diagram.

   The Global Input Factor Setup dialog box appears.



Global Input Factor Setup

Factor settings

Number of factors: 1

Symbol	Min	Max	Transform	Signal
G1	0	100	None	

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

a Click the up arrow button indicated by the cursor above to increase the number of factors to three.

b Edit the three factors to create the engine model input. In each case, change the symbols and signals to the following:

Symbol	Signal
N	`n`
L	`load`
A	`afr`

c Leave the Min and Max boxes at the defaults (you fill them during the data selection process). You might want to set factor ranges at this stage if you were designing an experiment, but in this case there is already data available, so you use the actual range of the data to model instead.

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

Specifying the Global Model Type

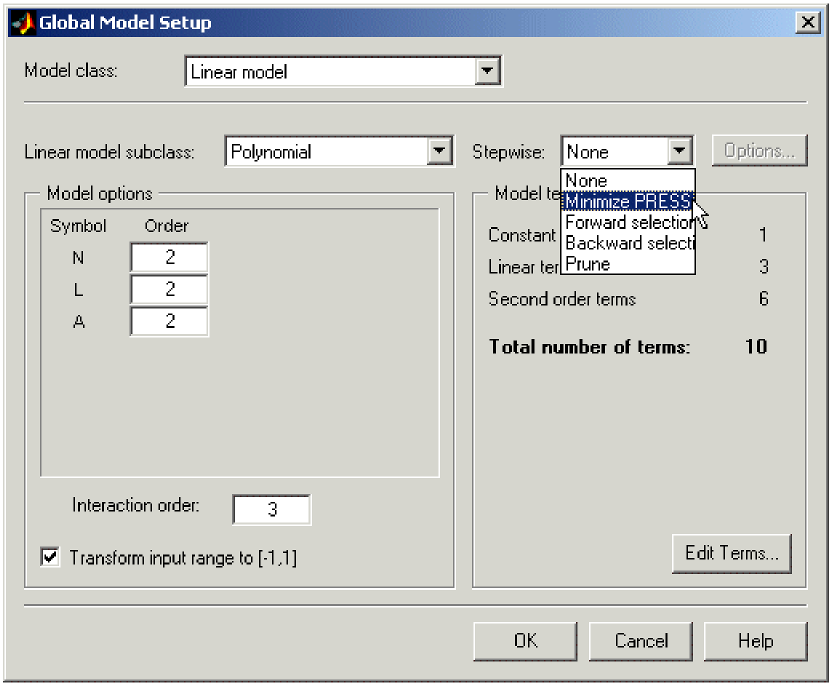
Fitting the local model finds values for each model coefficient or response feature (for example, `knot`) for each test. These coefficients then become the data to which you fit the global model.

By default, quadratic polynomials are used to build the global model for each response feature. In this case you use the default.

To specify quadratic curves as the global model curve fitting method,

1 Double-click the icon representing the global model in the two-stage model diagram.

   The Global Model Setup dialog box appears.



Global Model Setup

Model class: Linear model

Linear model subclass: Polynomial

Stepwise: None

Model options

Symbol	Order
N	2
L	2
A	2

Interaction order: 3

☑ Transform input range to [-1,1]

Model te[rms]

Constant	1
Linear ter[ms]	3
Second order terms	6
Total number of terms:	**10**

a `Polynomial` should already be selected from the **Linear Model Subclass** pop-up menu. Under **Model options**, the order for the three variables `N`, `L`, and `A` is set by default to `2`, which is required.

b Set **Stepwise** to `Minimize PRESS` (PREdicted Sum Square error).

2 Click **OK** to accept the settings and dismiss the Model Settings dialog box.

You use the Stepwise feature to avoid overfitting the data; that is, you do not want to use unnecessarily complex models that “chase points” in an attempt to model random effects. Predicted error sum of squares (PRESS) is a measure of the predictive quality of a model. `Minimize PRESS` throws away terms in the model to improve its predictive quality, removing those terms that reduce the PRESS of the model.

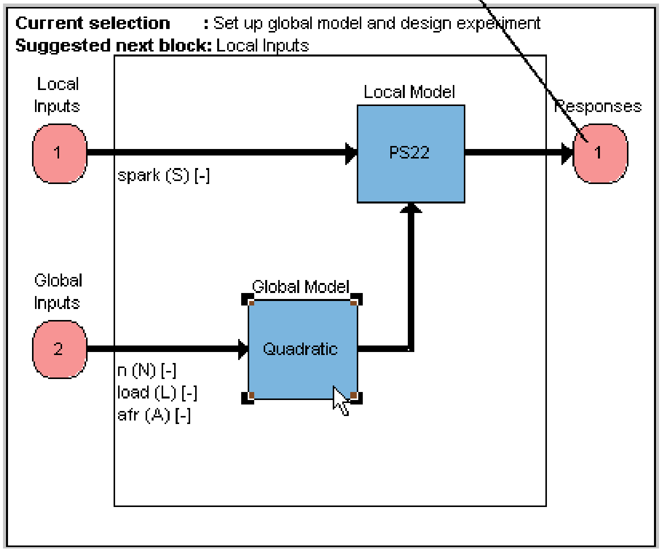
This completes the setup of the global model.

Selecting Data

The model you have set up now needs data:

1 Double-click the Responses block in the Test Plan diagram. As no data has yet been selected for this test plan, this launches the Data Wizard.

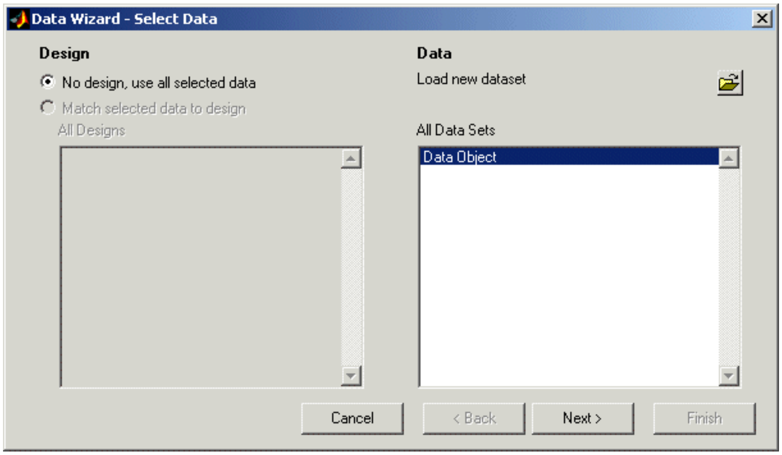
Launch Data Wizard by double-clicking here



Current selection : Set up global model and design experiment
Suggested next block: Local Inputs

For the same result, you could also click the Select Data button 123 in the toolbar of the Model Browser (or **TestPlan > Select Data** menu item). Also, if you did not already load a data set at the project node, you can do it at this point using **TestPlan > Load New Data**.

The Data Wizard dialog appears.



Data Wizard - Select Data

Design

● No design, use all selected data

○ Match selected data to design

All Designs

Data

Load new dataset

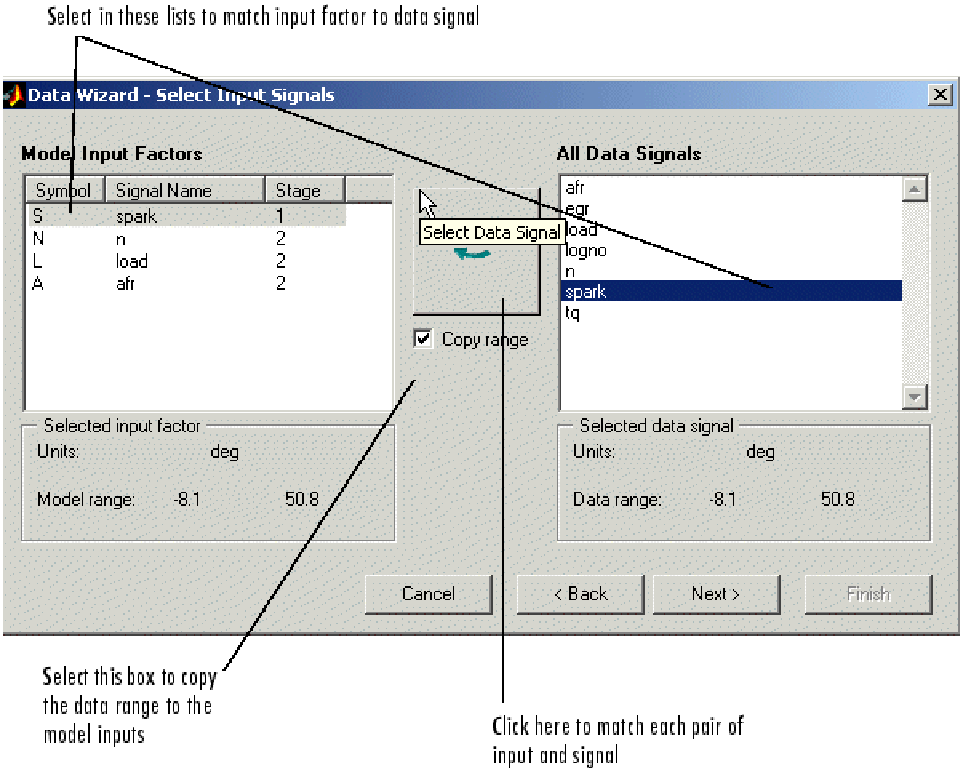
All Data Sets

Data Object

2 `Data Object` is already selected by default. Click **Next**.

3 Select `S` in the **Model Input Factors** box and `Spark` under **All Data Signals**.

4 Select the **Copy Range** check box, as shown. This makes the model use the range in the data for that factor.



Select in these lists to match input factor to data signal

Data Wizard - Select Input Signals

Model Input Factors

Symbol	Signal Name	Stage
S	spark	1
N	n	2
L	load	2
A	afr	2

All Data Signals

afr
egr
load
logno
n
spark
tq

Select Data Signal

☑ Copy range

Selected input factor
Units: deg
Model range: -8.1 50.8

Selected data signal
Units: deg
Data range: -8.1 50.8

Select this box to copy the data range to the model inputs

Click here to match each pair of input and signal

5 Click the large **Select Data Signal** button, as indicated above.

6 Repeat this process and match the correct data signals to the other three input factors, `N`, `L`, and `A` (from `n`, `load`, and `afr`).

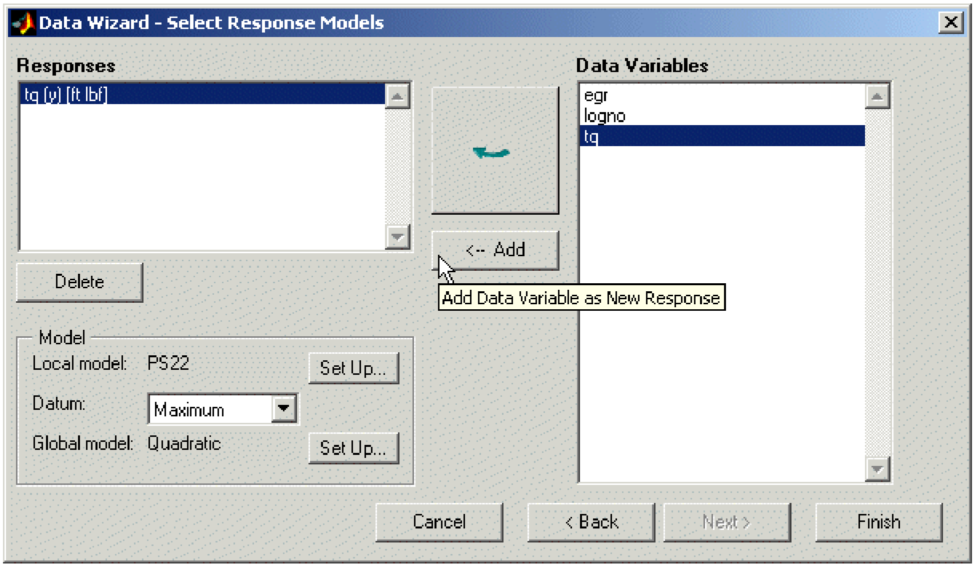
If the signal name entered during the input factor setup matches a signal name in the data set, the wizard automatically selects the correct signal when the input factor is selected. If the name is not correct, you must select the correct signal manually by clicking. This autoselect facility can save time if the data set has a large number of signals.

7 When you have matched all four input factors to the correct data signals (for both stages of the two-stage model), click **Next**.

Specifying the Response Model

The model you just set up now needs a response specified (that is, the factor you want the model to predict, in this case, torque).

The next screen of the Data Wizard is for selecting response models.



Responses: tq (y) [ft lbf]

Data Variables: egr, logno, tq

Local model: PS22

Datum: Maximum

Global model: Quadratic

1 Select `tq` (torque) as the response.

2 Click **Add**. Torque appears in the **Responses** list.

3 Select `Maximum` under **Datum**.

   Only certain model types with a clearly defined maximum or minimum can support datum models. See “Datum Models”.

4 Click **Next**. The Set Tolerances screen appears. This is for matching data to designs. As there is no design in this case, you can ignore this screen and click **Finish**.

   The Data Editor appears so you can inspect the data you have selected for modeling.

5 To use all the data, close the Data Editor.

   A dialog appears with the message that the torque response model will be built and the selected data will be added to the set of Actual Design points, followed by `Do you want to make these changes?`.

6 Click **Yes** to accept the data and create the models.

   The Model-Based Calibration Toolbox now calculates local and global models using the test plan models you just set up.

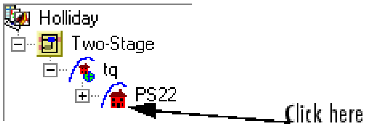
Notice that torque appears on the two-stage model diagram, and a new node appears on the tree in the **All Models** pane, called `PS22`.

Verifying the Model

Verifying the Local Model

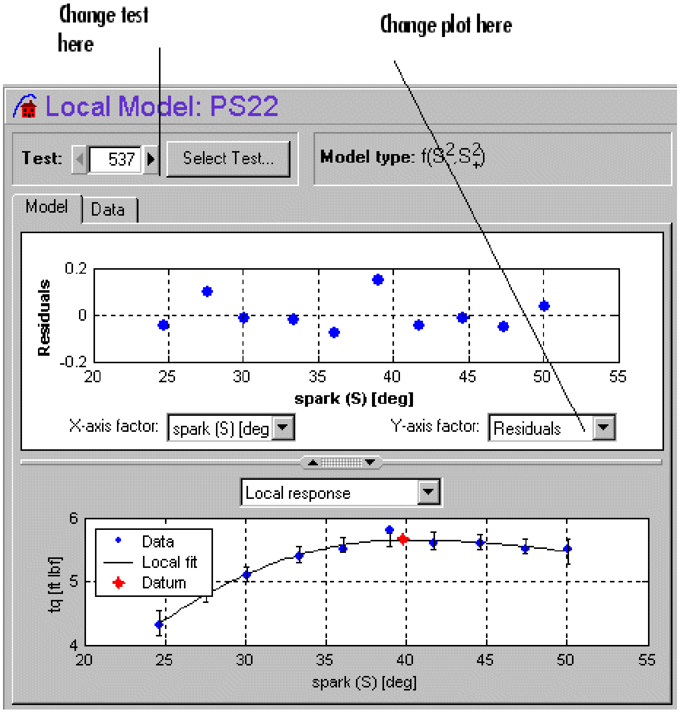
The first step is to check that the local models agree well with the data:

1 Select PS22 (the local node) on the Model Browser tree.



Click here

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


Change test here

Change plot here

Local Model: PS22

Test: 537

Model type: $f(S^2,S_+^2)$

X-axis factor: spark (S) [deg

Y-axis factor: Residuals

The lower plot shows the data being fitted by the model (blue dots) and the model itself (line). The red spot shows the position of the polyspline knot, at the datum (maximum) point.

2 In the upper scatter plot pane, click the **Y-axis factor** pop-up menu and select `Studentized residuals`.

3 To display plots and statistics for the other test data, scroll through the tests using the **Test** arrows at the top left, or by using the **Select Test** button.

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

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

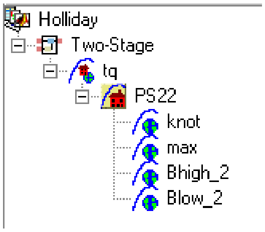
Both plots have right-click pop-up menus offering various options such as removing and restoring outliers and confidence intervals. Clicking any data point marks it in red as an outlier.

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

Verifying the Global Model

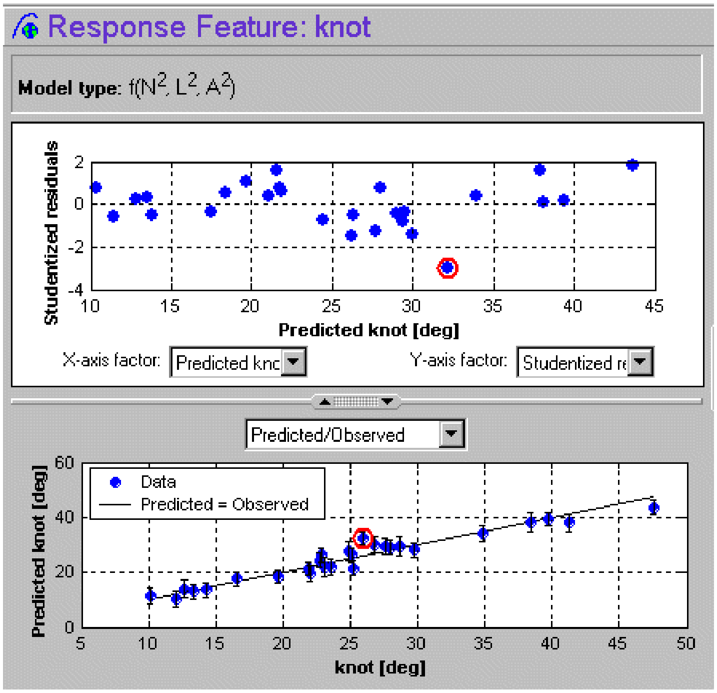
The next step is to check through the global models to see how well they fit the data:

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



2 Select the first of the global models, `knot`.

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


Response Feature: knot

Model type: $f(N^2, L^2, A^2)$

X-axis factor: Predicted knc

Y-axis factor: Studentized re

Predicted/Observed

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

Note Never remove outliers as a matter of course. However, this tutorial is designed to show you how the toolbox helps you to do this when required. The default outlier selection criterion is a studentized residual greater than 3, to bring your attention to possible outliers, but you should never remove data without good reasons. Remove enough points and the model will simply interpolate the data and become useless for prediction. You can customize the criteria for outlier selection. Use the plot of Cook's Distance to see the influence of each point on the model fit to help you decide whether to remove an outlier.

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

Return to the **Local Model** pane by clicking the local node `PS22` in the Model Browser tree.

Selecting the Two-Stage Model

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

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

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

Now you can use the model selection features to view the fit of this two-stage model in various ways, to compare it with both the data and the local model fit.

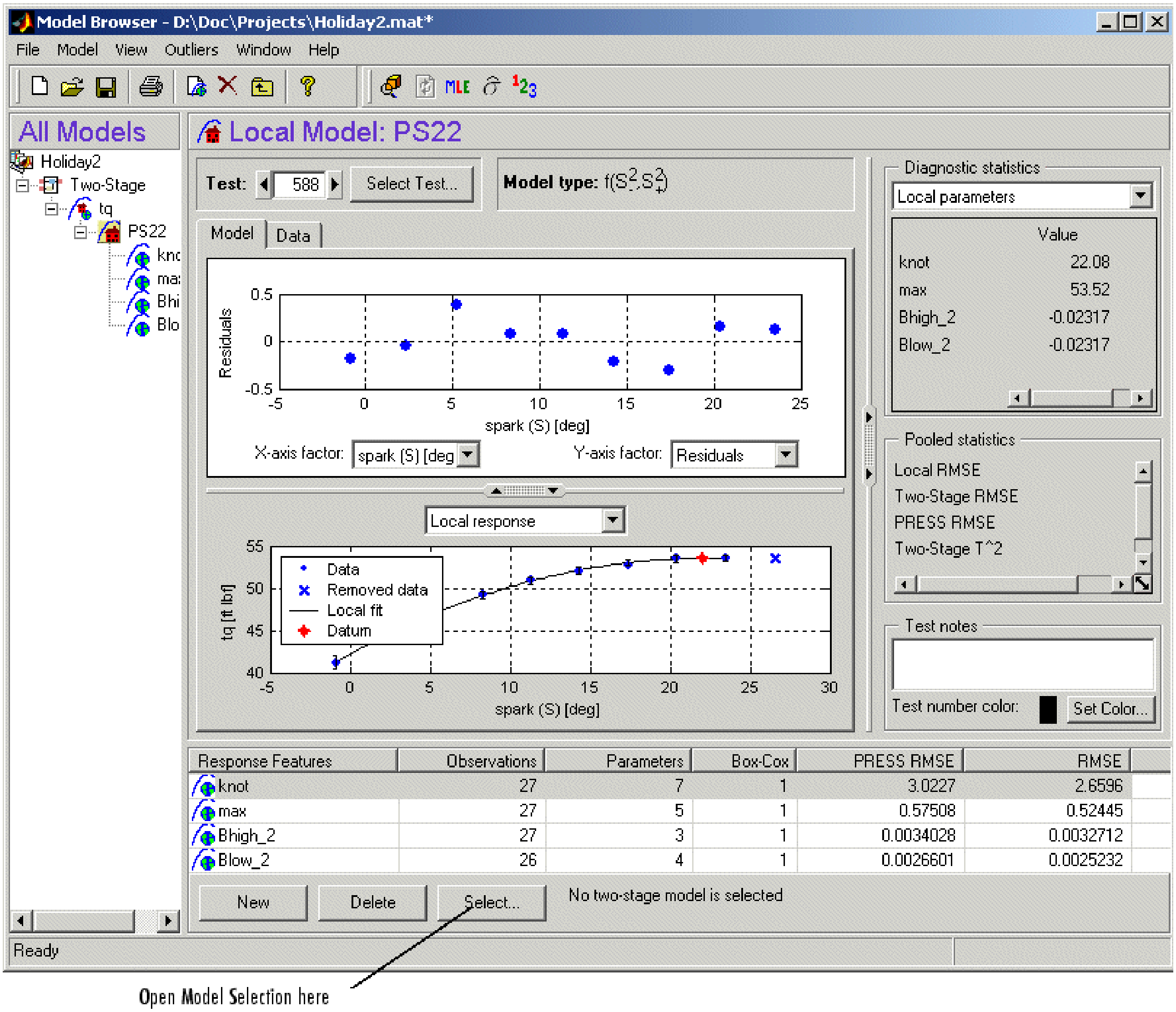
Within this tutorial, you use the following:

- "Tests View" on page 5-28
- "Response Surface View" on page 5-29

For more detailed help on all the views available in the Model Selection window, see "Selecting Models".

Note To construct a two-stage model from the local and global models, you click the local node in the model tree (with the house icon) and click the **Select** button. This is the next step in the tutorial.

Once you are satisfied with the fit of the local and global models, it is time to construct a two-stage model from them. Return to the Local Model view by clicking the local node PS22 in the Model Browser tree. The Model Browser should look like the following example.



Test: 588

Model type: $f(S^2_-, S^2_+)$

X-axis factor: spark (S) [deg

Y-axis factor: Residuals

Diagnostic statistics: Local parameters

	Value
knot	22.08
max	53.52
Bhigh_2	-0.02317
Blow_2	-0.02317

Test number color: ■

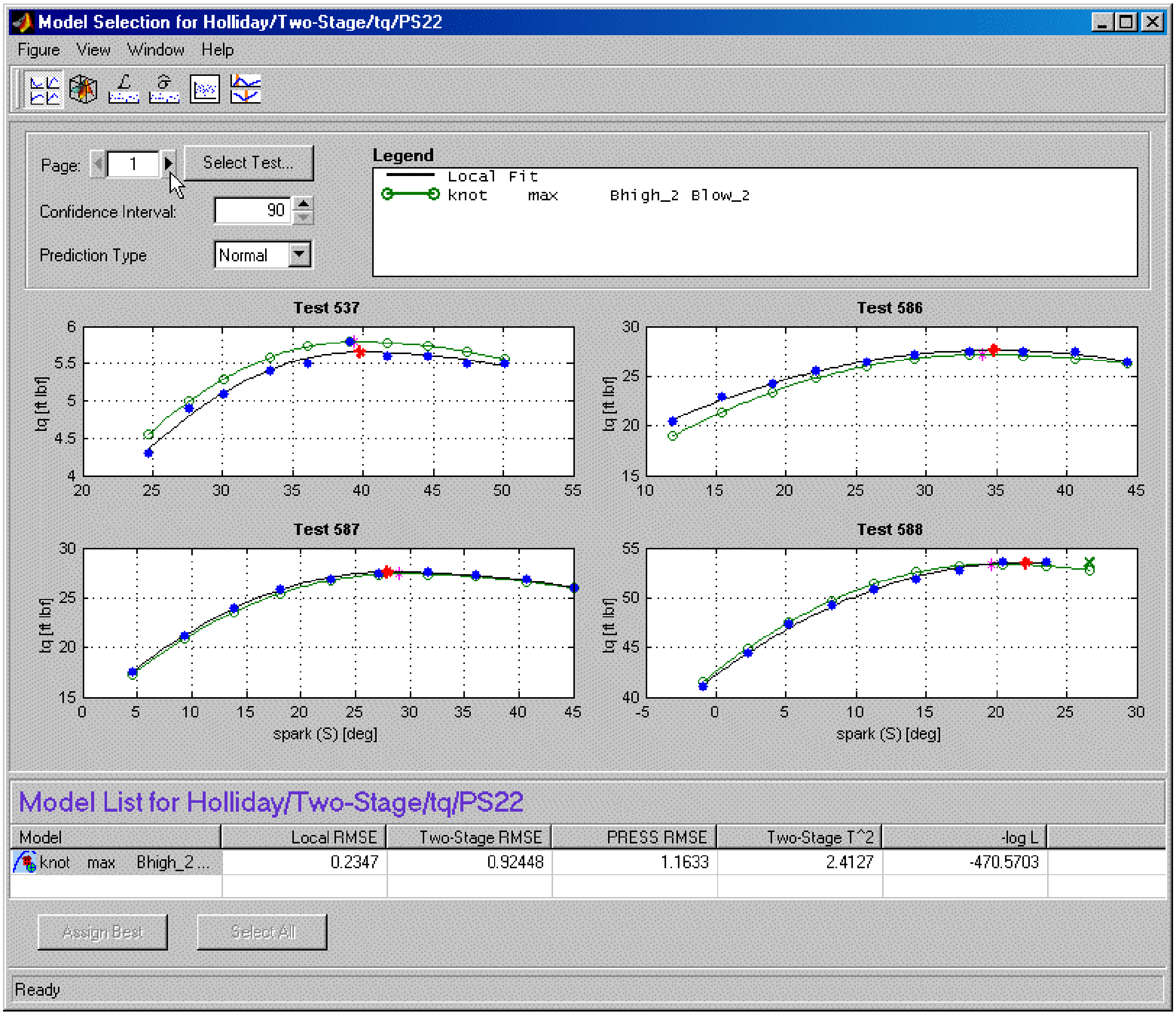
Response Features	Observations	Parameters	Box-Cox	PRESS RMSE	RMSE
knot	27	7	1	3.0227	2.6596
max	27	5	1	0.57508	0.52445
Bhigh_2	27	3	1	0.0034028	0.0032712
Blow_2	26	4	1	0.0026601	0.0025232

No two-stage model is selected

Open Model Selection here

Click **Select** in the **Response Features** list pane, and the Model Selection window appears. This window is intended to help you select a *best model* by comparing several candidate models. A number of icons in the toolbar enable you to view the fit of the model in various ways. By default, the Tests view appears. These plots show how well the two-stage model agrees with the data.

Tests View



Page: 1

Confidence Interval: 90

Prediction Type: Normal

Model List for Holliday/Two-Stage/tq/PS22

Model	Local RMSE	Two-Stage RMSE	PRESS RMSE	Two-Stage T^2	-log L
knot max Bhigh_2 ...	0.2347	0.92448	1.1633	2.4127	-470.5703

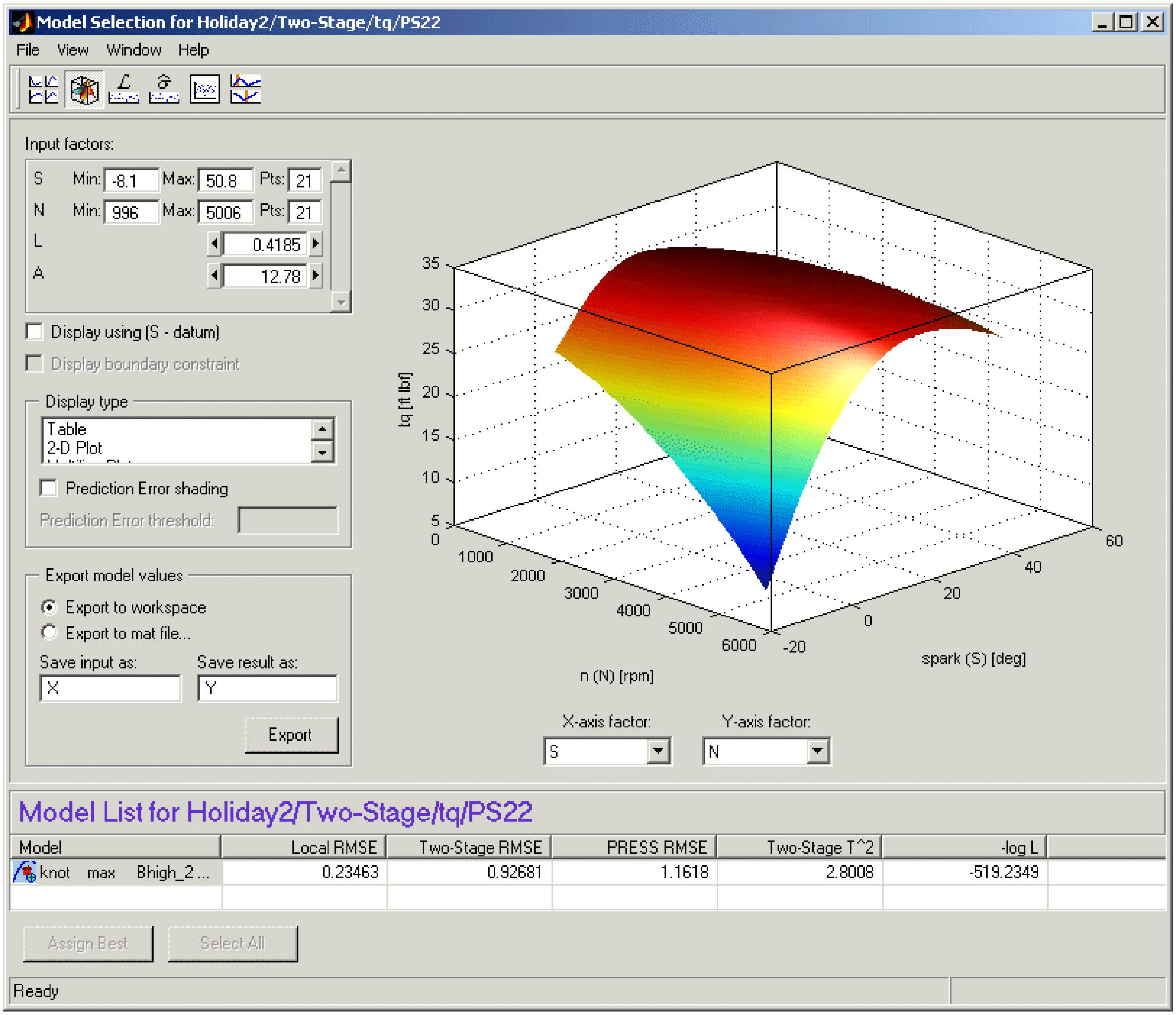
Ready

Scroll though the tests using the left/right arrows or the **Select Test** button at the top left. The plots show the fit of the two-stage model for each test (green open circles and line), compared with the fit of the local model (black line) and the data (blue dots). You can left-click (and hold) to see information on each test or zoom in on points of interest by **Shift**-click-dragging or middle-click-dragging. Double-click to return the plot to the original size.

Response Surface View

You view the model as a surface by clicking the **Response Surface** icon in the toolbar. You can rotate the plot by click-dragging it.

1. Click `Movie` in the **Display Type** list to see the surface (torque against spark and speed) vary through different values of `load`. Click **Replay** to see it again.
2. Take a look at some of the other display types.



Input factors:

S	Min:	-8.1	Max:	50.8	Pts:	21
N	Min:	996	Max:	5006	Pts:	21
L				0.4185		
A				12.78		

□ Display using (S - datum)

□ Display boundary constraint

Display type

Table
2-D Plot

□ Prediction Error shading

Prediction Error threshold:

Export model values

● Export to workspace

○ Export to mat file...

Save input as: X

Save result as: Y

X-axis factor: S

Y-axis factor: N

Model List for Holiday2/Two-Stage/tq/PS22

Model	Local RMSE	Two-Stage RMSE	PRESS RMSE	Two-Stage T^2	-log L
knot max Bhigh_2 ...	0.23463	0.92681	1.1618	2.8008	-519.2349

Ready

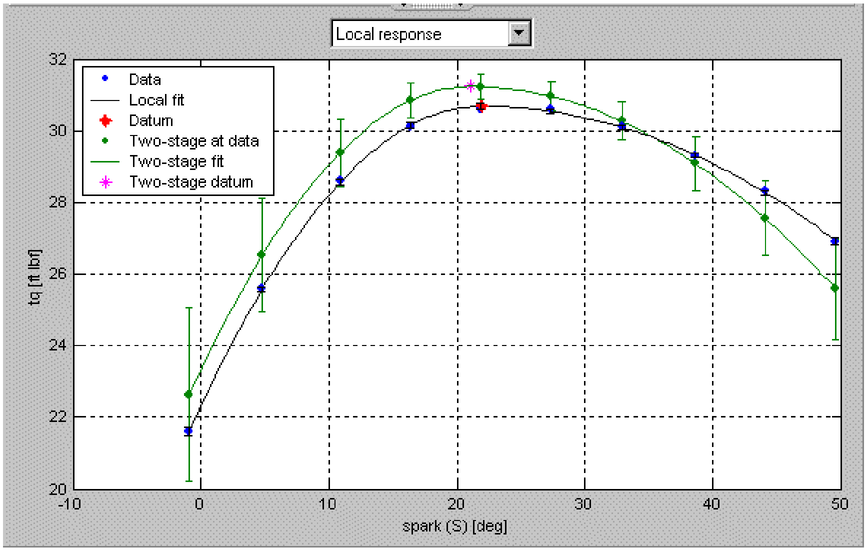
3 Dismiss the Model Selection window, and accept the best model by clicking **Yes** in the Model Selection dialog (it is the only two-stage model so far).

4 The MLE dialog appears, prompting you to calculate the maximum likelihood estimate (MLE) for the two-stage model. Click **Cancel**. You can calculate MLE later.

Comparing the Local Model and the Two-Stage Model

Now the lower plots in the **Local Model** pane show two lines fitted to the test data: the Local Model line (black), and the Two-Stage Model line (green). The plots also show the data (in blue), so you can compare how close the two-stage model fit is to both the data and the local fit for each test.

You can scroll through the various tests (using the arrows at the top left or the **Select Test** button) to compare the local and two-stage models for different tests.



Notice that the local model icon has changed (from the local icon showing a house, to a two-stage icon showing a house and a globe) to indicate that a two-stage model has been calcluated.

Click the MLE button in the toolbar to calculate the maximum likelihood estimate.

Maximum Likelihood Estimation

The global models were created in isolation without accounting for any correlations between the response features. Using MLE (maximum likelihood estimation) to fit the two-stage model takes account of possible correlations between response features. In cases where such correlations occur, using MLE significantly improves the two-stage model.

1 You reach the MLE dialog from the local node (PS22 in this case) by

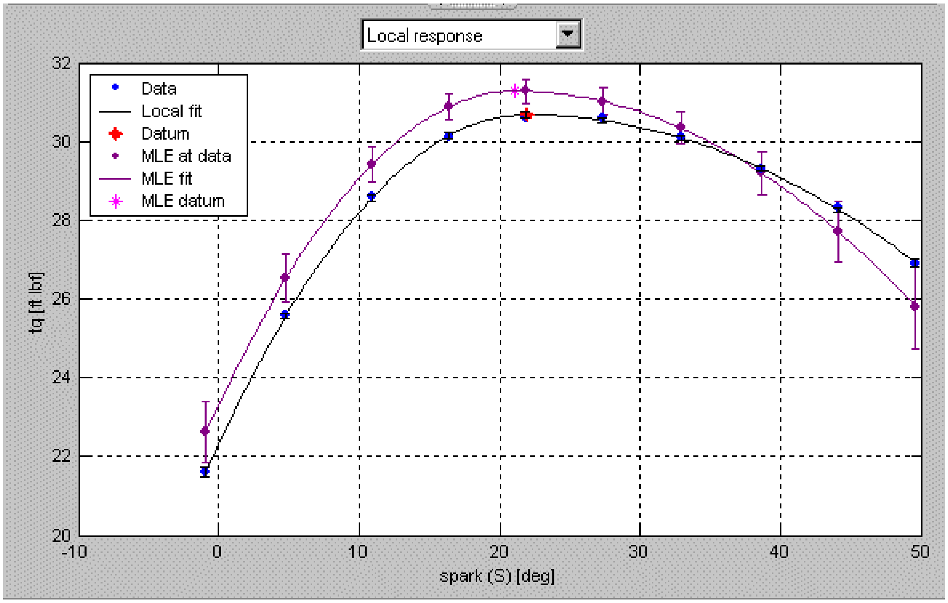
- Clicking the MLE button in the toolbar
- Or by choosing **Model > Calculate MLE**

2 Leave the algorithm default settings and click **Start** to calculate MLE.

3 Watch the progress indicators until the process finishes and a two-stage RMSE (root mean square error) value appears.

4 Click **OK** to leave the MLE dialog.

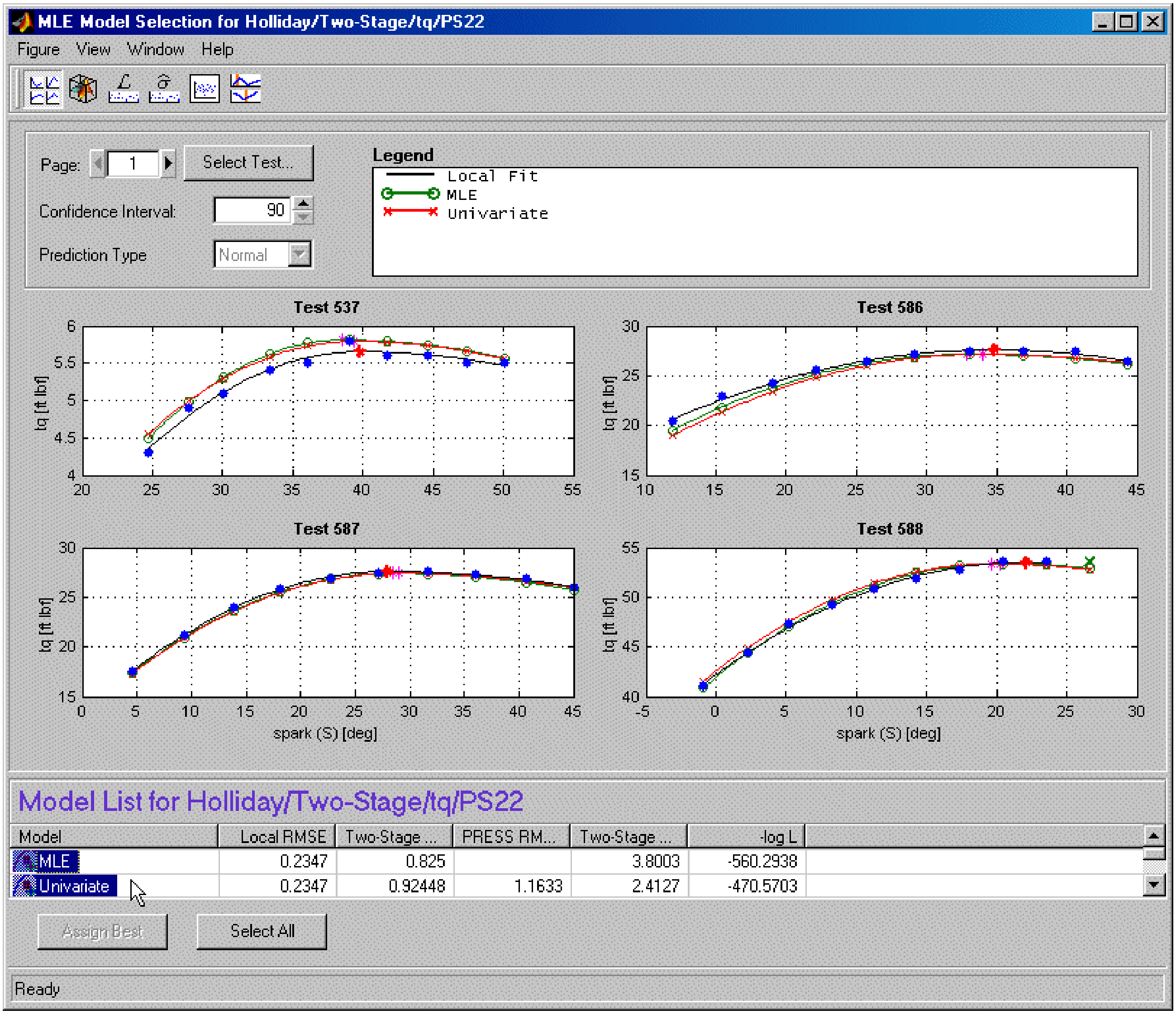
Now the plots on the **Local Model** pane all show the two-stage model in purple to indicate that it is an MLE model. This is also indicated in the legend. Notice that all the model icons in the tree (the response, the local model, and the response features) have also changed to purple to indicate that they are MLE models.



5 Click the **Select** button. This takes you to the Model Selection window.

Here you can compare MLE with the univariate model previously constructed (without correlations). By default, the local fit is plotted against the MLE model.

6 Select both MLE and the Univariate model for plotting by holding down **Shift** while you click the Univariate model in the **Model List** at the bottom of the view.



Page: 1

Confidence Interval: 90

Prediction Type: Normal

Model List for Holliday/Two-Stage/tq/PS22

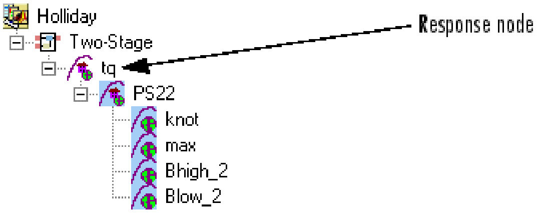
Model	Local RMSE	Two-Stage ...	PRESS RM...	Two-Stage ...	-log L
MLE	0.2347	0.825		3.8003	-560.2938
Univariate	0.2347	0.92448	1.1633	2.4127	-470.5703

Ready

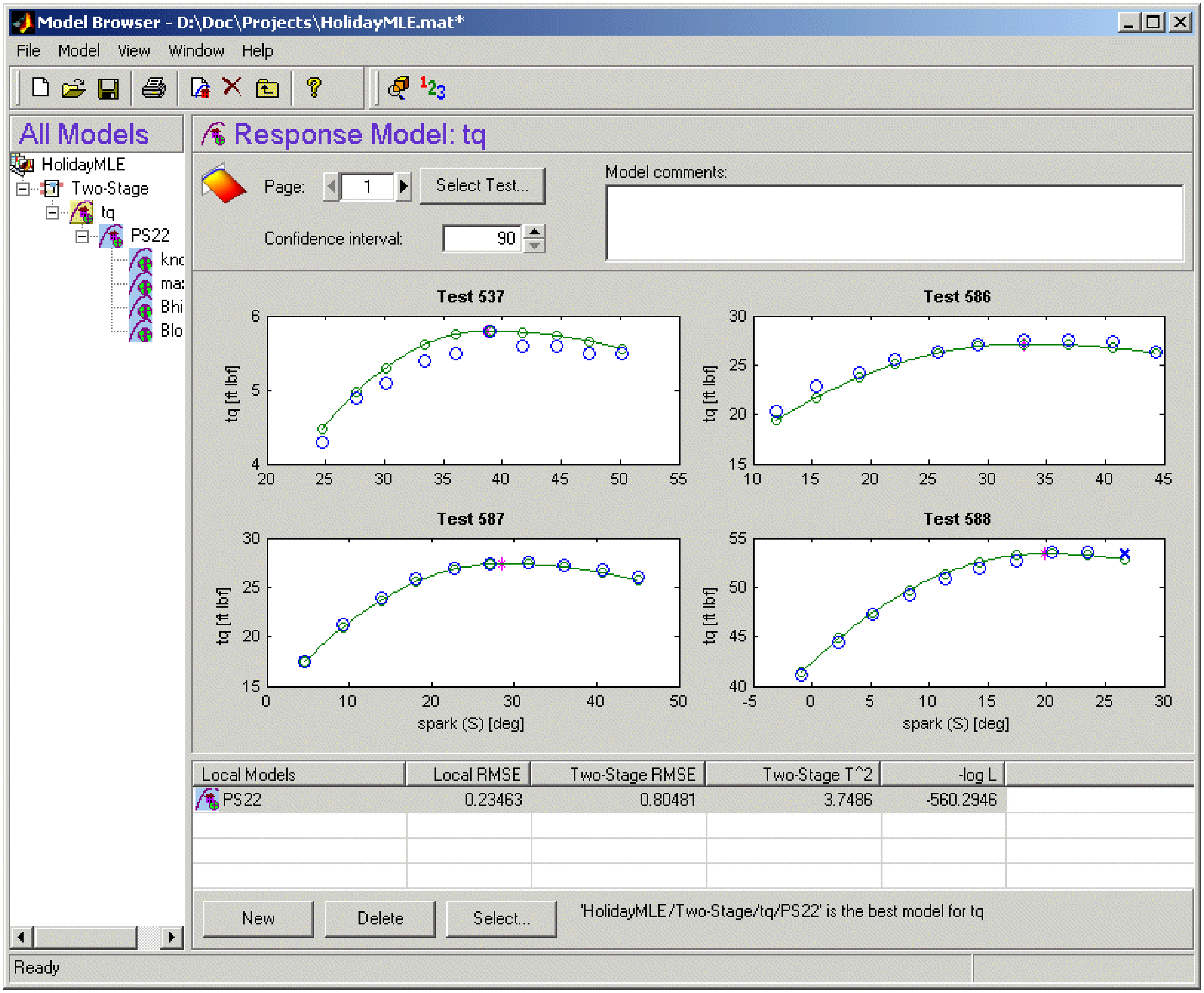
7 Close the Model Selection window. Click **Yes** to accept the MLE model as the best.

Response Node

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



Now at the Response node in the Model Browser tree (`tq`), which was previously blank, you see this:



Page: 1

Confidence interval: 90

Local Models	Local RMSE	Two-Stage RMSE	Two-Stage T^2	-log L
PS22	0.23463	0.80481	3.7486	-560.2946

'HolidayMLE/Two-Stage/tq/PS22' is the best model for tq

This shows you the fit of the two-stage model to the data. You can scroll through the tests, using the arrows at top left, to view the two-stage MLE model (in green) against the data (in blue) for each test.

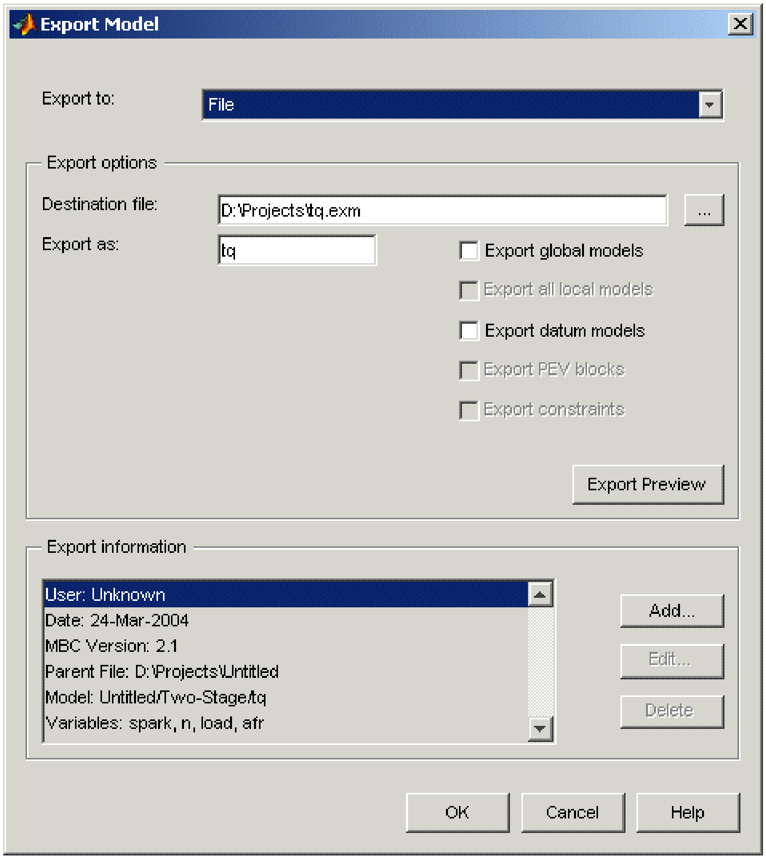
You have now completed setting up and verifying a two-stage model.

Exporting the Model

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

1 Click the `tq` node in the model tree.

2 Choose **File > Export Models**. The Export Model dialog box appears.



Export to: File

Export options

Destination file: D:\Projects\tq.exm

Export as: tq

☐ Export global models

☐ Export all local models

☐ Export datum models

☐ Export PEV blocks

☐ Export constraints

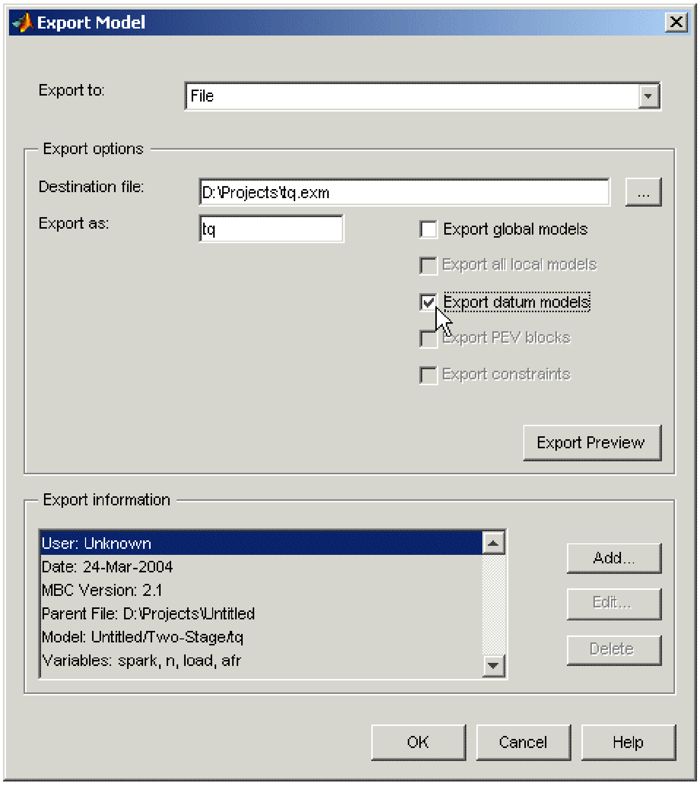
Export information

User: Unknown
Date: 24-Mar-2004
MBC Version: 2.1
Parent File: D:\Projects\Untitled
Model: Untitled/Two-Stage/tq
Variables: spark, n, load, afr

3 Choose `File` from the **Export to** pop-up menu. This saves the work as a file for use within the Model-Based Calibration Toolbox, for instance, to create calibrations in the CAGE Browser.

4 In the **Export Options** frame, select the destination of the file. You can do this by typing directly in the edit box, or using the Browse button if you want to locate a directory or use an existing file.

5 Ensure that **Export datum models** is selected, as this allows the datum global model to be exported. The datum model in this case is MBT (the spark angle at maximum brake torque).



Export to: File

Destination file: D:\Projects\tq.exm

Export as: tq

□ Export global models

□ Export all local models

☑ Export datum models

□ Export PEV blocks

□ Export constraints

User: Unknown
Date: 24-Mar-2004
MBC Version: 2.1
Parent File: D:\Projects\Untitled
Model: Untitled/Two-Stage/tq
Variables: spark, n, load, afr

6 Click **OK** to export the models.

Creating Multiple Models to Compare

Once you have fitted and examined a single model, you will normally want to create more models to search for the best fit. You can create individual new models, use the **Build Models** function to create a selection of models at once, or create a template to save a variety of model settings for reuse.

You can create new child nodes by clicking the **New** button from any modeling node. Then select the new child node on the tree and choose **Model > Set Up Model** to open the Model Setup dialog and change the type and settings. You can do this for multiple child nodes to create a selection of different model types fitted to the same data. You can also use the Build Models dialog to quickly create a selection of different child nodes to compare. The following exercises show you examples of these processes. Note that you need to complete the previous tutorial sections to have a complete two-stage model as a starting point.

