Model-Based Calibration Toolbox

Toolbox

For Use with MATLAB® and Simulink®

- Computation
- Visualization
- Programming
- Simulation

Model Browser User's Guide

Version 2

The MathWorks

How to Contact The MathWorks:

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

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

Contents

Tutorial: Design of Experiment

3

Tutorial: Data Editor

4

Designs

5

iii

Data

6

Modeling

7

Technical Documents

8

Radial Basis Functions

9

<p align="right">10</p>

Model Browser
Case Studies

Index

Getting Started

The following sections introduce the Model-Based Calibration toolbox.

What Is the Model-Based Calibration Toolbox? (p. 1-3)	Introducing the Model-Based Calibration Toolbox.
About the Model Browser (p. 1-3)	The Model Browser part of the toolbox is a powerful tool for design of experiment and statistical modeling.
About CAGE (p. 1-3)	The CAGE part of the toolbox can import statistical models created in the Model Browser to calibrate and optimize tables.
How to Use This Manual (p. 1-5)	For information on learning and using the Model Browser, this section contains an overview of this manual with links to the tutorials and reference sections.
System Requirements (p. 1-7)	Hardware and operating system requirements, and required and optional products from The MathWorks.

Starting the Model Browser

To start the application, type

```
mbcmodel
```

at the MATLAB® command prompt.

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. There are two main user interfaces:

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

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.

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.

How to Use This Manual

This manual is the Model Browser User's Guide.

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

Learning the Model Browser

For new users there are three tutorial chapters to guide you through using the Model Browser tools:

- Chapter 2, "Tutorial: Quickstart" provides a quick introduction to modeling with the toolbox. The tutorial describes how to set up and view a two-stage model using some engine data.

- Chapter 3, "Tutorial: Design of Experiment" covers the Design of Experiment tools 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 4, "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.

Using the Model Browser

- Chapter 5, "Designs" is a guide to constructing experimental designs using the Design Editor.

- Chapter 6, "Data" describes how to load, merge, filter, and view data using the Data Editor, including matching data to experimental designs.

- Chapter 7, "Modeling" is a complete guide to setting up, evaluating, and verifying models using the Model Browser. This includes all the functionality available in the different views of the Model Browser, descriptions and illustrations of model types, boundary modeling, and exporting models.

- Chapter 8, "Technical Documents" covers the modeling process and the mathematical basis of hierarchical models, including a guide to using the Stepwise window, the Box-Cox transformation dialog, the Design Evaluation tool, and user-defined and transient models.

- Chapter 9, "Radial Basis Functions" is a guide to all aspects of using radial basis functions in modeling, from setup to the mathematical basis.

System Requirements

This section lists the following:

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

Hardware Requirements

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

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

Minimum memory:

- 256 MB

Minimum disk space:

- 450 MB for the software and the documentation

Operating System Requirements

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

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

Required MathWorks Products

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

Tutorial: 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. You can see "Two-Stage Models for Engines" on page 8-40 in the Technical Documents for more statistical depth.

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.
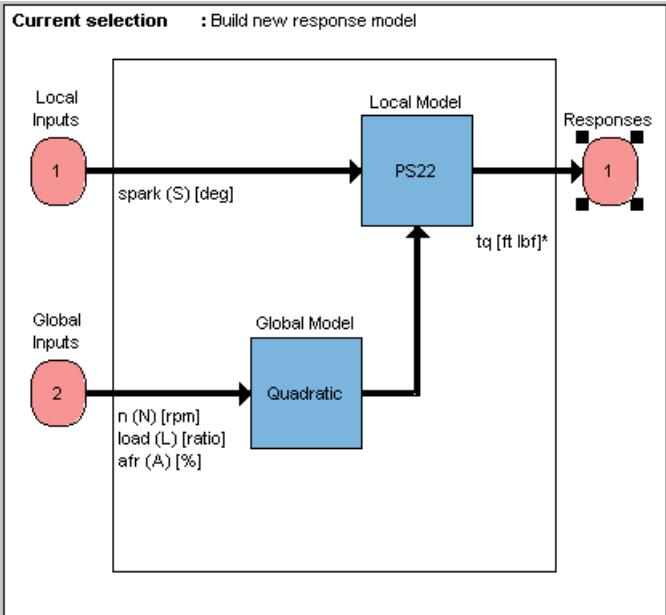
Starting the Toolbox

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.
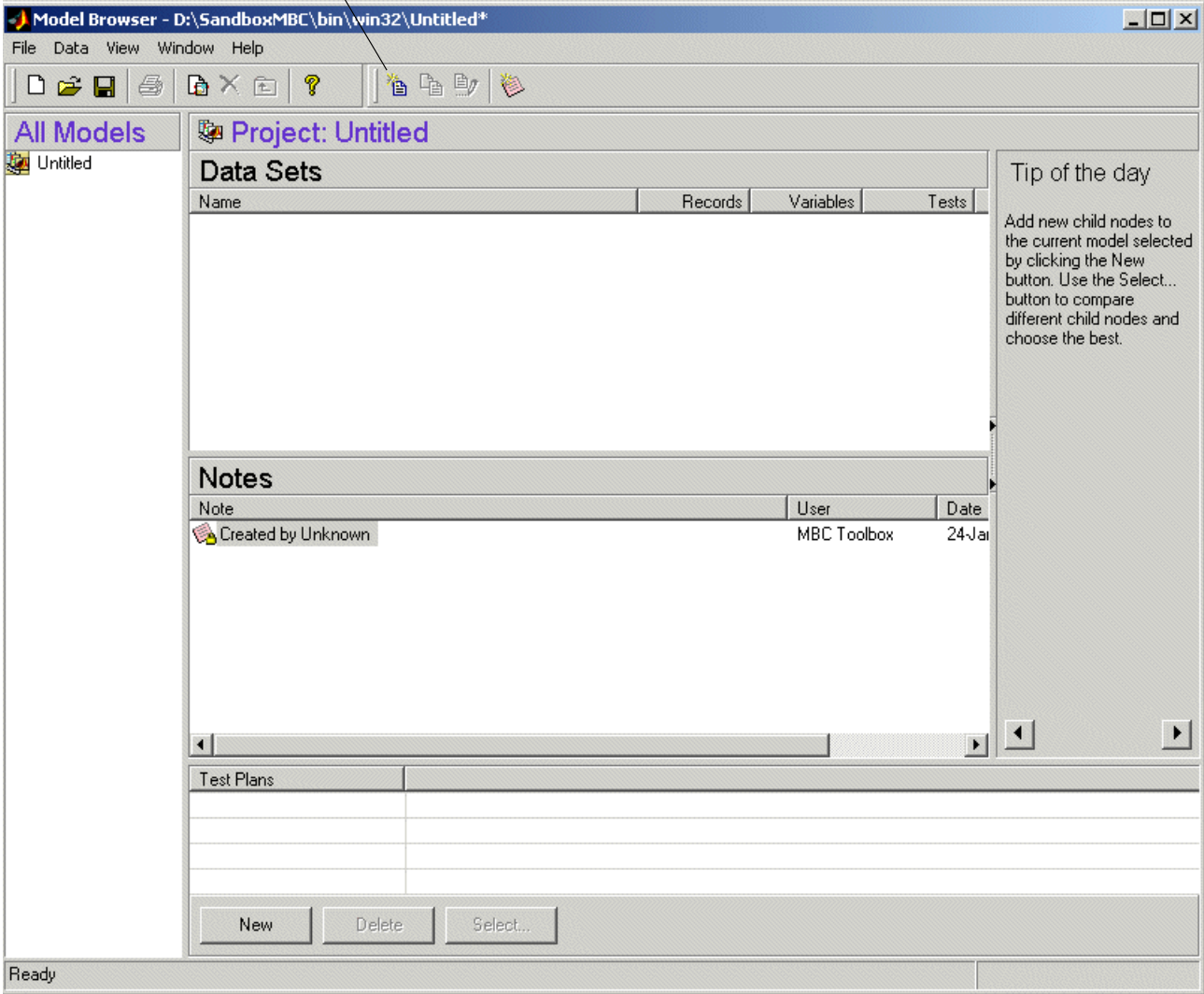
Load the example data file holliday.xls:

1 Click the 📄 button on the toolbar, or choose **Data –> New Data**.

This opens the Data Editor window.

2 Click the **Import File** icon on the toolbar (📂) or choose **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.
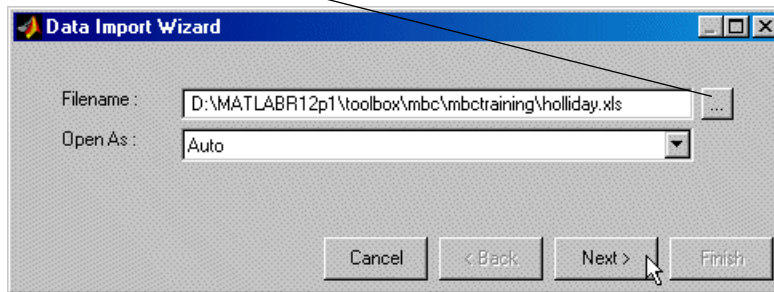
4 The file pathname appears in the Data Import Wizard. Click **Next**.

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" on page 6-1.

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. See the "Model Tree" on page 7-11 for more information.

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

  The **New Test Plan** dialog box appears.
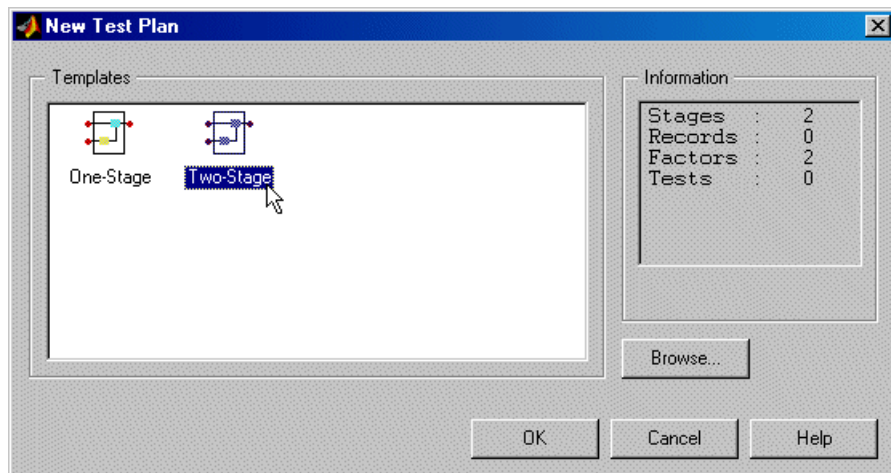
2 Click the two-stage test plan icon and click **OK**.

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 "Functions Implemented in the Block Diagram" on page 7-23.
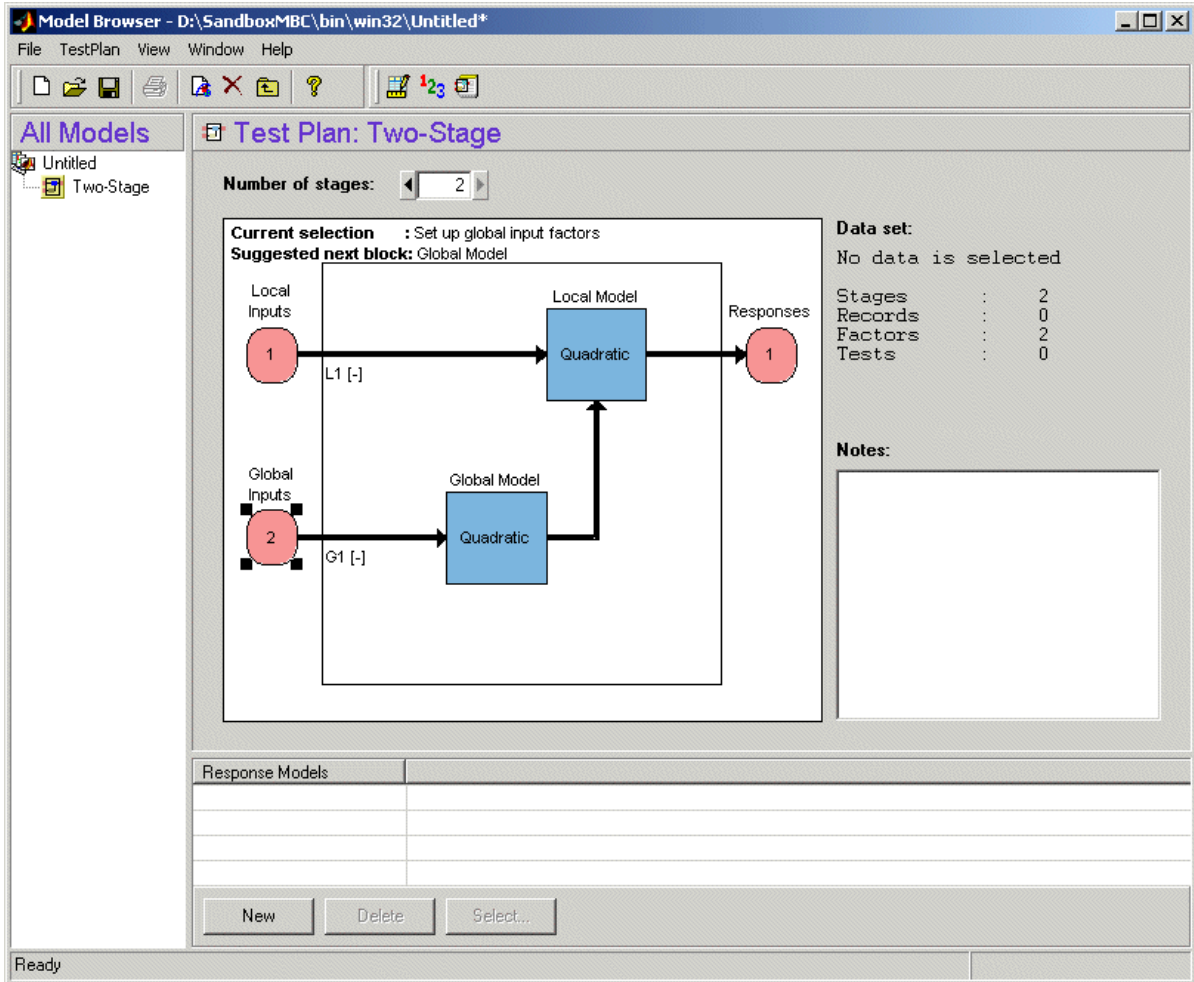
Setting Up the Local Model

Setting up the local model requires that you specify the model's inputs and type.
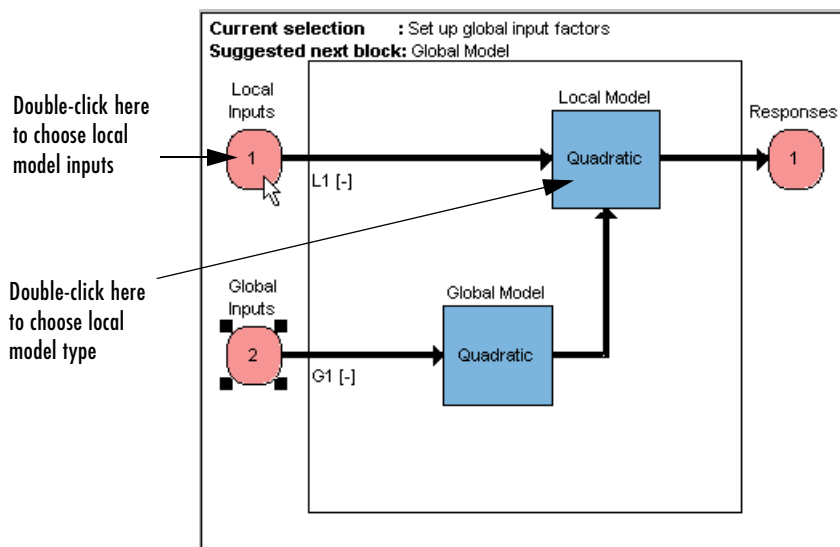
Specifying the Local Model Input

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.
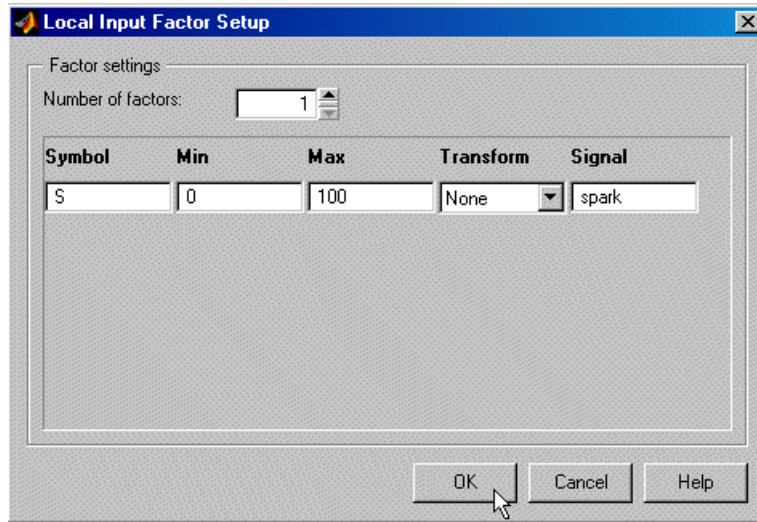
a Set **Symbol** to S.

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.
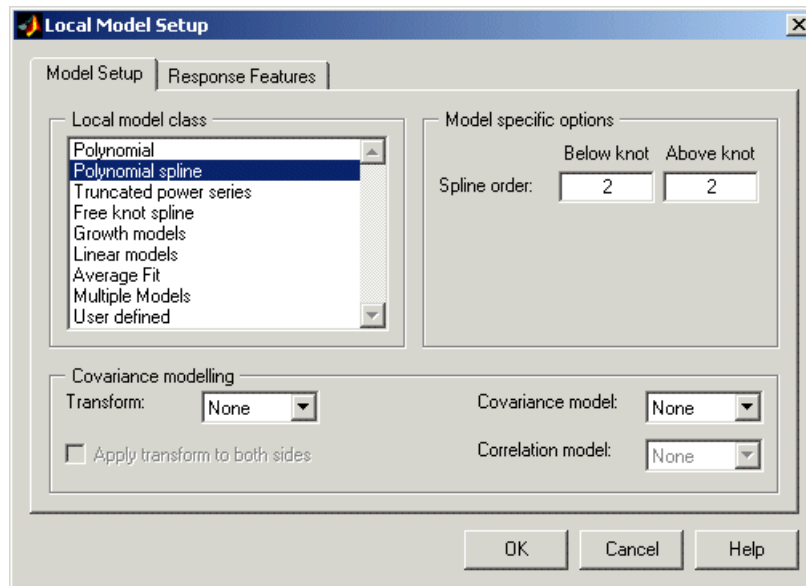
The **Local Model Setup** dialog box appears.

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

   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.
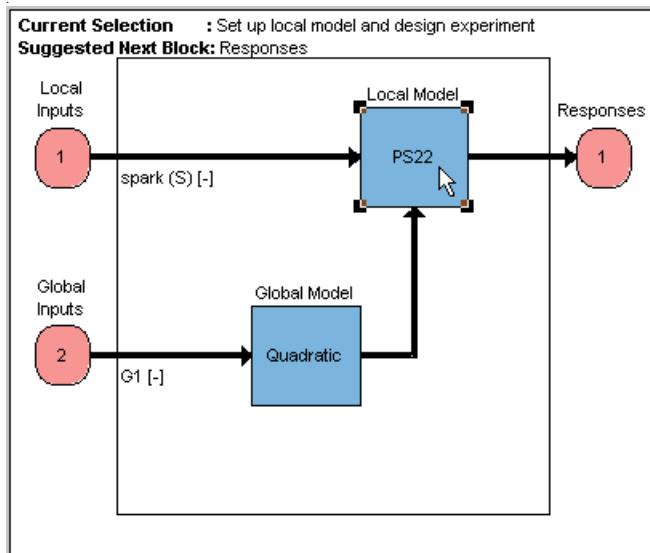
Setting Up the Global Model

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.

Double-click here to choose Global Model Inputs

Double-click here to choose Global Model Type
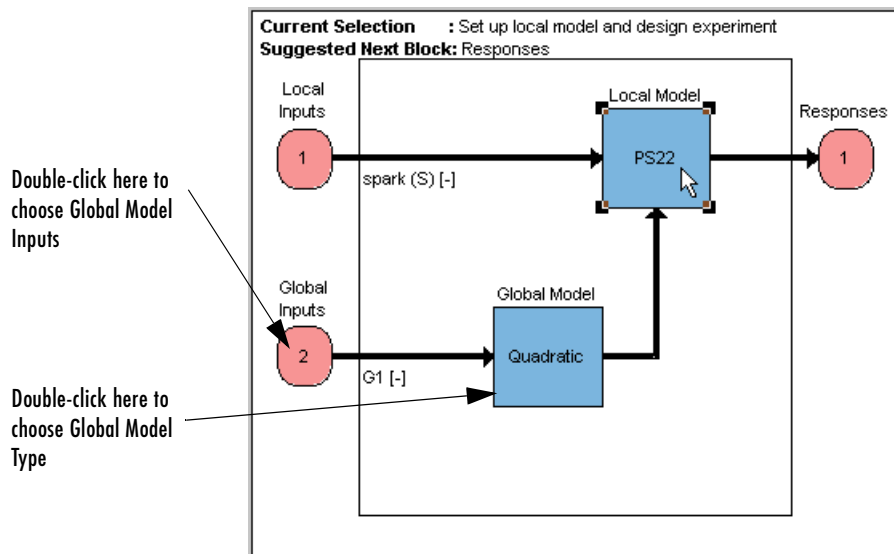
Specifying the Global Model Inputs

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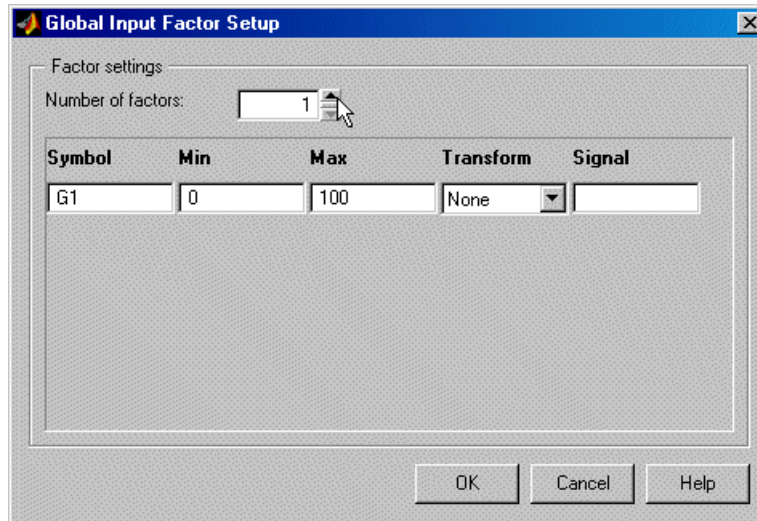
By default there is one input to the global model. Because this engine model has three input factors, you need to increase the input factors as follows:

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.

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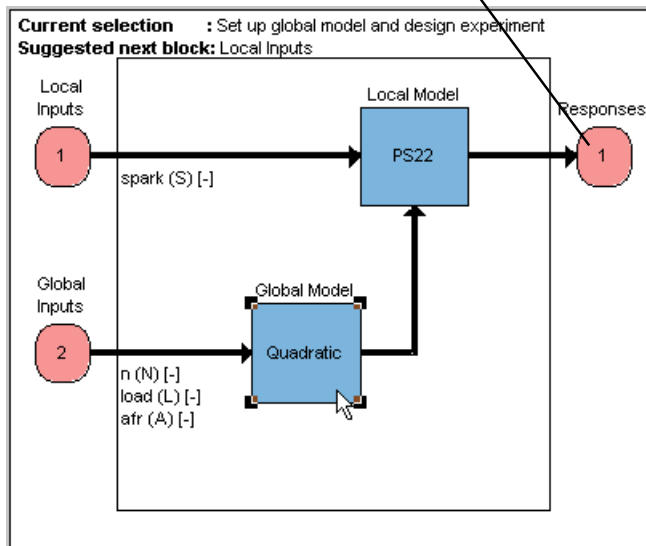
For the same result, you could also click the **Select Data** button in the toolbar of the model browser (or **TestPlan –> Select Data** menu item). Also, if you did not already load a data set at the project node, you can do it at this point using **TestPlan –> Load New Data**.

The **Data Wizard** dialog appears.

2-19
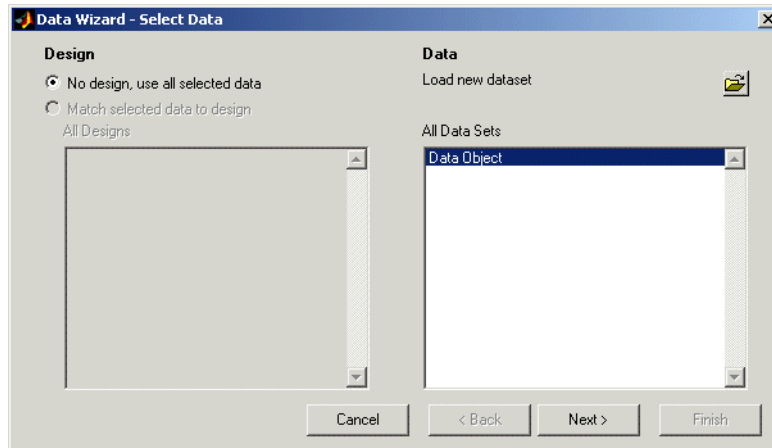
2 `Data Object` is already selected by default. Click **Next**.

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

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

Click here to match each pair of
input and signal
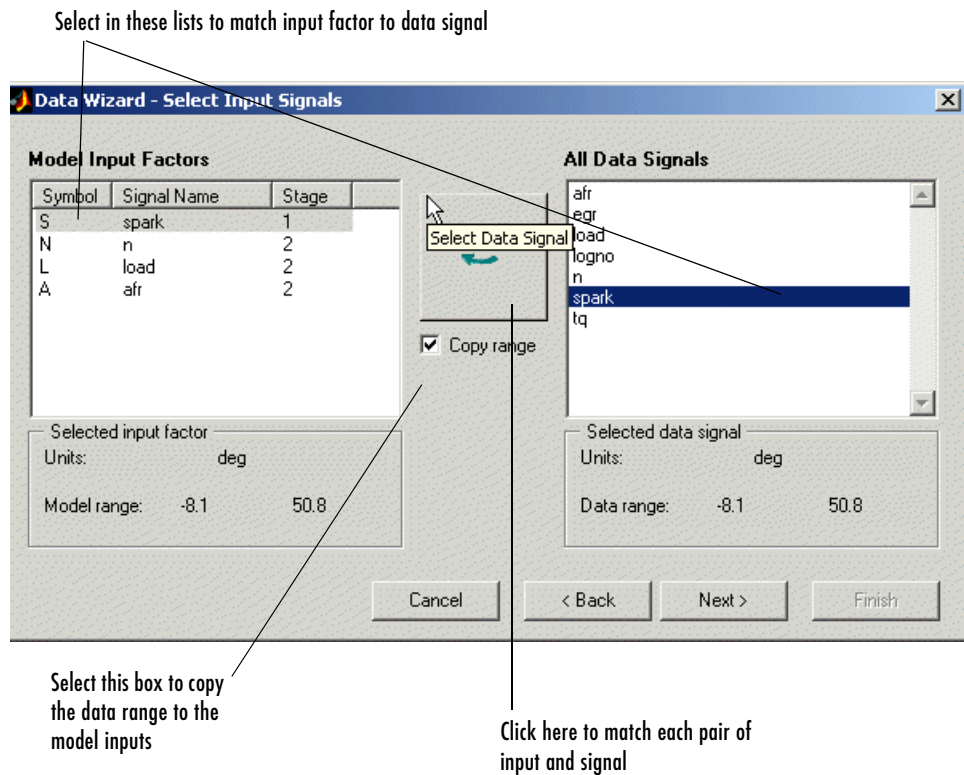
5 Click the large **Select Data Signal** button, as indicated above.

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.
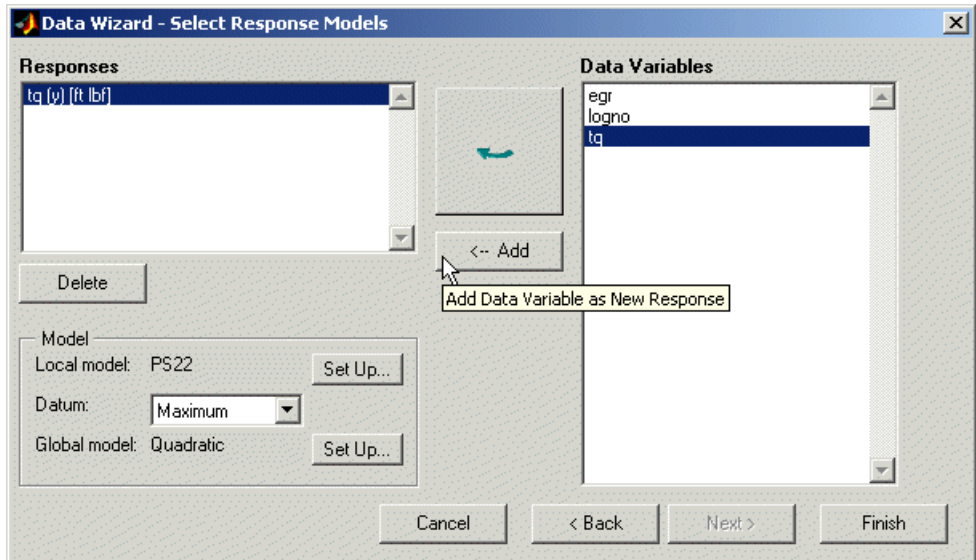
1 Select tq (torque) as the response.

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

3 Select Maximum under **Datum**.

   Only certain model types with a clearly defined maximum or minimum can support datum models. See "Datum Models" on page 7-79.

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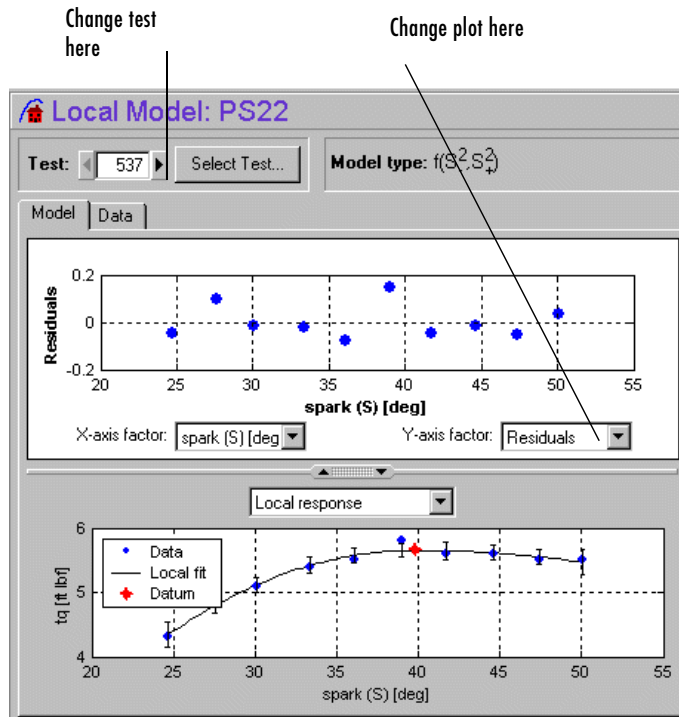
The lower plot shows the data being fitted by the model (blue dots) and the model itself (line). The red spot shows the position of the polyspline knot, at the datum (maximum) point.

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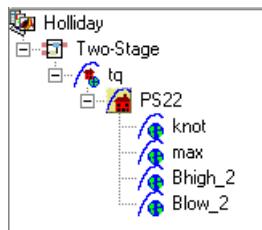
2 Select the first of the global models, knot.

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.
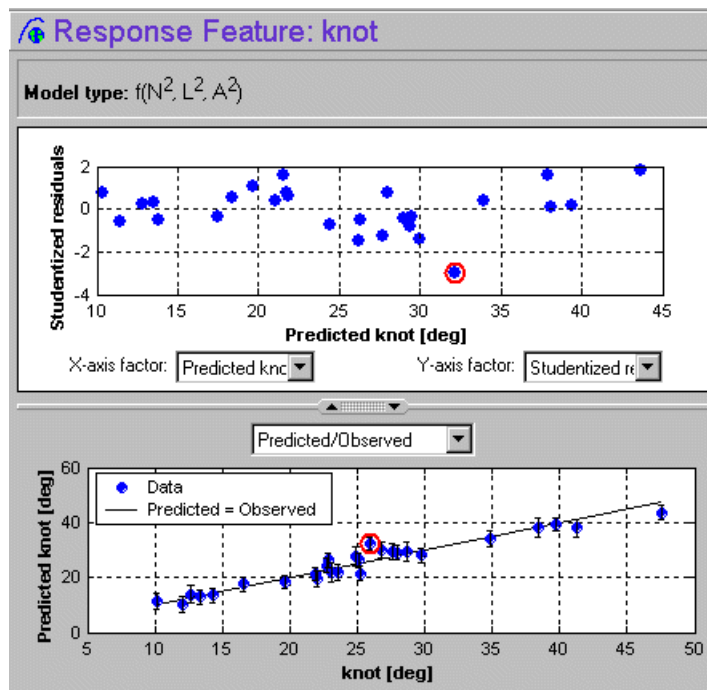
3 Select the response feature Bhigh_2. One outlier is marked. Points with an absolute studentized residual value of more than 3 are automatically suggested as outliers (but included in the model unless you take action). You can use the right-click menu to remove suggested outliers (or any others you 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 Obviously you should 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 2-31
- "Response Surface View" on page 2-32

For more detailed help on all the views available in the **Model Selection** window, see "Selecting Models" on page 7-111.

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.
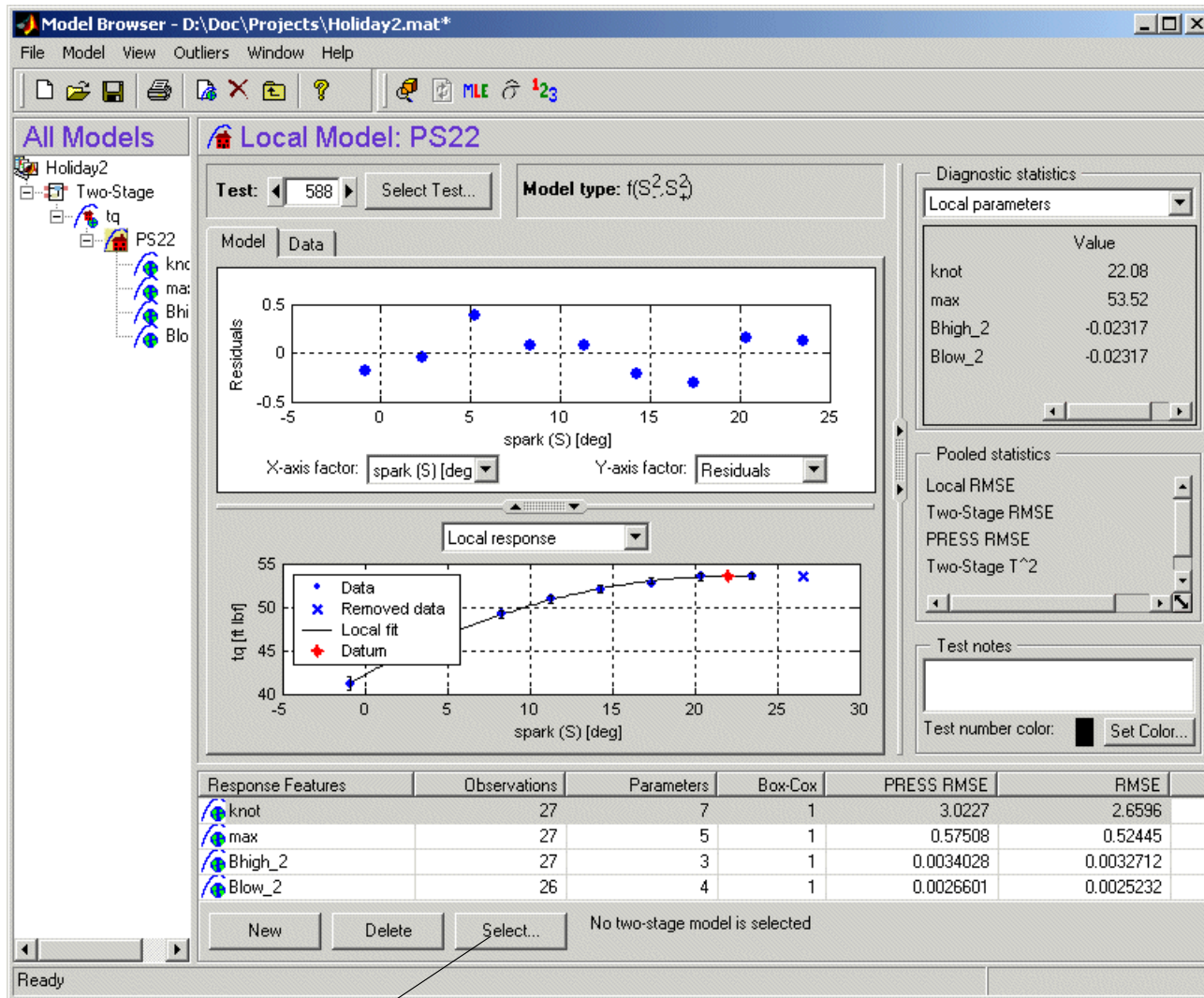
Open Model Selection here

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. There are a number of icons in the toolbar

that 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
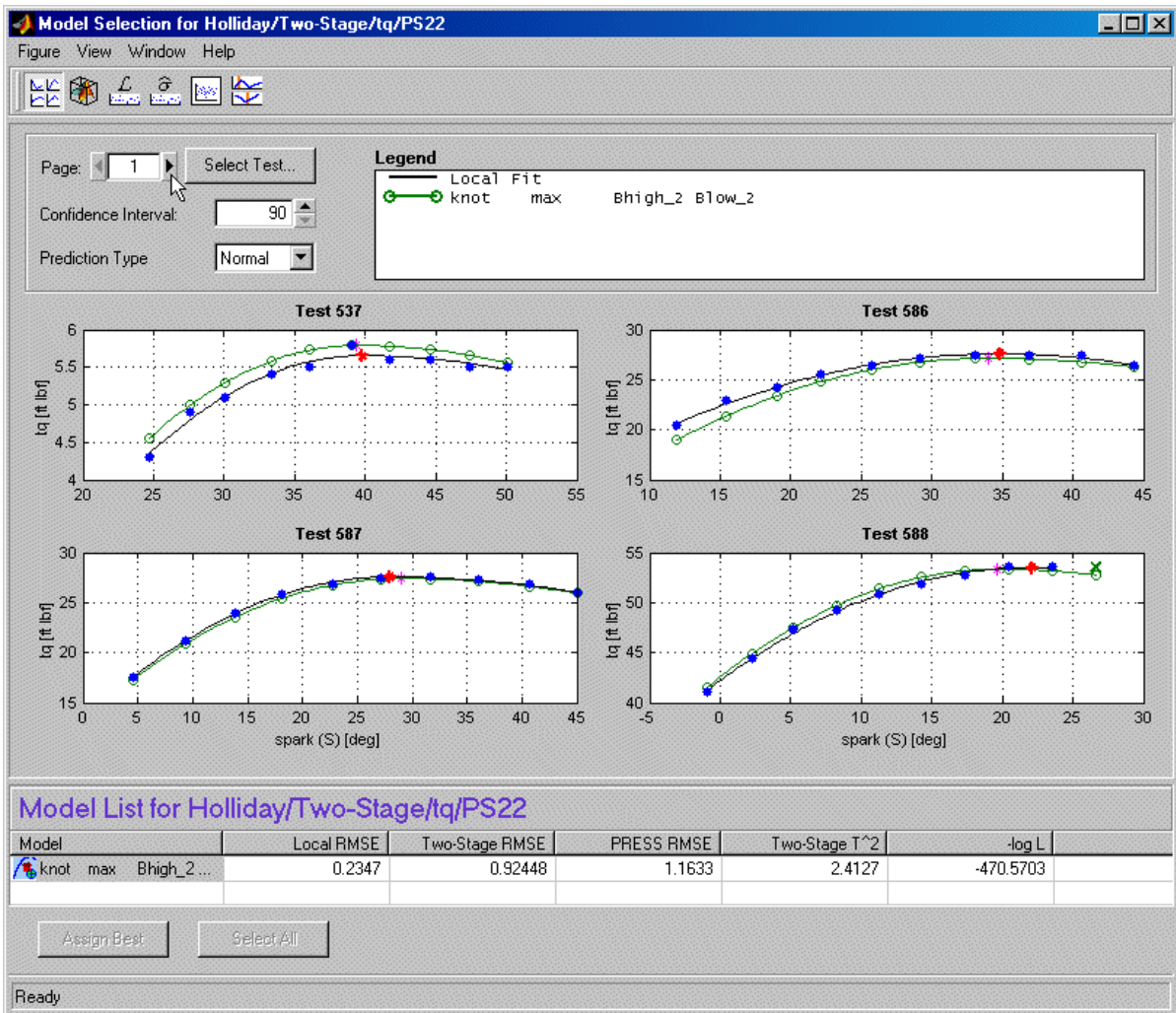
Scroll though the tests using the left/right arrows or the **Select Test** button at the top left. The plots show the fit of the two-stage model for each test (green open circles and line), compared with the fit of the local model (black line) and the data (blue dots). You can left-click (and hold) to 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.
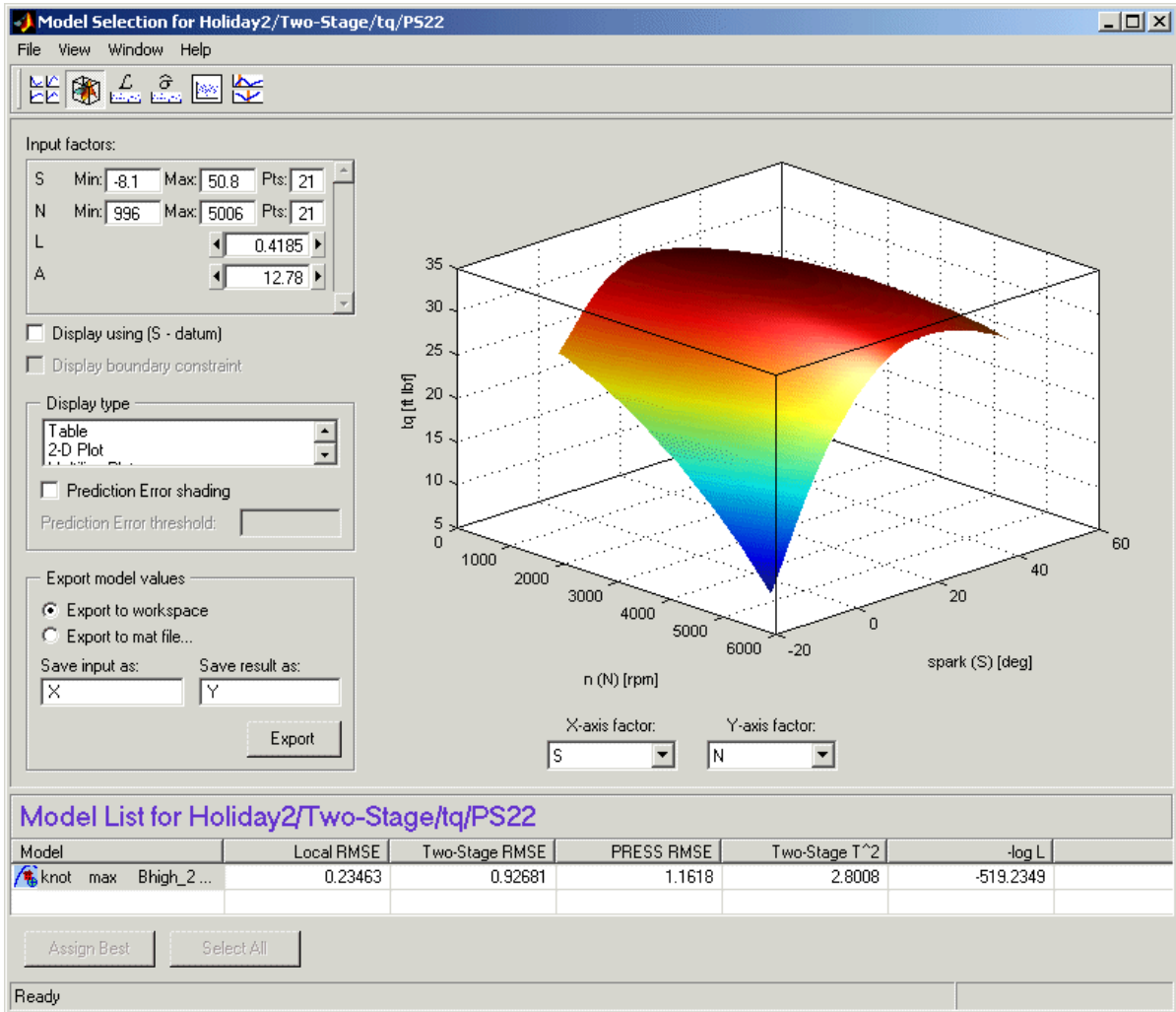
3 Dismiss the **Model Selection** pane, and accept the best model by clicking **Yes** in the **Model Selection** dialog (it is the only two-stage model so far).

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.
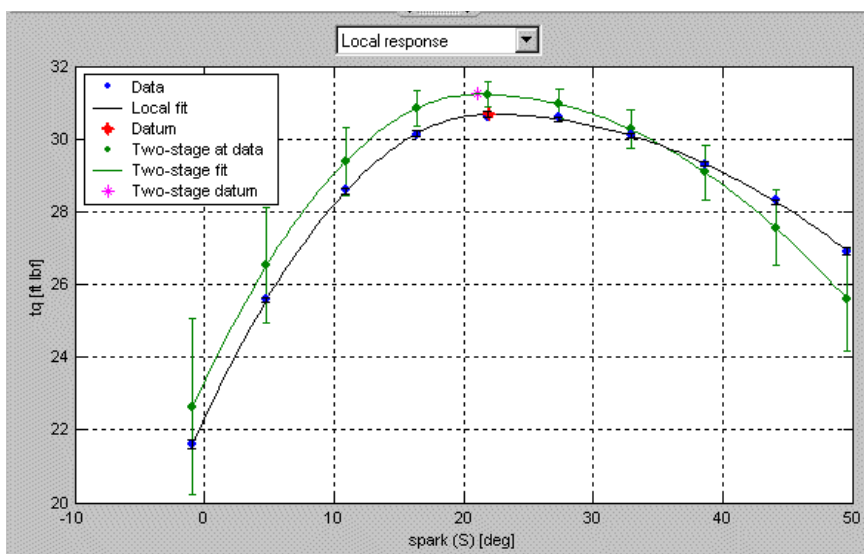
Notice that the local model icon has changed (from the local 🏠 icon showing a house, to a two-stage icon 🏠🌐 showing a house and a globe) to indicate that a two-stage model has been calcluated.

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.
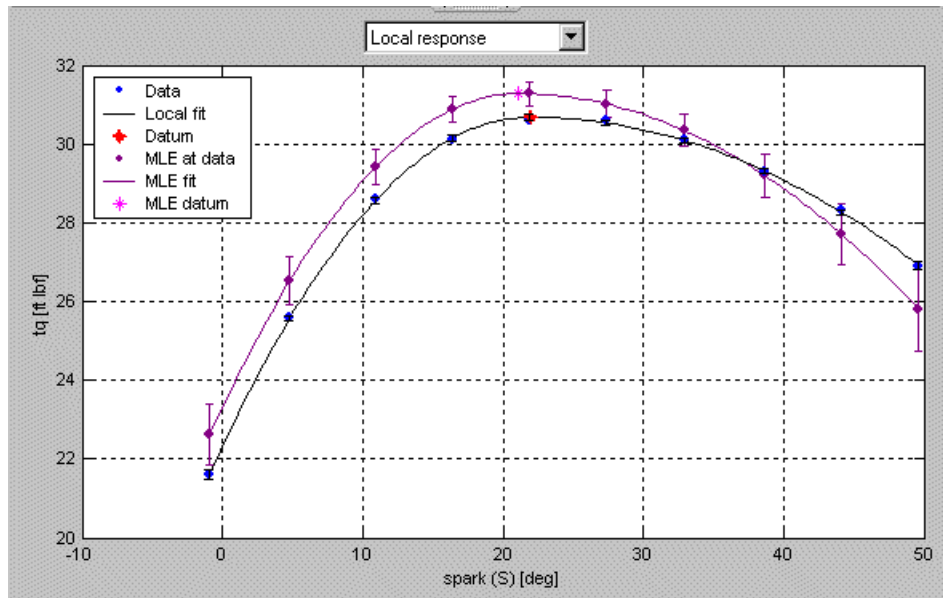
5 Click the **Select** button. This takes you to the **Model Selection** window.

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.
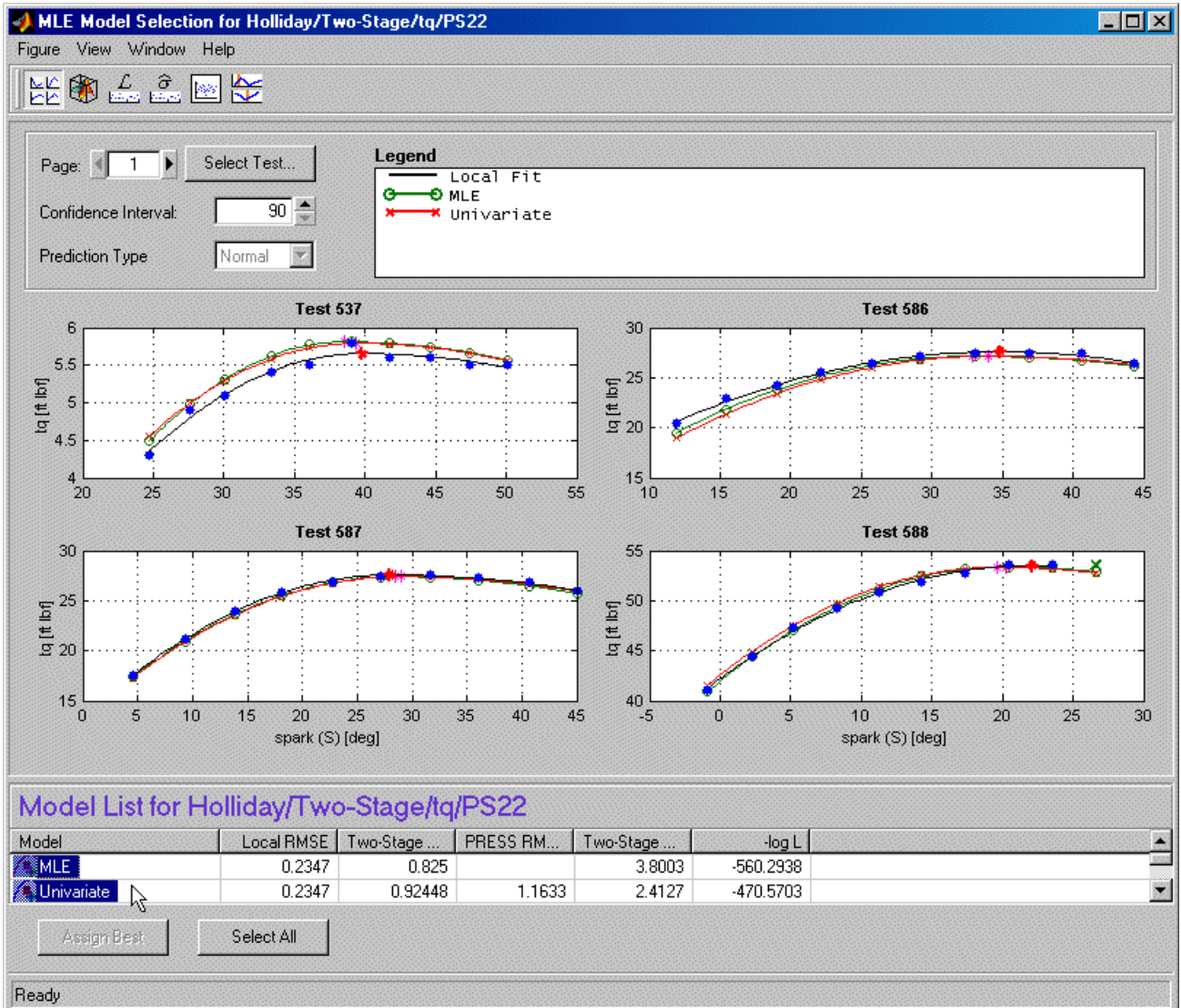
7 Close the **Model Selection** window. Click **Yes** to accept the MLE model as the best.