Creating New Local Models

1. As an example, select the `tq` response node and click **New** in the **Local Models** list pane.

   The Local Model Setup dialog appears.

2. Select a `Polynomial Spline` with a spline order of `3` below the knot and `2` above. Click **OK**.

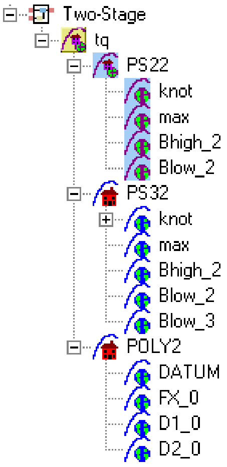
   A new set of local models (and associated response feature models) is calculated.

3. Click **New** again, in the **Local Models** list pane.

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

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

Now you have three alternative local models to compare: two polynomial splines (order 3,2 and order 2,2) and a polynomial (order 2), as shown.



You can select the alternative local models in turn and compare their statistics. For an example, follow these steps:

1 Select the new local model node `PS32`.

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

3 In the **Diagnostic statistics** pane, select `Local diagnostics` from the drop-down menu. Observe the value of `s_i` in this pane. This is the value of RMSE (root mean squared error) for the current (i^{th}) test.

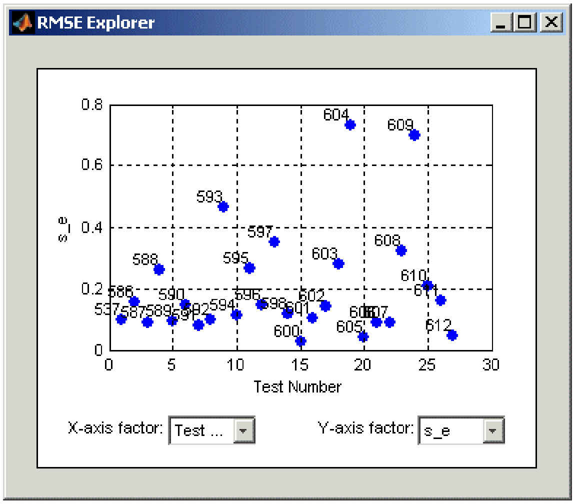
   The RMSE value is our basic measure of how closely a model fits some data, which measures the average mismatch between each data point and the model. This is why you should look at the RMSE values as your first tool to inspect the quality of the fit — high RMSE values can indicate problems.

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

   Observe that the shape of the torque/spark sweep for this test is better suited to a polynomial spline model than a polynomial model. The curve is not symmetrical because curvature differs above and below the maximum (marked by the red cross at the datum). This explains why the value of `s_i` is much lower for `PS32` (the polynomial spline) than for the `POLY2` (polynomial) for this test. The polynomial spline is a better fit for the current test.

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

6 To help you quickly identify which local models have the highest RMSE, indicating problems with the model fit, click RMSE Plots ($\hat{\sigma}$) in the toolbar (or select **View > RMSE Plots**) to open the RMSE Explorer dialog.



X-axis factor: Test ...

Y-axis factor: s_e

a Right-click to toggle the test number display to help you identify problem tests.

b Use the drop-down menus to change the display. For example, select `s_knot` to investigate the error values for knot (MBT), or `s_e` to look at overall error.

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

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

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

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

Adding New Response Features

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

1 Select the local model node PS32.

2 Click the **New** button under the list of response features.

A dialog appears with a list of available response features.

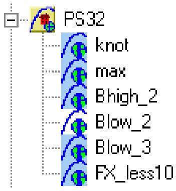
3 Select f(x+datum) from the list and enter -10 in the **Value** edit box. Click **OK**.

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

4 Ensure that the local node PS32 is selected, and click **Select**. Notice that there are four possible two-stage models this time. This is because you added a sixth response feature. Only five (which must include knot) are required for the two-stage model, so you can see the combinations available and compare them. Note that not all combinations of five response features can completely describe the shape of the curve for the two-stage model, so only the possible alternatives are shown.

5 Close the Model Selection window and click **OK** to accept one of the models as best. Click **Cancel** to avoid calculating MLE.

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



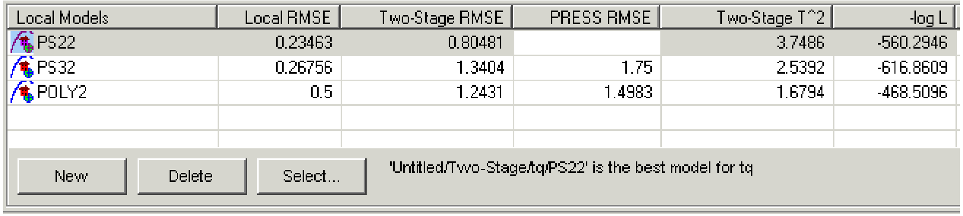
6 Select the tq response node to see a comparison of the statistics of both two-stage models (your original PS22 and the new PS32).

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

7 To calculate the two-stage model for POLY2, click **Select** at the POLY2 node. Either double-click to assign a model as best or close the Model Selection window and click **OK** to accept the best model. Click **Cancel** to avoid calculating MLE, then the two-stage model is calculated.

Comparing Models

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



Local Models	Local RMSE	Two-Stage RMSE	PRESS RMSE	Two-Stage T^2	-log L
PS22	0.23463	0.80481		3.7486	-560.2946
PS32	0.26756	1.3404	1.75	2.5392	-616.8609
POLY2	0.5	1.2431	1.4983	1.6794	-468.5096

'Untitled/Two-Stage/tq/PS22' is the best model for tq

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

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

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

Notice that your first two-stage model (PS22) does not have a PRESS RMSE value. This is because it cannot be displayed for MLE models. You need non-MLE models to be able to use PRESS for direct comparison.

- Look for lower T^2 values. A large T^2 value indicates that there is a problem with the response feature models.
- Look for large negative log likelihood values to indicate better fits.

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

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

- You can plot the models simultaneously on the Tests, Residuals and Cross Section views (**Shift**- or **Ctrl**-click to select models in the list)
- You can view each model in the Response Surface view as a surface; movie, contour or multiline plot, and as a table

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

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

Creating New Global Models

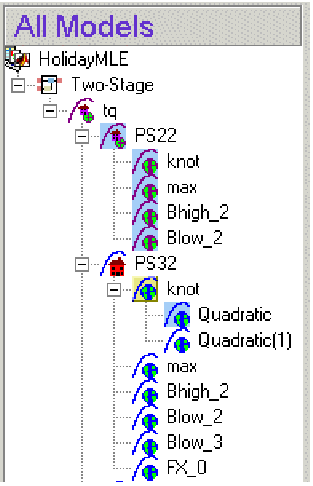
In this example, you have not yet searched for the best global model types. You would normally do this before creating and comparing two-stage models. For the purpose of this tutorial, you have already created two-stage models and used the Model Selection tool to introduce the use of RMSE and PRESS to help you identify better models. The principle is the same at each level in the model tree: add new child models and use the Model Selection window to choose the best.

1 Select one of the response feature nodes under the PS32 node, such as knot.

2 Click **New** twice.

Two new global model child nodes appear underneath knot, as shown. Both are labeled Quadratic, as they are currently copies of the parent model. You can create any number of child nodes to search for the best global model fit for each response feature in your tree. When you choose the best, it is copied to the parent node, in this case knot, including any outliers you decide to exclude.

A good technique for creating multiple models can be to leave the first child node unchanged, then you always have a copy of the original model for comparison.


All Models
HolidayMLE
Two-Stage
tq
PS22
knot
max
Bhigh_2
Blow_2
PS32
knot
Quadratic
Quadratic(1)
max
Bhigh_2
Blow_2
Blow_3
FX_0

3 Select one of the new Quadratic nodes, then select the menu item **Model > Set Up**.

The Global Model Setup dialog appears. Here you can change the type and settings of the model to see if you can find a better fit to the data with a different model type.

4 Use the drop-down menu to change the **Model class** to Hybrid RBF and click **OK**.

The new model fit is calculated, and the Quadratic node's name changes to Linear-RBF.

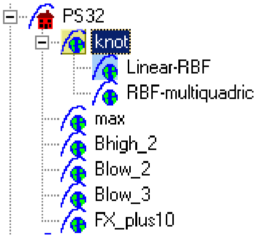
5 Select the remaining Quadratic node, then select **Model > Set Up**.

6 Use the drop-down menu to change the **Model class** to `Radial Basis Function`, and click **OK**. There are many other settings you can alter for both these model types, but for a quick exploration of the trends in the data it is worth trying the default model settings.

   The new model fit is calculated and the `Quadratic` node's name changes to `RBF-multiquadric`.

7 To compare the two child node models, select the parent node `knot` and click **Select**. Whichever model you assign as best is copied to the `knot` node when you close the Model Selection window and click **OK**.

   Notice that the child node model assigned as best is highlighted in blue, and the local node has changed from the two-stage icon back to the local model icon (a red house) as shown. This is because you have changed one of the response feature models, and so you need to recalculate the two-stage model using the new global model for this response feature. First you need to select best global models for every response feature.



8 Add two more child nodes to the `knot` global model (select `knot`, then click **New** twice).

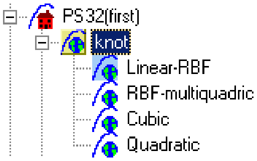
   Notice that now the new nodes are copies of Linear-`RBF`, because that model was selected as best.

9 Select the two new nodes in turn and change their model types. Try a cubic and quadratic polynomial model type.

   a Select the menu item **Model > Set Up**.

   b Choose `Linear model` from the **Model class** drop-down menu and set the polynomial order for each factor to `3` for one model, then `2` for the other. Click **OK**.

10 To compare all four child node models, select the parent node `knot` and click **Select**. `Linear-RBF` still performs the best for `PRESS RMSE`. Whichever model you assign as best is copied to the `knot` node.



11 Select the knot model node, then select **Model > Make Template**. Browse to a suitable work directory and enter the name `Mytemplate`. Click **OK**.

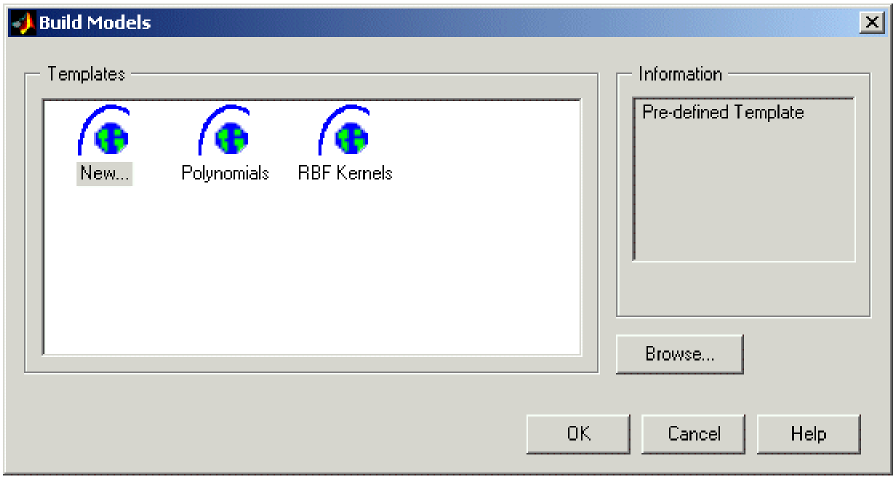
Creating Multiple Models Using Build Models

The quickest way to create multiple different models to compare is to use the Build Models function. You can use this to select a template and build a selection of models as child nodes of the current node. The best model of this selection of child nodes is automatically selected (it will have a blue icon), based on the selection criteria you choose (such as PRESS RMSE, RMSE, Box-Cox, and so on).

1 Before calculating MLE, select a global model such as `max`.

   You cannot reach the Build Models dialog from an MLE global model. Note that calculating MLE is not irreversible — to go back you can always go to Model Selection (from the local node) and select the `Univariate` model as best.

2 Click Build Models in the toolbar.



The Build Models dialog appears. Here you can choose a template for the type of models you want to build. There are predefined templates for polynomials and RBF kernels. You can also save templates of whatever models you choose by selecting the `New` template or using the **Model > Make Template** menu item, as you did in the previous section. Your user-defined templates can then be found via the Build Models dialog. You can use the Browse button to find stored templates that are not in the default directory.

3 Click Browse and select the directory containing the template you created earlier, named `Mytemplate`. Click **OK**.

Your new template (called `Mytemplate`) now appears in the Build Models dialog along with the defaults. Note that you can set the default directory where the toolbox looks for templates (and models, data, and projects) using **File > Preferences**.

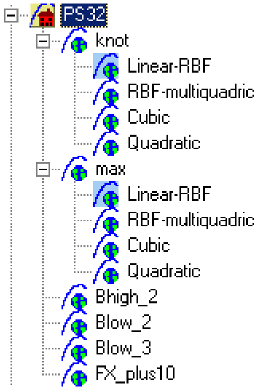
4 Select `Mytemplate`. Notice the four model types you saved in this template appear in the **Information** pane. Click **OK**.

Four child nodes appear: `Linear-RBF`, `RBF-multiquadric`, `Cubic`, and `Quadratic`. These are the model types you selected when you built the template and are now fitted to the data for `max`.

The Model Selection dialog appears, where you can choose a criterion for automatically selecting the best model out of the child nodes.

5 Use the drop-down menu to choose `PRESS RMSE` as the selection criteria for the best model, and click **OK**.

The best child node model, based on `PRESS RMSE`, is selected as best (highlighted in blue) as shown. This model is also copied to the parent node `max`, in the same way as if you had used the Model Selection window to assign a best model.



Try one of the default templates in the Build Models dialog as follows:

1 Select another global model such as `Blow_2`.

2 Click Build Models in the toolbar.

3 Select `RBF` and click **OK**. Click **Build** in the following Model Building Options dialog to build a selection of child nodes.

Similarly, you can use the Build Models dialog to automatically build a selection of polynomial or hybrid RBF models, or your own selection of model types, to search for the best fit.

4 Click **OK** in the Model Selection dialog to accept the default, `PRESS RMSE`, as the selection criteria for the best model.

5 Look at the statistics in the lower list pane to quickly compare all the different RBF kernel child models. If one model performs significantly better in terms of PRESS RMSE and RMSE you might choose not to click **Select** to compare all the child node models. However, it is usually useful to visually inspect the models to see how they compare.

6 When you have chosen a best model, it can be useful to select some (or all) of the rejected models in the bottom list pane and press **Delete**. You can also select **File > Clean Up Tree**. This deletes all rejected child models where best models have been chosen; only the child nodes selected as best remain.

Creating a template containing a list of all the models you want is a very efficient way to quickly build a selection of alternative model child nodes for many global models. Use these techniques to find models well suited to the data for each of your global models.

When you have chosen best global models for all your response features, you need to recalculate the two-stage model. Click **Select** at the local model (`PS32`) node to calculate the two-stage model.

6

Tutorial: Design of Experiment

This section discusses the following topics:

What Is Design of Experiment?

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

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

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

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

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

Design Styles

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

Optimal designs are best for cases with high system knowledge, where previous studies have given confidence on the best type of model to be fitted, and the constraints of the system are well understood. See "Creating Optimal Designs" on page 6-10.

Space-filling designs are better when there is low system knowledge. In cases where you are not sure what type of model is appropriate, and the constraints are uncertain, space-filling designs collect data in such as a way as to maximize coverage of the factors' ranges as quickly as possible. See "Creating Space-Filling Designs" on page 6-35.

Classical designs (including full factorial) are very well researched and are suitable for simple regions (hypercube or sphere). Engines have complex constraints and models (high-order polynomials and splines). See "Creating Classical Designs" on page 6-26.

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

Structure of This Design Editor Tutorial

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

1 To start the tutorial, you pick a model to design an experiment for, enter the Design Editor, and construct an optimal design. Once you create a

design, you can use the displays and tools to examine the properties of the design, save the design, and make changes.

See

- “Starting the Design Editor” on page 6-6
- “Creating Optimal Designs” on page 6-10
- “Viewing Design Displays” on page 6-16
- “Using the Prediction Error Variance Viewer” on page 6-20
- “Saving Designs” on page 6-42
- “Improving the Design” on page 6-23

2. Next you create a classical design, and use the PEV viewer to compare it with the previous design. You can also use the Design Evaluation tool to view all details of any design; it is introduced in this example.

   See

   - “Creating Classical Designs” on page 6-26
   - “Using the Design Evaluation Tool” on page 6-32

3. Lastly you construct a space-filling design and compare it with the others using the PEV viewer. Then you construct and apply two different constraints to this design and view the results. Normally you would design constraints before constructing a design, but for the purposes of this tutorial you make constraints last so you can view the effects on your design.

   See

   - “Creating Space-Filling Designs” on page 6-35
   - “Applying Constraints” on page 6-37

For more details on functionality in the Design Editor, see the reference section “Designs”.

Starting the Design Editor

Setting Up a Model

You must first have a model for which to design an experiment.

1 From the Model Browser at startup, click the button in the toolbar, or click **New** in the **Test Plans** pane, or choose **File > New Test Plan**.

2 Select `Two-Stage Model` and click **OK**.

3 Click the new `Two-Stage` node that appears in the model tree (in the **All Models** pane), or double-click `Two Stage` in the **Test Plans** list at the bottom. The Two-Stage Model diagram appears.

If you already have a project open, you can select any existing model within the test plans in the Model Browser tree. For the purposes of this tutorial, you design experiments for the default Two-Stage global model, which is a quadratic.

There is only one input to the global model by default. To increase the number of input factors:

1 Double-click the Global Model Inputs block in the diagram. The Input Factors Setup dialog appears.

2 Increase the number of factors to three by clicking the **Number of Factors** up/down buttons or entering 3 in the edit box.

3 Change the symbols of the three input factors to `N`, `L`, and `A`. This matches the global factors modeled in the Quick Start tutorial: speed (n), load (L), and air/fuel ratio (A).

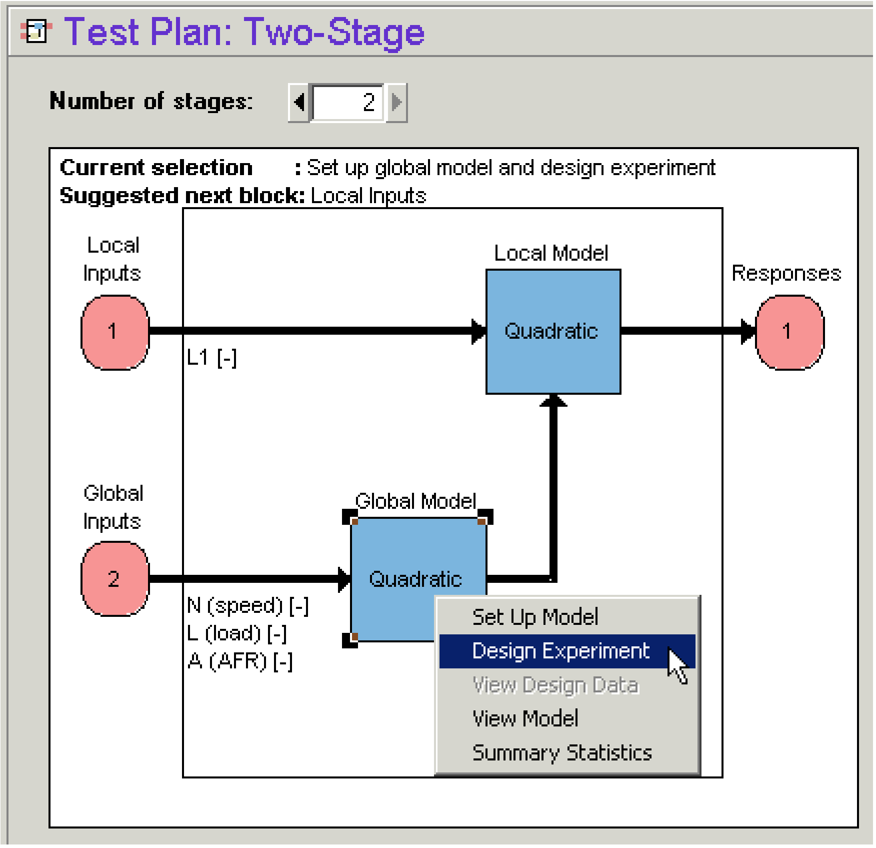
4 Click **OK** to leave the Input Factor Setup dialog.

Starting the Design Editor

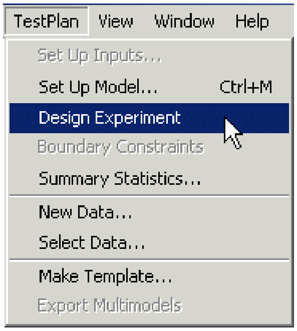
To access the Design Editor, use either of the following methods:

- Right-click the global model in the diagram and choose **Design Experiment**, as shown.

- You can also access the Design Editor by selecting the menu item **TestPlan > Design Experiment**.



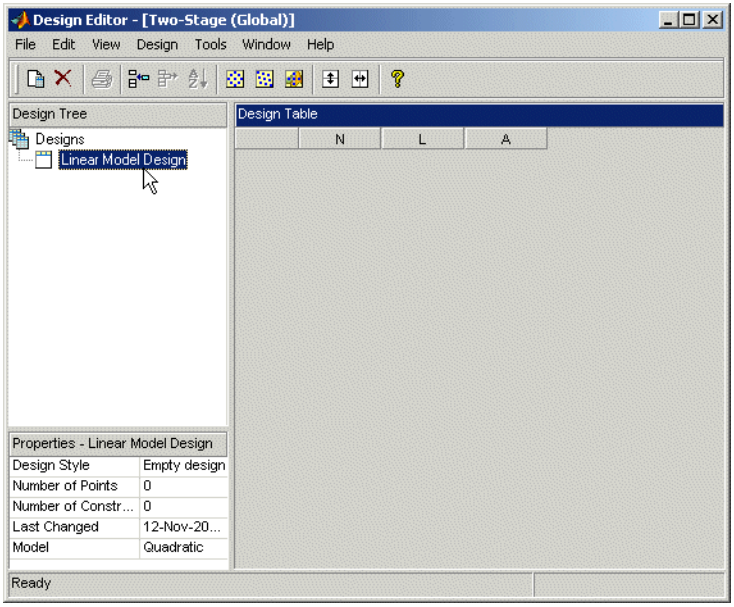
Current selection : Set up global model and design experiment
Suggested next block: Local Inputs



The Design Editor window appears.

Creating a New Design

1 Click the button in the toolbar or select **File > New**. A new node called `Linear Model Design` appears.



Properties - Linear Model Design	
Design Style	Empty design
Number of Points	0
Number of Constr...	0
Last Changed	12-Nov-20...
Model	Quadratic

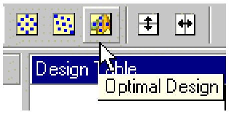
2 The new `Linear Model Design` node is automatically selected. An empty Design Table appears (see above) because you have not yet chosen a design. For this example you create an optimal design for the default global model, which is a quadratic.

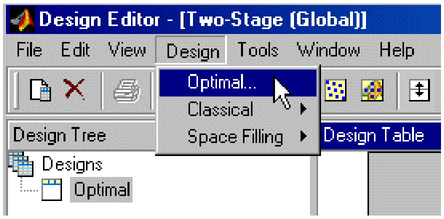
   You can change the model for which you are designing an experiment from within the Design Editor window by selecting **Edit > Model**.

3 Rename the new node `Optimal` (you can edit the names by clicking again on a node when it is already selected, or by pressing **F2**, as when selecting to rename in Windows Explorer).

Creating Optimal Designs

Choose an optimal design by clicking the ▦ button in the toolbar, or choose **Design > Optimal**.





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

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

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

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

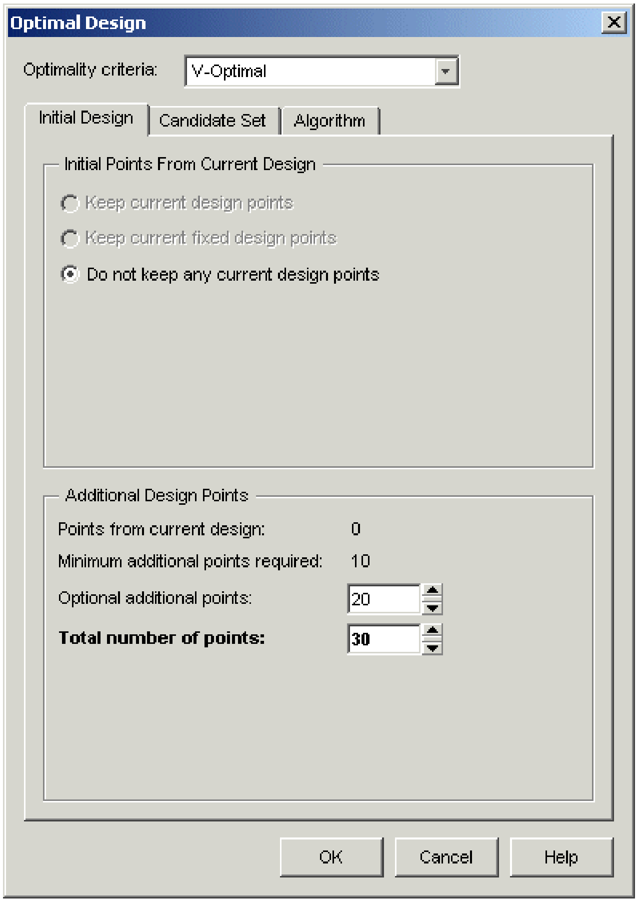
The Optimal Design dialog consists of several tabs that contain the settings for three main aspects of the design:

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

Start Point Tab

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

1. Leave the optimality criteria at the default to create a `V-Optimal` design.
2. Increase the total number of points to 30 by clicking the **Optional additional points** up/down buttons or by typing directly into the edit box. You can edit the additional points and/or the total number of points.



Optimality criteria: V-Optimal

Initial Points From Current Design

○ Keep current design points
○ Keep current fixed design points
● Do not keep any current design points

Additional Design Points

Points from current design: 0
Minimum additional points required: 10
Optional additional points: 20
Total number of points: 30

Candidate Set Tab

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

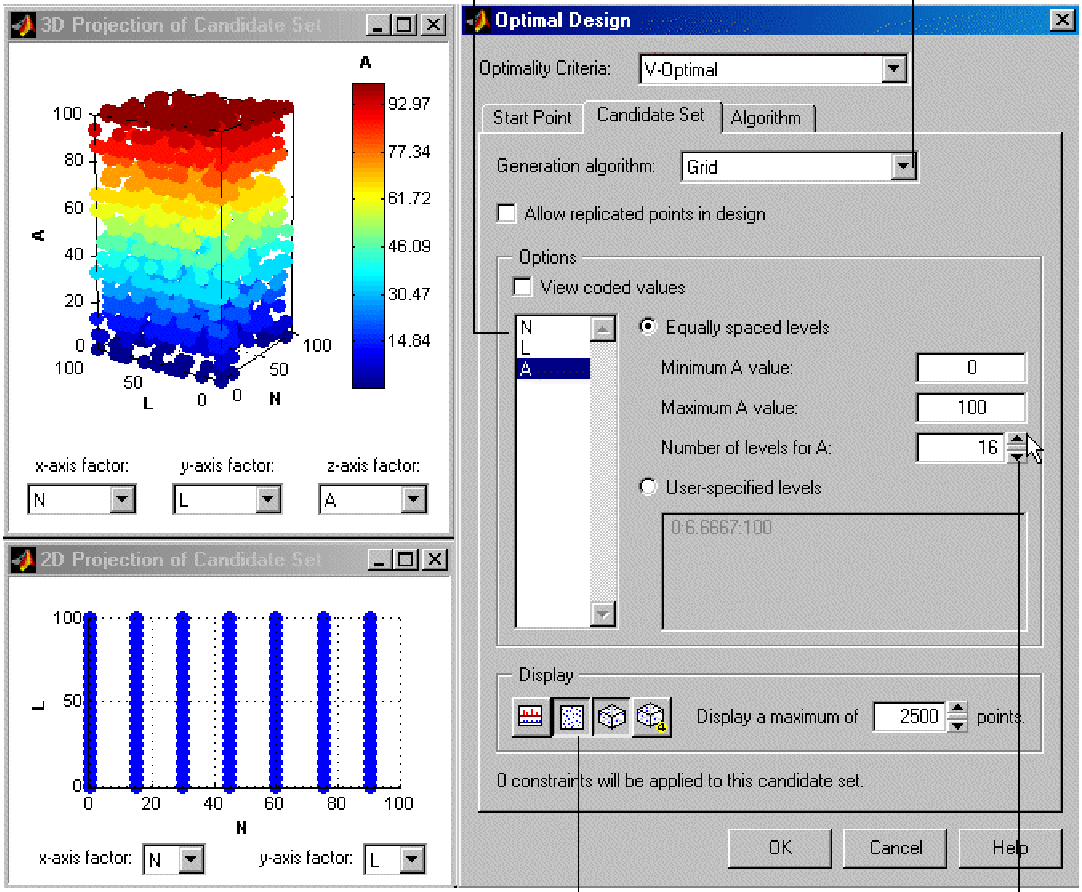
1 Choose `Grid` for this example. Note that you could choose different schemes for different factors.

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

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

Select variables in this list

Choose Grid from this list



x-axis factor: N
y-axis factor: L
z-axis factor: A

x-axis factor: N
y-axis factor: L

Optimality Criteria: V-Optimal

Generation algorithm: Grid

☐ Allow replicated points in design

☐ View coded values

N
L
A

● Equally spaced levels

Minimum A value: 0

Maximum A value: 100

Number of levels for A: 16

○ User-specified levels

0:6.6667:100

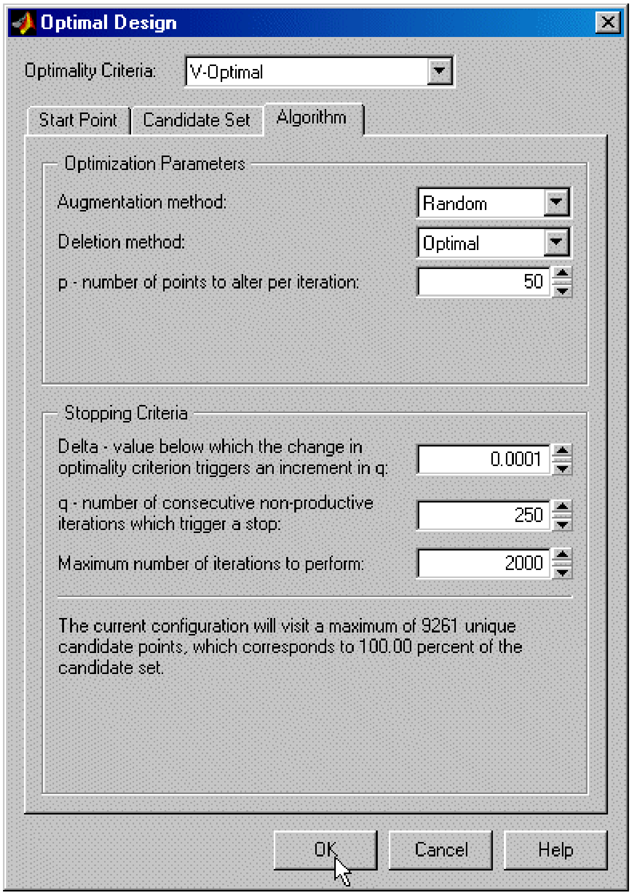
Display a maximum of 2500 points.

0 constraints will be applied to this candidate set.

Open display windows with these buttons

Change the number of levels of the selected variable here

Algorithm Tab



Optimality Criteria: V-Optimal

Optimization Parameters

Augmentation method: Random

Deletion method: Optimal

p - number of points to alter per iteration: 50

Stopping Criteria

Delta - value below which the change in optimality criterion triggers an increment in q: 0.0001

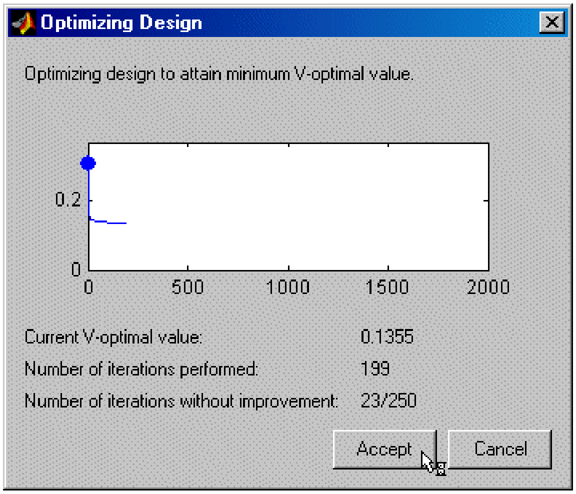
q - number of consecutive non-productive iterations which trigger a stop: 250

Maximum number of iterations to perform: 2000

The current configuration will visit a maximum of 9261 unique candidate points, which corresponds to 100.00 percent of the candidate set.

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

When you click the **OK** button on the Optimal Design dialog, the Optimizing Design dialog appears, containing a graph. This dialog shows the progress of the optimization and has two buttons: **Accept** and **Cancel**. **Accept** stops the optimization early and takes the current design from it. **Cancel** stops the optimization and reverts to the original design.



Optimizing design to attain minimum V-optimal value.

Current V-optimal value: 0.1355

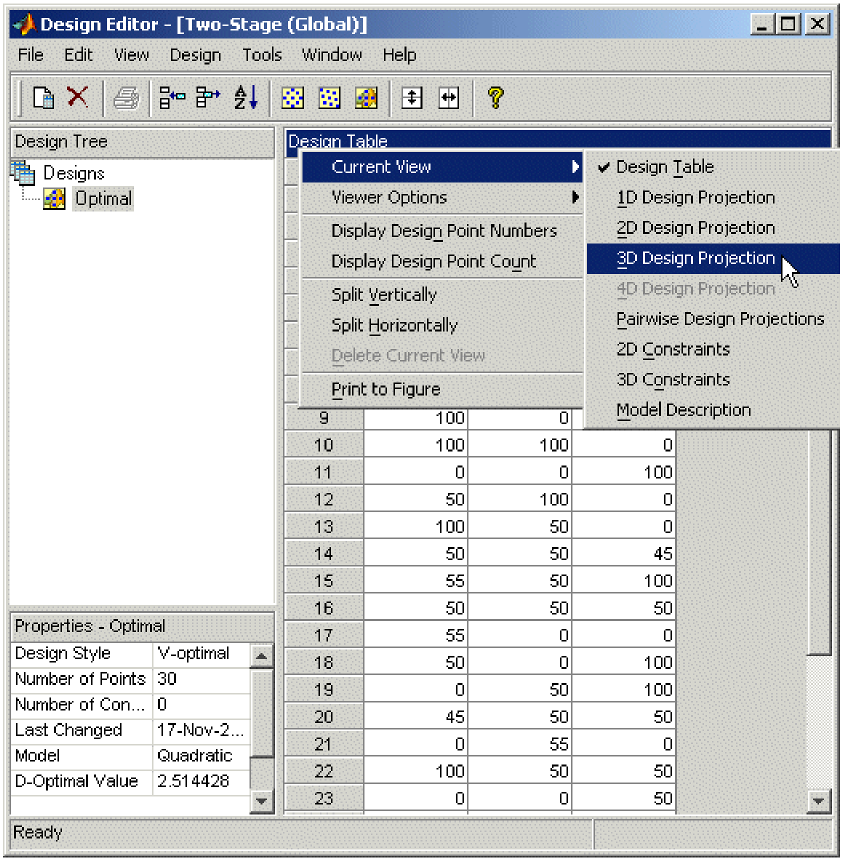
Number of iterations performed: 199

Number of iterations without improvement: 23/250

2 Click **Accept** when iterations are not producing noticeable improvements; that is, the graph becomes very flat.

Viewing Design Displays

When you press the **Accept** button, you return to the Design Editor.



Properties - Optimal

Design Style	V-optimal
Number of Points	30
Number of Con...	0
Last Changed	17-Nov-2...
Model	Quadratic
D-Optimal Value	2.514428

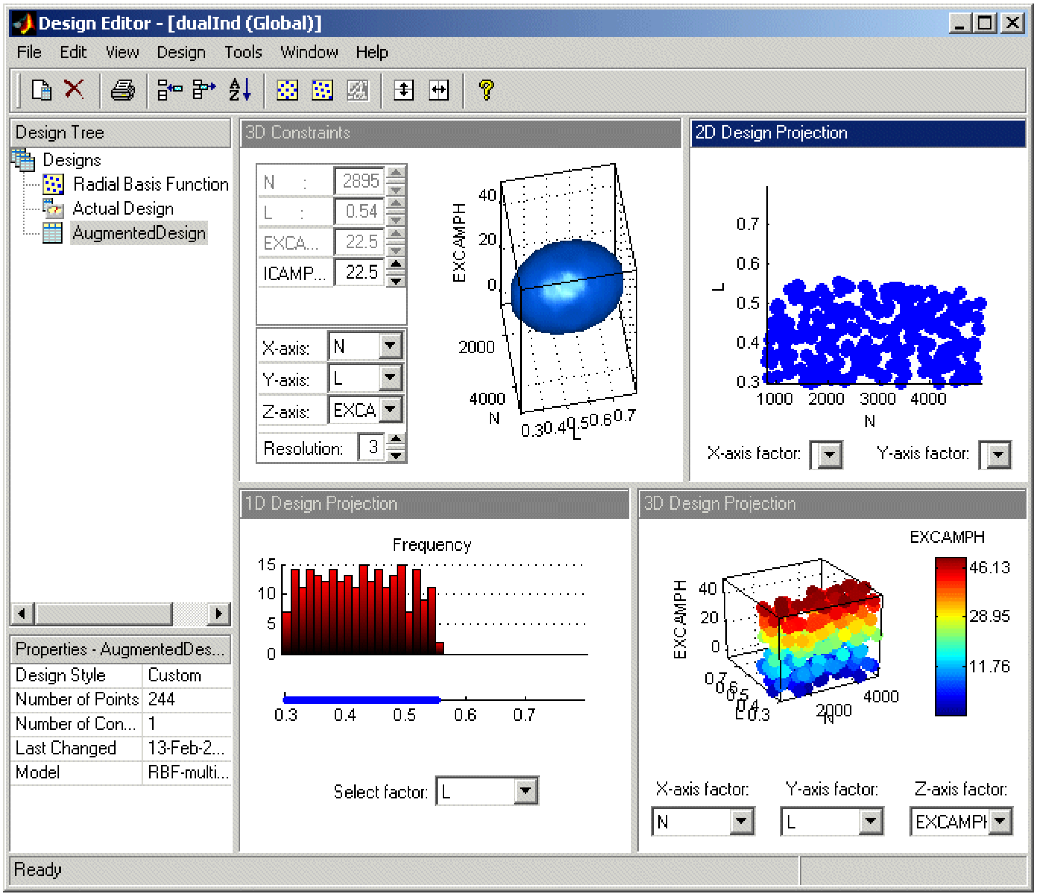
9	100	0	
10	100	100	0
11	0	0	100
12	50	100	0
13	100	50	0
14	50	50	45
15	55	50	100
16	50	50	50
17	55	0	0
18	50	0	100
19	0	50	100
20	45	50	50
21	0	55	0
22	100	50	50
23	0	0	50

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

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

Display Options

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



N : 2895
L : 0.54
EXCA... 22.5
ICAMP... 22.5

X-axis: N
Y-axis: L
Z-axis: EXCA
Resolution: 3

Properties - AugmentedDes...

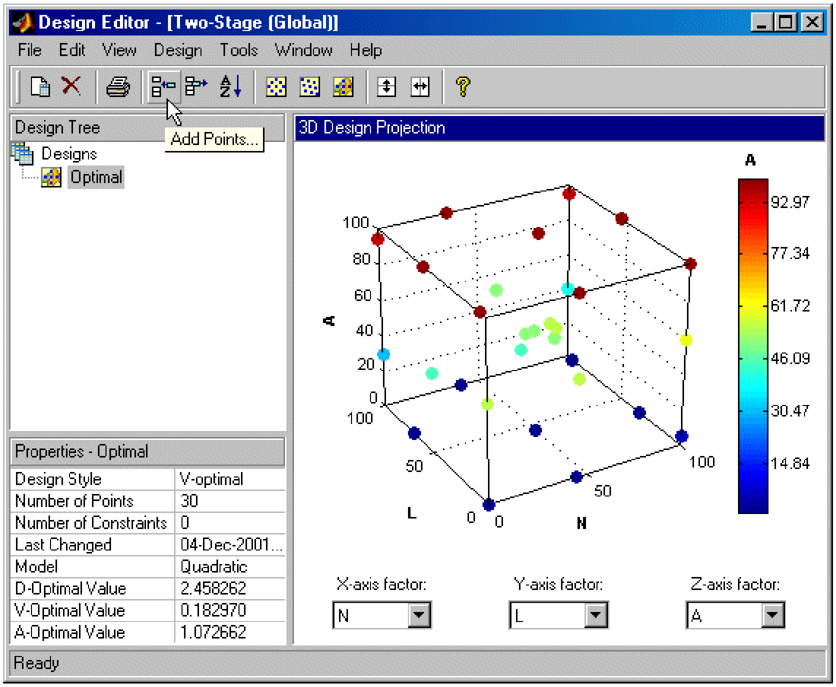
Design Style	Custom
Number of Points	244
Number of Con...	1
Last Changed	13-Feb-2...
Model	RBF-multi...

Select factor: L

X-axis factor: N
Y-axis factor: L
Z-axis factor: EXCAMPH

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

1 Change the main display to 3-D Projection view.



Properties - Optimal

Design Style	V-optimal
Number of Points	30
Number of Constraints	0
Last Changed	04-Dec-2001...
Model	Quadratic
D-Optimal Value	2.458262
V-Optimal Value	0.182970
A-Optimal Value	1.072662

X-axis factor: N

Y-axis factor: L

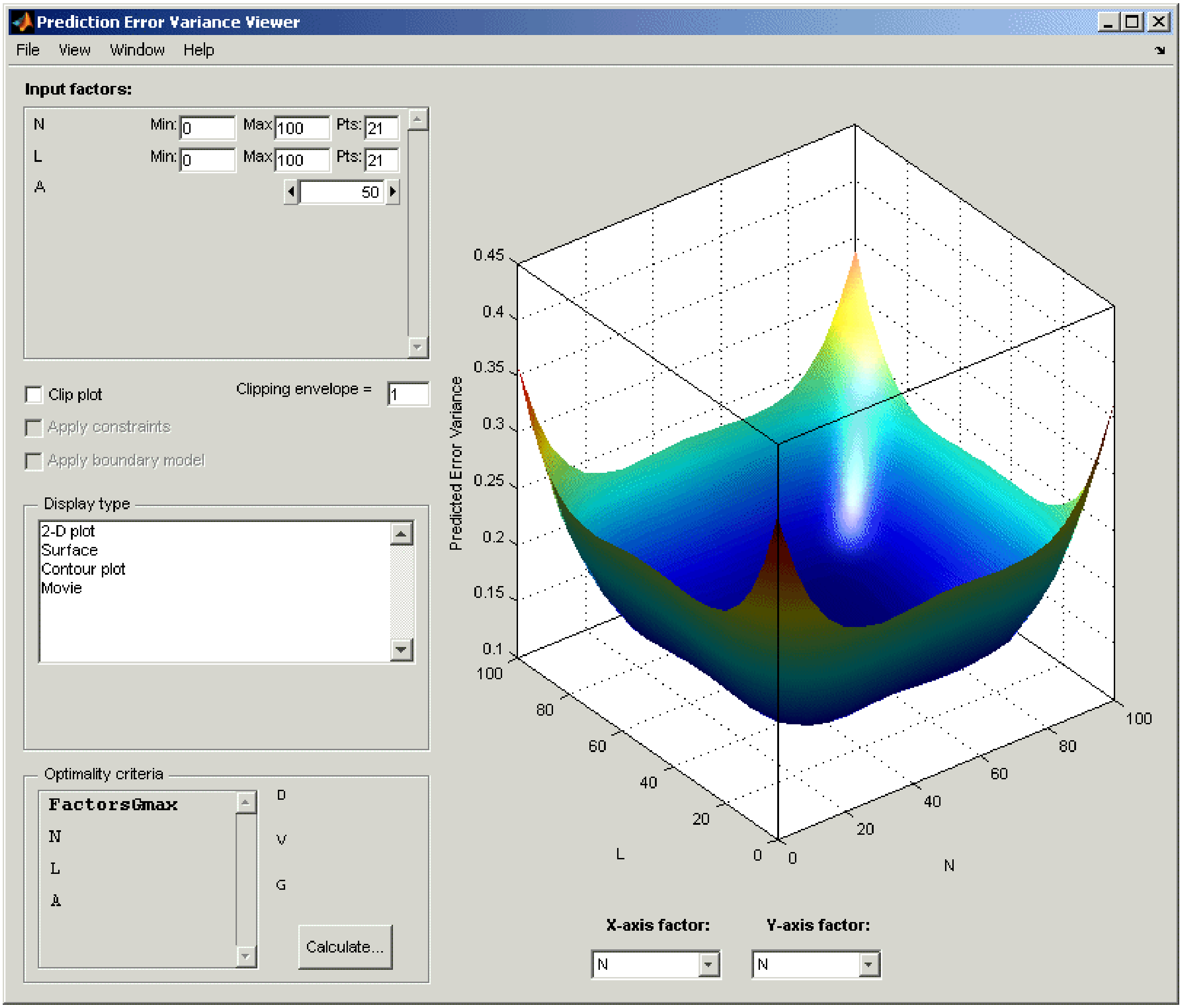
Z-axis factor: A

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

Using the Prediction Error Variance Viewer

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

Select **Tools > PEV Viewer**.



Input factors:

N Min: 0 Max 100 Pts: 21

L Min: 0 Max 100 Pts: 21

A 50

□ Clip plot Clipping envelope = 1

□ Apply constraints

□ Apply boundary model

Display type

2-D plot
Surface
Contour plot
Movie

Optimality criteria

FactorsGmax
N
L
A

X-axis factor: N

Y-axis factor: N

The default view is a 3-D plot of the PEV surface.

This shows where the response predictions are best. This example optimal design predicts well in the center and the middle of the faces (one factor high and the other midrange), but in the corners the design has the highest error. Look at the scale to see how much difference there is between the areas of higher and lower error. For the best predictive power, you want low PEV (close to zero).

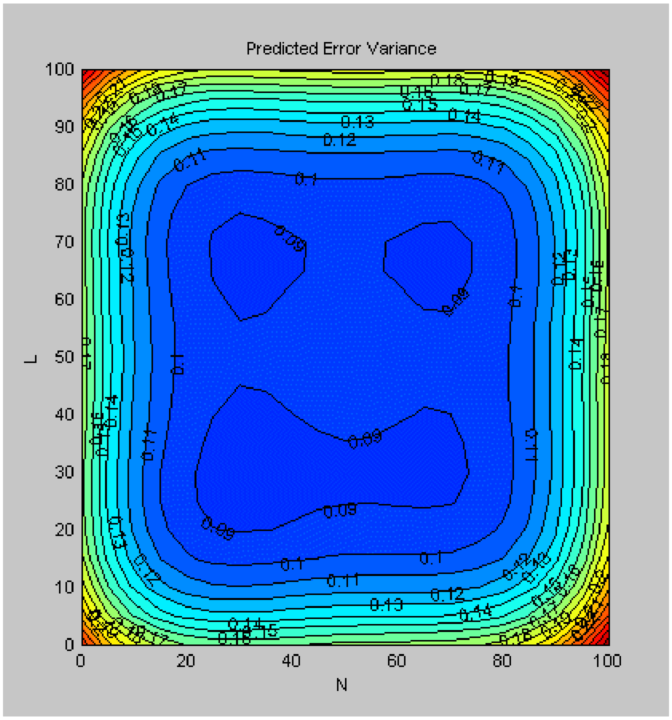
You can examine PEV for designs and models. The two are related in this way:

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

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

Try the other display options.

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



Improving the Design

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

To strengthen the current optimal design:

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

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

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

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

To see more dramatic changes to the design, return to the Design Editor window (no need to close the PEV viewer).

1 Split the display so you can see a 3-D projection at the same time as a Table view.

2 You can sort the points to make it easier to select points in one corner. For example, to pick points where N is 100 and L is 0,

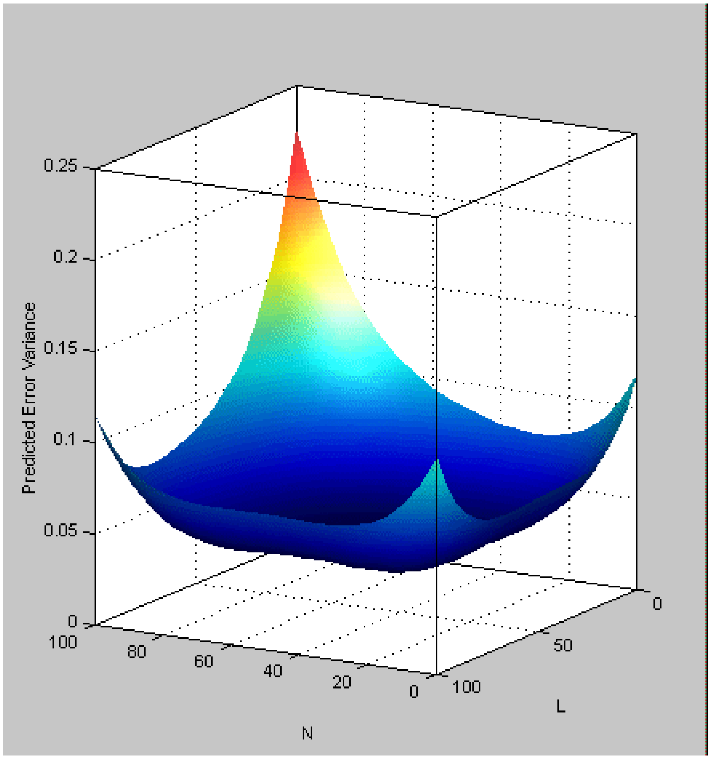
   a Select **Edit > Sort Points**.

   b Choose to sort by `N` only (reduce the number of sort variables to one) and click **OK**.

3 Choose **Edit > Delete Point**.

4 Using the Table and 3-D views as a guide, in the Delete Points dialog, pick six points to remove along one corner. Add the relevant point numbers to the delete list by clicking the add (>) button.

5 Click **OK** to remove the points. See the changes in the main design displays and look at the new Surface plot in the PEV viewer (see the example following).


Predicted Error Variance
0.25
0.2
0.15
0.1
0.05
0
100
80
60
40
20
0
N
100
50
0
L

Creating Classical Designs

1 In the Design Editor window, select the Optimal design in the design tree by clicking.

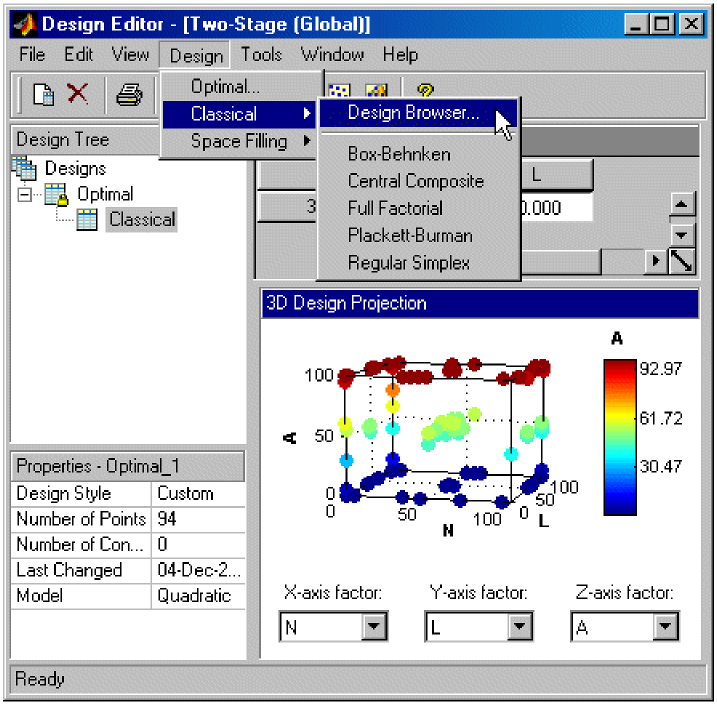
2 Add a new design. Use the first toolbar button, or select **File > New**.

A new child node appears in the tree, called `Optimal_1`. Notice that the parent node now has a padlock on the icon. This indicates it is locked. This maintains the relationship between designs and their child nodes. The tree arrangement lets you try different operations starting from a basic design, then select the most appropriate one to use. The hierarchy allows clear viewing of the effects of changes on designs. The locking of parent designs also gives you the ability to easily reverse out of changes by retreating back up the tree.

3 Select the new design node in the tree. Notice that the display remains the same — all the points from the previous design remain, to be deleted or added to as necessary. The new design inherits all its initial settings from the currently selected design and becomes a child node of that design.

4 Rename the new node `Classical` by clicking again or by pressing **F2**.

5 Click the ▦ button in the toolbar or select **Design > Classical > Design Browser**.



Properties - Optimal_1	
Design Style	Custom
Number of Points	94
Number of Con...	0
Last Changed	04-Dec-2...
Model	Quadratic

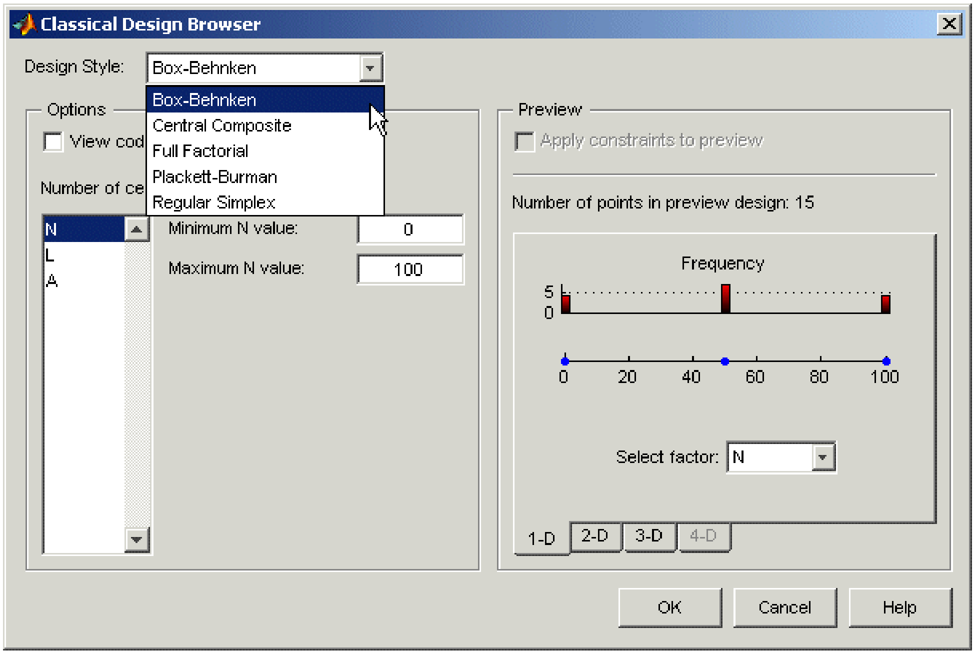
Note In cases where the preferred type of classical design is known, you can go straight to one of the five options under **Design > Classical**. Choosing the **Design Browser** option allows you to see graphical previews of these same five options before making a choice.

A dialog appears because there are already points from the previous design. You must choose between replacing and adding to those points or keeping only fixed points from the design.

6 Choose the default, replace current points with a new design, and click **OK**.

The Classical Design Browser appears.

Classical Design Browser



In the **Design Style** drop-down menu, there are five classical design options:

- `Central Composite`

  Generates a design that has a center point, a point at each of the design volume corners, and a point at the center of each of the design volume faces. You can choose a ratio value between the corner points and the face points for each factor and the number of center points to add. You can also specify a spherical design. Five levels are used for each factor.

- `Box-Behnken`

  Similar to Central Composite designs, but only three levels per factor are required, and the design is always spherical in shape. All the design points (except the center point) lie on the same sphere, so there should be at least three to five runs at the center point. There are no face points. These designs are particularly suited to spherical regions, when prediction at the corners is not required.

- `Full Factorial`

  Generates an *n*-dimensional grid of points. You can choose the number of levels for each factor and the number of additional center points to add.

- `Plackett Burman`

  These are “screening” designs. They are two-level designs that are designed to allow you to work out which factors are contributing any effect to the model while using the minimum number of runs. For example, for a 30-factor problem this can be done with 32 runs.

- `Regular Simplex`

  These designs are generated by taking the vertices of a k-dimensional regular simplex (k = number of factors). For two factors a simplex is a triangle; for three it is a tetrahedron. Above that are hyperdimensional simplices. These are economical first-order designs that are a possible alternative to Plackett Burman or full factorials.

Setting Up and Viewing a Classical Design

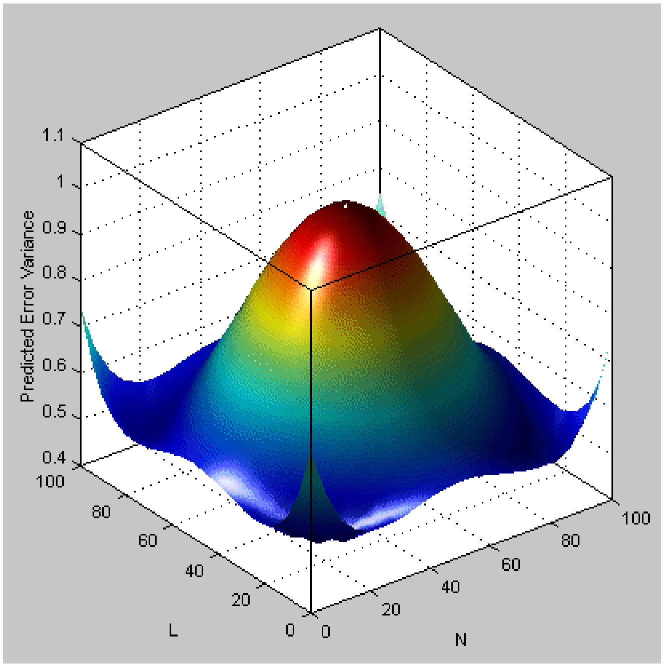
1. Choose a `Box-Behnken` design.

2. Reduce the number of center points to 1.

3. View your design in different projections using the tabs under the display.

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

5. Use the PEV Viewer to see how well this design performs compared to the optimal design created previously; see the following illustration.

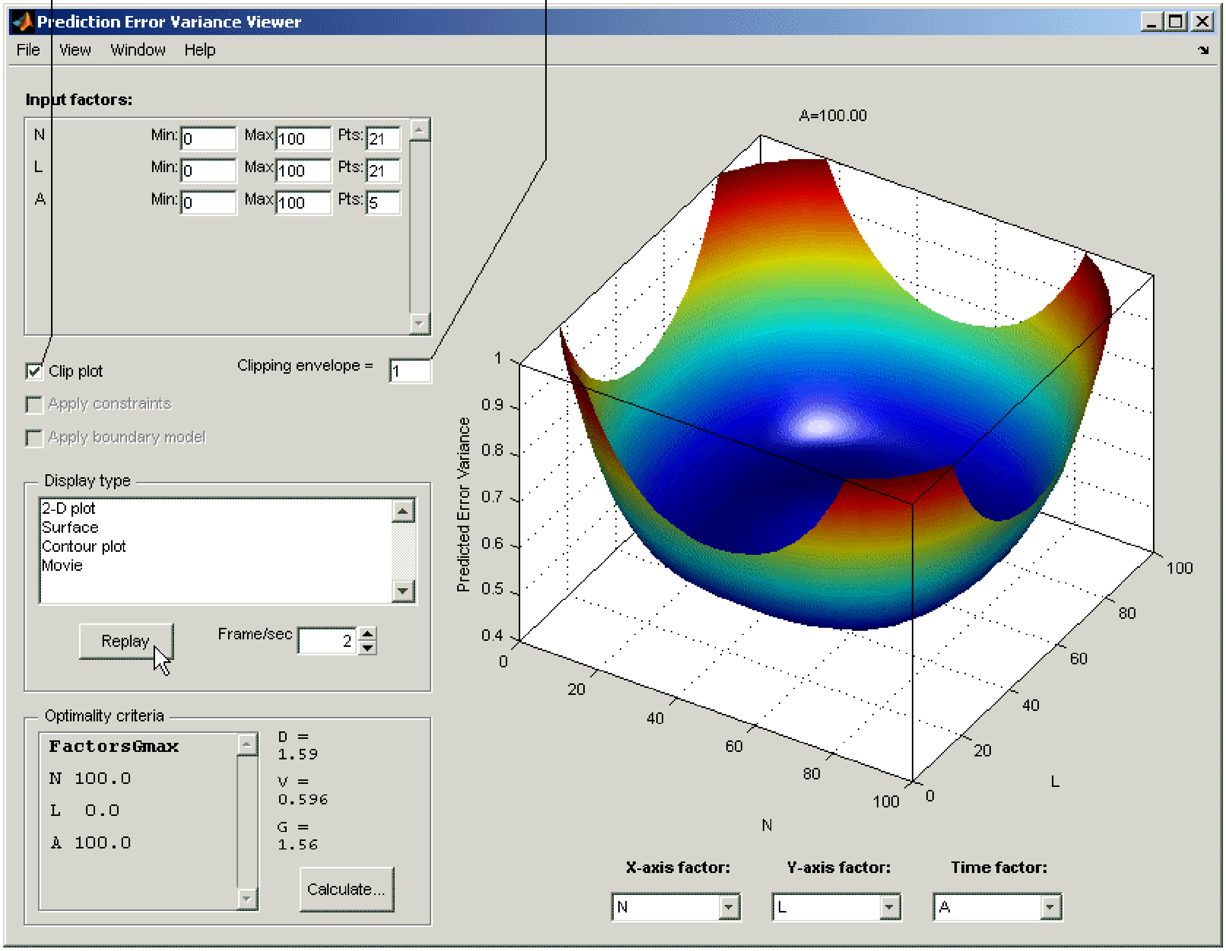


As you can see, this is not a realistic comparison, as this design has only 13 points (you can find this information in the bottom left of the main Design Editor display), whereas the previous optimal design had 100, but this is a good illustration of leverage. A single point in the center is very bad for the design, as illustrated in the PEV viewer surface plot. This point is crucial and needs far more certainty for there to be any confidence in the design, as every other point lies on the edge of the space. This is also the case for Central Composite designs if you choose the spherical option. These are good designs for cases where you are not able to collect data points in the corners of the operating space.

If you look at the PEV surface plot, you should see a spot of white at the center. This is where the predicted error variance reaches 1. For surfaces that go above 1, the contour at 1 shows as a white line, as a useful visual guide to areas where prediction error is large.

1 Select `Movie`, and you see this white contour line as the surface moves through the plane of value 1.

2 Select the **Clip Plot** check box. Areas that move above the value of 1 are removed. See the following example.



Turn clipping on and off here

Change clipping value here

Prediction Error Variance Viewer

Input factors:

N Min: 0 Max 100 Pts: 21

L Min: 0 Max 100 Pts: 21

A Min: 0 Max 100 Pts: 5

☑ Clip plot

Clipping envelope = 1

□ Apply constraints

□ Apply boundary model

Display type

2-D plot
Surface
Contour plot
Movie

Frame/sec 2

Optimality criteria

```
FactorsGmax
N 100.0
L   0.0
A 100.0
```

```
D =
1.59
V =
0.596
G =
1.56
```

X-axis factor: N

Y-axis factor: L

Time factor: A

Using the Design Evaluation Tool

The Design Evaluation Tool is available for linear models only. See also "Global Model Class: Multiple Linear Models".

1 Return to the Design Editor and select **Tools > Evaluate Designs**.

2 Choose the `Box-Behnken` design and click **OK** in the Select Designs dialog.

   The Design Evaluation Tool displays a large amount of statistical information about the design.

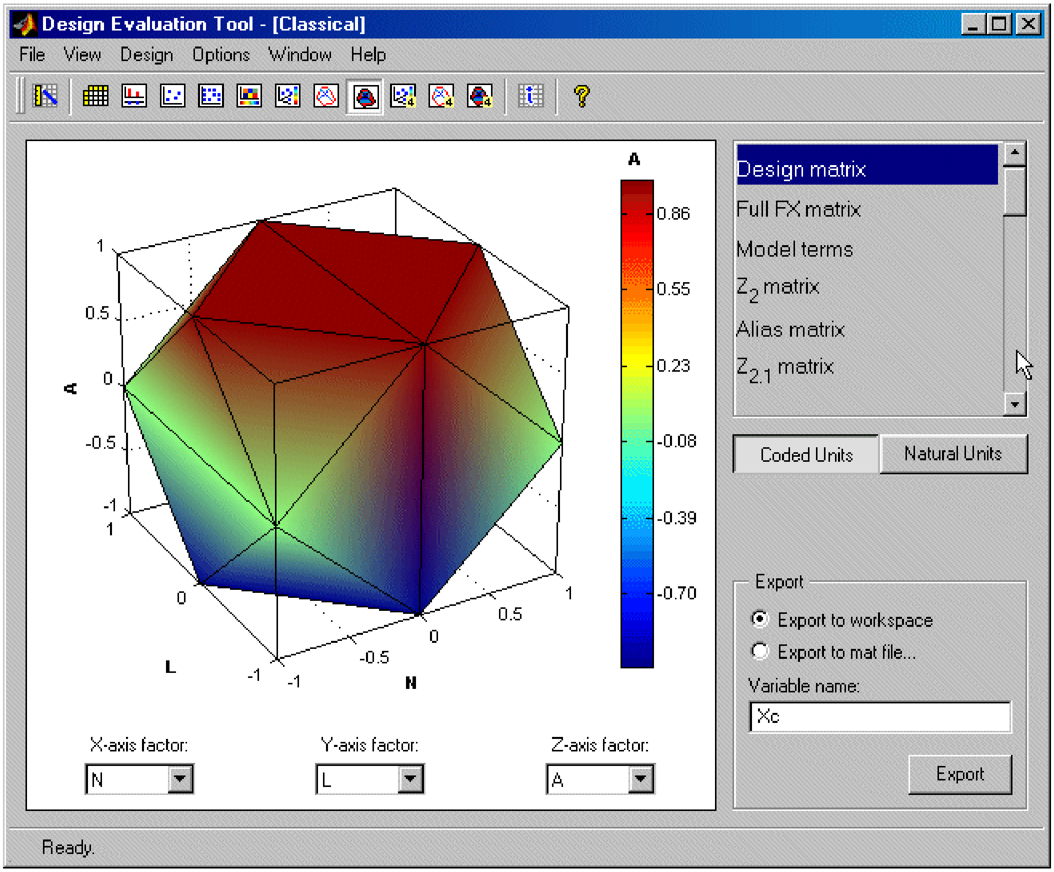
3 Select `Hat Matrix` from the list on the right.

4 Click the **Leverage Values** button.

   Note that the leverage of the central point is `1.00` (in red) and the leverage of all other points is less than this. The design would clearly be strengthened by the addition of more central points. Obviously, this is a special case, but for any kind of design, the Design Evaluation Tool is a powerful way to examine properties of designs.

5 Select `Design Matrix` from the list box.

6 Click the 3D Surface button in the toolbar.



X-axis factor: N

Y-axis factor: L

Z-axis factor: A

● Export to workspace

○ Export to mat file...

Variable name: Xc

Ready.

This illustrates the spherical nature of the current design. As usual, you can rotate the plot by clicking and dragging with the mouse.

There are many other display options to try in the toolbar, and in-depth details of the model terms and design matrices can all be viewed. You can export any of these to the workspace or a `.mat` file using the **Export** box.

For a description of all the information available here, see “Using the Design Evaluation Tool” on page 6-32.

Improving the Design

To strengthen the current Box-Behnken design near the center region:

1 Close the Design Evaluation Tool.

2 Return to the Design Editor window.

3 Select **Design > Classical > Box-Behnken**.

4 Click **OK** to replace the current points with a new design.

5 Increase the number of center points and click **OK**.

6 Once again select **Tools > PEV Viewer** and review the plots of prediction error variance and the new values of optimality criteria in the optimality frame (bottom left).

7 Review the leverage values of the center points. From the Design Editor window, use **Tools > Evaluate Design** and go to `Hat Matrix`.

8 Try other designs from the Classical Design Browser. Compare Full Factorial with Central Composite designs; try different options and use the PEV viewer to choose the best design.

Note You cannot use the PEV viewer if there are insufficient points in the design to fit the model. For example, you cannot fit a quadratic with less than three points, so the default Full Factorial design, with two levels for each factor, must be changed to three levels for every factor before you can use the PEV viewer.

9 When you are satisfied, return to the Design Editor window and choose **Edit > Select as Best**. You will see that this design node is now highlighted in blue in the tree. This can be applied to any design.

When you are creating designs before you start modeling, the design that you select as best is the one used to collect data.

Creating Space-Filling Designs

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

1. Add a new design by clicking the button in the toolbar.

   A new Classical child node appears in the tree. Select it by clicking. As before, the displays remain the same: the child node inherits all points from the parent design. Notice that in this case the parent node does *not* acquire a padlock to indicate it is locked — it is blue and therefore selected as the best design. Designs are locked when they are selected as best.

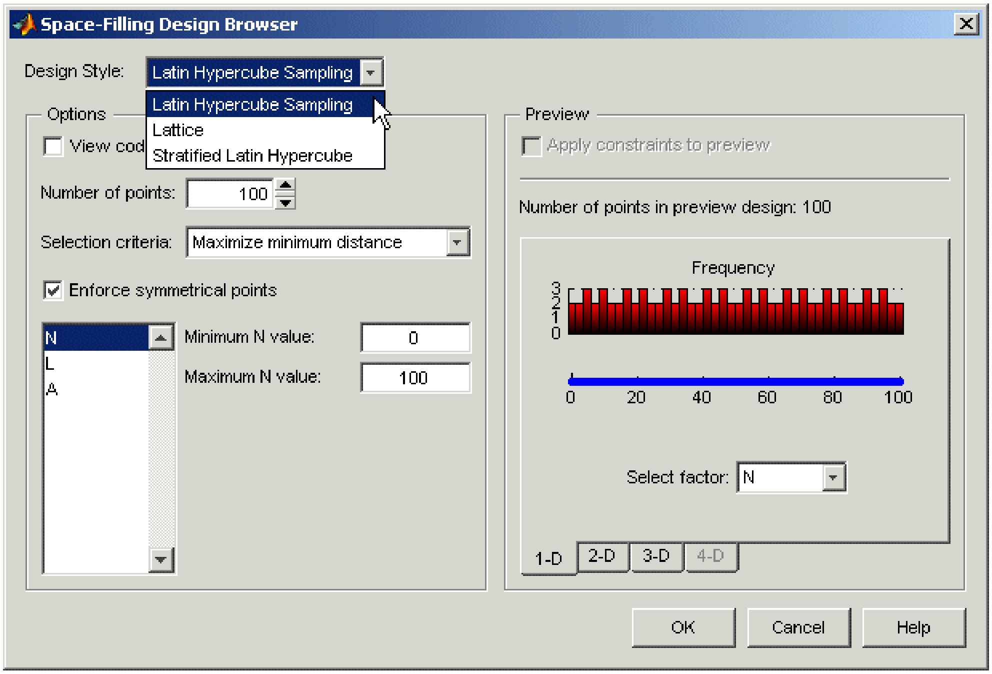
2. Rename the new node `Space Filling` (click again or press **F2**).

3. Select **Design > Space Filling > Design Browser**, or click the Space Filling Design button on the toolbar.

4. Click **OK** in the dialog to replace the current design points with a new design.

The Space Filling Design Browser appears.

Note As with the Classical Design Browser, you can select the three types of design you can preview in the Space Filling Design Browser from the **Design > Space Filling** menu in situations when you already know the type of space-filling design you want.



Space-Filling Design Browser

Design Style: Latin Hypercube Sampling

Options

□ View cod

Number of points: 100

Selection criteria: Maximize minimum distance

☑ Enforce symmetrical points

Minimum N value: 0

Maximum N value: 100

Preview

□ Apply constraints to preview

Number of points in preview design: 100

Select factor: N

Setting Up a Space-Filling Design

1 Leave the **Design Style** drop-down menu at the default `Latin Hypercube Sampling`.

2 Choose the default `Maximize minimum distance`.

3 Select the **Enforce Symmetrical Points** check box. This creates a design in which every design point has a *mirror* design point on the opposite side of the center of the design volume and an equal distance away. Restricting the design in this way tends to produce better Latin Hypercubes.

4 Use the tabs under the display to view 2-D, 3-D, and 4-D previews.

5 Click **OK** to calculate the Latin Hypercube and return to the main Design Editor.

6 Use the Design Evaluation Tool and PEV Viewer to evaluate this design.

Applying Constraints

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

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

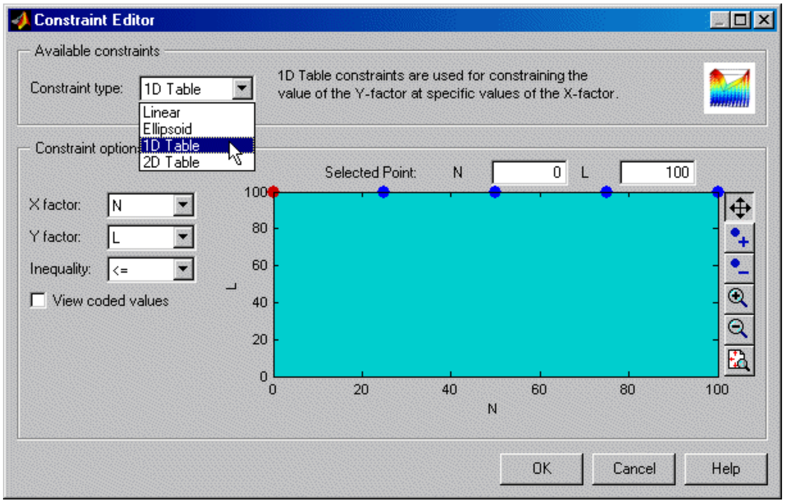
Designs can have any number of geometric constraints placed upon them. Each constraint can be one of four types: an ellipsoid, a hyperplane, a 1-D lookup table, or a 2-D lookup table.

To add a constraint to your currently selected design:

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

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

3. The Constraint Editor dialog with available constraints appears. Select `1D Table` from the **Constraint Type** drop-down menu.



Constraint Editor

Available constraints

Constraint type: 1D Table

1D Table constraints are used for constraining the value of the Y-factor at specific values of the X-factor.

Constraint options

Selected Point: N 0 L 100

X factor: N

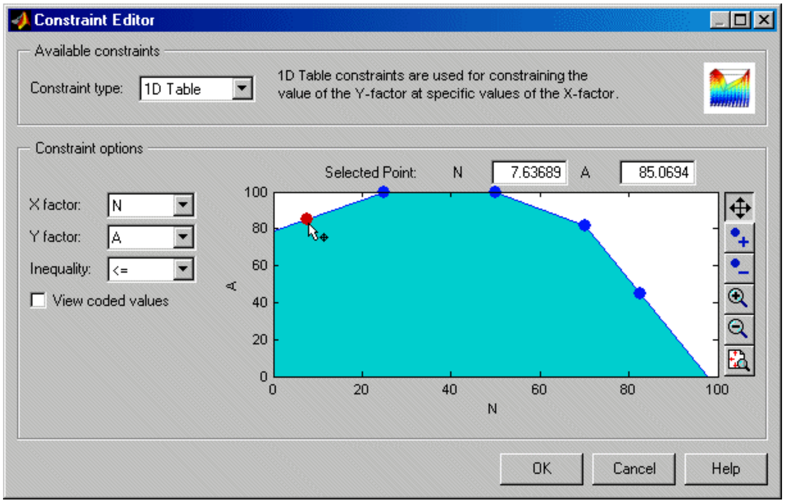
Y factor: L

Inequality: <=

□ View coded values

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

5 Move the large dots (clicking and dragging them) to define a boundary. The Constraint Editor should look something like the following.



Constraint Editor

Available constraints

Constraint type: 1D Table

1D Table constraints are used for constraining the value of the Y-factor at specific values of the X-factor.

Constraint options

Selected Point: N 7.63689 A 85.0694

X factor: N

Y factor: A

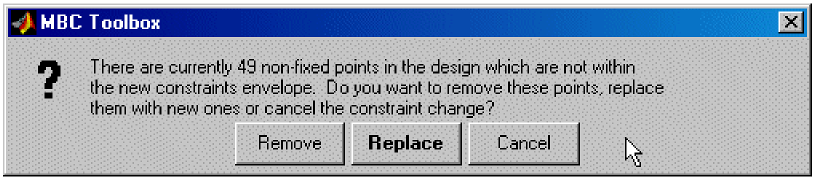
Inequality: <=

□ View coded values

6 Click **OK**.

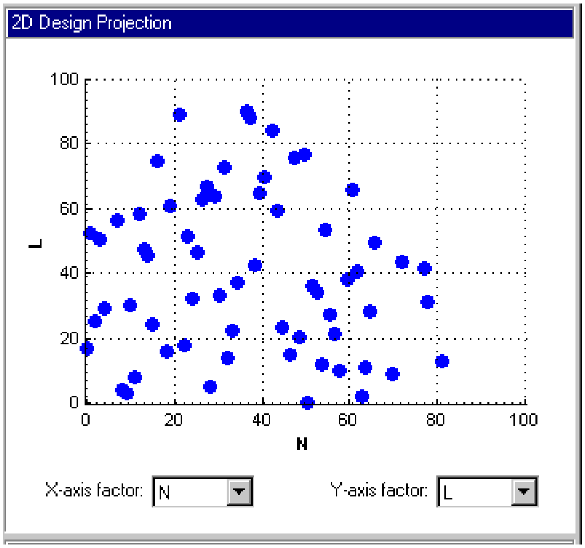
Your new constraint appears in the Constraint Manager list box. Click **OK** to return to the Design Editor. A dialog appears because there are points in the design that fall outside your newly constrained candidate set. You can simply delete them or cancel the constraint. Note that fixed points are not deleted by this process.

For optimal designs you get the dialog shown, where you also have the option to replace the points with new ones chosen (optimally if possible) within the new candidate set.



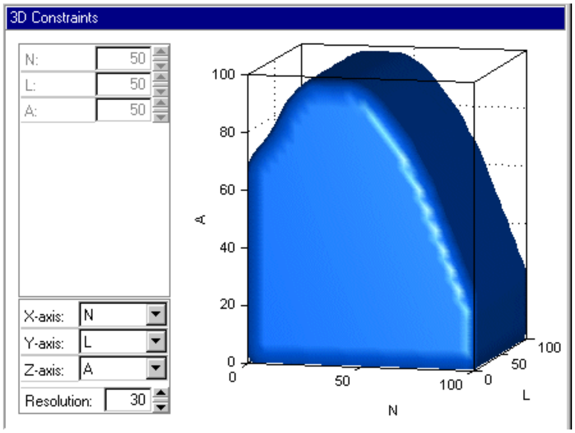
There are currently 49 non-fixed points in the design which are not within the new constraints envelope. Do you want to remove these points, replace them with new ones or cancel the constraint change?

7 The default is to remove the points outside the new constraint area; choose this.



If you examine the 2-D projection of the hypercube, you will notice the effects of the new constraint on the shape of the design, as shown in the preceding example.

8 Right-click the display pane to reach the context menu, and select **Current View > 3D Constraints**.



N: 50
L: 50
A: 50

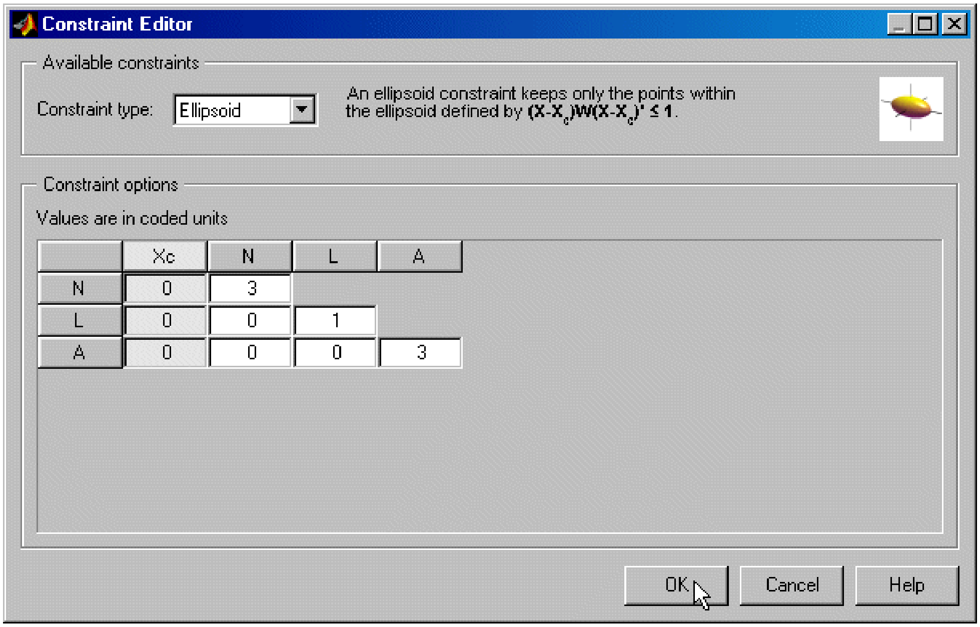
X-axis: N
Y-axis: L
Z-axis: A

Resolution: 30

These views are intended to give some idea of the region of space that is currently available within the constraint boundaries.

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

10 Add an ellipsoid constraint. Choose `Ellipsoid` from the drop-down menu of constraint types, and enter values in the table as shown. Note coded values are used in defining the ellipsoid.



Constraint Editor

Available constraints

Constraint type: Ellipsoid

An ellipsoid constraint keeps only the points within the ellipsoid defined by **$(X-X_c)W(X-X_c)' \le 1$**.

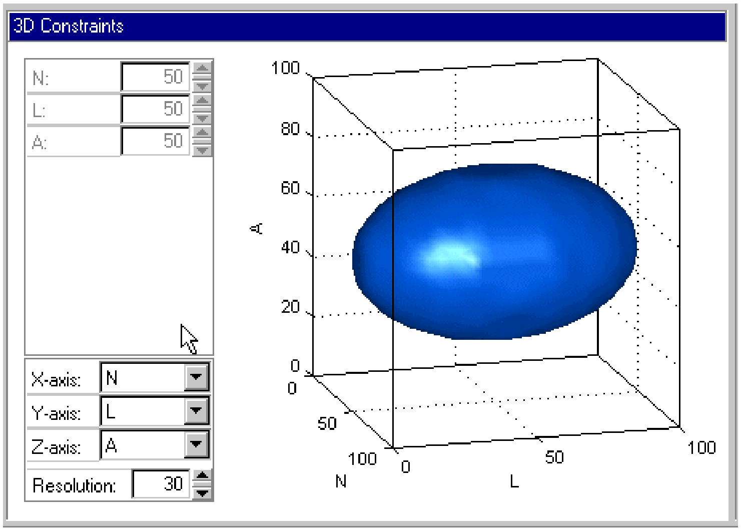
Constraint options

Values are in coded units

	Xc	N	L	A
N	0	3		
L	0	0	1	
A	0	0	0	3

This reduces the space available for the candidate set by a third in the A and N axes, forming an ellipsoid, as shown below. The L axis, left at 1, is not constrained at the midpoint of N and A. To leave L unconstrained (a cylinder), put the value of L=0.

11 Click **OK**, click **OK** again in the Constraint Manager, and click **Replace** to compensate for design points lost outside the new candidate set. Examine the new constraint 3-D plot illustrated.



Both constraints are applied to this design, but the ellipsoid lies entirely within the previous 1-D table constraint.

Saving Designs

To save your design:

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

There are three **Export to** options:

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

2 Choose a Design Editor file.

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

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

7

Tutorial: Data Editor

This section discusses the following topics:

Introduction to the Data Editor

The Data Editor is a GUI for loading data, creating new variables, and creating constraints for that data.

Data can be loaded from files (Excel files, MATLAB files, text files) and from the MATLAB workspace. You can merge data in any of these forms with previously loaded data sets (providing there is no conflict in the form of the data) to produce a new data set. Test plans can use only one data set, so the merging function allows you to combine records and variables from different files in one model.

You can define new variables, apply filters to remove unwanted data, and apply test notes to filtered tests. You can store and retrieve these user-defined variables and filters for any data set, and you can store plot settings. You can change and add records and apply test groupings, and you can match data to designs. You can also write your own data loading functions, see "Data Loading Application Programming Interface" in the "Data" section.

For comprehensive help on all data functions in the Model Browser, see "Data".

The following tutorial is a step-by-step guide to the following:

- Loading data from an Excel file
- Viewing and editing the data
- Creating a user-defined variable
- Applying a filter to the data
- Sequence of variables
- Deleting and editing variables and filters
- Placing user-defined variables and filters into storage
- Defining test groupings
- Matching data to experimental designs

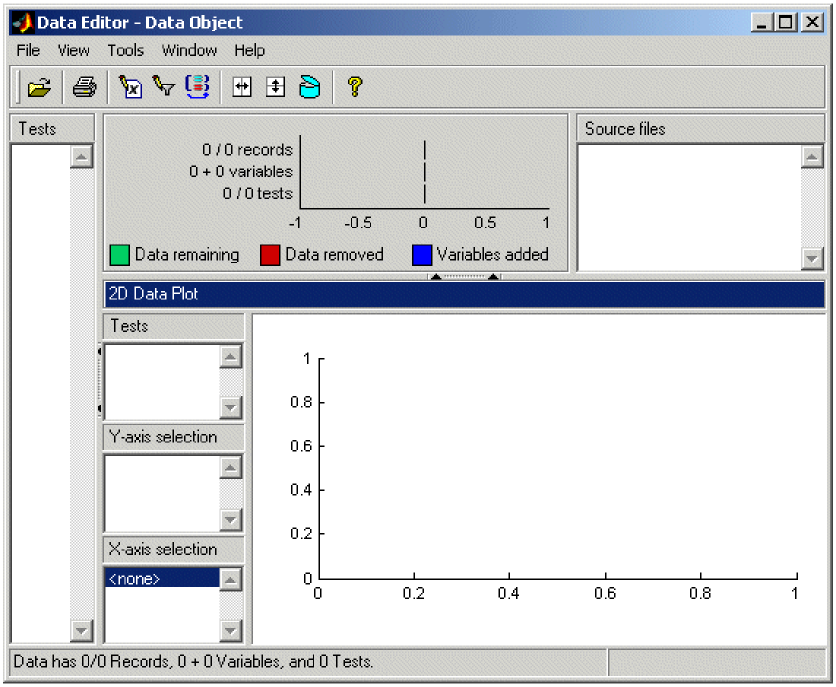
Loading the Data

Entering the Data Editor

You can create, copy, rename and delete data objects from the Project view in the Model Browser.

To enter the Data Editor and create a new data object, from the Project node, select **Data > New Data** (or click the New Data Object toolbar button).

The Data Editor appears.



Data has 0/0 Records, 0 + 0 Variables, and 0 Tests.

There are no plots until some data has been loaded. The views shown depend on whether you have previously looked at the Data Editor, as it retains memory of your previous layout.

By default the new data object is called `Data Object`. Select **File > Rename Data** to enter a new name.

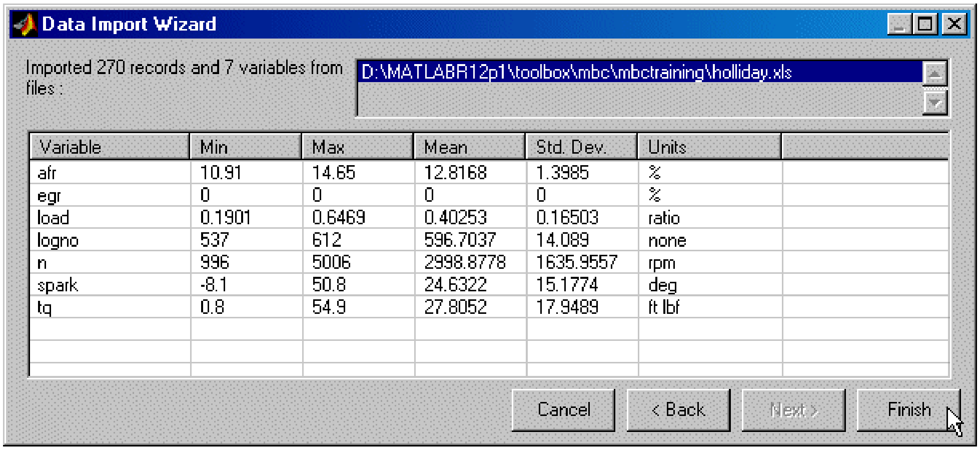
Loading a Data File

1 Click the Open File icon in the toolbar 📂 to load data from a file.

The Data Import Wizard appears to select a file.

2 Use the Browse button to find and select the `Holliday.xls` data file in the `mbctraining` folder. Double-click to load the file. You can also enter the file pathname in the edit box. The pop-up menu contains the file types recognized by the Model Browser (`Excel File`, `Delimited Text File`, `MATLAB Data File`). Leave this at the default, `Auto`. This setting tries to determine what type of file is selected by looking at the file extension.

3 Click **Next**.



Data Import Wizard

Imported 270 records and 7 variables from files :

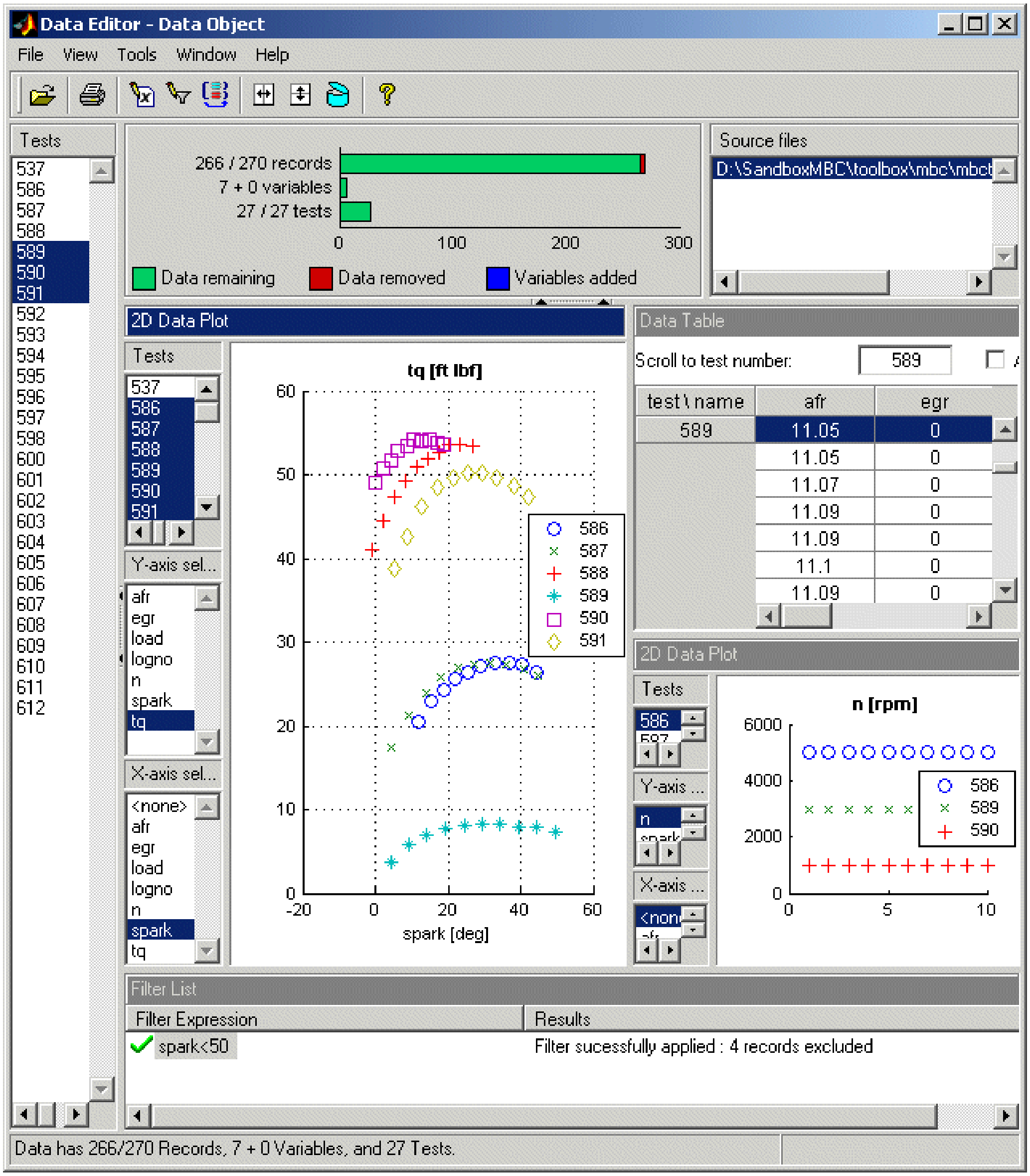
D:\MATLABR12p1\toolbox\mbc\mbctraining\holliday.xls

Variable	Min	Max	Mean	Std. Dev.	Units
afr	10.91	14.65	12.8168	1.3985	%
egr	0	0	0	0	%
load	0.1901	0.6469	0.40253	0.16503	ratio
logno	537	612	596.7037	14.089	none
n	996	5006	2998.8778	1635.9557	rpm
spark	-8.1	50.8	24.6322	15.1774	deg
tq	0.8	54.9	27.8052	17.9489	ft lbf

4 The Data Import Wizard displays a summary screen showing the total number of records and variables imported, and you can view each variable's range, mean, standard deviation, and units in the list box. You can double-click variables in the list to edit names and units. Click **Finish** to accept the data. (If you have data loaded already, you cannot click **Finish** but must continue to the data merging functions.)

The Data Import Wizard disappears and the view returns to the Data Editor, which now contains the data you just loaded.

Viewing and Editing the Data



Source files: D:\SandboxMBC\toolbox\mbc\mbct

Scroll to test number: 589 □

test \ name	afr	egr
589	11.05	0
	11.05	0
	11.07	0
	11.09	0
	11.09	0
	11.1	0
	11.09	0

Filter Expression	Results
✓ spark<50	Filter sucessfully applied : 4 records excluded

Data has 266/270 Records, 7 + 0 Variables, and 27 Tests.

As you can see from the example you can split the views to display several plots at once, as in the Design Editor. You can use the right-click context menus, the toolbar buttons, or the **View** menu to split views. You can choose 2-D plots, 3-D plots, multiple data plots, cluster plots, data tables, and list views of filters, variables, test filters, test notes and cluster information.

In the 2-D plot view, the list boxes on the left allow a combination of tests and variables to be plotted simultaneously. The example shown plots torque against spark for multiple tests on the left, and speed for three selected tests on the right. You can multiple-select tests and *y*-axes to compare the data in the tests (hold down **Shift** or **Control**).

You can use test notes to investigate problem data and decide whether some points should be removed before modeling. The following steps cover using notes and views to sort and investigate your data.

Using Notes to Sort Data for Plotting

1. Right-click a view and select **Current View > Multiple Data Plot**.

2. Right-click the new view and select **Viewer Options > Add Plot**.

   The Plot Variables Setup dialog appears.

3. Select `spark` and click to add to the `X Variable` box, then select `tq` and click to add to the `Y Variable` box. Click **OK** to create the plot.

4. Click in the **Tests** list to select a test to plot (or **Shift**-click, **Ctrl**-click, or click and drag to select multiple tests).

5. Right-click the view and select **Split View > Test Note Definitions**.

   The current view is divided into two.

6. Select **Tools > Test Notes > Add**.

   The Test Note Editor appears.

7. Enter `mean(tq)<10` in the top edit box to define the tests to be noted, and enter `Low torque` in the Test Note edit box. Leave the note color at the default and click **OK**.

8 Right-click the **Test Notes List** view and select **Split View > Notes View**.

The current view is split into two. You can sort records by notes in the new **Notes** view.

9 Click the column header of the new `Low torque` note in the **Notes** view. All the tests that satisfy the condition mean(tq)<10 are sorted to the top of the list.

10 Now create some more views.

- Right-click a view and select **Split View > Data Table**.
- Right-click a view and select **Split View > 3D Data Plot**.

11 In the **Notes** view, click particular tests with the `Low torque` note.

Notice that when you select a test here, the same test is plotted in the multiple data plots, the 3D data plot, and highlighted in the data table. You can use the notes in this way to easily identify problem tests and decide whether you should remove them.

Removing Outliers and Problem Tests

1 Click a point on the **Multiple Data Plots** view.

The point is outlined in red on the plot, and highlighted in the data table. You can remove points you have selected as outliers by selecting **Tools > Filters > Remove Outliers** (or use the keyboard shortcut **Ctrl+A**). Select **Tools > Filters > Restore Outliers** (or use the keyboard shortcut **Ctrl+Z**) to open a dialog where you can choose to restore any or all removed points.

You can remove individual points as outliers, or you can remove records or entire tests with filters.

2 For example, after examining all the `Low torque` noted tests, you could decide they should be filtered out.

a Select **Split View > Test Filter Definitions**.

b Select **Tools > Test Filters > Add**.

c The Test Filter Editor appears. Enter `mean(tq)>10` to keep all tests where the mean torque is greater than 10, and click **OK**.

In the new **Test Filter List** view, you should see the new test filter successfully applied and the number of records removed.

Similarly, you can use filters to remove individual records rather than entire tests, which you will cover in a later section "Applying a Filter" on page 7-13.

3 To view removed data in the table view, right-click and select **Viewer Options > Allow Editing**. Removed records are red. To view removed data in the 2-D and Multiple Data Plots, select **Viewer Options > Properties** and select the box **Show bad data**.

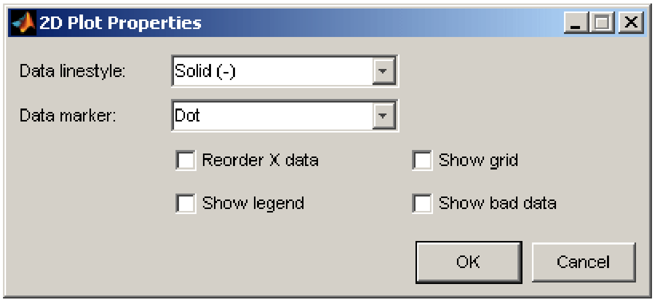
Reordering and Editing Data

To change the display, right-click a 2-D plot and select **Viewer Options > Properties**. You can alter grid and plot settings including lines to join the data points.

Reorder X Data in the Plot Properties dialog can be useful when record order does not produce a sensible line joining the data points. For an illustration of this:

1 Ensure you are displaying a 2-D plot. You can right-click on any plot and select **Current Plot > 2-D Plot**, or use the context menu split commands to add new views.

2 Right-click on a 2-D plot and select **Viewer Options > Properties** and choose `solid` from the **Data Linestyle** drop-down menu, as shown below. Click **OK**.



Data linestyle: Solid (-)

Data marker: Dot

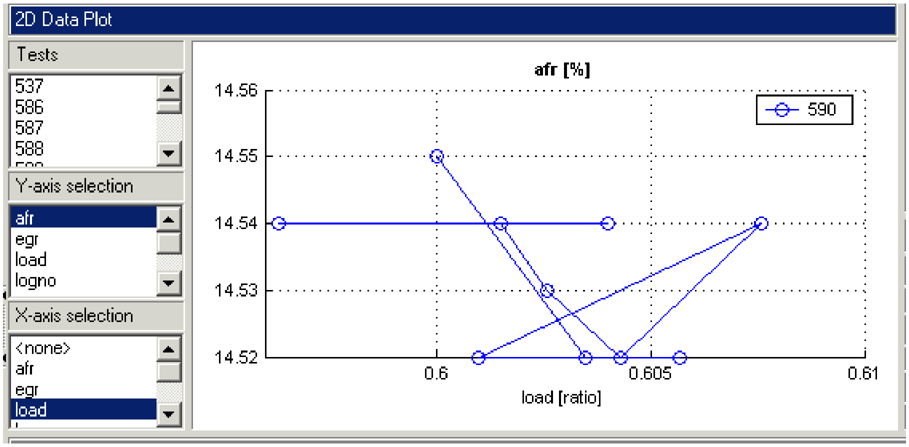
□ Reorder X data □ Show grid

□ Show legend □ Show bad data

3 Choose `afr` for the *y*-axis.

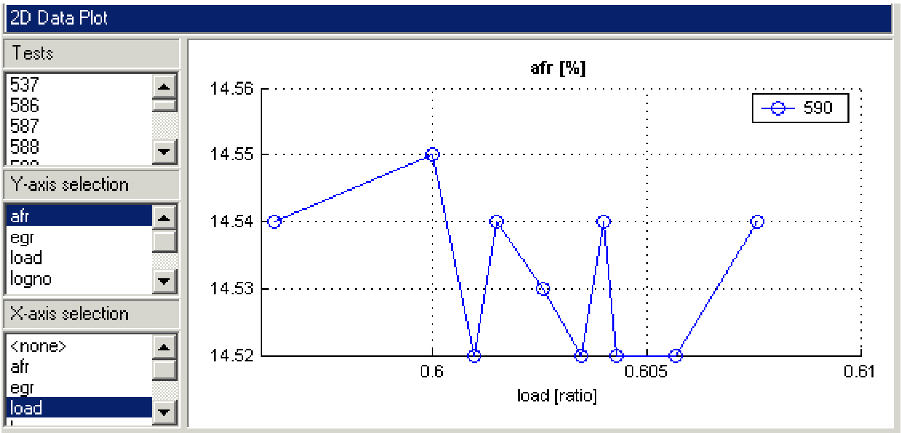
4 Choose `Load` for the *x*-axis.

5 Select test 590. You must use the test controls contained within the 2-D plot. The **Tests** pane on the left applies to other views: tables and 3-D and multiple data plots.



6 Right-click and select **Viewer Options > Properties** and choose **Reorder X Data**. Click **OK**.

This command replots the line from left to right instead of in the order of the records, as shown.



7 Right-click and select **Split Plot > Data Table** to split the currently selected view and add a table view. You can select particular test numbers in the **Tests** pane on the left of the Data Editor. You can right-click to select **Viewer Options > Allow Editing**, and then you can double-click cells to edit them.

User-Defined Variables and Filtering

You can add new variables to the data set, and you can remove records by imposing constraints on the data.

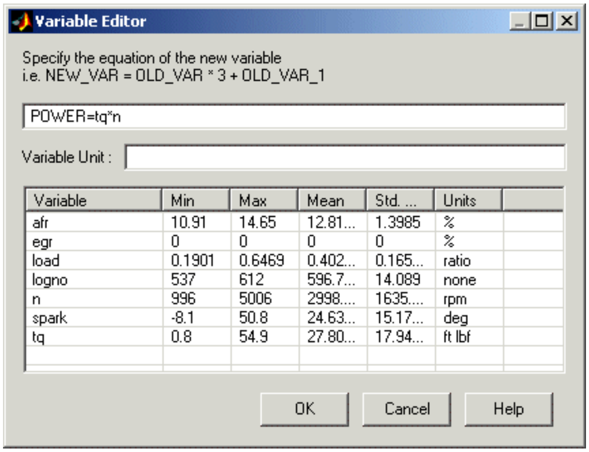
1 Select **Tools > Variables > Add**.

   Alternatively, click the toolbar button.

   The Variable Editor appears.

   You can define new variables in terms of existing variables. You define the new variable by writing an equation in the edit box at the top of the Variable Editor dialog.

2 Define a new variable called `POWER` that is defined as the product of two existing variables, `tq` and `n`, by entering `POWER=tq*n`, as seen in the example following. You can also double-click variable names and operators to add them, which can be useful to avoid typing mistakes in variable names, which must be exact including case.



Specify the equation of the new variable
i.e. NEW_VAR = OLD_VAR * 3 + OLD_VAR_1

POWER=tq*n

Variable Unit :

Variable	Min	Max	Mean	Std. ...	Units
afr	10.91	14.65	12.81...	1.3985	%
egr	0	0	0	0	%
load	0.1901	0.6469	0.402...	0.165...	ratio
logno	537	612	596.7...	14.089	none
n	996	5006	2998....	1635....	rpm
spark	-8.1	50.8	24.63...	15.17...	deg
tq	0.8	54.9	27.80...	17.94...	ft lbf

3 Click **OK** to add this variable to the current data set.

4 This new variable can be seen in the Data Editor by right-clicking in a view and selecting **Split Plot > Variable Definitions**. A new view appears

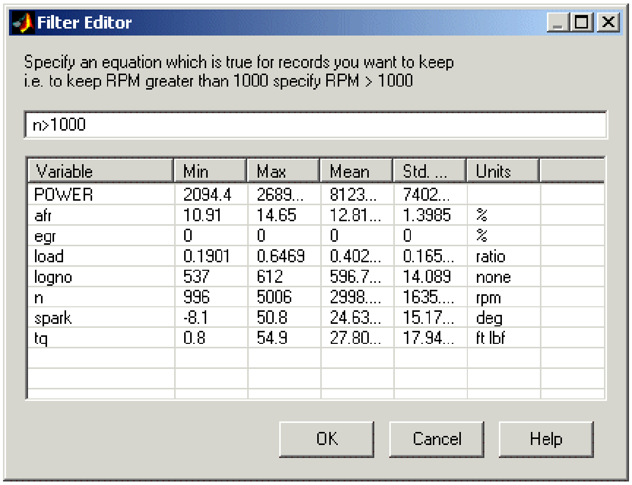
containing a list of your user-defined variables. You can also now see `7 + 1 variables` next to the top information bars.

Applying a Filter

A filter is a constraint on the data set used to exclude some records. You use the Filter Editor to create new filters.

1. Choose **Tools > Filters > Add**, or click the ▽ button in the Data Editor window.

   The Filter Editor dialog appears.