Response Node

Click the Response node (`tq`) in the Model Browser tree.
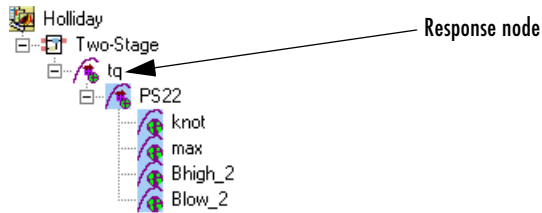
Now at the **Response node** in the Model Browser tree (tq), which was previously blank, you see this:
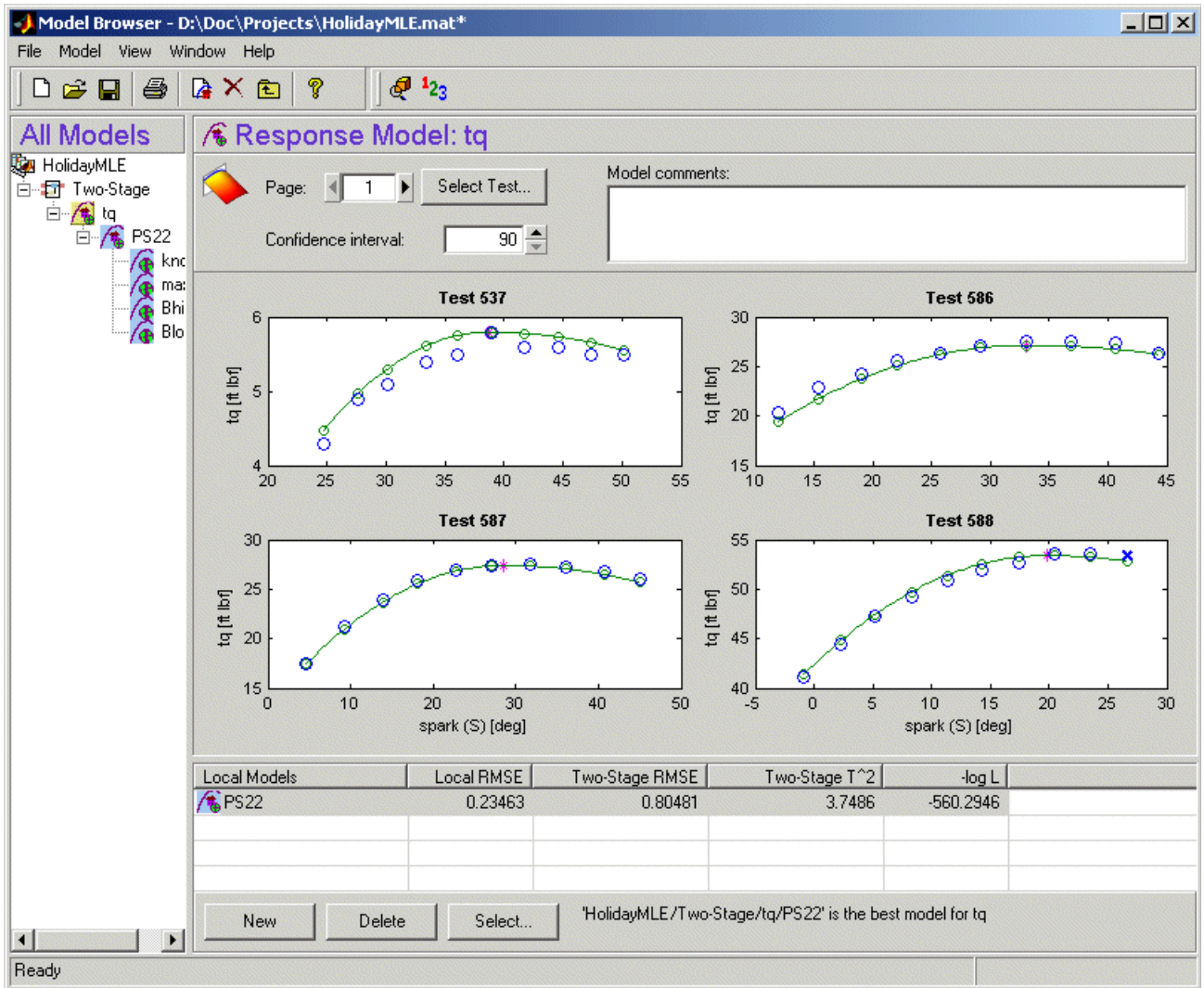
This shows you the fit of the two-stage model to the data. You can scroll through the tests, using the arrows at top left, to view the two-stage MLE model (in green) against the data (in blue) for each test.

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

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.
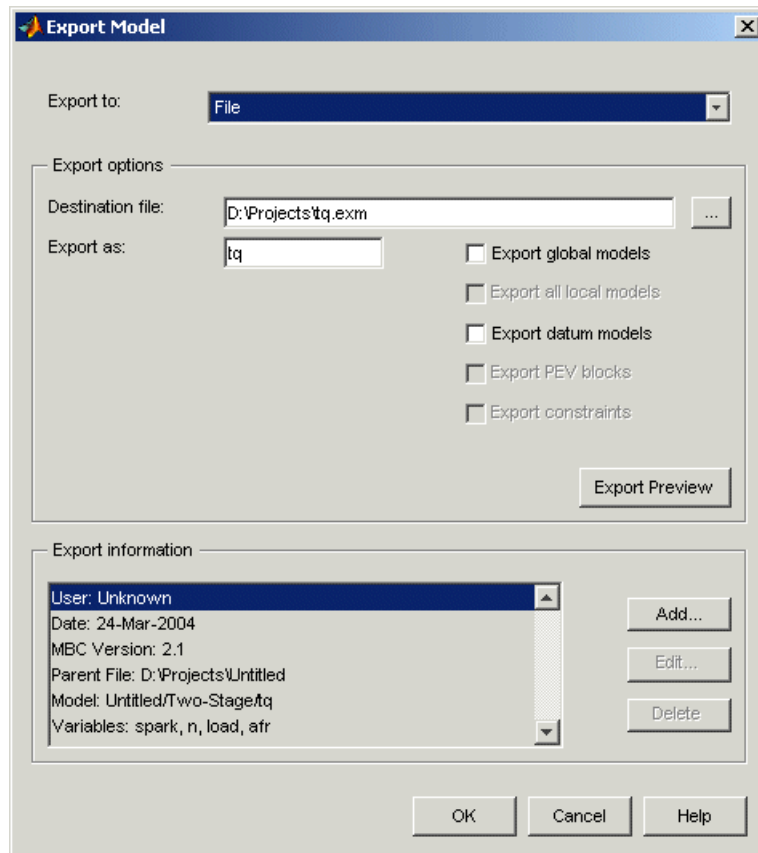
3 Choose File from the **Export to** pop-up menu. This saves the work as a file for use within the Model-Based Calibration (MBC) Toolbox, for instance, to create calibrations.

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).
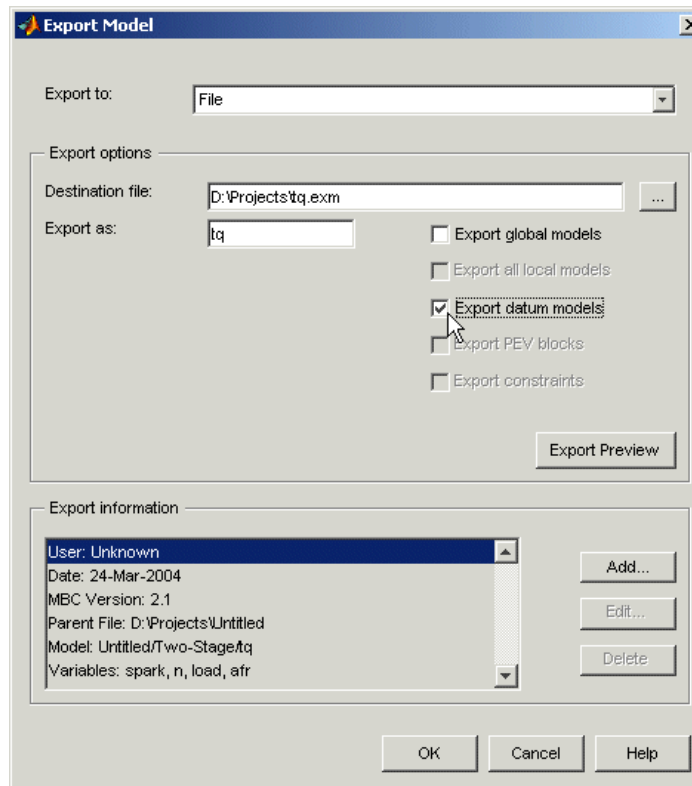
6 Click **OK** to export the models.

Creating Multiple Models To Compare

Once you have fitted and examined a single model, you will normally want to create more models in order 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 in order 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.
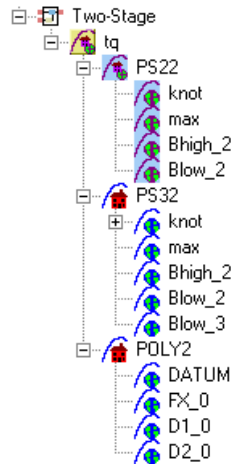
You can select the alternative local models in turn and compare their statistics. For an example follow these steps:

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.
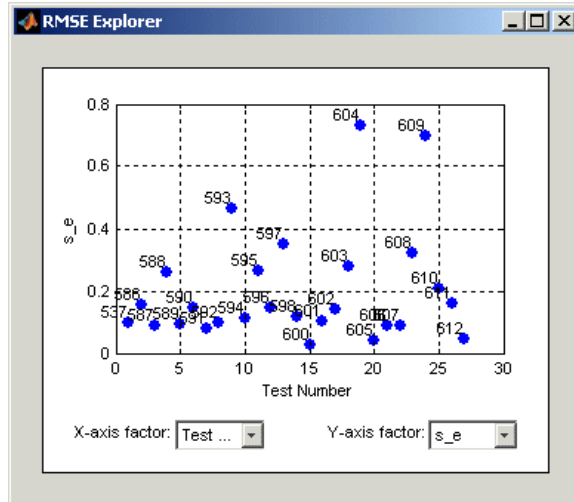
a Right-click to toggle test number display to help you identify problem tests.

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 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 in order to track a value such as maximum brake torque (MBT) minus ten 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.
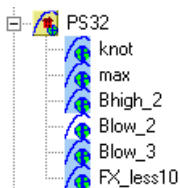
6 Select the tq response node to see a comparison of the statistics of both two-stage models (your original PS22 and the new PS32).

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. To calculate the two-stage model for POLY2:

7 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

New	Delete	Select...	'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 very 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 and how RMSE and PRESS can help you select good models, see "Model Selection Guide" on page 7-113. 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 only have 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" on page 8-25 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 principal 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.
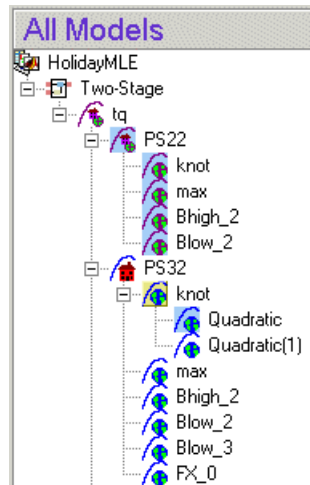
3 Select one of the new Quadratic nodes, then select the menu item **Model –> Set Up**.

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.
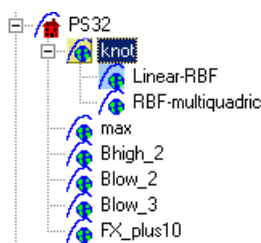
8 Add two more child nodes to the knot global model (select knot, then click **New** twice).

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.
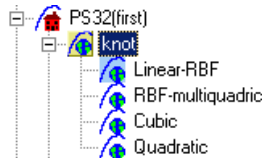
11 Select the knot model node, then select **Model –> Make Template**. Browse to a suitable work directory and enter the name `Mytemplate`. Click **OK**.

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.
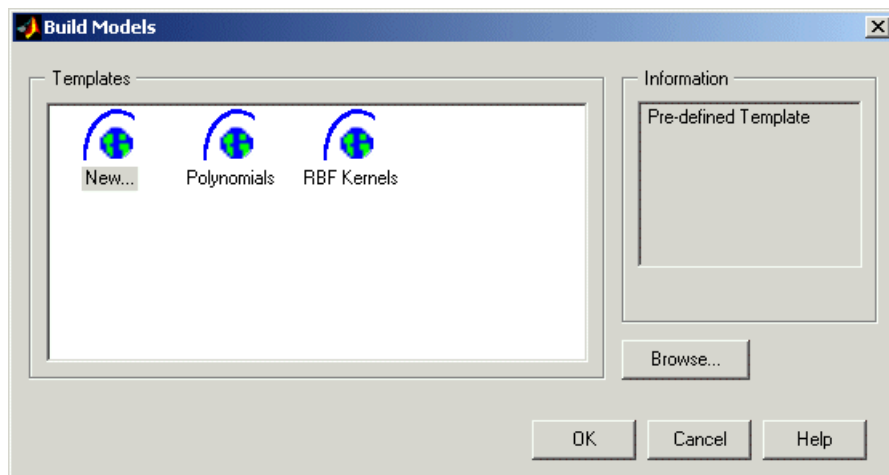
The **Build Models** dialog appears. Here you can choose a template for the type of models you want to build. There are predefined templates for polynomials and RBF kernels. You can also save templates of whatever models you choose by selecting the New template or using the **Model –> Make Template** 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.
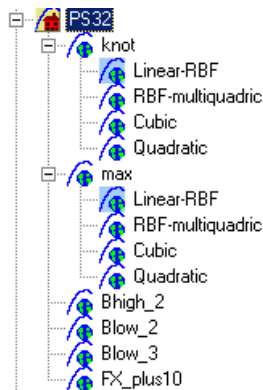
Try one of the default templates in the **Build Models** dialog as follows:

1 Select another global model such as Blow_2.

2 Click **Build Models** in the toolbar.

3 Select RBF Kernels and click **OK**. Click **OK** in the following RBF dialog to build a selection of child nodes. One of each type of RBF kernel is created.

Similarly you can use the **Build Models** dialog to automatically build a selection of polynomial models to search for the best fit.

4 Choose PRESS RMSE in the **Model Selection** dialog as the selection criteria for the best model, and click **OK**.

RBF-multiquadric
RBF-recmultiquadric
RBF-gaussian
RBF-thinplate
RBF-logisticrbf
RBF-wendland0
RBF-wendland2
RBF-wendland4
RBF-wendland6
RBF-linearrbf
RBF-cubicrbf

5 Look at the statistics in the lower list pane to quickly compare all the 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 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.

3

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 MBC 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 "Optimal Designs" on page 3-8.

Space-filling designs are better when there is low system knowledge. In cases where you are not sure what type of model is appropriate, and the constraints are uncertain, space-filling designs collect data in such as a way as to maximize coverage of the factors' ranges as quickly as possible. See "Space-Filling Designs" on page 3-34.

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 "Classical Designs" on page 3-25.

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 3-5
- "Optimal Designs" on page 3-8
- "Design Displays" on page 3-15
- "Prediction Error Variance Viewer" on page 3-19
- "Saving Designs" on page 3-41
- "Improving the Design" on page 3-22

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

- "Classical Designs" on page 3-25
- "Design Evaluation Tool" on page 3-31

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

- "Space-Filling Designs" on page 3-34
- "Applying Constraints" on page 3-36

For more details on functionality in the Design Editor, see the reference section "Designs" on page 5-1.

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**.
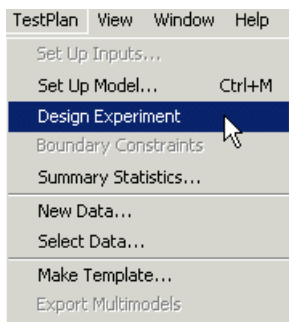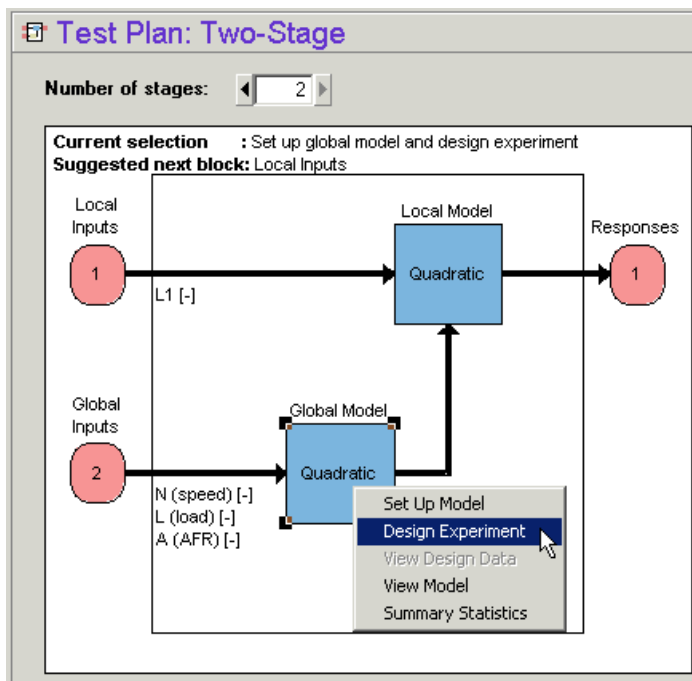
The **Design Editor** window appears.

Creating a New Design

1 Click the 🗋 button in the toolbar or select **File –> New**. A new node called Linear Model Design appears.
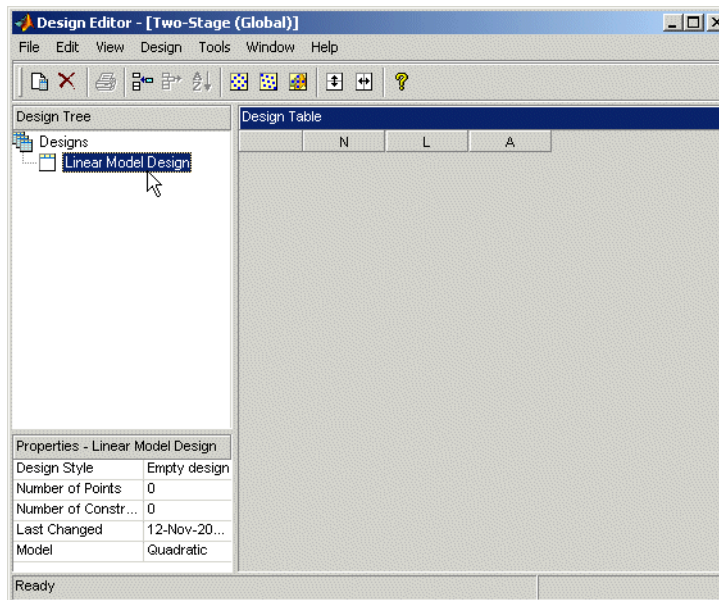
2 The new Linear Model Design node is automatically selected. An empty Design Table appears (see above) because you have not yet chosen a design. For this example you create an optimal design for the default global model, which is a quadratic.

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).

Optimal Designs

Choose an optimal design by clicking the ▦ button in the toolbar, or choose
Design –> Optimal.
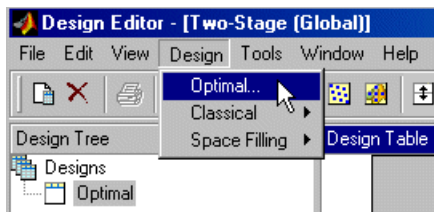
Optimal designs are best for cases with high system knowledge, where
previous studies have given confidence on the best type of model to be fitted,
and the constraints of the system are well understood.

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

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

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

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

- Starting point and number of points in the design
- Candidate set of points from which the design points are chosen

• Options for the algorithm that is used to generate the points

Start Point Tab

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

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

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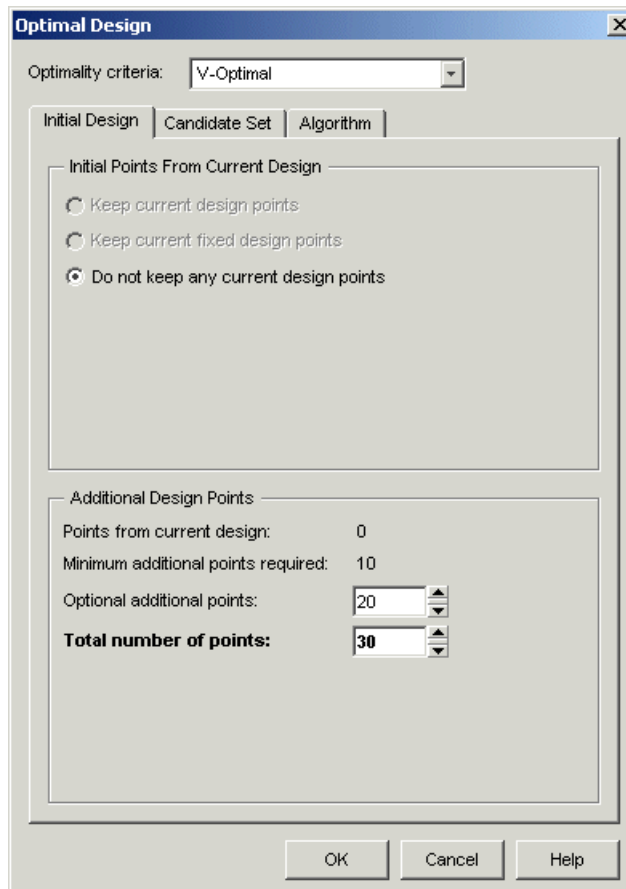
Candidate Set Tab

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

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

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

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

Select variables in this list

Choose Grid from this list

Open display windows with these buttons

Change the number of levels of the selected variable here

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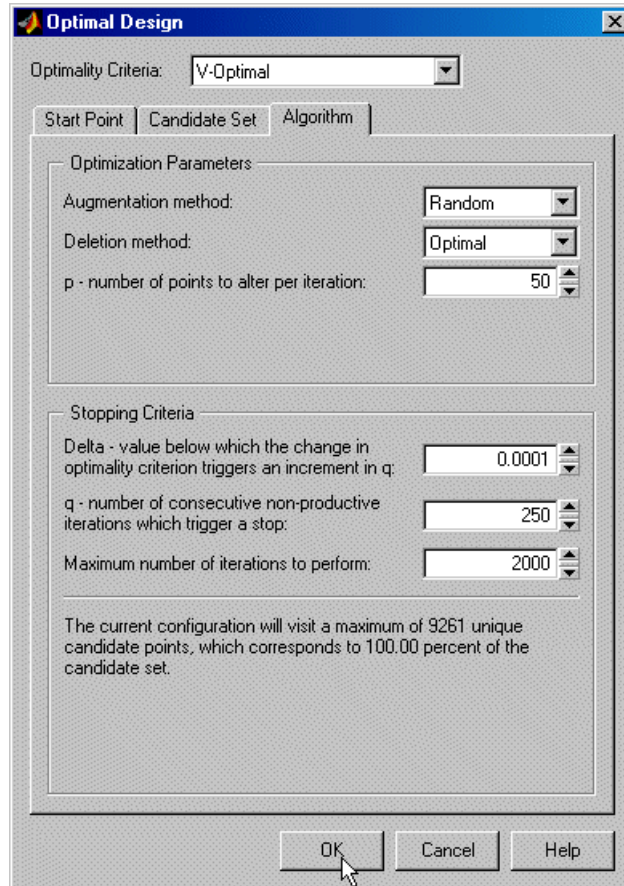
1 Leave the algorithm settings at the defaults and click **OK** to start optimizing the design.

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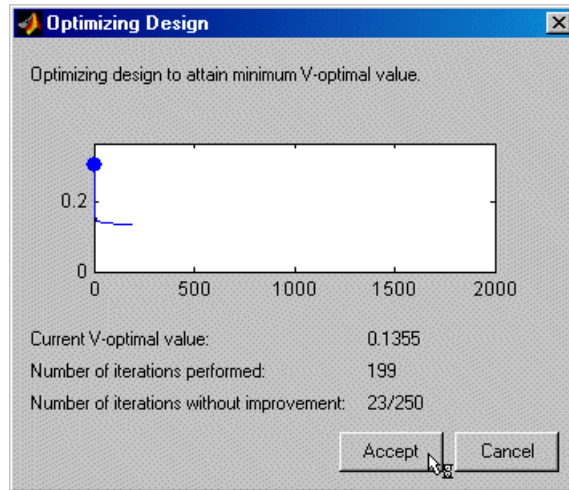
2 Click **Accept** when iterations are not producing noticeable improvements; that is, the graph becomes very flat.

A

Design Displays

When you press the **Accept** button, you return to the **Design Editor**.
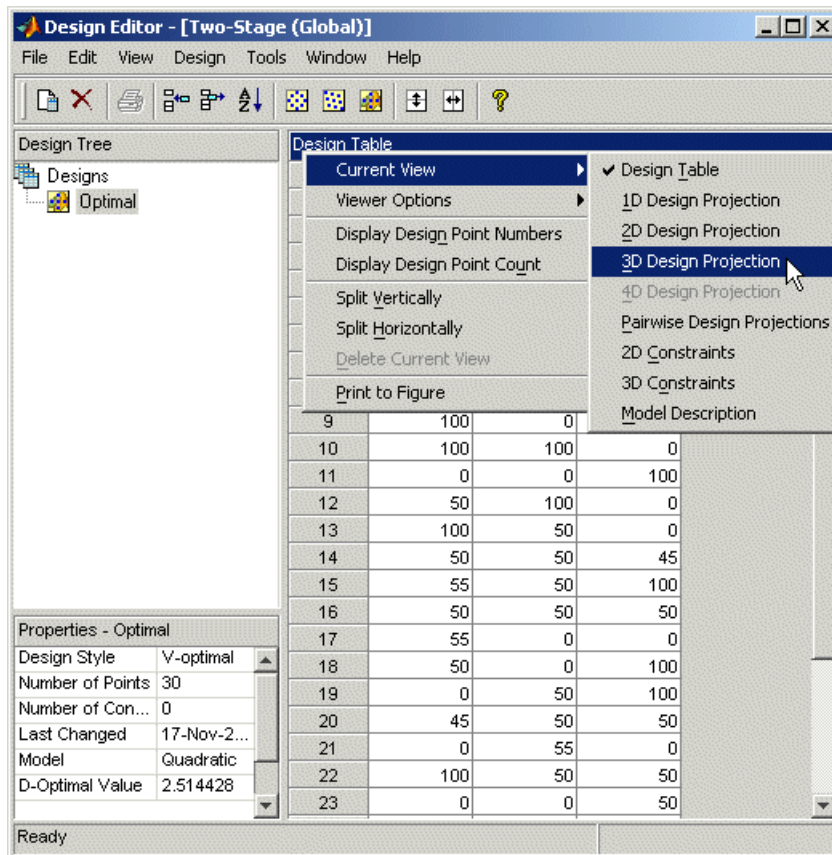
When you first see the main display area, it shows the default **Design Table** view of the design (see preceding example). There is a context menu, available by right-clicking on the title bar, in which you can change the view of the design to **1-D**, **2-D**, **3-D**, **4-D**, and **Pairwise Projections**, **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" on page 5-8.

Display Options

The **Design Editor** can display multiple design views at once, so while working on a design you can keep a table of design points open in one corner of the window, a 3-D projection of the constraints below it and a 2-D or 3-D plot of the current design points as the main plot. The following example shows several views in use at once.
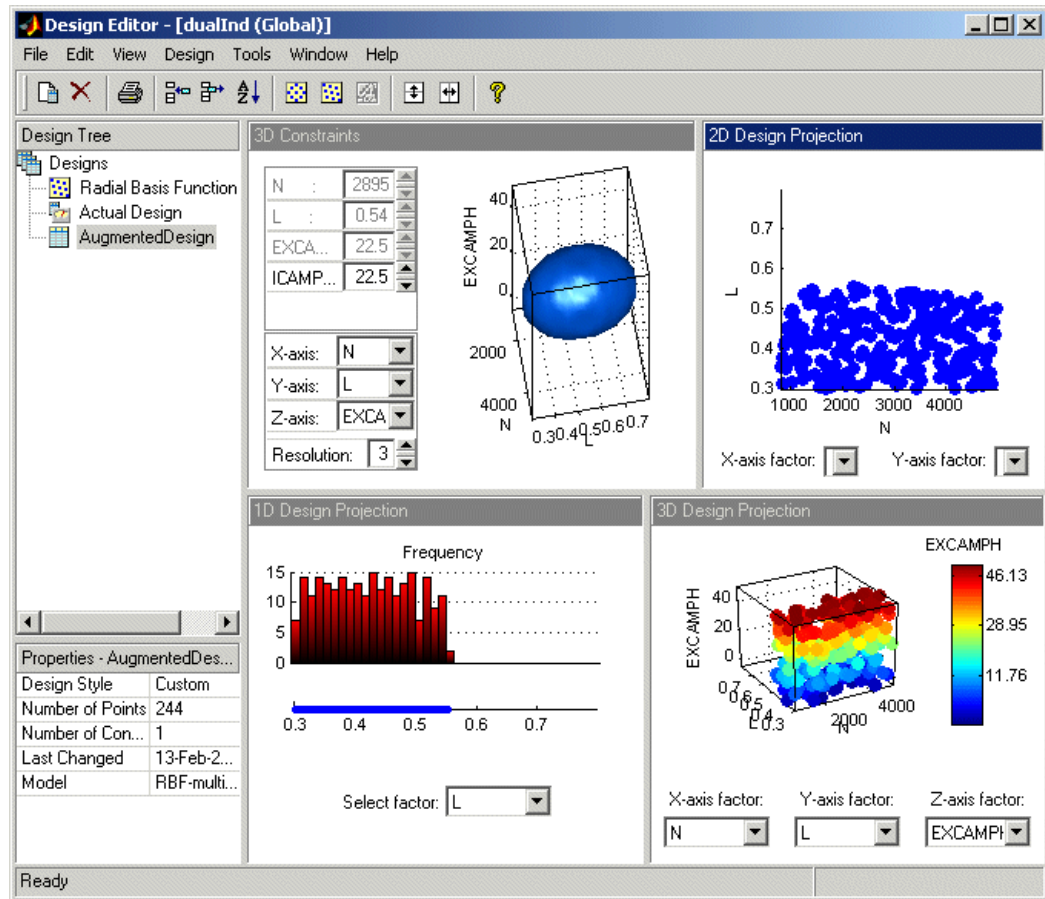
The current view and options for the current view are available either through the context menu or the **View** menu on the **Design Editor** window.

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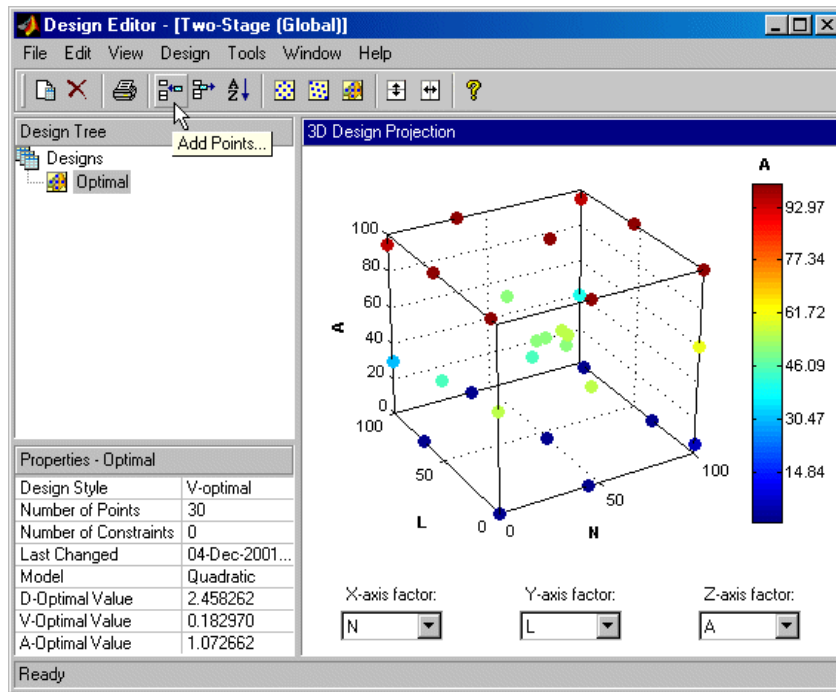
2 You can rotate the projection with click-drag mouse movement. View your design in several projections (singly, or simultaneously by dividing the pane) by using the right-click context menu in the display pane.

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 **PEV Viewer** is only available for linear models. The **PEV 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**.
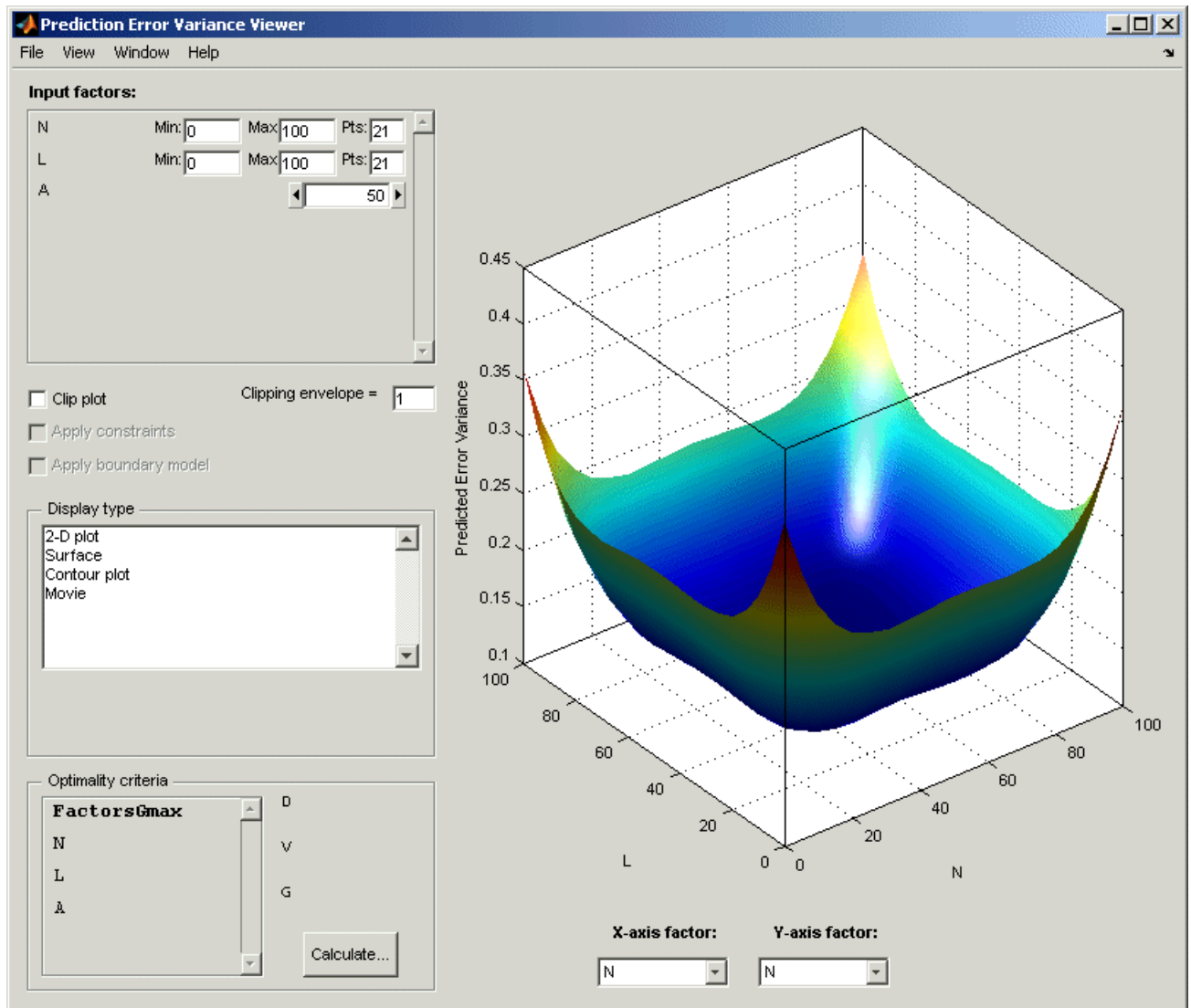
The default view is a 3-D plot of the PEV surface.

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" on page 8-7.

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.
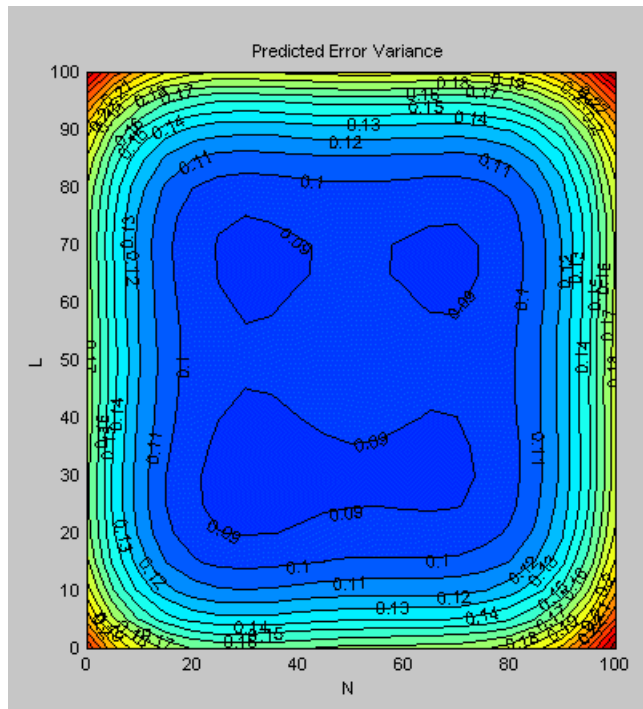
Improving the Design

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).
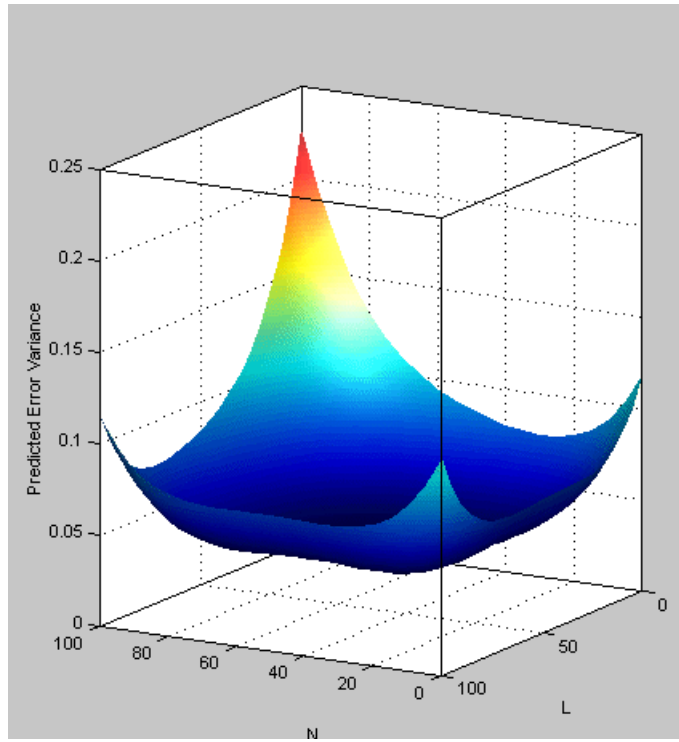
Classical Designs

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 out different operations starting from a basic design, then select the most appropriate one to use. The hierarchy allows clear viewing of the effects of changes on designs. The locking of parent designs also gives you the ability to easily reverse out of changes by retreating back up the tree.

3 Select the new design node in the tree. Notice that the display remains the same — all the points from the previous design remain, to be deleted or added to as necessary. The new design inherits all its initial settings from the currently selected design and becomes a child node of that design.

4 Rename the new node `Classical` by clicking again or by pressing **F2**.

5 Click the button in the toolbar or select **Design –> Classical –> Design Browser**.
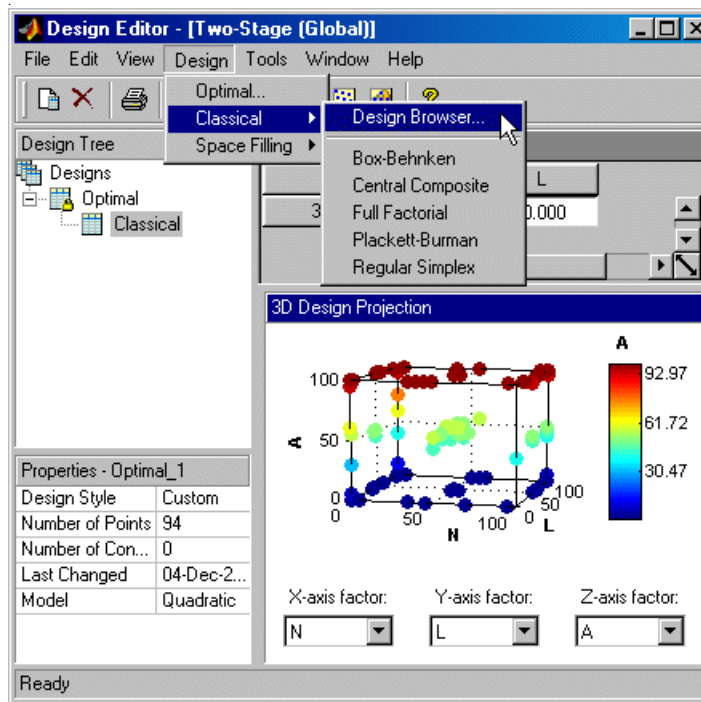
Note In cases where the preferred type of classical design is known, you can go straight to one of the five options under **Design –> Classical**. Choosing the **Design Browser** option allows you to see graphical previews of these same five options before making a choice.

A dialog appears because there are already points from the previous design. You must choose between replacing and adding to those points or keeping only fixed points from the design.

6 Choose the default, replace current points with a new design, and click **OK**.

The **Classical Design Browser** appears.
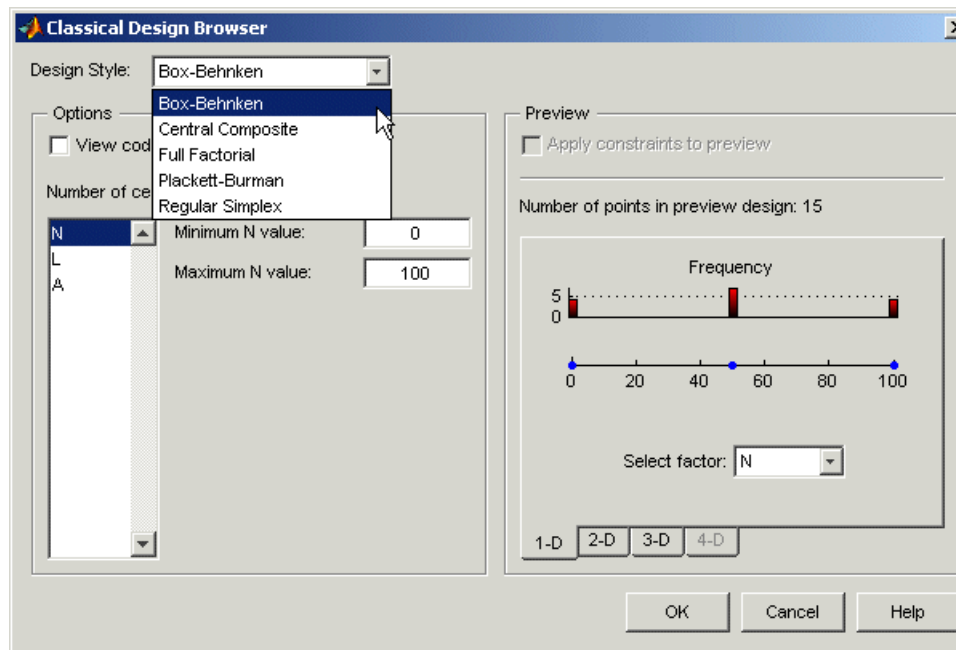
Classical Design Browser

In the **Design Style** drop-down menu there are five classical design options:

- **Central Composite**

  Generates a design that has a center point, a point at each of the design volume corners, and a point at the center of each of the design volume faces. You can choose a ratio value between the corner points and the face points for each factor and the number of center points to add. You can also specify a spherical design. Five levels are used for each factor.

- **Box-Behnken**

  Similar to Central Composite designs, but only three levels per factor are required, and the design is always spherical in shape. All the design points (except the center point) lie on the same sphere, so there should be at least three to five runs at the center point. There are no face points. These designs are particularly suited to spherical regions, when prediction at the corners is not required.

- **Full Factorial**

  Generates an *n*-dimensional grid of points. You can choose the number of levels for each factor and the number of additional center points to add.

- **Plackett Burman**

  These are "screening" designs. They are two-level designs that are designed to allow you to work out which factors are contributing any effect to the model while using the minimum number of runs. For example, for a 30-factor problem this can be done with 32 runs.

- **Regular Simplex**

  These designs are generated by taking the vertices of a k-dimensional regular simplex (k = number of factors). For two factors a simplex is a triangle; for three it is a tetrahedron. Above that are hyperdimensional simplices. These are economical first-order designs that are a possible alternative to Plackett Burman or full factorials.

Set Up and View 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.
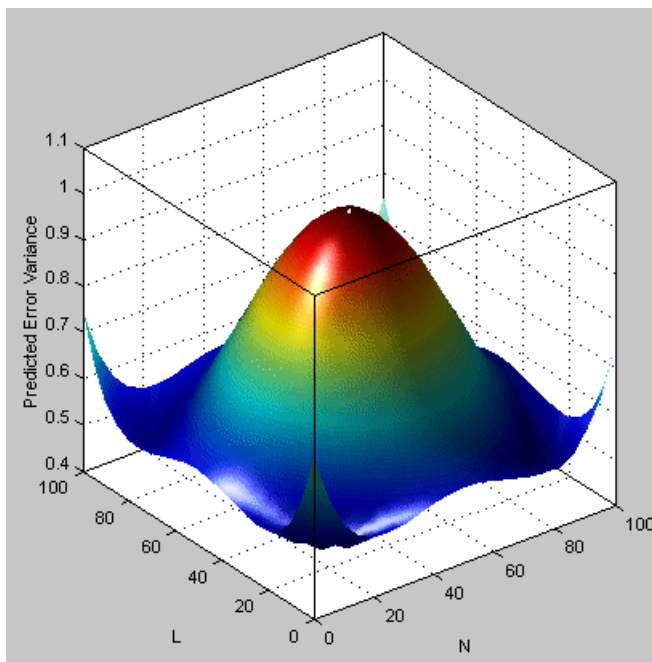
As you can see, this is not a realistic comparison, as this design has only 13 points (you can find this information in the bottom left of the main **Design Editor** display), whereas the previous optimal design had 100, but this is a good illustration of leverage. A single point in the center is very 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.

6 Select **Movie**, and you see this white contour line as the surface moves through the plane of value 1.

7 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
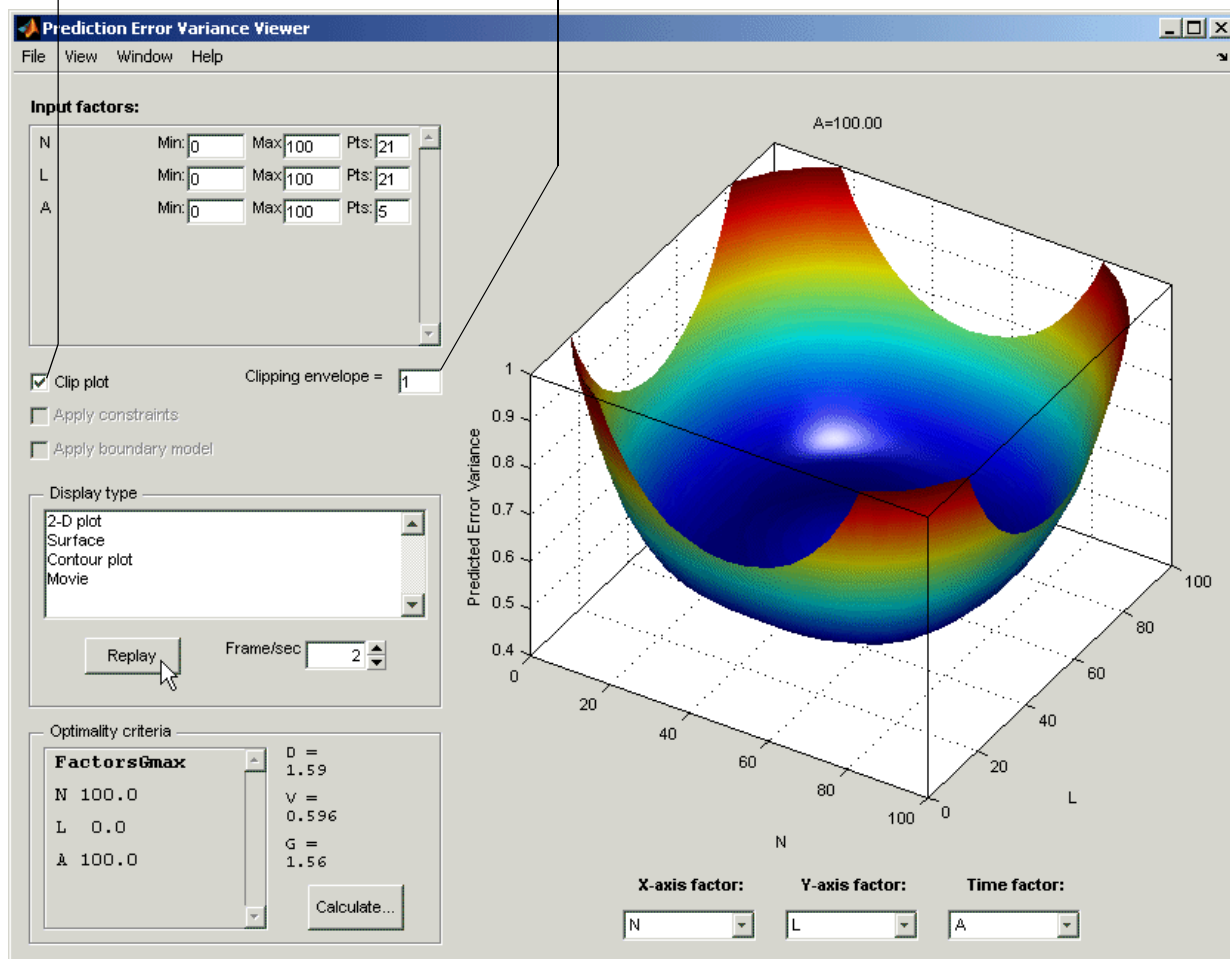
Design Evaluation Tool

The Design Evaluation Tool is only available for linear models. See also "Global Model Class: Multiple Linear Models" on page 7-55.

1 Return to the **Design Editor** and select **Tools –> Evaluate Designs**.

2 Choose the Box-Behnken design and click **OK** in the **Select Designs** dialog.

The **Design Evaluation Tool** displays a large amount of statistical information about the design.

3 Select **Hat Matrix** from the list on the right.

4 Click the **Leverage Values** button.

Note that the leverage of the central point is 1.00 (in red) and the leverage of all other points is less than this. The design would clearly be strengthened by the addition of more central points. Obviously this is a special case, but for any kind of design the Design Evaluation Tool is a powerful way to examine properties of designs.

5 Select **Design Matrix** from the list box.

6 Click the **3D Surface** button in the toolbar.
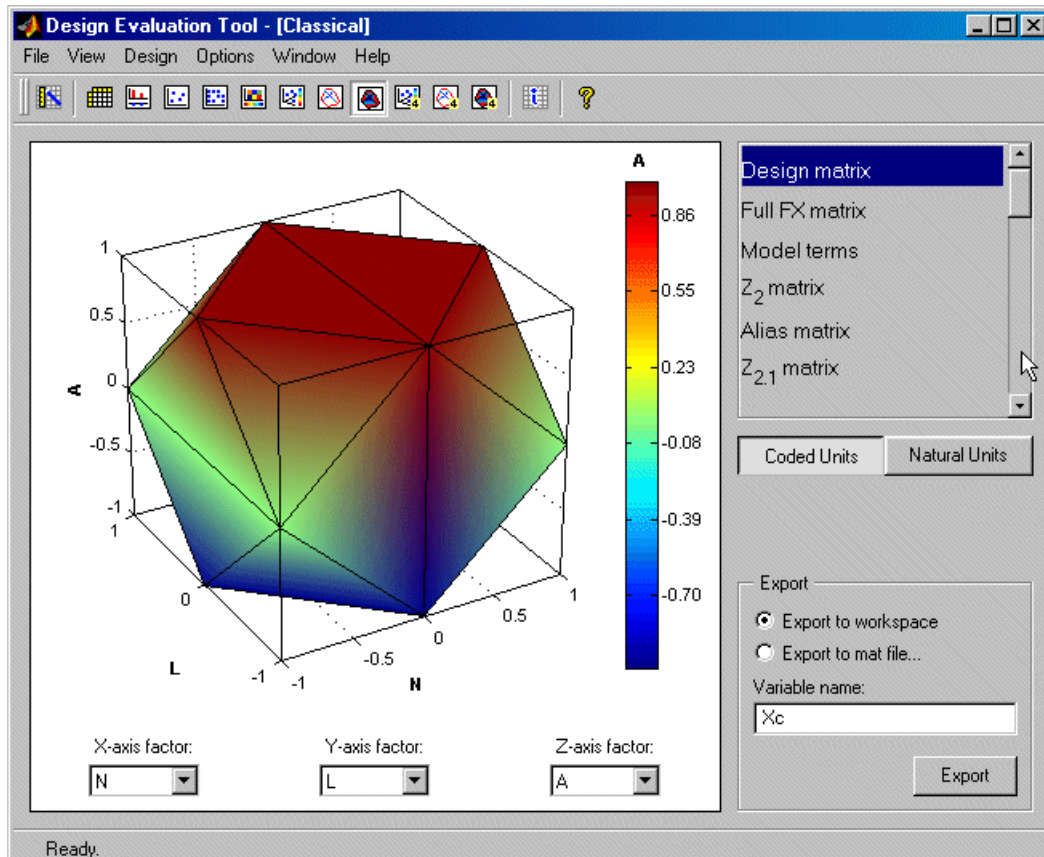
This illustrates the spherical nature of the current design. As usual, you can rotate the plot by clicking and dragging with the mouse.