   Specify an equation which is true for records you want to keep
   i.e. to keep RPM greater than 1000 specify RPM > 1000

   n>1000

Variable	Min	Max	Mean	Std. ...	Units
POWER	2094.4	2689...	8123...	7402...	
afr	10.91	14.65	12.81...	1.3985	%
egr	0	0	0	0	%
load	0.1901	0.6469	0.402...	0.165...	ratio
logno	537	612	596.7...	14.089	none
n	996	5006	2998....	1635....	rpm
spark	-8.1	50.8	24.63...	15.17...	deg
tq	0.8	54.9	27.80...	17.94...	ft lbf

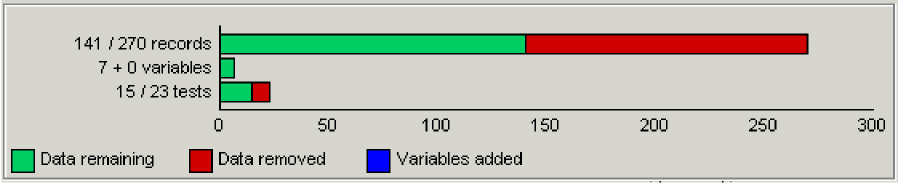
   You define the filter using logical operators on the existing variables.

2. Keep all records with speed (`n`) greater than 1000. Type `n` (or double-click on the variable `n`), then type `>1000`.

3. Click **OK** to impose this filter on the current data set.

4. This new filter can be seen in the Data Editor by right-clicking in a view (try the **Variable List** view) and selecting **Split Plot > Filter List**. A new view appears containing a list of your user-defined filters and information on how many records are removed by the new filter. You can also now

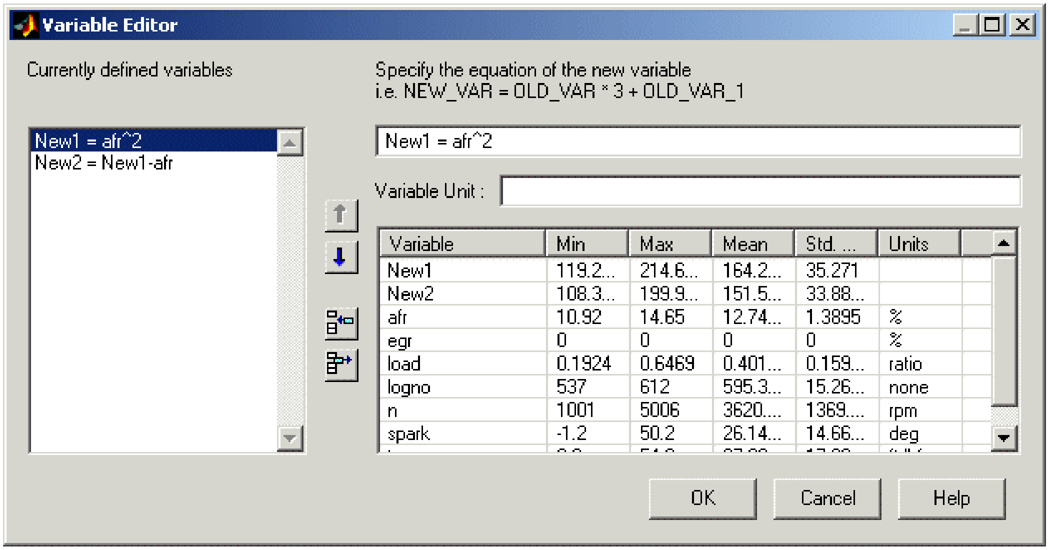
see `141/270 records` next to the top information bars and a red section illustrating the records removed by the filter.



Sequence of Variables

You can change the order of user-defined variables in the Variable Editor list using the arrow buttons.

Select **Tools > Variables > Edit** to open the Variable Editor.



Currently defined variables

New1 = afr^2
New2 = New1-afr

Specify the equation of the new variable
i.e. NEW_VAR = OLD_VAR * 3 + OLD_VAR_1

New1 = afr^2

Variable Unit :

Variable	Min	Max	Mean	Std. ...	Units
New1	119.2...	214.6...	164.2...	35.271	
New2	108.3...	199.9...	151.5...	33.88...	
afr	10.92	14.65	12.74...	1.3895	%
egr	0	0	0	0	%
load	0.1924	0.6469	0.401...	0.159...	ratio
logno	537	612	595.3...	15.26...	none
n	1001	5006	3620....	1369....	rpm
spark	-1.2	50.2	26.14...	14.66...	deg

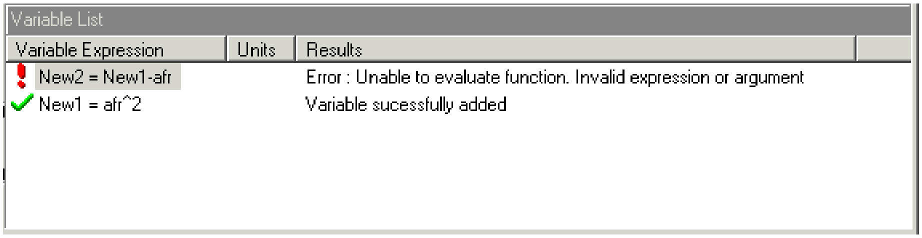
Example:

1 Define two new variables, `New1` and `New2`. Note that you can use the buttons to add or remove a list item to create or delete variables in this

view. Click the button to 'Add item' to add a new variable, and enter the definitions shown.

Notice that `New2` is defined in terms of `New1`. New variables are added to the data in turn and hence `New1` must appear in the list before `New2`, otherwise `New2` is not well defined.

2 Change the order by clicking the down arrow in the Variable Editor to produce this erroneous situation. Click **OK** to return to the Data Editor and in the variable list view you see the following error message:



Variable List

Variable Expression	Units	Results
! New2 = New1-afr		Error : Unable to evaluate function. Invalid expression or argument
✓ New1 = afr^2		Variable sucessfully added

3 Use the arrows to order user-defined variables in legitimate sequence.

Deleting and Editing Variables and Filters

You can delete user-defined variables and filters.

Example:

1 To delete the added variable `New1`, select it in a **Variable List** view and press the **Delete** key.

2 You can also delete variables in the Variable Editor by clicking the Remove Item button.

Similarly, you can delete filters by selecting the unwanted filter in a Filter List view and using the **Delete** key.

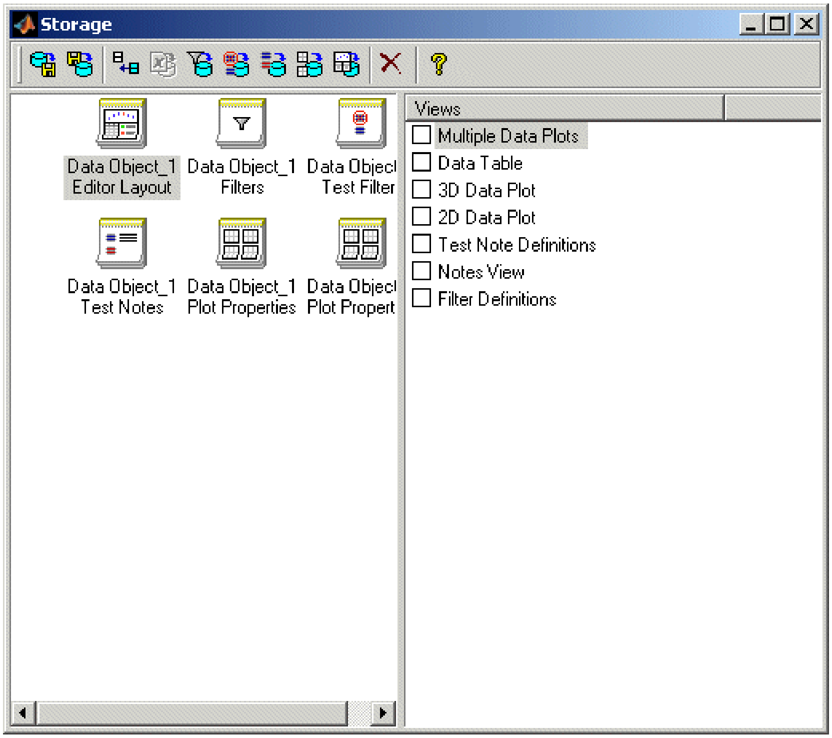
You can also edit current user-defined variables and filters using the relevant menu items or toolbar buttons.

Storage

Storage allows you to store plot preferences, user-defined variables, filters, and test notes so they can be applied to other data sets loaded later in the session, and to other sessions.

You can open the Storage window from the Data Editor window in either of these ways:

- Using the menu **Tools > Open Storage**
- Using the toolbar button



The example above contains a variety of stored objects. The toolbar buttons Store Current Variables and Store Current Filters, Test Filters or Test Notes allow you to put all user-defined variables and filters from the current session into storage. They appear in the Storage window. All stored user-defined variables and filters appear here regardless of which project is open — once created and brought into storage, they remain there. If you do not delete them, they are there indefinitely. You can also store view settings with the toolbar button Store Current Data Editor Layout.

The Data Editor retains memory of your plot type settings and when reopened will display the same types of views. You can also use Store Current Data Plots to save the details of your Multiple Data Plots, such as which factors to display, line style, grid, etc.

You can double-click any item in storage to append the object to the current views. For example if you double-click a Data Editor Layout object, the current views will be replaced by the saved views. Other objects add items to the current views.

You can select Export to File to send the stored objects to a file. You might do this to move the objects to a different user or machine. Select Import from File to bring such variables and filters into storage, and use Append Stored Object to add items from storage to your current project.

1 Use the controls to bring the variable `POWER` and the filter you just created into storage.

2 Close the Storage window.

For a detailed description of the functionality in Storage, see “Storage”.

Test Groupings

The Define Test Groupings dialog collects records of the current data object into groups; these groups are referred to as *tests*.

The dialog is accessed from the Data Editor in either of these ways:

- Using the menu **Tools > Change Test Groupings**
- Using the toolbar button

When you enter the dialog, a plot is displayed as the variable `logno` is automatically selected for grouping tests.

Select another variable to use in defining groups within the data.

1 Select `n` in the **Variables** list.

2 Click the button to add the variable (or double-click `n`).

   The variable `n` appears in the list view on the left as seen in the following example. You can now use this variable to define groups in the data. The maximum and minimum values of `n` are displayed. The **Tolerance** is used to define groups: on reading through the data, when the value of `n` changes by more than the tolerance, a new group is defined. You change the **Tolerance** by typing directly in the edit box.

   You can define additional groups by selecting another variable and choosing a tolerance. Data records are then grouped by `n` or by this additional variable changing outside their tolerances.

3 Clear the box **Group by** for `logno`. Notice that variables can be plotted without being used to define groups.

4 Add `load` to the list by selecting it on the right and clicking .

5 Change the `load` tolerance to 0.01 and watch the test grouping change in the plot.

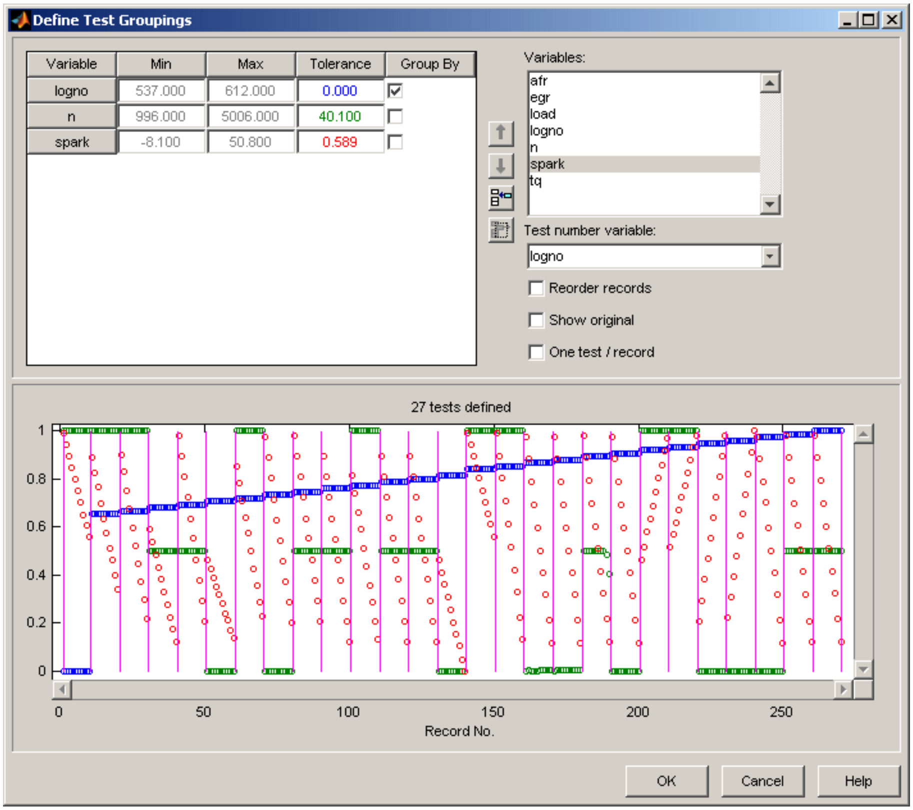
6 Clear the **Group By** check box for `load`. Now this variable is plotted without being used to define groups.

The plot shows the scaled values of all variables in the list view (the color of the tolerance text corresponds to the color of data points in the plot). Vertical pink bars show the tests (groups). You can zoom the plot by **Shift**-click-dragging or middle-click-dragging the mouse; zoom out again by double-clicking.

7 Select `load` in the list view (it becomes highlighted in blue) and remove it from the list by clicking the button.

8 Double-click to add `spark` to the list, and clear the **Group By** check box. Select `logno` as the only grouping variable.

It can be helpful to plot the local model variable (in this case spark) to check you have the correct test groupings, as shown below. The plot shows the sweeps of spark values in each test while speed (n) is kept constant. Speed is only changed between tests, so it is a global variable. Try zooming in on the plot to inspect the test groups; double-click to reset.



Define Test Groupings

Variable	Min	Max	Tolerance	Group By
logno	537.000	612.000	0.000	☑
n	996.000	5006.000	40.100	□
spark	-8.100	50.800	0.589	□

Variables:
afr
egr
load
logno
n
spark
tq

Test number variable: logno

□ Reorder records

□ Show original

□ One test / record

27 tests defined

Reorder records allows records in the data set to be reordered before grouping. Otherwise, the groups are defined using the order of records in the original data object.

Show original displays the original test groupings if any were defined.

One test/record defines one test per record, regardless of any other grouping. This is required if the data is to be used in creating one-stage models.

Test number variable contains a pop-up menu showing all the variables in the current data set. Any of these could be selected to number the tests.

9 Make sure `logno` is selected for the **Test number variable**.

This changes how the tests are displayed in the rest of the Model Browser. Test number can be a useful variable for identifying individual tests in Model Browser and Data Editor views (instead of 1,2,3...) if the data was taken in numbered tests and you want access to that information during modeling.

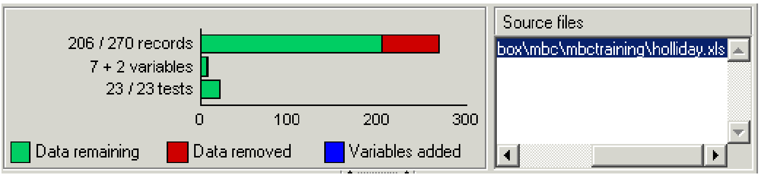
If you chose `none` from the **Test number variable** list, the tests would be numbered 1,2,3 and so on in the order in which the records appear in the data file. With `logno` chosen, you will see tests in the Data Editor listed as 586, 587 etc.

Every record in a test must share the same test number to identify it, so when you are using a variable to number tests, the value of that variable is taken in the first record in each test.

Test numbers must be unique, so if any values in the chosen variable are the same, they are assigned new test numbers for the purposes of modeling (this does not change the underlying data, which retains the correct test number or other variable).

10 Click **OK** to accept the test groupings defined and dismiss the dialog.

You return to the Data Editor window. At the top is a summary of this data set now that your new variable has been added and a new filter applied (example shown below).



11 The number of records shows the number of values left (after filtration) of each variable in this data set, followed by the original number of records. The color coded bars also display the number of records removed as a proportion of the total number. The values are collected into a number of tests; this number is also displayed. The variables show the original number of variables plus user-defined variables.

Matching Data to Designs

We provide an example project to illustrate the process of matching experimental data to designs.

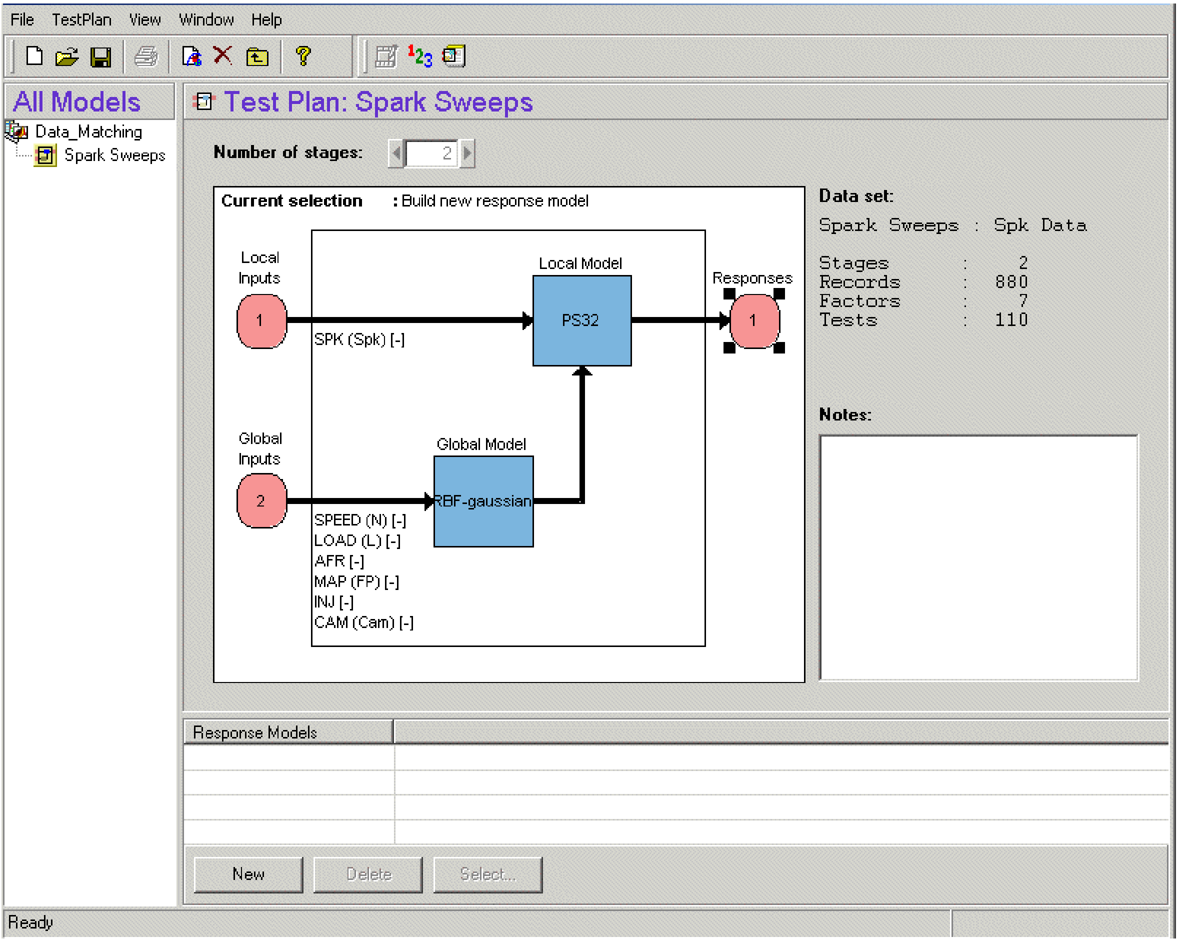
Experimental data is unlikely to be identical to the desired design points. You can use the Cluster Plot view in the Data Editor to compare the actual data collected with your experimental design points. Here you can select data for modeling. If you are interested in collecting more data, you can update your experimental design by matching data to design points to reflect the actual data collected. You can then optimally augment your design (using the Design Editor) to decide which data points it would be most useful to collect, based on the data obtained so far.

You can use an iterative process: make a design, collect some data, match that data with your design points, modify your design accordingly, then collect more data, and so on. You can use this process to optimize your data collection process in order to obtain the most robust models possible with the minimum amount of data.

1 To see the data matching functions, select **File > Open Project** and browse to the file `Data_Matching.mat` in the `mbctraining` directory.

2 Click the `Spark Sweeps` node in the model tree to change to the test plan view, as shown.

   Here you can see the two-stage test plan with model types and inputs set up. The global model has an associated experimental design (which you could view in the Design Editor). You are going to use the Data Editor to examine how closely the data collected so far matches to the experimental design.



Number of stages: 2

Current selection : Build new response model

Data set:

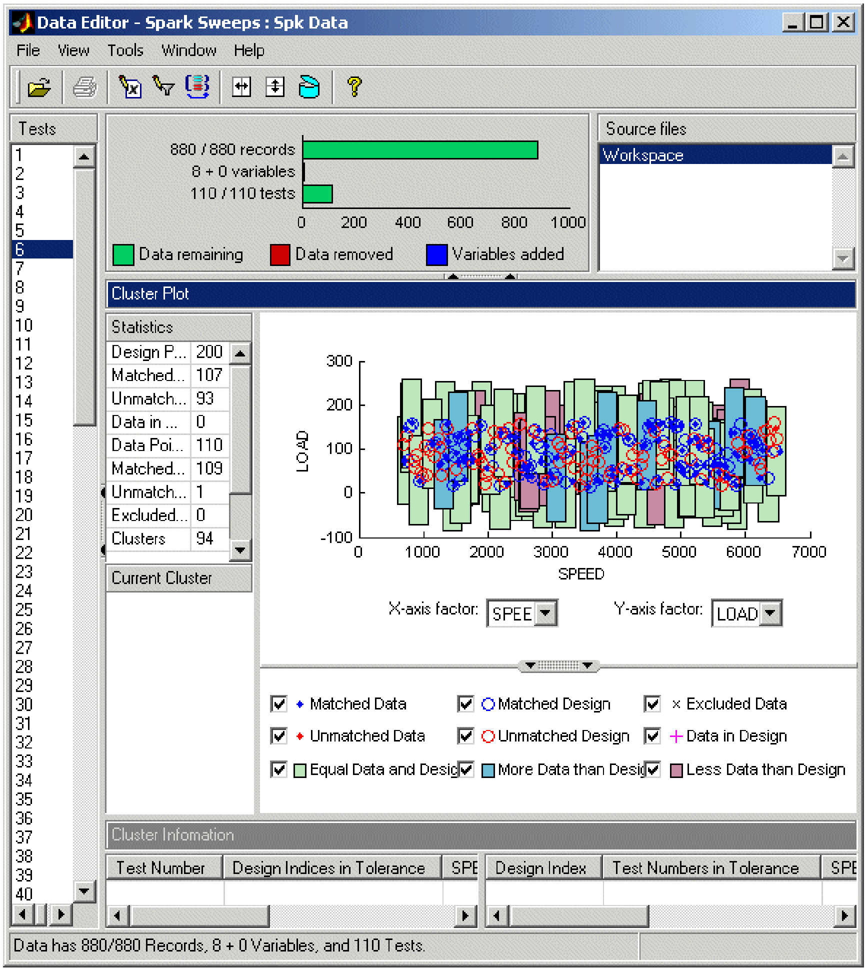
```
Spark Sweeps : Spk Data

Stages       :    2
Records      :  880
Factors      :    7
Tests        :  110
```

Ready

3 Click the Select Data button (123) in the toolbar.

The Data Editor appears.

4 You need a **Cluster View** to examine design and data points. Right-click a view in the Data Editor and select **Current View > Cluster View**.



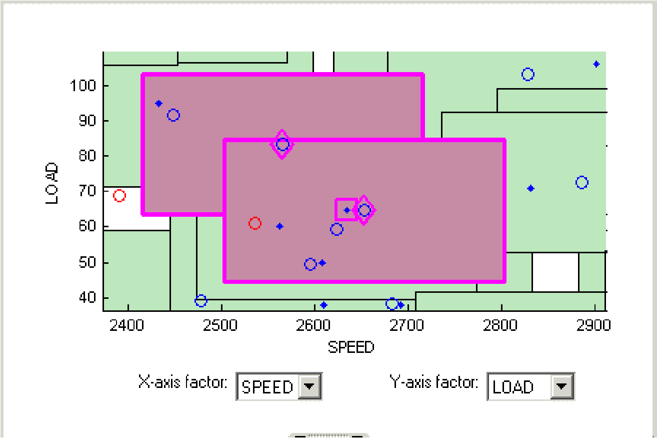
Design P...	200
Matched...	107
Unmatch...	93
Data in ...	0
Data Poi...	110
Matched...	109
Unmatch...	1
Excluded...	0
Clusters	94

X-axis factor: SPEE Y-axis factor: LOAD

Data has 880/880 Records, 8 + 0 Variables, and 110 Tests.

In the Cluster Plot you can see colored areas containing points. These are “clusters” where closely matching design and data points have been selected by the matching algorithm.

Tolerance values (derived initially from a proportion of the ranges of the variables) are used to determine if any data points lie within tolerance of each design point. Data points that lie within tolerance of any design point are matched to that cluster. Data points that fall inside the tolerance of more than one design point form a single cluster containing all those design and data points. If no data points lie within tolerance of a design point, it remains unmatched and no cluster is plotted.



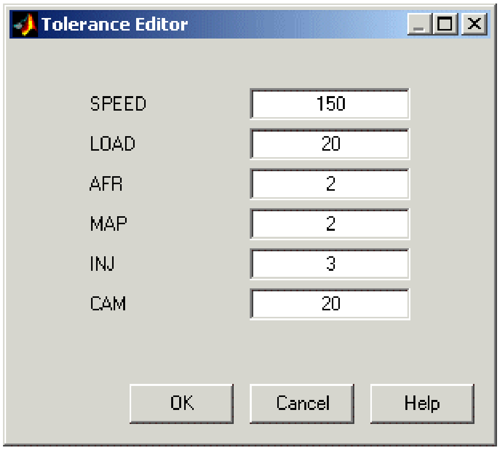
Notice the shape formed by overlapping clusters. The example shown outlined in pink is a single cluster formed where a data point lies within tolerance of two design points.

Note that on this plot you can see other unselected points that appear to be contained within this cluster. You need to track points through other factor dimensions using the axis controls to see where points are separated beyond tolerance. You will do this in a later step of this tutorial, “Understanding Clusters” on page 7-28.

Tolerances and Cluster Information

1 To edit tolerance values, select **Tools > Tolerances**.

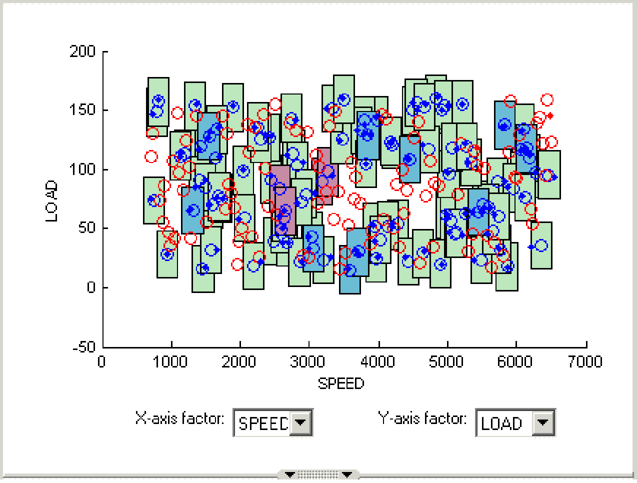
The Tolerance Editor appears. Here you can change the size of clusters in each dimension. Observe that the LOAD tolerance value is currently 100. This accounts for the elongated shape (in the LOAD dimension) of the clusters in the current plot, because this tolerance value is a high proportion of the total range of this variable.



SPEED 150
LOAD 20
AFR 2
MAP 2
INJ 3
CAM 20

2 Click the **LOAD** edit box and enter `20`, as shown. Click **OK**.

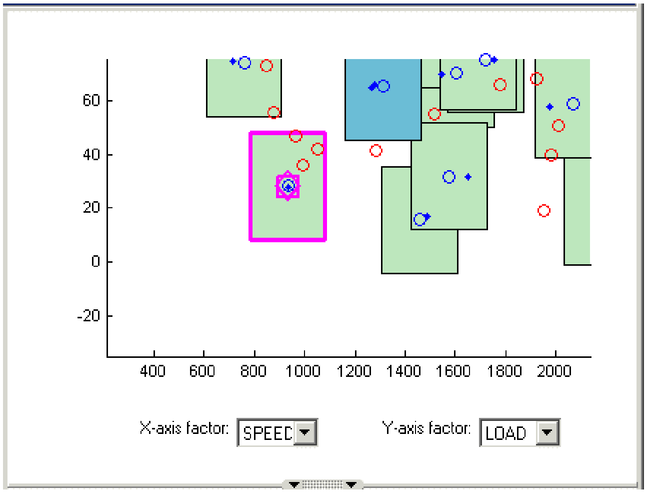
Notice the change in shape of the clusters in the **Cluster Plot** view.



X-axis factor: SPEED

Y-axis factor: LOAD

3 Shift click (center-click) and drag to zoom in on an area of the plot, as shown. You can double-click to return to the full size plot.

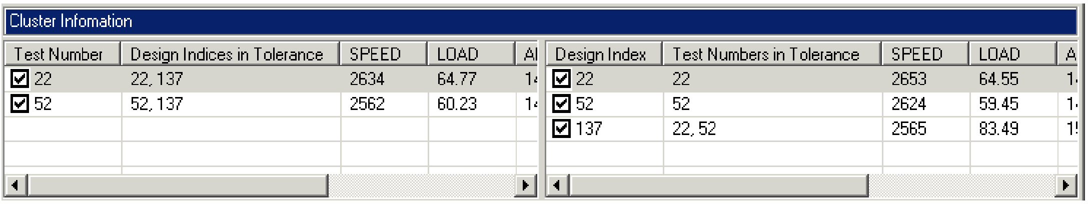


X-axis factor: SPEEC ▾ Y-axis factor: LOAD ▾

4 Click a cluster to select it. Selected points or clusters are outlined in pink. If you click and hold, you can inspect the values of global variables at the selected points (or for all data and design points if you click on a cluster). You can use this information to help you decide on suitable tolerance values if you are trying to match points.

You need to ensure you are displaying a **Cluster Information** list view to select or exclude points. The Data Editor retains memory of previous data views and if you had a cluster plot in your saved settings then this plot is used.

5 If you do not already have a Cluster Information list view displayed, right-click the **Cluster Plot** view and select **Split Vertically**. A new view appears underneath the cluster plot. Right-click the new view and select **Current Plot > Cluster Information**.



Cluster Information

Test Number	Design Indices in Tolerance	SPEED	LOAD	Al
☑ 22	22, 137	2634	64.77	14
☑ 52	52, 137	2562	60.23	14

Design Index	Test Numbers in Tolerance	SPEED	LOAD	A
☑ 22	22	2653	64.55	1
☑ 52	52	2624	59.45	1
☑ 137	22, 52	2565	83.49	1

Notice that the **Cluster Information** list view shows the details of all data and design points contained in the selected cluster. You use the check boxes here to select or exclude data or design points. Click different clusters to see a variety of points. The list view shows the values of global variables at each point, and which data and design points are within tolerance of each other. Your selections here determine which data will be used for modeling, and which design points will be replaced by actual data points.

Note All data points with a selected check box will be used for modeling. All data points with a cleared check box will be removed from the data set, and not seen in any other views. This cluster view is the only place you can restore these excluded data to the data set.

Understanding Clusters

If you are not interested in collecting more data, then there is no need to make sure the design is modified to reflect the actual data. All data (except those you exclude by clearing the check boxes) will be used for modeling.

However, if you want your new design (called `Actual Design`) to accurately reflect what data has been obtained so far, for example to collect more data, then the cluster matching is important. All data points with a selected check box will be added to the new `Actual Design`, except those in red clusters. The color of clusters indicates what proportion of selected points it contains as follows:

- Green clusters have equal numbers of selected design and selected data points. The data points will replace the design points in the `Actual Design`.

  Note that the color of all clusters is determined by the proportion of *selected* points they contain; excluded points (with cleared check boxes) have no effect. Your check box selections can change cluster color.
- Blue clusters have more data points than design points. All the data points will replace the design points in the `Actual Design`.
- Red clusters have more design points than data points. These data points will not be added to your design as the algorithm cannot choose which design points to replace, so you must manually make selections to deal with red clusters if you want to use these data points in your design. The

example **Cluster Information** list view shows a selected red cluster with more design than data points.

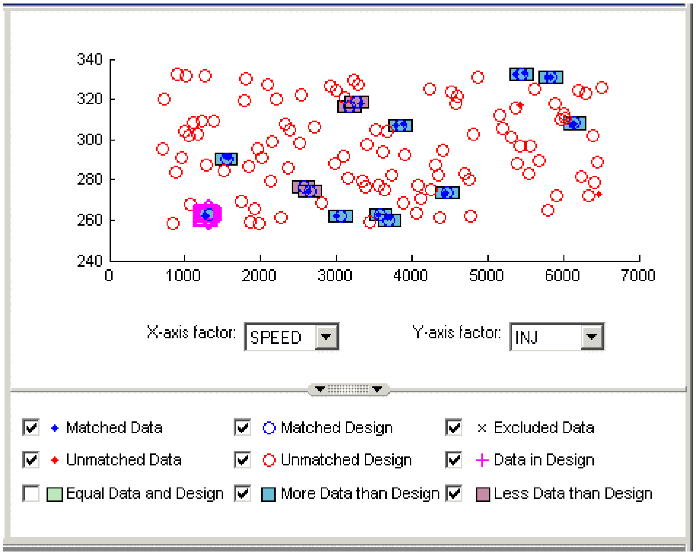
If you don't care about the `Actual Design` (for example, if you do not intend to collect more data) and you are just selecting data for modeling, then you can ignore red clusters. The data points in red clusters are selected for modeling.

1. Right-click the **Cluster Plot** and select **Viewer Options > Select Unmatched Data**. Notice that the remaining unmatched data points appear in the **Cluster Information** list view. Here you can use the check boxes to select or exclude unmatched data in the same way as points within clusters.

2. Select a cluster, then use the drop-down menu to change the **Y-Axis factor** to `INJ`. Observe the selected cluster now plotted in the new factor dimensions of `SPEED` and `INJ`.

   You can use this method to track points and clusters through the dimensions. This can give you a good idea of which tolerances to change in order to get points matched. Remember that points that do not form a cluster may appear to be perfectly matched when viewed in one pair of dimensions; you must view them in other dimensions to find out where they are separated beyond the tolerance value. You can use this tracking process to decide whether you want particular pairs of points to be matched, and then change the tolerances until they form part of a cluster.

3. Clear the **Equal Data and Design** check box in the **Cluster Plot** view. You control what is plotted using these check boxes.



X-axis factor: SPEED

Y-axis factor: INJ

☑ ◆ Matched Data ☑ ○ Matched Design ☑ × Excluded Data

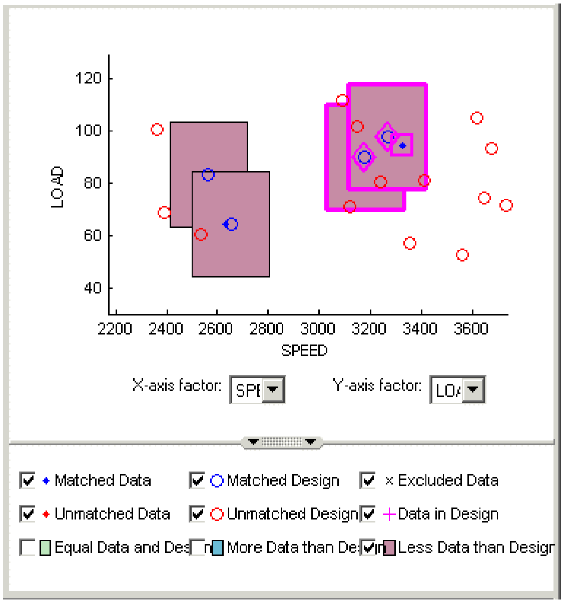
☑ ◆ Unmatched Data ☑ ○ Unmatched Design ☑ + Data in Design

□ 🟩 Equal Data and Design ☑ 🟦 More Data than Design ☑ 🟪 Less Data than Design

This removes the green clusters from view, as shown. These clusters are matched; you are more likely to be interested in unmatched points and clusters with uneven numbers of data and design points. Removing the green clusters allows you to focus on these points of interest. If you want your new `Actual Design` to accurately reflect your current data, your aim is to get as many data points matched up to design points as possible; that is, as few red clusters as possible.

4 Clear the check box for **More Data than Design**. You may also decide to ignore blue clusters, which contain more data points than design points. These design points will be replaced by all data points within the cluster. An excess of data points is unlikely to be a concern.

However, blue clusters may indicate that there was a problem with the data collection at that point, and you may want to investigate why more points than expected were collected.



X-axis factor: SPE Y-axis factor: LO

☑ ✦ Matched Data ☑ ○ Matched Design ☑ × Excluded Data

☑ ✦ Unmatched Data ☑ ○ Unmatched Design ☑ + Data in Design

☐ 🟩 Equal Data and Des ☐ 🟦 More Data than De ☑ 🟪 Less Data than Desigr

5 Select one of the remaining red clusters. Both of these have two design points within tolerance of a single data point.

6 Choose one of the design points to match to the data point, then clear the check box of the other design point. The cleared design point remains unchanged in the design. The selected design point will be replaced by the matched data point.

Notice that the red cluster disappears. This is because your selection results in a cluster with an equal number of selected data and design points (a green cluster) and your current plot does not display green clusters.

7 Repeat for the other red cluster.

Now all clusters are green or blue. There are two remaining unmatched data points.

8 Clear the **Unmatched Design** check box to locate the unmatched data points. Select **Unmatched Design** check box again — you need to see design points to decide if any are close enough to the data points that they should be matched.

9 Locate and zoom in on an unmatched data point. Select the unmatched data point and a nearby design point by clicking, then use the axis drop-down menus to track the candidate pair through the dimensions. Decide if any design points are close enough to warrant changing the tolerance values to match the point with a design point.

10 Recall that you can right-click the **Cluster View** and select **Viewer Options > Select Unmatched Data**to display the remaining unmatched data points in the **Cluster Information** list view. Here you can use the check boxes to select or exclude these points. If you leave them selected, they will be added to the `Actual Design`.

These steps illustrate the process of matching data to designs, to select modeling data and to augment your design based on actual data obtained. Some trial and error is necessary to find useful tolerance values. You can select points and change plot dimensions to help you find suitable values. If you want your new `Actual Design` to accurately reflect your experimental data, you need to make choices to deal with red clusters. Select which design points in red clusters you want to replace with the data points. If you do not, then these data points will not be added to the new design.

When you are satisfied that you have selected all the data you want for modeling, close the Data Editor. At this point, your choices in the cluster plots will be applied to the data set and a new design called `Actual Design` will be created. All the changes are determined by your check box selections for data and design points.

All data points with a selected check box are selected for modeling. Data points with cleared check boxes are excluded from the data set. Changes are made to the existing design to produce the new `Actual Design`. All selected data will be added to your new design, except those in red clusters. Selected data points that have been matched to design points (in green and blue clusters) replace those design points.

All these selected data points become fixed design points (red in the Design Editor) and appear as `Data in Design` (pink crosses) when you reopen the Data Editor.

This means these points will not be included in clusters when matching again. These fixed points will also not be changed in the Design Editor when you add points, though you can unlock fixed points if you want. This can be very useful if you want to optimally augment a design, taking into account the data you have already obtained.

See the reference section “Matching Data to Designs” for more information.

See also the reference section on all aspects of data handling in the toolbox, “Data”.

8

Tutorial: Feature Calibration

This section includes the following topics:

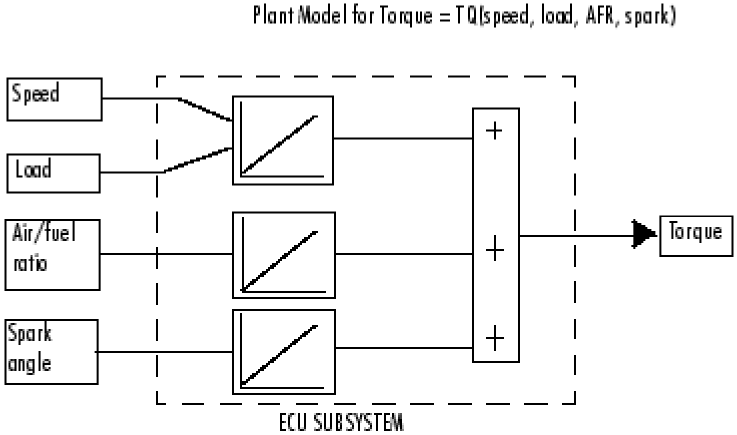
What Are Feature Calibrations?

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.

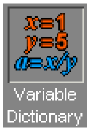
Note If you have a CAGE session open, select **File > New > Project**.

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** view displays all the variables, constants, and formulas in a session. This is empty until you add some variable items.

There are two ways in which you can set up variables:

- Import a variable dictionary
- Add variables and constants to your session

After setting up your variables and constants, you can export the variable dictionary to use in other calibrations.

Importing a Variable Dictionary

To import a variable dictionary,

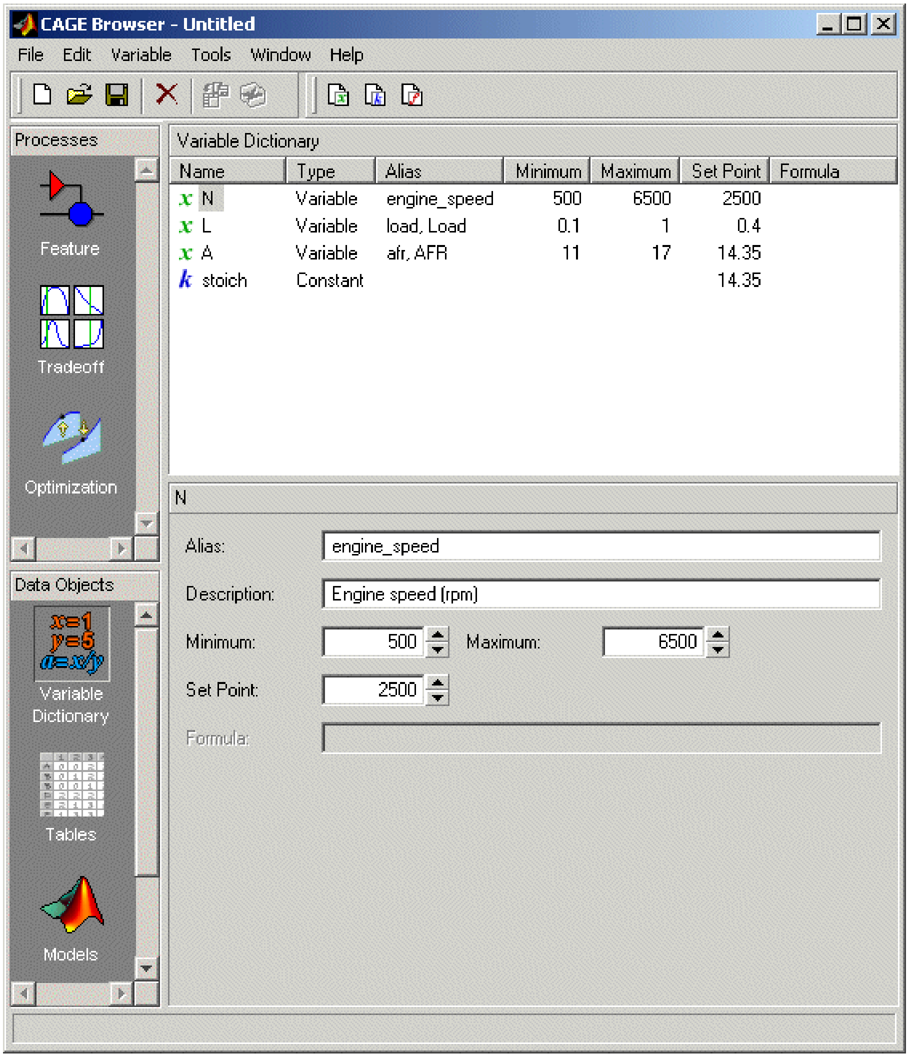
1 Select **File > Import > Variable Dictionary**.

2 Select the `tutorial.xml` file found in `matlab\toolbox\mbc\mbctraining` and click **Open**. CAGE automatically switches to the Variable Dictionary view.

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

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

Your display should resemble the following.



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 New Variable in the toolbar. This adds a new variable to your dictionary.

2 Right-click the new variable and select **Rename** (or press **F2**) to rename the variable.

3 Enter `SPK` as the name.

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

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

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

1 Enter `S, spark` in the **Alias** edit box.

2 Enter `Spark advance (deg)` in the **Description** edit box.

Note The **Variable Dictionary** is case sensitive: `s` and `S` are different.

The variable dictionary enables you to specify a preferred value for a variable. For example, in the preferred value of the variable, `AFR` is set as the stoichiometric constant 14.35.

1 Select `SPK` and enter `25` in the **Set Point** edit box to specify the preferred value.

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 the behavior of torque with varying spark angle, air/fuel ratio, engine speed, and load.

Importing a Model

To import a model built using the Model Browser,

1 Select **File > Import > Model**, which opens a file browser.

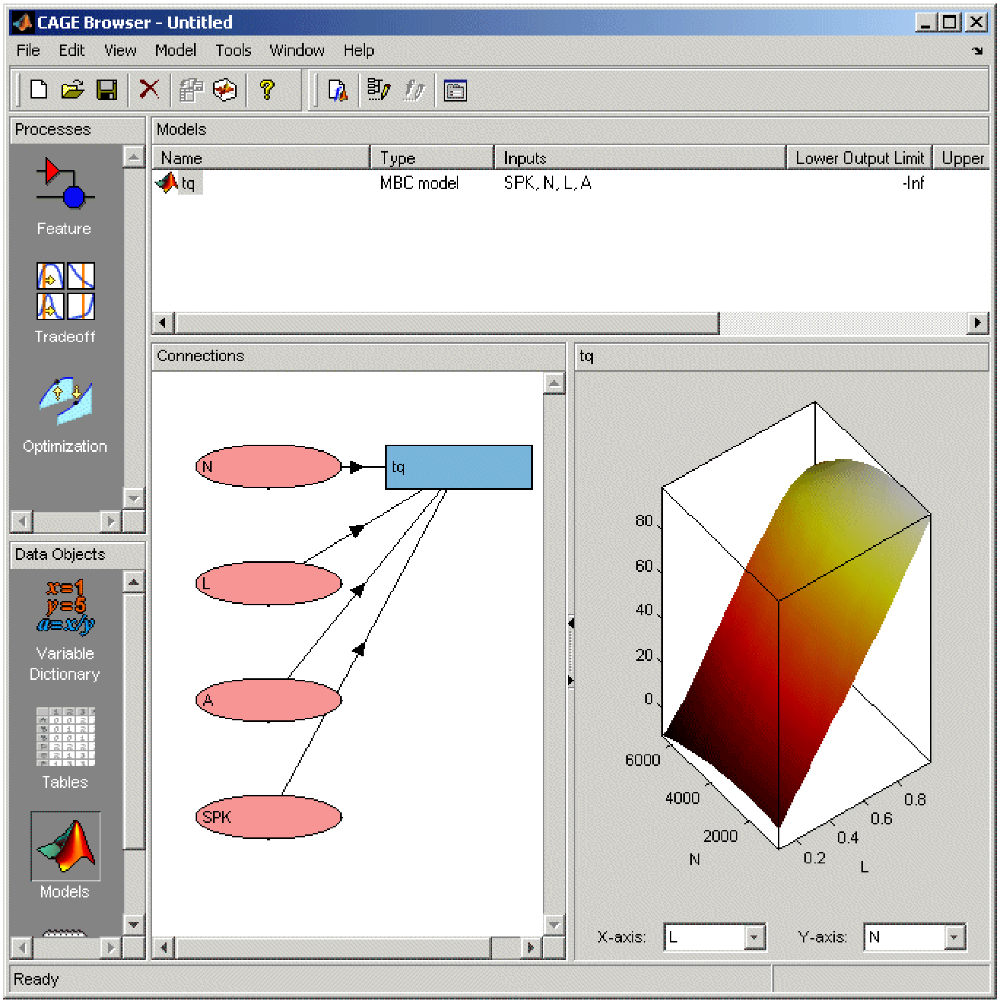
2 Browse to `matlab\toolbox\mbc\mbctraining`, select the `tutorial.exm` file (this is a copy of the torque model built in the Model Browser's Quick Start tutorial), and click **Open**. The Model Import Wizard appears.

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.

CAGE switches to the **Models** view, as shown following.



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

For more information about models, see “Setting Up Models” in the CAGE documentation.

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 as follows:

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

   CAGE automatically displays the **Feature** view and creates a new feature.

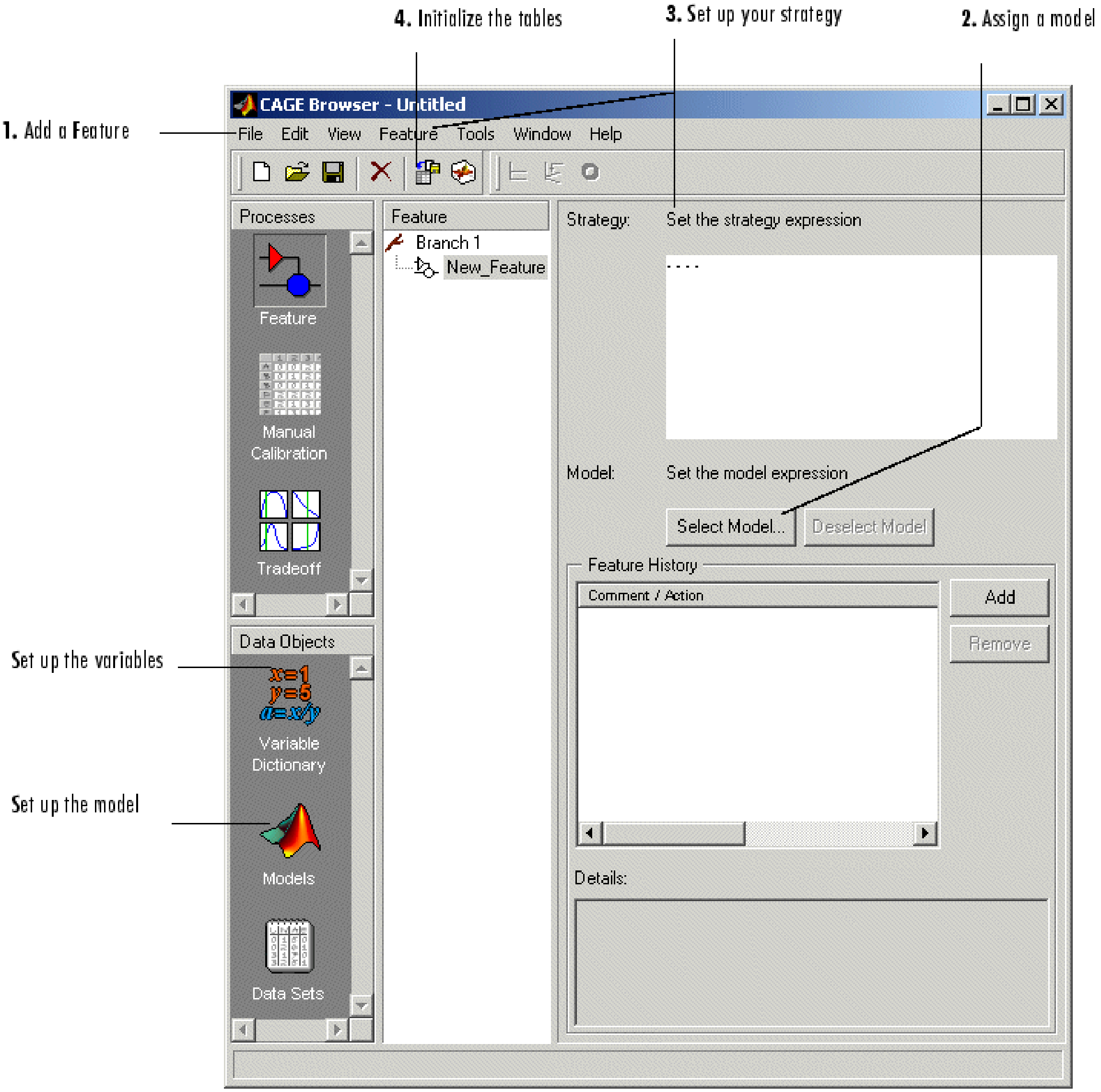
2. Click the **Select Model** button. This opens the Select Model dialog. Select `tq` (currently the only model in your project) and click **OK**.

   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 8-11.

   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 8-13.



Setting Up the Strategy

The toolbox uses Simulink to graphically specify the strategy.

Importing a Strategy

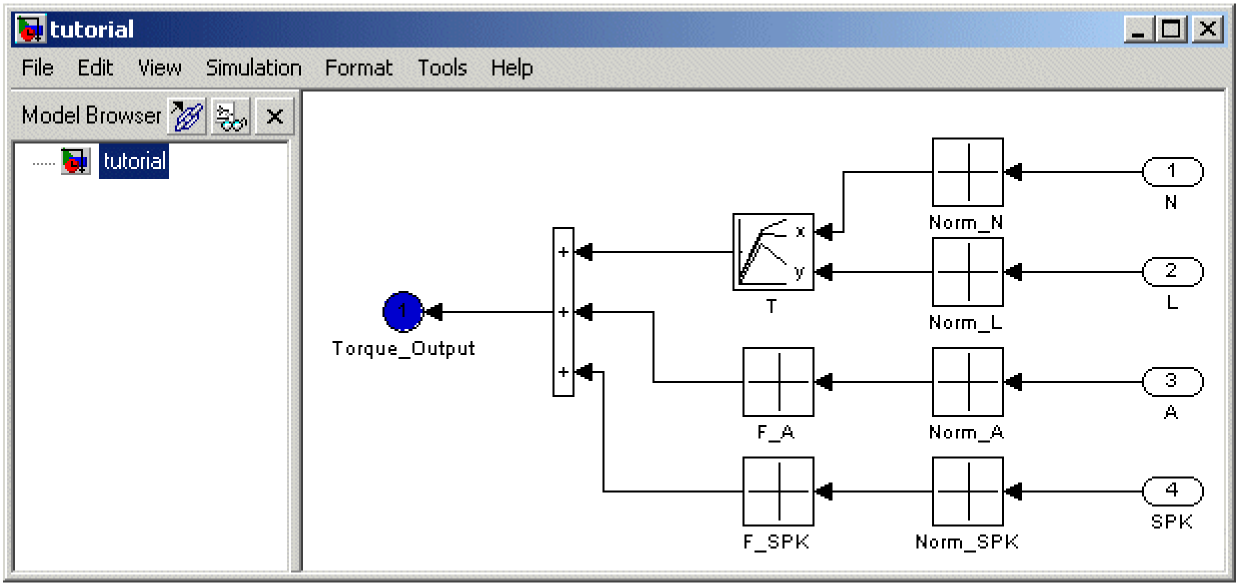
To import a strategy,

1 Select **File > Import > Strategy**.

2 Select the file called `tutorial.mdl`, found in `matlab\toolbox\mbc\mbctraining`, and click **Open**.

3 This opens the Import Strategy dialog, giving you the choice of automatic or manual import. To view the strategy, click **Manual**.

This opens the following Simulink window.



This shows how the strategy is built up.

4 Double-click the blue circle labeled `Torque_Output`.

Note This closes the Simulink window and parses the new feature into CAGE.

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.

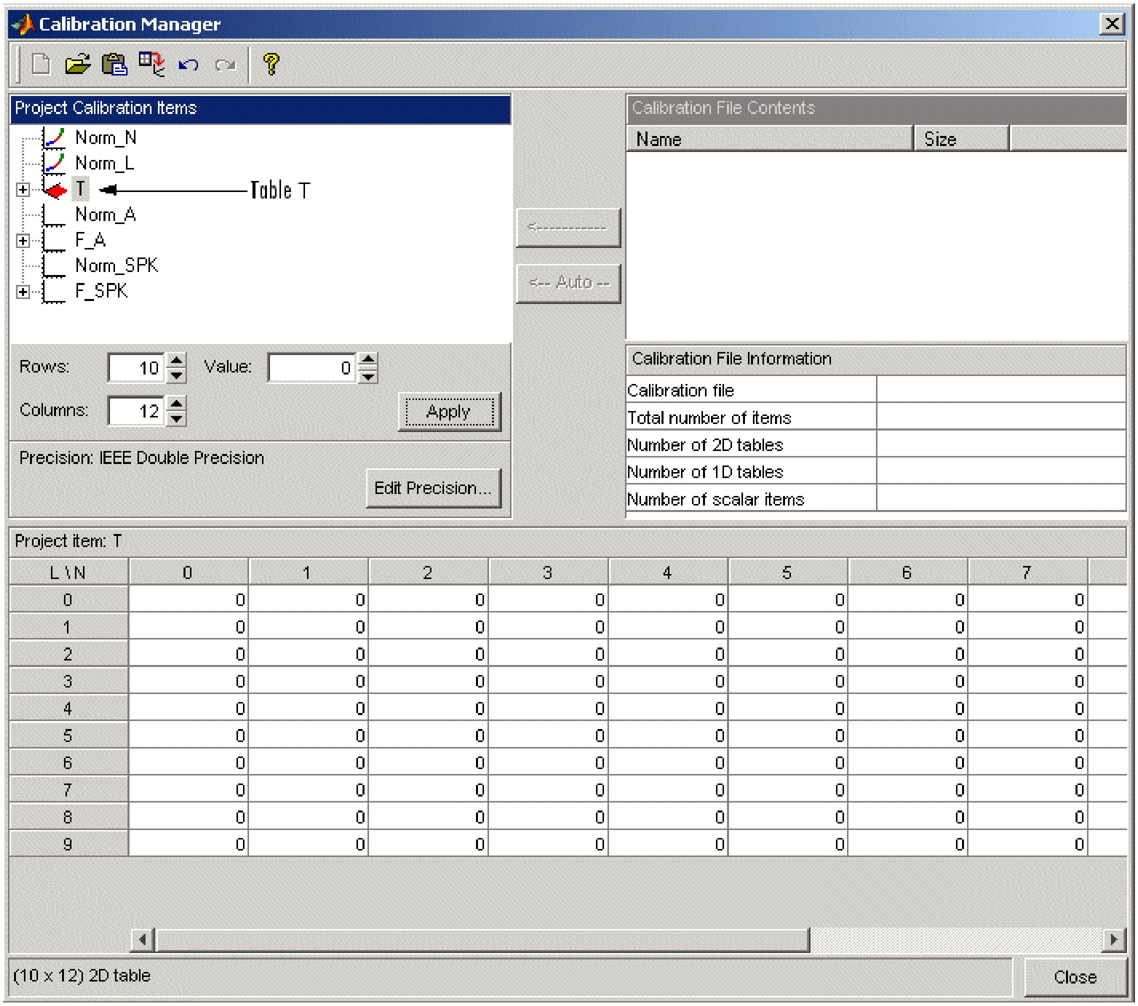
6. Click the plus next to New_Feature to expand the Feature tree and observe the tables created by importing the strategy: T, F_A, and F_SPK. Expand each table node in turn to view the normalizers of each. You will define the sizes of the tables next.

For a more detailed description of strategies, see "Setting Up Your Strategy" in the CAGE documentation.

Setting Up the Tables

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

Click Calibration Manager or select **Tools > Calibration Manager**. The Calibration Manager dialog box opens, so you can specify the number of breakpoints for each axis.



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.

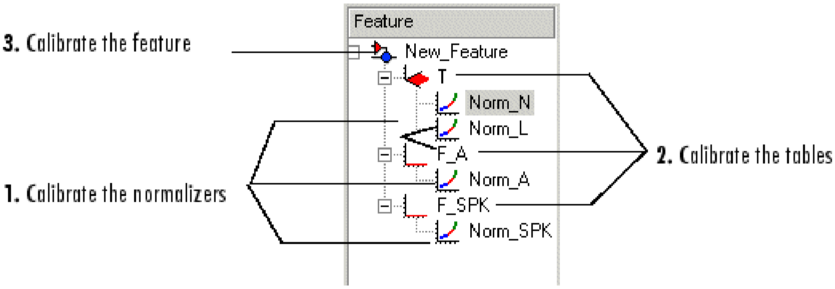
These are the three steps to calibrate a feature, described in these sections:

1. Calibrate the normalizers.

2. Calibrate the tables.

3. Calibrate the feature as a whole.

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

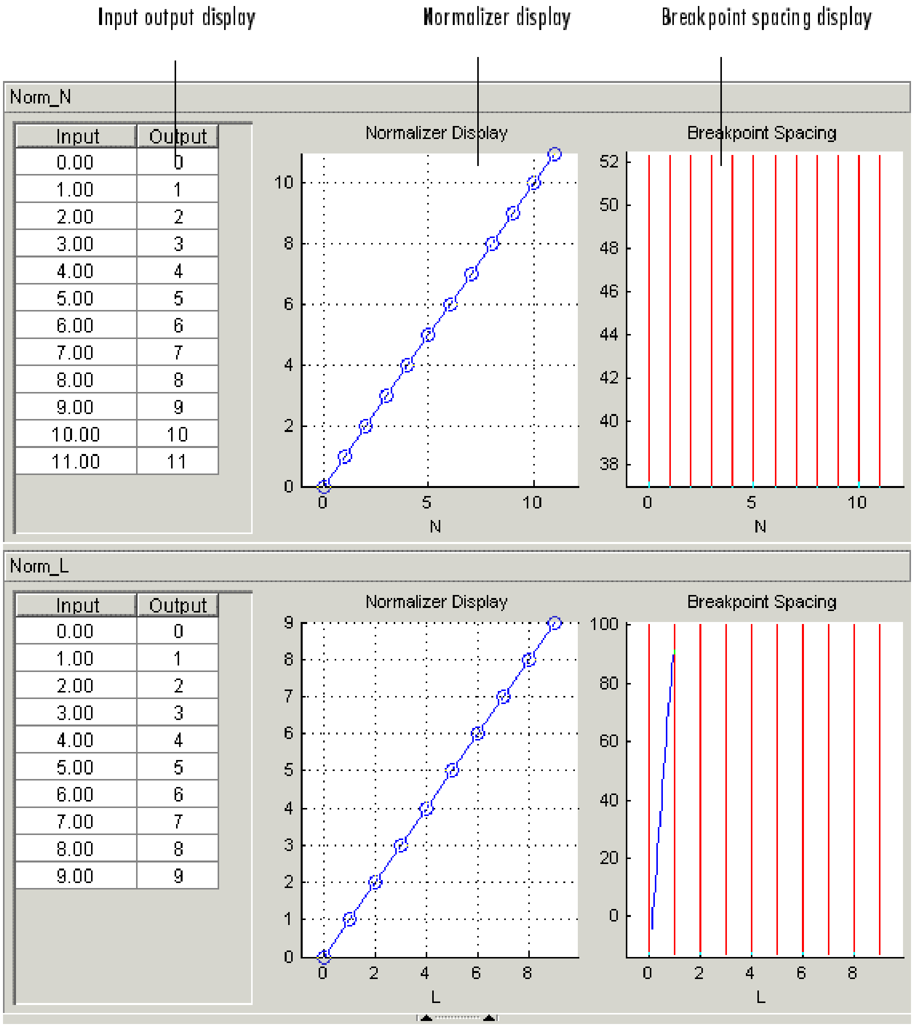


Each node in the display has a different view and different operations.

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.



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” in the CAGE documentation.

Placing the Breakpoints Automatically

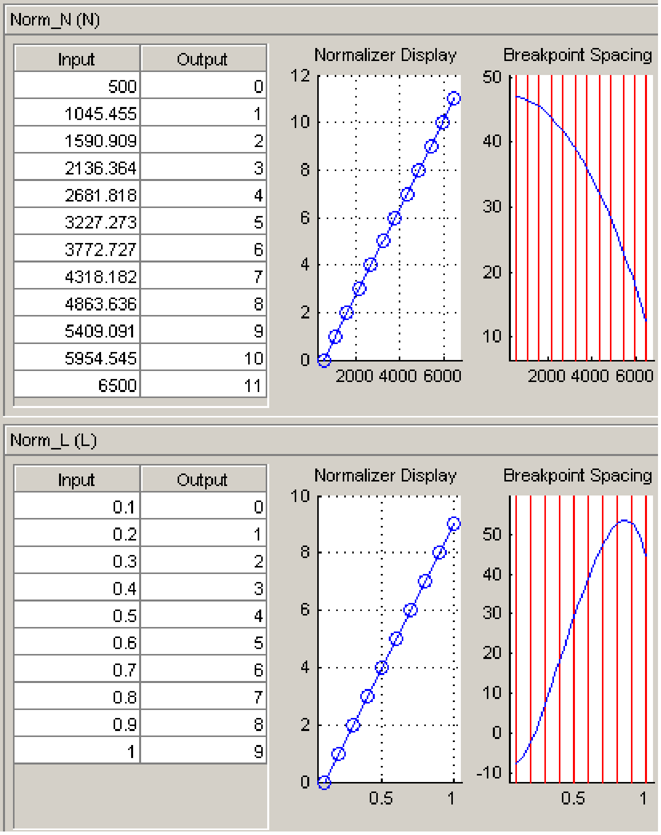
You now must space the breakpoints across the range of each variable. For example, `Norm_N` takes values from 500 to 6500, the range of the engine speed.

To space the breakpoints evenly throughout the data values,

1 Click Initialize in the toolbar. Alternatively, select **Normalizer > Initialize**.

   This opens a dialog box that suggests ranges for `Norm_N` and `Norm_L`.

2 To accept the default ranges of values of the data, click **OK**.



Norm_N (N)

Input	Output
500	0
1045.455	1
1590.909	2
2136.364	3
2681.818	4
3227.273	5
3772.727	6
4318.182	7
4863.636	8
5409.091	9
5954.545	10
6500	11

Norm_L (L)

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

A better fit between model and table can often be achieved by spacing the breakpoints nonlinearly.

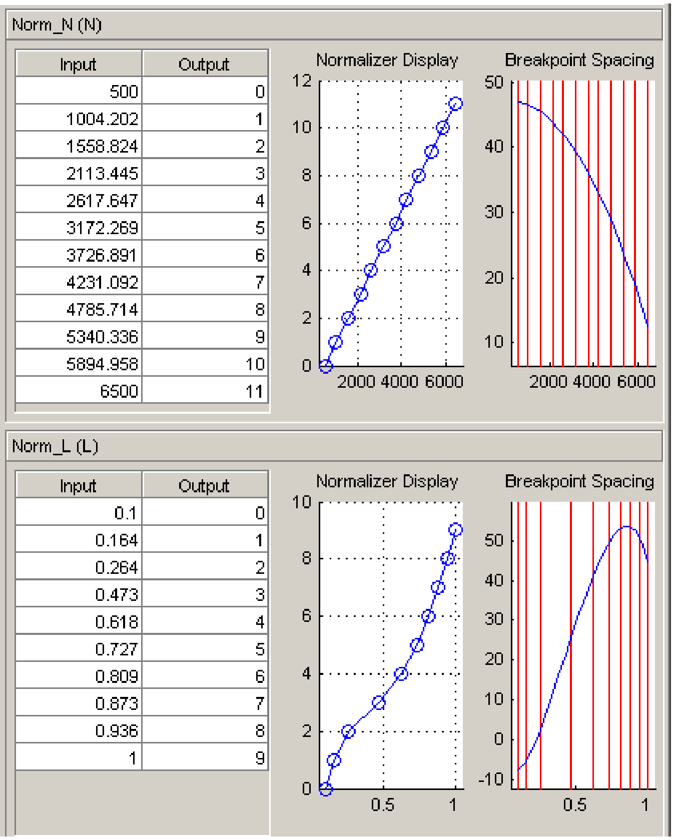
1 Click Fill in the toolbar. Alternatively, select **Normalizer > Fill**.

This opens a dialog box that suggests ranges for `Norm_N` and `Norm_L`. It also suggests values for `AFR` and `SPK`; these values are the set points for `AFR` and `SPK`.

2 To accept the values in the dialog box, click **OK**.

This ensures that the majority of the breakpoints are where the model is most curved. The table now has most values where the model changes most. So, with the same number of breakpoints, the table is a better match to the model.

For more information about calibrating the normalizers, see "About Normalizers" in the CAGE documentation.



Norm_N (N)

Input	Output
500	0
1004.202	1
1558.824	2
2113.445	3
2617.647	4
3172.269	5
3726.891	6
4231.092	7
4785.714	8
5340.336	9
5894.958	10
6500	11

Norm_L (L)

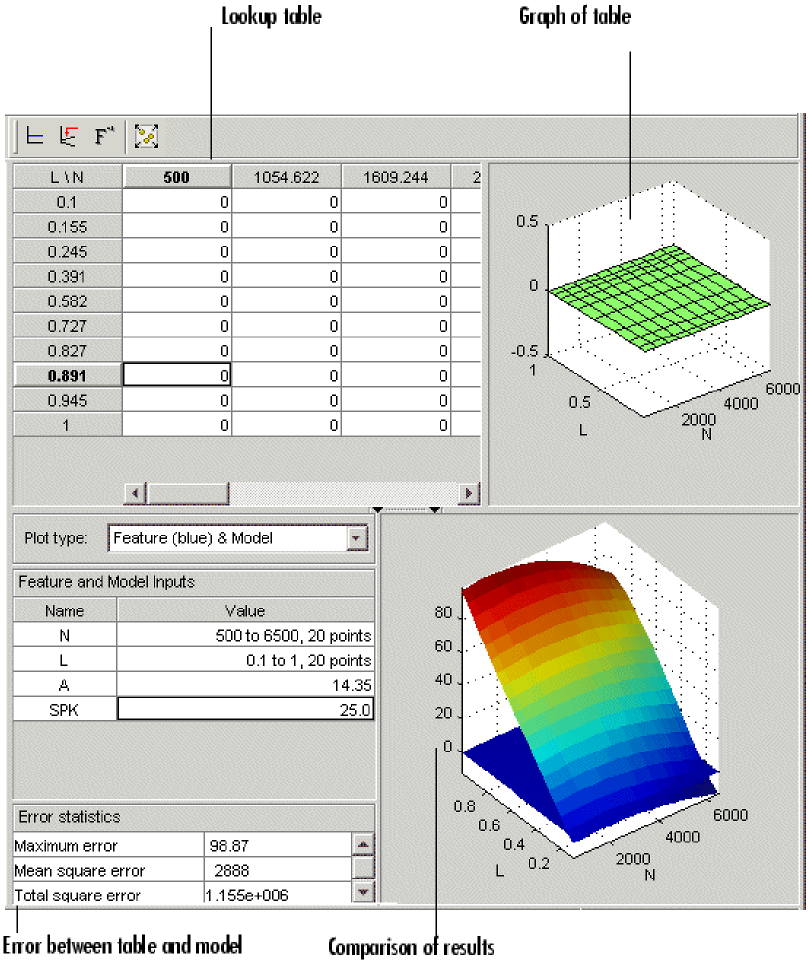
Input	Output
0.1	0
0.164	1
0.264	2
0.473	3
0.618	4
0.727	5
0.809	6
0.873	7
0.936	8
1	9

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.



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

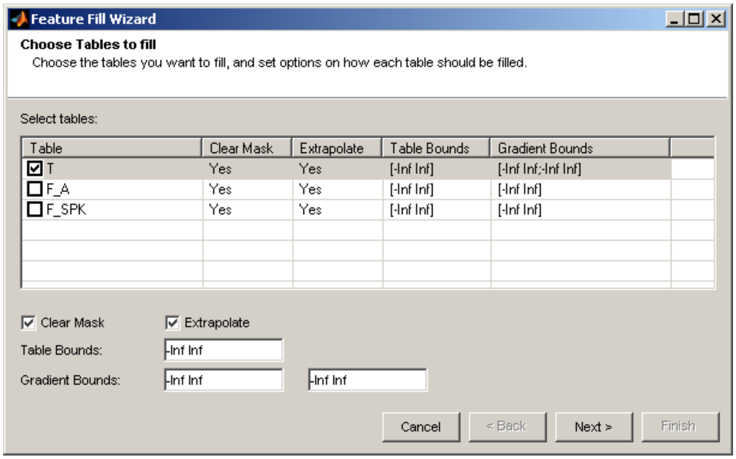
Maximum error	98.87
Mean square error	2888
Total square error	1.155e+006

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 Fill on the toolbar.

This opens the Feature Fill Wizard.



Feature Fill Wizard

Choose Tables to fill

Choose the tables you want to fill, and set options on how each table should be filled.

Select tables:

Table	Clear Mask	Extrapolate	Table Bounds	Gradient Bounds
☑ T	Yes	Yes	[-Inf Inf]	[-Inf Inf;-Inf Inf]
□ F_A	Yes	Yes	[-Inf Inf]	[-Inf Inf]
□ F_SPK	Yes	Yes	[-Inf Inf]	[-Inf Inf]

☑ Clear Mask ☑ Extrapolate

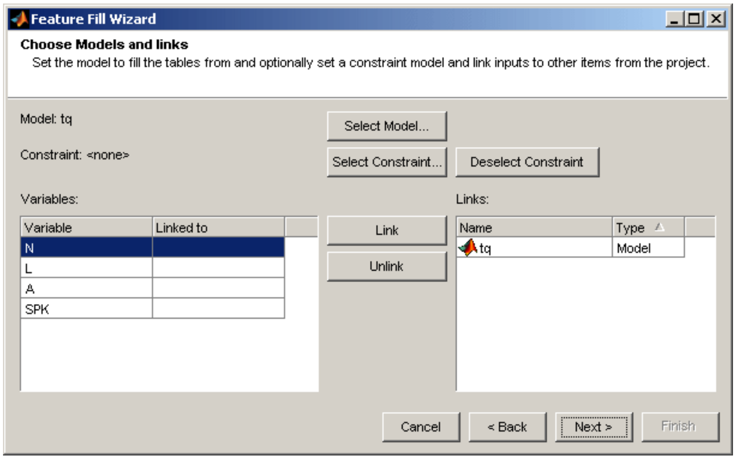
Table Bounds: -Inf Inf

Gradient Bounds: -Inf Inf -Inf Inf

Observe the `T` table check box is selected, but you can also fill multiple tables at once using the wizard. You can **Fill** from the top feature node or from any table node in a feature. On the first screen you can set table bounds to avoid extrapolating to infeasible values, choose whether to only fill cells in the extrapolation mask, and choose whether to extrapolate automatically.

Leave the settings at the defaults and click **Next**.

2 On this screen you can see the `tq` model is selected to fill the table. Here you could also set up constraints, for example using a boundary model to constrain filling to table areas where data was collected, and you can link other models or features to inputs.



Choose Models and links

Set the model to fill the tables from and optionally set a constraint model and link inputs to other items from the project.

Model: tq

Constraint: <none>

Variables:

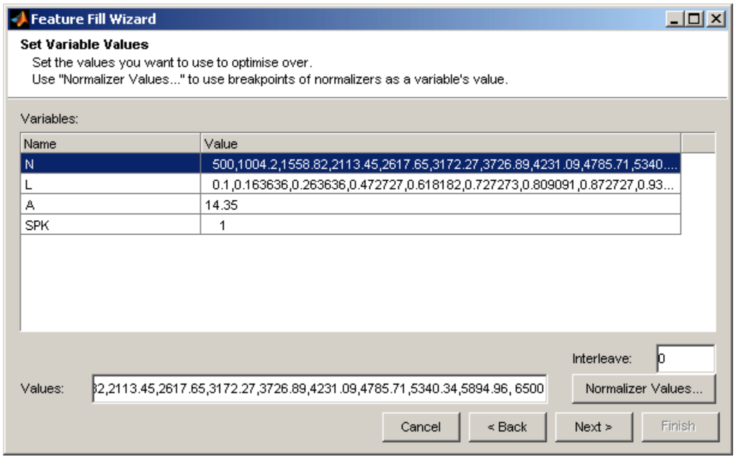
Variable	Linked to
N	
L	
A	
SPK	

Links:

Name	Type
tq	Model

Leave the settings at the defaults and click **Next**.

3 Here you can set variable values for optimizing over.



Set Variable Values

Set the values you want to use to optimise over.
Use "Normalizer Values..." to use breakpoints of normalizers as a variable's value.

Variables:

Name	Value
N	500,1004.2,1558.82,2113.45,2617.65,3172.27,3726.89,4231.09,4785.71,5340....
L	0.1,0.163636,0.263636,0.472727,0.618182,0.727273,0.809091,0.872727,0.93...
A	14.35
SPK	1

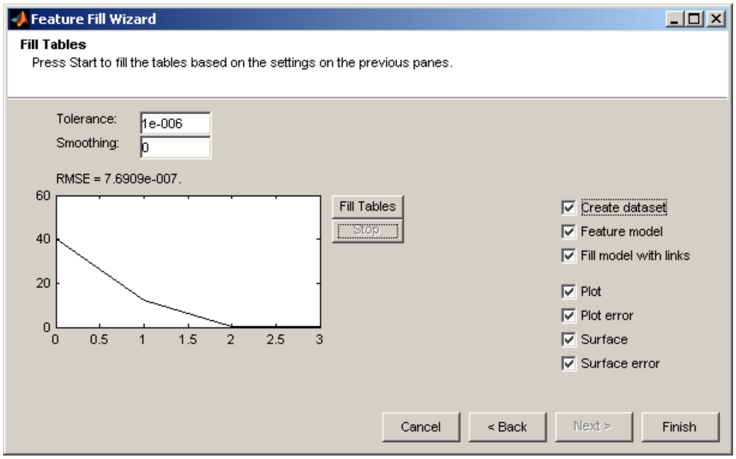
Interleave: 0

Values: 32,2113.45,2617.65,3172.27,3726.89,4231.09,4785.71,5340.34,5894.96, 6500

By default the table's normalizer breakpoints (here N and L) and the set points of the other variables (A and SPK) are selected. You can select different normalizers, and edit values in the **Values** edit box to optimize over a range rather than at a single point. If you choose a range of values the table will be filled using the average model value at each cell. You can use the Interleave setting to add values between normalizer breakpoints to optimize over a finer grid than the number of table cells.

Leave the settings at the defaults and click **Next**.

4 Click **Fill Tables**.



Feature Fill Wizard

Fill Tables

Press Start to fill the tables based on the settings on the previous panes.

Tolerance: 1e-006

Smoothing: 0

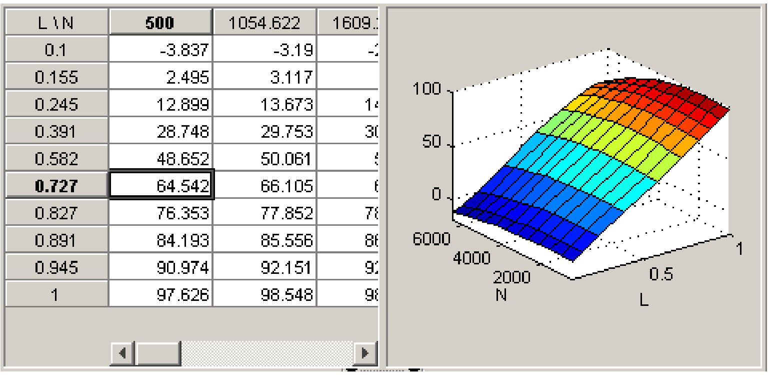
RMSE = 7.6909e-007.

The graph shows the progress of the optimization. Select all the check boxes and click **Finish**.

Plots are created of the filled table surface and the error between the table values and the model.

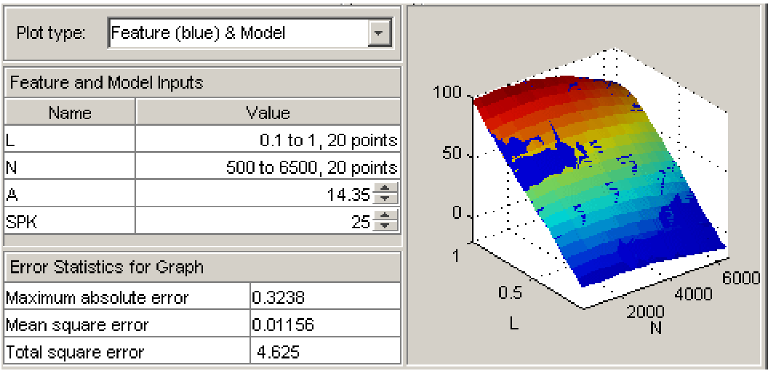
5 Click **OK**.

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



L \ N	**500**	1054.622	1609.1
0.1	-3.837	-3.19	-
0.155	2.495	3.117	
0.245	12.899	13.673	14
0.391	28.748	29.753	30
0.582	48.652	50.061	5
0.727	64.542	66.105	6
0.827	76.353	77.852	78
0.891	84.193	85.556	86
0.945	90.974	92.151	92
1	97.626	98.548	98

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



Plot type: Feature (blue) & Model

Feature and Model Inputs

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

Error Statistics for Graph

Maximum absolute 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 optimized values compared to the model at these operating points.

For more information about the process of filling tables, see “Calibrating the Tables” in the CAGE documentation

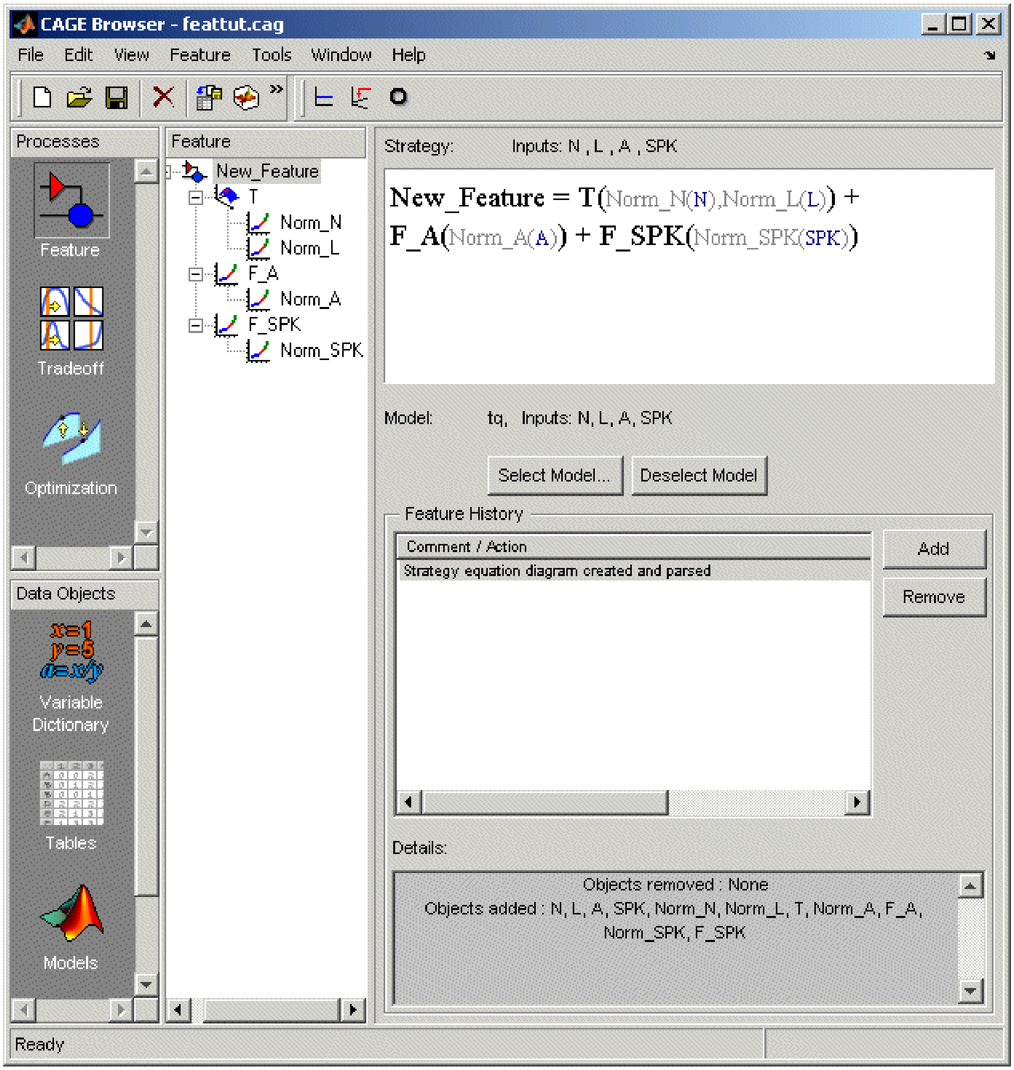
Now you must fill the tables F_A and F_SPK and their normalizers. The tables are modifiers for AFR and the spark angle respectively. These steps are described in the next section.

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 optimized values compared to the torque model, `tq`. You must now calibrate the normalizers and tables for `F_A` and `F_SPK`.

You could calibrate the normalizers and then the tables for F_A and F_SPK in turn. However, CAGE enables you to calibrate the entire feature in one procedure.

To view the Feature view following, click the `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

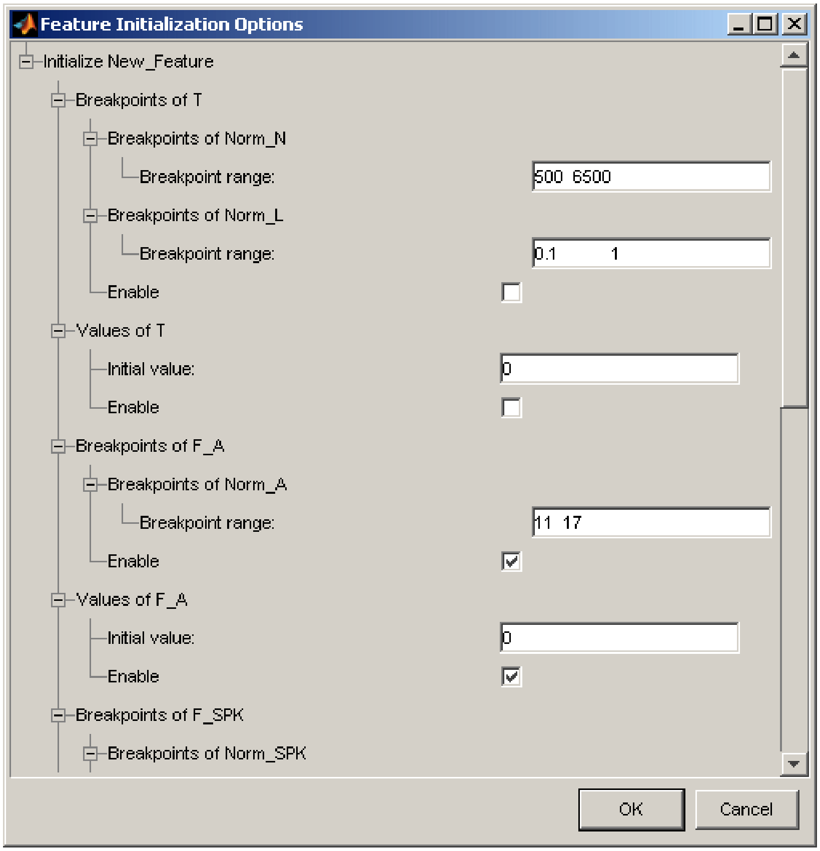
Strategy equation diagram created and parsed

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 > Initialize** (or use the Initialize toolbar button). The Feature Initialization Options dialog appears.

2 Clear the Enable check boxes for Breakpoints of T, and Values of T, as shown.



Feature Initialization Options

- Initialize New_Feature
  - Breakpoints of T
    - Breakpoints of Norm_N
      - Breakpoint range: 500 6500
    - Breakpoints of Norm_L
      - Breakpoint range: 0.1 1
    - Enable □
  - Values of T
    - Initial value: 0
    - Enable □
  - Breakpoints of F_A
    - Breakpoints of Norm_A
      - Breakpoint range: 11 17
    - Enable ☑
  - Values of F_A
    - Initial value: 0
    - Enable ☑
  - Breakpoints of F_SPK
    - Breakpoints of Norm_SPK

You have already optimized the breakpoints and table values for table T, so you only want to initialize the other tables `F_A` and `F_SPK`.

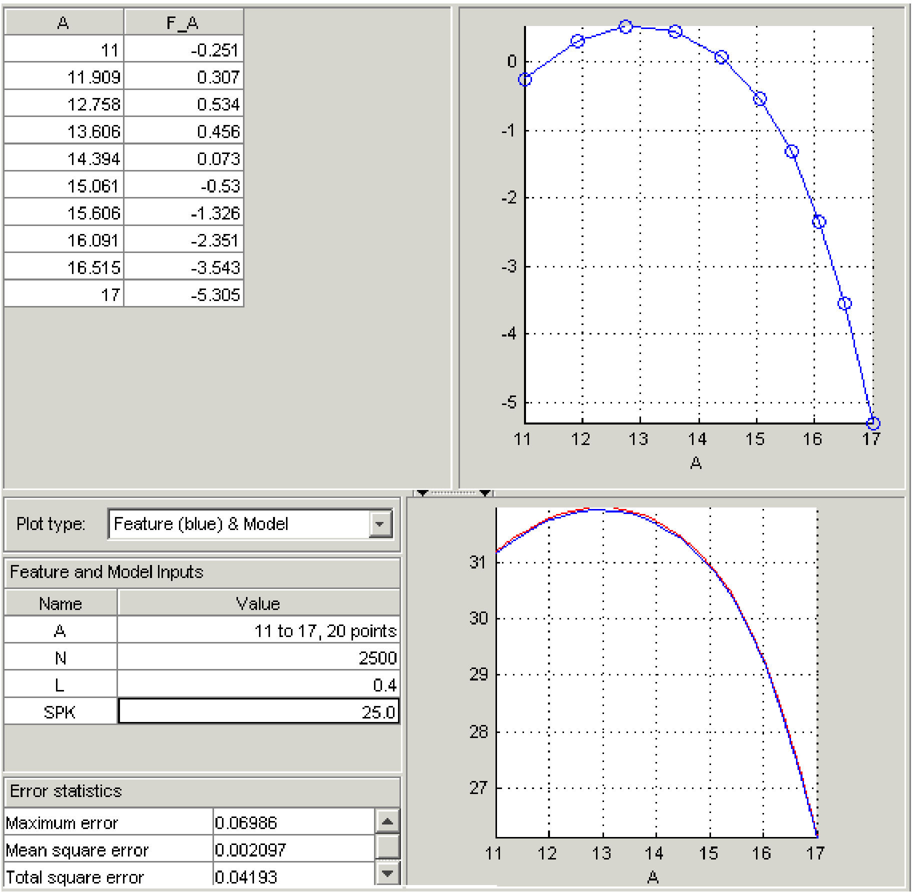
Click **OK**.

3 Select **Feature > Fill** (or use the Fill toolbar button) to open the Feature Fill Wizard. This time select only the F_A table to fill, and follow the steps in the wizard to fill this table: click Next 3 times then click **Fill Tables**. Select the F_A table node to view the results.

4 Select F_SPK table node and click Fill. Repeat the wizard steps to fill the table.

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. CAGE runs an optimization routine over the feature to minimize the total square error between the model and the feature.



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 blue line of the strategy and a red slice through the model, over the range of AFR.

This completes the calibration of the torque feature.

For more information about calibrating features, see “Calibrating the Feature Node” in the CAGE documentation.

You can now export the calibration.

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. For the purposes of this tutorial, select `Comma Separated Value (.csv)`.
4. Enter `tutorial.csv` as the file name and click **Save**.

This exports the calibration.

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.

9

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.

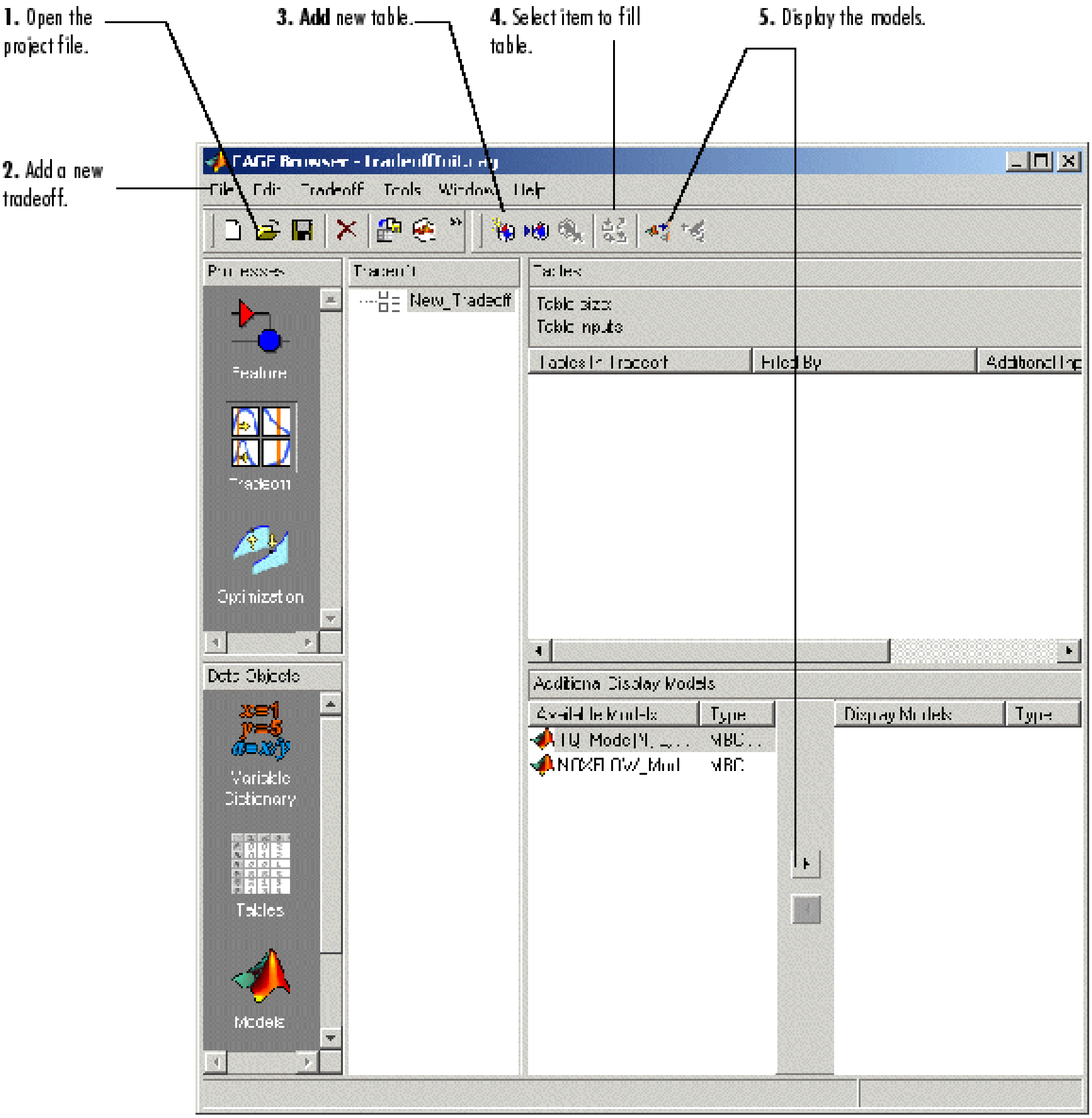
1 Select **File > Open Project** (or the toolbar button) to choose the `tradeoffInit.cag` file, found in the `matlab\toolbox\mbc\mbctraining` directory, then click **OK**.

   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 “Variables and Models” in the CAGE documentation.

2 To create a tradeoff calibration, select **File > New > Tradeoff**.

   This takes you to the **Tradeoff** view. You need to add tables and display models to the tradeoff, which are described step by step in the following sections:

   - “Adding Tables to a Tradeoff Calibration” on page 9-5.
   - “Displaying the Models” on page 9-6 describes how you display the models of torque and NOX emissions.



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.

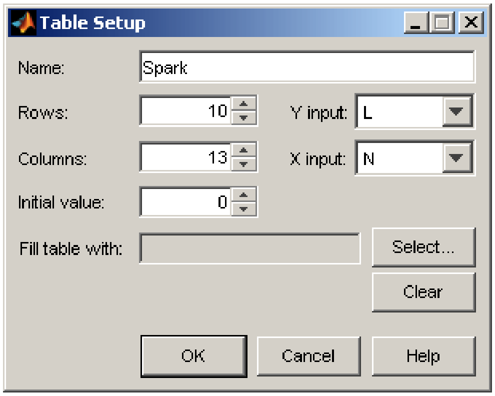


Table Setup

Name: Spark

Rows: 10   Y input: L

Columns: 13   X input: N

Initial value: 0

Fill table with:

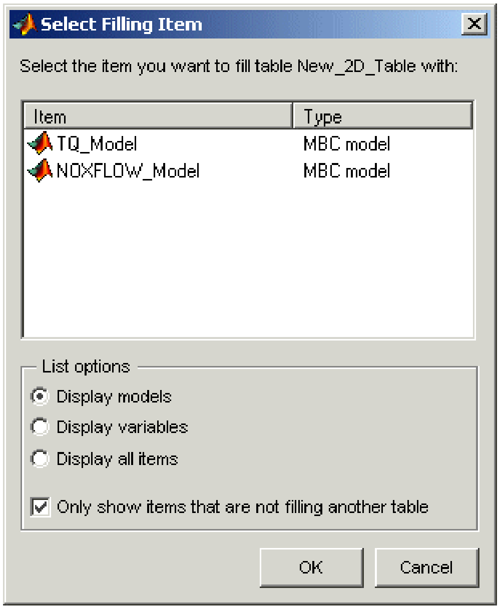
2 Enter `Spark` as the table **Name**.

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

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

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

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



Select Filling Item

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

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

7 Click **OK** to close the Table Setup dialog.

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

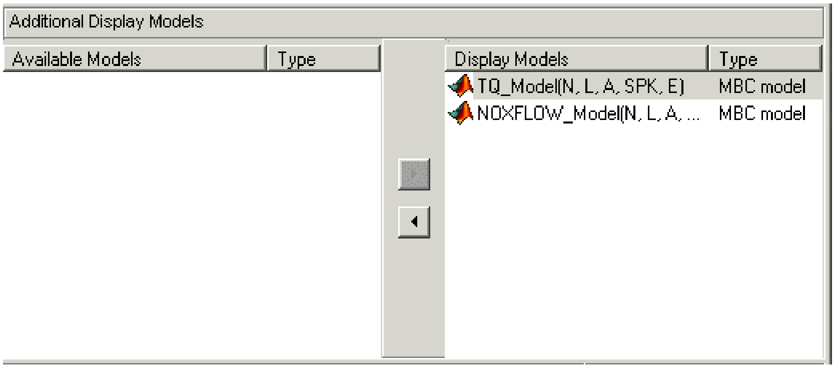
Displaying the Models

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

To display both models,

- Click Add Model to Display List in the toolbar twice. This will move both available models into the Display list.
- Alternatively, **Shift**-click to select both models in the **Available Models** list and click > to include both models in the current display. In this case you want to include all available models. You can click to select particular models in the list to display.

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



Additional Display Models

Available Models	Type

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.

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

4. Fill the table by extrapolation.

5. Export the calibration.

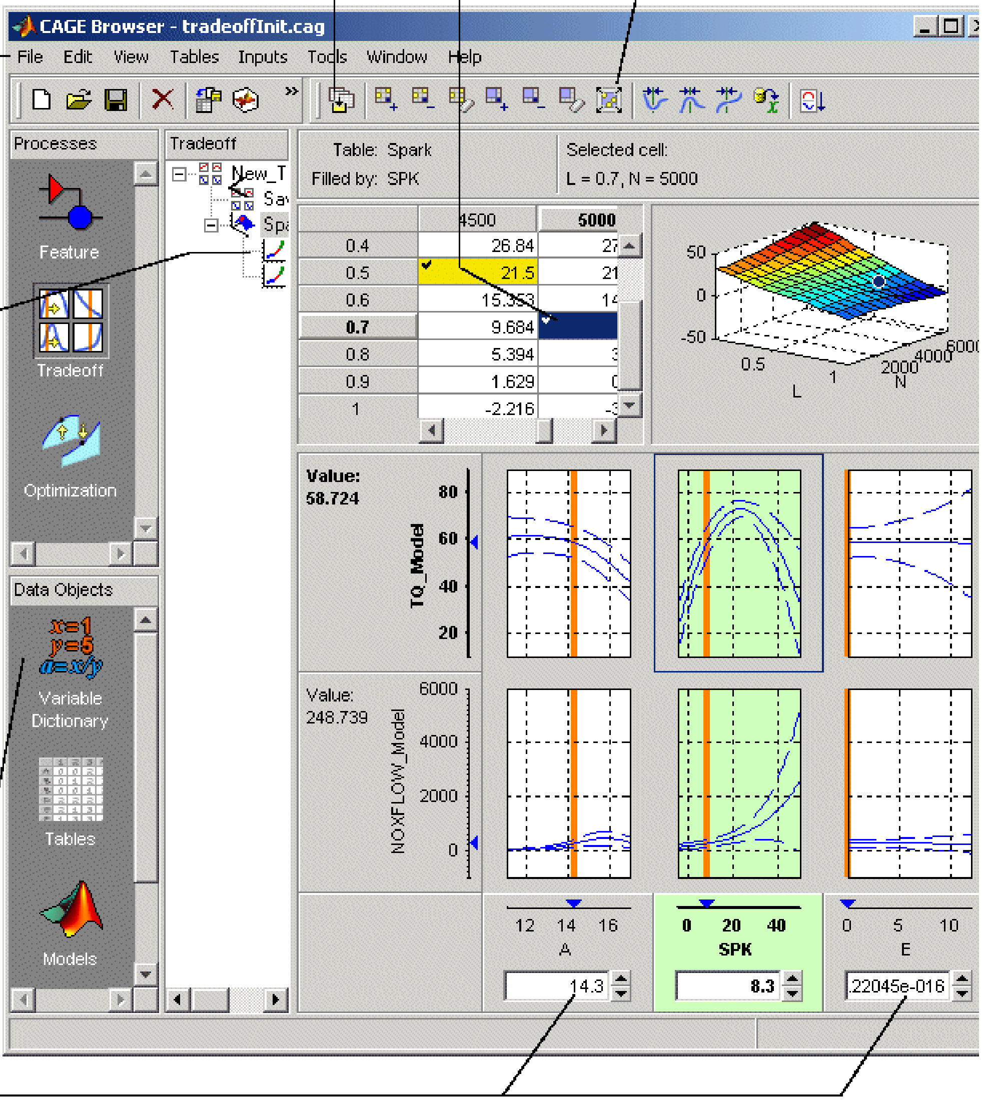


Table: Spark
Filled by: SPK

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

A: 14.3

SPK: 8.3

E: .22045e-016

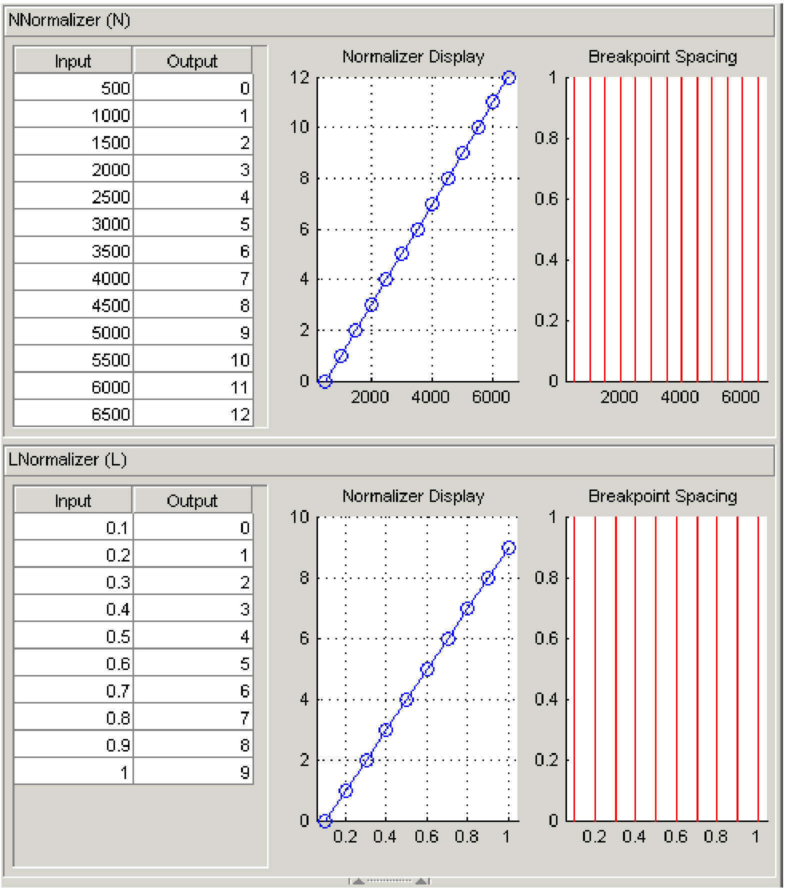
1. Check the normalizers.

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 `Spark` table and its normalizers `Speed` and `Load`. Click to highlight either normalizer to see the normalizer view. A tradeoff calibration does not compare the model and the table directly, so you cannot space the breakpoints by reference to the model.



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 edit the **Set Point** to `14.3`, the stoichiometric constant, and press **Enter**.