There are many other display options to try in the toolbar, and in-depth details of the model terms and design matrices can all be viewed. You can export any of these to the workspace or a .mat file using the **Export** box.

For a description of all the information available here, see "Design Evaluation Tool" on page 8-30.

Improving the Design

To strengthen the current Box-Behnken design near the center region:

1 Close the **Design Evaluation Tool**.

2 Return to the **Design Editor** window.

3 Select **Design –> Classical –> Box-Behnken**.

4 Click **OK** to replace the current points with a new design.

5 Increase the number of center points and click **OK**.

6 Once again select **Tools –> 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.

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 in conjunction 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.
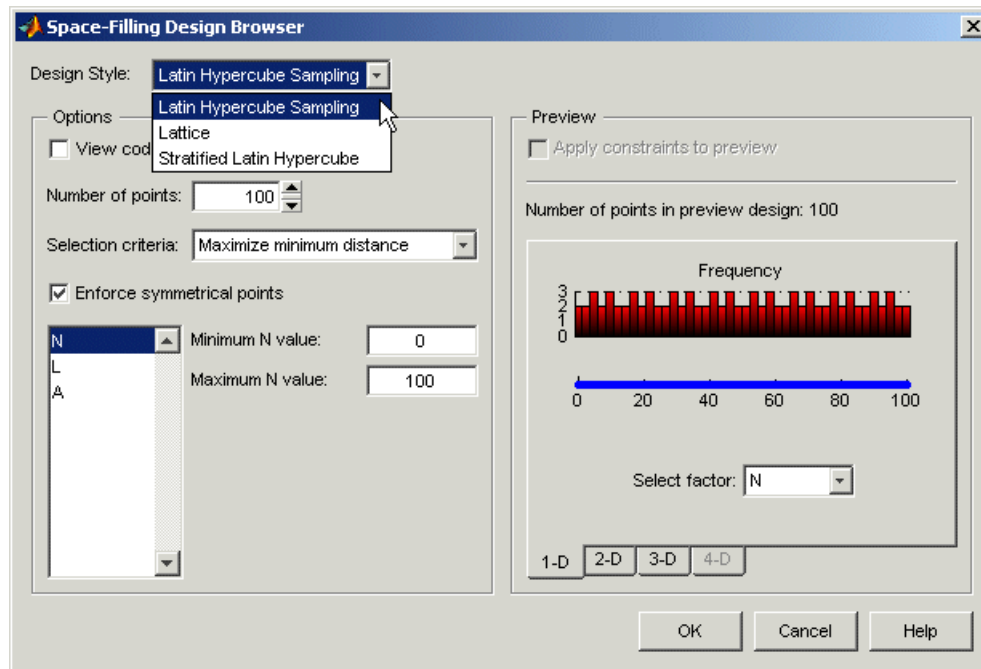
Setting Up a Space-Filling Design

1 Leave the **Design** 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 centre 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. For further details, see "Applying Constraints" on page 5-42 for a full description of the constraint functions.

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.
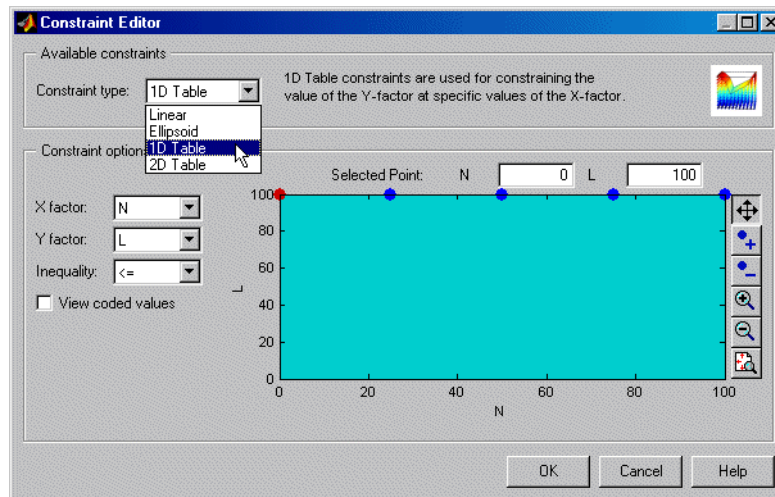
4 You can select the appropriate factors to use. For this example choose speed (N) and air/fuel ratio (A).

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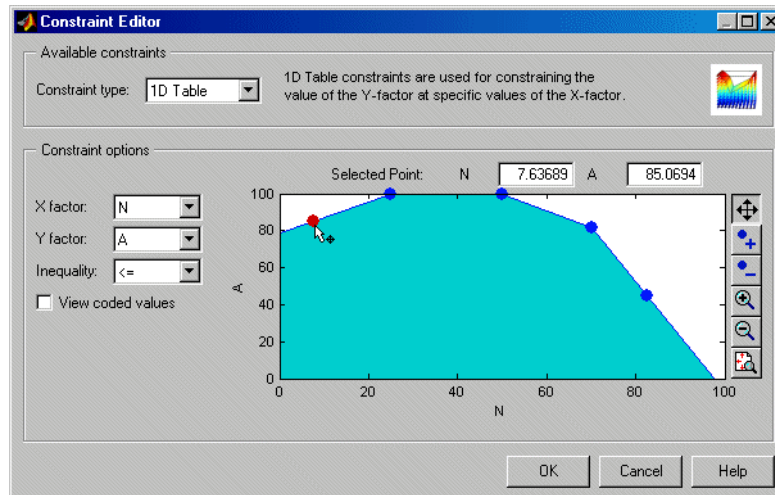
6 Click **OK**.

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.
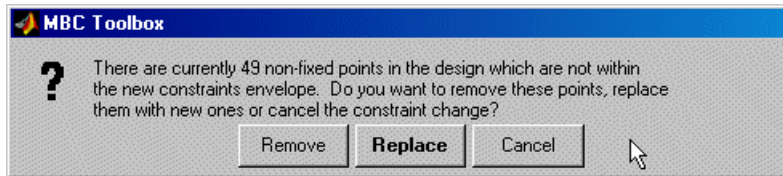
7 The default is to remove the points outside the new constraint area; choose this.
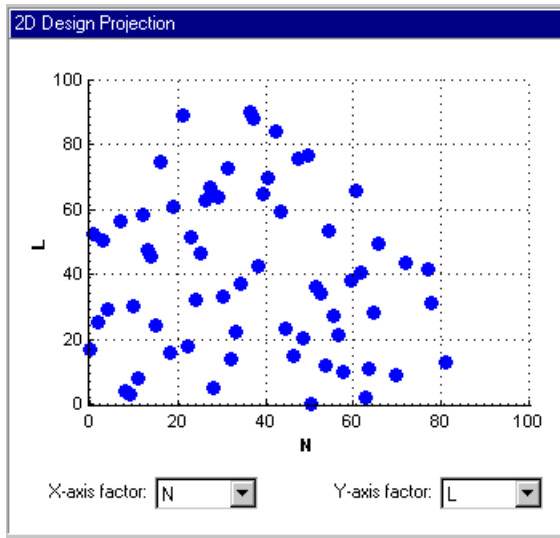
If you examine the 2-D projection of the hypercube you will notice the effects of the new constraint on the shape of the design, as shown in the preceding example.

8 Right-click the display pane to reach the context menu.

9 Select **Current View –> 3D Constraints**.
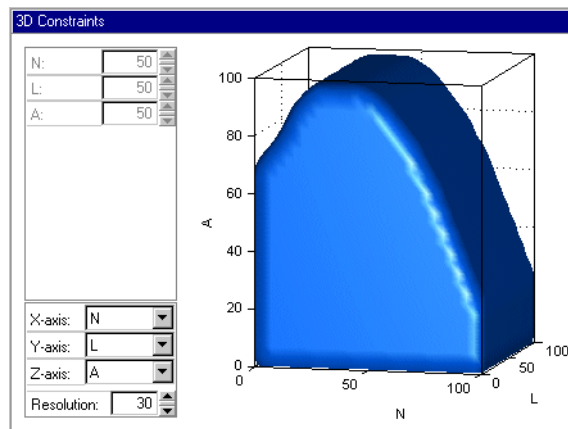
These views are intended to give some idea of the region of space that is currently available within the constraint boundaries.

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

11 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.
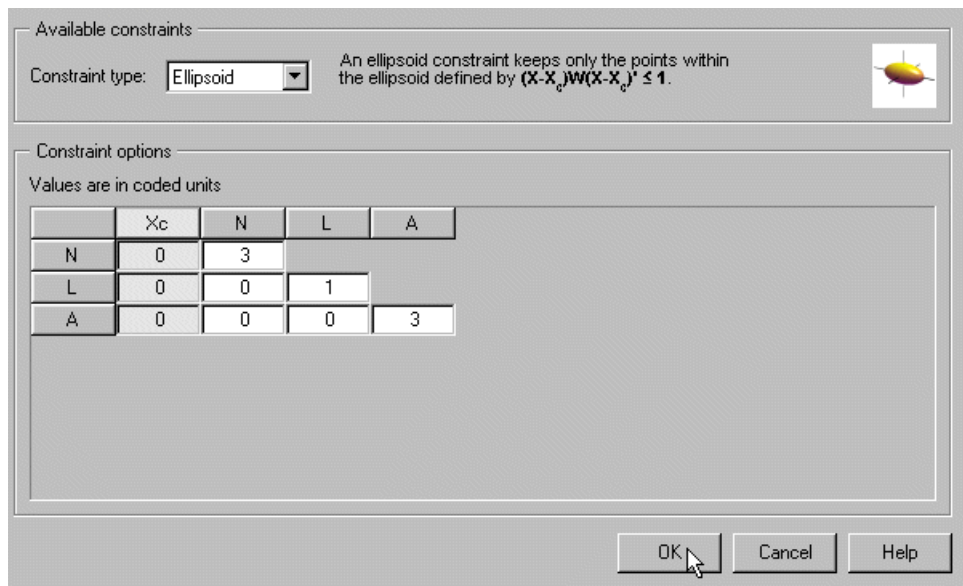
This reduces the space available for the candidate set by a third in the A and N axes, forming an ellipsoid, as shown below. The L axis, left at 1, is not constrained at the midpoint of N and A. To leave L unconstrained (a cylinder) put the value of L=0.

12 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.
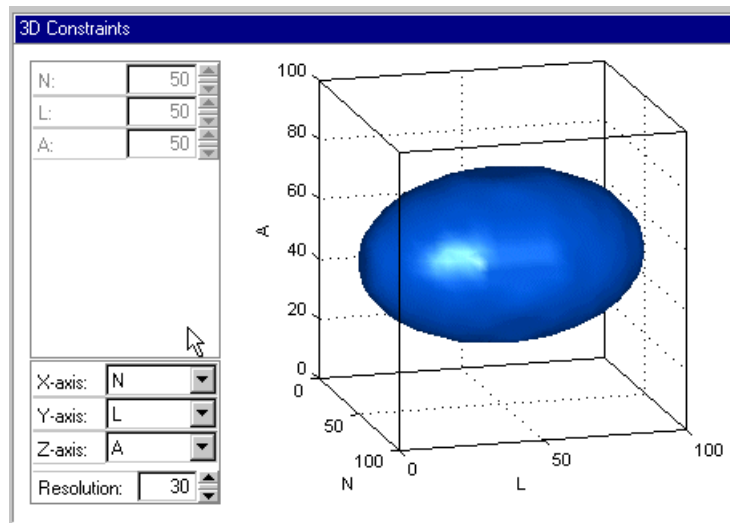
Both constraints are applied to this design, but the ellipsoid lies entirely within the previous 1-D table constraint.

Saving Designs

To save your design:

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

There are three options:

- **To File** generates a Design Editor file (`.mvd`).
- **To CSV 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.
- **To 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.

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, Concerto 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 only use 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" on page 8-77 in the "Technical Documents" section.

For comprehensive help on all data functions in the Model Browser, see "Data" on page 6-1.

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

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.

Note   You can delete data objects from the Project node. Select a data set in the Data Sets list and press the **Delete** key.
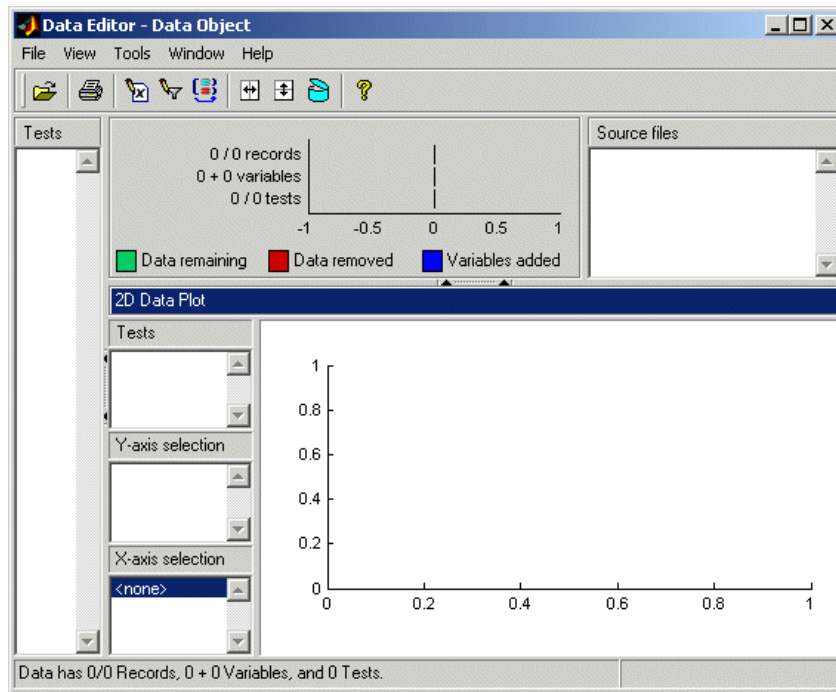
There is no plot until some data has been loaded.

By default the new data object is called **Data Object**. You can edit this name in the **Data Sets** pane in the project level view of the main Model Browser. You can change the names of data objects here by select-clicking a data object in the Data Sets list or pressing F2 (as when selecting to rename in Windows Explorer) and entering a new name.

Loading a Data File

1 Click the Open File icon in the toolbar ![icon] 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, Concerto 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**.

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	

Imported 270 records and 7 variables from files : D:\MATLABR12p1\toolbox\mbc\mbctraining\holliday.xls

[Cancel] [< Back] [Next >] [Finish]
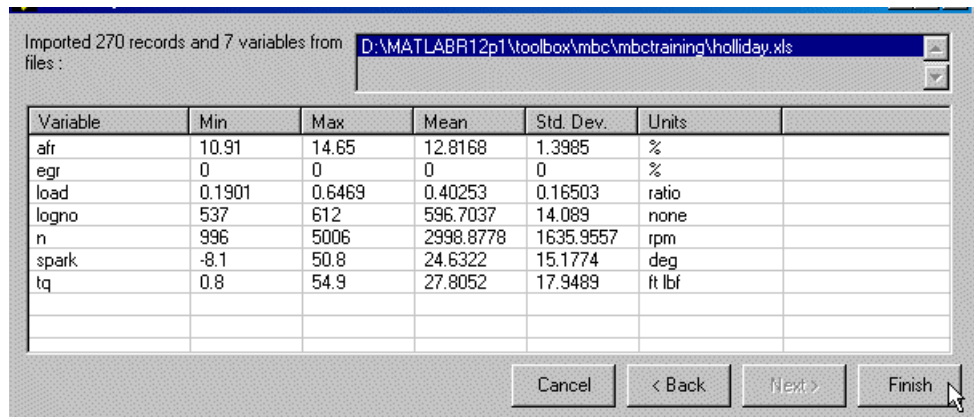
4 The **Data Import Wizard** displays a summary screen showing the total number of records and variables imported, and you can view each variable's range, mean, standard deviation, and units in the list box. Click **Finish** to accept the data. (If you have data loaded already, you cannot click **Finish** 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

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 Right-click the view and select **Split View –> Test Note Definitions**.

   The current view is divided into two.

5 Select **Tools –> Test Notes –> Add**.

   The **Test Note Editor** appears.

6 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 colour at the default and click **OK**.

7 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.

8 Click the column header of the new `Low torque` note in the **Notes** view. All the tests which satisfy the condition mean(tq)<10 are sorted to the top of the list.

9 Now create some more views.

- Right-click a view and select **Split View –> Data Table**.
- Right-click a view and select **Split View –> 3D Data Plot**.

10 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**.

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 4-13.

Reordering and Editing Data

To change the display right-click a 2-D plot and select **Viewer Options** –> **Properties**. You can alter grid and plot settings including lines to join the data points.

Reorder X Data in the **Plot Properties** dialog can be useful when record order does not produce a sensible line joining the data points. For an illustration of this:

1 Ensure you are displaying a 2-D plot. You can right-click on any plot and select **Current Plot** –> **2-D Plot**, or use the context menu split commands to add new views

2 Right-click on a 2-D plot and select **Viewer Options** –> **Properties** and choose solid from the **Data Linestyle** drop-down menu, as shown below. Click **OK**.
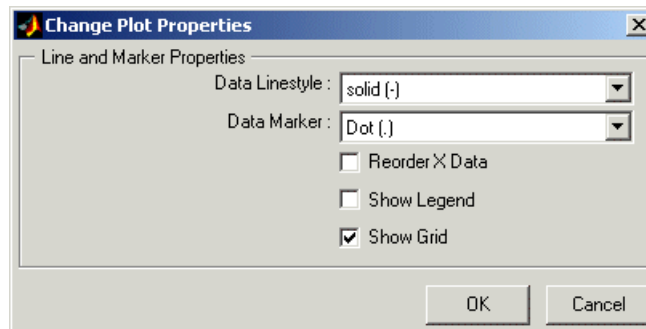
3 Choose afr for the *y*-axis.

4 Choose Load for the *x*-axis.

5 Select test 590. You must use the test controls contained within the 2-D plot. The **Tests** pane on the left applies to other views: tables and 3-D and multiple data plots.
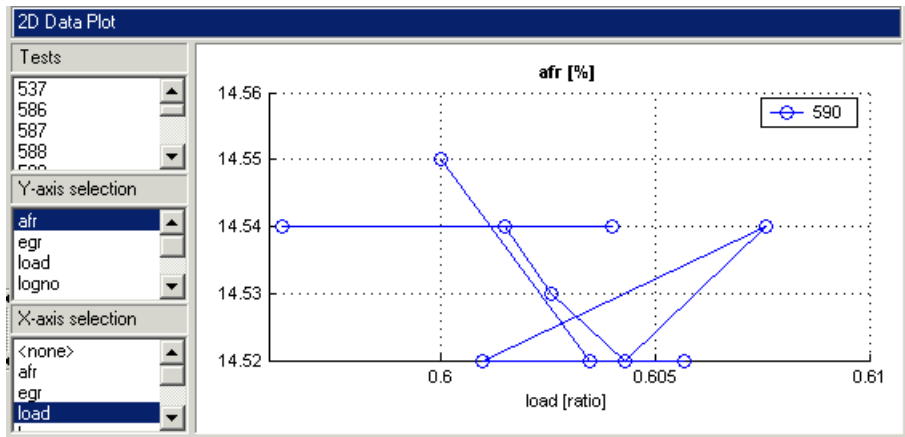
6 Right-click and select **Viewer Options** –> **Properties** and choose **Reorder X Data**. Click **OK**.

This command replots the line from left to right instead of in the order of the records, as shown.
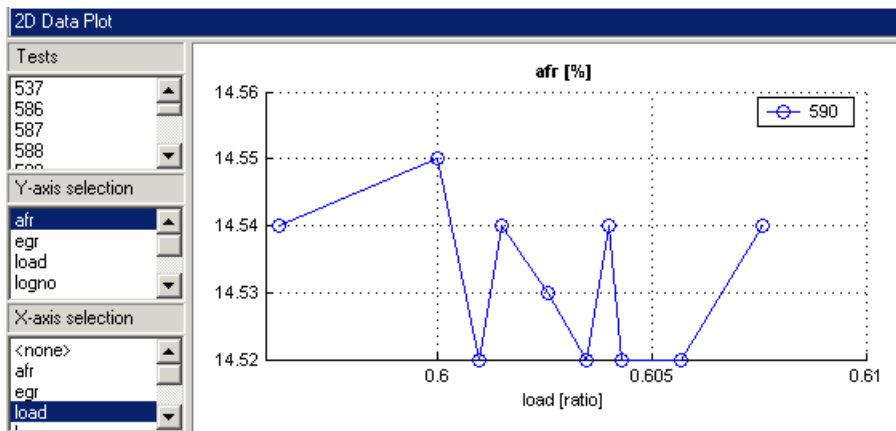
7 Right-click and select **Split Plot –> Data Table** to split the currently selected view and add a table view. You can select particular test numbers in the **Tests** pane on the left of the Data Editor. You can right-click to select **Viewer Options –> Allow Editing**, and then you can double-click cells to edit them.

User-Defined Variables & Filtering

You can add new variables to the data set, and you can remove records by imposing constraints on the data.

- 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.

8 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.

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

Specify the equation of the new variable
i.e. NEW_VAR = OLD_VAR * 3 + OLD_VAR_1

POWER=tq*n

Variable Unit :

OK   Cancel   Help
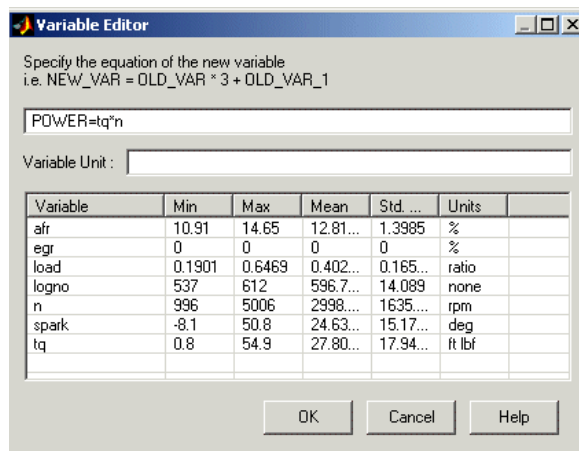
9 Click **OK** to add this variable to the current data set.

10 This new variable can be seen in the **Data Editor** by right-clicking in a view and selecting **Split Plot –> Variable List**. 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.
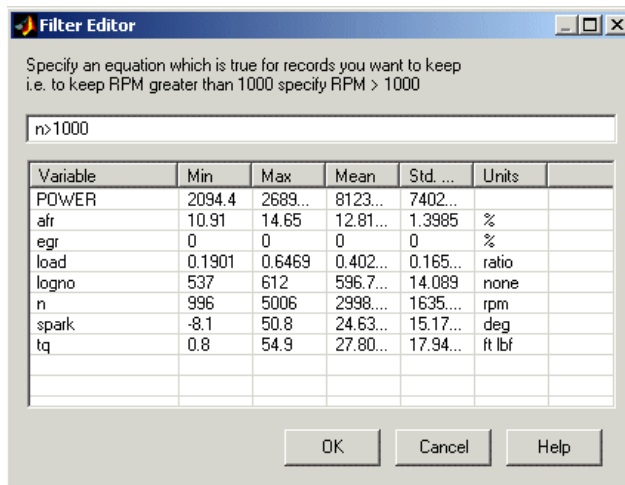
You define the filter using logical operators on the existing variables.

2 Keep all records with speed (**n**) greater than 1000. Type `n` (or double-click on the variable `n`), then type `>1000`.

3 Click **OK** to impose this filter on the current data set.

4 This new filter can be seen in the **Data Editor** by right-clicking in a view (try the **Variable List** view) and selecting **Split Plot –> Filter List**. A new view appears containing a list of your user-defined filters and information on how many records are removed by the new filter. You can also now see `141/270`

records next to the top information bars and a red section illustrating the records removed by the filter.
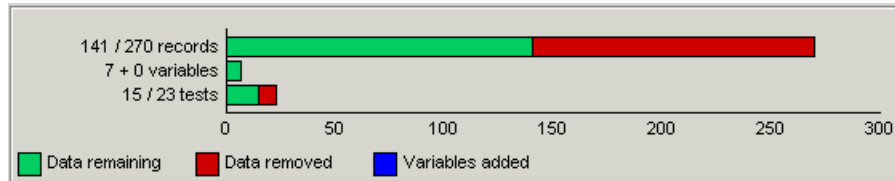
Sequence of Variables

You can change the order of user-defined variables in the Variable Editor list using the arrow buttons.

Select **Tools** –>**Variables** –> **Edit** to open the **Variable Editor**.
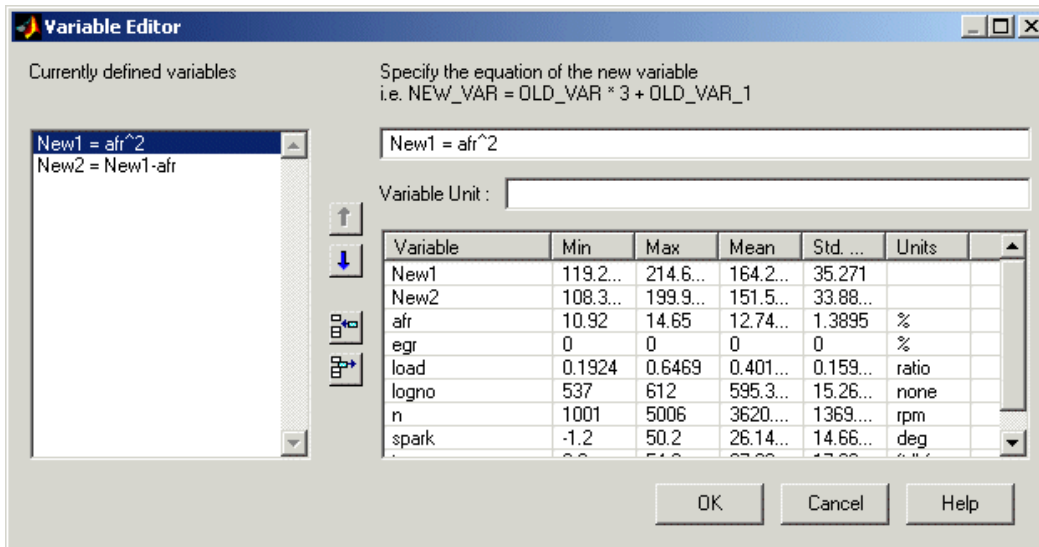
Example:

1 Define two new variables, **New1** and **New2**. Note that you can use the buttons to add or remove a list item to create or delete variables in this view. 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 to produce this erroneous situation. You see the following:

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
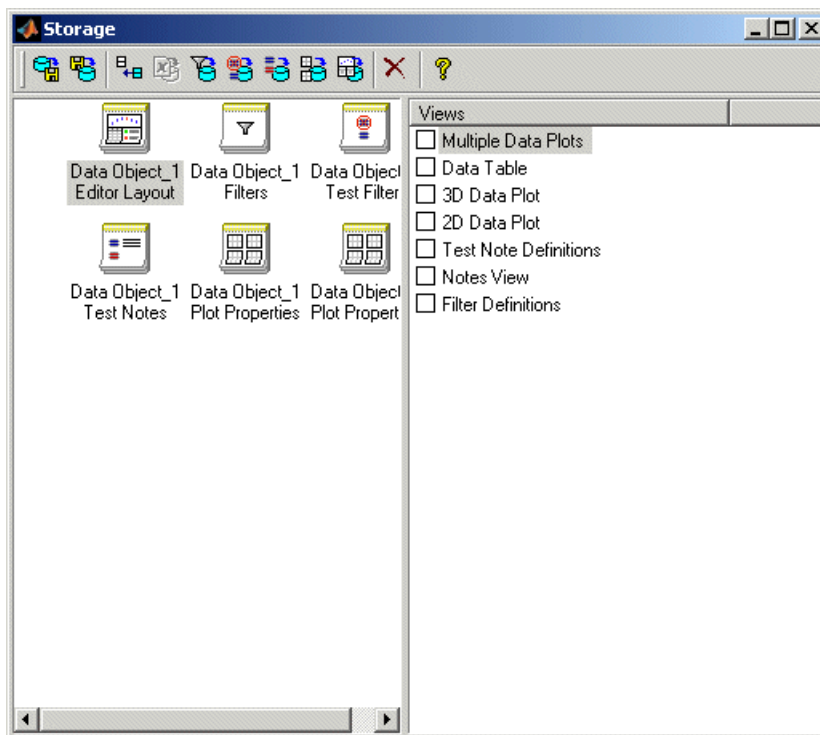
The example above contains a variety of stored objects. The toolbar buttons **Store Current Variables** and **Store Current Filters, Test Filters** or **Test Notes** allow you to put all user-defined variables and filters from the current session into Storage. They appear in the **Storage** window. All 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 (use the X button).

For a detailed description of the functionality in Storage, see "Storage" on page 6-25.

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, no plot is displayed.

Select a 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 Add **load** to the list by selecting it on the right and clicking .

4 Change the tolerance to 0.01 and watch the test grouping change in the plot.

5 Clear the **Group By** check box. Notice that variables can be 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.

6 Select load in the list view (it becomes highlighted in blue).

7 Remove the selected variable, load, by clicking the ⊞ button.

You should return to the view below, when only n was plotted and used to define groups.
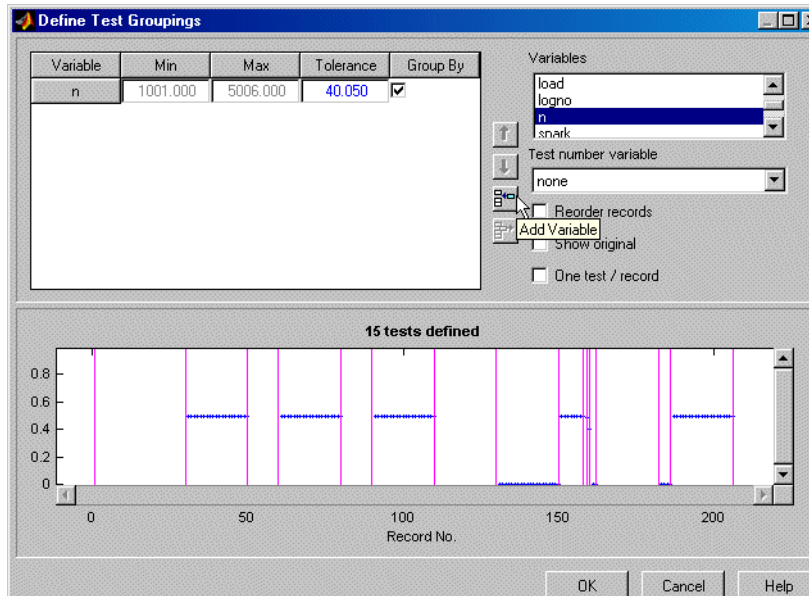
Reorder records allows records in the data set to be reordered before grouping. Otherwise the groups are defined using the order of records in the original data object.

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 can be selected to number the tests.

4-19

8 Choose logno from the pop-up list of **Test number variable**s.

Test number can be a useful variable for identifying individual tests in Model Browser 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.

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).

9 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).
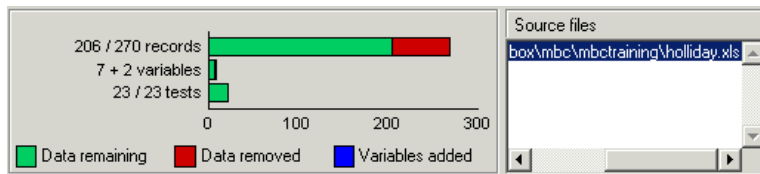
10 The number of records shows the number of values left (after filtration) of each variable in this data set, followed by the original number of records. The color coded bars also display the number of records removed as a proportion of the total number. The values are collected into a number of tests; this number is also displayed. The variables show the original number of variables plus user-defined variables. You can remove or show the legend explaining the color coding by clicking the button.

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 up 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 up 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 up to the experimental design.
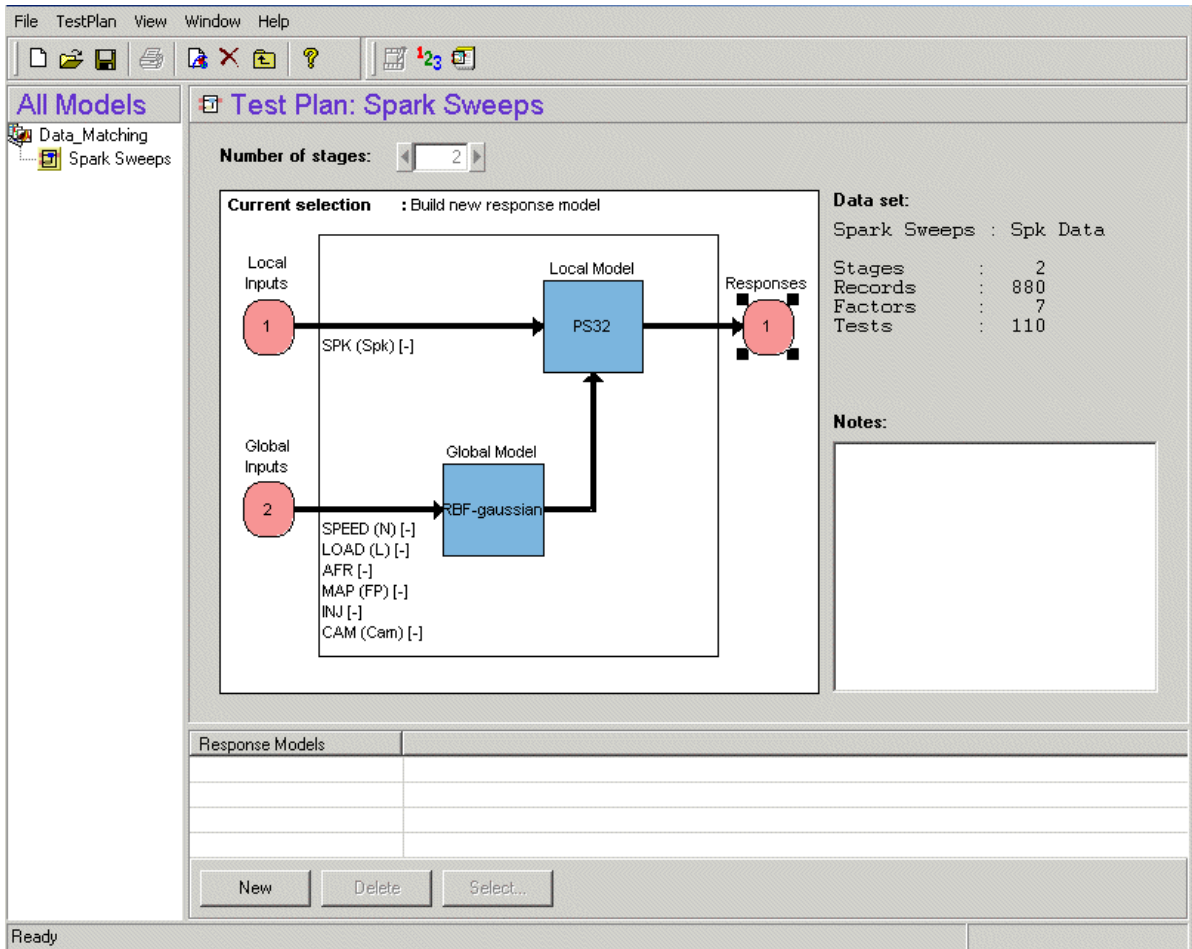
3 Click the **Select Data** button (¹23) in the toolbar.

   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**.
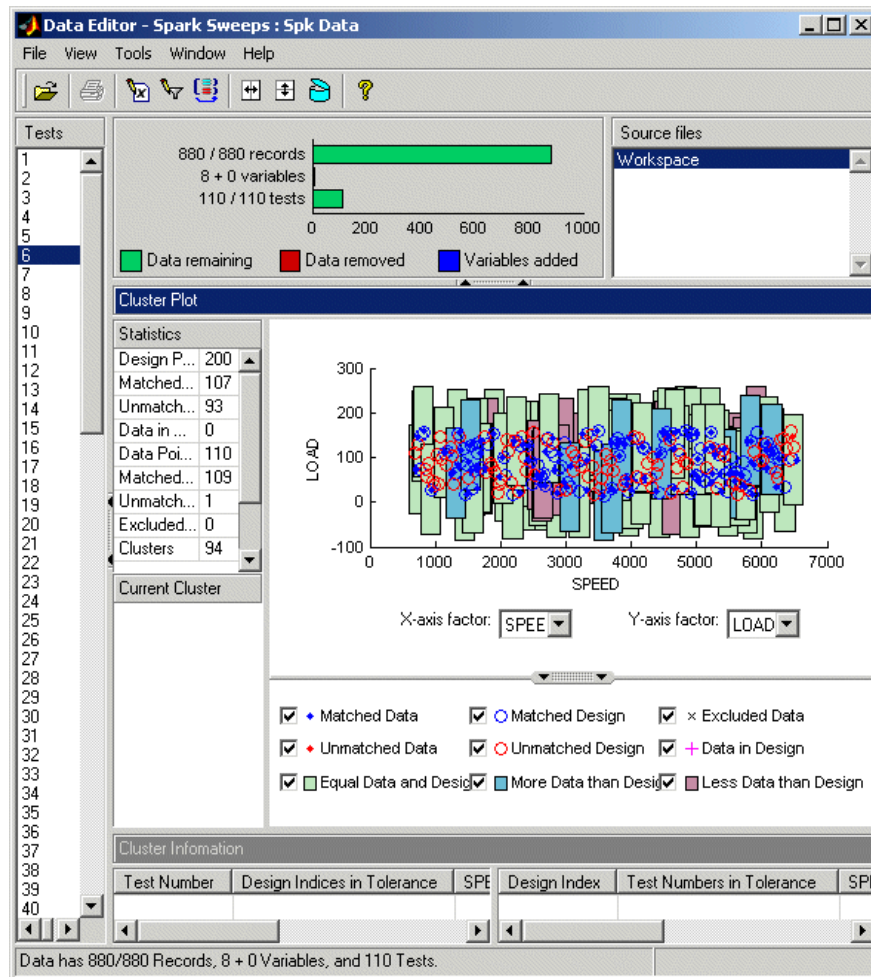
In the Cluster Plot you can see colored areas containing points. These are 'clusters' where closely matching design and data points have been selected by the matching algorithm.

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.
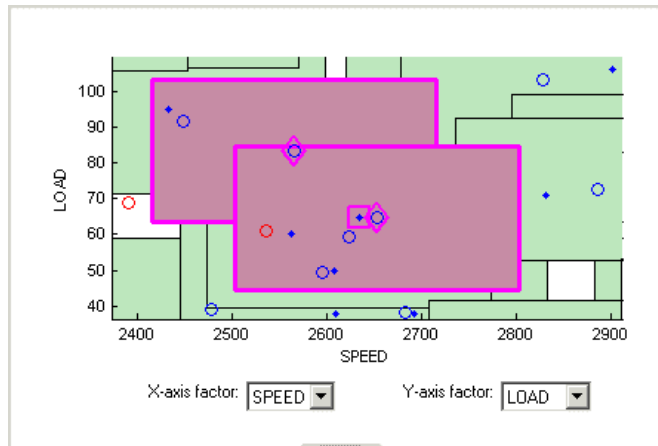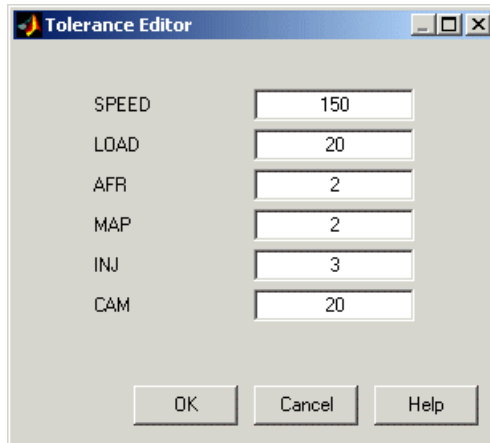
Notice the shape formed by overlapping clusters. The example shown outlined in pink is a single cluster formed where a data point lies within tolerance of two design points.

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 4-27.

Tolerances and Cluster Information

1 To edit tolerance values, select **Tools –> Tolerances**.

The **Tolerance Editor** appears. Here you can change the size of clusters in each dimension. Observe that the LOAD tolerance value is currently 100. This accounts for the elongated shape (in the LOAD dimension) of the clusters in the current plot, because this tolerance value is a high proportion of the total range of this variable.

2 Click the **LOAD** edit box and enter 20, as shown. Click **OK**.

Notice the change in shape of the clusters in the **Cluster Plot** view.
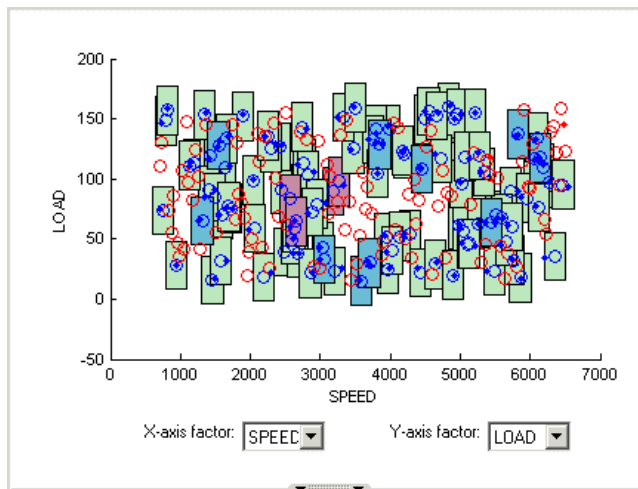
3 Shift click (centre-click) and drag to zoom in on an area of the plot, as shown. You can double-click to return to the full size plot.
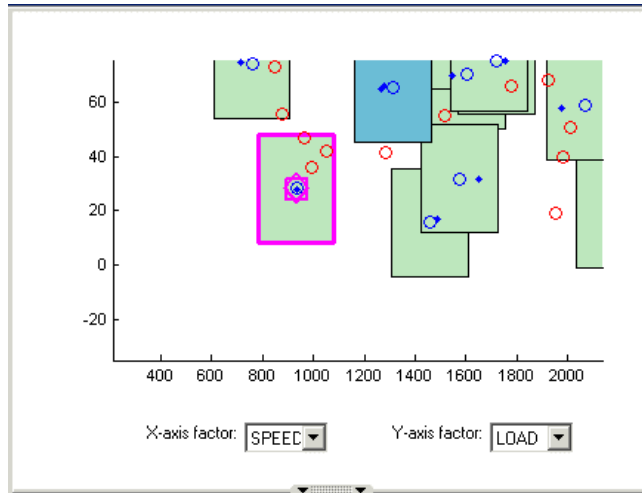
4 Click a cluster to select it. Selected points or clusters are outlined in pink. If you click and hold, you can inspect the values of global variables at the selected points (or for all data and design points if you click on a cluster). You can use this information to help you decide on suitable tolerance values if you are trying to match points up.

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 on the header bar of the **Cluster Plot** view and select **Split Vertically**. A new view appears underneath the cluster plot. Right-click the header bar of the new view and select **Current Plot –> Cluster Information**.

Cluster Infomation

Test Number	Design Indices in Tolerance	SPEED	LOAD	A	Design Index	Test Numbers in Tolerance	SPEED	LOAD	A
☑ 22	22, 137	2634	64.77	1	☑ 22	22	2653	64.55	1.
☑ 52	52, 137	2562	60.23	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 on 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. You can only restore these excluded data to the data set in this cluster view.

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 in order 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 on the **Cluster Plot** and select **Viewer Options –> Select Unmatched Data**. Notice that the remaining unmatched data points appear in the **Cluster Information** list view. Here you can use the check boxes to select or exclude unmatched data in the same way as points within clusters.

2 Select a cluster, then use the drop-down menu to change the **Y-Axis factor** to `INJ`. Observe the selected cluster now plotted in the new factor dimensions of `SPEED` and `INJ`.

You can use this method to track points and clusters through the dimensions. This can give you a good idea of which tolerances to change in order to get points matched. Remember that points that do not form a cluster may appear to be perfectly matched when viewed in one pair of dimensions; you must view them in other dimensions to find out where they are separated beyond the tolerance value. You can use this tracking process to decide whether you want particular pairs of points to be matched, and then change the tolerances until they form part of a cluster.

3 Clear the **Equal Data and Design** check box in the **Cluster Plot** view. You control what is plotted using these check boxes.
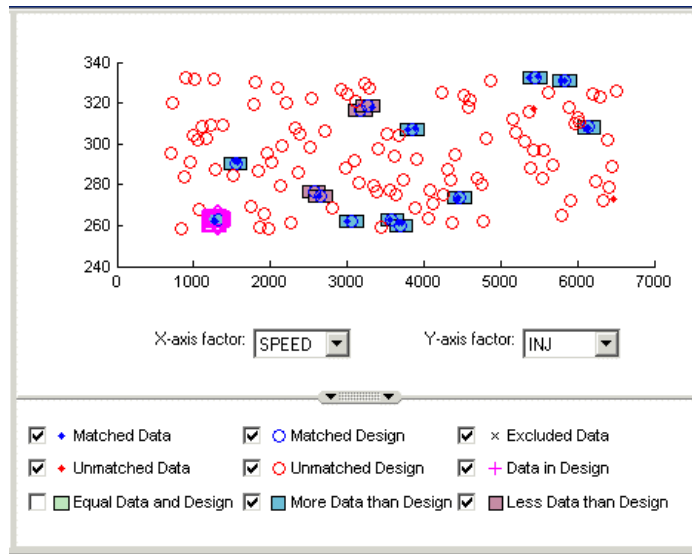
This removes the green clusters from view, as shown. These clusters are matched up; you are more likely to be interested in unmatched points and clusters with uneven numbers of data and design points. Removing the green clusters allows you to focus on these points of interest. If you want your new `Actual Design` to accurately reflect your current data, your aim is to get as many data points matched up to design points as possible; that is, as few red clusters as possible.

4 Clear the check box for **More Data than Design**. You may also decide to ignore blue clusters, which contain more data points than design points. These design points will be replaced by all data points within the cluster. An excess of data points is unlikely to be a concern.

However blue clusters may indicate that there was a problem with the data collection at that point, and you may want to investigate why more points than expected were collected.
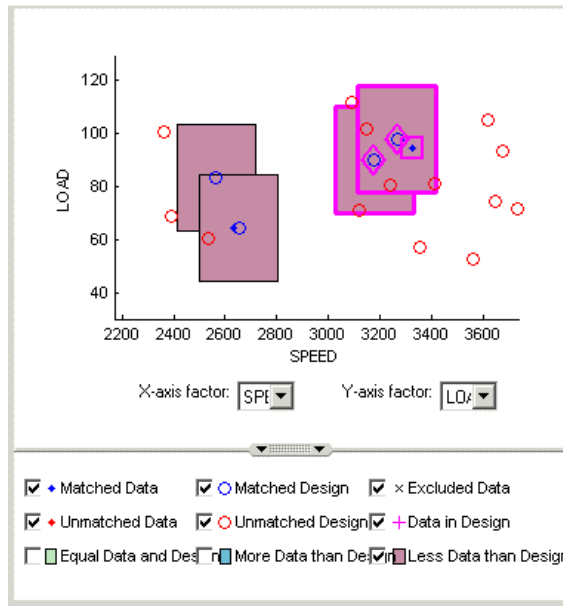
5 Select one of the remaining red clusters. Both of these have two design points within tolerance of a single data point.

6 Choose one of the design points to match to the data point, then clear the check box of the other design point. The cleared design point remains unchanged in the design. The selected design point will be replaced by the matched data point.

Notice that the red cluster disappears. This is because your selection results in a cluster with an equal number of selected data and design points (a green cluster) and your current plot does not display green clusters.

7 Repeat for the other red cluster.

Now all clusters are green or blue. There are two remaining unmatched data points.

8 Clear the **Unmatched Design** check box to locate the unmatched data points. Select **Unmatched Design** check box again – you need to see design

points to decide if any are close enough to the data points that they should be matched.

9 Locate and zoom in on an unmatched data point. Select the unmatched data point and a nearby design point by clicking, then use the axis drop-down menus to track the candidate pair through the dimensions. Decide if any design points are close enough to warrant changing the tolerance values to match the point with a design point.

10 Recall that you can right-click on 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, for the selection of modeling data and to augment your design based on actual data obtained. Usually some trial and error will be required 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" on page 6-1.

5

Designs

This section discusses the following topics:

The Design Editor

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.

There is a step-by-step guide to using the Design Editor in the "Tutorial: Design of Experiment" on page 3-1.

Before you design an experiment we recommend that you set up your input variables and model type as these affect your designs, as described below in "Opening the Design Editor" on page 5-3.

The functionality in the Design Editor is covered in the following sections:

Opening the Design Editor

You must first have a test plan before you can open the Design Editor.

1 From the startup (project) view of the Model Browser, click **New** and select a one or two-stage test plan. See "Project Level: Startup View" on page 7-3 and "Test Plans" on page 7-17 in the Modeling section.

You can design experiments at both stages, for local models and global models; for most two-stage models the global model is most appropriate for design of experiment.

2 Before you design an experiment we recommend that you set up your input variables, by double-clicking the Inputs blocks on the test plan diagram. See "Setting Up Inputs" on page 7-38.

You can choose the number of inputs for your model and set up their names and definitions, then you can design an experiment to collect data. It is much easier to understand your design points if they are labeled with the factor names. Also, if you do not set up model inputs first, then you can only create designs for the default number of variables (one).

3 If you want to use optimal designs, then the type of model you are going to use to fit the data is important, and you should choose a model type before opening the Design Editor. Double-click a model block in the test plan diagram to set up model types. 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, so in these cases you should pick your model type before designing an experiment. See Chapter 7, "Modeling," to find out about model types in the Model-Based Calibration Toolbox.

If you have no idea what model you are going to fit, choose a space-filling design. Model type has no effect on designs that are space-filling or classical, so if you want to create these designs you can leave the model type at the default and open the Design Editor.

You can invoke the Design Editor in several ways from the "Test Plan Level":

1 First you must select the stage (first/local or second/global) for which you want to design an experiment. Click to select the appropriate model block in the test plan diagram.

2 Right-click the model block and select **Design Experiment**.

Alternatively, click the Design Experiment toolbar icon .

You can also select **TestPlan –> Design Experiment**.

For an existing design, **View –> Design Data** also launches the Design Editor (also in the right-click menu on each Model block). This shows the selected data as a design.

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.

Classical designs (including full factorial) are very well researched and are suitable for simple regions (hypercube or sphere). See "Creating a Classical Design" on page 5-15.

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 a Space-Filling Design" on page 5-18.

Optimal designs are best for cases with high system knowledge, where previous studies have given confidence in the best type of model to be fitted, and the constraints of the system are well understood. See "Creating an Optimal Design" on page 5-24.

You can augment any design by optimally adding points. Working in this way allows new experiments to enhance the original, rather than simply being a second attempt to gain the necessary knowledge. See "Adding Design Points" on page 5-34.

Design Editor Displays

The following example shows the display after creating an optimal design.
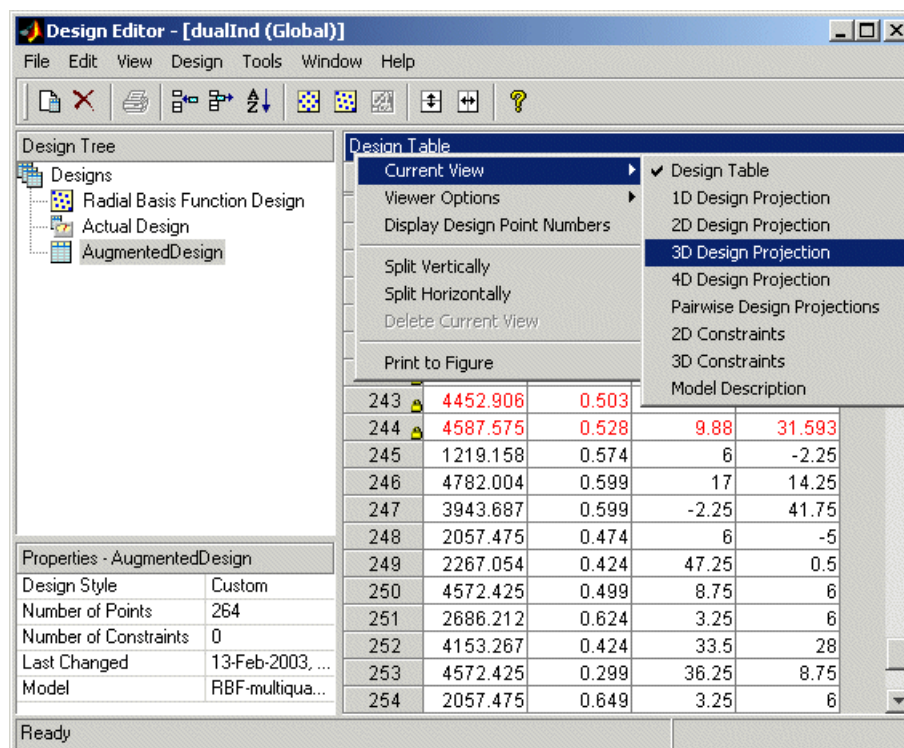
When you first create or open a design, the main display area shows the default **Design Table** view of the design (see example above). All the views on the right show the design selected in the left tree (see "The Design Tree" on page 5-8). There is a context menu for the views on the right, available by right-clicking the title bars, in which you can change the view of the design to **1-D**, **2-D**, **3-D**, **4-D**, and **Pairwise Projections**, **2-D**, and **3-D Constraint** views, and the **Table** view (also under **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. The split can also be merged again. You can also use the toolbar buttons. 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 current active

view. See "Design Editor Toolbar and Menus" on page 5-9 for more information about how to change your display options.

The information pane, bottom left, displays pieces of information for the current design selected in the tree. The amount of information in this pane can change depending on what the design is capable of; for example, only certain models can support the optimal designs and only these can show current optimal values. You can also see this information and more by selecting **File –> Properties** or using the context menu in the tree.

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, 3-D, or pairwise plot of the current design points as the main plot. The following example shows several views in use at once.
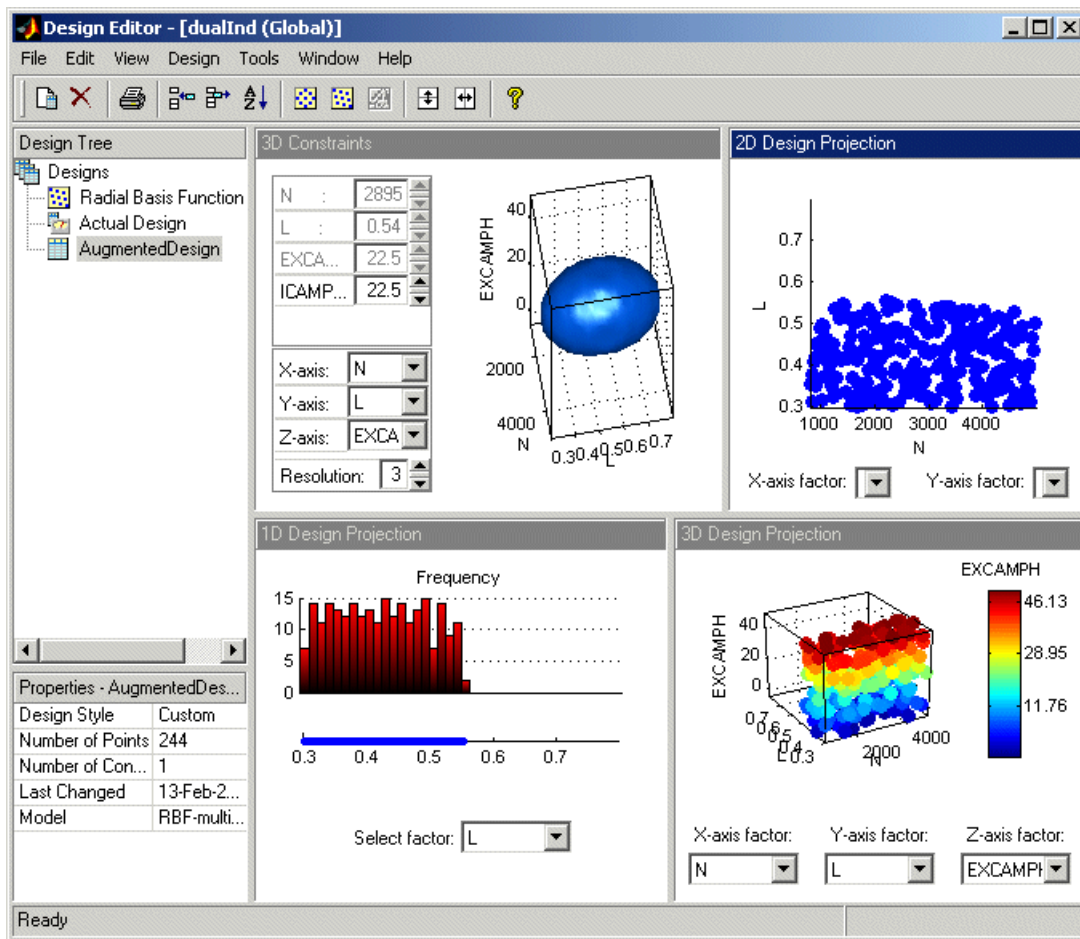
The Design Tree

The currently available designs are displayed on the left in a tree structure.

The tree displays three pieces of information:

- The name of the design, which you can edit by clicking it
- The state of the design
  - The icon changes from ⬚ if it is empty, to the appropriate icon for the design type when it has design points (for example, ▦ optimized, as in the toolbar buttons for Optimal, Classical, and Space-Filling designs).
  - The icon changes to ▦ when design points have been added using a different method (for example, augmenting a classical design with optimally chosen points). It becomes a *custom* design style. You can mix and match all design options in this way.
  - A padlock appears (▦) if the design is locked. This happens when it has child nodes (to maintain the relationship between designs, so you can retreat back up the design tree to reverse changes).
- The design that is selected as best. This is the design that is used for matching against experimental data. The icon for the selected design is the normal icon turned blue. When you have created more than one design, you must select as best the design to be used in modeling, using the **Edit** menu. Blue icons are also locked designs, and do not acquire padlocks when they have child nodes.
- You can reach a context menu by right-clicking in the design tree pane. Here you can delete or rename designs and add new designs. Choose **Evaluate Design** to open the Design Evaluation window. **Properties** opens the **Design Properties** dialog, which displays information about the size, constraints, properties (such as optimality values), and modification dates of the selected design.

Design Editor Toolbar and Menus
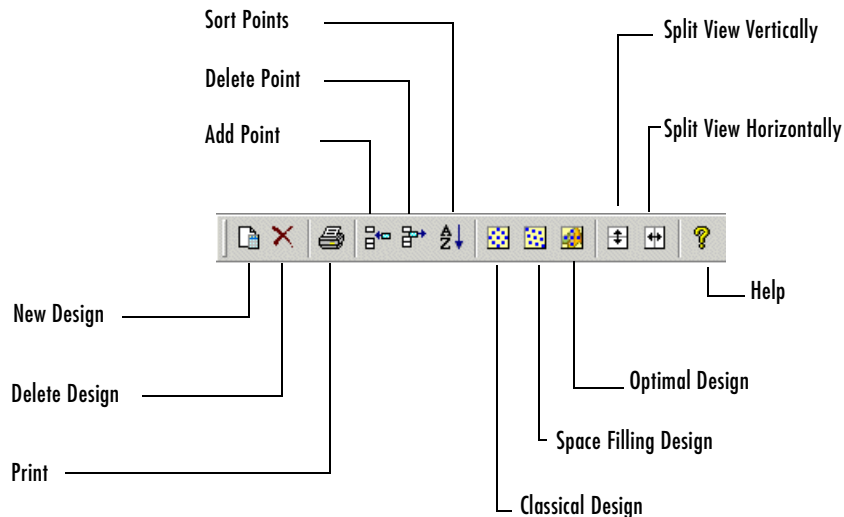
- **New Design**, **Delete Design**, **Print** — See the "File Menu" on page 5-9.
- **Add, Delete**, and **Sort Point** — See the "Edit Menu" on page 5-11.
- **Classical, Space Filling**, and **Optimal Design** — See the "Design Menu" on page 5-13.
- **Split View Horizontally** and **Vertically**— See the "View Menu" on page 5-12.

File Menu

- **New Design** — Creates a new design node in the tree. This is blank if it is the first design you create, or, if you have an existing design, the child node is a copy of the parent design. Also in the toolbar and the context menu in the design tree. Use the **Design** menu or toolbar buttons to set up designs.
- **Delete Design** — Deletes the currently selected design and its subdesigns. Also in the toolbar and the context menu in the design tree.

- **Rename Design** — Enables you to edit the name of the current design. You can also do this by clicking again on the selected design name, or right-click to use the context menu in the design tree.

- **Properties** — Opens the **Design Properties** dialog, where you can see information about your current design, such as the number of factors, points, and constraints; the design style; when the design was last modified; optimality values, space filling, and classical settings. Also in the context menu in the design tree.

- **Import Design** — Opens the **Import Design** dialog, where you can import designs from Design Editor files (`*.mvd`), comma-separated-values files (`*.csv`), or from the workspace. You can browse to the required file or specify the source variable if importing from the workspace. If it is not a Design Editor file you can choose to convert the design points from [-1, 1] range. See "Exporting and Importing Designs" on page 5-41.

- **Export Design** — Opens the **Export Design** dialog, where you can export designs to Design Editor files (`*.mvd`), comma-separated-values files (`*.csv`), or from the workspace. You can specify the name of the destination file or variable. For `.csv` files you can use the check box to choose whether to include factor names, and for the workspace and `.csv` files you can choose whether to convert design points to [-1, 1] range. See "Exporting and Importing Designs" on page 5-41. Note that you do not have to export your designs to save them — they are saved when you save your project in the Model Browser.

- **Merge Designs** — Opens the **Merge Designs** dialog, where you can choose which designs to merge and a base design. See "Merging Designs" on page 5-37.

- **Import Constraints** — Opens the **Import Constraints** dialog. Here you can import any suitable constraints for the currently selected design, from any existing constraints in the design tree, or you can import constraints from a design file. See "Importing Constraints" on page 5-50.

- **Print** — Prints the current view (plots only). See also **Print to Figure** in the **View** menu. If you want to print information from the Table view you can copy the information to the clipboard by using **Edit –> Copy Design Data**, or save the design as a `.csv` file. Also in the toolbar.

- **Close** — Closes the Design Editor. You return to the Model Browser window. Note that you do not lose your designs, you simply close the Design Editor.

The designs reappear when you reopen the Design Editor. When you save your project in the Model Browser your designs remain part of that project.

Edit Menu

- **Copy Current View** — Copies the current view to the clipboard.
- **Copy Design Data** — Copies the design data to the clipboard. This can be useful if you want to print the contents of the Table view.
- **Clear** — Deletes all points in the current design.
- **Add Point** — Opens the **Add Design Points** dialog. Here you can choose how many points to add optimally, randomly, or at specified points. Also in the toolbar. See "Adding Design Points" on page 5-34.
- **Delete Point** — Opens the **Delete Design Points** dialog. Here you can choose the points to delete. Also in the toolbar. See "Fixing, Deleting, and Sorting Design Points" on page 5-39.
- **Sort Points** — Opens the **Sort** dialog. Here you can choose to sort by any or all of your factors, by custom expression, or at random. Also in the toolbar. Se "Fixing, Deleting, and Sorting Design Points" on page 5-39.
- **Fix/Free Points** — Opens the **Fix Design Points** dialog. You can fix design points so they are not moved by design optimization processes. See "Fixing, Deleting, and Sorting Design Points" on page 5-39.
- **Randomize** — Select this option as a quick way of randomly resorting the points in the current design. This is a shortcut to the same functionality provided by the **Random** option in the **Sort** dialog.
- **Round Factor** — Opens the **Round Design** dialog. Here you can select a factor to round, and limit it to a specified number of significant figures or to a fixed interval.
- **Constraints** — Opens the **Constraints Manager** dialog. Here you can add, edit, duplicate, and delete constraints on your designs. See "Applying Constraints" on page 5-42.
- **Model** — Opens the **Global** or **Local Model Setup** dialog (depending on which stage you are designing an experiment for). Here you can change the model for which you are designing an experiment. The model type is important for optimal designs.

• **Select As Best** — Selects the current design as best. This changes the icon in the tree blue. This is the design that is used for matching against experimental data.

View Menu

• **Current View** — Changes the current view to your selection from the submenu:

- Design Table
- 1-D Design Projection
- 2-D Design Projection
- 3-D Design Projection
- 4-D Design Projection
- Pairwise Design Projections
- 2-D Constraints
- 3-D Constraints
- Model Description

• **Viewer Options**

- **General** — Opens dialogs for configuring details of the current display. You can change basic properties such as color on the projections (1-D, 2-D, 3-D, and 4-D). You can rotate all 3-D views as usual. You can double-click the color bar to edit the colormap.

- **Value Filter** — For the table view, you can set up a filter to selectively display certain ranges of values.

• **Display Design Point Numbers** — You can select this option to toggle the display of design point numbers in views that support the feature. A design point number is the index of a particular point in the design: this value is permanently displayed in the table view. Views that support the display of design point numbers are

- 2-D Design Projection
- 3-D Design Projection
- 4-D Design Projection

- Pairwise Design Projections

Because all these views are projections that use a subset of the design's input factors, it is often the case that the resulting view contains points that have been plotted on top of each other. In this case, the design point numbers will stack up in a column above the common point to aid readability. You can use **Display Design Point Count** to see at a glance how many points are overlapping in any stack. You can select point count or point numbers but not both.

Note Displaying multiple views with design point numbers for large designs can significantly slow down the display. You might want to turn off the design point number display in these cases.

- **Display Design Point Count** — If points are plotted on top of each other (in 2-D, 3-D, 4-D, or pairwise plots) this option allows you to see how many points are overlapping in each cluster. A number next to a point indicates that more than one point is plotted there.
- **Split Vertically** — Splits the current view vertically to produce a new view in addition to the currently selected view. Also in the toolbar. New plot types are produced for each new view. This is a quick way to produce a variety of different plots. Remember that you can change any existing view type by selecting **View –> Current View** or, alternatively, using the context menu to select from the **Current View** submenu.
- **Split Horizontally** — Splits the current view horizontally to produce new views, as for **Split Vertically**.
- **Delete Current View** — Deletes the current view.
- **Print to Figure** — This option copies the current view into its own figure, allowing you to use the standard MATLAB plotting tools to annotate and print the display.

Design Menu

- **Optimal** — Opens the **Optimal Design** dialog. Also in the toolbar. See "Creating an Optimal Design" on page 5-24.
- **Classical** — You can use the submenu here to go directly to the type of classical design you want, or select **Design Browser** to see all the options.

See "Creating a Classical Design" on page 5-15. The toolbar button opens the Design Browser.

- **Space Filling** — You can use this submenu to go directly to the type of space filling design you want, or select **Design Browser** to see all the options. See "Creating a Space-Filling Design" on page 5-18. The toolbar button opens the Design Browser.

Tools Menu

- **PEV Viewer** — Opens the **Prediction Error Variance Viewer** where you can evaluate the predictive power of your designs. See "Prediction Error Variance Viewer" on page 5-52.
- **Evaluate Designs** — Opens the Design Evaluation window where you can examine detailed mathematical properties of your design. Also in the context menu in the design tree. See "Design Evaluation Tool" on page 8-30.

Window Menu

This allows you to switch between the **Model Browser** and **Design Editor** windows.

Help Menu

As with everywhere in the toolbox, the **Help** menu provides access to general toolbox help and help specific to the current view. Here you can select **MBC Help** to browse all the toolbox help, **Design Editor Help** to go straight to the Designs documentation, or **About MBC** to see the current version number.

Creating a Classical Design

1 Add a new design by clicking the ⬜ button in the toolbar or select **File –> New**.

2 Select the new design node in the tree. An empty Design Table appears if you have not yet chosen a design. Otherwise if this is a new child node the display remains the same, because child nodes inherit all the parent design's properties. 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.

3 Click the 🔲 button in the toolbar or select **Design –> Classical –> Design Browser**.

Note In cases where the preferred type of classical design is known, you can go straight to one of the five options under **Design –> Classical**. Choosing the **Design Browser** option allows you to see graphical previews of these same five options before making a choice.

• A dialog appears if 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. The default is replacement of the current points with a new design. Click **OK** to proceed, or **Cancel** to change your mind.

The **Classical Design Browser** appears.
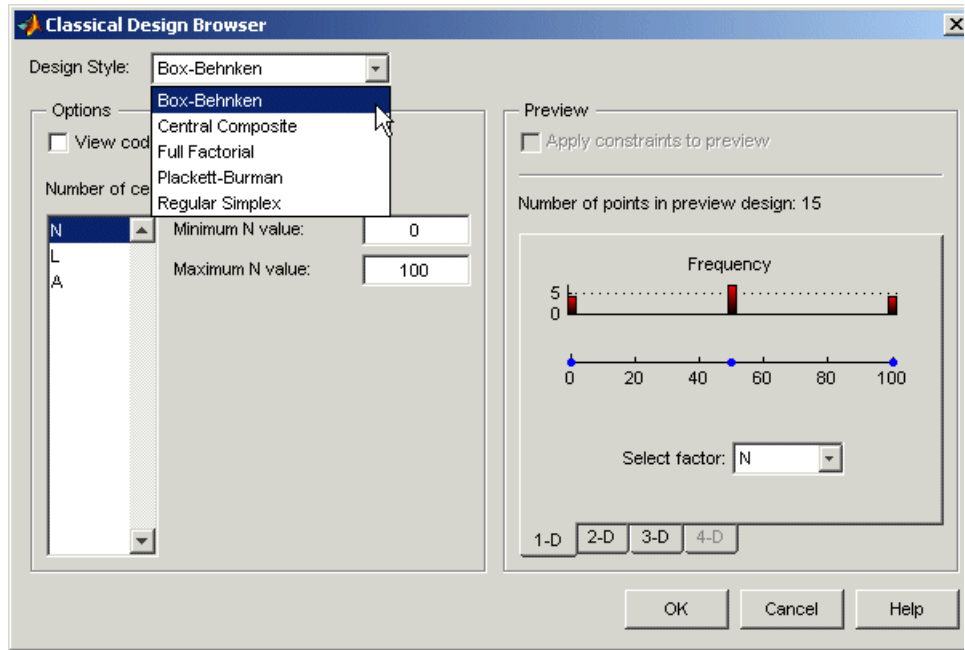
Classical Design Browser

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. The options are **Face-center cube**, **Spherical**, **Rotatable**, or **Custom**. If you choose **Custom**, you can then choose a ratio value (α) between the corner points and the face points for each factor and the number of center points to add. Five levels for each factor are used. You can set the ranges for each factor. **Inscribe star points** scales all points within the coded values of 1 and -1 (instead of plus or minus α outside that range). When this box is not selected, the points are circumscribed.

- **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 you should choose 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. You can set the ranges of each factor.

- **Full Factorial**

  Generates an *n*-dimensional grid of points. You can choose the number of levels for each factor, the number of additional center points to add, and the ranges for each factor.

- **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. They are constructed from Hadamard matrices and are a class of two-level orthogonal array.

- **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.

You can always toggle coded values by selecting the check box at the top.

Creating a Space-Filling Design

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

1 Add a new design by clicking the ⬚ button in the toolbar or select **File** –> **New**.

2 Select the node in the tree by clicking. An empty Design Table appears if you have not yet chosen a design. Otherwise, if this is a new child node the display remains the same, because child nodes inherit all the parent design's properties.

3 Select **Design** –> **Space Filling** –> **Design Browser**, or click the ▦ **Space Filling Design** button on the toolbar.

4 A dialog appears if 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. The default is replacement of the current points with a new design. Click **OK** to proceed, or **Cancel** to change your mind.

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.
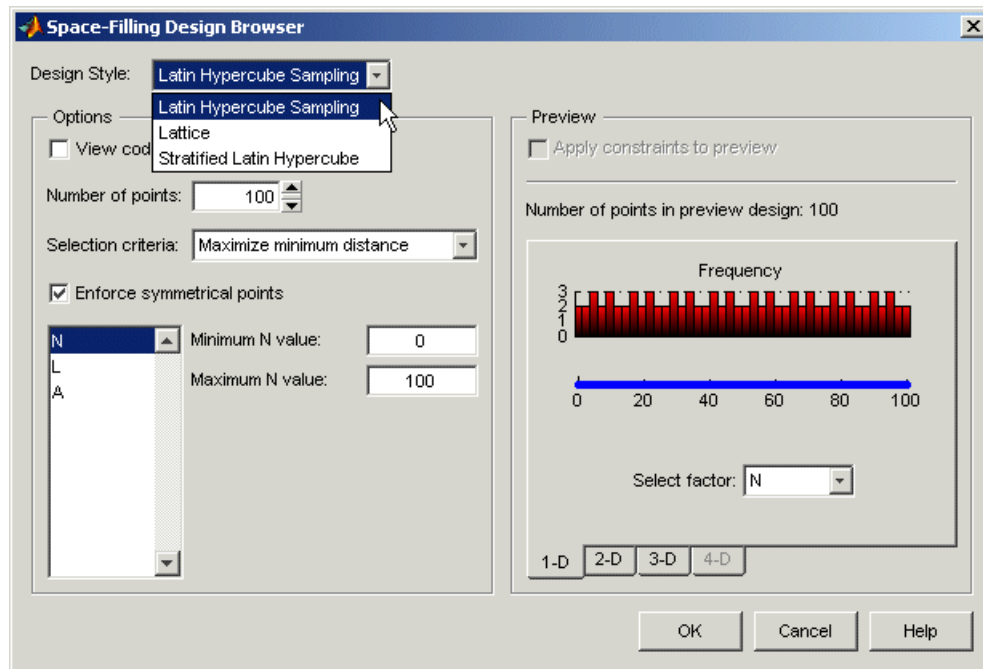
Space-Filling Design Styles

In the **Design** drop-down menu you can see the three design styles available:

- **Latin Hypercube Sampling**
- **Lattice**
- **Stratified Latin Hypercube**

Latin Hypercube Sampling

Latin Hypercube Sampling (LHS) designs are sets of design points that, for an N point design, project onto N different levels in each factor. Here the points are generated randomly. You choose a particular Latin Hypercube by trying several such sets of randomly generated points and choosing the one that best satisfies user-specified criteria.

Lattice

Lattice designs project onto N different levels per factor for N points. The points are not randomly generated but are produced by an algorithm that uses a prime number per factor. If good prime numbers are chosen, the lattice spreads points evenly throughout the design volume. A poor choice of prime numbers results in highly visible lines or planes in the design projections. If all the design points are clustered into one or two planes, it is likely that you cannot estimate all the effects in a more complex model. When design points are projected onto any axes, there are a large number of factor levels.

For a small number of trials (relative to the number of factors) LHS designs are preferred to Lattice designs. This is because of the way Lattice designs are generated. Lattice designs use prime numbers to generate each successive sampling for each factor in a different place. No two factors can have the same generator, because in such cases the lattice points all fall on the main diagonal of that particular pairwise projection, creating the visible lines or planes described above. When the number of points is small relative to the number of factors, the choice of generators is restricted and this can lead to Lattice designs with poor projection properties in some pairwise dimensions, in which the points lie on diagonals or double or triple diagonals. This means that Latin Hypercube designs are a better choice for these cases.

Stratified Latin Hypercube

Stratified Latin Hypercubes separate the normal hypercube into N different levels on user-specified factors. This can be useful for situations where the preferred number of levels for certain factors might be known; more detail might be required to model the behavior of some factors than others. They can also be useful when certain factors can only be run at given levels.
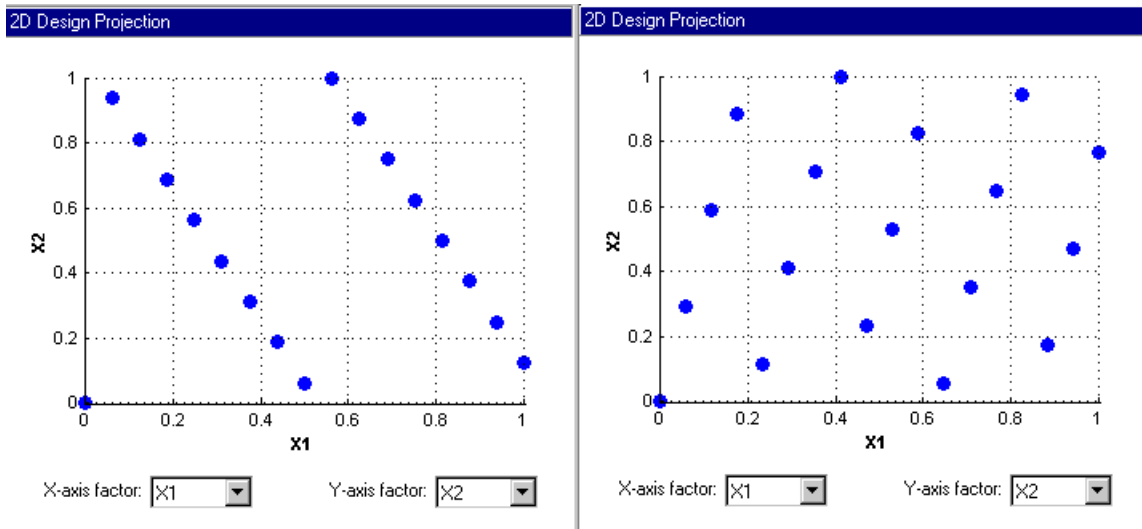
The preceding example shows the different properties of a poor lattice (left) and a good lattice (right), with a similar number of points. The poorly chosen prime number produces highly visible planes and does not cover the space well. An example of an LHS design of the same size is shown for comparison.
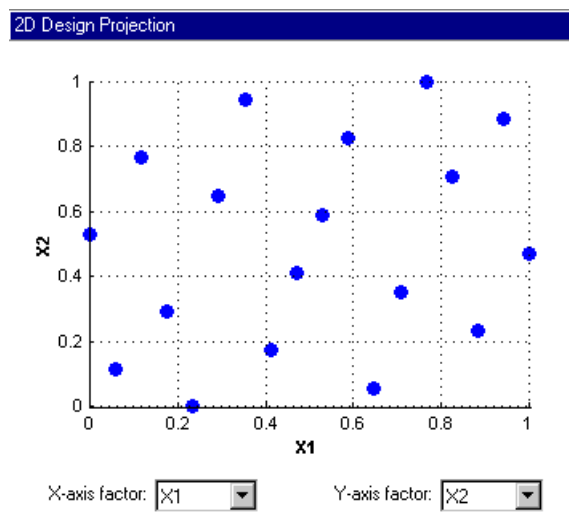
Setting Up a Space-Filling Design

You can toggle coded units for all space-filling designs by selecting the check box at the top.

The default **Design** drop-down menu item is **Latin Hypercube Sampling**.

For both **Latin Hypercube Sampling** and **Stratified Latin Hypercube**, you can choose from several selection criteria available in a drop-down menu:

- **Maximize minimum distance** (between points). This is the default.
- **Minimize maximum distance** (between points)
- **Minimize discrepancy** — Minimizes the deviation from the average point density
- **Minimize RMS variation from CDF** — Minimizes the Root Mean Square (RMS) variation of the Cumulative Distribution Function (CDF) from the ideal CDF
- **Minimize maximum variation from CDF** — Minimizes the maximum variation of the CDF from the ideal CDF

The final two (CDF variation) options scale best with the number of points and it is advisable to choose one of these options for large designs.

The same criteria are available for the **Stratified Latin Hypercube**.

- You can set the number of points using the controls or by typing in the edit box.
- You can set the ranges for each factor.
- If you select the **Enforce Symmetrical Points** check box, you create 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.
- You can use the tabs under the display to view 2-D, 3-D, and 4-D previews.
- Click **OK** to calculate the Latin Hypercube and return to the main Design Editor.

For a **Lattice** space-filling design:

- You can choose the lattice size by using the buttons or typing in the edit box.

- You can choose the prime number generator by using the up/down buttons on the **Prime number for X** edit box.
- You can choose the range for each factor.
- Click **OK** to calculate the lattice and return to the Design Editor.

Creating an Optimal Design

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. Optimal designs require linear models.

The Design Editor can average optimality across several linear models. This is a flexible way to design experiments using optimal designs. If you have no idea what model you are going to fit, you would choose a space-filling design. However, if you have some idea what to expect, but are not sure which model to use, you can specify a number of possible models. The Design Editor can average an optimal design across each model.

For example, if you expect a quadratic and cubic for three factors but are unsure about a third, you can specify several alternative polynomials. You can change the weighting of each model as you want (for example, 0.5 each for two models you think equally likely). This weighting is then taken into account in the optimization process in the Design Editor. See "Global Model Class: Multiple Linear Models" on page 7-55.

1 Click the ⬜ button in the toolbar or select **File –> New**. A new node appears in the design tree. It is named according to the model for which you are designing, for example, `Linear Model Design`.

2 Select the node in the tree by clicking. An empty Design Table appears if you have not yet chosen a design. Otherwise, if this is a new child node the display remains the same, because child nodes inherit all the parent design's properties.

3 Set up any constraints at this point. See "Applying Constraints" on page 5-42.

4 Choose an Optimal design by clicking the ⬛ button in the toolbar, or choose **Design –> Optimal**.

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 three tabs that contain the settings for three main aspects of the design:

- **Start Point** tab: Starting point and number of points in the design
- **Candidate Set** tab: Candidate set of points from which the design points are chosen
- **Algorithm** tab: Options for the algorithm that is used to generate the points

Optimal Design: Start Point Tab
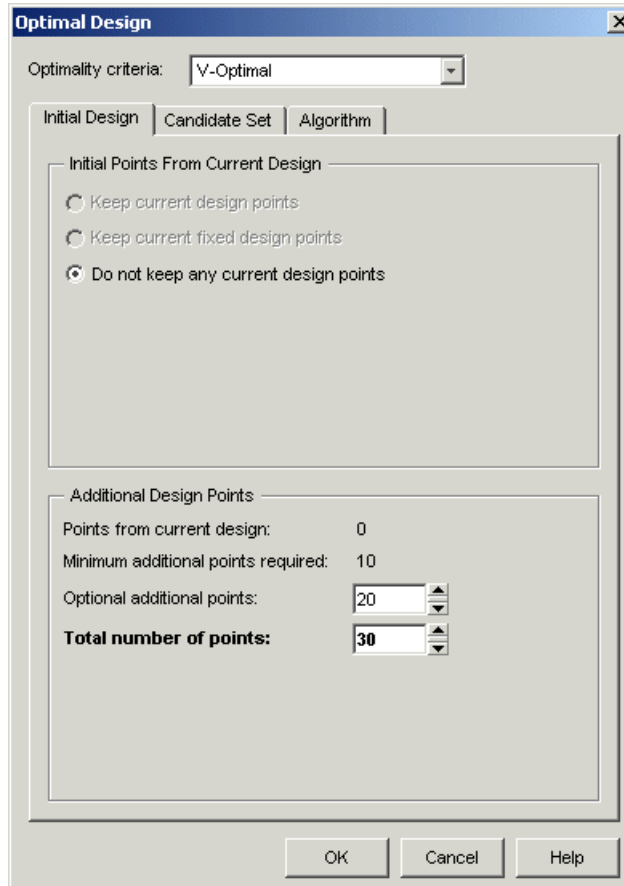
The **Start Point** tab allows you to define the composition of the initial design: how many points to keep from the current design and how many total or additional points to choose from the candidate set.

1 Choose the type of the optimal design, using the **Optimality criteria** drop-down menu:

- **D-Optimal** designs — Aims to reduce the volume of the confidence ellipsoid to obtain accurate coefficients. This is set up as a maximization problem, so the progress graph should go up with time.

  The D-optimality value used is calculated using the formula

  $$D_{eff} = \frac{\log_e(\det(X'X))}{k}$$

  where X is the regression matrix and k is the number of terms in the regression matrix.

- **V-Optimal** designs — Minimizes the average prediction error variance, to obtain accurate predictions. This is better for calibration modeling problems. This is a minimization process, so the progress graph should go down with time.

  The V-optimality value is calculated using the formula

  $$V_{eff} = \frac{1}{n_C}\sum_j x_j'(X_C'X_C)^{-1}x_j$$

  where x_j are rows in the regression matrix, X_C is the regression matrix for all candidate set points, and n_C is the number of candidate set points.

- **A-Optimal** designs — Minimizes the average variance of the parameters and reduces the asphericity of the confidence ellipsoid. The progress graph also goes down with this style of optimal design.

  The A-optimality value is calculated using the formula

  $$A_{eff} = \text{trace}((X'X)^{-1})$$

  where X is the regression matrix.

2 You might already have points in the design (if the new design node is a child node, it inherits all the properties of the parent design). If so, choose from the radio buttons:

  - **Keep current design points**
  - **Keep current fixed design points**
  - **Do not keep any current points**

3 You can choose the total number of points and/or the number of additional points to add by clicking the up/down buttons or by typing directly into the edit boxes for **Optional additional points** or **Total number of points**.

Optimal Design: Candidate Set Tab

The **Candidate Set** tab allows you to set up a candidate set of points for your optimal design. Candidate sets are a set of potential test points. This typically ranges from a few hundred points to several hundred thousand. The set generation schemes are as follows:

- **Grid** — Full factorial grids of points, with fully customizable levels.
- **Lattice** — These have the same definition as the space-filling design lattices, but are typically used with about 10,000 points. The advantage of a lattice is that the number of points does not increase as the number of factors increases; however, you do have to try different prime number generators to achieve a good lattice. See "Creating a Space-Filling Design" on page 5-18.
- **Grid/Lattice** — A hybrid set where the main factors are used to generate a lattice, which is then replicated over a small number of levels for the remaining factors.
- **Stratified Lattice** — Another method of using a lattice when some factors cannot be set to arbitrary values. Stratified lattices ensure that the required number of levels is present for the chosen factor. Note that you cannot set more than one factor to stratify to the same N levels.
- **User-defined** — Import custom matrices of points from MATLAB or MAT-files.

For each factor you can define the range and number of different levels within that range to select points.

1 Choose a type of generation algorithm from the drop-down menu. Note that you could choose different parameters for different factors (within an overall scheme such as **Grid**).

2 This tab also has buttons for creating plots of the candidate sets. Try them to preview your candidate set settings. If you have created a custom candidate set you can check it here. The edit box sets the maximum number of points that will be plotted in the preview windows. Candidate sets with many factors can quickly become very large, and attempting to display the

entire set will take too long. If the candidate set has more points than you set as a maximum, only every `Nth` point is displayed, where `N` is chosen such that (a) the total displayed is less than the maximum and (b) `N` is prime. If you think that the candidate set preview is not displaying an adequate representation of your settings, try increasing the maximum number of points displayed.

3 Notice that you can see 1-D, 2-D, 3-D, and 4-D displays (fourth factor is color) all at the same time as they appear in separate windows (see the example following). Move the display windows (click and drag the title bars) so you can see them while changing the number of levels for the different factors.

4 You can change the factor ranges and the number of levels using the edit boxes or buttons.

Select variables in this list ⎯⎯⎯⎯

Choose algorithm type from this list ⎯⎯⎯⎯

Open display windows with these buttons⎯⎯⎯

Change the number of levels of the selected variable here⎯⎯⎯

Optimal Design: Algorithm tab

The **Algorithm** tab has the following algorithm details:

- **Augmentation method** - **Random/Optimal** — Optimal can be very slow (searches the entire candidate set for points) but converges using fewer iterations. Random is much faster per iteration, but requires a larger number of iterations. The Random setting does also have the ability to lower the optimal criteria further when the Optimal setting has found a local minimum.

- **Deletion method** - **Random/Optimal** — Optimal deletion is much faster than augmentation, because only the design points are searched.

- **p value** — The number of points added/removed per iteration. For optimal augmentation this is best kept smaller (~5); for optimal deletion only it is best to set it larger.

- **Delta** — This is the size of change below which changes in the optimality criteria are considered to be not significant.

- **q value** — Number of consecutive iterations to allow that do not increase the optimality of the design. This only has an effect if random augmentation or deletion is chosen.

- **Maximum number of iterations to perform** — Overall maximum number of iterations.
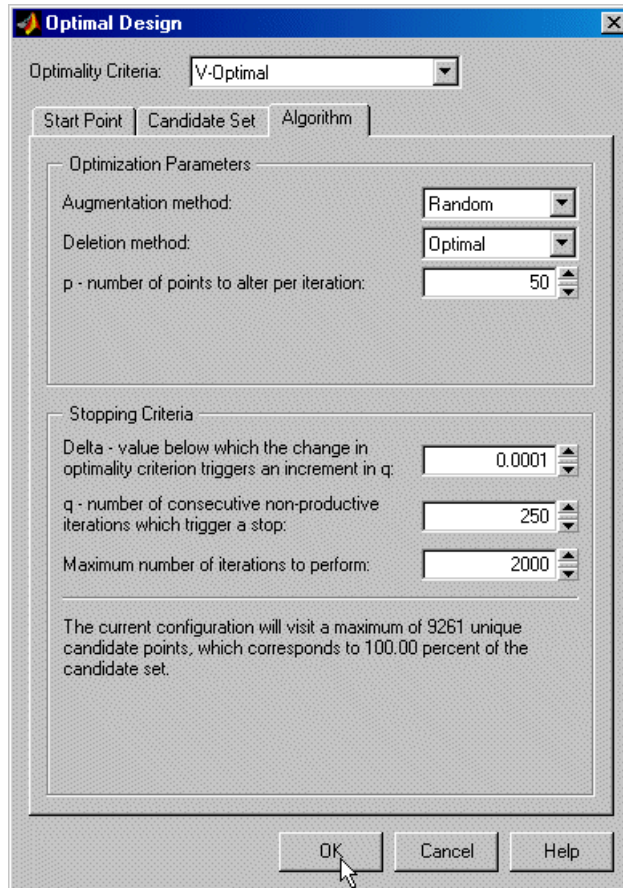
1 Choose the augmentation and deletion methods from the drop-down menus (or leave at the defaults).

2 You can alter the other parameters by using the buttons or typing directly in the edit boxes.

3 Click **OK** to start optimizing the design.

When you click the **OK** button on the **Optimal Design** dialog, another window appears that contains a graph. This window 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.

4 You can click **Accept** at any time, but it is most useful to wait until
   iterations are not producing noticeable improvements; that is, the graph
   becomes very flat.
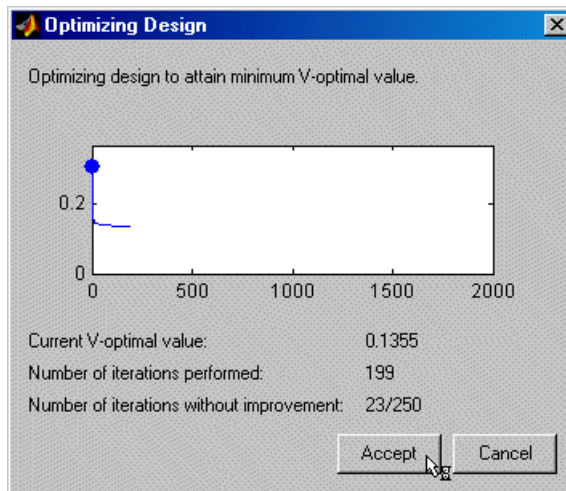
You can always return to the **Optimal Design** dialog (following the same steps)
and choose to keep the current points while adding more.

Adding Design Points

In any design, you can add points using the **Edit** menu. You can specify how many points to add and how to do so: optimally, randomly, or at specified values.

1 Select **Edit** –> **Add Point** or click the ⬚ button. A dialog appears, as shown following.

2 Choose an augmentation method from the drop-down menu: optimal (D,V, or A), random, or user-specified.

Note You can add points optimally to any design based on a linear or multilinear model, as long as it has the minimum number of points required to fit that model. This means that after adding a constraint you might remove so many points that a subsequent replace operation does not allow optimal addition.

---

3 Choose the number of points to add, using the buttons or typing into the edit box. For user-specified custom points, you also enter the values of each factor for each point you want to add.

4 If you choose an optimal augmentation method and click **Edit**, the **Candidate Set** dialog appears, as shown in the following example. Here you can edit the ranges and levels of each factor and which generation algorithm to use. These are the same controls you see on the **Candidate Set** tab of the **Optimal Design** dialog.
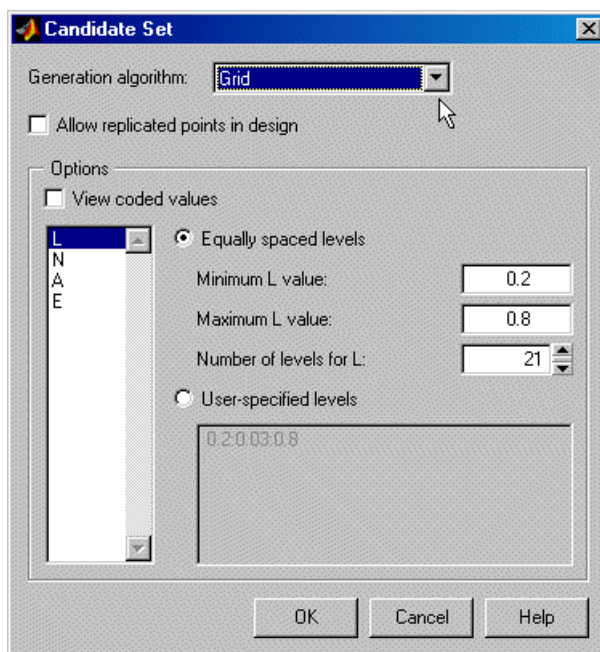
5 Click **OK** to add the points and return to the main display.
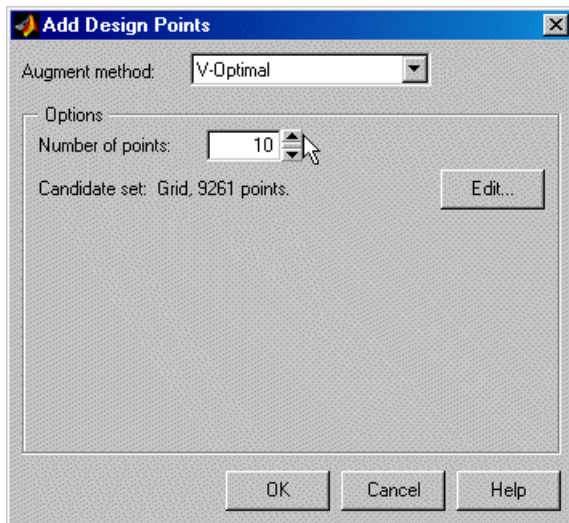
Merging Designs

You can merge the points from two or more designs together using the **File** menu. You can merge designs together to form a new design, or merge points into one of the chosen designs. Points that are merged retain their fixed status in the new design.

1 Select **File –> Merge Designs**. A dialog appears, as shown.
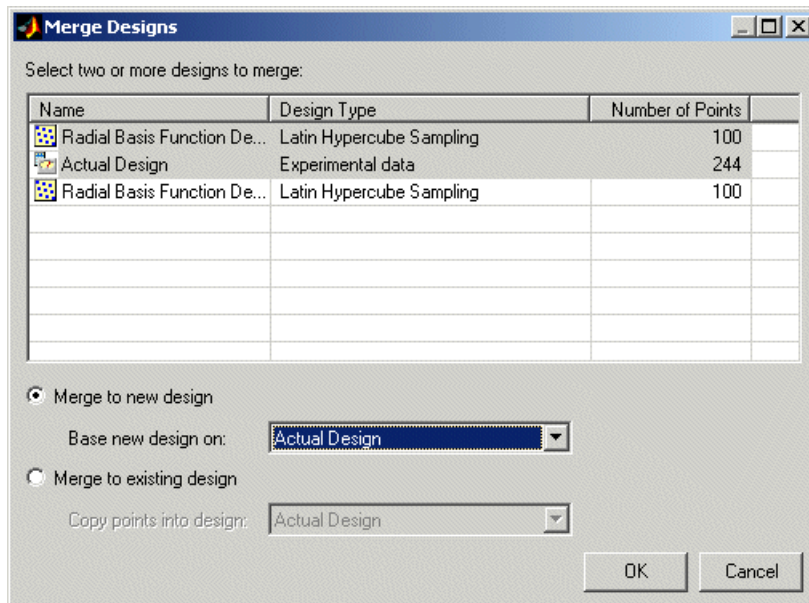
2 A list of all the designs from the Design Editor is shown, along with the associated design style and number of points. Select two or more designs from the list by dragging with the mouse or **Ctrl**+clicking.

3 When at least two designs have been selected, the options at the bottom of the dialog are enabled. Choose whether you want to create a new design or put the design points into an existing design.

4 If you choose to create a new design, you must also choose one of the selected designs to act as a base. Properties such as the model, constraints, and any optimal design setup are copied from this base design. If you choose to reuse

an existing design you must choose one of the selected designs to receive the points from other designs.

5 Click **OK** to perform the merging process and return to the main display. If you choose to create a new design, it appears at the end of the design tree.

Fixing, Deleting, and Sorting Design Points

You can fix or delete points using the **Edit** menu. You can also sort points or clear all points from the design.

Fixed points become red in the main Design Editor table display. If you have matched data to a design or used experimental data as design points, those points are automatically fixed. You already have the data points, so you do not want them to be changed or deleted. Design points that have been matched to collected data are also fixed. Since these points have already been run they cannot be freed — you will not see them in the **Fix Design Points** dialog. Once you have fixed points, they are not moved by design optimization processes. This automatic fixing makes it easy for you to optimally augment the fixed design points.

1 Select **Edit –> Fix/Free Points** or **Edit –> Delete Point** (there is also a toolbar button, 🖫).

   A dialog appears in which you can choose the points to fix or delete.
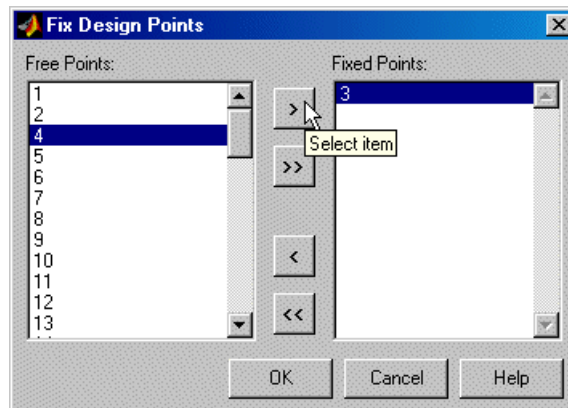
The example above shows the dialog for fixing or freeing points. The dialog for deleting points has the same controls for moving points between the **Keep Points** list and the **Delete Points** list.

2 In order to see where these points are in the design, you must change the main Design Editor display pane to the Table view. This gives a numbered list of every point in the design.

3 Move points from the **Free Points** list to the **Fixed Points** list; or from the **Keep Points** list to the **Delete Points** list, by using the buttons.

4 Click **OK** to complete the changes specified in the list boxes, or click **Cancel** to return to the unchanged design.

Design points that are matched to experimental data are set to fixed points so you can redesign unmatched points easily.

• Selecting **Edit –> Clear** deletes all points in the current design.

• **Edit –> Sort** opens a dialog (see example following) for sorting the current design — by ascending or descending factor values, randomly, or by a custom expression.
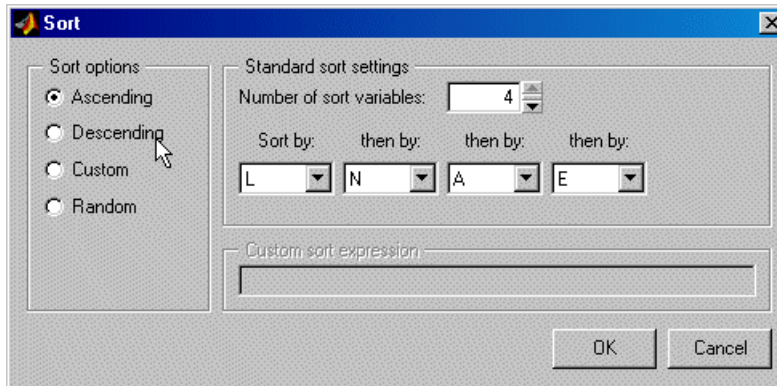
To sort by custom expression you can use MATLAB expressions (such as abs(N) for the absolute value of N) using the input symbols. Note that sorts are made using coded units (from -1 to 1) so remember that points in the center of your design space will be near zero in coded units, and those near the edge will tend to 1 or -1.

Select **Edit –> Randomize** as a quick way of randomly resorting the points in the current design. This is a shortcut to the same functionality provided by the **Random** option in the **Sort** dialog.

Exporting and Importing Designs

Note that you do not need to save your designs separately from the project. When you select **File –> Close** you do not lose your designs, you simply close the Design Editor. When you save your project in the Model Browser, your designs remain part of that project. You can also export designs to a file or the workspace.

You can export any design by choosing **File –> Export Design**. The selected design *only* is exported.

There are three options:

- **To File** generates a Design Editor file (`.mvd`).
- **To CSV File** exports the matrix of design points to a CSV (comma-separated-values) file. You can include factor symbols by selecting the check boxes.
- **To Workspace** exports the design matrix to the workspace. You can convert design points to a range of (1, -1) by selecting the check box.

You can choose the destination file or variable by typing in the edit box or using the browse button.

Import designs by selecting **File –> Import Design**. The controls on the dialog are very similar to the **Export Design** dialog: you can import from Design Editor files, CSV files, or the workspace, and you can convert design points from a (1,-1) range.

Applying Constraints

In many cases designs might not coincide with the operating region of the system to be tested. For example, an automobile engine normally does not operate in a region of low speed (n) and high exhaust gas recirculation (EGR). You cannot run 15% EGR at 1000 RPM. There is no point selecting design points in impractical regions, so you can constrain the candidate set for test point generation.

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 a design:

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

2 The **Constraints Manager** dialog appears.
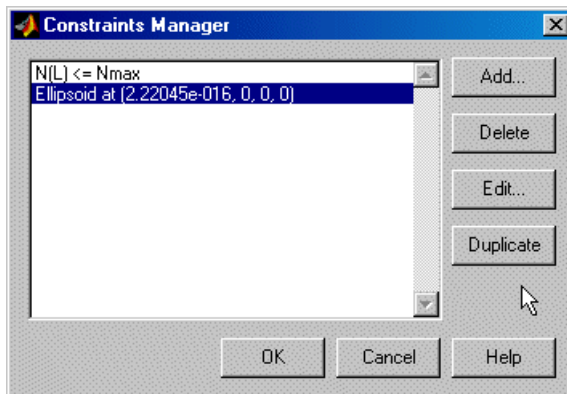
The example shows the **Constraints Manager** dialog with two constraints. Here you can add new constraints, and delete, edit, or duplicate existing constraints. If there are no constraints yet, the **Constraints Manager** is empty and you can only click **Add** to construct a new constraint.

3 Click **Add**.

4 The **Constraint Editor** dialog with available constraints appears. You can select **Linear, Ellipsoid**, **1-D Table**, or **2-D Table** from the **Constraint Type** drop-down menu, as shown.
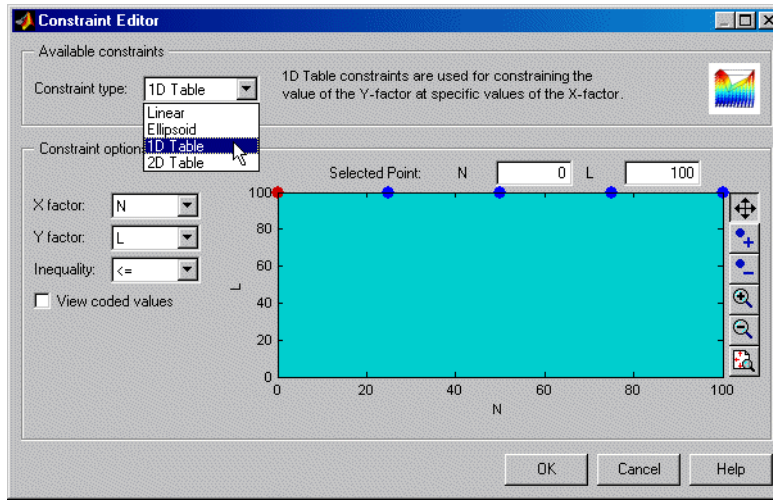
Constraint Types

Linear Constraints

You specify the coefficients of the equation for an (N-1) dimensional hyperplane in the N-factor space. The form of the equation is A.x = b where A is your defined coefficient vector, x is the vector of factor settings, and b is a scalar. The equation is applied by substituting design point settings (in coded values) for x. For example,

In two dimensions: A=(1, 2), x=(L, A), b=3

Then A.x = b expands to

$1*L + 2*A = 3$

Rearranging this, you can write it as

$A = -L/2 + 3/2$

which corresponds to the traditional equation of a 2-D straight line, y = mx + c, with m = -1/2 and c = 3/2. A.x = b is thus the higher dimensional extension of this equation.

The linear constraints work by selecting the region below the defined plane (that is, A.x <= b). To select a region above the plane, multiply all your values by -1: A -> -A, b -> -b.

For example, to select a simple plane where N<0.8 as a constraint boundary, enter 8 under N and set all the other factors to 0.

Ellipsoid Constraints

The ellipsoid constraint allows you to define an N-dimensional ellipsoid. You can specify the center of the ellipsoid, the length of each axis, and the rotation of the ellipsoid.

Ellipsoid center.   You specify the center of the ellipsoid by entering values in the column marked Xc. These are the values, in coded units, that mark where you want the ellipsoid to be centered in each of the factor dimensions.

Axis length.   You specify the size of the ellipsoid by entering values along the diagonal of the matrix to the right of Xc. The default values of 1 create an ellipsoid that touches the edge of the design space in each of the factor dimensions. Changing an entry to less than 1 extends the ellipsoid edge outside the design space along that factor axis (the extreme in this direction, 0, creates a cylinder). Changing an entry to greater than 1 contracts the ellipsoid edge to be inside the design space. In general, for an entry value X, the ellipsoid size in that factor is sqrt(1/X) times the size of the design space in that factor.

Rotation.   The matrix entries that are not on the main diagonal control rotation of the ellipsoid.

The following example shows a defined ellipsoid constraint.

- You must enter values in the table to define the ellipsoid. If you leave the values at the defaults, the candidate set is a sphere.

  If you change a 1 to a 3, you reduce that axis to $1/(\sqrt{3})$ times its original size. A value of 2 reduces that axis to $1/(\sqrt{2})$, and so on.