3 Click `E` and change the **Set Point** to `0` and press **Enter**.

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

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

5 Highlight the `Spark` table node in the Tradeoff tree display.

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

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

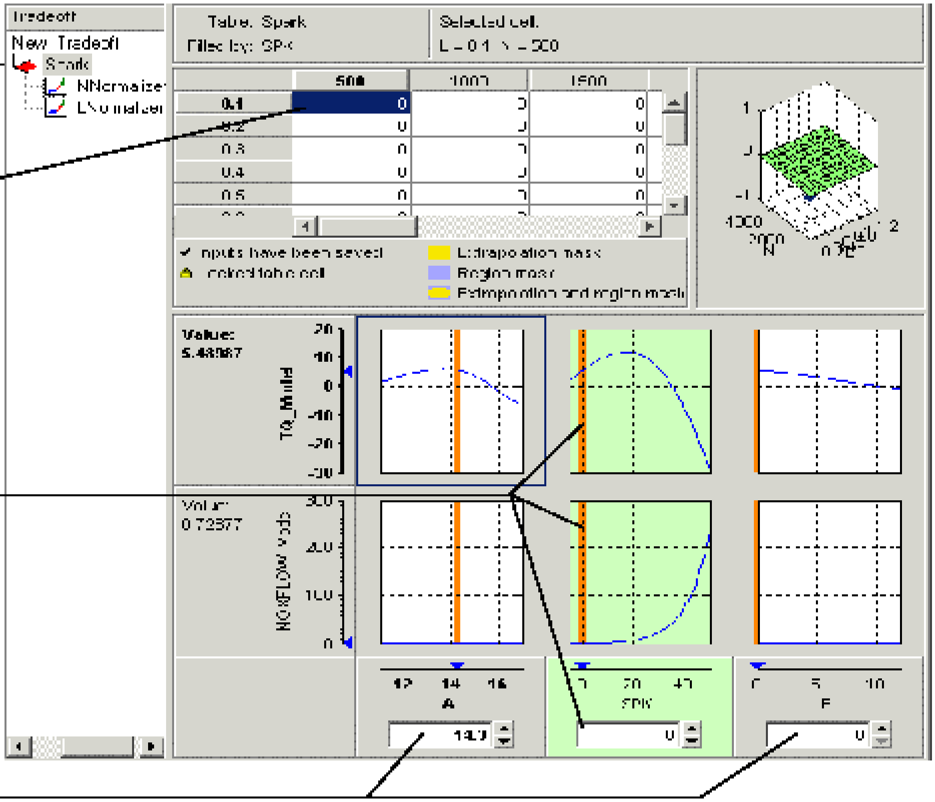


Table: Spark
Filled by: SPK

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

1. Highlight the **Spark** node.

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

3. Change values of **SPK** to tradeoff 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.

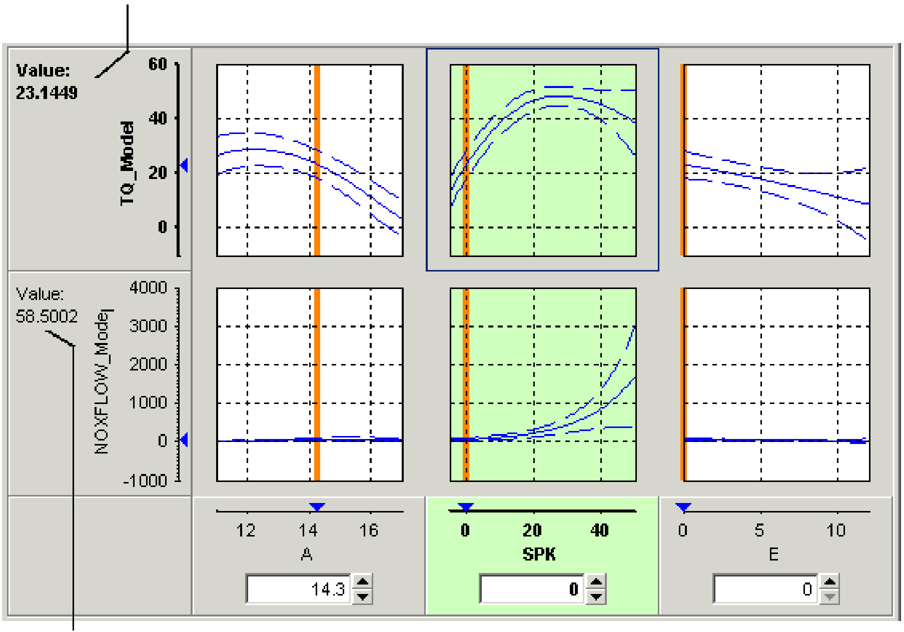
The upper pane displays the lookup table, and the lower pane displays the behavior of the torque and NOX emissions models with each variable.

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:
23.1449

Value:
58.5002

A 14.3

SPK 0

E 0

Value of the NOX model

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

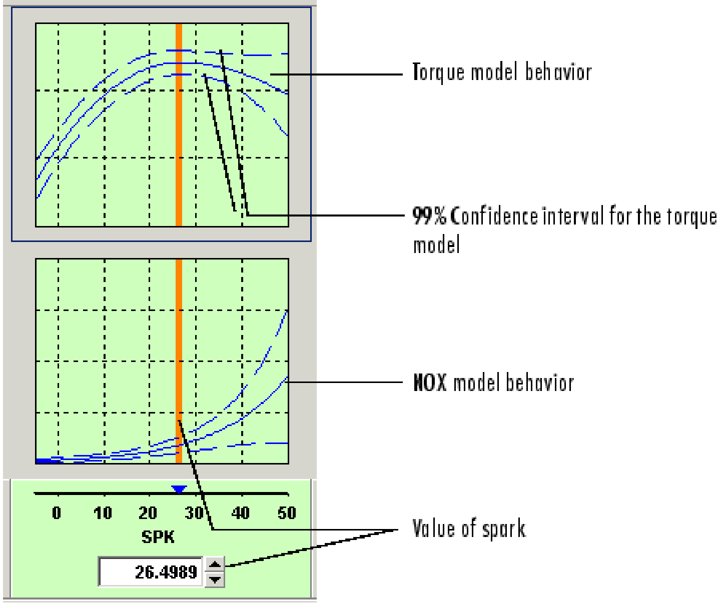
You are calibrating the `Spark` table, so the two spark (`SPK`) graphs are green, indicating that these graphs are directly linked to the currently selected lookup table.

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

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

3 Click to select the top SPK - TQ_Model graph (`TQ_Model` row, `SPK` column). When selected the graph is outlined as shown in the following example.

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



SPK 26.4989

At this operating point, the maximum torque that is generated is `48.136` when the spark angle is `26.4989`. However, the value of NOX is `348.968`, which is greater than the restriction of 250 g/hr. Clearly you have to look at another value of spark angle.

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

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

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

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

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

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

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

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

Speed, N	Load, L	Spark Angle, SPK
2500	0.3	25.75
3000	0.8	10.7

Speed, N	Load, L	Spark Angle, SPK
5000	0.7	8.2
6000	0.2	41.3

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

Filling the Table by Extrapolation

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

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

Click in the toolbar to fill the table by extrapolation.

The lookup table is filled with values of spark angle.

The following figure displays the view after extrapolation over the table.

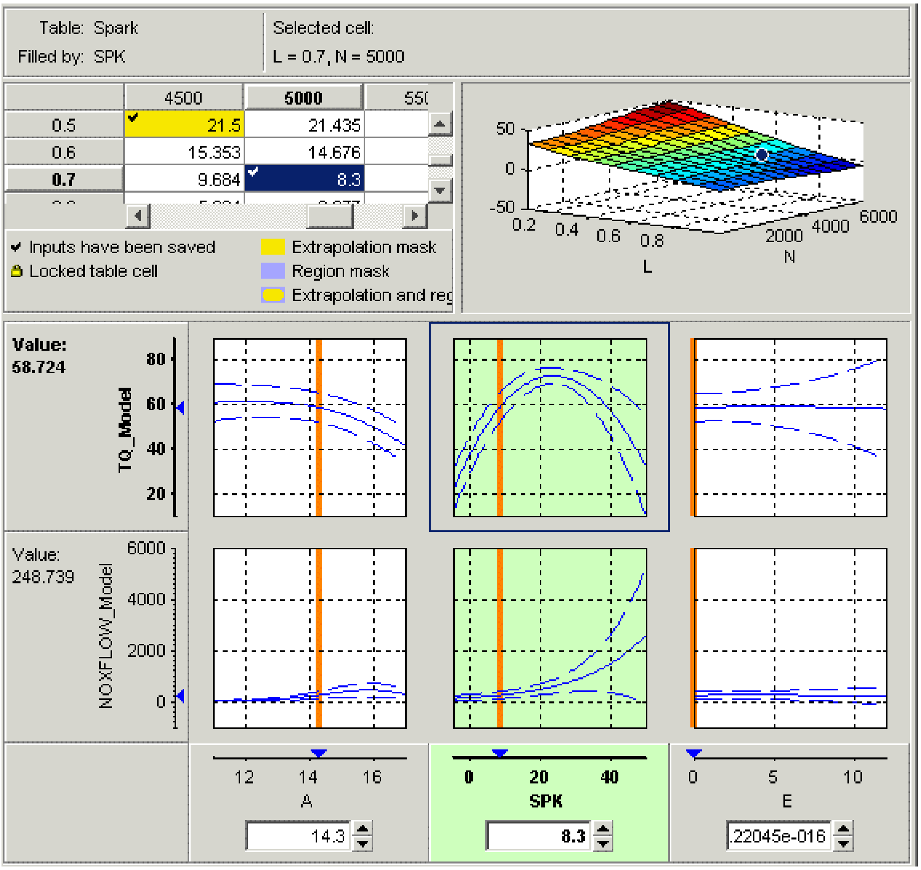


Table: Spark

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” in the CAGE documentation.

You can now export this calibration to file.

Exporting Calibrations

To export your table and its normalizers,

1. Select the `Spark` node in the branch display.
2. Select **File > Export > Calibration**.
3. Choose the file type you want for your calibration. For the purposes of this tutorial, select `Comma Separated Value (.csv)`.
4. Enter `tradeoff.csv` as the file name and click **Save**.

This exports the spark angle table and the normalizers, `Speed` and `Load`.

You have now completed the tradeoff calibration tutorial.

10

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” in the CAGE documentation.

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**.

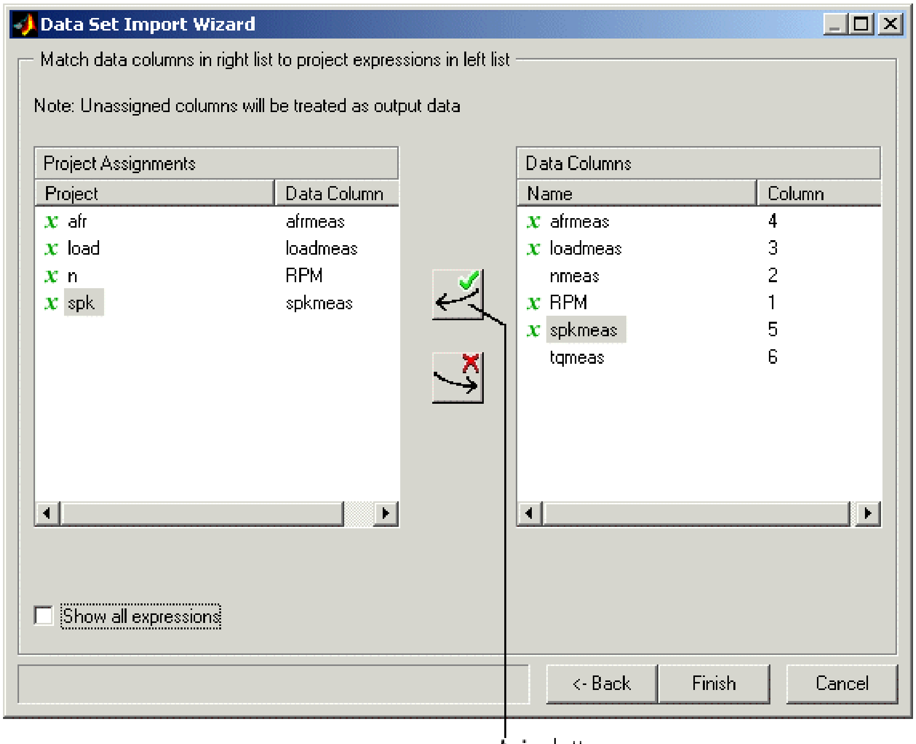
   This set of data includes six columns of data, the test cell settings for engine speed (`RPM`), and the measured values of torque (`tqmeas`), engine speed (`nmeas`), air/fuel ratio (`afrmeas`), spark angle (`spkmeas`), and load (`loadmeas`).

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.



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,

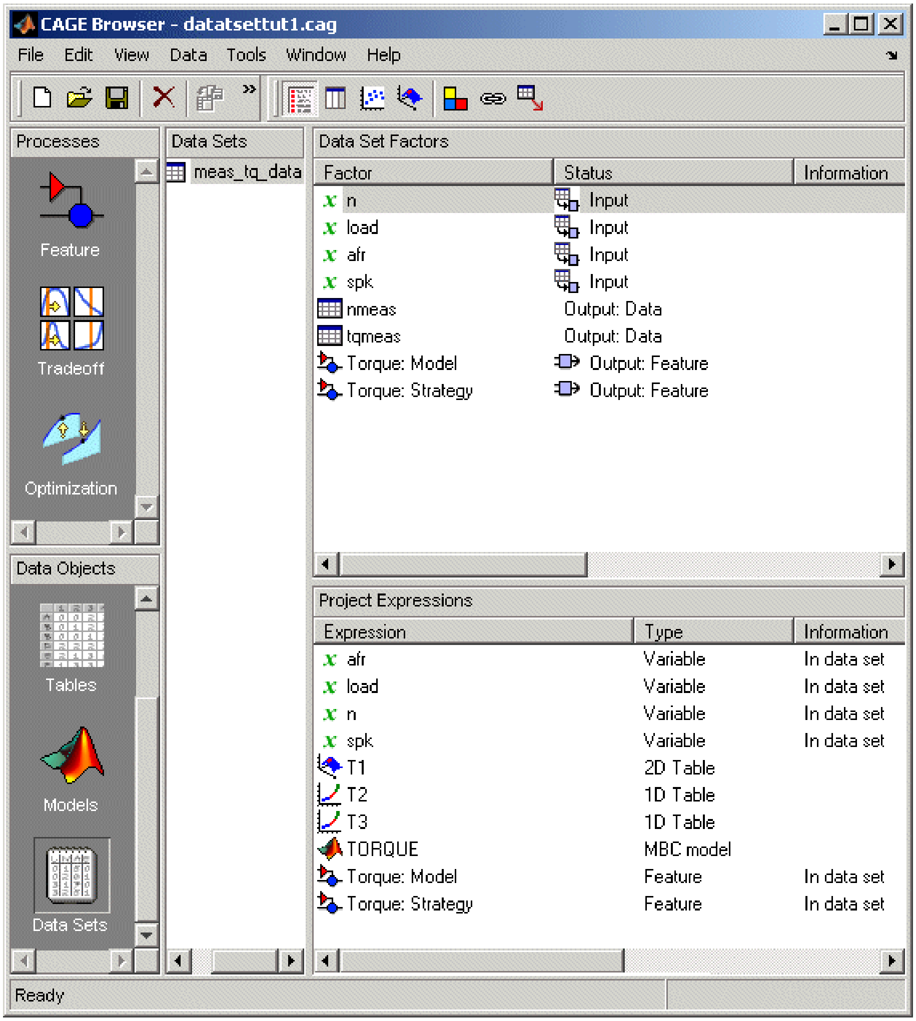
1 Highlight the `Torque` feature in the lower list of **Project Expressions**.

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	

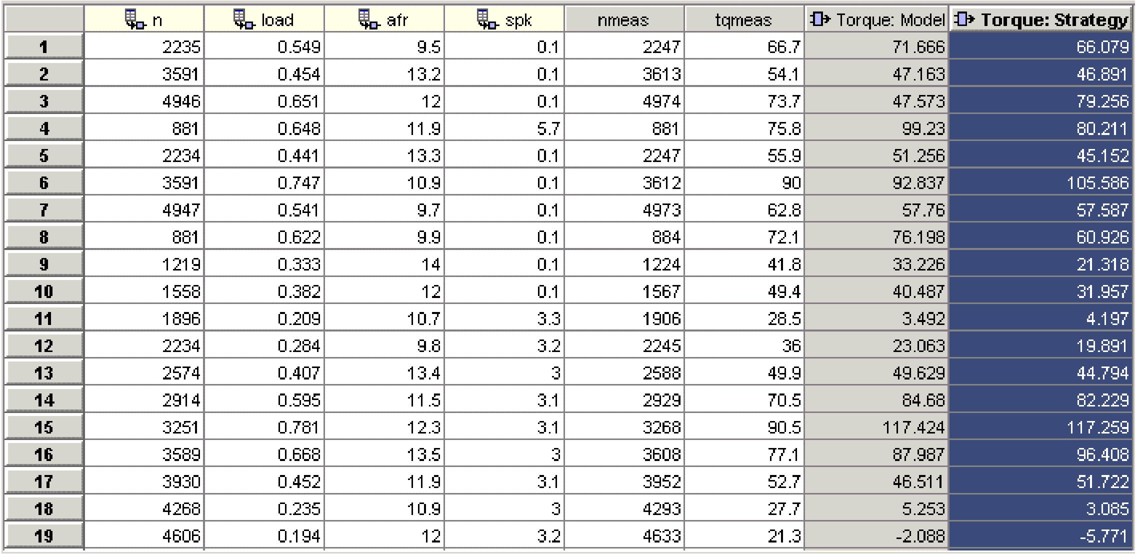
Expression	Type	Information
afr	Variable	In data set
load	Variable	In data set
n	Variable	In data set
spk	Variable	In data set
T1	2D Table	
T2	1D Table	
T3	1D Table	
TORQUE	MBC model	
Torque: Model	Feature	In data set
Torque: Strategy	Feature	In data set

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

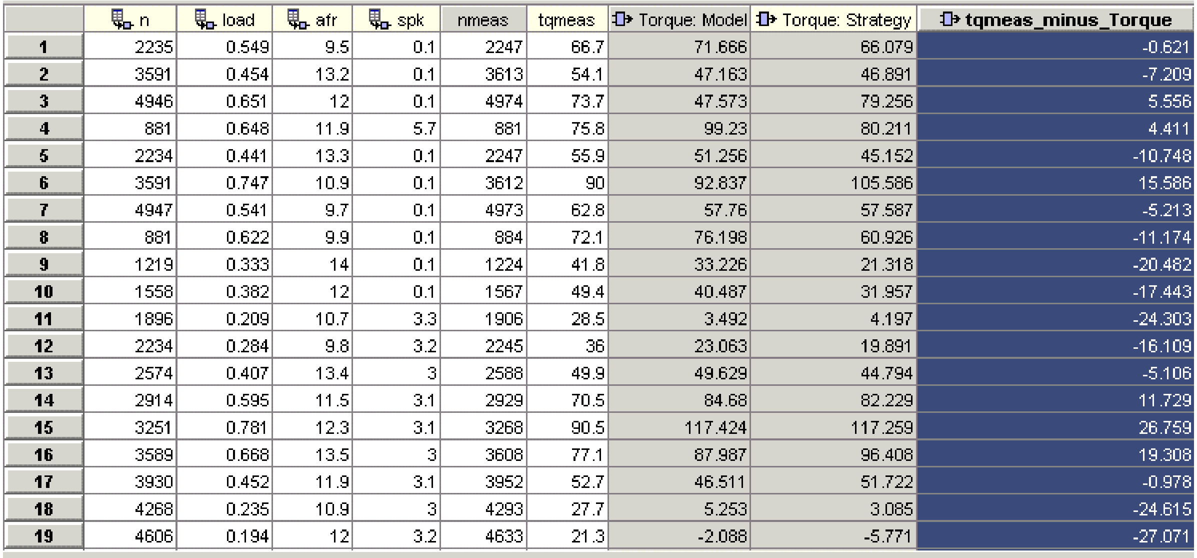
In the table, the input cells are white and the output cells are grey. Select the `Torque: Strategy` column header to see the view shown. The selected column turns blue and the column headers of the strategy's inputs (`n`, `load`, `afr` and `spk`) turn cream. Column headers are always highlighted in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

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

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

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

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



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

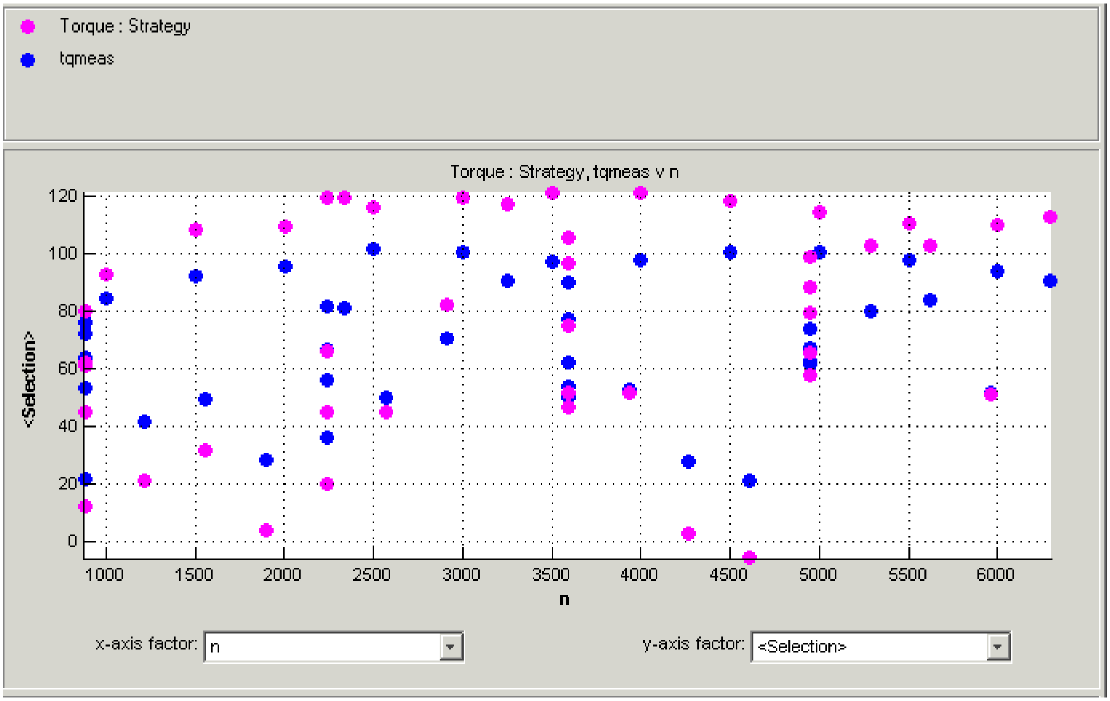
The error column is simply the difference between `tqmeas` and `Torque: Strategy`. This provides a simple way of comparing the feature and the measured data.

Viewing the Data Set as a Plot

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

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

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



x-axis factor: n

y-axis factor: <Selection>

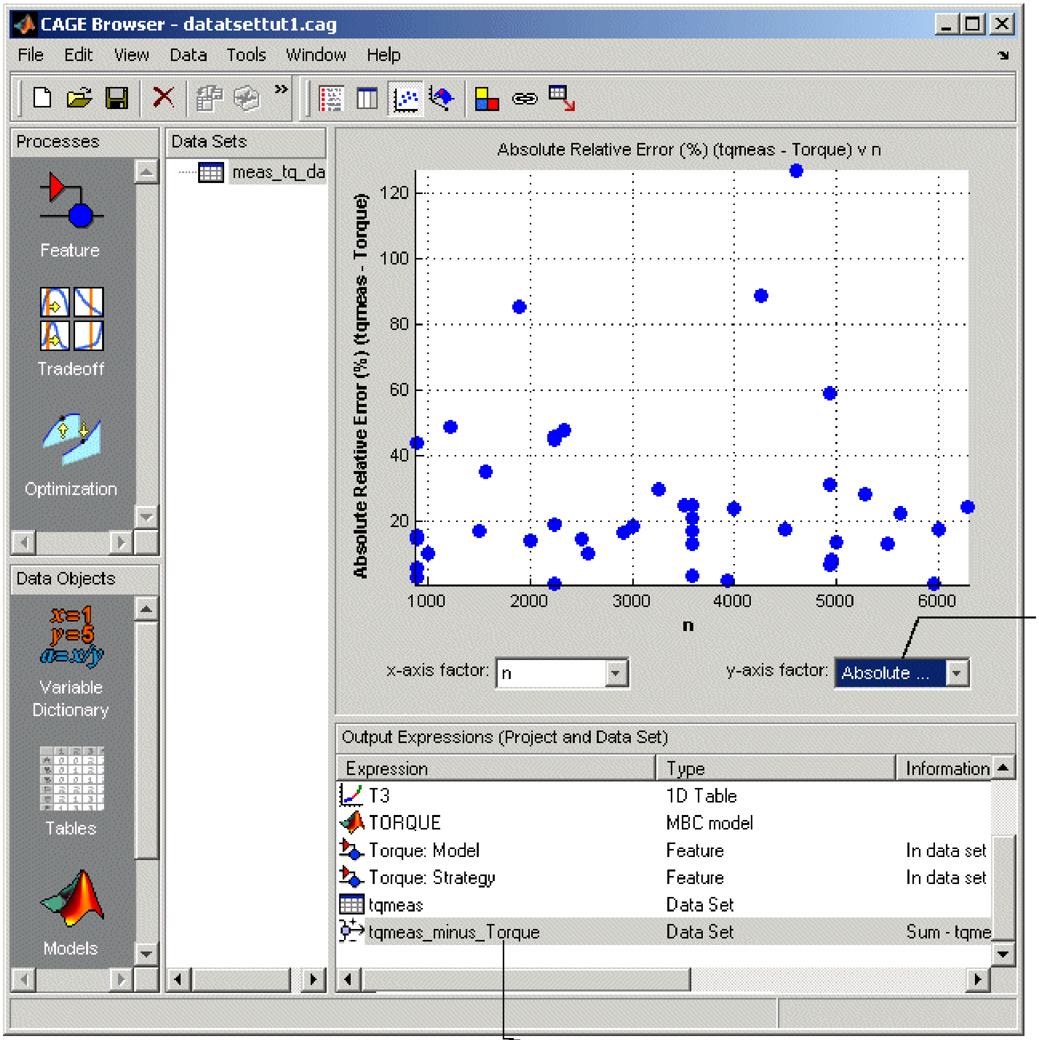
3 Change the **x-axis factor** to n from the drop-down menu.

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 - datatsettut1.cag
File Edit View Data Tools Window Help
Processes
Feature
Tradeoff
Optimization
Data Objects
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

x-axis factor: n

y-axis factor: Absolute ...

Output Expressions (Project and Data Set)

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**).

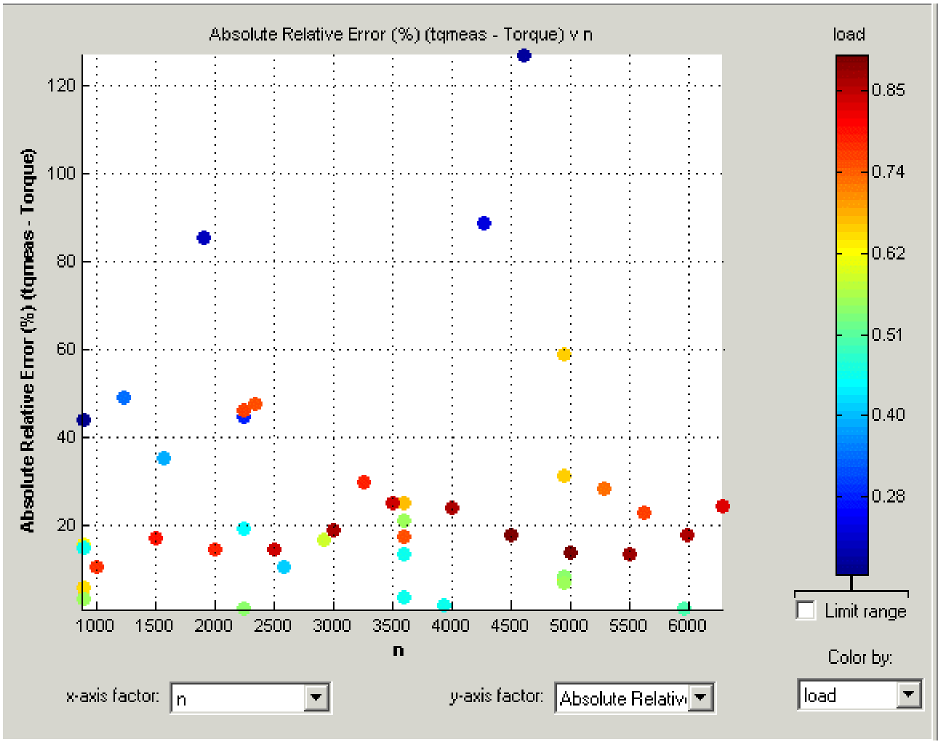
2 For the **y-axis factor**, select `Absolute Relative Error (tqmeas - Torque)` from the drop-down menu.

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`.



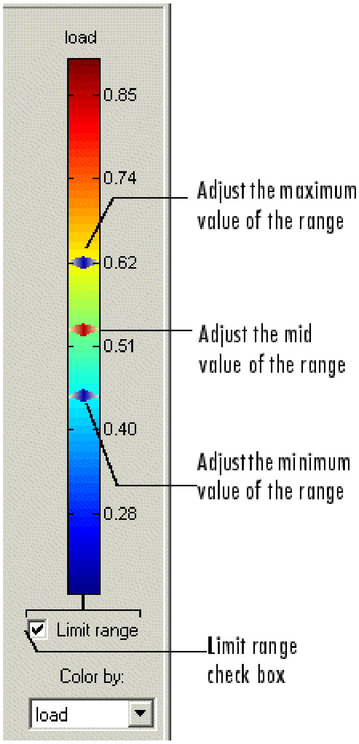
x-axis factor: n

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

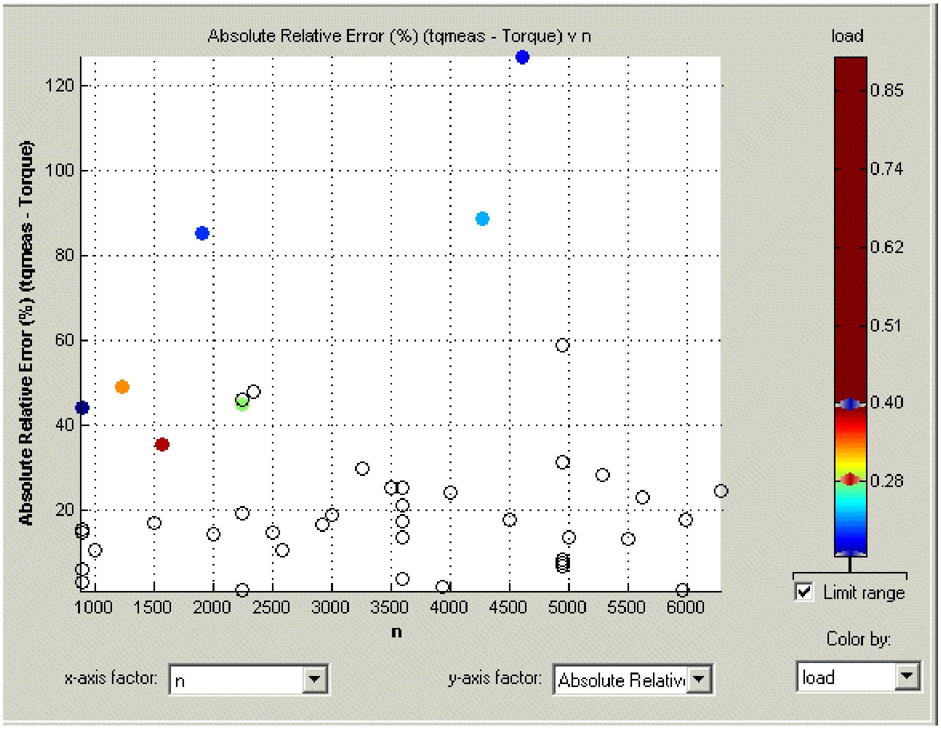

load
0.85
0.74
0.62
0.51
0.40
0.28
☑ Limit range
Color by:
load
Adjust the maximum value of the range
Adjust the mid value of the range
Adjust the minimum value of the range
Limit range check box

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

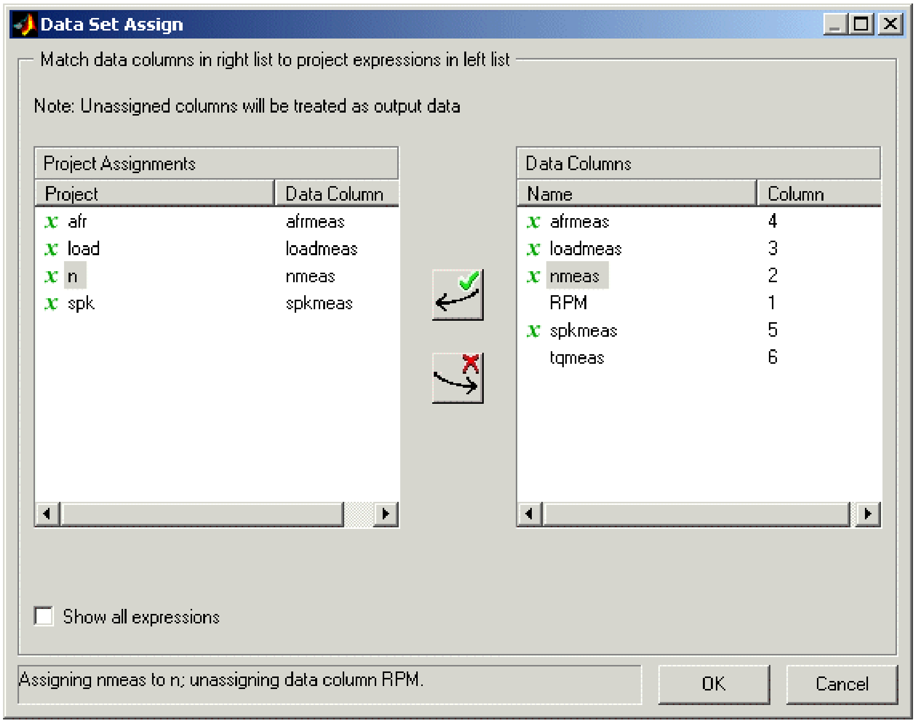
Instead of using the test cell settings for the engine speed (`RPM`), you might want to use the measured values of engine speed (`nmeas`). So you have to reassign the variable `n` to `nmeas`.

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” in the CAGE documentation.

You have now completed the data sets tutorial.

11

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 11-8 describes the process of filling the lookup table.
- "Selecting Regions of the Data" on page 11-12 describes how you can select some of the data for inclusion when you fill the table.
- "Exporting the Calibration" on page 11-14 describes how to export your completed calibration.

Start CAGE by typing

```
cage
```

at the MATLAB prompt.

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

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

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

1. Add the variables for speed and load by importing a variable dictionary.

2. Add a new table to your session.

3. Import your experimental data.

The next sections describe each of these processes in detail.

Adding Variables

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

To add a variable dictionary,

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

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

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

You can now add a table to your session.

Adding a New Table

You must add a table to fill.

To add a new table,

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

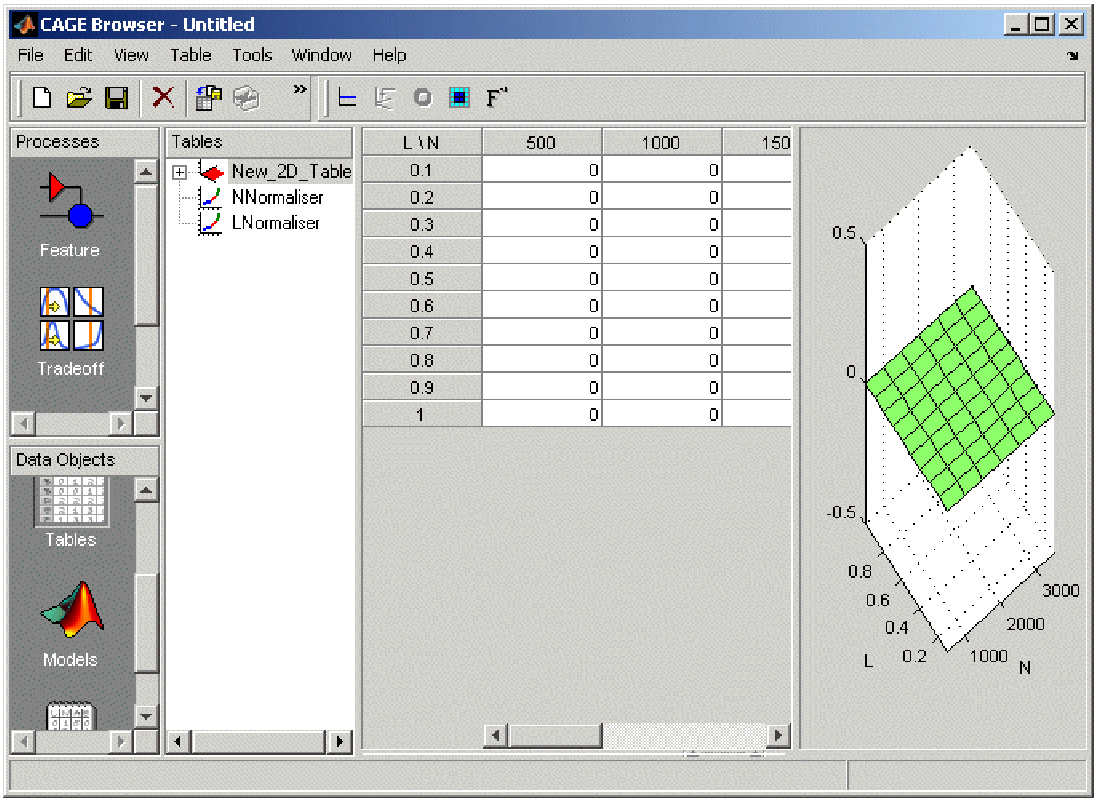
   This opens a dialog box that asks you to specify the variable names for the normalizers. As you can see in the dialog controls, accepting the defaults will create a table with ten rows and ten columns with an initial value of `0` in each cell.

2. Change the number of columns to `7`.

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

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

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

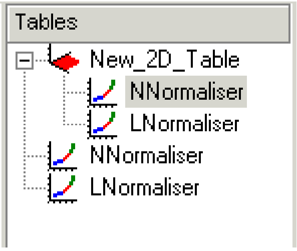


L \ N	500	1000	150
0.1	0	0	
0.2	0	0	
0.3	0	0	
0.4	0	0	
0.5	0	0	
0.6	0	0	
0.7	0	0	
0.8	0	0	
0.9	0	0	
1	0	0	

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.

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**.

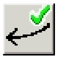
2 In the file browser, select `meas_tq_data.csv` from the `matlab\toolbox\mbc\mbctraining` directory and click **Open**.

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`).

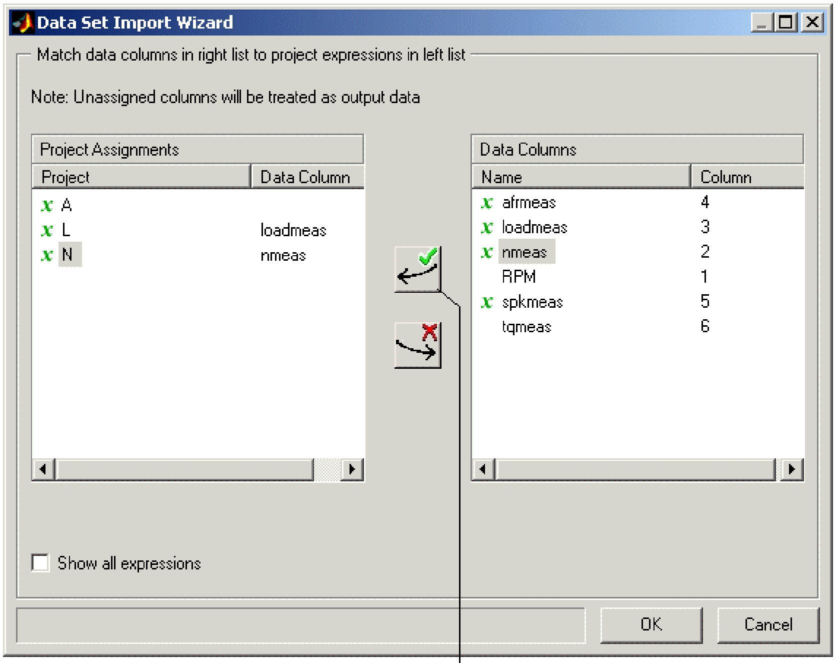
3 This opens the Data Set Import Wizard. The first screen asks which of the columns of data you want to import. Click **Next** to import them all.

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.



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

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,

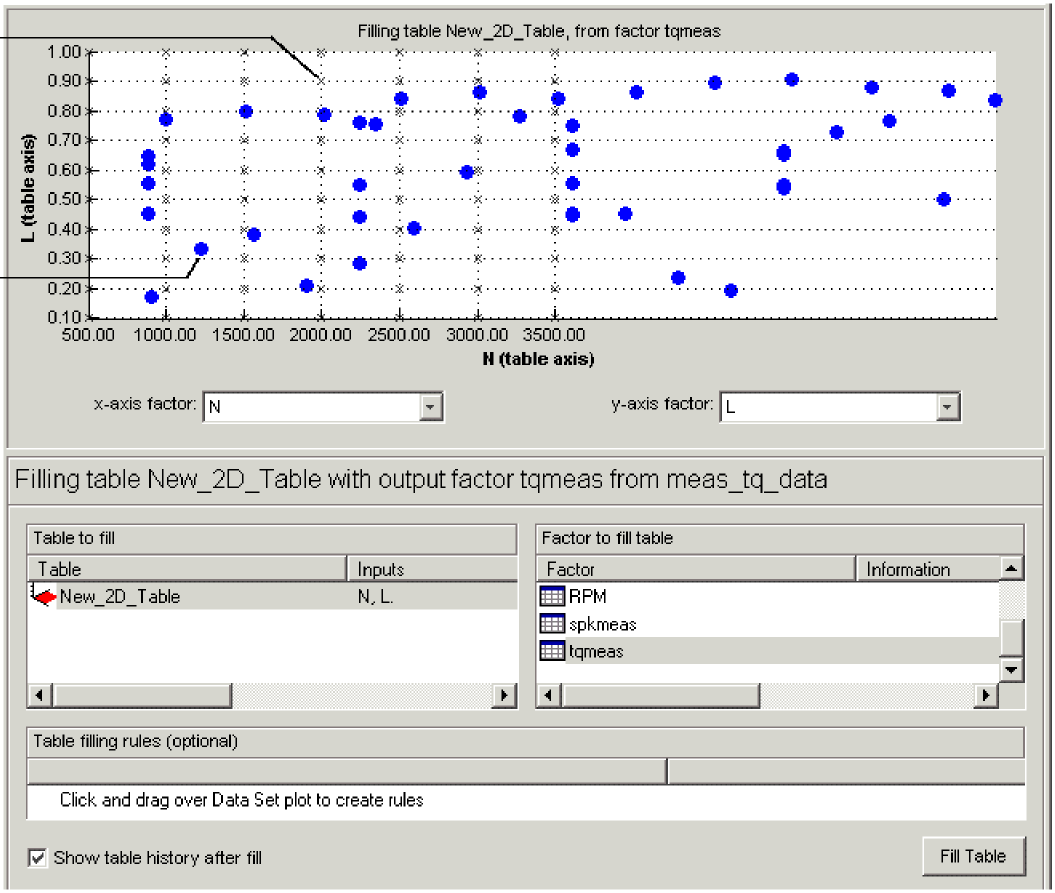
1 To view the **Table Filler** display, click (Fill Table From Data Set) in the toolbar in the **Data Sets** view; or select **View > Table Filler**.

  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.



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

A breakpoint in your lookuptable (a cross)

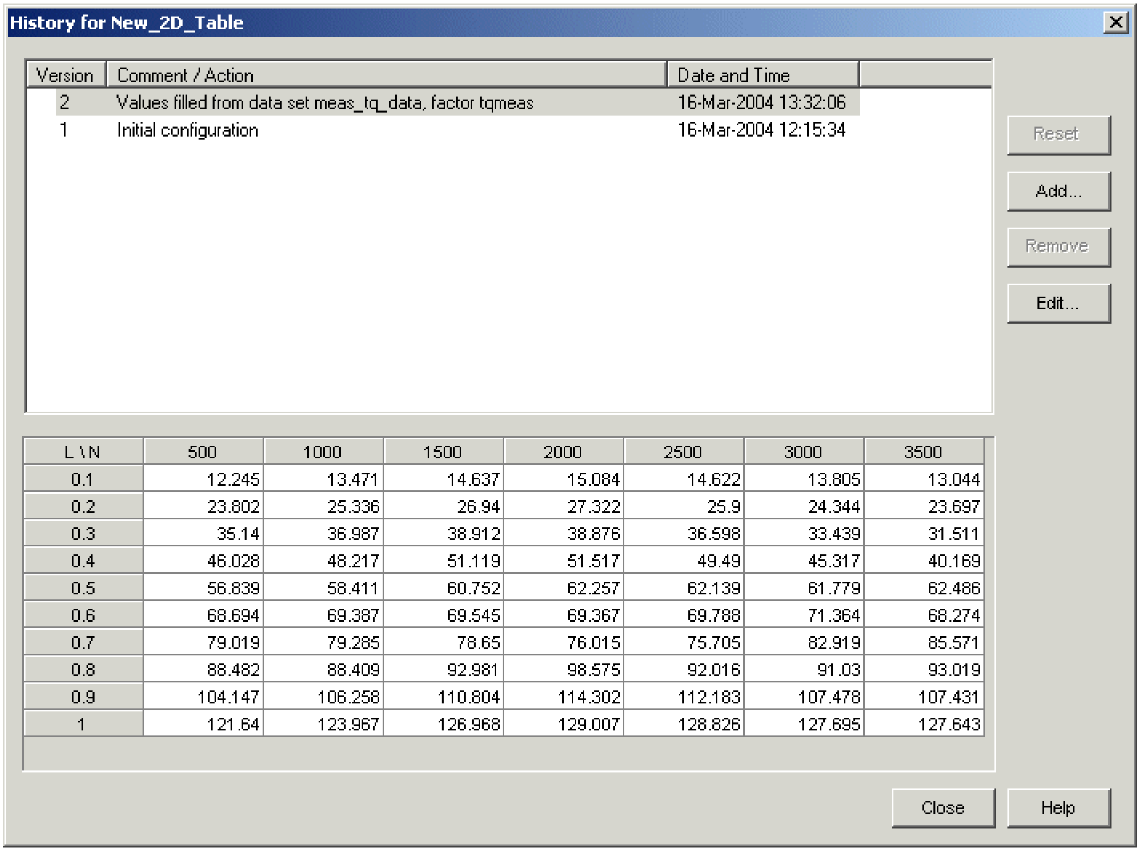
An operating point from the experimental data (a blue dot)

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.



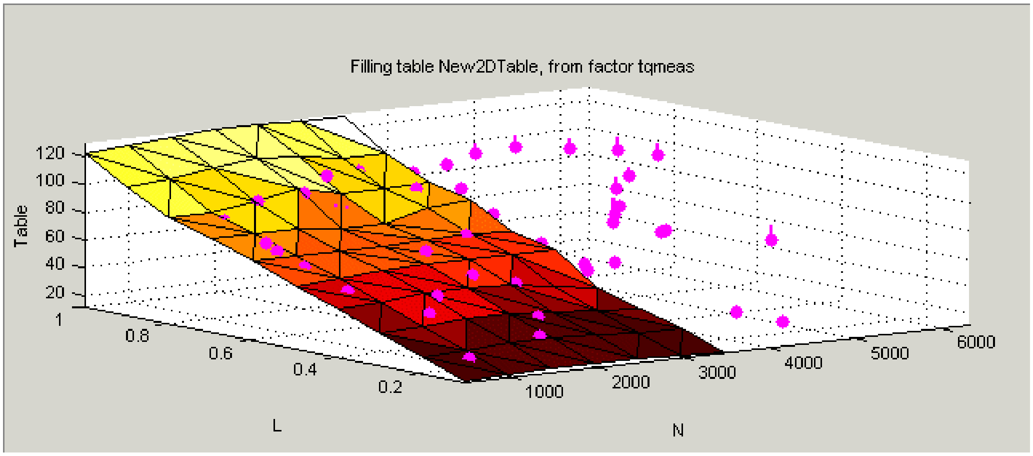
History for New_2D_Table

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

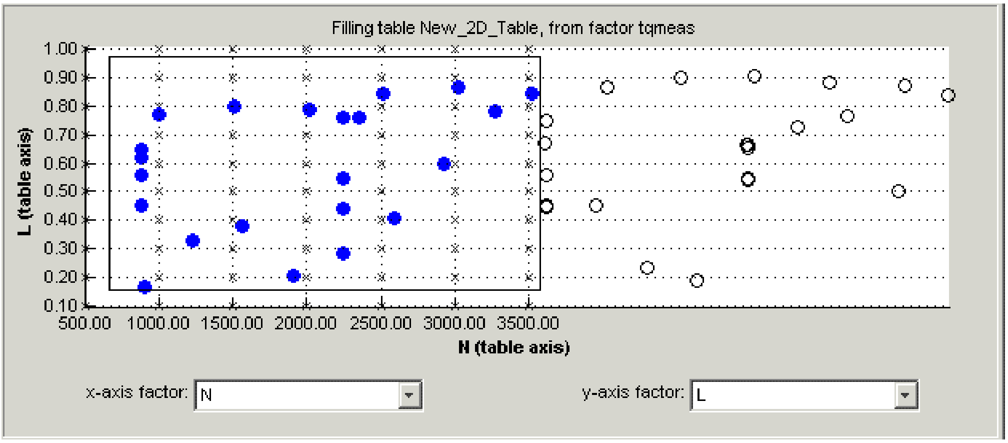
You can ignore points in the data set when you fill your lookup table.

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.

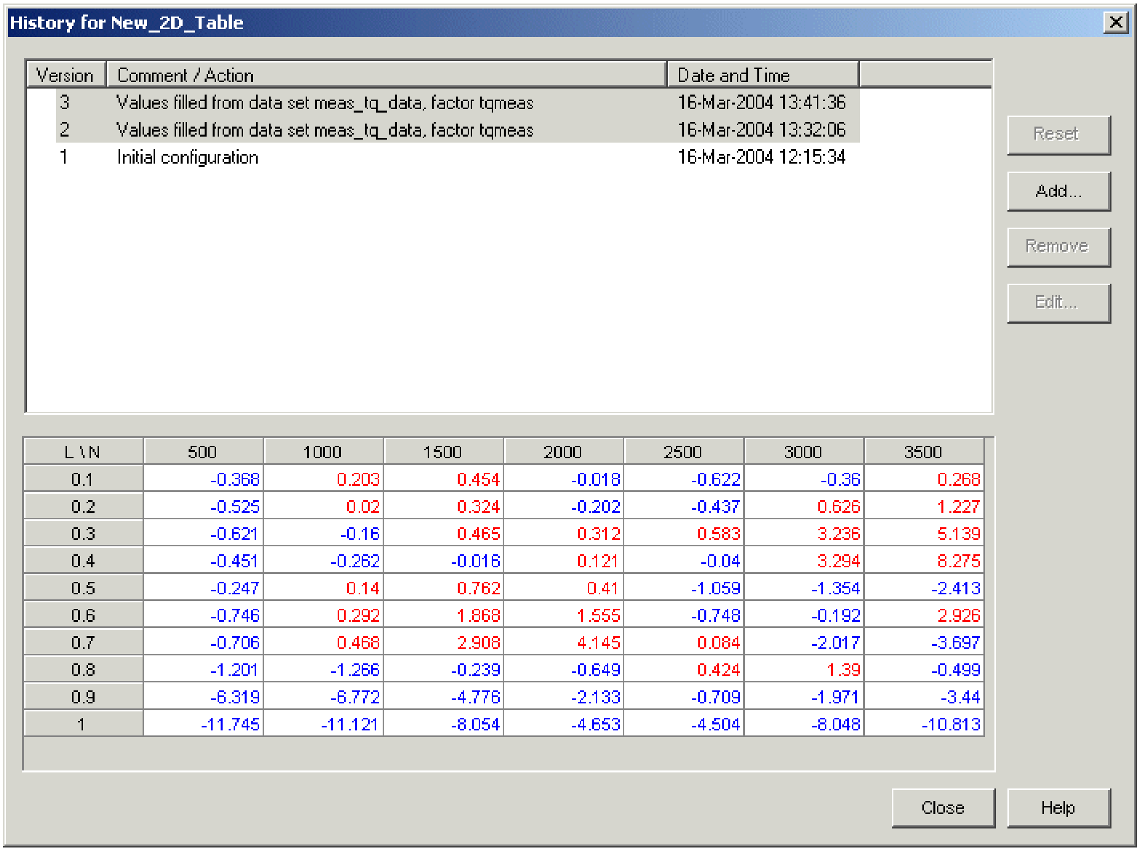


x-axis factor: N

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 2 and 3, using **Ctrl**+click. The following display shows a comparison between the table filled with two different extrapolations.



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.

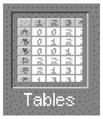
The display of the surface now shows the table filled only by reference to the data points that are included in the range of the table.

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

Exporting the Calibration

To export the calibration,

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



2 Highlight the `New_2D_Table`.

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

4 Choose the type of file you want to save your calibrations as. For the purposes of this tutorial, select `Comma Separated Value (.csv)`.

5 Enter `table_filling_tutorial.csv` as the file name and click **Save**.

This exports the calibration.

You have now completed this tutorial.

Tutorial: Optimization and Automated Tradeoff

This section includes the following topics:

Worked Example Optimization (p. 12-42)	How to use the Worked Example. We provide this to illustrate modifying the template to create your own user-defined optimizations.
Creating an Optimization from Your Own Algorithm (p. 12-48)	A detailed walk-through of the steps involved in incorporating your own algorithm into an optimization function for use in CAGE.