The example above reduces the space available for the candidate set by a third in the A and N axes, forming an ellipsoid, as shown. A 3-D display of this constraint can be seen below.
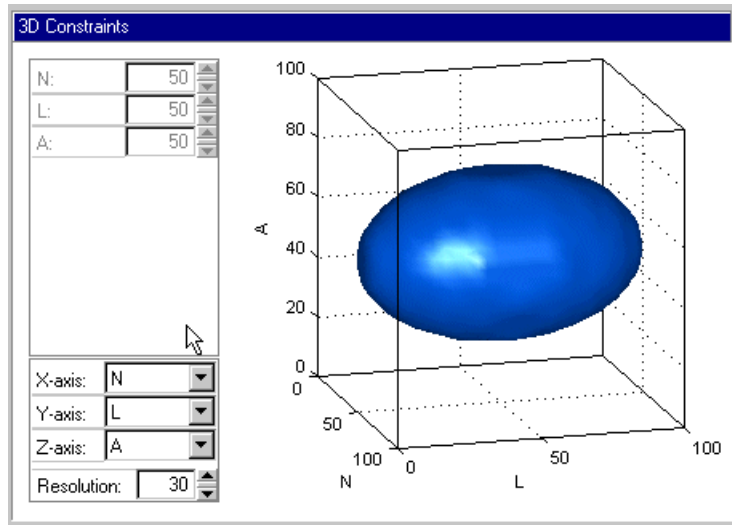
1-D Table Constraints

1-D table constraints limit the maximum or minimum setting of one factor as a function of another factor. Linear interpolation between user-defined points is used to specify the constraint.

- You can select the appropriate factors to use.

- Move the large dots (by clicking them and dragging) to define a boundary. You must click the dots *within* the boundary of the space to select them. The following example shows a 1-D Table constraint. You can choose whether to constrain your design above or below the defined boundary using the **Inequality** drop-down menu.
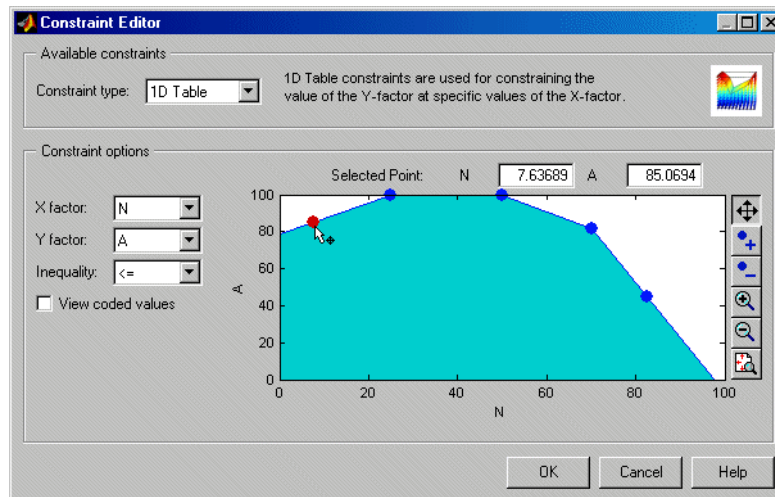
2-D Table Constraints

2-D table constraints are an extension of the 1-D table. Constraint boundary values for a factor are specified over a 2-D grid of two other factors.

- You can specify these grid locations by entering values in the top row and left column, while the matrix of values for the third factor is entered in the rest of the edit boxes. To specify grid values, you can enter values directly or just choose the number of breakpoints for your grid and space them over the factors' ranges, using the controls described below.

- You can specify the number of breakpoints for the X and Y factors by using the buttons or typing directly into the edit boxes.
- You can click **Span Range** to space your breakpoints evenly over the range of X or Y. This is useful if you add some breakpoints, as new points are often all at the maximum value for that factor. It is much easier to use the **Span Range** button than to change points manually.
- You can specify to keep the region below (<=) or above (>=) the constraint boundary, as for the 1-D table. Do this by choosing above or below from the **Inequality** drop-down menu for the Z factor.
- You can switch to coded values using the check box. See the example.

The constraint boundary between the defined grid points is calculated using bilinear interpolation.

- After defining any constraint, click **OK**. Your new constraint appears in the **Constraint Manager** list box. Click **OK** to return to the Design Editor, or **Add** to define more constraints.
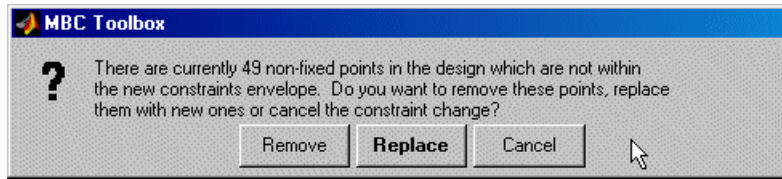
A dialog appears if there are points in the design that fall outside your newly constrained candidate set. You can simply continue (delete them) or cancel the constraint. Fixed points are *not* removed by this process. For optimal designs you can also replace them with new random points within the new candidate set, as shown in the preceding example dialog.

Note You only get the **Replace** points option for optimal designs. If you want to replace points removed by constraints from other designs, you can always use **Edit –> Add Point** to add points optimally, randomly, or at chosen places. However, if so many points have been removed by a constraint that there are not enough left to fit the current model, optimal addition is not possible. See "Adding Design Points" on page 5-34.

To view constraints,

1 Right-click the Design Editor display pane to reach the context menu.

2 Select **Current View –> 3D Constraints**. (You can also select 2-D constraints view). An example is shown.
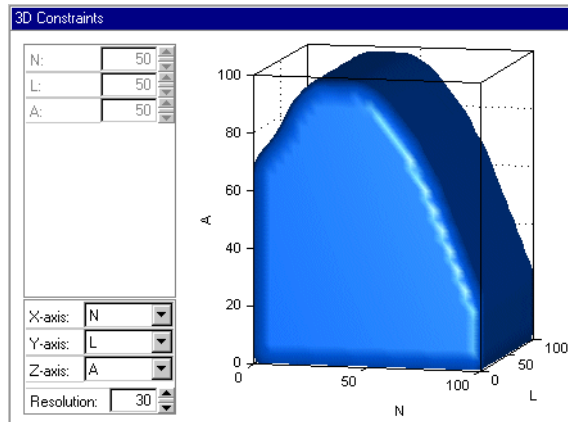
These views are intended to give some idea of the region of space that is currently available within the constraint boundaries.

Importing Constraints

Select **File –> Import Constraints**. The **Import Constraints** dialog appears, as shown in the following example.
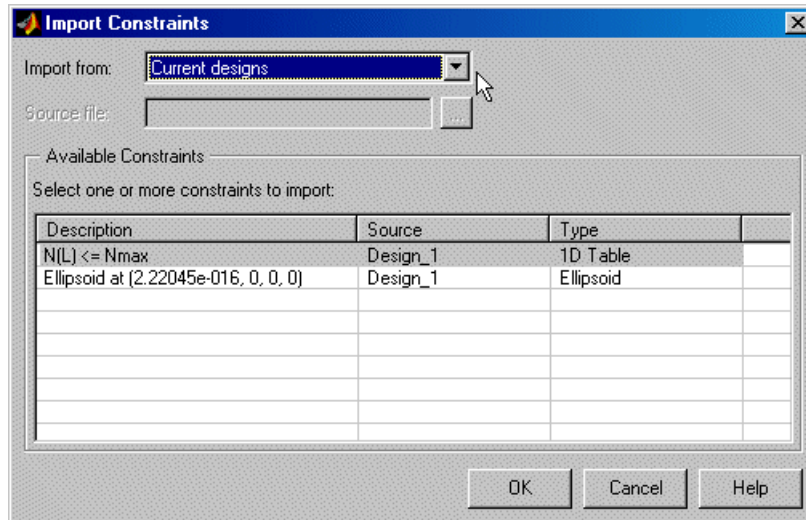
Here you can import constraints for the currently selected design from any existing constraints in the design tree.

Note You can only import constraints from designs that have the same number of factors and have the same coded range for each factor.

Select constraints in the list by clicking, or **Ctrl**+click to select multiple constraints.

You can choose **Design Editor file (.mvd)** from the **Import from** drop-down menu, and type the filename in the edit box or use the browse button to find another design file. In this way you can extract constraints from any other design file.

Prediction Error Variance Viewer

You can use the Prediction Error Variance (PEV) viewer to examine the quality of the model predictions. You can examine the properties of designs or global models. When you open it from the Design Editor, you can see how well the underlying model predicts over the design region. When you open it from a global model, you can view how well the current global model predicts. A low PEV (tending to zero) means that good predictions are obtained at that point.

The PEV viewer is only available for linear models and radial basis functions.

When designs are rank deficient, the PEV Viewer appears but is empty; that is, the PEV values cannot be evaluated because there are not enough points to fit the model.

- From the Design Editor, select **Tools –> PEV Viewer**.
- From the global level of the Model Browser, if the selected global model is linear or a radial basis function,
    - Click the ![icon] toolbar button to open the PEV viewer.
    - Alternatively, select **Model –> Utilities –> Prediction Error Surface**.

      If a model has child nodes you can only select the Prediction Error Surface from the child models.

Turn clipping on and off here   Apply boundary model clipping here   Change clipping value here
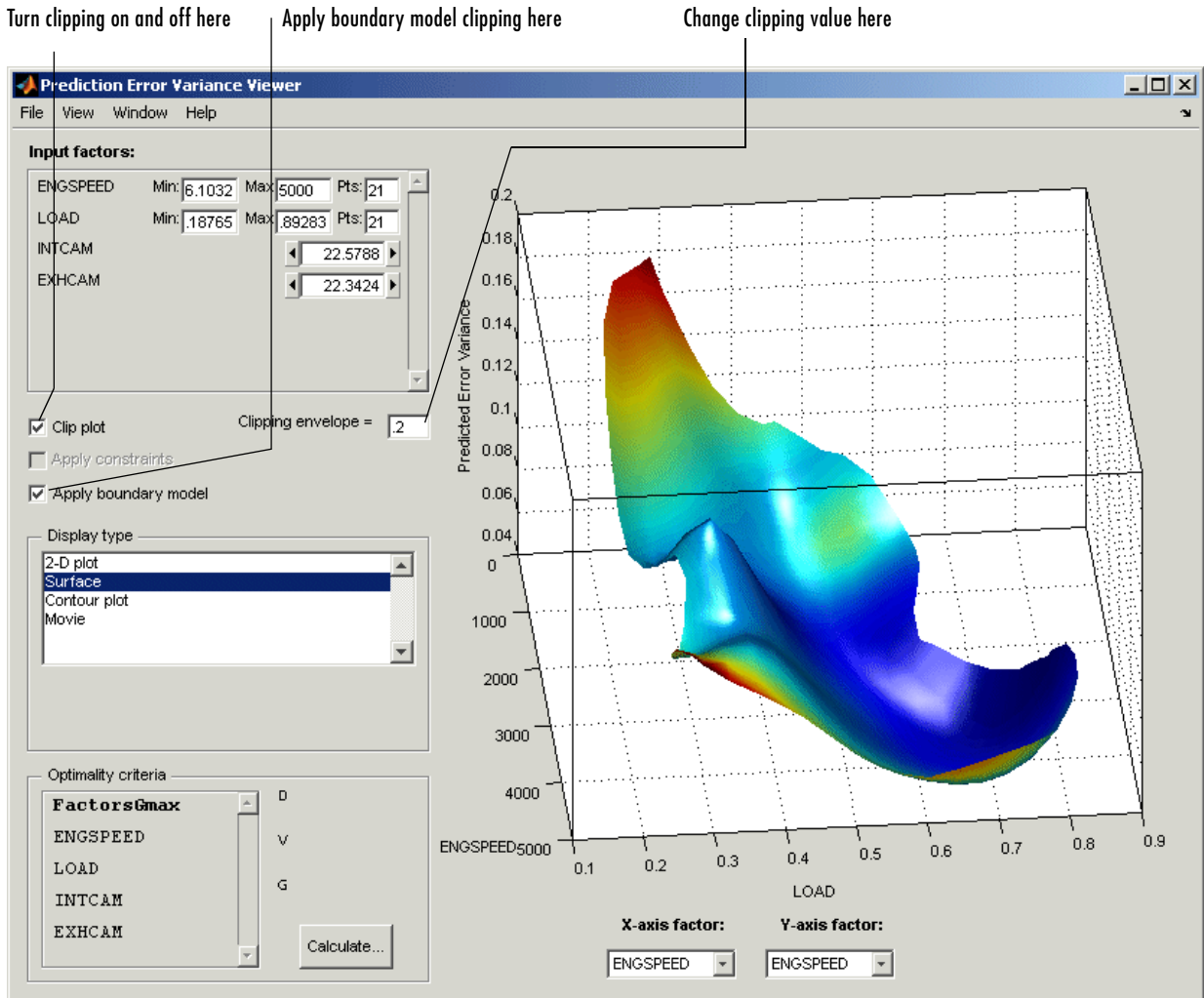
The default view is a 3-D plot of the PEV surface.

The plot shows where the model predictions are best. This example shows an MBT model response feature. The model predicts well where the PEV values are lowest.

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 drop-down 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 or edit boxes in the frame, 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. See the contour plot section (in "Response Surface View" on page 7-124) for a description of the controls.

- You can select the **Clip Plot** check box, as shown in the preceding example. Areas that move above the value in the **Clipping envelope** edit box are removed. You can enter the value for the clipping envelope; a white line appears on the plot where the PEV values pass through the clipping value.

- You can also clip with the boundary model or design constraints if available.

When you use the PEV viewer to see design properties, optimality values for the design appear in the **Optimality criteria** frame.

Note that you can choose Prediction Error shading in the Response Feature view (in Model Selection or Model Evaluation). This shades the model surface according to Prediction Error values (sqrt(PEV)). This is not the same as the PEV viewer, which shows the shape of a surface defined by the PEV values. See "Response Surface View" on page 7-124.

Optimality Criteria

No optimality values appear in the **Optimality criteria** frame until you click **Calculate**. Clicking **Calculate** opens the **Optimality Calculations** dialog. Here iterations of the optimization process are displayed.

In the **Optimality criteria** frame in the PEV viewer are listed the values of the input factors at the point of maximum PEV (**Gmax**). This is the point where the G optimality value is calculated. The D and V values are calculated for the entire design, not just at the point Gmax.

For statistical information about how PEV is calculated, see "Prediction Error Variance" on page 8-7 and "Prediction Error Variance for Two-Stage Models" on page 8-46 in the Technical Documents section.

6

Data

This section discusses the following topics:

Data

This section describes all aspects of loading, manipulating, and selecting data in the Model Browser. The Data Editor provides a powerful graphical interface for dealing with data:

- You use the **Data Editor** for loading, filtering, grouping, and editing data, and you can define new variables. You can match data to designs. You can reach the **Data Editor** from every node in the model tree, so you can also examine and export your modeling data. The **Data Editor** contains various graphical interfaces for these tasks:
  - **Data Import Wizard** for loading and merging data
  - **Variable Editor**, **Filter Editor**, **Test Filter Editor**, and **Test Notes Editor** are dialogs for creating and editing new variables and data filters.
  - **Storage** dialog for storing new variables, data filters, and plot settings
  - **Test Groupings** dialog can be used for plotting and manipulating data groups.
  - Within the **Data Editor** there are 2-D, 3-D, and multiple data plots for viewing data, and cluster plots for viewing data and design points.
  - You use the **cluster plots** views within the **Data Editor** for matching data to experimental designs. You can set tolerances for automatic selection of the nearest data points to the specified design points, or select data points manually.
- You use the **Data Wizard** to select data for modeling. You can also set up matching data to designs by setting tolerances and automatically opening the **cluster plots** views within the **Data Editor**. You reach the **Data Wizard** from test plan level.

You can load and merge data from the following:

- From files (Excel, Concerto, MATLAB)
- From the workspace

See "Data Loading and Merging" on page 6-10.

You can also write your own data-loading functions. See "Data Loading Application Programming Interface" on page 8-77.

Within the **Data Editor**, you can do the following:

- View plots, edit and add data records. See "Data Editor Views" on page 6-15.
- Define new variables. See "New Variables" on page 6-21.
- Apply filters to remove unwanted records, filter by test, and apply notes to tests that fulfill set criteria. See "New Filters" on page 6-23.
- Store and retrieve user-defined variables and filters. See "Storage" on page 6-25.
- Define test groupings to collect data into groups.
- Match data to experimental designs using the **cluster plots** views.
- Export data and modeling output to file and to the workspace.

You use the Data Wizard to do the following:

- Select the data set and design to use for modeling.
- Select the data signals to use for model input factors (one-stage, or local and global for two-stage).
- Select matching tolerances (if matching data to a design).
- Select data signals for response model input factors.

There is a tutorial to guide you through using the Data Editor. See Chapter 4, "Tutorial: Data Editor."

Tip for Fast Data Editor Access

When you open the Data Editor from the test plan level, a Cluster Plot is displayed. If you want to open the Data Editor frequently from the test plan level, it will save time if you previously had a Cluster Plot view open when you last used the Data Editor.

The Data Editor retains memory of the types of views open when you last used the Data Editor. If the previous views did not include a Cluster Plot view, then when you open the Data Editor from the test plan level the existing views are saved and replaced with a Cluster Plot view and a Cluster Information list view. The views revert to the previous views when you close the Data Editor. This process takes longer to display than if you had an existing Cluster Plot view (the Cluster Information list view is not required).

The Data Editor

To reach the Data Editor,

- From the test plan node, choose **TestPlan** –> **Select Data**.

- Alternatively, from the project node, do one of the following:
  - Choose **Data** –> **New Data**, **Copy Data**, or **Edit Data**.
  - Select any of the equivalent toolbar buttons.
  - Double-click a data set in the **Data Sets** pane.
- From any modeling node, click the **View Modeling Data** toolbar button.
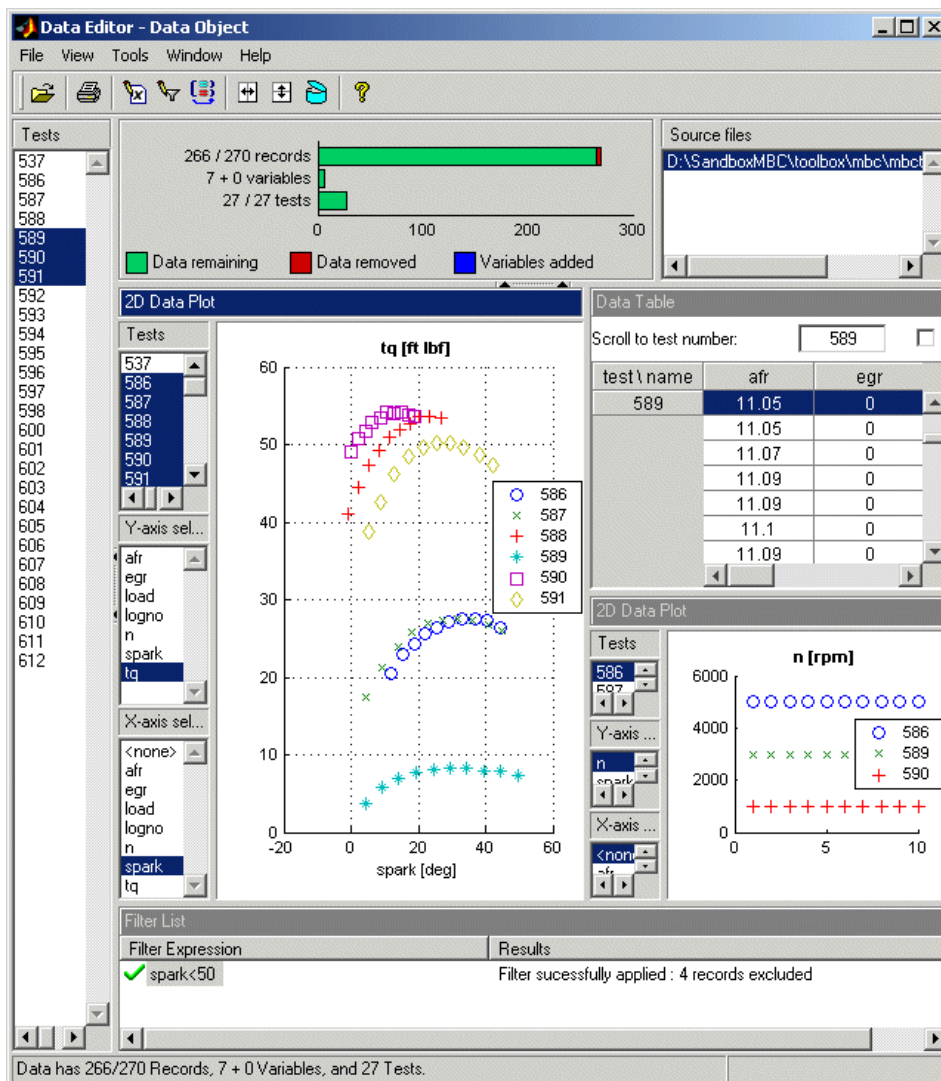
As can be seen in the preceding example, you can split the view to display several plots at once. As in the **Design Editor**, you use the right-click context menu to split the views, or use the toolbar buttons or the **View** menu. You can choose 2-D plots, 3D plots, multiple data plots, data tables, and list views of

filters, variables, test filters, and test notes. See "Data Editor Views" on page 6-15 for details on the functions available in each type of view.
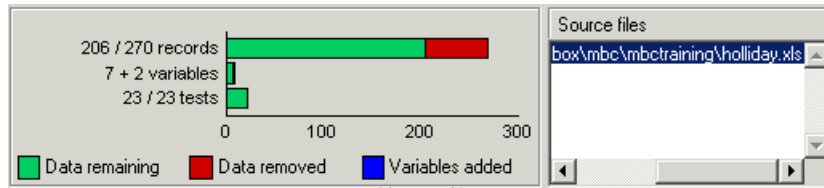
The list box at the top right contains the source file information for the data, and other information is displayed on the left: the numbers of records, variables, and tests it contains. See the preceding example. The bars and figures show the proportion of records removed by any filters applied, and the number of user-defined variables is shown. For this example with two user-defined variables added to a data set originally containing seven variables, you see '**7 + 2 variables**'.

The **Tests** list pane on the left is constant. Tests selected here apply to 3-D plots, multiple data plots, and tables. It does not apply to 2-D plots because they have their own independent test controls. If you are viewing read-only local modeling data, the selected test is shown in the **Tests** pane and remains synchronized if you change test in the Model Browser.

See "Data Editor Toolbar and Menus" on page 6-7 for other controls.

By default new data sets are called Data Object. You can change the names of data sets at the project node by select-clicking a data set in the **Data Sets** list, or by pressing **F2** (as when selecting to rename in Windows Explorer).

To edit data plot properties (2-D, 3-D, or multiple), you can right-click the plot and select **Viewer Options** –> **Properties**. Here you can choose to show the legend and the grid. You can choose the line style if you want to connect the data points and the data marker point style, if any. **Reorder X Data** (2-D plots) redraws the line joining the points in order from left to right. The line might not make sense when drawn in the order of the records.

Note Dismissing the Data Editor automatically brings up the Data Wizard if you entered it from the test plan level.

Data Editor Toolbar and Menus

Import Data

Edit Variables

Edit Filters

Change Test Grouping

Help

Split View Vertically

Split View Horizontally
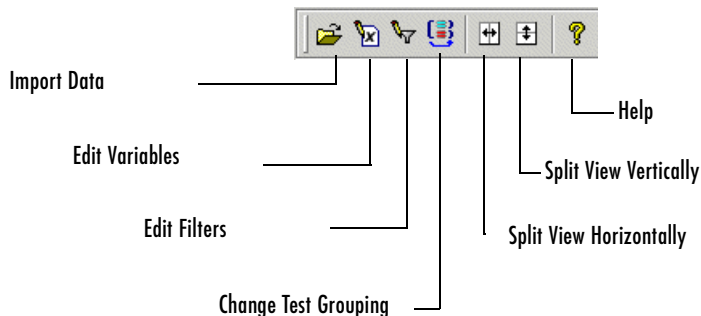
- **Import Data** — See "Loading Data from File" on page 6-10; also in the **File** menu.
- **Edit Variables** — Opens the **Variable Editor**.
- **Edit Filters** — Opens the **Filter Editor**.
- **Change Test Grouping** — Opens the **Test Groupings** dialog; also in the **Tools** menu.
- **Split View Horizontally** — Divides the currently selected plot into two plots side by side. Also in the **View** menu.
- **Split View Vertically** — Divides the currently selected plot into two plots one above the other. Also in the **View** menu.
- **Open Storage Window** — Opens **Storage** where you can store and retrieve variables, filters, and plot settings.

The **Window** and **Help** menus are the same as everywhere in the Model Browser. See "Window Menu" on page 7-10 and "Help Menu" on page 7-10.

File Menu

- **Import**
  - **File** — See "Loading Data from File" on page 6-10.
  - **Workspace** — See "Loading Data from the Workspace" on page 6-12.
  - **Excel** — Loads Excel files.

- **Export**
  - **Workspace** — Exports data to the workspace.
  - **Excel** — Exports data to an Excel file.
- **Print** — Prints your currently selected plot.
- **Close** — Closes the Data Editor.

View Menu

- **Current View**
  - Here you can choose from **2-D Data Plot, 3-D Data Plot, Multiple Data Plots, Data Table**, **Variable Definitions**, **Filter Definitions**, **Test Filter Definitions**, and **Test Note Definitions**, **Notes View**, **Cluster View, Cluster Information**. A tick shows the type of view currently selected.
- **Viewer Options** — These change depending on the currently selected view, and are duplicated in the right-click menu in each view.
  - **Properties** — Only for data plots (2-D, 3-D, or multiple), not table or list views. Opens a properties dialog for the current plot. Here you can change settings for grids, lines, markers, legends, and axes. For 2-D and multiple data plots you can choose whether to plot bad data (that is, outliers you have removed with the **Tools** menu).
  - Multiple data plot views have several specific options. You can select **Add Plot** to add another subplot to the current view, or **Remove All Plots** to clear the current view. For the selected subplot (right-click the plot, or click first and use the menu) you can select **Plot Variables** to set up axes or **Remove Plot** to delete.
- **Split View** — Divides the currently selected view and adds the selected option. You can choose from the same options available under **Current View**.
- **Split View Horizontally** — Divides the currently selected plot into two plots side by side. The new plot is a 1-D plot by default; you can select other views using the right-click context menu. Also in the toolbar.
- **Split View Vertically** — Divides the currently selected plot into two plots one above the other. Also in the toolbar.
- **Delete Current View** — Deletes the currently selected view.
- **Print to Figure** — Copies the currently selected view to a Figure window.

See Toolbar above.

Tools Menu

Note that if you are viewing modeling data in the Data Editor (from a modeling node) you cannot change the data, so editing and user-defined variables and filtering are not enabled; you cannot use the **Tools** menu to alter your modeling output.

- **Variables**
  - **Add** opens the **Variable Editor**. Here you can add new variables to the data set by typing into the edit box.
  - **Edit** is only enabled if you already have user-defined variables. This also opens the **Variable Editor** with an additional list on the left containing your defined variables. Select these from the list and edit in the edit box. You can also add new variables here by clicking the button to add a new item to the list.

    The toolbar button **Edit Variables** performs the same functions.

- **Filters**
  - **Add** and **Edit** both open the **Filter Editor**, in a similar way to the variable items. Here you can remove records by imposing constraints on the data. Also in the toolbar.
  - **Remove Outliers** — Removes points you have selected in plots (outlined in red). Use the check box to **Show Bad Data** (in the **Plot Properties** dialog) to see removed points on plots.
  - **Restore Outliers** — Opens a dialog where you can choose any or all removed points to restore.

- **Test Filters** — **Add** and **Edit** both open the **Test Filter Editor**.

- **Test Notes** — **Add** and **Edit** both open the **Test Note Editor**.

- **Change Test Groupings** — Opens the **Test Groupings** dialog; also in the toolbar.

- **Tolerances** — Opens the Tolerance Editor. These settings are for matching data to designs using the cluster views. See "The Tolerance Editor" on page 6-41.

- **Open Storage**— Opens **Storage** where you can store and retrieve variables, filters, and plot settings.

Data Loading and Merging

Data can be loaded and merged from files, from the workspace, and from tailor-made Excel sheets. A test plan can only use a single data set, so you need to use the merge functions to combine data variables from more than one source file in order to incorporate desired variables into one model.

Note that you can also write your own data-loading functions. See "Data Loading Application Programming Interface" on page 8-77. This includes a description of the correct structure for importing data from the workspace. This is the same structure of data as when you export to the workspace.

Loading Data from File

- From the project node choose **Data** –> **New Data**.
  - The Data Editor appears. Click the Open File button in the toolbar 📂 to load data from a file. Alternatively, choose **File** –> **Import** –> **File**.

  The Data Import Wizard appears, to help you select a file.

- Alternatively, from the test plan node, choose **TestPlan** –> **Select Data**.
  - The Data Wizard appears. Click **Load new data set** and the Data Import Wizard appears.

Note If you already have some data loaded, the Data Import Wizard can merge new data into the existing data set.

The Data Import Wizard

1 To import data from a file, enter the file pathname in the edit box, or use the Browse button to find and select the data file. Double-click to load the file.

  The drop-down menu contains the file types recognized by the Model Browser (Excel, Concerto, MATLAB). The default **Auto** tries to determine the type of file it is by looking at the file extension.

2 For Excel files with multiple sheets, you must next select the sheet you want to use and click **OK**.

3 The Import Wizard now 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 edit variable names and units in this list. Click **Finish** to accept the data, unless you have data to merge. See the following.

The Data Import Wizard disappears and the view returns to the Data Editor (or the Data Wizard if you are selecting data from a test plan — see "Data Wizard" on page 6-32).

Merging Data

4 If you already have some data loaded, you cannot click **Finish** but must click **Next** instead. This brings you to the data merging screen of the wizard.

5 Here you must choose one of the radio buttons:
- Merge extra channels (more variables)
- Merge extra records (more data, same variables)
- Overwrite old data (use only the new data)

6 Click **Next**. The wizard summary screen appears, showing the results of the merge. You can click **<Prev** to go back and change merging options.

7 Click **Finish** to accept the data and return to the Data Editor.

Note Obviously the merge might not succeed if the data sets do not contain the same variables. A message appears if the merge is impossible when you click **Next** (step 6), and you must make another choice.

Loading Data from the Workspace

1 From the project node choose **Data** –> **New Data**.

The Data Editor appears.

You can import variables from the MATLAB® workspace by choosing **File** -> **Import** –> **Workspace**.

See "Data Editor Toolbar and Menus" on page 6-7.

The **Loading Data from Workspace** dialog appears.
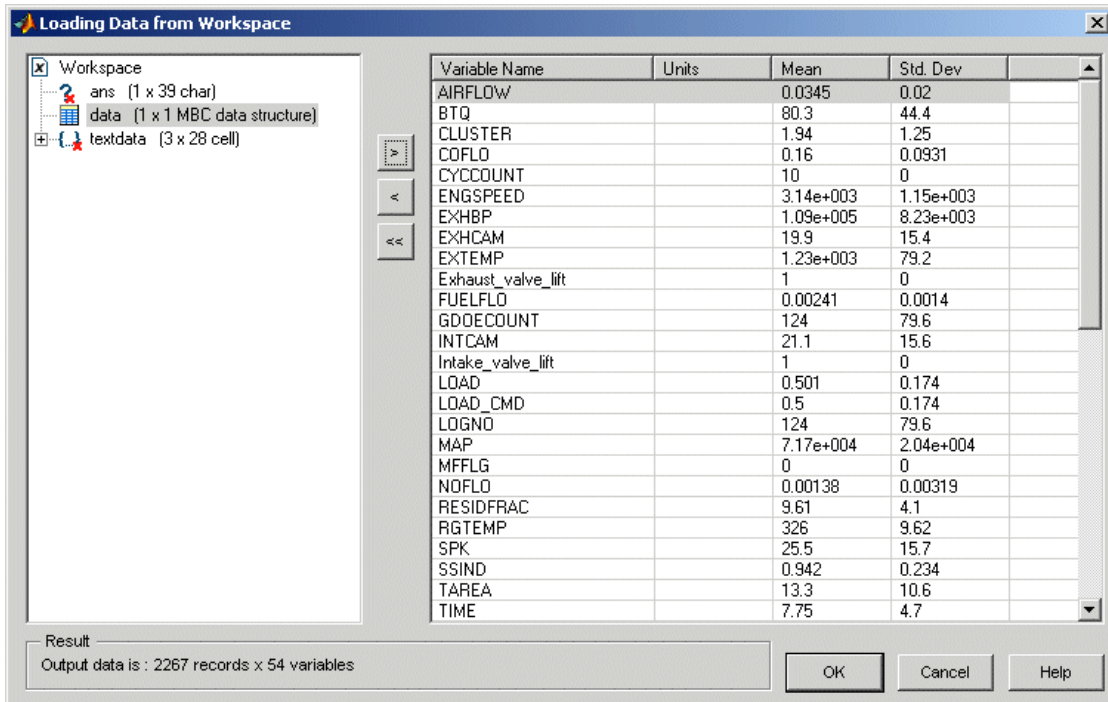
Variables in the workspace are displayed in hierarchical form in the top left pane. Select a variable here, and information about that variable is displayed in the pane on the right.

1 Select a variable to import in the tree at top left.

2 Click the **Add** button.

The number of records and variables appears in the **Output Data** pane. You can add variables (one at a time) as many times as you like (as long as there are no name conflicts).

You can double click to edit variable names and units.

3 You can use the buttons to remove selected variables one at a time from the right list, or remove all variables at once.

4 Click **OK** to accept the data to import and return to the Data Editor.

Data Merging

If you already have data loaded, the Data Merging Wizard appears, where you must choose one of three radio buttons:

- Merge extra channels (more variables)
- Merge extra records (more data, same variables)
- Overwrite old data (use only the new data)

Click **Next**. The wizard summary screen appears, showing the results of the merge. You can click **<Prev** to go back and change merging options.

Note Obviously the merge might not succeed if the data sets do not contain the same variables. A message appears if the merge is impossible when you click **Next**, and you must make another choice.

Click **Finish** to accept the data and return to the Data Editor.

Tailor-Made Excel Sheets

The Data Editor can create a tailor-made Excel sheet for you to fill with data and then import. This sheet will be in the format the Data Editor expects to import data.

1 Select **File –> Import Excel**.

Excel is opened with a new sheet created, containing the labeled rows Name, Unit, and Data.

2 Copy your data and variable names into this sheet, then click **Next** in the wizard to import into the Data Editor.

If the data has been entered in the columns in a way that the Data Editor expects, a summary screen shows you information about the numbers, ranges, means, units, and standard deviations of the records and variables you can import.

3 Click **Finish** to import the data.

Data Editor Views

You can open multiple different views at the same time in the Data Editor, as shown in the following example. You can display 2-D plots, 3-D plots, multiple data plots, cluster plots, data tables, and list views of your filters, variables, and test notes. Use the toolbar buttons, the **View** menu, or right-click a view title bar to split views and change view types.

Functions of different views are described in the following sections:

- "2-D Plots" on page 6-17
- "3-D Plots" on page 6-17
- "Multiple Data Plots" on page 6-17
- "Cluster Plots" on page 6-18
- "List Views — Variables, Filters, Test Filters, Test Notes, and Clusters" on page 6-19
- "Table View" on page 6-19
- "Notes View" on page 6-20

2-D Plots

In the 2-D plot view you can select combinations of variables and tests to plot from the list boxes on the left. Multiple selection of tests and *y*-axis variables is possible — in the preceding example, multiple tests are selected to view several tests simultaneously. See "Data Editor Toolbar and Menus" on page 6-7. Note that you use the 2-D plot view controls for selecting the tests to display. This is unlike the other plots (3-D and multiple data plots), which you control by using the **Tests** list box, always visible on the left of the Data Editor.

To edit data plot properties, you can right-click the plot and select **Viewer Options** –> **Properties**. Here you can choose to show the legend and the grid. You can choose the line style if you want to connect the data points and the data marker point style, if any. **Reorder X Data** redraws the line joining the points in order from left to right. The line might not make sense when drawn in the order of the records. You can use the **Show Bad Data** check box to plot bad data (that is, outliers you have removed).

Click points to select outliers, and remove them 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.

3-D Plots

In 3-D data plot views you can select the variables for each axis from the drop-down menus, and rotate the plot by clicking and dragging. To edit data plot properties, you can right-click the plot and select **Viewer Options** –> **Properties**. Here you can choose the color and style of the axes, whether to show the grid in each axis, and perspective or orthographic axes projection.

Multiple Data Plots

Here you can add as many 2-D plots as desired to the same view, plotting the same selection of tests in a variety of different plots. Use the right-click context menu to add and remove plots, select plot variables, and edit plot properties. You can select single or multiple Y variables to plot against a single X variable (or no X variable) in the **Plot Variables** dialog. Select tests to display in the list on the left of the Data Editor, as for 2-D and 3-D plots and the table view.

Note that you can select different plot properties and variables for each plot within the Multiple Data Plots view, as shown in the example following. Click

to select a plot (or right-click) before selecting **Viewer Options** –> **Plot Variables** or **Plot Properties**. For each plot you can use the same plot properties options as for the single 2-D data plots. You can choose to show the legend and the grid. You can choose the line style if you want to connect the data points and the data marker point style, if any. **Reorder X Data** redraws the line joining the points in order from left to right. The line might not make sense when drawn in the order of the records. You can use the **Show Bad Data** check box to plot bad data (that is, outliers you have removed).

Click points to select outliers, and remove them by selecting **Tools** –> **Filters** –> **Remove Outliers** (or use the keyboard shortcut **Ctrl**+A). Selected outliers are outlined in red, as shown in the example following. 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.
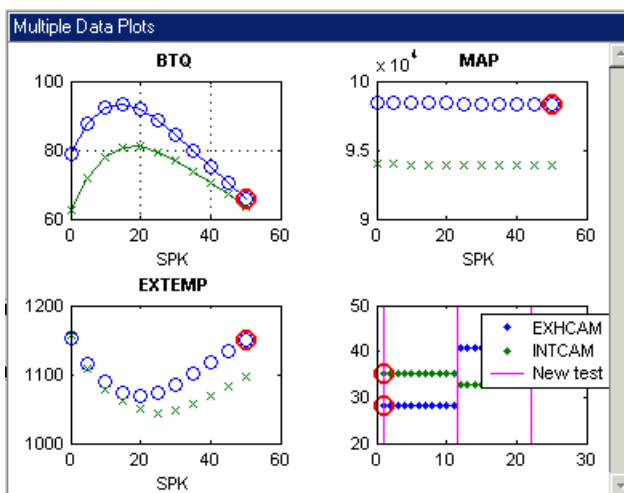
Cluster Plots

You use these views for matching data to design points. Use the cluster information list view with the cluster view to examine your data and design. Click points in the cluster view to select them across the Data Editor — that is, the selected points are displayed in the table view and other data plot views (except 2-D plots, which have separate controls).

See "Matching Data to Designs" on page 6-37 for detailed information on cluster plot controls.

List Views — Variables, Filters, Test Filters, Test Notes, and Clusters

These views show lists and information such as variable and filter definitions, the notes applied to filtered tests, and the data and design points in selected clusters. Variable and Filter List views show the definitions of each variable or filter. Double-click to select particular filters or variables to edit. See "The Variable Editor" on page 6-21 and "The Filter Editors" on page 6-23.

The Test Notes List view shows the rules used to define notes on the data, along with the actual note, the color of the note, and the number of tests to which that note applies. The specified rule is applied to each test in turn to decide if that test should be noted: e.g., mean(TQ) > 0 evaluates the mean torque for each test and notes those tests where the value is greater than zero. The list of individual tests and notes appears in the Notes View.

In the Cluster List view you can choose to include or exclude design and data points by using the check boxes in cluster list views. See "Matching Data to Designs" on page 6-37 for more information on using these.

Table View

In the Table view, you can view your data, edit, and add records.

Points you have selected in plots (by clicking) are green in the table. Points you have removed as outliers are light blue in the table. Cells containing the value "NaN" are black.

Select **View –> Viewer Options** or right-click the title bar to reach these options.

- **Select Columns** — Opens a dialog where you can use the check boxes to select the columns to display in the table. Select a column, then press **Ctrl**+A to select all columns, and then you can select or clear all check boxes with one click.
- **Allow Editing** — Toggles editing, and, as a side effect, causes all records to be shown, including those which are filtered out. Records which are filtered out appear dark blue in the table. You can alter records by clicking a cell and then typing a new value. Editing the value of a cell may cause that row to be

filtered out. If so, the background colour of the row will change after the cell has been edited. When editing is allowed, the next two options are enabled.

- **Undo Edits in Selected Region** — You can click and drag to highlight a region, then use this option to reverse any edits in the highlighted area.

- **Duplicate Selected Records** — First select one or more records, then use this option to duplicate them. Each duplicate appears directly underneath the parent record. Edit duplicates to create new records.

Notes View

This view shows a list of all tests in the data, along with those notes that apply to particular tests. Each note that applies to one or more tests has its own column in the view. You can sort the columns in the list by clicking the column header (once to sort ascending and once more to sort descending) allowing you to quickly identify the tests that satisfy a given note definition — this facility could be used to screen incoming data for problem tests.

You can click a particular test in the Notes view list to select that test across the Data Editor. 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.

The Variable Editor

New Variables

New variables you create in the Variable Editor can be viewed in the **Data Editor** by using the right-click menu in any view to select **Current View –> Variable List**. Alternatively, select the menu item **View –> Split View –> Variable List** to add this new view to the currently selected plot.

- You create variables by doing the following:
  - Clicking the 📝 toolbar button
  - Selecting **Tools –> Variables –> Add**
  - Alternatively, by selecting an existing **Variable List** view by clicking in it, then pressing **Insert**

  The Variable Editor appears. See below for instructions.

You can also load user-defined variables from storage.

- You can edit existing variables
  - Directly, by select-clicking in the **Variable Expression** pane or pressing **F2** (as in renaming in Windows Explorer)
  - By double-clicking, which opens the Variable Editor
  - By choosing **Tools –> Variables –> Edit**

- You can also delete variables
  - By selecting them in the **Variable Expression pane** and pressing **Delete**

Variable Editor

You can define new variables in terms of existing variables:

- Define the new variable by writing an equation in the edit box at the top of the **Variable Editor** dialog.

You can type directly in the edit box or add variable names by double-clicking them. In the case of variable names especially, this latter method avoids typing mistakes. Variable names are case sensitive.
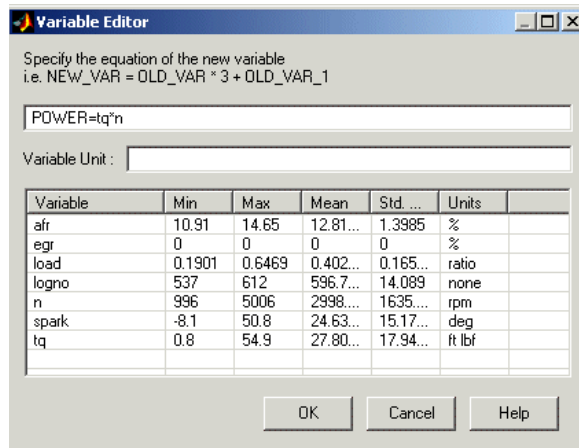
The preceding example shows a definition for a new variable called POWER that is defined as the product of two existing variables, tq and n, by entering POWER = tq x n.

- Click **OK** to add the new variable to the current data set.

Note that the Variable Editor looks different depending on whether you opened it to create a new variable or edit an existing one. The example above shows the editor when adding a new variable. If you open the editor to edit a variable there is an additional list on the left. You can choose which of your existing variables to edit from this list, or click the button to add a new item to the list if you want to add a new variable.

The Filter Editors

New Filters

New filters you create in the Filter Editor can be viewed in the **Data Editor** by using the right-click menu in any view to select **Current View –> Filter List**. Alternatively, select the menu item **View –> Split View –> Filter List** to add this new view to the currently selected plot. Their effects are shown graphically in the bars at the top of the **Data Editor** — removed data is shown in red.

- You can create filters
  - By clicking the toolbar button
  - By selecting **Tools –> Filters –> Add**
  - Alternatively, by selecting an existing **Filter List** view by clicking in it, then pressing **Insert**

  The Filter Editor appears. See below for instructions.

You can also load user-defined filters from storage.

- After you create them, filters can be edited in the same way as variables,
  - Directly, after you select-click them in a **Filters List** pane, or by pressing **F2**
  - Using **Tools –> Filters –> Edit**, which opens the Filter Editor
  - By double-clicking, which also opens the Filter Editor
- Delete filters by selecting them and pressing **Delete.**

Test Filters and Test Notes

Similarly you can add test filters (to filter out entire tests, instead of individual observations) and test notes (to mark every test that fulfills set criteria). The **Test Filter Editor** and **Test Notes Editor** can be reached from the **Tools** menu. You can view these filters in the **Data Editor** in the same way as the other filters by selecting **Test Filter List** or **Test Notes List** as the type of view. You define, edit, store, and delete these filters in the same way, and they can also be loaded from storage.

Filter Editor

A filter is a constraint on the data set used to exclude some records. You define the filter using logical operators on the existing variables.
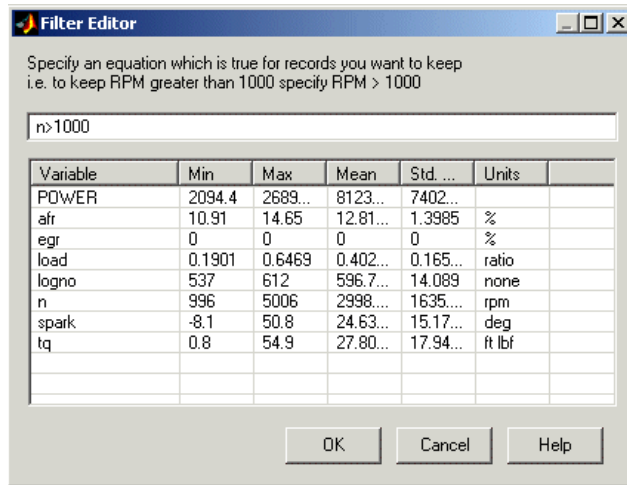
In the preceding example, n>1000, the effect of this filter is to keep all records with speed (**n**) greater than 1000.

Click **OK** to impose new filters on the current data set.

Note that the Filter Editor looks different depending on whether you opened it to create a new filter or edit an existing one. The example above shows the editor when adding a new filter. If you open the editor to edit a filter there is an additional list on the left. You can choose which of your existing filters to edit from this list, or click the button to add a new item to the list if you want to add a new filter.

Storage

Storage allows you to store plot preferences, user-defined variables, filters, and test notes so that they can be applied to other data sets loaded later in the session.

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

Store Current Filters

Store Current Variables

Store Current Test Filters

Append Stored Object

Store Current Test Notes

Store Current Data Plots

Import from File

Store Current Editor Layout

Delete Stored Object

Export to File

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 plot settings for multiple data plot views with the toolbar button Get Current Data Plots. All the toolbar functions are described below.

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.

You can edit the names of stored objects, by select-clicking as in Windows Explorer, or by clicking once and pressing **F2**.

Note that if you enter the data editor from a modeling node you cannot append variables or filters, as modeling data is read only.

The toolbar buttons in **Storage** allow the following functions:

- **Export to File** sends **Storage Objects** to file so they can be sent to another user or machine. Objects remain in storage indefinitely unless deleted; export is only for transporting elsewhere.
- **Import from File** loads storage objects from file; it does not overwrite current objects.
- **Append Stored Object** appends the currently selected **Storage Object** to those already in the **Data Editor** window. Double-clicking an object in storage also appends it to the current session (except Data Editor Layout, which replaces all the current views with the stored layout). If you double-click an object in the left pane the entire object is appended. The pane on the right displays the details of the current object selected in the left pane (variables, filters, or plot properties). In the right pane you can double-click to add these elements individually. For example, you could add a single plot out of several, or a single variable. You can also hold down **Shift** to make multiple selections here.
  - Variables and filters will appear in variable and filter list views. Note that the views are not a requirement for appending variables or filters.
  - Stored Plot Properties (plotted variables, grid style, legend, line style, and so on) apply to multiple data plot views. If you have more than one multiple data plot view open, the stored plot properties are appended to the currently selected view (with the highlighted title bar), or to all views if none are currently selected. The plots stored in a Plot Properties object

are added to multiple data plot views as additional plots — they do not overwrite existing plots. You cannot append stored plot properties if you do not have a multiple data plot view open.

- **Store Current Variables** creates a **Storage Object** containing definitions of all current user-defined variables.

- **Store Current Filters** creates a **Storage Object** containing definitions of all current filters. Similarly, **Store Current Test Filters** and **Store Current Test Notes** create storage objects containing test filters and test notes.

- **Store Current Data Plots** creates a storage object containing the current settings for multiple data plot views (variables plotted and styles for grids, lines, and legends). The settings are taken from the currently selected multiple data plot view. If there are multiple views and none are selected the settings from all the plots are concatenated into a single Plot Properties object.

- **Store Current Data Editor Layout** creates a storage object containing the current settings for all the views. Saving a set of views and then double-clicking this object in Storage can be a very quick way to achieve a variety of plots and list views in a new session.

Delete Stored Object deletes the currently selected **Storage Object**.

Test Groupings

The **Define Test Groupings** dialog collects records of the current data set into groups; these groups are referred to as **tests**. Test groupings are used to define hierarchical structure in the data for two-stage modeling.

You access the dialog from the Data Editor by doing one of the following:

- Using the menu **Tools –> Change Test Groupings**
- Using the toolbar button ⊞

When you enter the dialog, no plot is displayed.

1 Click to select a variable in the list box to use in defining groups within the data.

2 The Add Variable button (⊞) adds the currently selected variable in the **Variables** list to the list view on the left. Alternatively, double-click variables to add them.

You can now use this variable to define groups in the data.

In the following example, the variable **n** is being used to define groups. 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 can 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.

You can plot variables without using them to define groups by clearing the **Group By** check box.

You can remove variables from consideration by selecting the unwanted variable in the list view (the selection is highlighted in blue) and clicking the Remove variable button ⊞.
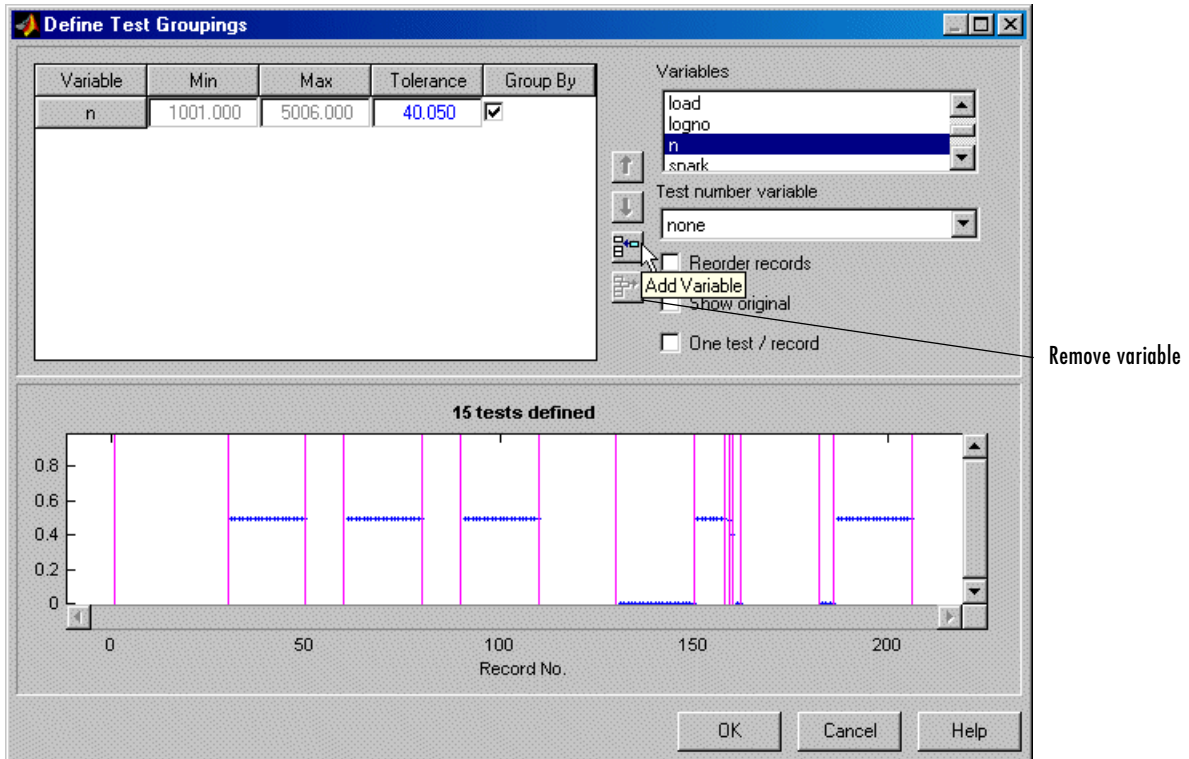
The plot shows the scaled values of all variables in the list view (the color of the **Tolerance** text corresponds to the color of data points in the plot). Vertical pink bars show the tests (groups). You can zoom the plot by **Shift**-click-dragging or middle-click-dragging the mouse.

Test number variable contains a drop-down menu showing all the variables in the current data set. You can select any of these to number the tests (for example, lognumber could be useful (instead of 1,2,3…) if the data was taken in numbered tests and you want access to that information during modeling).

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 `lognumber` or other variable.)

Reorder records allows you to reorder records in the data set. This sorts records before grouping. Otherwise, the groups are defined using the order of records in the original data set.

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.

Clicking **OK** accepts the test groupings defined and dismisses the dialog.

Data Wizard

After setting up a new test plan, *if no data has been selected in that test plan*, then either of the following brings up the Data Wizard.

- Choosing **Select Data** from the test plan node (toolbar button or **TestPlan** menu item)
- Double-clicking the Responses block in the test plan diagram

Step 1: Select Data Set

Use the first screen of the wizard to select the data set to build models from. You can also select whether to use all the data set or to match the data to a design, if any designs are in use in the test plan. Designs appear in the left list box.
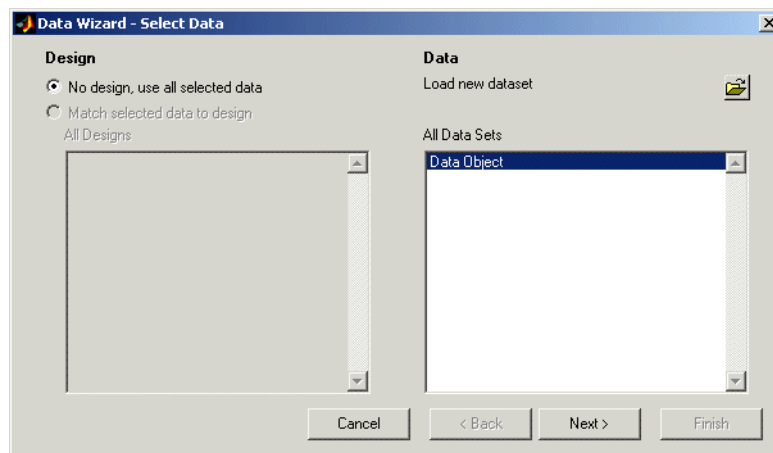
Step 2: Select Input Signals

Select the input signals for the model (on all stages of the hierarchical model) from the list box of data signals on the right, and match them to the correct model variables using the big button with the arrow. Double-clicking an item in the data signals list selects that signal for the currently selected input factor (with the range if the **Copy range** check box is selected) and then moves to the next input.

If you entered the correct signal name at the model setup stage, the appropriate signal is automatically selected as each model input factor is selected. This can be time-saving if there are many data signals in a data set. If the signal name is not correct, you must select the correct variable for each input by clicking.

Select the check box **Copy range** if you want to use the range of the selected data signal for the model input range. Ranges are not automatically copied, although stored templates have the ranges that were set when the template was saved.
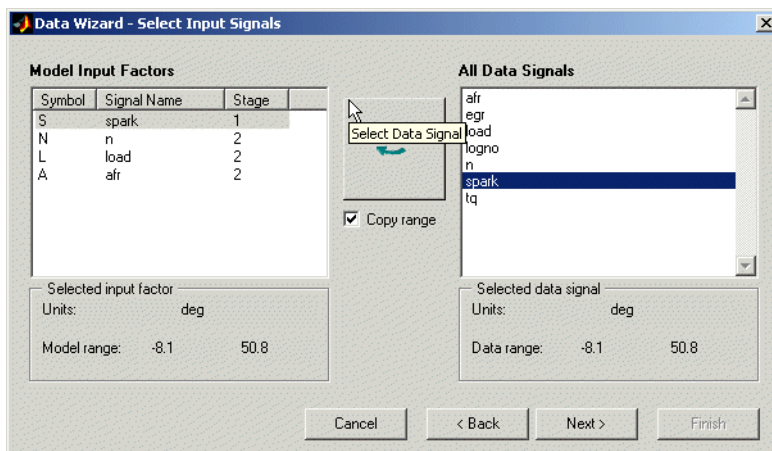
Step 3: Select Response Models
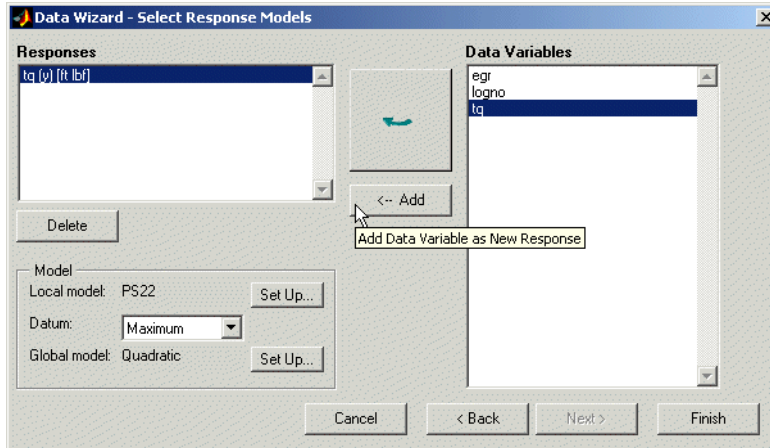
Starting from scratch (with an empty **Responses** list box), select the desired response in the **Data Variables** list and click **Add**.

In the preceding example, the test plan template specified torque as the response model, so it already appears in the **Responses** list box. If you want to change the response, select another variable and click the large button with the arrow. This replaces the current selected response. The previous response appears in brackets to inform you what has changed.

When there is already a response in the list box, clicking **Add** does not replace the selection but increases the number of responses. The replace button (with the arrow) is not available when the **Responses** box is empty.

You can use **Delete** to remove selected responses. You can select datum models (if the local model supports them), and you can change the local and global models by using the **Set Up** buttons. See "Global Model Setup" on page 7-40, "Local Model Setup" on page 7-60, and "Datum Models" on page 7-79 for details.

Step 4: Set Tolerances

Setting tolerances is only relevant if you are matching data to a design. This screen only appears if you selected the radio button option **Match selected data to design** in step 1. You can only match data to designs for global models.

You can also edit tolerances later using the **Tools** menu in the **Data Editor** window. See "The Tolerance Editor" on page 6-41 for definitions of clusters.

Set **Input Factor Tolerances** for each variable to determine the size of the tolerance in each dimension. This is used for selecting data. The tolerance in each dimension determines the size of "clusters" centered on each design point. Data points that lie within tolerance of a design point are included in 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 fall within tolerance of a particular design point, they remain unmatched. Default tolerance values are related to variable ranges.

The choices you make in the **Default Data Selection** options determine how the cluster algorithm is first run to select matching data and design points. This only affects the status of the check boxes for data and design points when you first see the Cluster Plot view in the Data Editor. You can always alter the

results of this later in the Cluster Information list view, where you can manually select the data and design points you want to use.

When you exit the Data Editor these selections determine what data is used for modeling and how design points are augmented and replaced. Selected data is used for modeling and added to the design. Data you have decided to exclude is not used for modeling or added to the design.

- **Unmatched data** — `Use` or `Do not use`

  Data that does not lie within tolerance of any design point is unmatched. You can decide what to do with these. If you select `Use`, this data is selected for modeling and added to your design.

  If you select `Do not use`, then unmatched data is not used for modeling or added to the design; it is excluded data.

- **Clusters with more data** — `Use all data` or `Use closest match only`

  This refers to clusters containing more data points than design points. If you choose `Use all data`, all the data points in these clusters are selected for modeling and added to the design, replacing the design points in those clusters.

  If you choose `Use closest match only`, then a one-to-one match of the data point closest to each design point is selected, and these are the only points that are selected for modeling and added to the design (replacing a design point each).

- **Cluster with more design** — `Do not replace design points` or `Replace design with closest`

  All data from these clusters is selected for modeling. The setting here only affects selections for the design.

  Where clusters contain more design points than data points, you can choose to leave the design unchanged by selecting `Do not replace design points`.

  If you choose `Replace design with closest`, this replaces the design points where possible with the closest data point and leaves the rest of the design points unchanged.

Remember you can override any of these selections manually in the Data Editor; changes are only applied when you close the Data Editor after matching. The cluster plots view in the **Data Editor** window appears by default when you close the Data Wizard while you are matching data to designs. See the next section, "Matching Data to Designs" on page 6-37.

Matching Data to Designs

The Cluster Plot view is designed for matching data to experimental designs for global models. Here you can select data for modeling. 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.

Use the Cluster Plot view and associated Cluster Information List view to select data for modeling. All data you select is also added to a new design called Actual Design. You can use the matching process to produce an Actual Design that accurately reflects your current data. You can then use this new design to decide the best points to use if you want to augment your current design in order to collect more data.

The following example shows the Data Editor displaying a Cluster Plot view and Cluster Information List view. You can right-click to change and add views as for the rest of the Data Editor displays.

For instructions, see the following section, "How to Use the Cluster Plot View" on page 6-39.
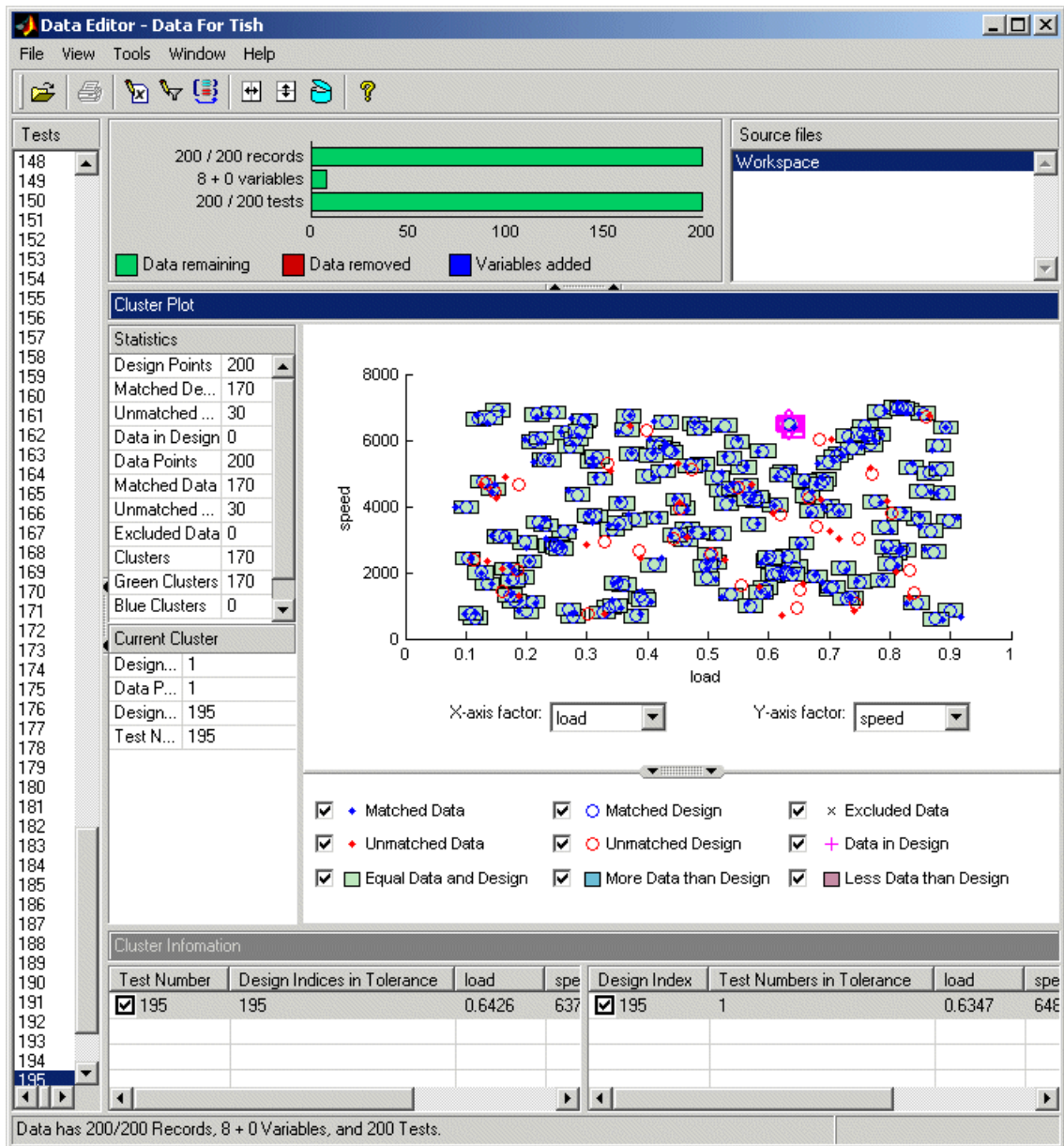
How to Use the Cluster Plot View

For a step-by-step guide to matching data to a design using an example project, see the tutorial section "Matching Data to Designs" on page 4-21.

You can **Shift**+click (or center+click) and drag to zoom in on clusters of interest, as shown in the following example. Double-click the plot to return to full size.
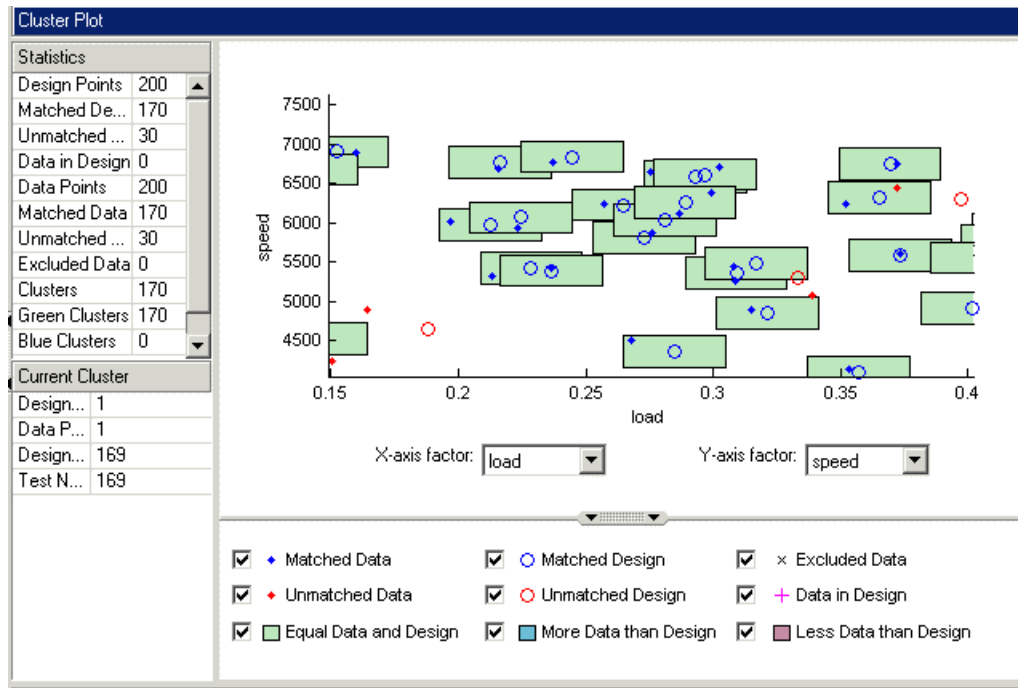
We suggest the following sequence as a guideline for matching data to designs using the Cluster Plot:

1 It is unlikely that you will get the tolerances right immediately. Open the **Tolerance Editor** using the **Tools** menu and try different values for different variables. These values determine the size of clusters centered on each design point. Data points that lie within tolerance of any design point in a cluster are matched to that cluster. See "The Tolerance Editor" on page 6-41 for cluster definitions.

2 For matching data to designs, you might want to clear the check box in the Cluster Plot for green clusters (with equal data and design points). 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 to design points as possible, that is, as few red clusters as possible. See "Red Clusters" on page 6-41.

3 You can see the values of variables at different points by clicking and holding. Selected points have a pink border. Once points are selected, you can change the plot variables using the X- and Y-axis factor drop-down menus to track those points through the different dimensions.

This can give you a good idea of which tolerances to change in order to match points. Remember that points that do not form a cluster can 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. 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.

Remember that points you select in the cluster view are selected across the Data Editor, so if you have other data plots or a table view open you can investigate the same points in different views.

4 Once you have found useful values for the tolerances by trial and error, you can make selections of points within clusters that have uneven numbers of data and design points. These clusters are blue (more data than design) or red (more design than data). Select any cluster by clicking it. The details of every data and design point contained in the selected cluster appear in the Cluster Information List view. Choose the points you want to keep or discard by selecting or clearing the check boxes next to each point in the Cluster Information List view. Notice that your selections can cause clusters to change color as you adjust the numbers of data and design points within them.

5 You can also select unmatched points by right-clicking and selecting **Viewer Options** –> **Select Unmatched Data**. All unmatched points then appear in the Cluster Information List view. You can decide whether to include or

exclude them in the same way as points within clusters, by using the check boxes in the Cluster Information List view. If you decide to exclude data points (within clusters or not) they appear on the plot as black crosses (if the **Excluded Data** check box is selected for display).

Note that it is a single fast operation to multiple-select points before selecting or clearing a check box, rather than selecting points individually. To do this, use **Shift**+click to select multiple points and hold the **Shift** key when clicking one of the check boxes.

You can right-click and select **Viewer Options –> Show Labels** to see design and data point numbers on the plot (also in the View menu).

Continue this process of altering tolerances and making selections of points until you are satisfied that you have selected all the data you want for modeling. All selected data is also added to your new Actual Design, except that in red clusters.

Red Clusters. These contain more design points than data points. These data points are not added to your design, because the algorithm cannot choose the design points to replace, so you must manually make selections to deal with red clusters if you want to use these data points in your design. If you don't care about the Actual Design (for example, if you do not intend to collect more data) and you are just selecting data for modeling, then you can ignore red clusters. The data points in red clusters are selected for modeling. For information about the effects of your selections, see "What Will Happen to My Data and Design?" on page 6-43.

The Tolerance Editor

You can open the Tolerance Editor by selecting **Tools –> Tolerances**.

Here you can edit the tolerance for selecting data points. You can choose values for each variable to determine the size of tolerance in each dimension.

- Data points within the tolerance of a design point are included in 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.

- Excluded data (shown as black crosses) that lies within tolerance appearsin the Cluster Information list when that cluster is selected. You can then choose whether to use it or continue to exclude it.
- Data in Design (pink crosses) is the only type of data that is not included in clusters.

Note Tolerances are set for global variables. Data used for matching uses test means of global variables, not individual records, unlike other Data Editor views. Click points to inspect values of global variables.

Using the Tolerance Editor is the same process as setting tolerances within the Data Wizard. In the Data Wizard you can also choose in advance what to do with unmatched data and clusters with uneven numbers of data and design points. These choices affect how the cluster algorithm is first run; you can always change selections later in the Data Editor. See "Step 4: Set Tolerances" on page 6-35.

Note If you modify the data in any way while the cluster plot view is open (e.g., by applying a filter) the cluster algorithm will be rerun. You might lose your design point selections.

See the next section, "What Will Happen to My Data and Design?" on page 6-43, for information about what happens to your data set and design when you close the Data Editor after data selection in the cluster plot view.

What Will Happen to My Data and Design?

As with everywhere else in the Data Editor, the changes you make are only applied to the data set when you exit. When you close the Data Editor, your choices in the cluster plots are applied to the data set and a new design called `Actual Design` is created. All the changes are determined by your check box selections for data and design points.

Note All data points with a selected check box are selected for modeling. All data points with a cleared check box are excluded from the data set.

All data points with a selected check box are put into the new `Actual Design` *except* those in red clusters. See below.

When you close the Data Editor, these changes are applied:

- Green clusters — equal number of data and design points

  The design points are replaced by the equal number of data points. These points become fixed design points (red in the Design Editor table) and appear as Data in Design (pink crosses) when you reopen the Data Editor.

  This means that these points are not included in clusters when matching again. These fixed points are also not changed in the Design Editor when you add points, although 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.

- Blue clusters — more data than design points

  The design points are replaced by all the data points.

Note Design points with selected check boxes in green or blue clusters are the points that will be replaced by your selected data points. You may have cleared the check boxes of other design points in these clusters, and these points will be left unchanged.

- Red Clusters — more design than data points

  Red clusters indicate that you should make a decision if you want your new Actual Design to reflect your most current data. The algorithm cannot choose the design points to replace with the data points, so no action is taken. Red clusters do not make any changes to the design when you close the data editor. The existing design points remain in the design. The data points are included or excluded from the data set depending on your selections in the Cluster Information list, but they are not added to the design.

- Unmatched Design Points

  These remain in the design.

- Unmatched Data Points

  If you have selected the check boxes for unmatched data, they become new fixed design points, which are red in the Design Editor. When you reopen the Data Editor these points are Data in Design, which appear as pink crosses. Note that in the Data Wizard you could choose Use to select all these initially, or you could choose Do not use, which clears all their check boxes. See "Step 4: Set Tolerances" on page 6-35.

- Data in Design

  These remain in the design.

- Excluded Data

  These data points are removed from the data set and are not displayed in any other views. If you want to return them to the data set you can only do so by selecting them in the Cluster Plot view.

7

Modeling

This section discusses the following topics:

Modeling with the Model-Based Calibration Toolbox

This is a list of the main topics of this section. Functionality is described in the order you see it during the process of model building.

For a quick guide to setting up models, then searching for the best fit, see these overview pages:

The different views of the Model Browser are described in these sections:

To construct models, you must navigate using the model tree, use test plans, and load and manipulate data. These topics are covered in these sections:

Model construction, evaluation and export are covered in these sections:

Project Level: Startup View

When you open the Model Browser part of the Model-Based Calibration Toolbox there is a single node, the project (named Untitled), in the model tree. This node is automatically selected.

When the project node in the model tree is selected at any time, the following functionality is available. This state is called *project level.* When you start you are automatically at project level, as there are not yet any other nodes to select.

See "Model Browser User's Guide" on page -1 for information about using the Model Browser.

Note The node selected in the model tree determines what appears in the menus and panes of the rest of the Model Browser.
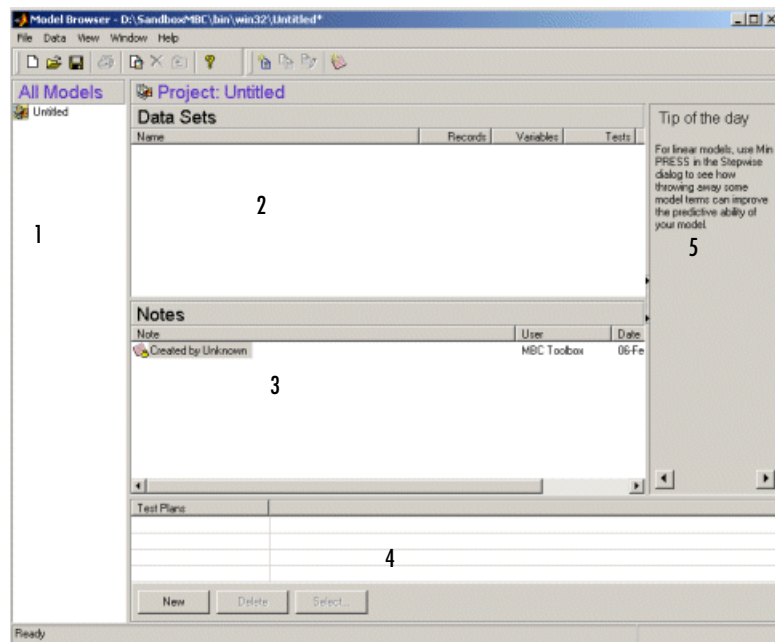
All Models Pane

(Labeled 1 above). This pane contains a hierarchical structure showing all the models created within the current project. See "Model Tree" on page 7-11 for a detailed description of the information contained in this pane.

Data Sets Pane

(Labeled 2 above). All data sets loaded in the current project are displayed in the **Data Sets** pane (whether in use for modeling or not).

You can select a data set (by clicking it) and then

- Delete it by pressing the **Delete** key.
- Rename it, by clicking again or pressing **F2** (as when selecting to rename in Windows Explorer), then editing the name.
- Open it by double-clicking. Double-clicking a data set opens the Data Editor; unless it is already associated with a test plan, see below. See "The Data Editor" on page 6-4.

Note All data sets loaded are visible at the project node and appear in the **Data Sets** pane. However, they are not necessarily used by any test plan child nodes of that project until you select them within a particular test plan. For example, with a data set loaded at the project node, when you switch to a new test plan node, the **Data Sets** pane at top right displays 'No Data is selected until you use the Data Wizard to attach data to that test plan. See "Data Wizard" on page 6-32.

The same data set can be used by many test plans within a project, although each individual test plan can only ever use one data set.

When you have associated a data set with a test plan, a new data set icon (with a name specific to that test plan) appears here in the Data Sets pane. The same data set may be used by several test plans, at which you may have applied different filters, groupings or edits, and so each time you associate a data set with a new test plan a new icon appears here. You cannot edit these test plan-specific data sets at project level, you must edit them from the associated test plan.

Notes Pane

(Labeled 3 above). The **Notes** pane contains a list box showing all previous notes on the current project. You use notes to record changes within a project, perhaps by different users, over time.

- You add new notes by clicking the **Add new note** button 📝 in the toolbar, or by pressing the **Insert** key after selecting the **Notes** pane by clicking. Notes automatically have the user login name and the date.

- You edit notes (only the user that created them can edit them; user names must match) by select-clicking or by pressing **F2** when selected (as when selecting to rename in Windows Explorer). Edited notes have updated time and date fields.

- You remove notes by selecting them and pressing **Delete** (but only the same user that created them can delete them).

- Notes are automatically added to the project when it is modified (for example, the initial "Created by <username>" note). These notes (listed as user "info") cannot be deleted or edited.

Test Plans List Pane

(Labeled 4 above). You generate new test plans from the **Test Plans** list pane by clicking the **New** button. See "Test Plans" on page 7-17.

This pane is the **Test Plans** list pane at startup but changes depending on the level in the model tree that is selected. The list box always displays all the child nodes of whichever node is currently selected in the tree in the **All Models** pane, and always contains three buttons: **New**, **Delete**, and **Select**.

Double-clicking any item within this pane changes the view directly to that node. (This is equivalent to selecting that node in the model tree.) You can also use the **Delete** and **Insert** keys to remove or add new test plans (select a test plan first).

- The **Test Plans** list becomes the **Response Models** list, the **Local Models** list, the **Response Features** list, and the **Models** list as you select the nodes at subsequent levels of the model tree. In each case this pane displays the

immediate child nodes of the current node selected. You can use the buttons to delete selected nodes or create new nodes.

- The feature added by clicking **New** always corresponds to the list items. For example, clicking **New** when the pane shows a list of test plans adds a new test plan. Clicking **New** when the pane shows a list of response features opens the **New Response Feature** dialog, as shown in the following example. The response features you can add are model-specific. This example shows the response features available for a polyspline model.
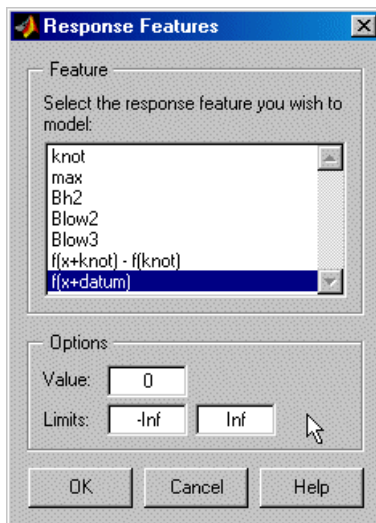
For example, if you choose **f(x + datum)** and enter 10 in the **Value** edit box, the new response feature tracks the datum +10. For a torque/spark polyspline model, the datum is MBT (maximum brake torque); so the new response feature is MBT + 10 degrees of spark angle. This allows you to create response features that have engineering interest.

The response features available depend on the model type. For more details on which response features are available, see "Local Models and Associated Response Features" on page 8-51.

- You can use the **Select** button to select the best child node, but only when the child nodes are local models, response features, or submodels of response features. In each case clicking the **Select** button takes you to a selection

window where you can compare the child nodes. See "Selecting Models" on page 7-111.

Tip of the Day

(Labeled 5 above). Hints about using the Model Browser appear here. You can scroll through more tips using the buttons at the bottom, and you can snap this pane closed or open by clicking the "snapper point" where the cursor changes if you roll the mouse over it.

Project Level: Toolbar
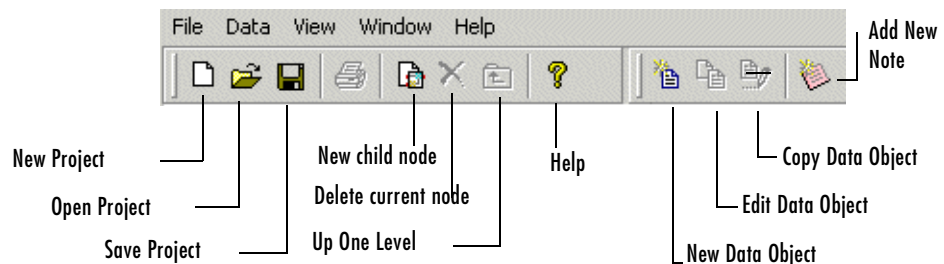
This is how the toolbar appears when you first start the toolbox. The last two Data buttons are grayed out; the **Edit data object** and **Copy data object** buttons are not enabled until a data set has been loaded.

- All the toolbar items are duplicated under the menus except **New Note**.
- For the **Project** buttons, see the **File** menu.
- For the **Data** buttons, see the **Data** menu.
- **New Note** adds a note to the **Notes** pane.
- The **New** and **Delete** node buttons are duplicated in the **File** menu. In both cases, their function depends on the node selected in the model tree. In every case, **New** generates a new child node of the one currently selected, and **Delete** removes the current node (and all its children).
- The **Up One Level** button moves the current selection (and hence all the views) one level up the model tree. For example, if a test plan node is selected, clicking this button moves one level up to the project node.

Two buttons, **Delete** and **Up One Level**, are grayed out at startup because the default selection in the model tree is the project node, so there are no levels above, and you cannot delete the project node (although you can replace it with a new one).

- The print icon is only enabled in views with plots, for example, the local node, response feature nodes, and response nodes *after* selection of a best two-stage models (response nodes are blank until then).
- The help button opens the Help Browser and displays the appropriate help documentation for the current view in your Model Browser.

Project Level: Menus

File Menu

Note The **File** menu remains constant in each Model Browser view. The **New** *child node* function always creates a new child node, and the **Delete** *current node* function always deletes the current node. These change according to which node in the model tree is currently selected.

- **New Project** opens a new project file. You are prompted to save or lose the current project.
- **Open Project** opens a file browser to select the project to open.
- **Save Project** and **Save Project As** save the project with all the models it contains as a .mat file.
- **New Test Plan** opens a dialog with the choice of One-Stage or Two-Stage test plans, or you can browse for other test plans. The **New** (child node) menu option always creates a new child node of whichever node is selected in the model tree.

    At startup, the project node is automatically selected, so the appropriate child node is a new test plan.

Note **File** –> **New** changes depending on which node in the model tree is selected. In each case the option offers a new child node immediately below the one currently selected, that is, a **New Test Plan** (if a project node is selected), a **New Response Model** (from a test plan) or a **New Model** child node (from a one-stage response). For two-stage models you can add a **New Local Model** (from a response node), a **New Response Feature** (from a local node) and a **New Model** from a response feature node.

- **Export Models** brings up the **Export Models** dialog. This allows you to export any models selected in the tree (along with their child nodes, in some cases) to the MATLAB workspace, to file for importing into CAGE, or to Simulink. See "Exporting Models" on page 7-148.
- **Delete "Untitled"** Like the **New** item in this menu, this option changes depending on which node in the model tree is selected. This menu item deletes whichever node is currently selected in the model tree (along with any child nodes), and the appropriate name appears here.
- **Clean Up Tree** From any modeling node where a best model has been selected (from the child nodes), you can use this to delete all other child nodes. Only the child nodes selected as best remain.
- **Preferences** brings up the **MBC File Preferences** dialog, in which you can specify default locations for projects, data, models, and templates. You can also edit and save user information: name, company, department and contact details. This information is saved with each project level note, and you can use this to trace the history of a project.
- **Print** is only enabled in views with plots — not test plan or project level.
- **1,2,3,4:** A list of the four most recent project files, including their pathnames.
- **Close** Exits from the Model Browser part of the toolbox (CAGE and MATLAB are unaffected).

Data Menu

- **New Data** — Opens the Data Editor. See "The Data Editor" on page 6-4.
- **Copy Data** — Copies the selected data set.
- **Edit Data** — Opens the Data Editor to enable data editing.

• **Delete Data** — Deletes the selected data set.

View Menu

• **Tip of the Day** — Choose to display or hide the tips pane.
• **Note Information** — Opens a dialog where you can decide which categories of information to display for each project note. You can specify user information for display with notes by using **File –> Preferences**.

Window Menu

Depending on which toolbox windows are open, a list appears under this menu and whichever window is selected is brought to the front. The **Window** menu remains constant throughout the Model Browser.

Help Menu

The **Help** menu remains consistent throughout the Model Browser.

• **MBC Help** — Opens the Model-Based Calibration Toolbox Roadmap with links to the help tutorials and the indexed help pages.
• **Context Help** — Depending on what part of the Model Browser is currently active, **Context Help** links to different places in the Help files.
• **About MBC** — Displays version notes.

Model Tree

The tree in the **All Models** pane displays the hierarchical structure of the models you have built. Views and functionality within the browser differ according to which node is selected.

The following is an example of a model tree.
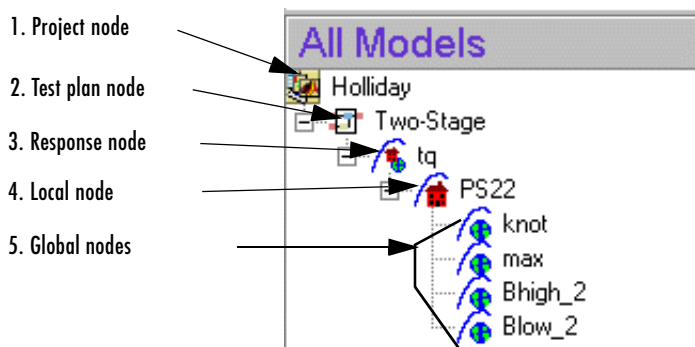
The elements of the tree consist of the following:

1 Project node

2 Test plan node

3 Response node

4 Local node

5 Global nodes — All one-stage model nodes are global models. For two-stage models, global models are fitted to response features of the local models. Each step down in the tree is a *child node* of the node above. Global models are child nodes of local nodes and so on.

Note The selected node in the tree governs the model that is displayed in the various other panes of the browser and which menu items are available. The selected node governs the level displayed: project level, test plan level, and so on. The functionality of each level is described in the Help.

You can rename all nodes, as in Windows Explorer, by clicking again or by pressing **F2** when a node is selected.

There is a context menu available. When you right-click any node in the model tree, you can choose to delete or rename that node, or create a new child node.

Tree Structure
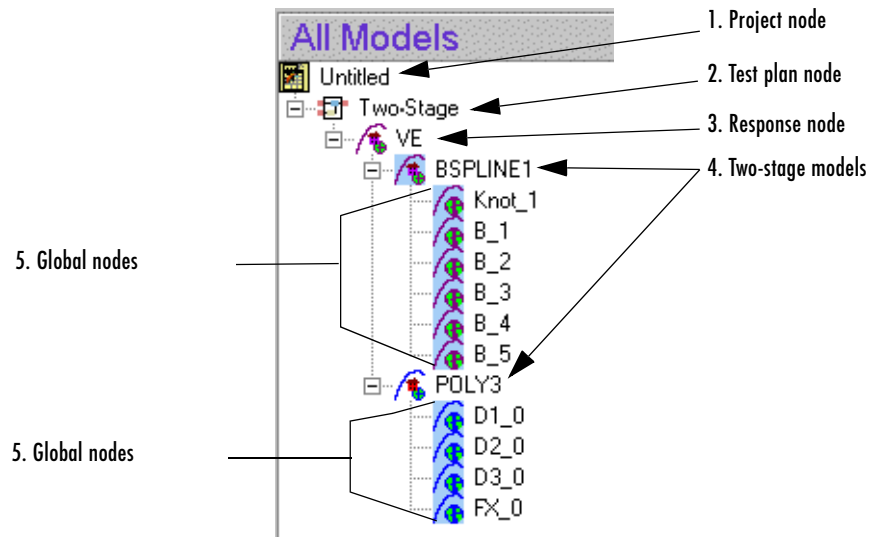
The preceding example shows a more extensive model tree, with two two-stage models as child nodes of a single response model.

There can be many models within (or under, as *child nodes* in the tree) each two-stage global node, or any one-stage model node.

There can also be many different response nodes within a single test plan, and each project can contain several different test plans. However, there is only one project node in the tree at any time.

> **Note** You can only have one project open at any one time; if you open another, you are prompted to save or discard your current project.

You can add child nodes to all global models — several candidate models can be tried at each global node and the best selected. There is an example showing this at the end of the section on "Icons: Blue Backgrounds and Purple Worlds" on page 7-14 and the process is illustrated in the "Tutorial: Quickstart" on page 2-1.

Icons: Curves, Worlds, and Houses

The icons are designed to give visual reminders of their function.

- Test plan icons have a tiny representation of the test plan diagram. You can see the one-stage and two-stage icons in the following example.
- The local model icon shows a curve over a house.
- Global model icons show a curve over a globe. All one-stage models are global models and for two stage models, all nodes below the local node are global models.
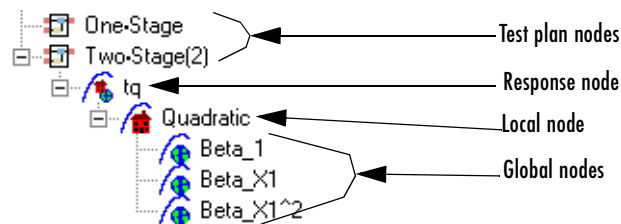
- The response node (empty until a two-stage model is calculated) has an icon that combines the local and global motifs — a curve over a house *and* a globe — to symbolize the two-stage process.
- When a two-stage model has been calculated, the icon at the local node changes to show the combination of local and global motifs.

Icons: Blue Backgrounds and Purple Worlds
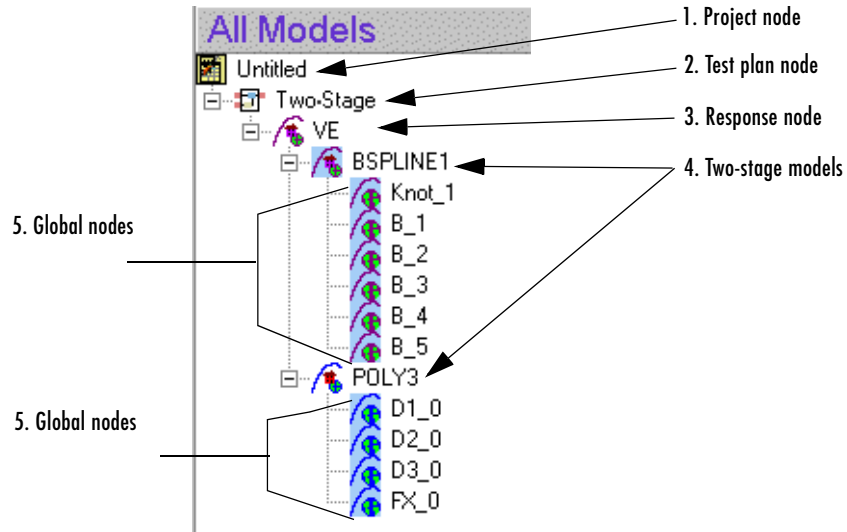
Icon changes convey information about modeling processes.

- When a model is selected as the *best model*, its icon changes color and gains a blue background, like the BSPLINE1 model in the preceding example.

- When the maximum likelihood estimate (MLE) is calculated and chosen as the best model, the associated model icon and all its child nodes (along with the plots of that model) become purple.

  You can see this in the preceding example: the B Spline model and all its response features have purple curves, globes, and house, indicating that they are MLE models. The Poly3 model and its children have blue curves and globes and a red house, indicating that they are univariate models.

- Observe the other difference between the B Spline and the Poly3 icons: the B Spline has a blue background. This indicates that this is selected as best model, and is used to calculate the two-stage model at the response node, so the response node is also purple. If an MLE model (with purple worlds) is selected as best model and is used to create the two-stage model, the response node always reflects this and is also purple.

- Notice also that the response features all have blue backgrounds. This shows they are selected as best and are all being used to calculate the two-stage model. In this case they are all needed. That is, a B Spline model needs six response features, and a Poly3 model requires four. If more response features are added, however, some combination must be selected as best, and the response features not in use do not have a blue background. An example is shown in the Quick Start Tutorial.

In the following example you can see child nodes of a global model. You can try different models within a global model, and you must select one of the attempts as best. In this example you can see that Cubic is selected as best, because it has a blue background, so it is the model being used for the Blow_2 global model.
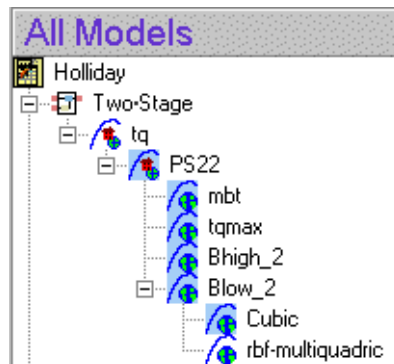
When a model is selected as best it is copied up a level in the tree together with the outliers for that model fit.

When a new global or local model is created the parent model and outliers are copied from the current level to the new child node. This gives a mechanism for copying outliers around the model tree.

A tree node is automatically selected as best if it is the only child, except two-stage models which are never automatically selected - you must use the **Model Selection** window.

If a best model node is changed the parent node loses best model status (but the automatic selection process will reselect that best model if it is the only child node).

Note Try the Quick Start tutorial to understand how to use the model tree. See "Tutorial: Quickstart" on page 2-1. The last section, "Creating Multiple Models To Compare" on page 2-42, guides you through the process of creating a variety of models of different types and how to understand the information in the model tree. You need to complete the previous sections of the tutorial first, which guides you through setting up a single two-stage model to get started.

Test Plans

You need to select a test plan to construct one-stage or two-stage models.

You can select the one- or two-stage test plans provided, as described next. You can also use these to create your own test plan template so you can reuse the setup for one test plan with another set of data. See "Creating New Test Plan Templates" on page 7-18.

Creating a New Test Plan

To create a new test plan:

• Click **New** in the **Test Plans** pane (visible at startup and whenever the project node is selected in the model tree).

Alternatively, make sure the project node is selected first, and then do one of the following:

• Click the **New Test Plan** icon (🗋) in the toolbar.
• Select **File –> New Test Plan**.
• Press **Insert** immediately after clicking the tree.

These steps all open a dialog with the choice of One Stage or Two Stage test plans, or you can browse for other test plans (as new templates can be created and saved). See "Local Level" on page 7-80.

A new test plan node appears in the model tree. To view the new test plan:

Change to test plan level:

• Select the node 🗗 in the tree by clicking it.

  Alternatively, double-click the new test plan in the **Test Plans** pane, as in Windows Explorer.

The Model Browser changes to test plan level, showing the block diagram representations of models in the main display, and the **Test Plans** pane changes to the **Response Models** pane (empty until models are set up).

For the next steps in model construction, see

• "Setting Up Models" on page 7-30
• "Setting Up Inputs" on page 7-38
• "Loading Data from File" on page 6-10

• "Selecting Models" on page 7-111

Creating New Test Plan Templates

You build user-defined templates from existing test plans using the **Make Template** toolbar icon 🗗 or **TestPlan –> Make Template**.

The procedures for modeling engines for calibrations are usually repeated for a number of different engine programs. The test plan template facility allows you to reuse the setup for one test plan with another set of data. You can alter the loaded test plan settings without restriction.

A list of test plan templates is displayed when you build a new test plan. There are built-in templates for one- and two-stage models.

Test plan templates store the following information:

• Hierarchical model — Whether the model is one- or two-stage and the default models for each level.

• Designs — *If* they were saved with the template (check box in the **Test Plan Template Setup** dialog)

  The design for one type of engine might or might not be appropriate for a similar type of engine. You can redefine or modify the design using the Design Editor.

• All response models (for example, torque, exhaust temperature, emissions) — *If* they were saved with the template (check box in the **Test Plan Template Setup** dialog)

• Numbers and names of input factors to models

• Model types (local and global)

• Summary Statistics for display (see "Summary Statistics" on page 7-24)

• No model child nodes are included, just the top level of the test plan (response models, and local and global models for two-stage models).

The response models are automatically built after you assign data to the test plan; see "Using Stored Templates" on page 7-19.

Saving a New Template

From the test plan node that you want to make into a template:

- Click the **Make Template** toolbar icon or choose **TestPlan –> Make Template**.

  The templates are stored in the directory specified in the **File –> Preferences** dialog.

The **Test Plan Template Setup** dialog appears, in which you can change the name of the new template and choose whether to include designs and/or response models.

Using Stored Templates

- When you load a new test plan from the project node, any stored templates appear in the **New Test Plan** dialog.

  From the project node, select **File –> New Test Plan**, or use the toolbar button, or click **New** in the **Test Plans** pane.

- Selecting templates in the **New Test Plan** dialog displays all templates found in the directory specified in the **Preferences** dialog (**File** menu). Select by clicking to see the information on a particular template; the number of stages and factors is displayed in the **Information** pane. You can use the Browse button if the required template is not in the directory specified in the **File –> Preferences** dialog.

- Click **OK** to use the selected test plan template. The new test plan node appears in the model tree.

Use stored templates in exactly the same way as the default blank one- and two-stage templates. Models and input factors are already selected (including the response if that was saved with the template) so you can go straight to selecting new data to model. You can still change any settings and design experiments.

Double-click the Responses block to launch the Data Wizard and select data for the test plan. The response models are automatically built after selection of data.

7-19
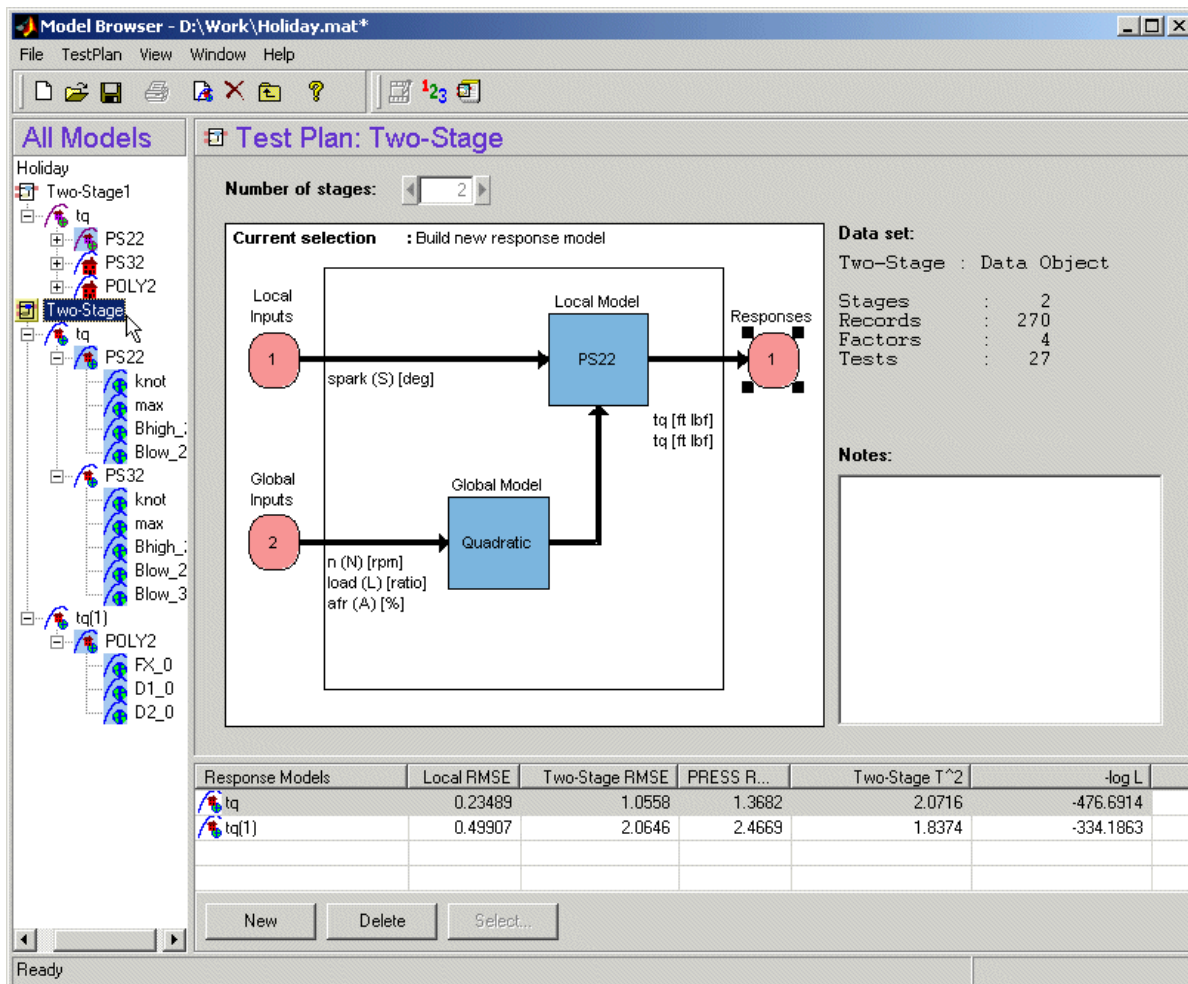
Note The data selection process takes you through the Data Wizard. If any signal names in the new data do not match the template input factors, you must select them here, including the responses. If signal names match the factor names stored in the template, they are automatically selected by the Data Wizard, and you just click **Next** all the way to the end of the wizard. When you click **Finish** the response models are built automatically.

Test Plan Level

When you select a test plan node (with the icon 🗂) in the model tree, this view appears.

This example is a two-stage model, the same model used in the Quick Start tutorial. All test plan nodes (one- and two-stage) bring up this view with a block diagram of the test plan and the functionality described below.

- At test plan level, the block diagram representations of models provides a graphical interface so you can set up inputs and set up models by double-clicking the blocks in the test plan diagram. These functions can also be reached using the **TestPlan** menu.

- You can access the Design Editor via the right-click menus on the model blocks or the **TestPlan** menu (for a particular model — you must select a model or input block before you can design an experiment). **View**–> **Design Data** also opens the Design Editor where you can investigate the design properties of the data. See "Designs" on page 5-1.

- You can attach data to a new test plan by choosing **TestPlan** –> **Select Data** or by double-clicking the Responses block in the diagram, which launches the Data Wizard (if the project already has data loaded).

- If a test plan already has data attached to it, you can reach the **Data Selection** views in the Data Editor using the **Select Data** toolbar button <img> or the **TestPlan** menu item. Here you can select data for modeling and match data to a design. For example, after the design changes, new data matching might be necessary. See "Matching Data to Designs" on page 6-37 for details.

- From the test plan level you can access the Boundary Constraint Modeling functionality. See "Boundary Constraint Modeling" on page 7-154.

- You can save the current test plan as a template using the **TestPlan** –> **Make Template** command or the toolbar button <img> . See "Local Level" on page 7-80.

See also "Test Plan Level: Toolbar and Menus" on page 7-27

Block Diagram Representations of Models

A block diagram in the test plan view represents the hierarchical structure of models. Following is an example of a two-stage test plan block diagram.
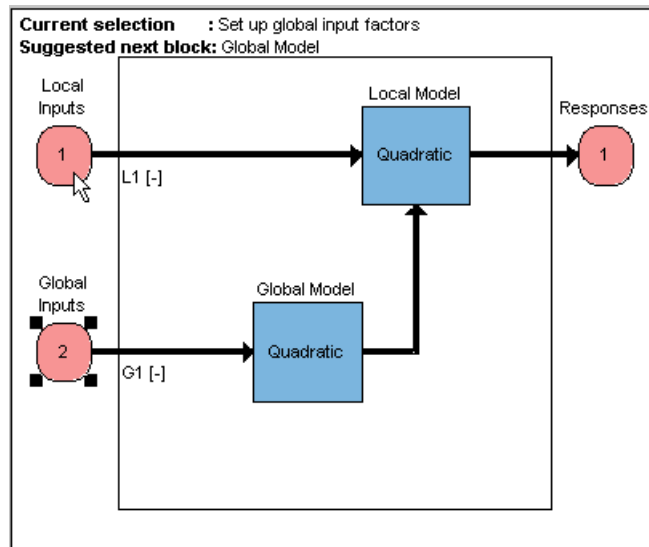
Functions Implemented in the Block Diagram

The diagram has functionality for setting up the stages in hierarchical modeling. At present MBC only supports one- and two-stage models.

1 "Setting Up Inputs" on page 7-38 — Setting the number of inputs for each stage of the model hierarchy.

2 "Setting Up Models" on page 7-30 — Setting up the new default models for each stage in the model hierarchy.

3 "Designing Experiments" on page 7-76 —Using the Design Editor.

4 "Loading Data from File" on page 6-10 and "New Response Models" —

5 "Viewing Designs" on page 7-76 — Viewing the statistical properties of the collected data.

The selected Model block is highlighted in yellow if a **Setup** dialog is open; otherwise it is indicated by blocks at the corners. The selected Model block indicates the stage of the model hierarchy that is used by the following functions:

- Set Up Model
- Design Experiment
- View Design Data
- View Model
- Summary Statistics

You can reach these functions via the right-click context menu (on each block) or the menus.