Tutorial Overview

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 12-6.
- A multiobjective optimization to maximize torque and minimize NOX emissions. See “Multiobjective Optimization” on page 12-20.
- A sum optimization to maximize torque while minimizing NOX, weighted to give more importance to idle speed. See “Sum Optimization” on page 12-33.
- 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 12-39.

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 Set up a new optimization.

3 Define an operating point set if required.

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.

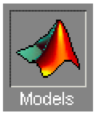
Setting Up Your Optimization Session

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 “Variables and Models” in the CAGE documentation.

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



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

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

The **Variable Dictionary** view appears, displaying the variables, constants, and formulas in the current project. The project already has the relevant variables defined, so you do not need to import a variable dictionary. Note that the variables have ranges and set points defined.

Single-Objective Optimization

The following sections describe these stages:

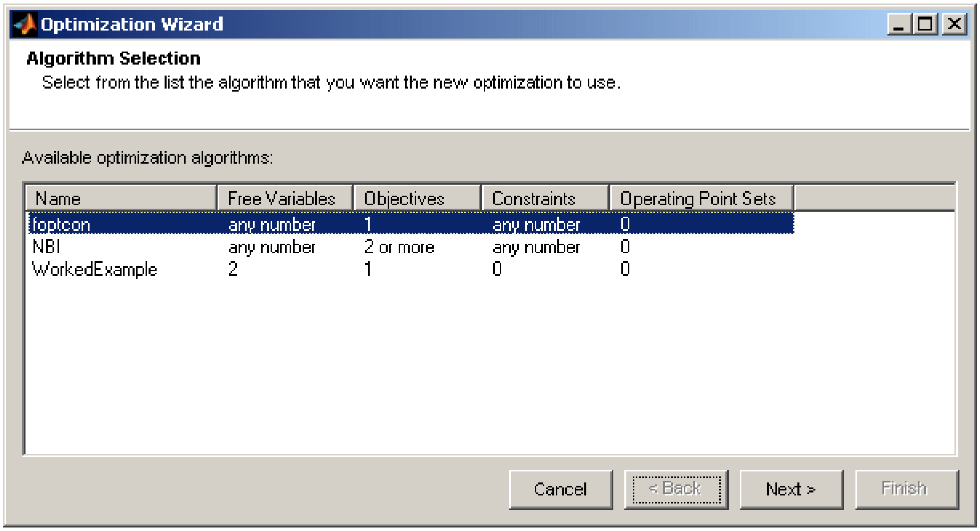
1. Using the Optimization Wizard to choose
   - Your optimization algorithm
   - How many objectives and constraints
   - What free variables to use
2. Using the Optimization view to choose
   - A model for your objective
   - A model, type, and value for your constraint
   - An operating point set for 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

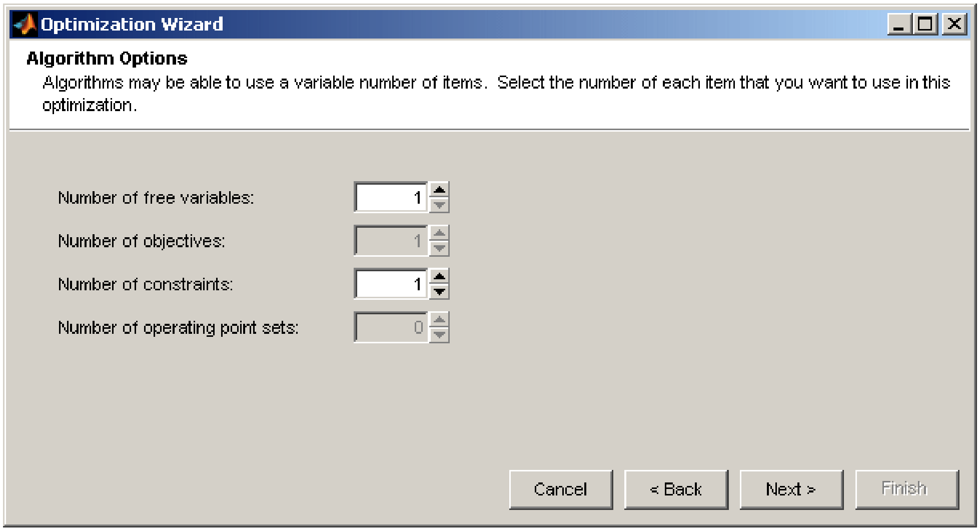
Select from the list the algorithm that you want the new optimization to use.

Available optimization algorithms:

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

2 `foptcon` is selected by default, and 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 to `1`, as shown.



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

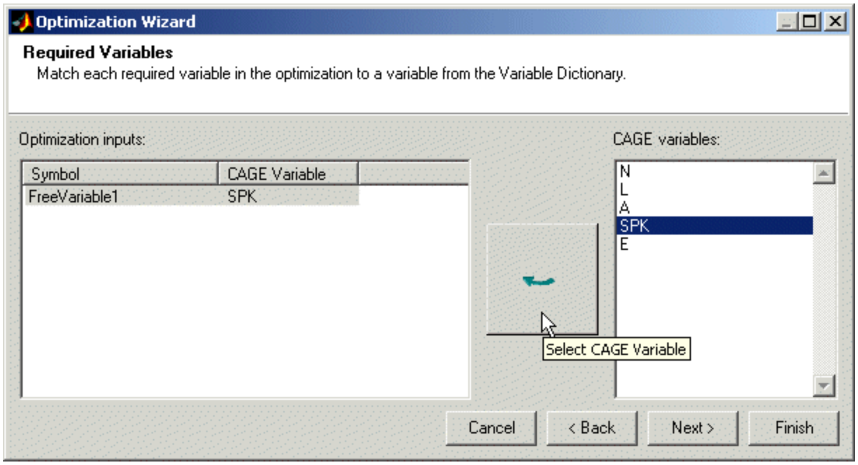
Number of objectives: 1

Number of constraints: 1

Number of operating point sets: 0

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. Click **Finish**.



Optimization Wizard

Required Variables

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

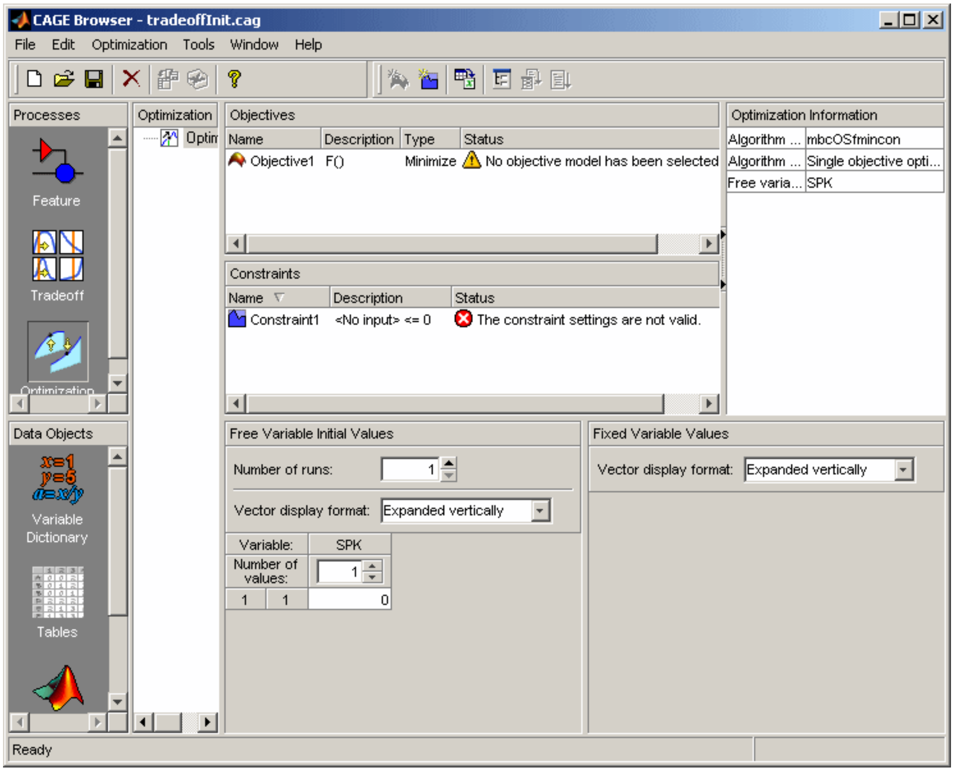
Optimization inputs:

Symbol	CAGE Variable
FreeVariable1	SPK

CAGE variables:

N
L
A
SPK
E

A new branch named Optimization 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 **Objectives** and **Constraints** panes there are status messages informing you that you need to specify a model for an objective, and valid constraint settings.



Objectives

Name	Description	Type	Status
Objective1	F()	Minimize	⚠ No objective model has been selected

Constraints

Name	Description	Status
Constraint1	<No input> <= 0	✖ The constraint settings are not valid.

Optimization Information

Algorithm ...	mbcOSfmincon
Algorithm ...	Single objective opti...
Free varia...	SPK

Free Variable Initial Values

Number of runs: 1

Vector display format: Expanded vertically

Variable:		SPK
Number of values:		1
1	1	0

Fixed Variable Values

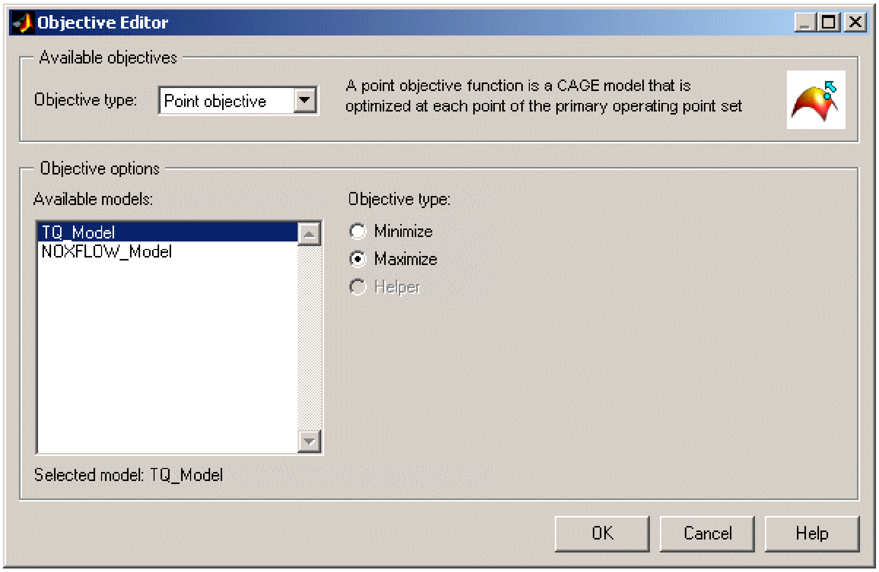
Vector display format: Expanded vertically

Ready

Setting Objectives, Constraints, and Operating Points

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

The Edit Objective dialog 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

Selected model: TQ_Model

Objective type:

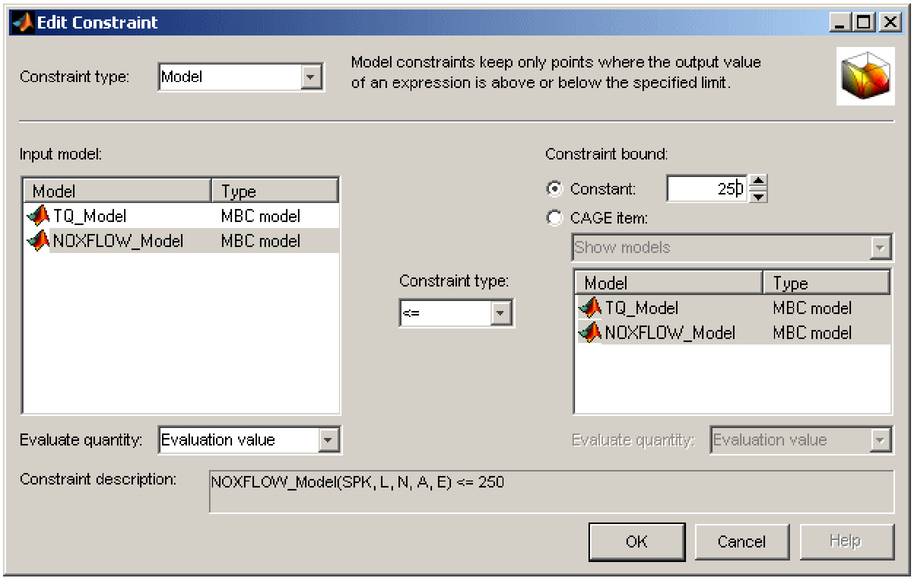
○ Minimize
● Maximize
○ Helper

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 Edit Constraint dialog appears.



Constraint type: Model

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

Input model:

Model	Type
TQ_Model	MBC model
NOXFLOW_Model	MBC model

Evaluate quantity: Evaluation value

Constraint type: <=

Constraint bound:

● Constant: 250

○ CAGE item: Show models

Model	Type
TQ_Model	MBC model
NOXFLOW_Model	MBC model

Evaluate quantity: Evaluation value

Constraint description: NOXFLOW_Model(SPK, L, N, A, E) <= 250

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

5 Select `NOXFLOW_Model` from the **Input model** list, and enter `250` in the **Constant** edit box as the maximum value for the constraint, as shown above. Make sure the inequality is <=. 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.

Note that the toolbar button Run Optimization ( ) is now enabled, because your optimization setup has provided enough information to start an optimization.

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

Running the optimization requires the selected models to be evaluated (many times over) and hence values are required for all the model input factors (L, N, A, E, and SPK). The defaults of the fixed variables (L, N, A, E) are their set points, as shown in the **Fixed Variable Values** pane. You have chosen SPK as a free variable, so the optimization will choose different values for SPK in trying to find the best. The default initial value for a free variable is the set point, as shown in the **Free Variable Initial Values** pane.

To define the set of operating points for the optimization,

a In the **Free Variable Initial Values** pane, increase the **Number of runs** to 6. Notice 6 rows appear in both fixed and free variable values panes, all containing the default set point values of each variable.

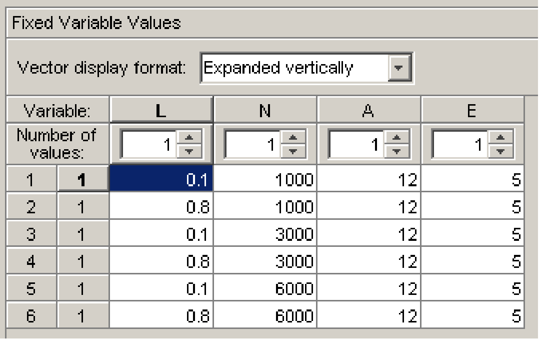
b Copy and paste these values into the N column of the **Fixed Variable Values** pane:

N
1000
1000
3000
3000
6000
6000

c Copy and paste these values into the L column of the **Fixed Variable Values** pane:

L
0.1
0.8
0.1
0.8
0.1
0.8

The **Fixed Variable Values** pane should look as shown.



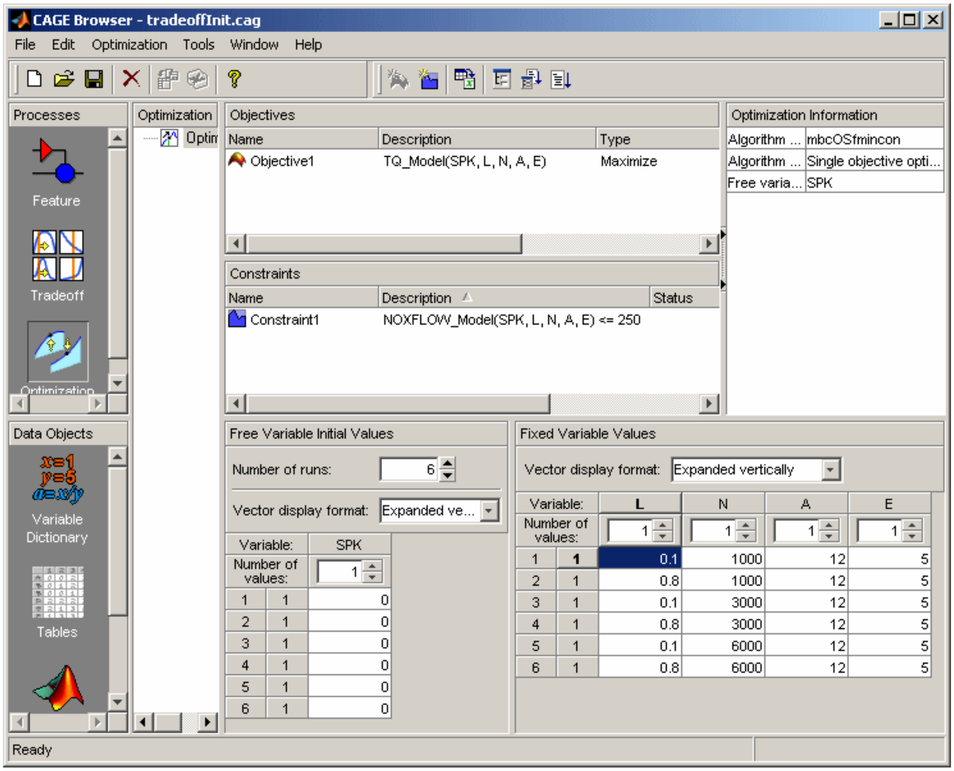
Fixed Variable Values

Vector display format: Expanded vertically

Variable:		L	N	A	E
Number of values:		1	1	1	1
1	1	0.1	1000	12	5
2	1	0.8	1000	12	5
3	1	0.1	3000	12	5
4	1	0.8	3000	12	5
5	1	0.1	6000	12	5
6	1	0.8	6000	12	5

Leave the other fixed variables and the free variable values at the defaults. If you wished to restrict the range of the free variables, you could select **Optimization > Edit Free Variable Ranges**. The default is the range of the variable as defined in the Variable Dictionary. For this example, leave the default.

7 Your CAGE Browser should now look like the following example, with an objective, constraint, and operating point set. The optimization is ready to run.



Objectives

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

Optimization Information

Algorithm ...	mbcOSfmincon
Algorithm ...	Single objective opti...
Free varia...	SPK

Constraints

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

Free Variable Initial Values

Number of runs: 6

Vector display format: Expanded ve...

Variable:		SPK
Number of values:		1
1	1	0
2	1	0
3	1	0
4	1	0
5	1	0
6	1	0

Fixed Variable Values

Vector display format: Expanded vertically

Variable:		L	N	A	E
Number of values:		1	1	1	1
1	1	0.1	1000	12	5
2	1	0.8	1000	12	5
3	1	0.1	3000	12	5
4	1	0.8	3000	12	5
5	1	0.1	6000	12	5
6	1	0.8	6000	12	5

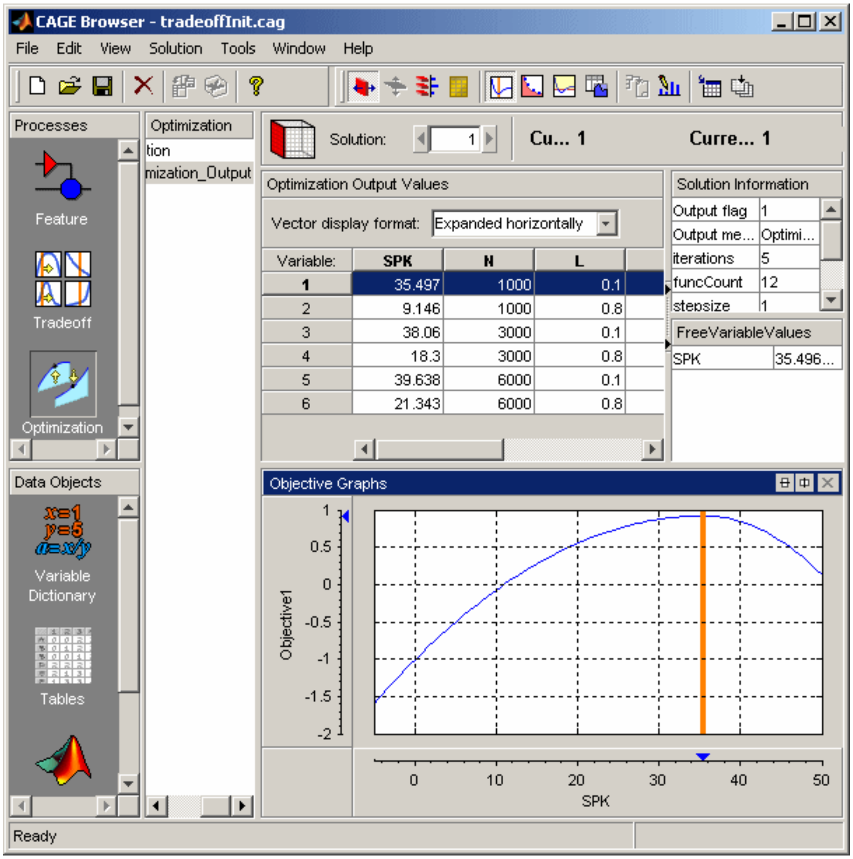
Running the Optimization

1 Click Run Optimization ( ) in the toolbar.

The optimization runs, showing progress messages as each point is evaluated until the optimization is complete. On completion of the optimization, a new node appears in the Optimization tree.

2 The view switches to the new node `Optimization_Output` where you can 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.



Vector display format: Expanded horizontally

Variable:	SPK	N	L
1	35.497	1000	0.1
2	9.146	1000	0.8
3	38.06	3000	0.1
4	18.3	3000	0.8
5	39.638	6000	0.1
6	21.343	6000	0.8

Output flag	1
Output me...	Optimi...
iterations	5
funcCount	12
stepsize	1

SPK	35.496...

Using Optimization Results to Fill Tables

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

1. Build a SPK table in N and L. Select **File > New > 2-D Table**.
2. Leave 10 in the **Rows** and **Columns** edit boxes and 0 in the **Initial Value** edit box.
3. Use the drop-down menus to select L and N for the Y and X inputs.
4. Rename the table to SPK.

5 Click **OK**. Your CAGE browser switches to the **Tables** view. CAGE has automatically initialized the normalizers to space breakpoints evenly over the ranges of `N` and `L`. The table name is now `SPK_1` as the session already has a variable called SPK.

There are two methods for filling tables with optimization results.

1 Click the Optimization button in the **Processes** pane to return to the Optimization view, and select the `Optimization_Output` node.

2 Select **Solution > Fill Tables**.

The Table Filling wizard appears.

3 Select the `SPK_1` table and click the button to add it to the list of tables to be filled. Click **Next**.

4 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.

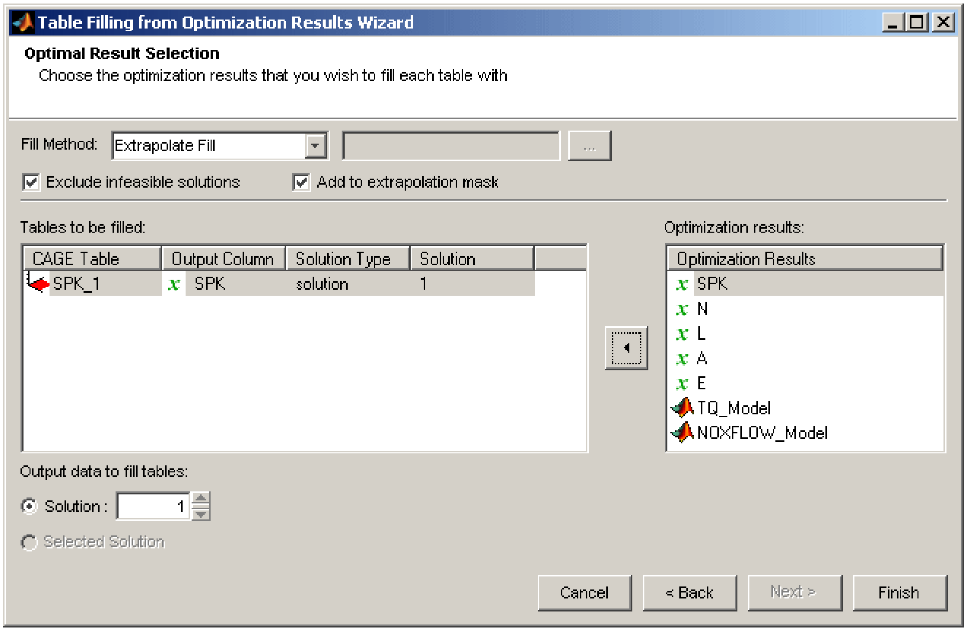


Table Filling from Optimization Results Wizard

Optimal Result Selection

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

Fill Method: Extrapolate Fill

☑ Exclude infeasible solutions ☑ Add to extrapolation mask

Tables to be filled:

CAGE Table	Output Column	Solution Type	Solution
SPK_1	SPK	solution	1

Optimization results:

Optimization Results
SPK
N
L
A
E
TQ_Model
NOXFLOW_Model

Output data to fill tables:

● Solution : 1

○ Selected Solution

5 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 `Optimization_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 the **Data Sets** button in the **Data Objects** pane) to see that the table of optimization results is contained in the new data set `Optimization_Output_sol1`. The new data set takes the name of the optimization output and the suffix `sol1` identifies that you exported solution one from the optimization solution slice 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 12-29.

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

3 Select the data set and click (Fill Table From Data Set) in the toolbar.

4 Clear the check box to **Show table history after fill**.

5 Choose to fill the spark table with the `SPK` optimization output by selecting them in the two lists, then click the button **Fill Table** at the bottom right.

6 Right-click the display and select **Surface** to see the filled table surface and the optimization output spark values.

See also Chapter 11, “Tutorial: Filling Tables from Data” for more details on using data sets to fill tables.

In the next section you will use a custom fill routine to fill the table.

Using a Custom Fill Routine to Fill Tables

It can be useful to create your own custom fill function to fill tables from the results of an optimization. Some example situations are:

- You have your own smoothing strategy for certain regions of your look-up tables
- Implementation of an alternative method to the two fill methods supplied
- You want to produce some customized output

You can use a custom fill routine to fill the `SPK_1` table from the optimization results.

1 Create a custom fill function. For this example, you can use the supplied example, `griddataTableFill.m`, which can be found in the `mbctraining` directory. Copy `griddataTableFill.m` to a directory away from the MATLAB root directory and make sure this directory is on the MATLAB path.

2 Select **Solution > Fill Tables**.

3 Select the SPK_1 table and click the button to add it to the list of tables to be filled. Click Next.

4 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.

5 Select `Custom` from the **Fill Method** drop down menu. Use the file selector, or enter the name of the fill function you wish to use to fill your tables. In this case, select or enter `griddataTableFill`. Note that this function must be on the MATLAB path.

6 Click **Finish** to fill the SPK_1 table.

In the next section you will add a multiobjective optimization to this project.

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**.

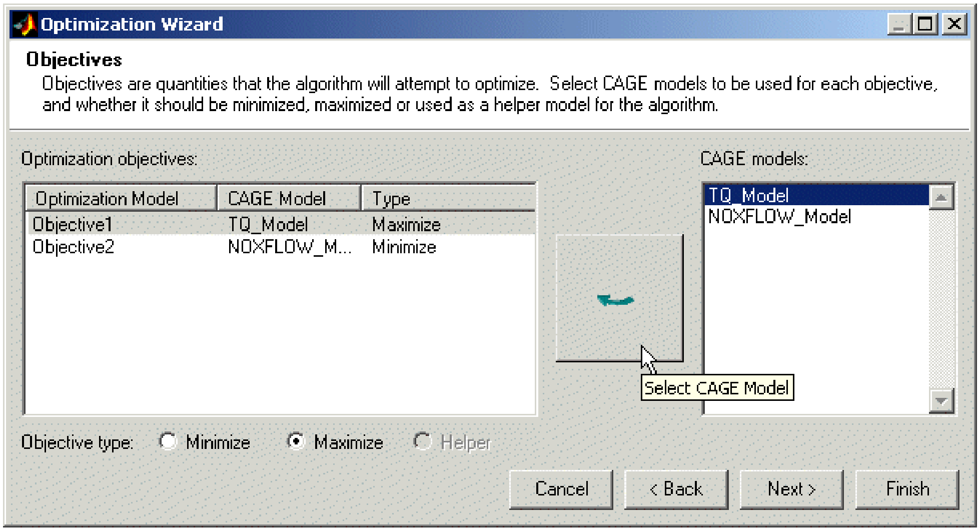
   This opens the Optimization Wizard.

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 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.



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
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`.

b Select `NOXFLOW_Model` and click the button to match it up to `Objective2`.

c Ensure `Objective1` is selected , then click the **Maximize** radio button. Check `Maximize` appears next the `Tq_Model` as shown.

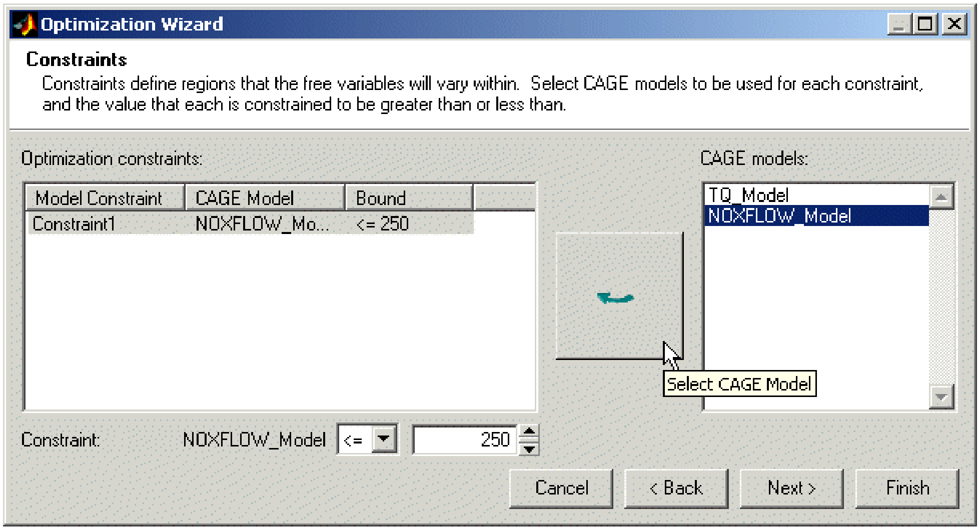
Leave the `NOXFLOW_Model` objective set to **Minimize**.

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.

a 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:

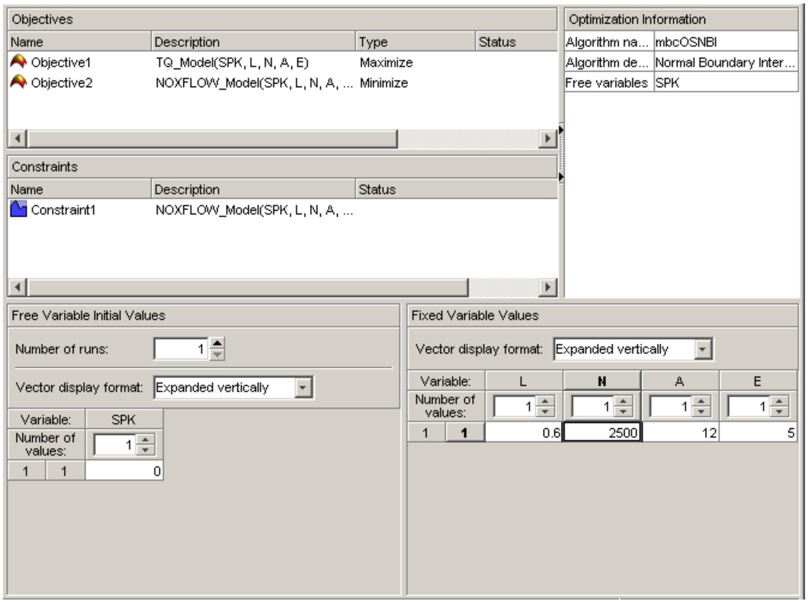
Model Constraint	CAGE Model	Bound
Constraint1	NOXFLOW_Mo...	<= 250

CAGE models:

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 **Finish**.

A new node, Optimization_1, appears in the Optimization tree. Your CAGE browser should look like the following example. Your optimization has objectives and a constraint set up and is ready to run. However unless you edit the fixed variable values it will run at a single point, the set point of the variables.



Objectives

Name	Description	Type	Status
Objective1	TQ_Model(SPK, L, N, A, E)	Maximize	
Objective2	NOXFLOW_Model(SPK, L, N, A, ...	Minimize	

Optimization Information

Algorithm na...	mbcOSNBI
Algorithm de...	Normal Boundary Inter...
Free variables	SPK

Constraints

Name	Description	Status
Constraint1	NOXFLOW_Model(SPK, L, N, A, ...	

Free Variable Initial Values

Number of runs: 1

Vector display format: Expanded vertically

	Variable:	SPK
	Number of values:	1
1	1	0

Fixed Variable Values

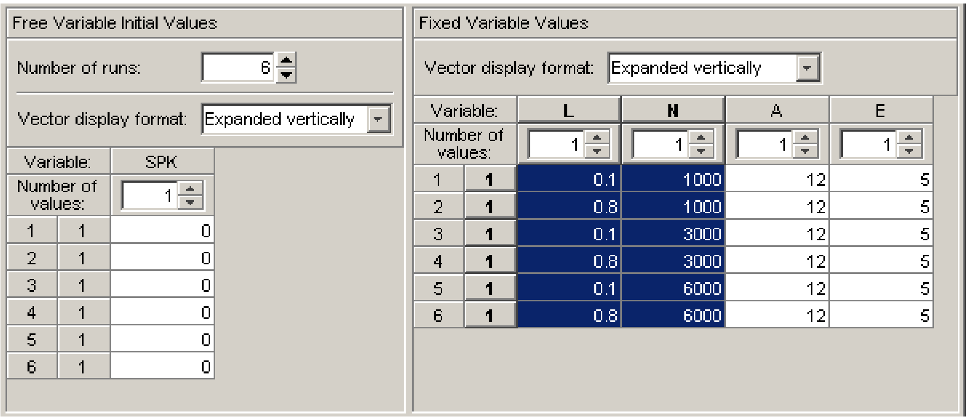
Vector display format: Expanded vertically

	Variable:	L	N	A	E
	Number of values:	1	1	1	1
1	1	0.6	2500	12	5

7 In the **Free Variable Initial Values** pane, increase the **Number of runs** to 6. Notice 6 rows appear in both fixed and free variable values panes, all containing the default set point values of each variable.

8 Select the previous single objective optimization node, click and drag to select the N and L columns in the **Fixed Variable Values** pane, and press **Ctrl+C** to copy these values.

9 Return to the new Optimization_1 node and paste these values into the N and L columns in the **Fixed Variable Values** pane.



Free Variable Initial Values

Number of runs: 6

Vector display format: Expanded vertically

Variable:		SPK
Number of values:		1
1	1	0
2	1	0
3	1	0
4	1	0
5	1	0
6	1	0

Fixed Variable Values

Vector display format: Expanded vertically

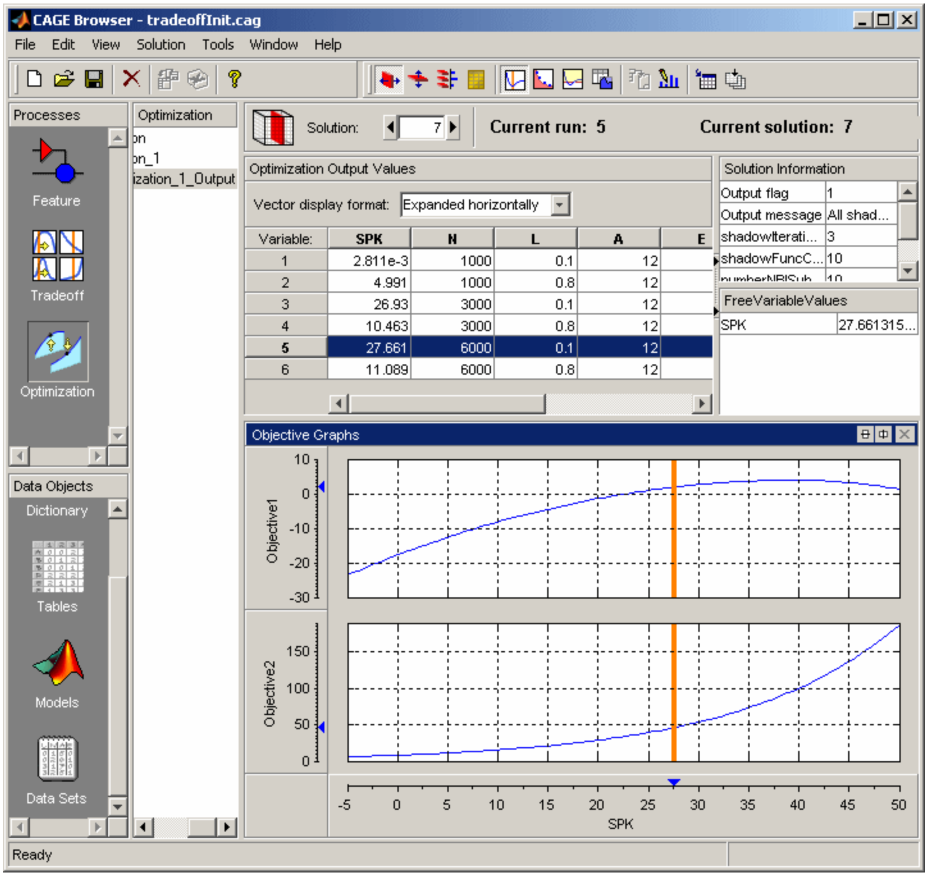
Variable:		L	N	A	E
Number of values:		1	1	1	1
1	1	0.1	1000	12	5
2	1	0.8	1000	12	5
3	1	0.1	3000	12	5
4	1	0.8	3000	12	5
5	1	0.1	6000	12	5
6	1	0.8	6000	12	5

10 Click Run Optimization ( ) in the toolbar.

The optimization runs, showing progress messages as each point is evaluated until the optimization is complete. A new node, `Optimization_1_Output`, appears under `Optimization_1` in the Optimization tree.

Optimization Output View

1 Expand the `Optimization_1_Output` node in the Optimization tree by clicking the plus sign. Then click the `NBI_Output` node to view the optimization output.



Solution: 7

Current run: 5

Current solution: 7

Optimization Output Values

Vector display format: Expanded horizontally

Variable:	SPK	N	L	A	E
1	2.811e-3	1000	0.1	12	
2	4.991	1000	0.8	12	
3	26.93	3000	0.1	12	
4	10.463	3000	0.8	12	
5	27.661	6000	0.1	12	
6	11.089	6000	0.8	12	

Solution Information

Output flag	1
Output message	All shad...
shadowIterati...	3
shadowFuncC...	10
numberNBISub	10

FreeVariableValues

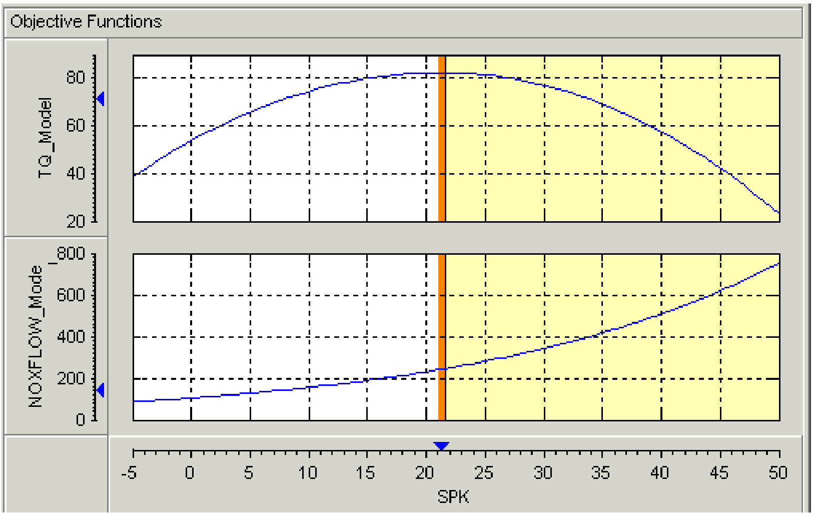
SPK	27.661315...

The toolbar buttons determine which view is displayed. The default is the Solution Slice ( ). The Solution Slice shows one solution at all operating points. That is, you can see a table 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.

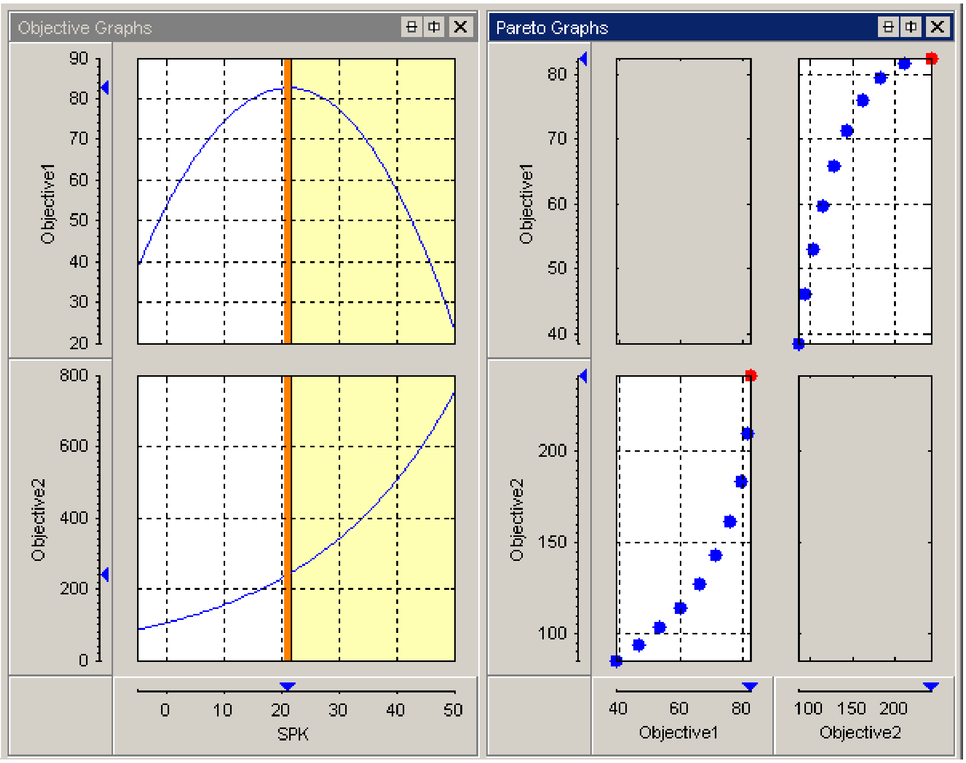
The graphs show the objective functions at the currently selected operating point (highlighted in the table), with the solution value shown in red.

Note that before you run an 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 use a boundary constraint model exported from the Model Browser as a constraint, areas outside the boundary appear in optimization graphs as yellow areas. Note that for some problems the optimization might fail to find a value within the constraints (depending on the constraints and starting values) in which case you might need to run the optimization again to find valid solutions. Choosing more suitable starting values and changing your settings to make constraints less stringent can help in these cases.



3 Right-click the Objective Graphs view and select **Split View > Pareto Graphs**.

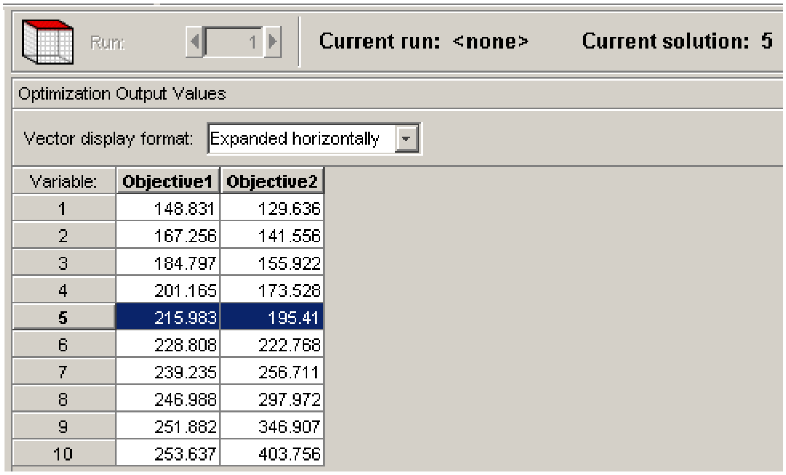


The view splits to show both objective and pareto graphs. You can right-click and select **Graph Size** to adjust how many plots can be displayed. In the Pareto Graphs view you can see all solutions found by the optimization at the selected operating point (the selected solution is highlighted in red). Try clicking different points in the pareto graph to see the different solutions in the objective graphs.

4 Click Pareto Slice ( ) in the toolbar. This changes the table to display all solutions at a single operating point. You can scroll through the operating points using the **Run** buttons at the top. The pareto graphs always show the currently selected solution in red. Click in the table or the graph to select different solutions.

Recall that the first example, a single-objective optimization, produced a single solution at each point, so you could view the Pareto Slice but it only contained one solution at each operating point. The Pareto Slice is useful to show you the set of optimal tradeoff solutions when using multiobjective optimizations, as in this case. You can use these plots to help you select the best solution for each operating point. As you can see, this example trades off NOX emissions for torque, so it is a judgment call to choose the best depending on your priorities. You will select best solutions in a later section, "Selecting Best Solutions" on page 12-29.

5 Click Weighted Pareto Slice ( ) in the toolbar.



Current run: <none>   Current solution: 5

Optimization Output Values

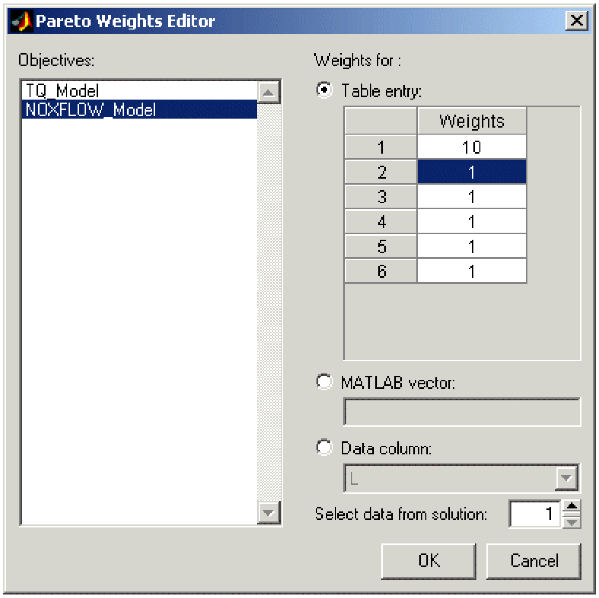
Vector display format: Expanded horizontally

Variable:	Objective1	Objective2
1	148.831	129.636
2	167.256	141.556
3	184.797	155.922
4	201.165	173.528
5	215.983	195.41
6	228.808	222.768
7	239.235	256.711
8	246.988	297.972
9	251.882	346.907
10	253.637	403.756

This table view displays a weighted sum objective output across all operating points for each solution.

The value in the `Objective2` 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.

6 You can alter these weights by clicking Edit Pareto Weights ( ) in the toolbar. The Pareto Weights Editor appears.



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, 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 Slice do not alter the results of your optimization as seen in other views. You can use the weighted sums to investigate your results only. You need to perform a sum optimization if you want to optimize using weighted operating points.

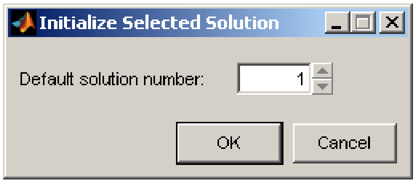
Selecting Best Solutions

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

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

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

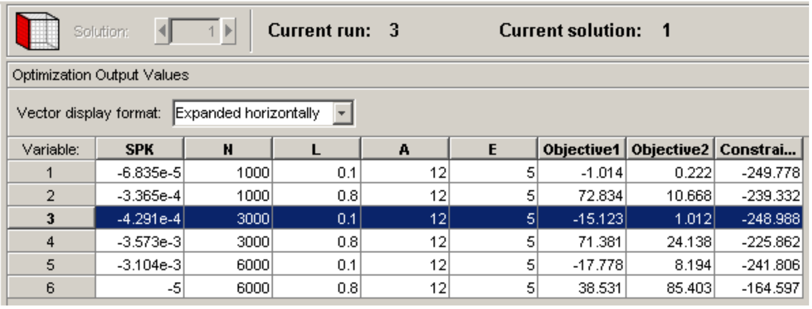
A dialog called Initialize Selected Solution appears. Click **OK**.



Default solution number: 1

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

2 Click the Selected Solution Slice button (▦) which is now enabled in the toolbar. Observe that the **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 Slice and Solution Slice to decide which solution is best for each point.



Current run: 3

Current solution: 1

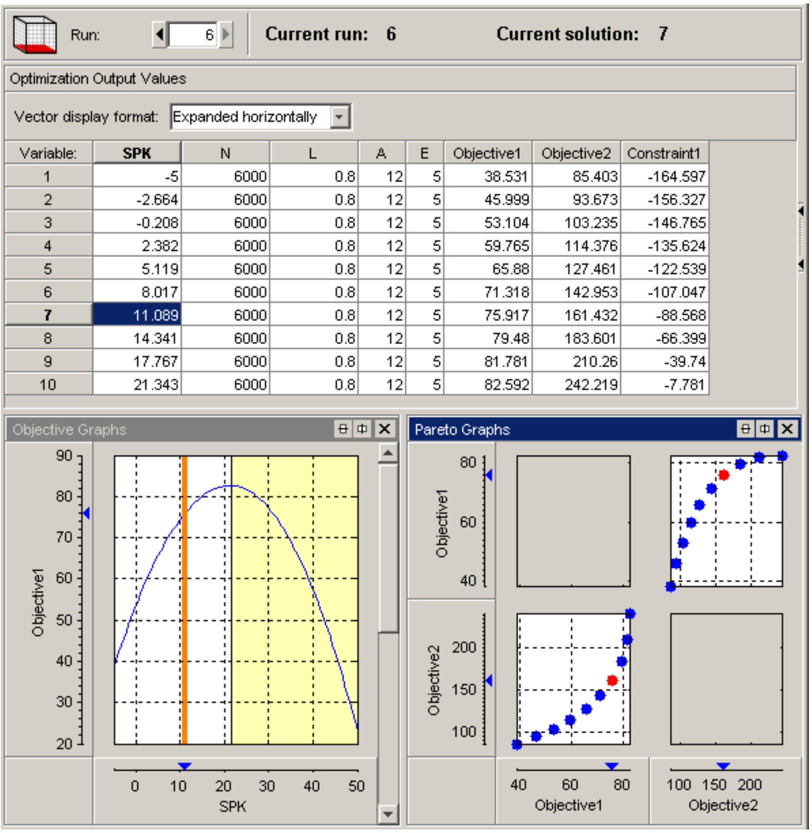
Optimization Output Values

Vector display format: Expanded horizontally

Variable:	SPK	N	L	A	E	Objective1	Objective2	Constrai...
1	-6.835e-5	1000	0.1	12	5	-1.014	0.222	-249.778
2	-3.365e-4	1000	0.8	12	5	72.834	10.668	-239.332
3	-4.291e-4	3000	0.1	12	5	-15.123	1.012	-248.988
4	-3.573e-3	3000	0.8	12	5	71.381	24.138	-225.862
5	-3.104e-3	6000	0.1	12	5	-17.778	8.194	-241.806
6	-5	6000	0.8	12	5	38.531	85.403	-164.597

3 Return to the Pareto Slice and select a solution for run 6. 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**.



Run: 6 Current run: 6 Current solution: 7

Optimization Output Values

Vector display format: Expanded horizontally

Variable:	SPK	N	L	A	E	Objective1	Objective2	Constraint1
1	-5	6000	0.8	12	5	38.531	85.403	-164.597
2	-2.664	6000	0.8	12	5	45.999	93.673	-156.327
3	-0.208	6000	0.8	12	5	53.104	103.235	-146.765
4	2.382	6000	0.8	12	5	59.765	114.376	-135.624
5	5.119	6000	0.8	12	5	65.88	127.461	-122.539
6	8.017	6000	0.8	12	5	71.318	142.953	-107.047
7	11.089	6000	0.8	12	5	75.917	161.432	-88.568
8	14.341	6000	0.8	12	5	79.48	183.601	-66.399
9	17.767	6000	0.8	12	5	81.781	210.26	-39.74
10	21.343	6000	0.8	12	5	82.592	242.219	-7.781

4 If you return to the Selected Solution Slice you can now see that solution 7 is now present for operating point 6, 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 use the Table Filling From Optimization Results Wizard (**Solution > Fill Tables**) and choose to fill with **Selected Solution** on screen 2 of the wizard.

Alternatively you could fill from a data set containing the selected solutions. To do this, click Export to Data Set ( ). Go to the **Data Sets** view (click

Data Sets in the **Data Objects** pane) to see that the table of optimization results is contained in a new data set. You could use these optimization results to fill tables. Both these table-filling methods are described in "Using Optimization Results to Fill Tables" on page 12-15.