Summary Statistics

Right-click on the global model block and select **Summary Statistics** to reach the **Summary Statistics** dialog. Here you can choose which summary statistics you want displayed to help you evaluate models. The standard summary statistics are PRESS RMSE (for linear models only) and RMSE, and these are always displayed. You can choose additional ones here by selecting the check boxes. You can also reach the dialog using the **Model** menu from any global model node.
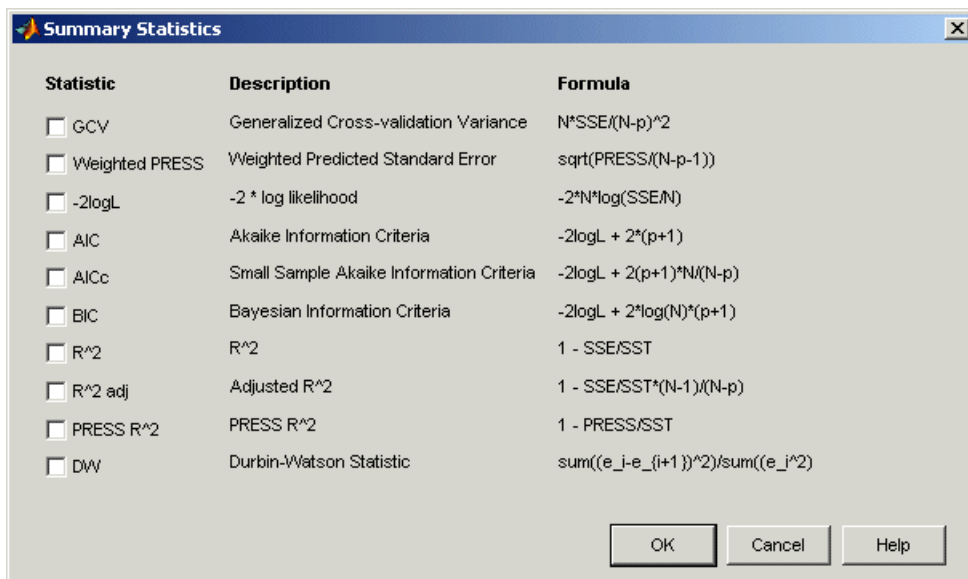
The choices you make in the **Summary Statistics** dialog determine what summary statistics are displayed in the global model views and the Model Selection window. These choices also determine what criteria you can use for automatically selecting best models in the **Local Model Setup** dialog for Multiple Models, and the **Build Models** dialog.

Available summary statistics are:

- GCV, Weighted PRESS, -2LogL, AIC, AICc (small sample), BIC, R^2, R^2 adj, PRESS R^2, and DW — full names and formulae are given in the dialog.

- The three Information Criteria (AIC, AICc (small sample) and BIC) are only available for comparison if exactly the same data is used for all models (same outliers, same y transform). The values of these will still appear for individual model display, but will disappear from the models list for comparison if, for example, you used a different Box-Cox transform.

    - The differences between two Information Criteria are of interest, not the absolute value. As a rule of thumb a difference of 2 or less means models are roughly the same quality of fit.

- All the Information Criteria impose a penalty for overfitting, which increases linearly (approximately linearly for AICc) with the number of parameters in the model.
- As a rough guide, typical models favoured by BIC are less complicated than those chosen by AIC or AICc.
- AICc (small sample) is designed for cases where parameters/observations < 40. Use AIC or AICc but not both.

  See "Model Selection Guide" on page 7-113 for some guidelines and "Information Criteria" on page 8-27 for statistical background.

• The Durbin-Watson statistic measures the correlation of adjacent observations, usually the residuals of a time series regression. A value of 2.0 for the Durbin-Watson statistic suggests that there is no time correlation.

  e_i is the residual at time i.

Summary statistics are inherited from the test plan node or the parent node on creation of a child node. You can select them from individual global models, or select them from the test plan before building models if you want them to apply to all the models within the test plan.

Changes made using the **Model** menu from a global model node are applied immediately to the Summary table in the global view, but not applied to the Models list pane until you change the model setup or build a new response model. Summary statistic selections appear in the Models list pane only when there are child nodes to compare. Resetting a model to the default test plan model also resets the summary statistics.

For definitions of any of the terms in the Summary Statistics formulae, see "Definitions" on page 8–4. Note that for ordinary least squares cases, 'p' is the number of parameters, but for non-ordinary least squares cases (rols and ridge least squares) 'p' is the effective number of parameters (p = N-df).

Test Plan Level: Toolbar and Menus
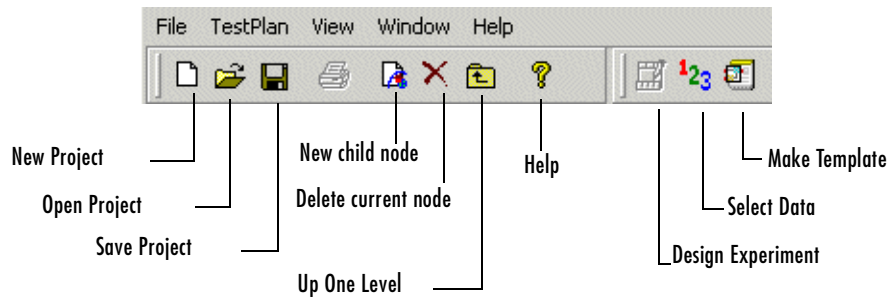
The eight buttons on the left (project and node management, plus the Print and Help buttons) appear in every view level. See "Project Level: Toolbar" on page 7-7 for details.

The right buttons change at different levels.

In the test plan level view, the right buttons are as follows:

- **Design Experiment** opens the Design Editor. Only available when a model or input has been selected in the test plan block diagram. You must specify the stage (local or global) you are designing for. See "Designs" on page 5-1.
- **Select Data** opens the Data Wizard, or opens the **Data Selection** views in the Data Editor if data sets have already been selected. See "Data" on page 6-1.
- **Make Template** opens a dialog to save the current test plan as a template, including any designs and response models. See "Local Level" on page 7-80.

Test Plan Level: Menus

File Menu
Only the **New** (child node) and **Delete** (current node) functions change according to the node level currently selected. Otherwise the **File** menu remains constant. See "File Menu" on page 7-8.

Window Menu

The **Window** menu remains the same throughout the Model Browser. It allows you to switch windows if there is more than one toolbox window open. See "Window Menu" on page 7-10.

Help Menu

The **Help** menu remains the same throughout the Model Browser. You can always reach the MBC Toolbox Help Roadmap by selecting **Help –> MBC Help**. The context help takes you to relevant Help pages, and **Help –> About MBC** shows the version notes. See "Help Menu" on page 7-10.

TestPlan Menu

- **Set Up Inputs** — See "Setting Up Models" on page 7-30.
- **Set Up Model** — See "Setting Up Models" on page 7-30.

  You can also reach these two functions by double-clicking the blocks in the test plan diagram, and both can only be used when a Model block is first selected in the diagram. You must specify the model to set up, local or global.

- **Design Experiment** — See "The Design Editor" on page 5-2.

  This is also available in the toolbar and in the right-click context menu on the blocks in the test plan diagram.

- **Boundary Constraints** — Opens the **Constraint Modeling** window. See "Boundary Constraint Modeling" on page 7-154.
- **New Data** — Opens the Data Editor to load new data.
- **Select Data** — Opens the **Data Selection** views of the Data Editor.

  You can reach both these functions with the toolbar **Select Data** button. If no data is selected, this button opens the Data Wizard, and if a data set is already selected, it takes you straight to the **Data Selection** views. See "Data" on page 6-1.

- **Make Template** — Opens a dialog for saving the current test plan as a new template, with or without designs and response models. Same as the toolbar button. See "Local Level" on page 7-80.
- **Export Multimodels** — This provides a smooth interface with the Multimodel Tradeoff in the CAGE browser part of the Model-Based Calibration toolbox. Two global inputs are required (normally speed and load). This item is only enabled if you have set up a two-stage model with the correct number of inputs. This useful application for multiple models allows

you to calibrate from local maps. See "Local Model Class: Multiple Models" on page 7-70 for details.

View Menu (Test Plan Level)

- **Design Data** — Opens the Design Editor. The view design facility enables you to investigate the statistical properties of the collected data. This provides access to all the Design Editor and design evaluation utility functions with the current design rather than the prespecified design (after data matching, the data points are used as the new design points). See "The Design Editor" on page 5-2.

  For two-stage models, viewing level 1 designs creates a separate design for each test.

- **Model** — Opens a dialog showing the terms in the current model.

Both of these are only available when a model or input block is selected in the test plan block diagram.

Setting Up Models

The following are one-page overviews for a quick guide to setting up one-stage and two-stage models, then how to search for the best fit:

These sections cover setting up inputs and models in detail:

• "Designing Experiments" on page 7-76 tells you how to proceed to experimental design after setting up models.

What Models Are Available?

Model Type	One Stage	Two Stage: Global or 2nd Stage	Two Stage: Local or 1st Stage
Linear model	Yes	Yes	Yes
Radial Basis Function	Yes	Yes	
Hybrid RBF	Yes	Yes	
Multiple linear models	Yes	Yes	
Free knot spline	Yes, one factor only	Yes, one factor only	Yes, one factor only
Neural net (requires Neural Nets Toolbox)	Yes		
Average fit			Yes
Multiple models*			Yes
Growth models			Yes, one factor only
Polynomial**			Yes, one factor only
Polynomial spline**			Yes, one factor only
Truncated power series			Yes, one factor only
User defined#	Yes, one factor only		Yes
Transient#	Yes, one factor must be time		Yes

*Multiple Models gives you access to global model types for your local model.

** Polynomial and polynomial spline are two special case linear models for local models with one input factor. You can use polynomial and polynomial spline models (with more settings) for local models with more factors by choosing Linear Models.

User defined and transient models must be checked into the toolbox before you can use them. They will be available as local models for the number of factors specified. There is an example user-defined model for a single factor preregistered with the toolbox. The example transient model provided must have exactly two factors, one of which must be time. See "User-Defined Models" on page 8–61 and "Transient Models" on page 8–68 for details.

Instant One-Stage Model Setup

The following steps are necessary to set up a one-stage model:

1 From the project node, create a new one-stage test plan. See "Creating a New Test Plan" on page 7-17.

2 Select the new node in the model tree to change to the test plan level.

3 Set up the inputs and model type by double-clicking the Inputs block and the Model block in the test plan diagram. See "Setting Up Inputs" on page 7-38 and "Setting Up Models" on page 7-30.

4 At this point, you might want to design an experiment. See "The Design Editor" on page 5-2.

5 From the test plan node, load a new data set to use. Choose **TestPlan –> Select Data**, which opens the Data Wizard. Click **Load new data set** and the Data Import Wizard appears. See "Loading Data from File" on page 6-10. Use the Data Wizard to match up data signals with model inputs and outputs.

6 Dismissing the Data Wizard opens the Data Editor, where you can select data for modeling and match data to designs. See "The Data Editor" on page 6-4.

On closing the Data Editor the model fit is calculated and the new model node appears in the model tree. Select the new node to view the model fit.

Functionality available for viewing and refining the model fit is described in "Global Level" on page 7-97 and "Selecting Models" on page 7-111.

Once you have built a single model, you should create more models for comparison, in order to search for the best fit. You can follow the guidelines in "Creating Multiple Models To Compare" on page 7-36.

Instant Two-Stage Model Setup

The following steps are necessary to set up a two-stage model:

1 From the project node, create a new two-stage test plan. See "Creating a New Test Plan" on page 7-17.

2 From the test plan node, set up the inputs and models at the local and global stages. See "Setting Up Models" on page 7-30 and "Setting Up Models" on page 7-30.

3 At this point, you might want to design an experiment. See "The Design Editor" on page 5-2.

4 From the test plan node, load the data set you want to use. Select **TestPlan –> Select Data**. See "Loading Data from File" on page 6-10.

This opens the Data Wizard, where you can load a data set, match data signals to model variables and then set up the response model.

5 On completing the Data Wizard, the Data Editor opens. Here you can select data for modeling and match data to designs. See "The Data Editor" on page 6-4. Close the Data Editor and click **Yes** to accept the data for modeling.

Note On closing the Data Editor, the local and global models are calculated.

---

6 At the local node, you can view the fit of the local models to each test, and you can also view the global models at the response feature nodes.

7 The two-stage model is not calculated until you use the **Select** button (from the local node, in the **Response Features** pane) and choose a model as best (even if it is the only one so far), *unless* you go straight to MLE. See below.

See "Selecting Models" on page 7-111.

Note At this point, the two-stage model is calculated, and the icon changes at the local node to reflect this. See "Icons: Curves, Worlds, and Houses" on page 7-13.

8 You are prompted to calculate the maximum likelihood estimate (MLE) at this point. You can do this now, or later by selecting **Model –> Calculate MLE**. See "MLE" on page 7-137 for a detailed explanation.

Note If there are exactly enough response features for the model, you can go straight to MLE calculation without going through the **Select** process. The **MLE** toolbar button and the **Model –> Calculate MLE** menu item are both active in this case. If you add new response features, you cannot calculate MLE until you go through model selection to choose the response features to use.

See "Two-Stage Models for Engines" on page 8-40 for a detailed explanation of two-stage models.

Double-click the Model blocks of the block diagram or select the **TestPlan –> Set Up Model** menu item. MBC supports a wide range of models.

For model descriptions, see "Global Model Setup" on page 7-40 and "Local Model Setup" on page 7-60.

For further statistical details, see "Technical Documents" on page 8-1.

Once you have built a single model, you should create more models for comparison, in order to search for the best fit. You can follow the guidelines in the next section, "Creating Multiple Models To Compare" on page 7-36.

Creating Multiple Models To Compare

Once you have fitted and examined a single model (either one- or two-stage), you will normally want to create more models in order to search for the best fit. You can:

- Create individual new models.
  - You can create new child nodes by clicking the **New** button from any modeling node. Then select the new child node and choose **Model –> Set Up Model** to reach 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.
- Create a template to save a variety of model settings for reuse.
  - From any global or one-stage model with child nodes, select **Model –> Make Template**. You can save the child node model types of your currently selected modeling node as a template. You can then use the **Build Models** dialog to find your user-defined templates and quickly build all those model types again for any global model you choose (see below).
  - From any global model node (before calculating MLE), click **Build Models** in the toolbar. You can save a template containing whatever models you choose by selecting **New** and adding the model types you want.
- Use the **Build Models** function to create a selection of models at once. You can use the **Browse** button to find your stored templates that are not in the default directory, and there are predefined templates for polynomials and RBF kernels. All the child node model types in the template you select are built as child nodes of the currently selected global model.

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 that fit well to the data for each of your global models.

We provide a detailed tutorial example to guide you through using these techniques to make a number of models to compare. See "Creating Multiple Models To Compare" on page 2-42. This tutorial section requires you to have completed the previous steps in the Quick Start tutorial, so that you have an initial model as a starting point.

Note that you can click **New** from any modeling node to create new child nodes.

- For all global model nodes (one- and two-stage) a new child node appears that is a copy of the parent, and you can select **Model –> Set Up Model** to change the type and settings.

- For the other two-stage modeling nodes (local and response models) you also get a new child node but the procedure is slightly different — see the tutorial example for details.

Once you have a variety of models to compare, you should use the diagnostic statistics and powerful visual aid plotting capabilities of the Model Browser to help you decide which models are best. For a description of the views and statistics available in each modeling view, see:

- "Selecting Models" on page 7-111
- "Model Evaluation Window" on page 7-144
- "Local Level" on page 7-80
- "Global Level" on page 7-97
- "Response Level" on page 7-141

For linear models, make use of the Stepwise functions (open the **Stepwise** window for existing models, and/or choose a **Stepwise** option during model setup) to refine your models and remove less useful model terms. Make sure you examine outliers but do not automatically remove them without good reason. Pay attention to the diagnostic statistics to help you find the best models. The following sections describe how to use Stepwise to make better models and how to understand the diagnostic statistics:

- "Model Selection Guide" on page 7-113 for guidelines.
- "Stepwise" on page 7-47 for an introduction to Stepwise.
- "PRESS statistic" on page 8-23 — see this section for guidelines on what to look for in the statistics to indicate a good fit.
- "High-Level Model Building Process Overview" on page 8-9
- "Univariate Model Building Process Overview" on page 8-10
- "Stepwise Regression Techniques" on page 8-13

Setting Up Inputs

You set up the number and definition of model input factors for each stage by double-clicking or right-clicking the inports of the block diagram.

You can create a block diagram by clicking **New** in the **Test Plans** pane at project level (the view where you begin). See "Creating a New Test Plan" on page 7-17. Then double-click the blocks labeled inputs.

See "Functions Implemented in the Block Diagram" on page 7-23.
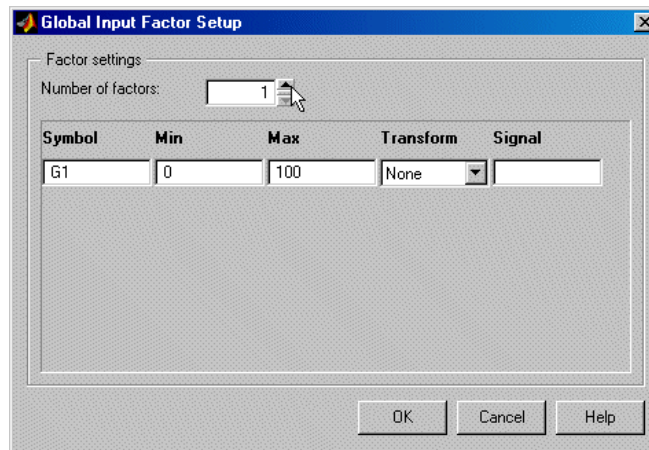
The preceding example shows the input setup dialog for the global model. The dialog for the local model contains exactly the same controls.

Number of Factors

You can change the number of input factors using the buttons at the top.

Symbol

The input symbol is used as a shortened version of the signal name throughout the application. The symbol should contain a maximum of three characters.

Min and Max Model Range

This setting is important before you design experiments. The default range is [0.100]. There is usually some knowledge about realistic ranges for variables.

If you are not designing an experiment you can use the data range as the model range later, in the data selection stage. In some cases you might not want to use the data range (for example, if the data covers too wide a range, or not wide enough) if you are interested in modeling a particular region. In that case you can set the range of interest here.

Transform

You can use input transformations to change the distribution of the input factor space for designing experiments. The available input transformations are `1/x`, `sqrt(x)`, `log10(x)`, `x^2`, `log(x)`. See "Transforms" on page 7-75 for a discussion of the uses of transformations.

Signal

You can set up the signal name in the input factor setup dialog. It is not necessary to set this parameter at this stage, as it can be defined later at the data selection stage (as with the range). However, setting the signal name in this dialog simplifies the data selection procedures, as the Model Browser looks for matching signal names in loaded data sets. When the number of data set variables is large this can save time.

Global Model Setup

The following example shows the default settings for the global model.
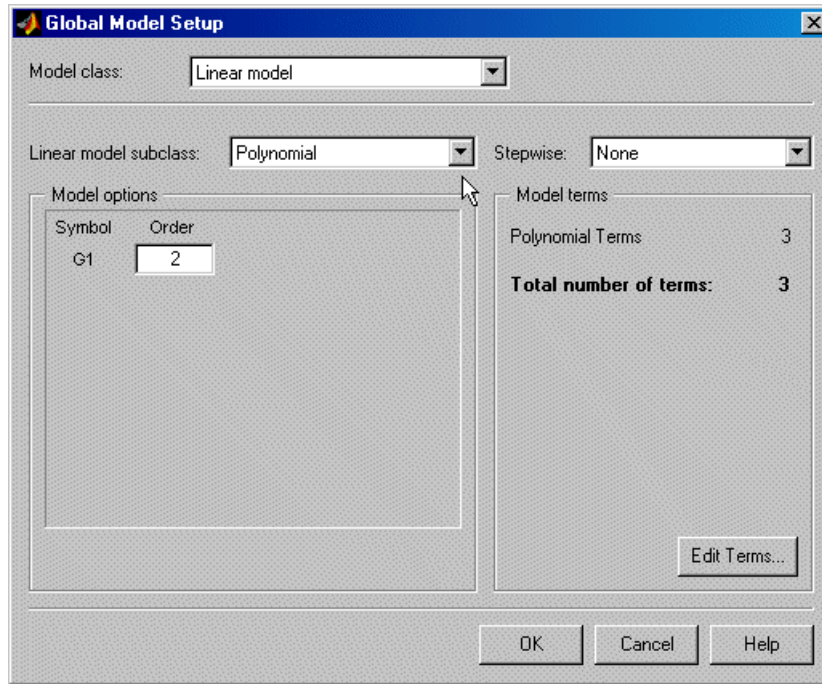
Global Model Classes
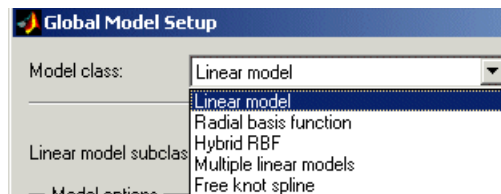
The default global model is linear polynomial. There are a number of different global model classes available, covered in the following sections:

- "Global Linear Models: Polynomials and Hybrid Splines" on page 7-42
- "Global Model Class: Radial Basis Function" on page 7-49

- "Global Model Class: Hybrid RBF" on page 7-54
- "Global Model Class: Multiple Linear Models" on page 7-55
- "Global Model Class: Free Knot Spline" on page 7-56

Note that some other models are also available for one-stage models:
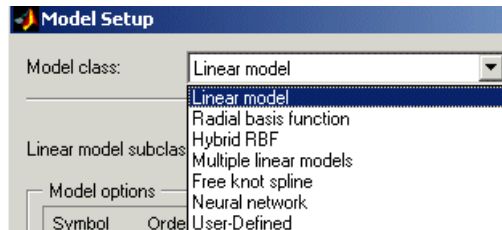
Of these, neural network models are only available for one-stage models. The single preregistered user-defined model is only available for a single input. You can check in your own user-defined models and transient models with as many factors as you choose; these only appear as options when the appropriate number of inputs are present. Previous versions of the toolbox offered the example user-defined model (single input only) for the local level of two-stage models, but not for one-stage models. Now both user-defined and transient models can also be used for one-stage models.

- "Global Model Class: Neural Network" on page 7-58
- "Global Model Class: User-Defined and Transient Models" on page 7–59

"Linear Model Statistics Displays" on page 8-22 has a description of the statistics displayed when this type of model is analyzed.

See "Setting Up Models" on page 7-30 for a list of all information about setting up local and global models.

Global Linear Models: Polynomials and Hybrid Splines

Global linear models can be

- Polynomial
- Hybrid Spline

Polynomials

Polynomials of order n are of the form

$$\beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 \ldots \beta_n x^n$$

You can choose any order you like for each input factor.
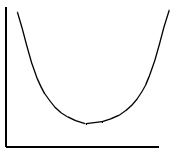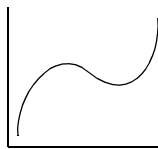
Quadratic curve   Cubic curve

As shown, a quadratic polynomial $y = ax^2 + bx + c$ can have a single turning point, and a cubic curve $y = ax^3 + bx^2 + cx + d$ can have two. As the order of a polynomial increases, it is possible to fit more and more turning points. The curves produced can have up to $(n\text{-}1)$ bends for polynomials of order n.

Term Editor

Click the **Edit Terms** button to see the terms in the model. This opens the **Term Editor** dialog. Here you can remove any of the terms (see example).
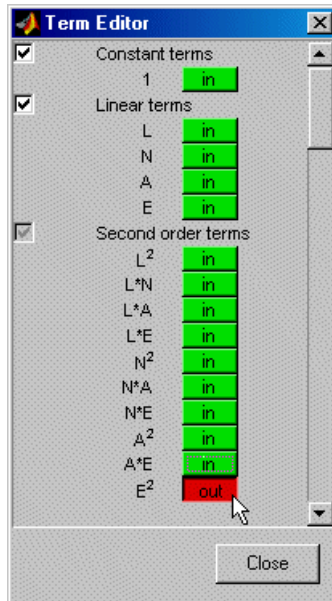
See also "Interaction" on page 7-46.

Hybrid Splines

You can use the Hybrid Spline model to fit a spline to one factor and polynomials to all other factors.

A spline is a piecewise polynomial function, where different sections of polynomials are fitted smoothly together. The locations of the breaks are called knots. You can choose the required number of knots (up to a maximum of 50) and their positions. In this case all the pieces of curves between the knots are formed from polynomials of the same order. You can choose the order (up to 3).

The example following illustrates the shape of a spline curve with one knot and third-order basis functions. The knot position is marked on the N axis.
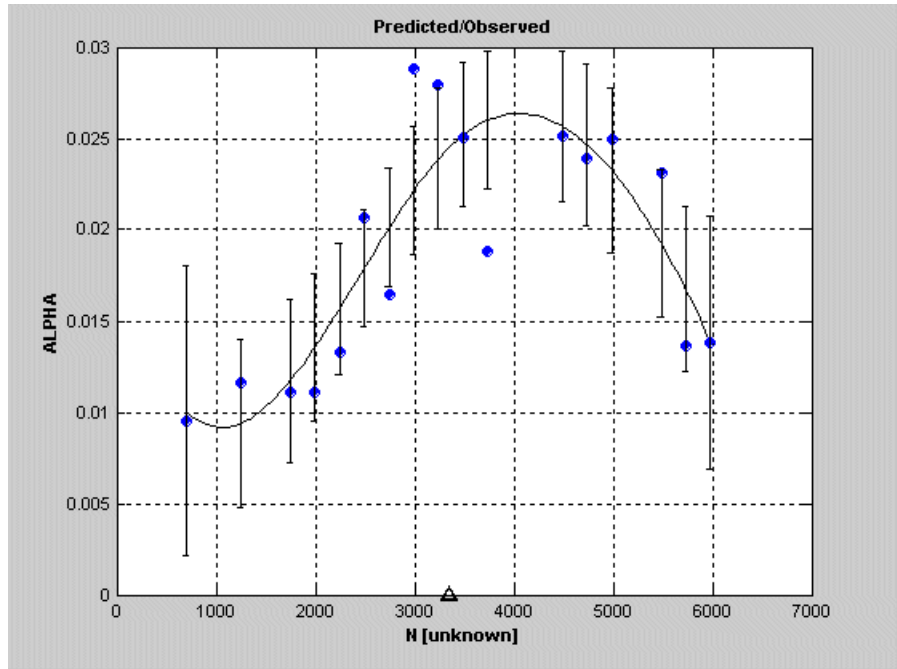
You can fit more complicated curves using splines, so they can be useful for the factor you expect to behave in the most complex way. This allows you to model detailed fluctuations in the response for one factor, while simpler models are sufficient to describe the other factors. The following example clearly shows that the response (Blow_2 in this case) is quadratic in the Load (L) axis and much more complex in the RPM (N) axis.
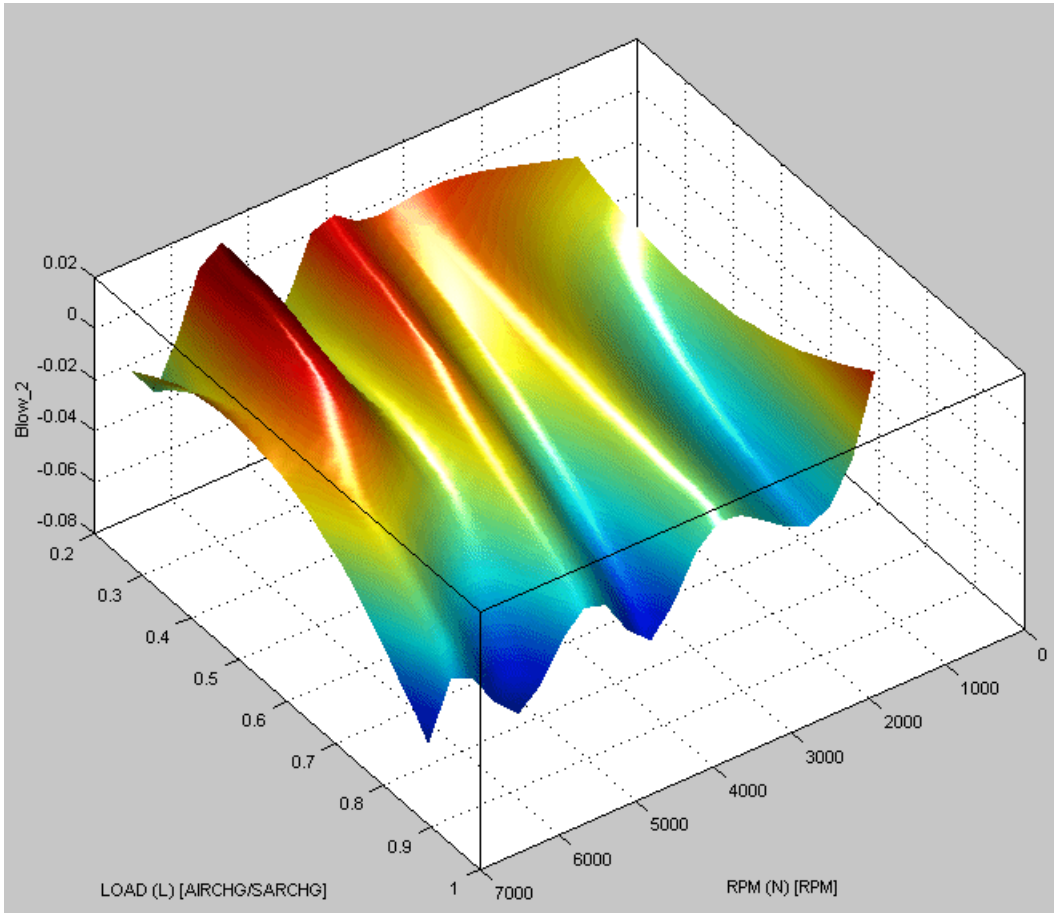
You can choose the order of the polynomial for each factor and the factor to fit the spline to. The maximum order for each factor is cubic.

The following example shows the options available for the Hybrid Spline linear model subclass.
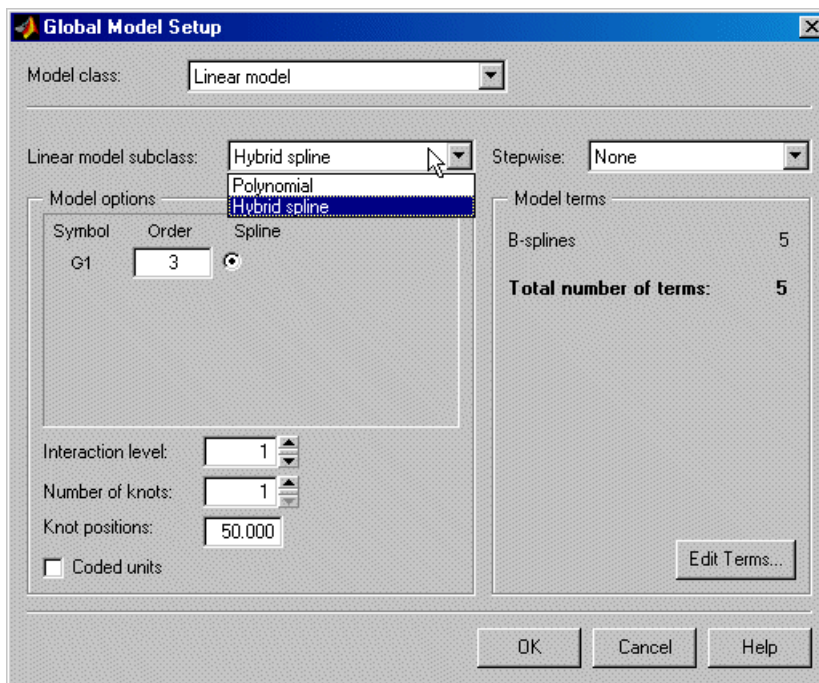
This example only shows one input factor. When there are more, you use the radio buttons to select which factor is modeled with a spline. Select the order for each factor in the edit boxes.

See also the other linear model subclass, "Polynomials" on page 7-42.

Interaction

You can choose the interaction level on both linear model subclasses (polynomial and hybrid spline). For polynomials, the maximum interaction level is the same as the polynomial order (for example, 3 for cubics). For hybrid splines, the maximum interaction level is one less than the maximum order of the polynomials.

The interaction level determines the allowed order of cross-terms included.

You can use the Term Editor to see the effects of changing the interaction level. Click the **Edit Terms** button. The number of constant, linear, second- and third-order (and above) terms can be seen in the **Model Terms** frame.

For polynomials: With an interaction level of 1, there are no terms in the model involving more than one factor. For example, for a four-factor cubic, for factor L, you see the terms for L, L^2, and L^3, but no terms involving L *and* other factors. In other words, there are no cross-terms included.

If you increase the interaction level to 2, under second-order terms you see L^2 and also L multiplied by each of the other factors; that is, second-order cross-terms (for example, LN, LA, and LS).

Increase the interaction to 3, and under third-order terms you see L^2 multiplied by each of the other factors (L^2N, L^2A, L^2S), L multiplied by other pairs of factors (LNA, LNS, LAS), and L multiplied by each of the other factors squared (N^2L, A^2L, S^2L). Interaction level 3 includes all third-order cross-terms.

The preceding also applies to all four factors in the model, not just L.

For hybrid splines: The interaction function is different: it refers only to cross-term interactions between the spline term and the other variables. For example, at interaction order 1, raw spline terms only; interaction 1, raw terms and the spline terms x the first-order terms; interaction 3, includes spline terms x the second-order terms; and so on.

Stepwise
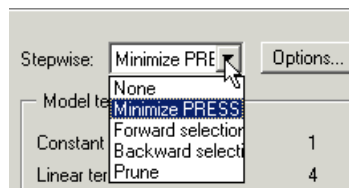
Take care not to overfit the data; that is, you do not want to use unnecessarily complex models that "chase points" in an attempt to model random effects.

The **Stepwise** feature can help. Stepwise will select the terms using various criteria. Stepwise generally means that terms are removed in steps (one at a time). The stepwise algorithms are Minimize Press, Forward Selection, Backwards Selection, and Prune. The most commonly used stepwise algorithm is `Minimize PRESS`, where at each step the term that will improve the PRESS statistic the most is removed or included. `Minimize PRESS` throws away terms in the model to improve its predictive quality, removing those terms that

reduce the PRESS of the model. Forward and Backwards Selection uses statistical significance at the alpha % level.

Predicted sum of squares error (PRESS) is a measure of the predictive quality of a model. See "PRESS statistic" on page 8-23 for an explanation of PRESS and why it is useful as a diagnostic statistic.

Prune is one of the alternative algorithms for stepwise. The order of the terms matter, and the terms will only be removed from the end, provided they improve the quality of the fit (measured by various criteria: PRESS, GCV etc.). The other stepwise algorithms do not have this restriction - they can remove any term in any order. Removing terms only from the end is valid when there is ordering in the terms e.g. polynomials (from low order terms to high order terms) or rbfs where the fitting algorithms select the most important terms first.

Click **Options** to open a dialog containing further settings for the selected **Stepwise** option. You should choose from the list a criteria for removing terms (PRESS, RMSE, AIC, BIC etc.). For the Prune settings, see "Global Model Class: Radial Basis Function" on page 7-49. For a guide to all the settings in the **Stepwise** window (which explains the other Stepwise settings available here), see "Stepwise Regression Techniques" on page 8-13. Note you can also use the **Stepwise** window after model fitting to try other Stepwise settings, and replace excluded model terms if you want.

Global Model Class: Radial Basis Function

A variety of radial basis functions (RBFs) are available in MBC. They are all radially symmetrical functions that can be visualized as mapping a flexible surface across a selection of hills or bowls, which can be circular or elliptical.

Networks of RBFs can model a wide variety of surfaces. You can optimize on the number of centers and their position, height and width. You can have different widths of centers in different factors. RBFs can be very useful for investigating the shapes of surfaces when system knowledge is low. Combining several RBFs allows complicated surfaces to be modeled with relatively few parameters. The following example shows a surface of an RBF model.
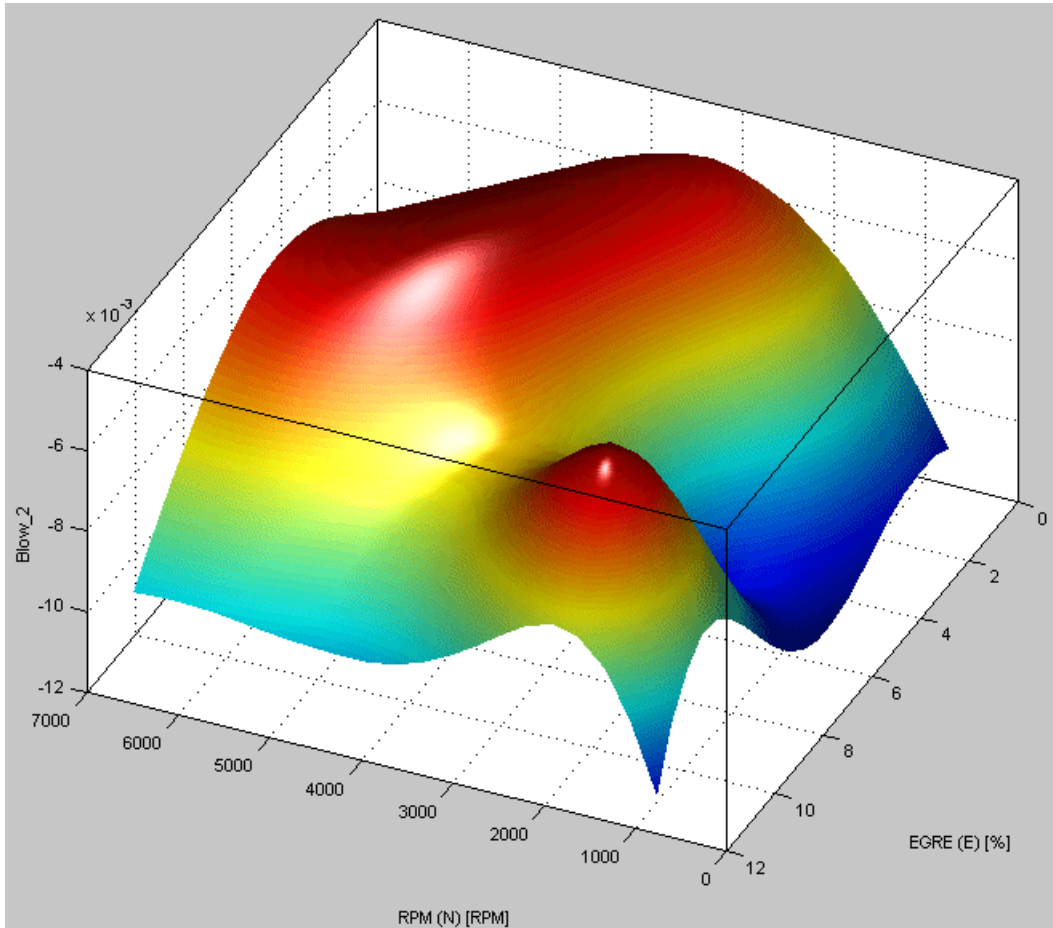
There is a detailed user guide for modeling using RBFs in "Radial Basis Functions" on page 9-1. See especially "Tips for Modeling with Radial Basis Functions" on page 9-32 for guidelines and "Types of Radial Basis Functions" on page 9-3 for illustrations of different types.

The statistical basis for each setting in the RBF global models is explained in detail in "Guide to Radial Basis Functions for Model Building" on page 9-2.

The following example illustrates the types of RBF kernel available.
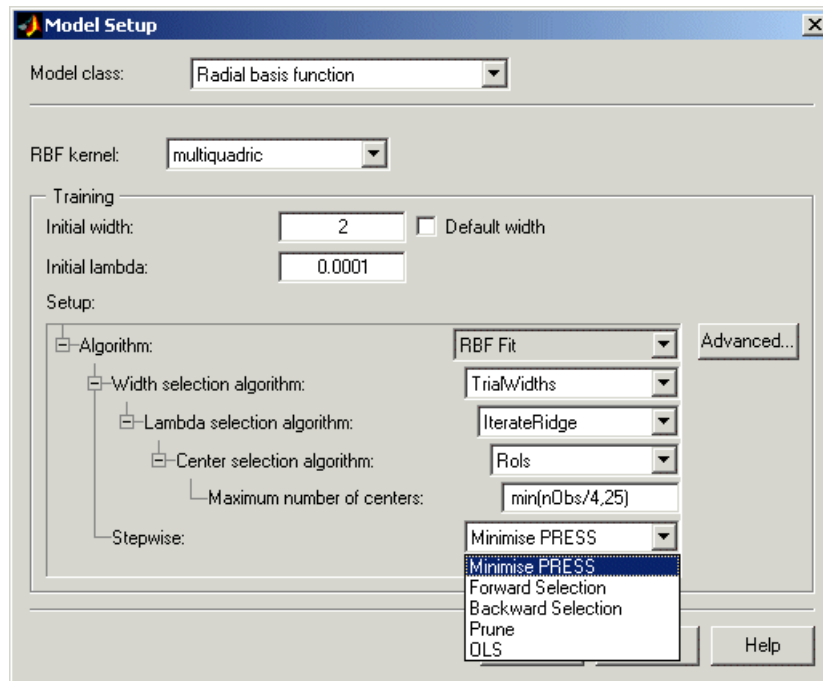
You can use the drop-down menus to set RBF kernel type and width and center selection algorithm. Once you have fitted a model once to get some idea of what to expect you can try different maximum numbers of centers as a useful method for homing in on a good fit. There are more options for fine tuning in the Advanced options dialog, but you can use the main controls from here to narrow down the search for the best model. See below.

You can use the last drop-down menu to choose to automatically run Stepwise at the end of the center/width selection algorithm to remove less useful model terms, Ordinary Least Squares (OLS) or the Prune algorithm to home in on the best number of centers (using your choice of the Summary Statistics as selection criteria). Note if you choose Prune here, there are further settings you need which can be found by clicking **Advanced**. This opens the **Radial Basis Function Options** dialog, as shown.
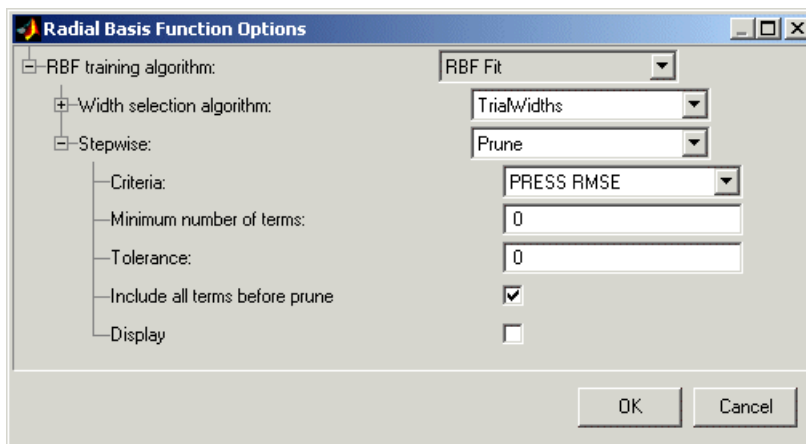
All the settings under Width selection algorithm are for fine tuning the RBF model. See "Tips for Modeling with Radial Basis Functions" on page 9-32 for guidelines, and details on specific parameters for different algorithms can be found here: "Radial Basis Functions" on page 9-1.

The options in the **Stepwise** drop-down menu are the same as the main **Model Setup** dialog — Min. PRESS, Forward, Backward, Prune and OLS (Ordinary Least Squares). If you choose Prune the further options appear below as shown in this example. You need to choose one of the Summary Statistics as selection criteria for the Prune algorithm. All the Summary Statistics options are available as criteria, and do not depend on your choices of these statistics in the **Summary Statistics** dialog. See "Summary Statistics" on page 7-24 for more information.

We recommend you select the check box to **Include all terms before prune** (otherwise the current number of terms is used at the start). You can choose a **Minimum number of terms**, and the **Tolerance** you set determines how far from this number of terms the algorithm can go — within the limits of the tolerance, the algorithm searches for fewer terms that reduce the value of your selection criteria. If you select the **Display** check box a figure appears illustrating the Prune process, like the example shown following, plotting the number of parameters against the selection criteria, in this case PRESS RMSE. You can use this information to see if you should change the minimum number of terms and the tolerance and refit, in case you need to avoid a local minimum.
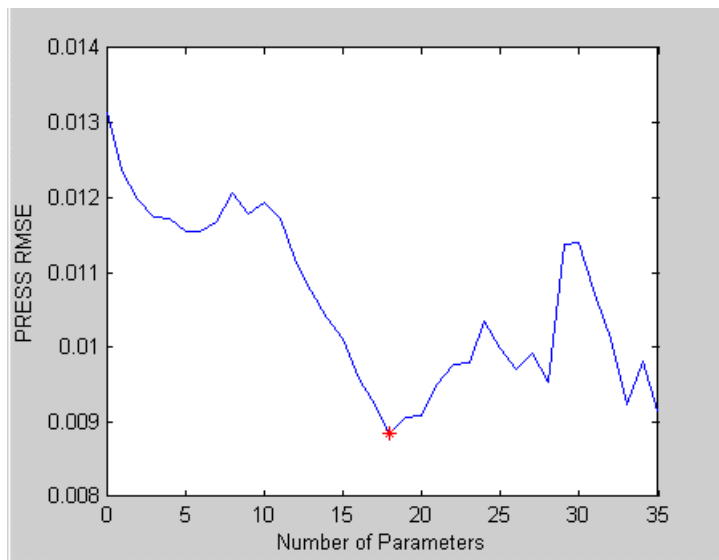
Global Model Class: Hybrid RBF

This option combines an RBF model with a linear model.
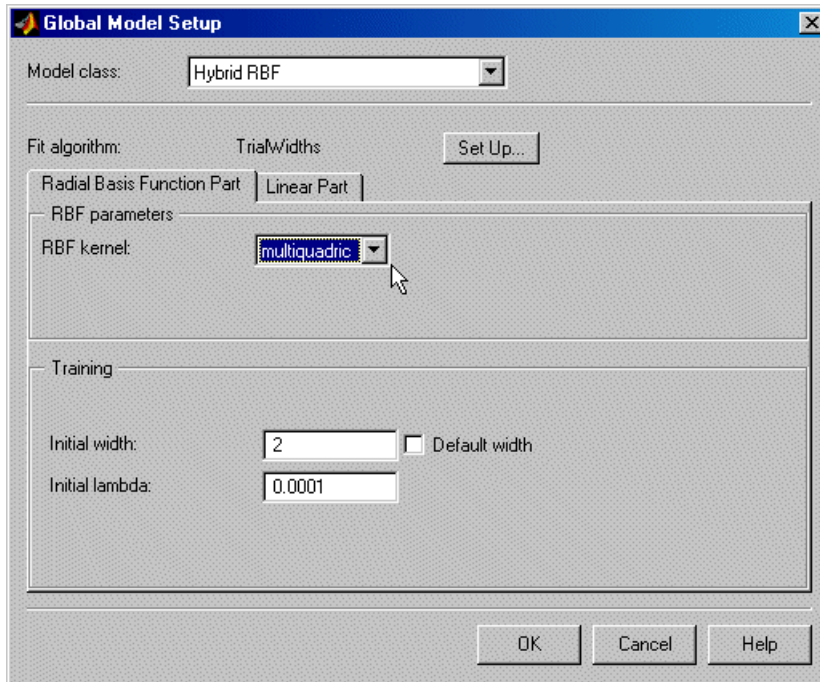
The **RBF kernel** drop-down menu offers the same options as for normal RBF.

The **Linear Part** tab contains the same options as the other global linear models; see "Global Linear Models: Polynomials and Hybrid Splines" on page 7-42.

See "Hybrid Radial Basis Functions" on page 9-30.

Click **Set Up** to reach the Hybrid RBF Options dialog where you can change all the settings for the RBF part of the model. Here you can also choose to automatically run Stepwise, OLS or Prune. These settings are the same as the **Radial Basis Functions Options** dialog, see "Global Model Class: Radial Basis Function" on page 7-49 for details.

See also "Radial Basis Functions" on page 9-1 for a detailed guide to the use of all the available RBFs in modeling.

Global Model Class: Multiple Linear Models
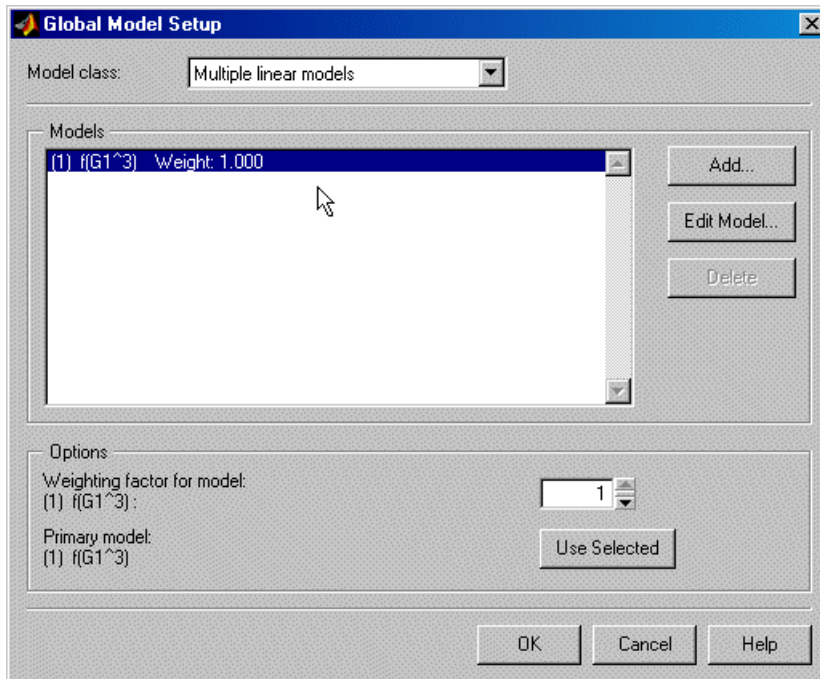
The preceding example shows the defaults for multiple linear models. You can add linear models (as seen in the single linear model settings).

This is primarily for designing experiments using optimal designs. If you have no idea what model you are going to fit, you would choose a space-filling design. However, if you have some idea what to expect, but are not sure exactly which model to use, you can specify a number of possible models here. The Design Editor can average optimality across each model.

For example, if you expect a quadratic and cubic for two factors but are unsure about a third, you can enter several alternative polynomials here. You can change the weighting of each model as you want (for example, 0.5 each for two models you think equally likely). This weighting is then taken into account in the optimization process in the Design Editor.

The model that appears in the model tree is the one you select, listed as **Primary model**. Click the model in the list, then click **Use Selected**. The

Primary model changes to the desired model. If you do not select a primary model, the default is the first in the list.

When the model has been fitted, you can view the primary model at the global node. To compare the fit of all the alternatives, click **Build Models** in the toolbar, select **Multiple Linear Models** in the dialog, and click **OK**. One of each model is then built as a selection of child nodes.

See also "Polynomials" on page 7-42, and "Local Model Class: Multiple Models" on page 7-70.

Global Model Class: Free Knot Spline
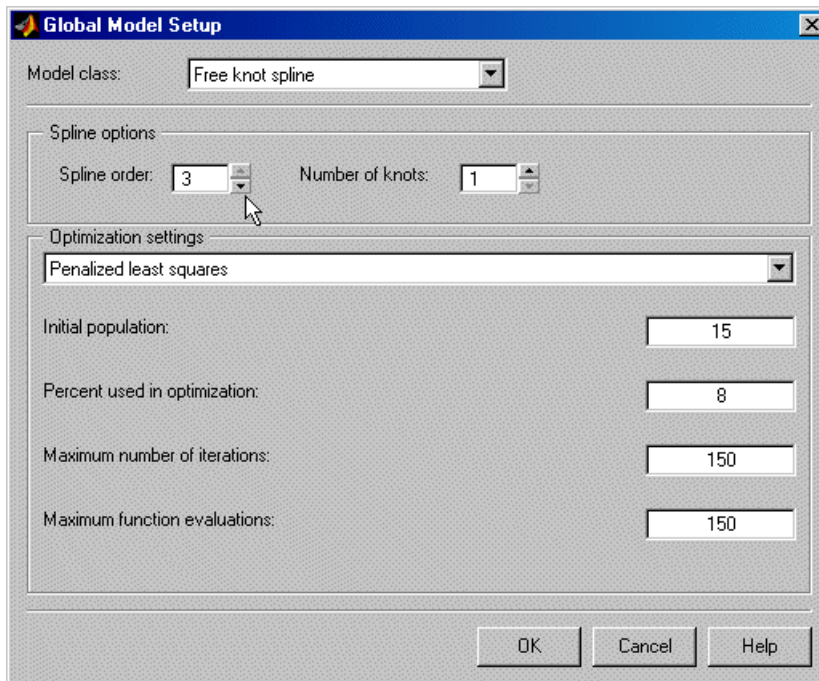
This option is only available for global models with only one input factor. See also "Hybrid Splines" on page 7-43 for a description of splines. The major difference is that you choose the position of the knots for hybrid splines; here the optimal knot positions are calculated as part of the fitting routine.

You can set the number of knots and the spline order can be between one and three.

There are three different algorithms under **Optimization settings**: `Penalized least squares`, `Genetic algorithm`, and `Constrained least squares`.

For all three methods, you can set the **Initial population**. This is the number of initial guesses at the knot positions. The other settings put limits on how long the optimization takes.