Note that the table in the current view is exported to the data set. If you want to export your selected best solutions for each operating point, make sure you display the Selected Solution Slice before exporting the data. If you export from the Pareto Slice, the new data set contains all solutions at the single currently selected operating point set. If you export from the Solution Slice the new data set will contain the current solution at all operating points.

Recall that the previous example was a single-objective optimization and therefore only had one solution per operating point. In that case the optimization results could be exported directly from the Solution Slice, as there was no choice of solutions to be selected. See "Single-Objective Optimization" on page 12-6.

In the next tutorial section you will duplicate this NBI optimization example and alter it to create a sum optimization.

Sum Optimization

In this exercise you will use a copy of the NBI optimization from the last example to create a sum optimization.

Up to this point, you have found the optimal values of each objective function at each point of an operating point set individually. A sum optimization finds the optimal value of a weighted sum of each objective function over all free variables simultaneously. The weighted sum is taken over each operating point in the run, and the weights can be edited.

A sum optimization problem without constraints, for M operating points, is the same as M individual optimizations; the minimum of the sum is the same as the sum of the minima.

To illustrate this, consider the following example:

Say the objective function is *f(a,b)*. Consider *a* to be the free variable and *b* to be the fixed variable. To set up a sum optimization, at different values of *b*, we want to find [*a1, a2, a3 .. aM*] which minimizes:

*w1*f(a1,b1) + w2*f(a2,b2) + w3*f(a3,b3) + ... + wM*f(aM,bM)*, [1]

where *w1*, *w2* etc. are the weights.

There are two ways of viewing this problem. It can be viewed as a big optimization problem in an *M*-dimensional space for the vector [*a1, a2, a3, ..aM*] as shown in [1]. Alternatively, as each element of the sum depends on its own subset of the free variables, the problem can be written as *M* separate optimization problems, as in [2]:

*min wi*f(ai,bi) for i=1:M, [2]*

Once the *M* individual problems are solved, the weighted sum can be constructed to get the answer.

When there are sum constraints present, it is not true that a M-point sum optimization problem is equivalent to solving M individual optimizations. In this case, all points must be evaluated together to find the optimal solution that meets the sum constraints across all of the points.

In order to create a sum optimization, all objectives must be sum objectives. You can use a mixture of point and model sum constraints, but for an `fmincon` optimization, if you include a sum constraint, then all objectives *must* be sum objectives. The following instructions describe these settings.

1 Right-click the `Optimization_1` node in the Optimization tree and select **Duplicate Optimization_1**.

   A copy of the `Optimization_1` node called `Optimization_2` appears in the tree. You do not need an existing NBI optimization to create a sum optimization. This example is used for convenience and also to illustrate copying optimizations. This feature can be useful when you are trying different settings to improve your optimizations while keeping previous attempts for comparison.

   You use this copy to create a sum optimization. This means that instead of performing the optimization at each point individually, the optimization takes the sum of solutions at all points into consideration. You can apply different weights to operating points, allowing more flexibility for some parts of the optimization.

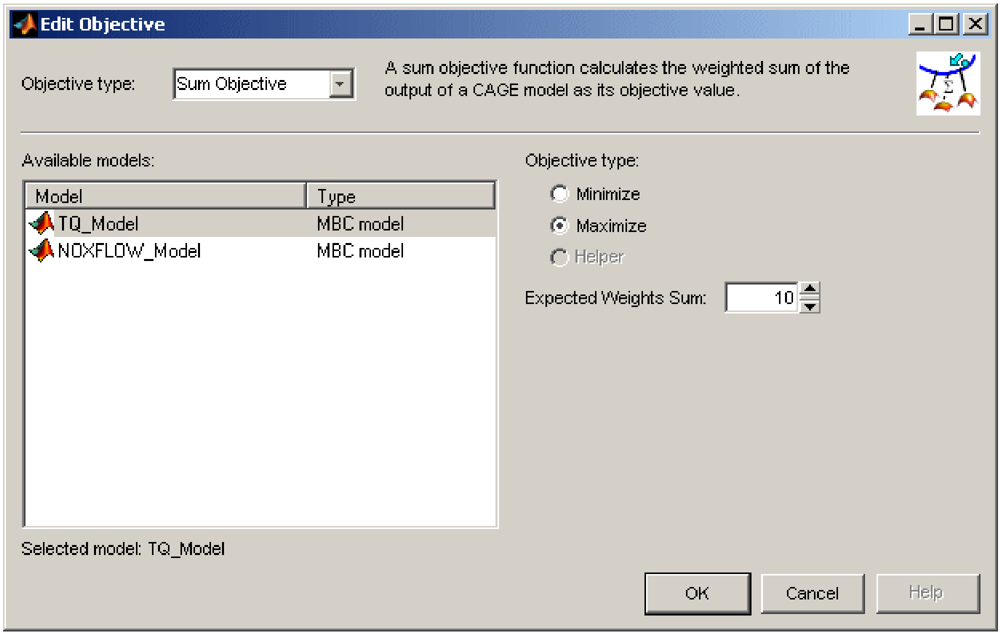
2 Select the node `Optimization_2` and select **Edit > Rename** (or press **F2**). Edit the name to read `SUM_NBI`.

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

3 Double-click `Objective1` (or right-click and select **Edit Objective**). The Edit Objective dialog appears.

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

   You need to set up your new objective.



Edit Objective

Objective type: Sum Objective

A sum objective function calculates the weighted sum of the output of a CAGE model as its objective value.

Available models:

Model	Type
TQ_Model	MBC model
NOXFLOW_Model	MBC model

Objective type:

○ Minimize

● Maximize

○ Helper

Expected Weights Sum: 10

Selected model: TQ_Model

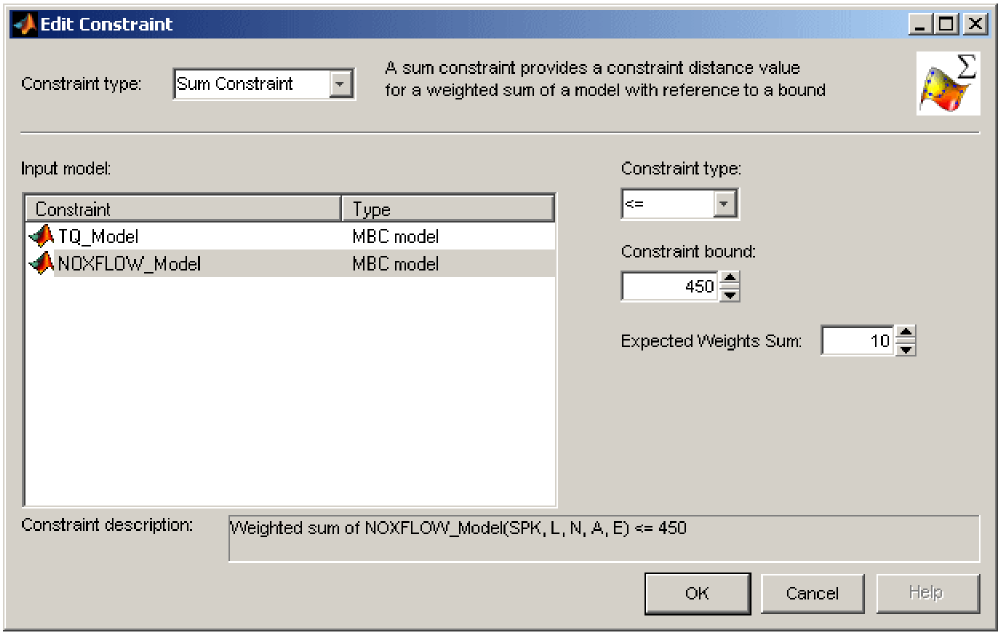
5 Select `TQ_Model` and **Maximize**. Torque has a strong correlation with fuel consumption so this sort of problem could be useful for a fuel consumption study.

6 You can edit the weights to make certain operating points more important (e.g. engine idle speed), giving more flexibility to other points. You can edit the weights later in the **Optimization** view, in the **Fixed Variable Values** pane. Here the **Expected sum of weights** is a normalization value per run which can improve the performance of the optimization. All weights are divided by this, to try to make the normalized weights sum to 1 for each run. With this normalization, the objective sum should approximately vary on the same range as the other optimization objectives and constraints. As in the example shown, edit the **Expected sum of weights** to `10`. When you edit the weights later, they will sum to 10.

7 Click **OK** to finish editing the objective.

8 Repeat to edit `Objective2`. Set up a sum objective to minimize NOXFLOW, with a weights sum of 10.

9 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 type: Sum Constraint

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

Input model:

Constraint	Type
TQ_Model	MBC model
NOXFLOW_Model	MBC model

Constraint type: <=

Constraint bound: 450

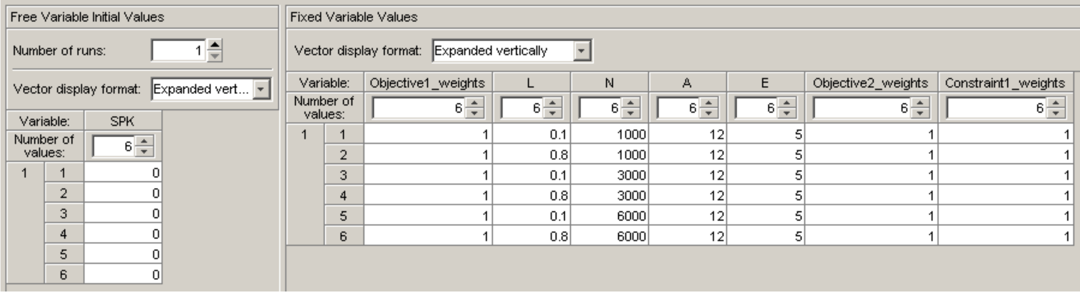
Expected Weights Sum: 10

Constraint description: Weighted sum of NOXFLOW_Model(SPK, L, N, A, E) <= 450

10 Select `Sum Constraint` from the **Constraint Type** drop-down menu, then ensure that `NOXFLOW_Model` is selected under **Input Models**.

11 Enter `450` in the constraint bound edit box, and enter `10` for the **Expected sum of weights**. Click **OK**.

12 You want to perform a sum optimization across a series of operating points. Select **Optimization > Convert to Single Run**. Look at the change in the **Variable Values** panes. Instead of 6 separate runs you now have one run containing 6 operating points. **Number of runs** is now 1, and **Number of Values** is 6 for all variables (you use these controls to set the number of operating points within each run). Each solution will be a sum across all the points in the run.



Free Variable Initial Values

Number of runs: 1

Vector display format: Expanded vert...

Variable:		SPK
Number of values:		6
1	1	0
	2	0
	3	0
	4	0
	5	0
	6	0

Fixed Variable Values

Vector display format: Expanded vertically

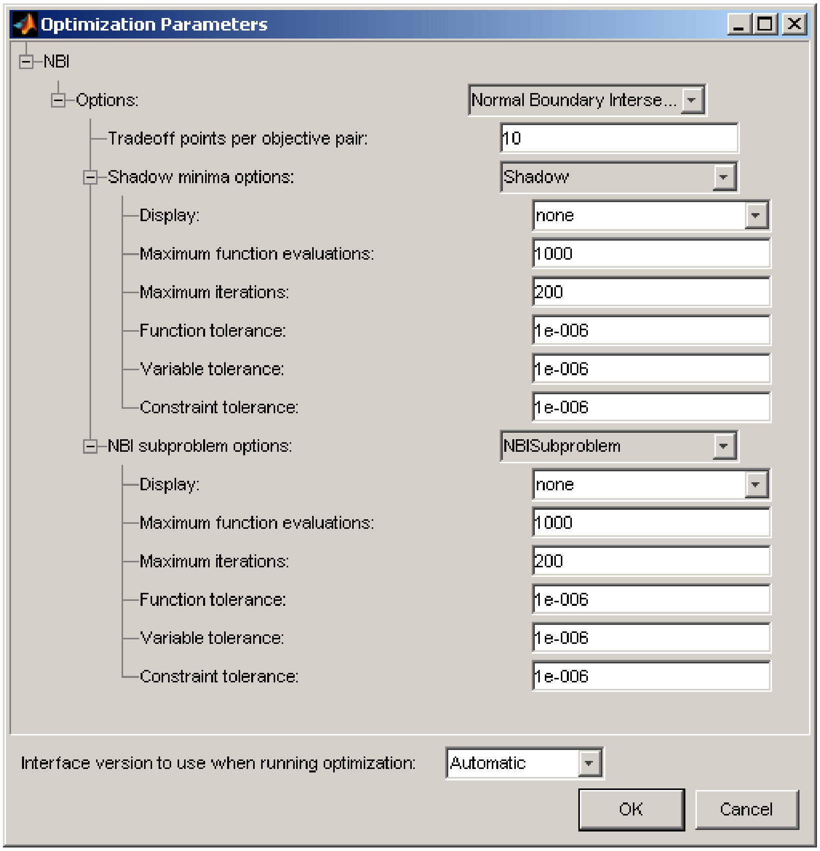
Variable:		Objective1_weights	L	N	A	E	Objective2_weights	Constraint1_weights
Number of values:		6	6	6	6	6	6	6
1	1	1	0.1	1000	12	5	1	1
	2	1	0.8	1000	12	5	1	1
	3	1	0.1	3000	12	5	1	1
	4	1	0.8	3000	12	5	1	1
	5	1	0.1	6000	12	5	1	1
	6	1	0.8	6000	12	5	1	1

13 Look in the **Fixed Variable Values** pane for the weights columns. As shown, enter 5 in the first row (run 1, point 1) for **Objective1_weights**, **Objective2_weights**, and **Constraint1_weights**, to weight the first operating point by five. Note that these weights add up to 10 for run 1, to match what you specified in the **Expected sum of weights** in the objective and constraint editors.

14 Click Set Up Optimization in the toolbar. Change the following parameters in the Optimization Parameters dialog.

For both Shadow minima and NBI subproblem options:

- **Maximum function evaluations** — 1000
- **Maximum iterations** — 200
- **Function tolerance** — 1e-006
- **Variable tolerance** — 1e-006
- **Constraint tolerance** — 1e-006



Optimization Parameters

- NBI
  - Options: Normal Boundary Interse...
    - Tradeoff points per objective pair: 10
    - Shadow minima options: Shadow
      - Display: none
      - Maximum function evaluations: 1000
      - Maximum iterations: 200
      - Function tolerance: 1e-006
      - Variable tolerance: 1e-006
      - Constraint tolerance: 1e-006
    - NBI subproblem options: NBISubproblem
      - Display: none
      - Maximum function evaluations: 1000
      - Maximum iterations: 200
      - Function tolerance: 1e-006
      - Variable tolerance: 1e-006
      - Constraint tolerance: 1e-006

Interface version to use when running optimization: Automatic

Click **OK** to close the dialog.

15 You have modified your objectives and constraint for a sum optimization, which is ready to run. Click Run Optimization ( ) in the toolbar.

16 There is a wait notice as the optimization runs. There are no progress messages as points are evaluated because sum optimizations do not evaluate points individually. When the optimization is complete, examine the results at the output node.

Automated Tradeoff

Once you have set up an optimization you can fill tables in a tradeoff using automated tradeoff. You can select cells and fill them from the results of an optimization. The cells you select in the tradeoff table define the operating point set for the optimization.

Set up a tradeoff as follows (also described in "Creating a Tradeoff Calibration" on page 9-3).

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 Setup dialog.

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

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

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

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

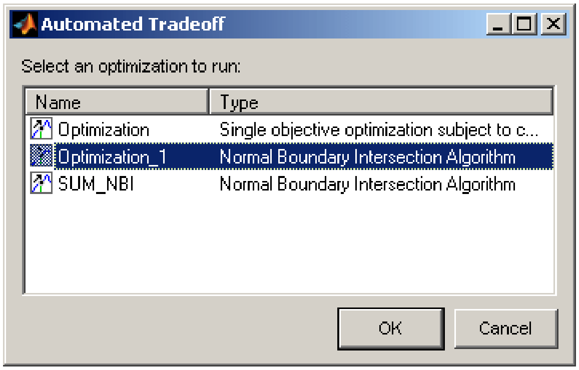
1 Highlight a rectangle of cells in the SPK table by clicking and dragging. Note that a large region can take a very long time to evaluate. Try four cells to start with.

2 Click (or right-click and select **Extrapolation Regions > Add Selection**) to define the region. The cells become light blue.

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

1 Select **Inputs > Automated Tradeoff** or click the toolbar button .

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



Automated Tradeoff

Select an optimization to run:

Name	Type
Optimization	Single objective optimization subject to c...
Optimization_1	Normal Boundary Intersection Algorithm
SUM_NBI	Normal Boundary Intersection Algorithm

2 Select your `Optimization_1` multiobjective optimization to apply to the tradeoff, and click **OK** in the following dialog to optimize only the table cells that are in the region, rather than all cells.

The automated tradeoff optimization runs. The results appear in the selected cells in the table, as shown in the example. You could optimize several regions and then use these results to extrapolate across the whole table.

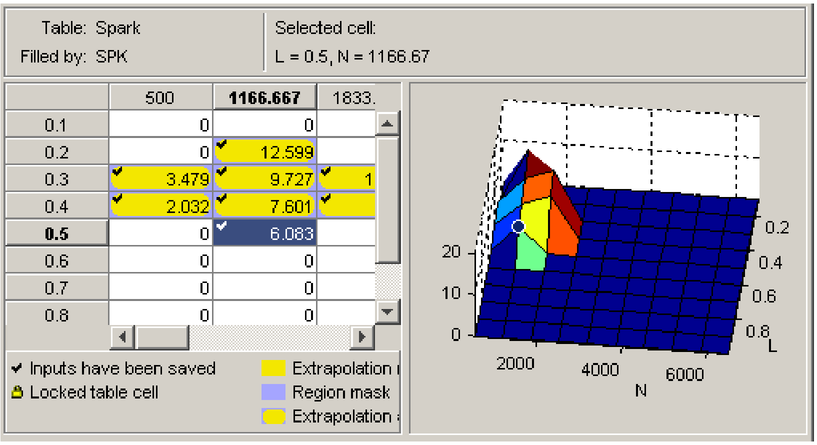


Table: Spark
Filled by: SPK

Selected cell:
L = 0.5, N = 1166.67

	500	1166.667	1833.
0.1	0	0	
0.2	0	✓ 12.599	
0.3	✓ 3.479	✓ 9.727	✓ 1
0.4	✓ 2.032	✓ 7.601	✓
0.5	0	✓ 6.083	
0.6	0	0	
0.7	0	0	
0.8	0	0	

✓ Inputs have been saved
🔒 Locked table cell
🟨 Extrapolation
▨ Region mask
🟨 Extrapolation

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.

- 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, in this case speed = 1000, load = 0.2, 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 algorithms within CAGE. You can find detailed information on all the available CAGE optimization interface functions in “User-Defined Optimization” in the CAGE documentation.

- 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.

The next section, “Using the Worked Example Optimization” on page 12-43, demonstrates how to use the example within CAGE.

The section “Creating an Optimization from Your Own Algorithm” on page 12-48 is a detailed tutorial example explaining how to incorporate an example user-defined optimization algorithm into a CAGE optimization function.

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.

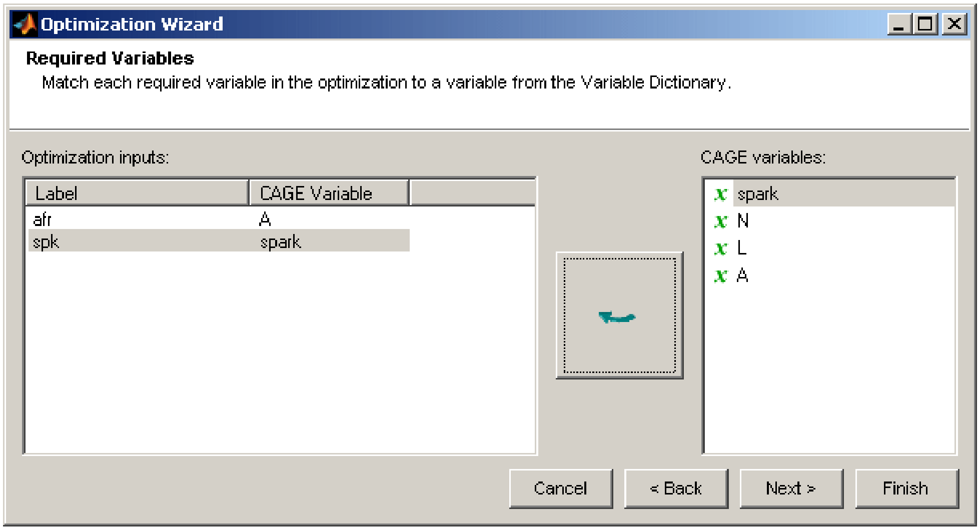
- The `tq` model was fitted to the Holliday engine data and exported from the Model Browser quick start tutorial (also used in the CAGE feature calibration tutorial). It can be found in `tutorial.exm` in the `mbctraining` directory. To view this model in your current session, click the **Models** button 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. Note you can specify fixed variables for optimizations either directly in the optimization view or import them from a data set.

2 Select **File > New > Optimization**.

The Optimization Wizard appears.

3 Select `Worked Example`, and click **Next**.

4 Associate each pair of inputs and variables, by clicking `spark` and `spk` in the left and right lists, and then click the Select button. Similarly associate `afr` with `A`. Click **Next**.



Optimization Wizard

Required Variables

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

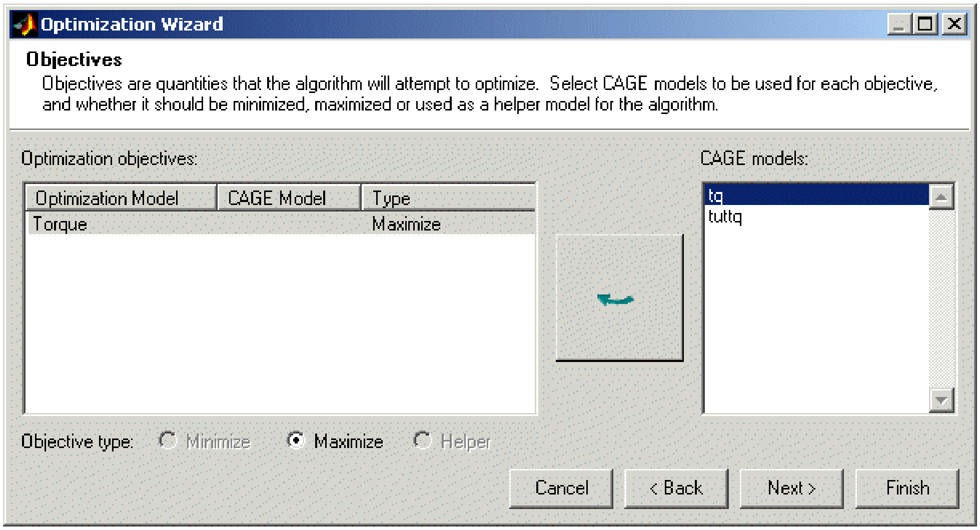
Optimization inputs:

Label	CAGE Variable
afr	A
spk	spark

CAGE variables:

- spark
- N
- L
- A

5 The next screen of the wizard automatically shows the `Torque` 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 **Finish**.



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

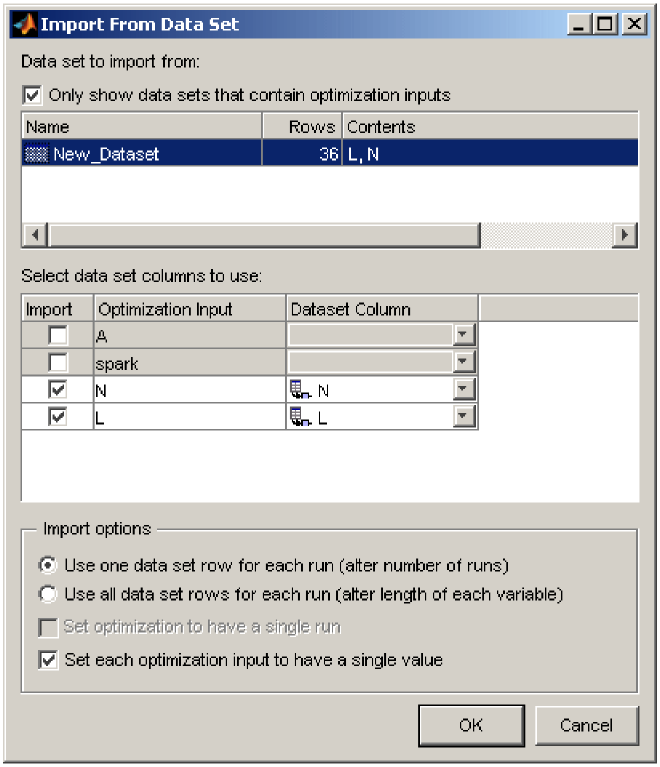
CAGE models:

tq
tuttq

Objective type: ○ Minimize ● Maximize ○ Helper

CAGE switches to the **Optimization** view and the new `Optimization` node appears in the tree.

6 If you ran the optimization now it would run at one point, the set point of all the variables. You use the free and fixed **Variable Values** panes to select operating points. You can edit points manually or import them. Select **Optimization > Import From Data Set**.



Import From Data Set

Data set to import from:

☑ Only show data sets that contain optimization inputs

Name	Rows	Contents
New_Dataset	36	L, N

Select data set columns to use:

Import	Optimization Input	Dataset Column
□	A	
□	spark	
☑	N	N
☑	L	L

Import options

● Use one data set row for each run (alter number of runs)

○ Use all data set rows for each run (alter length of each variable)

□ Set optimization to have a single run

☑ Set each optimization input to have a single value

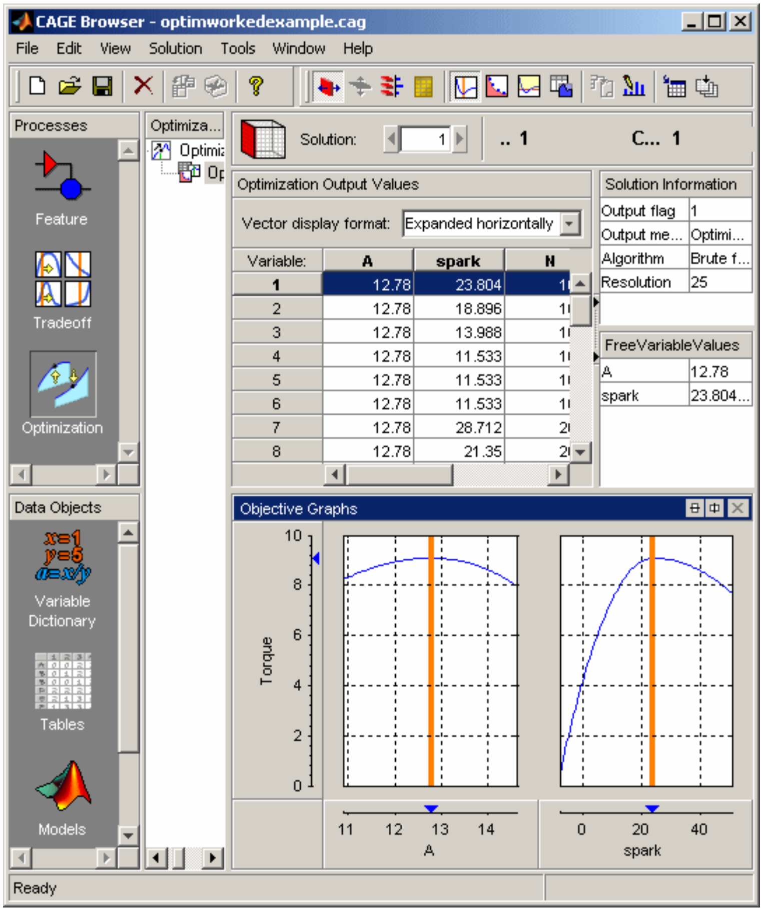
The project file contains a data set with N and L values, and these are automatically selected . Click **OK** to import.

Notice 36 rows appear in both fixed and free variable values panes, and operating point values have been imported into the N and L columns in the **Fixed Variable Values** pane. The initial values for A and spark for each point are the set points in the variable dictionary.

7 Click Run Optimization in the toolbar.

8 When the optimization completes, the view switches to the new `Optimization_Output` node.

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.



Vector display format: Expanded horizontally

Variable:	A	spark	N
1	12.78	23.804	1
2	12.78	18.896	1
3	12.78	13.988	1
4	12.78	11.533	1
5	12.78	11.533	1
6	12.78	11.533	1
7	12.78	28.712	2
8	12.78	21.35	2

Solution Information

Output flag	1
Output me...	Optimi...
Algorithm	Brute f...
Resolution	25

FreeVariableValues

A	12.78
spark	23.804...

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 12-48.

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` and can be imported to the fixed variable values pane in the Optimization view.

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

Process Overview

1. Start with your own algorithm. We will use `fminunc` from the Optimization Toolbox as 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; either change **Current Directory** in MATLAB to the new working folder, or add the folder to the path as follows:

   a. Select **File > Set Path**.

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

   c. Click **OK**.

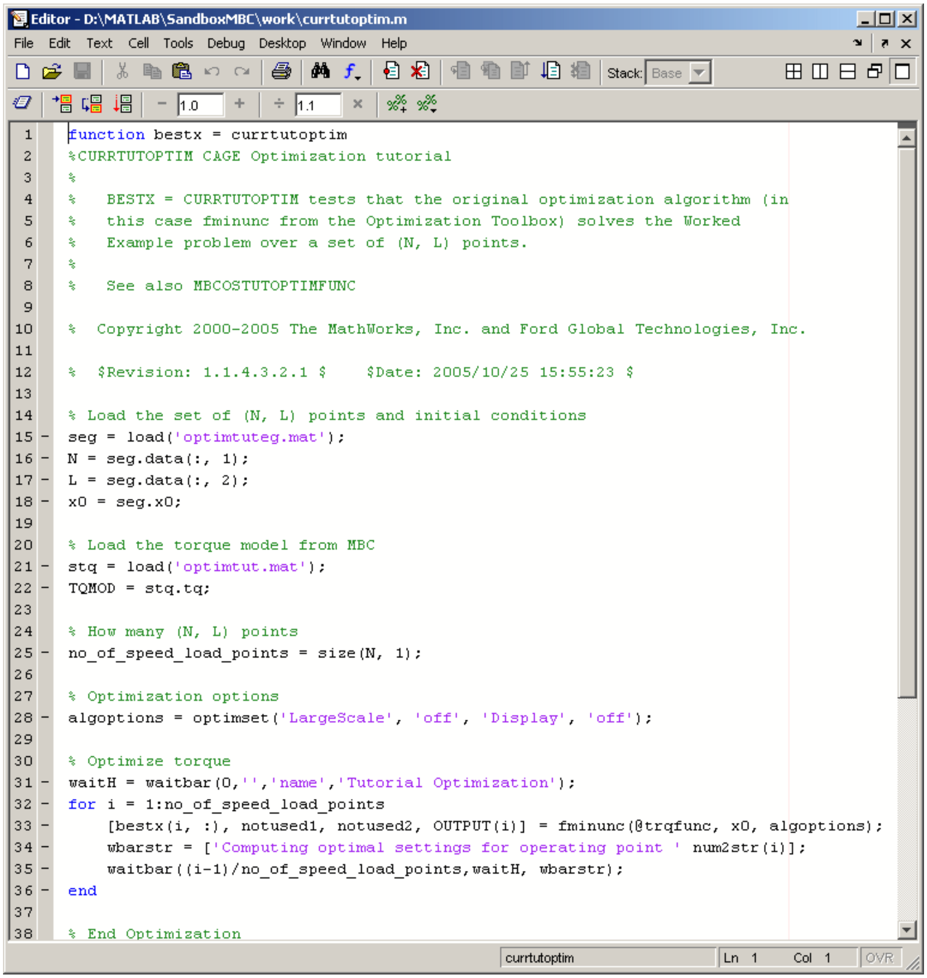
   d. Click **Save**.

   e. Click **Close**.

Step 1: Verify the Algorithm

`currtutoptim.m` is an example file to verify that `fminunc` solves the worked example problem. You can try this at the MATLAB command line.

1. To open the algorithm file in the Editor, either enter `open currtutoptim.m` at the command line, or if the **Current Directory** in MATLAB is your new working folder, select **Desktop > Current Directory**, then double-click `currtutoptim.m` . You should see the following code in the MATLAB editor.


Editor - D:\MATLAB\SandboxMBC\work\currtutoptim.m
File Edit Text Cell Tools Debug Desktop Window Help
Stack: Base
1.0
1.1
currtutoptim
Ln 1
Col 1
OVR

```
function bestx = currtutoptim
%CURRTUTOPTIM CAGE Optimization tutorial
%
%   BESTX = CURRTUTOPTIM tests that the original optimization algorithm (in
%   this case fminunc from the Optimization Toolbox) solves the Worked
%   Example problem over a set of (N, L) points.
%
%   See also MBCOSTUTOPTIMFUNC

%  Copyright 2000-2005 The MathWorks, Inc. and Ford Global Technologies, Inc.

%  $Revision: 1.1.4.3.2.1 $    $Date: 2005/10/25 15:55:23 $

% Load the set of (N, L) points and initial conditions
seg = load('optimtuteg.mat');
N = seg.data(:, 1);
L = seg.data(:, 2);
x0 = seg.x0;

% Load the torque model from MBC
stq = load('optimtut.mat');
TQMOD = stq.tq;

% How many (N, L) points
no_of_speed_load_points = size(N, 1);

% Optimization options
algoptions = optimset('LargeScale', 'off', 'Display', 'off');

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

% End Optimization
```

2 To verify that `fminunc` solves the worked example problem, type the following command at the MATLAB prompt:

```
bestX = currtutoptim
```

After the progress messages complete the workspace output should resemble the following:

```
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.

`fminunc` is the example optimization algorithm that you want to transfer to CAGE for use in the optimization GUI.

This tutorial shows how to make `fminunc` 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.

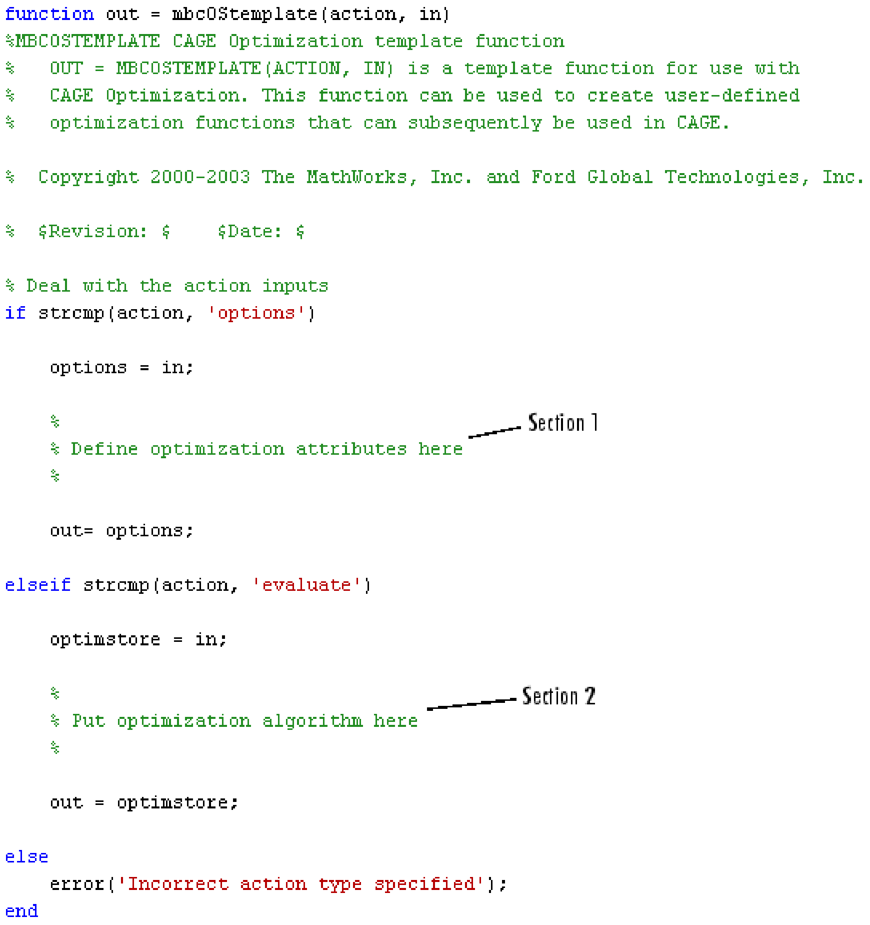
The first section defines the following attributes of the optimization:

- 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 (you can match in CAGE, so labels do not have to be exact in the optimization function)
- Number of constraints
- Labels for constraints, so the user can match models in CAGE to the required models in your algorithm constraints
- Number of data sets
- Labels for data sets, so the user can match data sets in CAGE to the required variable data for your algorithm
- Any other parameters required by the optimization algorithm

The second section contains the optimization algorithm.

Open `mbcOStemplate.m`.

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
```

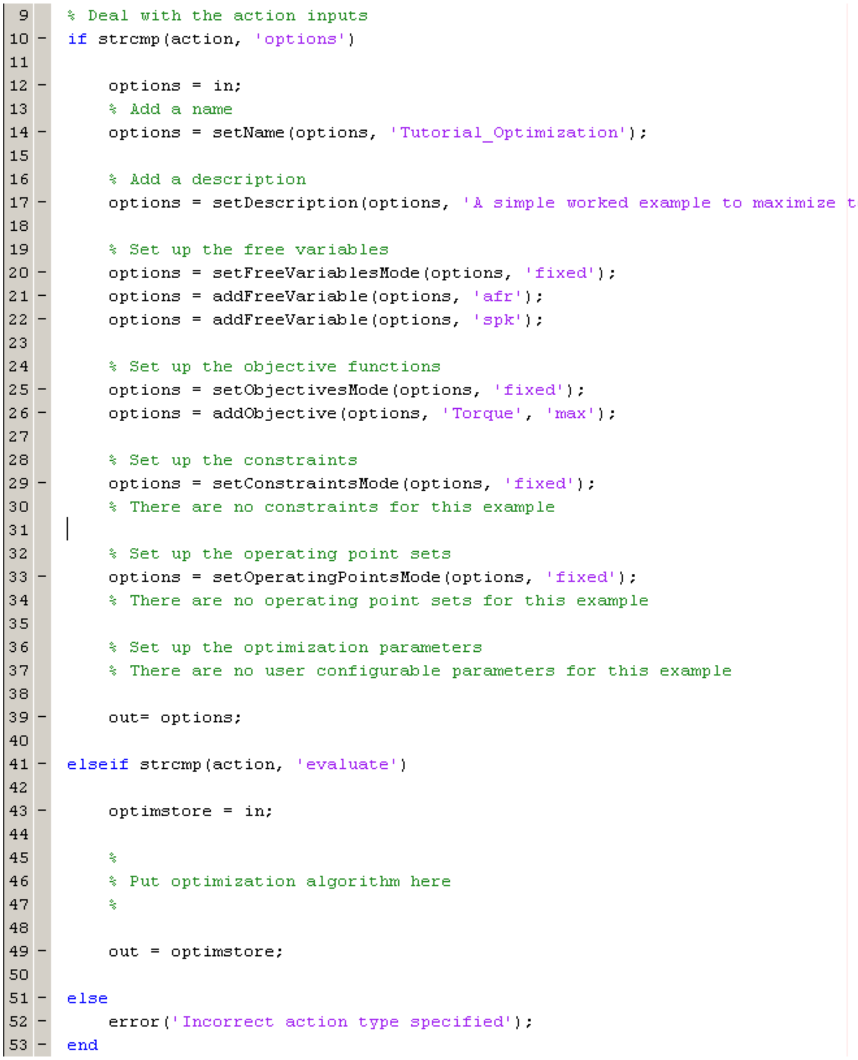
mbcOStemplate.m is an empty CAGE optimization function. The two (currently empty) sections of the function are labeled above. Note that this M-file can be used as a template for any optimization function that you write.

Step 3: Define the Optimization Options

The next step is to define the attributes of your optimization (in Section 1 of the template).

Open `mbcOStutoptimfunc_s1.m`. 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');
    % There are no operating point sets for this example

    % Set up the optimization parameters
    % There are no user configurable parameters for this example

    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, helper data sets, and optimization parameters for the optimization.

For detailed information on all the available functions, see "Optimization Function Reference" in the CAGE documentation. 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 Helper Data Sets	Cannot be changed by the user in the GUI (the mode has been set to `'fixed'`). There are no helper data sets for this example.	`Set up the operating point sets -- setOperatingPointsMode`
Optimization Parameters	There are no additional optional parameters	`Set up the optimization parameters`

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)
```

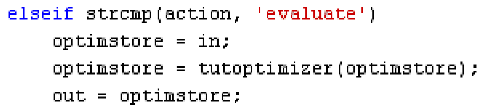
This is how your optimization function receives the `cgoptimoptions` object. Note that your optimization function *must* support this interface.

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 code that calls the algorithm, 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`.

   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 call to the 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

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

Original code:

`x0` passed in from the MATLAB workspace.

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 to retrieve `x0`.

Action 2

Perform the optimization (in Section 2 of the template).

Original code (from `currtutoptim`):

```
     [bestx(i, :), notused1, notused2, OUTPUT(i)] = fminunc(trqfunc,
   x0, algoptions);
```

which calls the following code to evaluate the cost function:

```
     function tq = trqfunc(x)

          % Evaluate torque. Note x = [SPK, AFR]
          tq = EvalModel(TQMOD, [x(1), N(i), L(i), x(2)]);

          % Maximising torque, so need to return -tq
          tq = -tq;

      end
```

Modified code:

```
  [bestx, unused, exitFlag, OUTPUT] = fminunc(@trqfunc_new,
   x0, algoptions);
```

which calls the following code to evaluate the cost function:

```
     function y = trqfunc_new(x)
          % Evaluate the torque objective function
          y = -evaluate(optimstore, x);
```

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)

Action 3

Retrieve output data.

Original code:

Optimal free variable settings are returned to the workspace through the variable `bestX` in `currtutoptim`.

Modified code:

```
% Write results to the optimstore
optimstore = setFreeVariables(optimstore, bestx);

% Set termination message - copied from help for FMINUNC
switch exitFlag
    case 1
        termMsg = 'Magnitude of gradient smaller than the
             specified tolerance.';
    case 2
        termMsg = 'Change in X smaller than the specified
             tolerance.';    case 3
        termMsg = 'Change in the objective function value
             smaller than the specified tolerance
             (only occurs in the large-scale method).';
    case 0
        termMsg = 'Maximum number of function evaluations
             or iterations reached.';
    case -1
        termMsg = 'Algorithm terminated by the output function.'
    case -2
        termMsg = 'Line search cannot find an acceptable point
             along the current search direction
             (only occurs in the medium-scale method).';
    otherwise
```

```
        termMsg = 'Unexpected exit flag';
end

% Set all information in the optimstore, and leave ....
optimstore = setExitStatus(optimstore, exitFlag, termMsg);
optimstore = setOutput(optimstore, OUTPUT);
```

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 three functions you should invoke on the optimstore object to return optimization results to CAGE:

- setFreeVariables — Returns the optimal free variable values to CAGE
- setExitStatus — Returns an integer that indicates whether the algorithm terminated successfully or not (positive is successful). This sets the termination message.
- setOutput — 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.

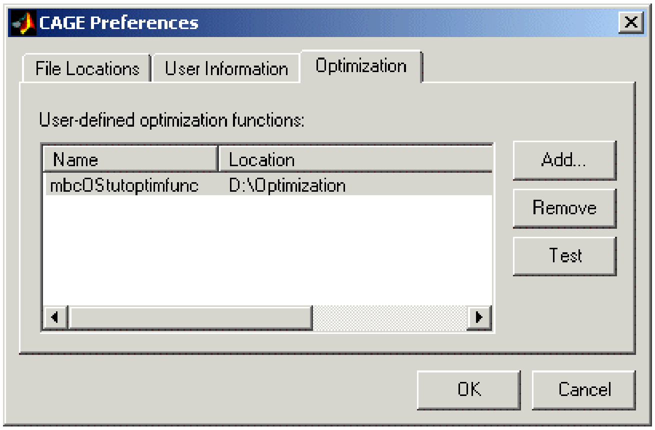
1. In CAGE, select **File > Preferences**.

   The CAGE Preferences dialog appears.

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

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

This registers the optimization function with CAGE.



User-defined optimization functions:

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.

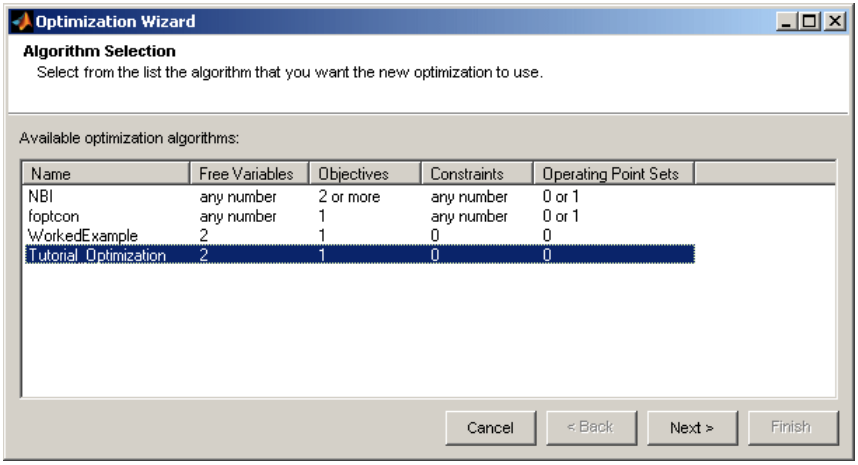
5 Click **OK** to leave the CAGE Preferences dialog. If the optimization function tested successfully, it is registered as an optimization function that can be used in CAGE, and appears in the Optimization Wizard.

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 1. 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 > New > Optimization**.

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



Optimization Wizard

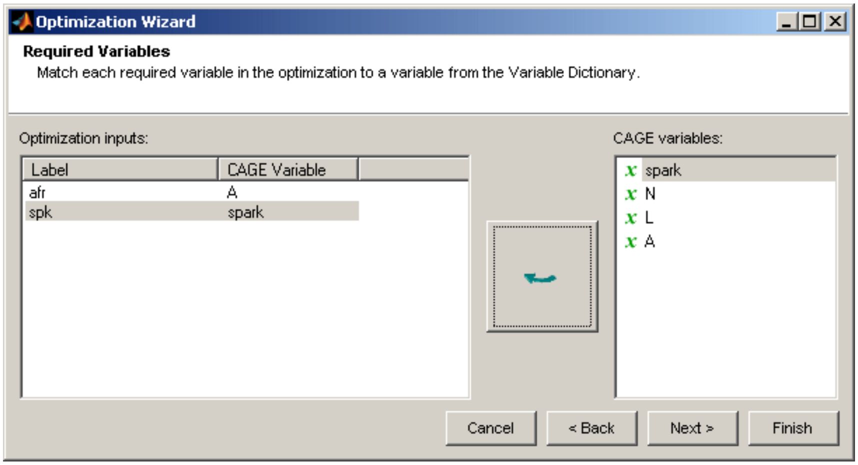
Algorithm Selection

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	0
Tutorial_Optimization	2	1	0	0

3 Match the variables as shown.



Optimization Wizard

Required Variables

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

Optimization inputs:

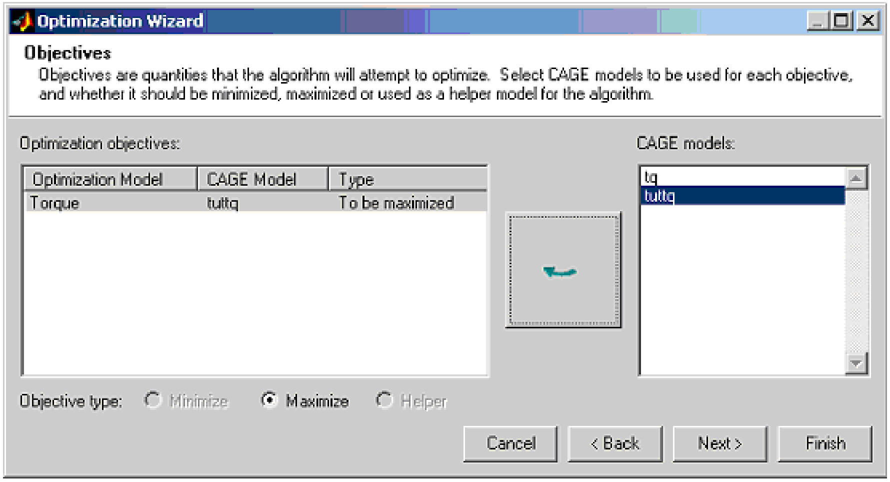
Label	CAGE Variable
afr	A
spk	spark

CAGE variables:

- spark
- N
- L
- A

Click **Next**.

4 Match the `Torque` model to the `tuttq` CAGE model as shown.



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:

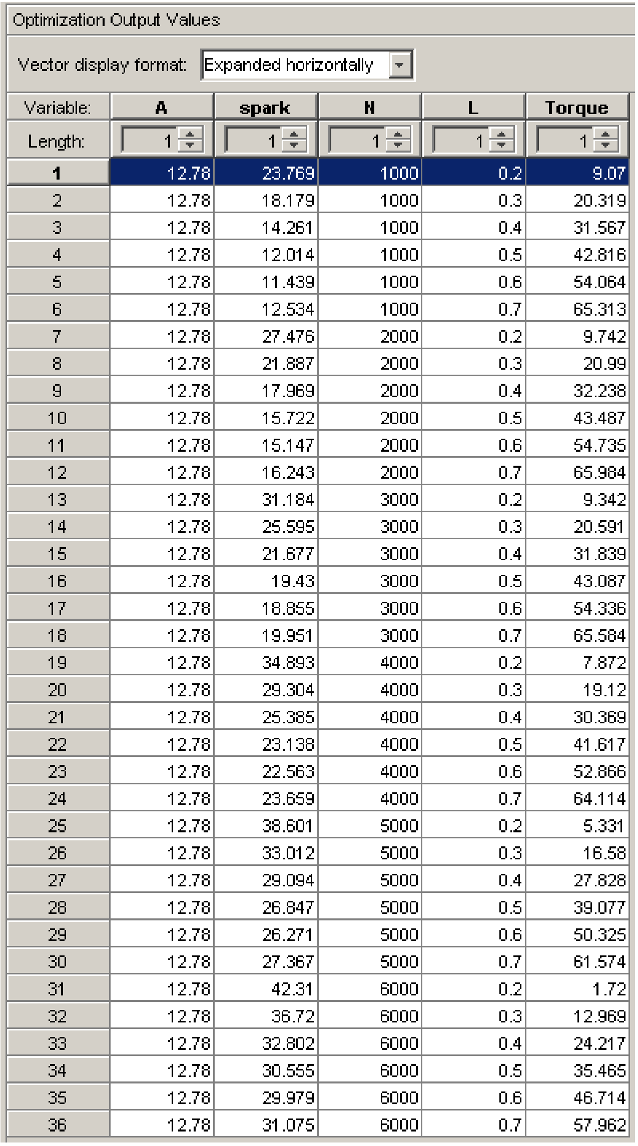
- tq
- tuttq

Objective type: ○ Minimize ● Maximize ○ Helper

Click **Finish**.

5 In the optimization view increase the **Number of runs** to 36, and copy and paste the fixed variable values from the previous worked example optimization.

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



Optimization Output Values

Vector display format: Expanded horizontally

Variable:	A	spark	N	L	Torque
Length:	1	1	1	1	1
1	12.78	23.769	1000	0.2	9.07
2	12.78	18.179	1000	0.3	20.319
3	12.78	14.261	1000	0.4	31.567
4	12.78	12.014	1000	0.5	42.816
5	12.78	11.439	1000	0.6	54.064
6	12.78	12.534	1000	0.7	65.313
7	12.78	27.476	2000	0.2	9.742
8	12.78	21.887	2000	0.3	20.99
9	12.78	17.969	2000	0.4	32.238
10	12.78	15.722	2000	0.5	43.487
11	12.78	15.147	2000	0.6	54.735
12	12.78	16.243	2000	0.7	65.984
13	12.78	31.184	3000	0.2	9.342
14	12.78	25.595	3000	0.3	20.591
15	12.78	21.677	3000	0.4	31.839
16	12.78	19.43	3000	0.5	43.087
17	12.78	18.855	3000	0.6	54.336
18	12.78	19.951	3000	0.7	65.584
19	12.78	34.893	4000	0.2	7.872
20	12.78	29.304	4000	0.3	19.12
21	12.78	25.385	4000	0.4	30.369
22	12.78	23.138	4000	0.5	41.617
23	12.78	22.563	4000	0.6	52.866
24	12.78	23.659	4000	0.7	64.114
25	12.78	38.601	5000	0.2	5.331
26	12.78	33.012	5000	0.3	16.58
27	12.78	29.094	5000	0.4	27.828
28	12.78	26.847	5000	0.5	39.077
29	12.78	26.271	5000	0.6	50.325
30	12.78	27.367	5000	0.7	61.574
31	12.78	42.31	6000	0.2	1.72
32	12.78	36.72	6000	0.3	12.969
33	12.78	32.802	6000	0.4	24.217
34	12.78	30.555	6000	0.5	35.465
35	12.78	29.979	6000	0.6	46.714
36	12.78	31.075	6000	0.7	57.962

Index