The example following shows a free knot spline model with three knots. The position of the knots is marked on the `N` axis.
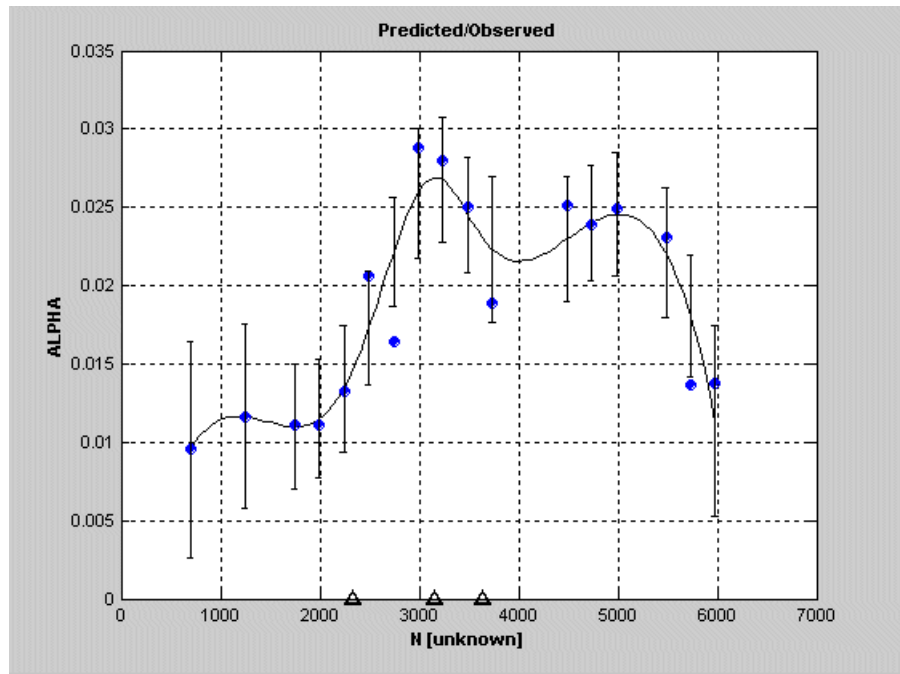
See also the local models involving splines:

- "Polynomial Spline" on page 7-62
- "Local Model Class: Truncated Power Series" on page 7-64
- "Local Model Class: Free Knot Spline" on page 7-65

Global Model Class: Neural Network

Neural networks are only available for one-stage models.

Neural nets contain no preconceptions of what the model shape will be, so they are ideal for cases with low system knowledge. They are useful for functional prediction and system modeling where the physical processes are not understood or are highly complex.

The disadvantage of neural nets is that they require a lot of data to give good confidence in the results, so they are not suitable for small data sets. Also, with higher numbers of inputs, the number of connections and hence the complexity increase rapidly.

MBC provides an interface to some of the neural network capability of the Neural Network Toolbox. Therefore these functions are only available if the Neural Network Toolbox is installed. See "Neural Networks" on page 8-60.
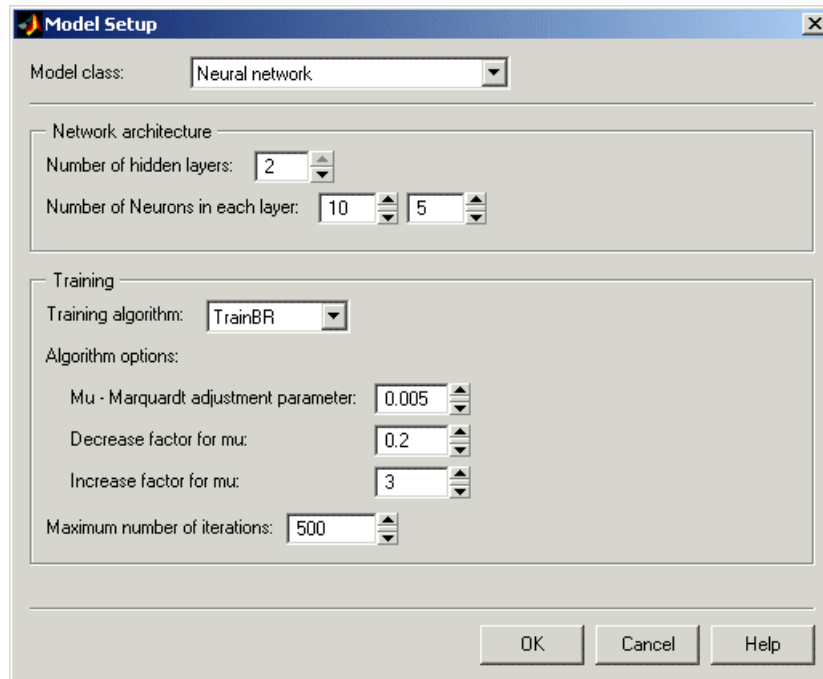
Global Model Class: User-Defined and Transient Models

These models can be one-stage models, or local model types for two-stage modeling. For set up information see "Local Model Class: User-Defined Models" on page 7–73 and "Local Model Class: Transient Models" on page 7–73.

Local Model Setup
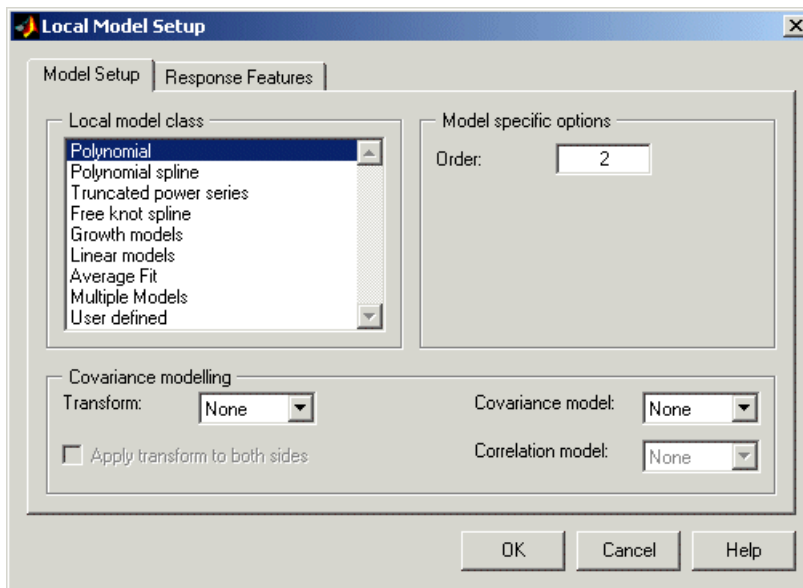
The preceding example shows the default settings for the local model.

Local models can be of the following types:

- Polynomial
- Polynomial Spline
- Truncated Power Series
- Free Knot Spline
- Growth Models
- Linear Models
- Average Fit
- Multiple Models
- User-Defined Models

You can choose additional response features at this stage using the **Response Features** tab. These can also be added later. The Model Browser automatically chooses sufficient response features for the current model.

See also

- "Covariance Modeling" on page 7-73
- "Correlation Models" on page 7-75
- "Transforms" on page 7-75

See "Local Model Definitions" on page 8-51 for statistical details of all local models.

See "Setting Up Models" on page 7-30 for a list of all information about setting up local and global models.

Local Model Class: Polynomials and Polynomial Splines

Polynomials

At the local level, if you have one input factor, you can choose Polynomial directly from the list of local model classes. Here you can choose the order of polynomials used, and you can define a datum model for this kind of local model (see below).

If there is more than one input factor, you can only choose Linear Models, Average Fit, Multiple Models or Transient Models from the Local Model Class list.

Under Linear Models you can choose Polynomial or Hybrid Spline. This is a different polynomial model where you can change more settings such as Stepwise, the Term Editor (where you can remove any model terms) and you can choose different orders for different factors (as with the global level polynomial models). See "Local Model Class: Linear Models" on page 7-67.

Different response features are available for this polynomial model and the Linear Models: Polynomial choice. You can view these by clicking the **Response Features** tab on the **Local Model Setup** dialog. Single input polynomials can have a datum model, and you can define response features relative to the datum. See "Datum Models" on page 7-79

See "Polynomials" on page 7-42 for a general description of polynomial models.

Polynomial Spline
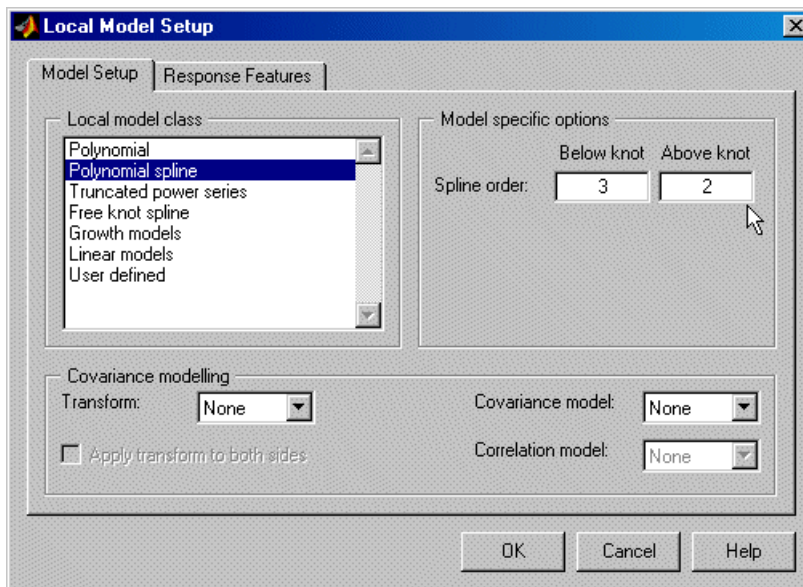
A spline is a piecewise polynomial, where different sections of polynomial are fitted smoothly together. The location of each break is called a knot.

This model has only one knot. You can choose the orders of the polynomials above and below the knot. See also "Hybrid Splines" on page 7-43. These global models also use splines, but use the same order polynomial throughout.

Polynomial splines are only available for single input factors. The following example shows a typical torque/spark curve, which requires a spline to fit properly. The knot is shown as a red spot at the maximum, and the curvature above and below the knot is different. In this case there is a cubic basis function below the knot and a quadratic above.
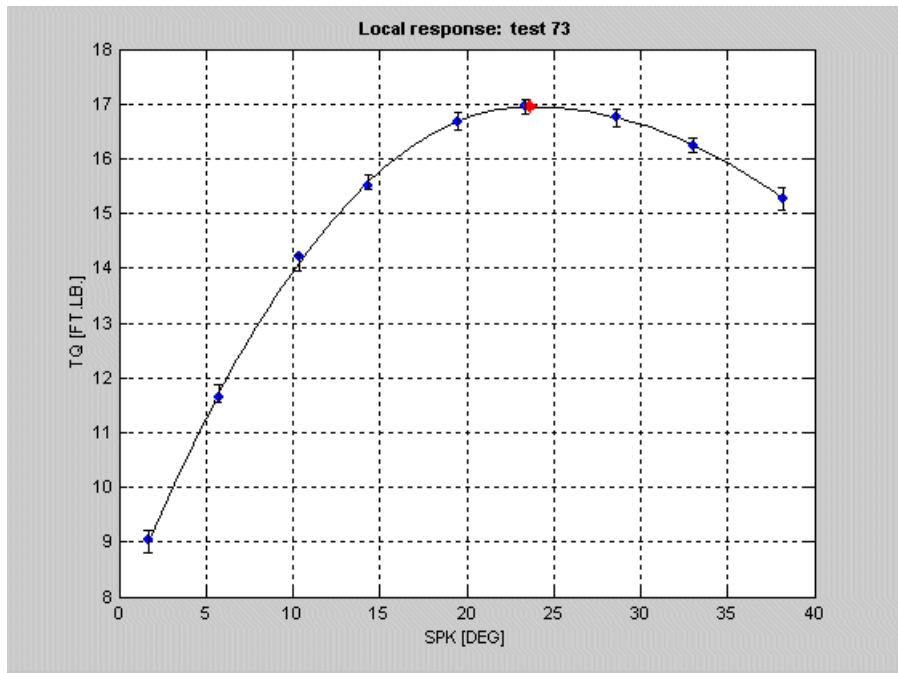
See "Polynomial Splines" on page 8-52 for statistical details.

Local Model Class: Truncated Power Series
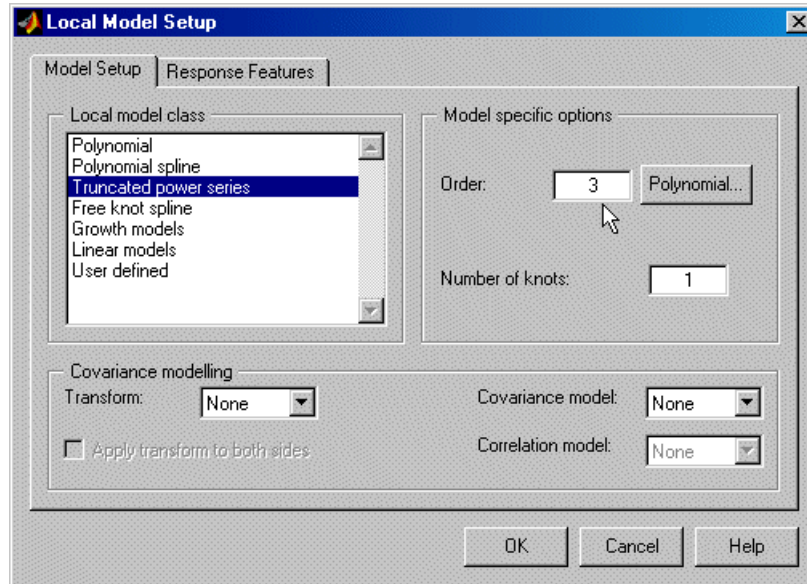
This is only available for a single input factor.

You can choose the order of the polynomial basis function and the number of knots for Truncated Power Series Basis Splines. A spline is a piecewise polynomial, where different sections of polynomial are fitted smoothly together. The point of each break is called a knot. The truncated power series changes the coefficient for the highest power when the input passes above the knot value.

It is *truncated* because the power series is an approximation to an infinite sum of polynomial terms. You can use infinite sums to approximate arbitrary functions, but clearly it is not feasible to fit all the coefficients.

Click **Polynomial** to see (and remove, if you want) the polynomial terms. One use of the remove polynomial term function is to make the function linear until the knot, and then quadratic above the knot. In this case we remove the quadratic coefficient.

See also

- "Polynomial Spline" on page 7-62, where you can choose different order basis functions either side of the knot
- "Hybrid Splines" on page 7-43, a global model where you can choose a spline order for one of the factors (the rest have polynomials fitted)
- "Local Model Class: Free Knot Spline" on page 7-65, free knot splines , where you can choose the number of knots and the order of the basis functions

Local Model Class: Free Knot Spline

These are the same as the "Global Model Class: Free Knot Spline" on page 7-56 (which is also only available for one input factor). See the global free knot splines for an example curve shape.
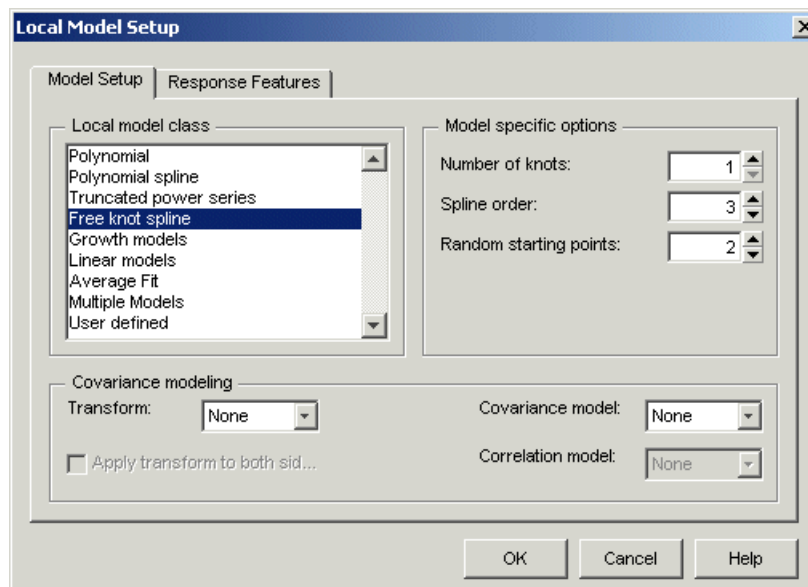
A spline is a piecewise polynomial, where different sections of polynomial are fitted smoothly together. The point of the join is called the knot.

You can choose the number of knots. You can choose the order of polynomial fitted (in all curve sections) from 1 to 3. The default is cubic.

You can set the number of **Random starting points**. These are the number of initial guesses at the knot positions.

See also

- "Polynomial Spline" on page 7-62, where you can choose different order basis functions for either side of the knot
- "Local Model Class: Truncated Power Series" on page 7-64, where you can choose the order of the basis function
- "Hybrid Splines" on page 7-43, a global model where you can choose a spline order for one of the factors (the rest have polynomials fitted)

Local Model Class: Growth Models

Growth models have a number of varieties available, as shown. They are only available for single input factors.
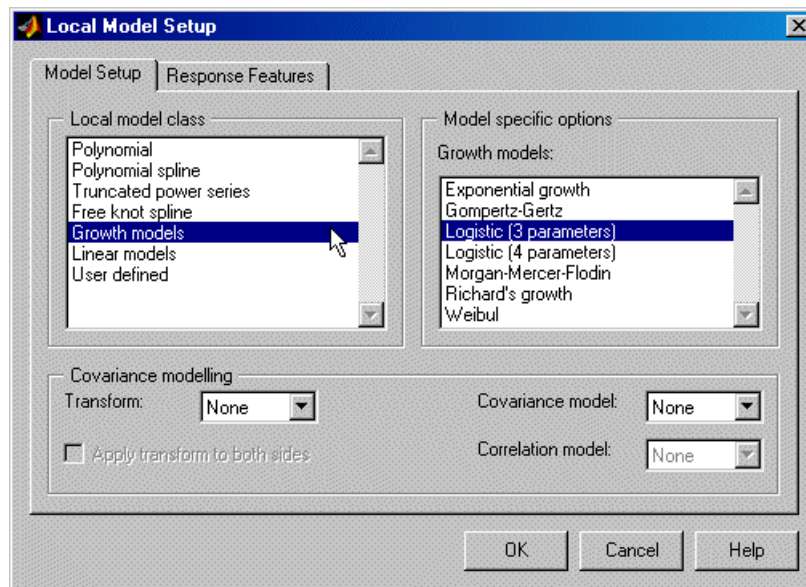
These are all varieties of sigmoidal curves between two asymptotes, like the following example.
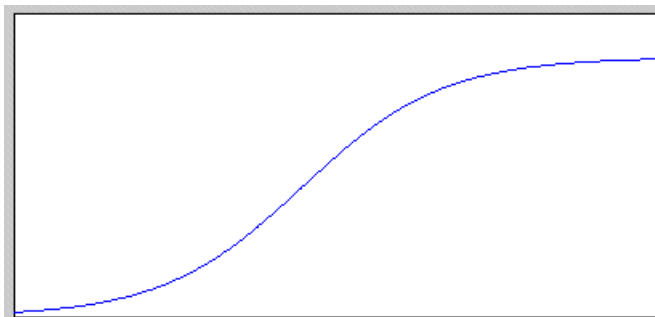
Growth models are often the most appropriate curve shape for air charge engine modeling.

See "Local Model Definitions" on page 8-51 for mathematical details on the differences between these growth models.

Local Model Class: Linear Models

Select **Linear Models** and then click **Setup**.

You can now set up polynomial or hybrid spline models. The settings are exactly the same as the global linear models.

These models are for multiple input factors - for single input factors you can use a different polynomial model from the Local Model Class list, where you can only change the polynomial order. See "Local Model Class: Polynomials and Polynomial Splines" on page 7-61.

If there is more than one input factor, you can only choose Linear Models, Average Fit, Multiple Models or Transient Models from the Local Model Class list. Under Linear Models you can choose Polynomial or Hybrid Spline. This polynomial is a different model where you can change more settings such as Stepwise, the Term Editor (where you can remove any model terms) and you can choose different orders for different factors (as with the global level polynomial models).

See "Global Linear Models: Polynomials and Hybrid Splines" on page 7-42 for details.
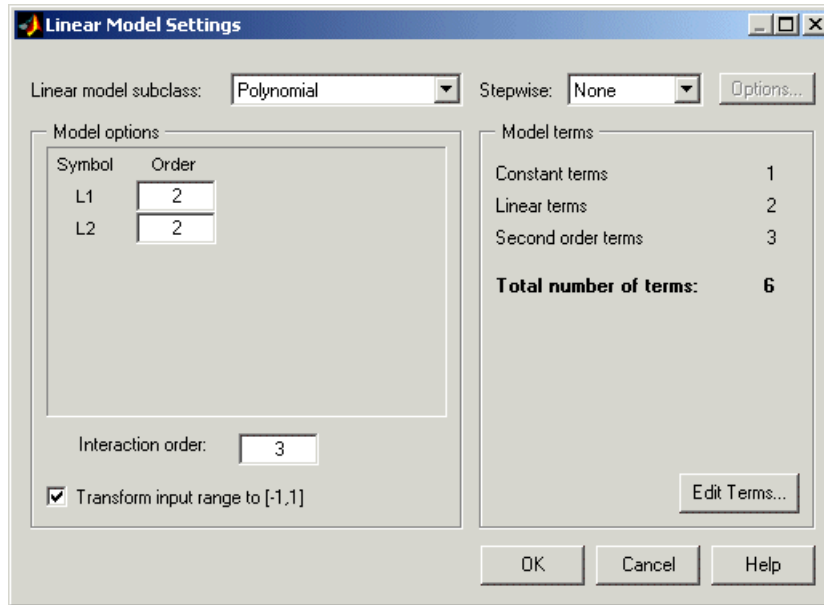
Note that by default inputs are transformed to [-1, 1] before fitting and evaluating polynomials. This is important as differences in scales between inputs can cause numerical problems. Generally it is a good idea to transform inputs as this alleviates problems with variables of different scales. In some circumstances you may be concerned with the values of polynomials (for example if your strategy requires raw polynomial coefficients). In this case you can clear the **Transform input range** check box to use untransformed units to calculate natural polynomials.

Different response features are available for this Linear Models: Polynomial model and the other Polynomial choice (for single input factors). You can view these by clicking the **Response Features** tab on the **Local Model Setup** dialog. Single input polynomials can have a datum model, and you can define response features relative to the datum. See "Datum Models" on page 7-79.

These linear models are labeled Quadratic, Cubic, and so on, on the test plan block diagram, while the single input type of polynomials is labeled Poly2, Poly3, and so on. For higher orders, both types are labeled Poly *n*.

Local Model Class: Average Fit

You can use this local model class to fit the same model to all tests. Sometimes it is desirable to try to fit a single one-stage model to two-stage data (for example, fitting an RBF over the whole operating region of spark, speed, load, air/fuel ratio and exhaust gas recirculation). However, it can still be useful to be able to examine the model fit on a test-by-test basis. You can use Average Fit for this purpose.

Select **Average Fit** and click **Setup**. The **Model Setup** dialog appears.
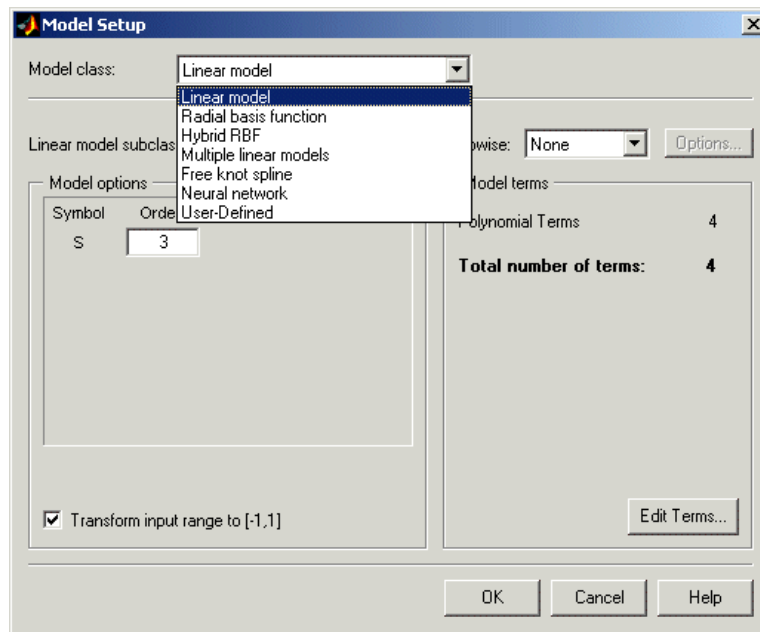
In the **Model class** drop-down menu is a list of available models. This list contains the same models that you would find in the global model setup of a one-stage model. Note the number of inputs changes which models are available. A local model with only one input can access all the models seen in the example above. See "What Models Are Available?" on page 7-31.

The Average Fit local model class allows you to use any of these global model options to fit to all tests. In the same way that global models are fitted to all the data simultaneously, using average fit allows you to fit the same model to every test in your data, instead of fitting a separate local model for each test.

The advantage of this is that you can use these one stage models to fit your data while also being able to view the fit to each test individually. You should set up your global model with an input such as record number or a dummy variable. Make all the variables you want to model local inputs. It does not matter what dummy variable or model type you use for the global input - it is only there to distinguish the Local Average Fit model from a one-stage model. The dummy global variable has no influence on the model fit. The Average Fit model is fitted in the same way as a one-stage model (across all tests simultaneously) but the main difference is that you can analyze the fit to each test individually. You cannot do this when fitting one-stage models. No two-stage model is available with local Average Fit models.

Local Model Class: Multiple Models

You can use multiple models to try a variety of models for your data. These can be very useful if you want to try several kinds of models, especially if you think there is a lot of variation between tests in your data. All the models you choose will be fitted and the best one chosen for each test in your data. In this way you can have a variety of local models at once, for example for some tests a spline or radial basis function may fit best, while for others a quadratic would be fine. You can choose any of the summary statistics as the selection criteria for deciding which model fits best to each test. Which statistics appear in the options is determined by your selections for the Summary Statistics.

The main purpose of this model is to provide a convenient mechanism to model a number of different tests using the same set of models. The data can be divided into tests and modeled within a single test plan rather than having a separate one-stage test plan for each test. You can use any model available as one-stage models (including radial basis functions (RBF) and hybrid RBF). Two-stage models or response feature models are not constructed (with different model types for different tests it is impossible to choose response features that apply to all tests). You must have at least one global variable (e.g. speed, injection timing, load) and you cannot use covariance modeling.

This provides a smooth interface with the Multimodel Tradeoff in the CAGE browser part of the Model-Based Calibration toolbox. In this case two global inputs are required (often speed and load) which form the axes of the tradeoff tables. This useful application for multiple models allows you to calibrate from local maps.

You can export multimodels and import them directly into CAGE. From the Test Plan node in the Model Browser you can select **TestPlan –> Export Multiple Models**. There will be a single CAGE model for each response. The models will only be defined at operating points corresponding to the global inputs — for example at the values of speed and load where you performed tests.

Select **Multiple Models** from the model class menu.

- Click **Edit**. The **Multi-Model Settings** dialog appears. This is similar to the method for using multiple models at the global level, see "Global Model Class: Multiple Linear Models" on page 7-55. Controls are described below.

- Alternatively you can click **Template**. The Build Models dialog opens. Here you can choose a template to build a selection of models. There are predefined templates for polynomials, radial basis functions or free knot splines, and you can also save your own templates of any models you choose.
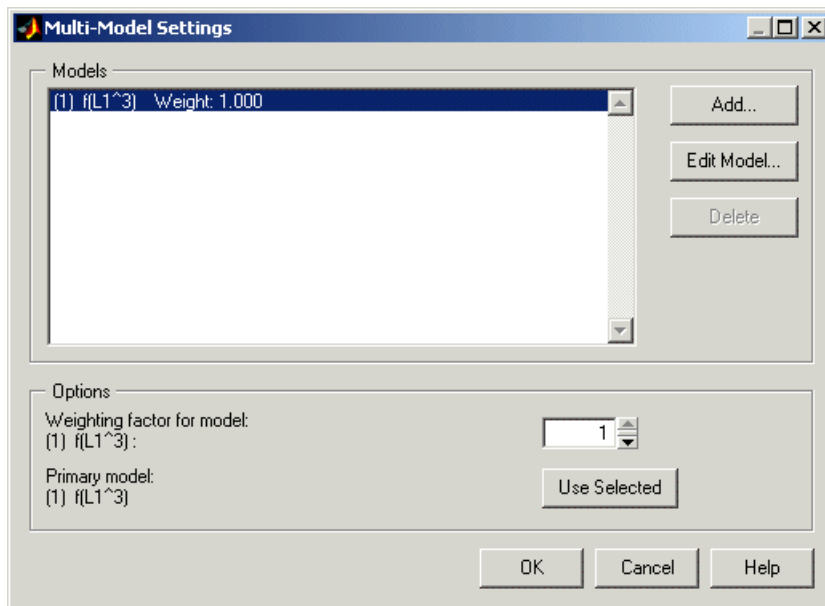
Click **Add** to reach the **Model Setup** dialog. Here you can choose from all the global models that would be available for a one-stage model with the same

number of inputs as your current local model. The example below shows the options for a single input local model.
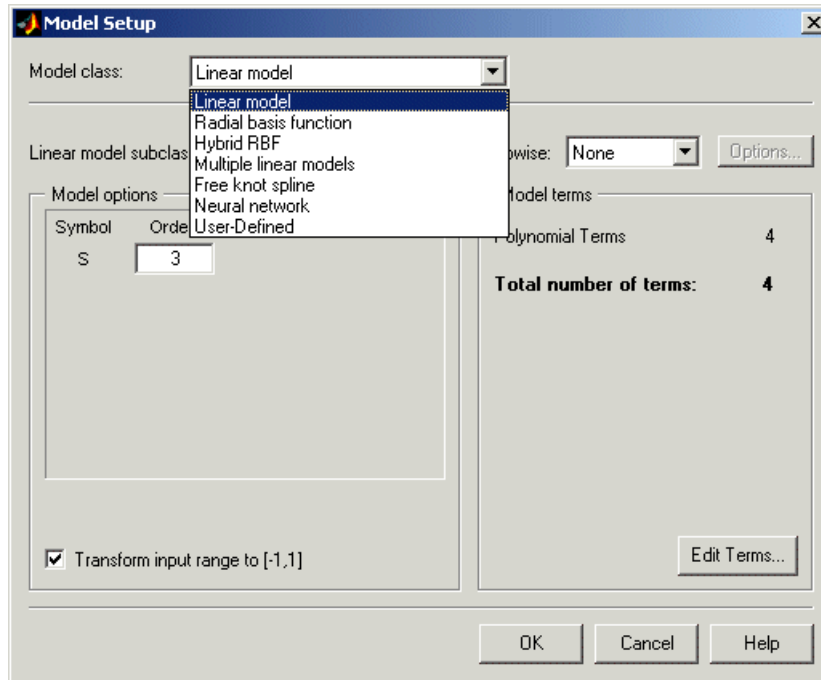
You can choose and set up these models using the same global model options. Click **OK** to add the model to your Multi-Model and return to the **Multi-Model Settings** dialog. You can always click **Edit Model** to change the settings for any of the models in the list. Model weights are not used, the selection criteria used are the ones you choose from the drop-down menu (such as RMSE or PRESS RMSE) in the **Model specific options** pane of the **Local Model Setup** dialog. You return to this dialog when you click **OK** in the **Multi-Model Settings** dialog. The selection criteria that appear in this menu are determined by your selected options for the Summary Statistics displayed.

All the models you have chosen are fitted to each test individually, and the best fit to each test is chosen by the selection criteria you picked.

Local Model Class: User-Defined Models

You must check user-defined models into the Model-Based Calibration Toolbox before you can use them here. The only model checked in by default is Exponential, which requires a single input factor.

See "User-Defined Models" on page 8-61 of the Technical Documents for detailed instructions on this.
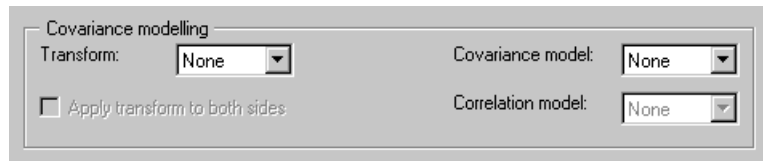
Local Model Class: Transient Models

These are supported for multiple input factors, where time is one of the factors. You can define a dynamic model using Simulink and a template file that describes parameters to be fitted in this model. We provide an example already checked in to the Model-Based Calibration Toolbox that you can use them for modeling. The example requires two inputs.

See "Transient Models" on page 8-68 of the Technical Documents for detailed instructions about how to use the example.

Covariance Modeling

This frame is visible no matter what form of local model is selected in the list.

Covariance modeling is used when there is heteroscedasticity. This means that the variance around the regression line is not the same for all values of the predictor variable, for example, where lower values of engine speed have a smaller error, and higher values have larger errors, as shown in the following example. If this is the case, data points at low speed are statistically more trustworthy, and should be given greater weight when modeling. Covariance models are used to capture this structure in the errors.
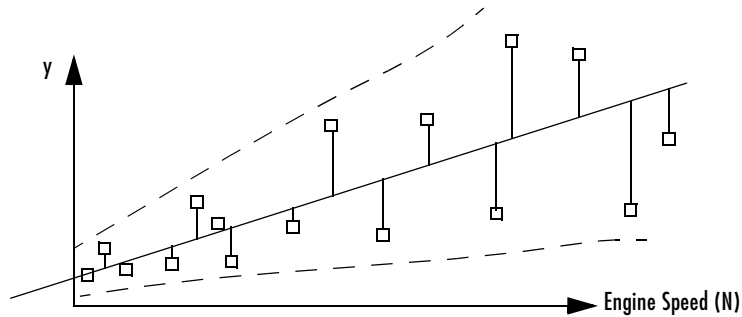
You can fit a model by finding the smallest least squares error statistic when there is homoscedasticity (the variance has no relationship to the variables). Least squares fitting tries to minimize the least squares error statistic

$\sum \varepsilon_i^2$, where ε_i^2 is the error squared at point *i*.

When there is heteroscedasticity, covariance modeling weights the errors in favor of the more statistically useful points (in this example, at low engine speed N). The weights are determined using a function of the form

$$\sum \frac{\varepsilon_i^2}{W_i}$$

where W_i is a function of the predictive variable (in this example, engine speed N).

There are three covariance model types.

Power

These determine the weights using a function of the form $W_i = \hat{y}^\alpha$. Fitting the covariance model involves estimating the parameter α.

Exponential

These determine the weights using $W_i = e^{\alpha \hat{y}}$.

Mixed

These determine the weights using $W_i = \alpha + \hat{y}^\beta$. Note that in this case there are two parameters to estimate, therefore using up another degree of freedom. This might be influential when you choose a covariance model if you are using a small data set.

Correlation Models

These are only supported for equally spaced data in the Model-Based Calibration Toolbox. When data is correlated with previous data points, the error is also correlated.

There are three methods available.

MA(1) – The Moving Average method has the form $\varepsilon_n = \alpha_1 \xi_{n-1} + \xi_n$.

AR(1) – The Auto Regressive method has the form $\varepsilon_n = \alpha_1 \varepsilon_{n-1} + \xi_n$.

AR(2) – The Auto Regressive method of the form $\varepsilon_n = \alpha_1 \varepsilon_{n-1} + \alpha_2 \varepsilon_{n-2} + \xi_n$

ξ is a stochastic input, $\xi_n \sim N(0, \sigma_\xi^2)$.

Transforms

The following example shows the transforms available.
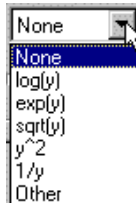
Input transformation can be useful for modeling. For example, if you are trying to fit

$$y = e^{a + bx + cx^2}$$

using the `log` transform turns this into a linear regression:

$$\log(y) = a + bx + cx^2$$

Transforms available are logarithmic, exponential, square root, y^2, $\frac{1}{y}$, and `Other`. If you choose `Other`, an edit box appears and you can enter a function.

Apply transform to both sides is only available for nonlinear models, and transforms both the input and the output. This is good for changing the error structure without changing the model function. For instance, a log transform might make the errors relatively smaller for higher values of *x*. Where there is heteroscedasticity, as in the Covariance Modeling example, this transform can sometimes remove the problem.

Designing Experiments

You can design experiments after setting up models. You can design experiments for both stages, local and global. You invoke the Design Editor in several ways from the test plan level:

• Right-click a Model block in the test plan diagram and select `Design Experiment`.

  You must select (by clicking) a stage to design for (first or second stage) or the following two options are not possible.

• Click the **Design Experiment** toolbar button ▦.

• Select **TestPlan –> Design Experiment**.

For an existing design, **View –> Design Data** also launches the Design Editor (also in the right-click menu on each Model block). In this case you can only view the current data being used as a design at this stage. If you enter the Design Editor by the other routes, you can view all alternative designs for that stage.

See the "Tutorial: Design of Experiment" on page 3-1.

Viewing Designs

The view design facility enables the user to investigate the statistical properties of the current data.

• From the test plan node, select the model stage you are interested in by clicking, then choose **View –> Design Data**. Alternatively, use the right-click menu on a Model block.

This provides access to all the Design Editor and design evaluation utility functions with the current data rather than the prespecified design. If you have done some data-matching to a design, each data point is used as a design point. You can now investigate the statistical properties of this design.

For two-stage models, viewing stage one (local model) designs creates a separate design for each test.

See "The Design Editor" on page 5-2 or the step-by-step guide in the "Tutorial: Design of Experiment" on page 3-1.

New Response Models

When you first set up a test plan, the Data Wizard automatically contains the response model setup after the data matching functions.

The following applies when you return to a previously setup test plan to add a new response model, or when you click the **New** button at test plan level to add new response models to an existing test plan.

Double-click the Responses outport of the block diagram or use the **New Response Model** item in the **File** menu, or use the toolbar icon. (None of these is available unless you are in the Test Plan view, that is, have the test plan node selected in the model tree. This should be obvious: you can only see the test plan block diagram with the test plan node selected.)
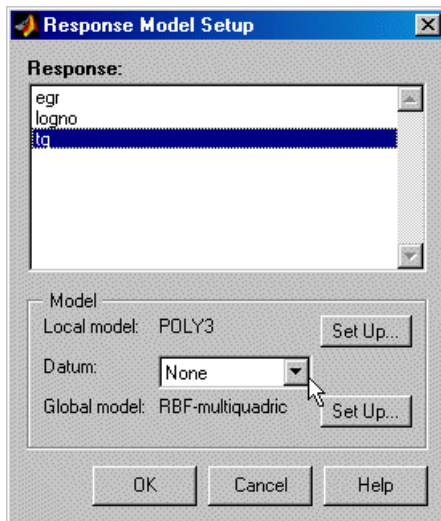
The **Response Model Setup** dialog has a list box containing all the variables in the selected data set *except* the inputs to the local and global models; you cannot use an input also as a response.

You can reach the controls for setting up models using the **Set Up** buttons to change the local and global models also, and you can add datum models (maximum or minimum) if the local model supports this.

You can return to the local or global setup options individually at any time by double-clicking the block in the test plan diagram.

Datum Models

Under **Datum** you can choose a datum model, but only for some local models — polysplines and polynomials (but see Linked Datum Models following). Other local models cannot have a datum model, as they do not necessarily have a unique turning point.

The datum model tracks the maximum or minimum of the local models. This is equivalent to adding the maximum or minimum as a response feature, which can be very useful for analysis if those points are interesting from an engineering point of view.

The **Datum** options are

- **None**
- **Maximum** — This can be useful in cases using polyspline modeling of torque against spark. The maximum is often a point of engineering interest.
- **Minimum** — This can be useful for cases where the object is to minimize factors such as fuel consumption or emissions.
- **Linked datum model** — This is only available to subsequent two-stage models within a test plan in which the *first* two-stage model has a datum model defined. In this case you can make use of that datum model. The linked datum option carries the name of the response of the first two-stage model, where it originated.

If the maximum and minimum are at points of engineering interest, like MBT or minimum fuel consumption, you can add other response features later using the datum model (for example, MBT plus or minus 10 degrees of spark angle) and track these across local models too. It can be useful to know the value of MBT when modeling exhaust temperature, so you can use a linked datum model from a previous torque/spark model. Having responses relative to datum can also be a good thing as it means the response features are more likely to relate to a feature within the range of the data points.

You can also export the datum model along with local, global, and response models if required. See "Exporting Models" on page 7-148.

Local Level

When you select a local node (with the ⬛ icon) in the model tree, this view appears.

Note that after the two-stage model is calculated the local node icon changes to a two-stage icon (🔵) to reflect this. See the model tree for clarification. The response node also has a two-stage icon, but produces the response level view instead.

See "Local Level: Toolbar" on page 7-86 and "Local Level: Menus" on page 7-87 for details on these controls.

The display features are described in the following sections:

- "Local Special Plots" on page 7-82
- "Local Scatter Plots" on page 7-83
- "Diagnostic Statistics Pane" on page 7-84 and "Pooled Statistics" on page 8-25
- "Response Features List Pane" on page 7-84
- "Test Notes Pane" on page 7-85
- "Data Tab" on page 7-85

The following example shows a local model of torque/spark curves.

Click here to change test

Click-and-drag anywhere along this
line to change the size of the Response
Features pane

Click-and-drag or double-click here to expand
plots/hide statistics pane

The default view is the **Model** tab, with plots described below. You can click the **Data** tab to view plots of other variables. See "Data Tab" on page 7-85.

Local Special Plots

The lower plots are referred to as *special plots* as they can be different for different models.

The lower plot at the local level shows the local model fit to the data *for the current test only*, with the datum point if there is a datum model. If there are multiple inputs to the local model, a predicted/observed plot is displayed. In this case to examine the model surface in more detail you can use **Model –> Evaluate**. See "Model Evaluation Window" on page 7-144.

You can scroll through all the local models by using the up/down test buttons, type directly in the edit box, or go directly to test numbers by clicking **Select Test**.

To examine the local fit in more detail, double-click the arrows (indicated in the preceding figure) to hide the scatter plot and expand the lower plot. You can zoom in on parts of the plot by **Shift**-click-dragging or middle-click-dragging on the place of interest on the plot. Return to full size by double-clicking.

You can change the lower plot from the **Local Response** to a **Normal Plot** by using the drop-down menu at the top of the plot.

Special plot right-click context menu

Scatter plot right-click context menu
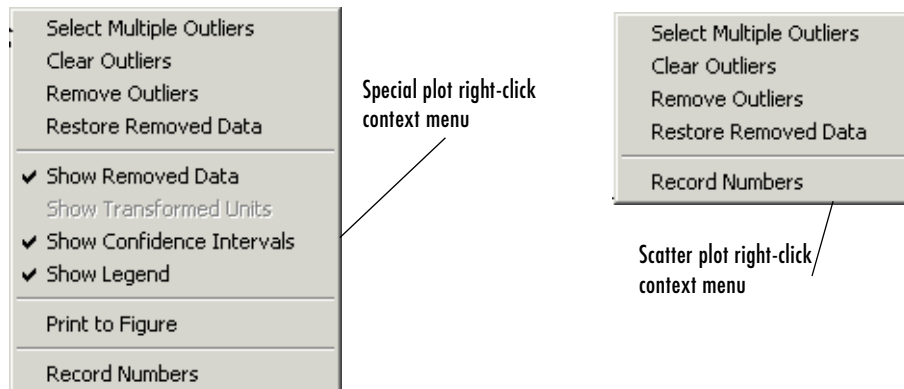
Above are the right-click context menus for both plots. On both plots you can manipulate outliers with all the same commands available in the **Outliers** menu. See "Outliers Menu (Local Level)" on page 7-92 for details.

The **Print to Figure** command opens a MATLAB figure plot showing the current plot. On the special plots you can switch the confidence intervals and legend on and off, and hide or show removed data. For both plots you can switch the display of **Record Numbers** on and off. This is similar to test number for global models but relates to individual records within tests.

Local Scatter Plots

The upper plots are referred to as *scatter plots*. They can show various scatter plots of statistics for assessing goodness-of-fit for the current local model shown. Upper scatter plots are replaced by an icon if you resize the Browser too small.

The statistics available for plotting are model dependent.

```
Studentized residual ▼
spark (L1) [deg]
Residuals
Weighted residuals
Studentized residuals
tq [ft lbf]
Predicted tq [ft lbf]
Leverage
Obs. number
```

The preceding is an example drop-down menu on the scatter plot for changing *x* and *y* factors. In this case spark is the local input factor and torque is the response. The local inputs, the response, and the predicted response are always available in these menus. The observation number is also always available.

The other options are statistics that are model dependent, and can include residuals, weighted residuals, studentized residuals, and leverage.

Diagnostic Statistics Pane
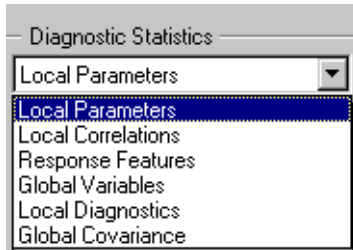
The **Diagnostic Statistics** pane drop-down menu is shown, where you can select the information to be displayed in the pane. If there is not enough room there are scroll bars.

- Local Parameters — Shows the values and standard errors of the parameters in the local model for the current test selected.

- Local Correlations — Shows a table of the correlations between the parameters.

- Response Features — Shows the values and standard errors of the response features defined for this local model, from the current test (often some or all of them are the same as the parameters; others are derived from the parameters).

- Global Variables — Shows the values and standard errors of the global variables at the position of the current test.

- Local Diagnostics — s_i (the standard error for the current (i^{th}) test), number of observations, degrees of freedom on the error, R squared, Cond(J) and Cond(Sigma): the condition indices for the Jacobian matrix and the covariance matrix.

- Global Covariance — For MLE models, shows a covariance matrix for the response features at the global level.

For information on the Pooled Statistics, see "Pooled Statistics" on page 8-25.

Response Features List Pane

Here is a list of all the response features calculated for the local model. A two-stage model using the local and global models is formed by using **Select**.

Click the **Select** button here to enter the **Model Selection** window. This step is required before two-stage models can be calculated.

The list view displays the number of parameters and observations, the value of any Box-Cox transformation (1 indicates no transform), and the values of RMSE and PRESS RMSE (linear models only) for each response feature model. For definitions of RMSE and PRESS RMSE, see "Linear Model Statistics Displays" on page 8-22. For information on Box-Cox transforms, see "Box-Cox Transformation" on page 8-18.

Click **New** to add a new response feature model (or **Delete** to remove one). For more information see the "Test Plans List Pane" on page 7-5. The contents of this pane change in different views; it always contains the child nodes of the node selected in the model tree (and the **New**, **Delete**, and **Select** buttons). At the local level it contains a list of response features.

Test Notes Pane

You can use the **Test Notes** pane to record information on particular tests. Each test has its own notes pane. Data points with notes recorded against them are colored in the global model plots. You choose the color using the **Test Number Color** button in the **Test Notes** pane.

Data Tab

- When you click the **Data** tab at the local level or select **View –> Data Plots**, you can view plots of the data for the current test.
- Use the right-click menu item **Set Up Plot Variables** to open the **Plot Variables Setup** dialog. The dialog appears automatically if you open the tab using the **View** menu.

Here you can choose to view any of the data signals in the data set for the current test (including signals not being used in modeling). Choose variables from the list on the left and use the buttons to move them into the **Y Variable(s)** list or the **X variable** edit box. You can use the **No X Data** button to plot a variable against record number only. Remember you can also display values of global variables in the Diagnostic Statistics pane if you want to see these values at the same time as the **Model** tab.

Local Level: Toolbar

This toolbar appears when a local node is selected in the model tree.
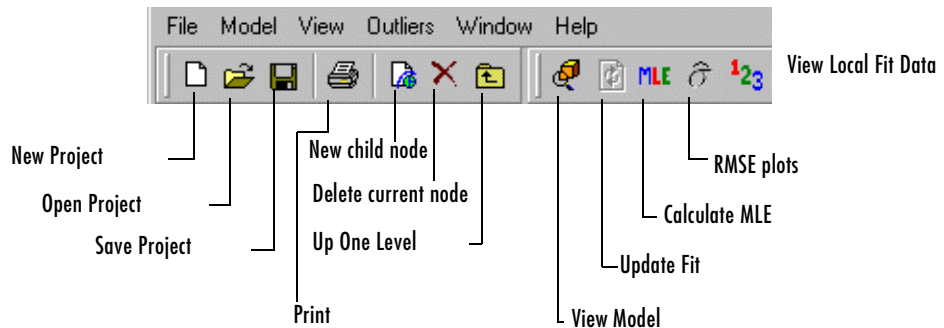
The eight left icons remain constant throughout the levels. They are for project and node management, and the help button, and here the print icon is enabled as there are plots in this view. See "Project Level: Toolbar" on page 7-7 for details on these buttons. In the example shown the slider bar has been dragged to hide the Help button.

- **View Model** — Opens a dialog displaying the terms in the current model.
- **Update Fit** — This button is only enabled when data has been excluded from the plot using the **Remove Outliers** command (in the right-click context menu or the **Outliers** menu). At this point the local fit in the view is updated to fit only the remaining data, but this change also affects a point in all the global models. You can make this update to all the global models by using this toolbar button, or it happens automatically when another node is selected.

Note Update fit can affect several models. Removing an outlier from a best local model changes all the response features for that two-stage model. The global models all change; therefore the two-stage model must be recalculated. For this reason the local model node returns to the local (house) icon and the response node becomes blank again. If the two-stage model has a datum model defined, and other models within the test plan are using a datum link model, they are similarly affected.

- **Calculate MLE** — Calculates the two-stage model using maximum likelihood estimation. This takes correlations between response features into account. See "MLE" on page 7-137 for details.
- **RMSE Plots** — Opens the **RMSE Explorer** dialog, where you can view plots of the standard errors of all the response features. There is one value of standard error per test for each response feature. You can also plot these standard errors against the global variables to examine the global distribution of error.
- **View Local Fit Data** — Opens the Data Editor, where you can view a read-only version of the inputs, predicted and actual responses. This allows you to examine and export your modeling data, using all the powerful display features of the Data Editor. Also in the **View** menu.

Local Level: Menus

File
- Only the **New** (child node) and **Delete** (current node) functions change according to the node level currently selected. Otherwise the **File** menu remains constant.

  See "File Menu" on page 7-8.

Window and Help Menus
- The **Window** and **Help** menus remain the same throughout the Model Browser, offering access to different windows, general help and context help.

  See "Window Menu" on page 7-10 and "Help Menu" on page 7-10.

See also

Model Menu (Local Level)
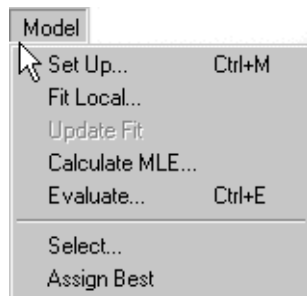
- **Set Up** — Opens the **Local Model Setup** dialog where you can change the model type. See "Local Model Setup" on page 7-60.
- **Fit Local**— Opens the **Local Model Fit Tool** dialog. Without covariance modeling, you see the following controls. This example shows the results after clicking **Fit** once. The optimization process can be stopped early by clicking **Stop** or you can wait until it finishes. The Ordinary Least Squares (OLS) parameters are displayed. You can click **Fit** to run the process again as many times as required, or **Close** to exit the dialog. You can enter a different change in parameters in the edit box.
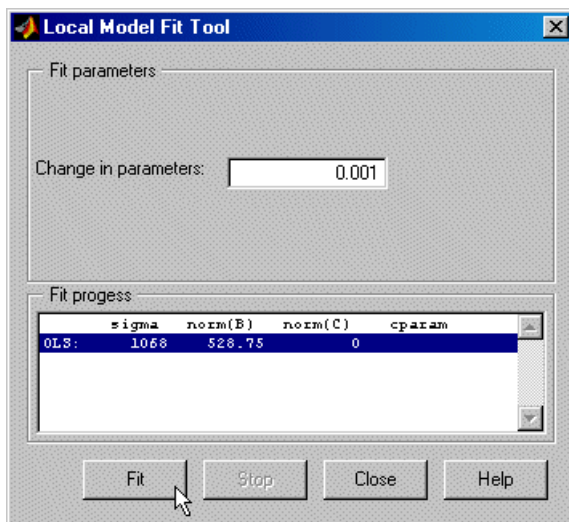
- For covariance models this offers three different algorithms: **REML** (Restricted Maximum Likelihood – the default), **Pseudo-likelihood**, and **Absolute residuals**. The following example shows that there is also an additional button, **One Step**. Using the **Fit** button might take several steps to converge, but if you use the **One Step** button only one step in the optimization process is taken. Every time you run a process, the initial and final Generalized Least Squares parameter values are displayed for each iteration.
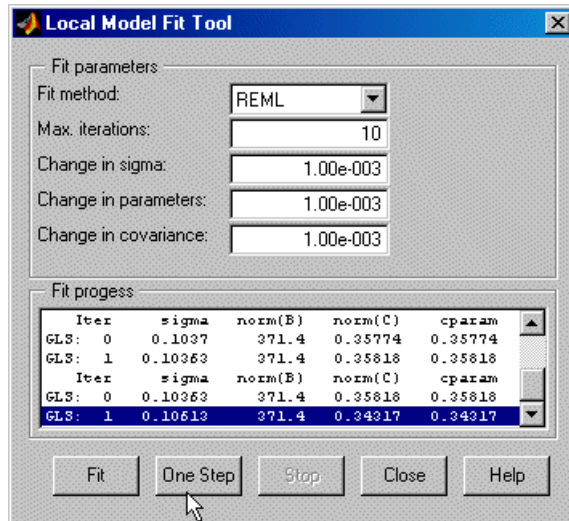
- **Update Fit** — Only enabled when outliers have been removed. The **Update Fit** option updates the global models to take these changes into account. This happens automatically when you select a different node. Duplicated in the toolbar. This can affect many models; see "Local Level: Toolbar" on page 7-86.

- **Calculate MLE** — Calculates the two-stage model using maximum likelihood estimation. This takes interactions between response features into account. Duplicated in the toolbar. See "MLE" on page 7-137 for details.

- **Evaluate** — Opens the **Model Evaluation** window.

- **Select** — Available whenever the **Select** button is also enabled in the lower right pane (when it is titled **Local Models**, **Response Features**, or **Models**). This item opens the **Model Selection** window to allow you to choose the best model. See "Select Button" on page 7-111.

- **Assign Best** selects the current model as best. If it is one of several child node models of a response model, selecting it as best means that this local model (and associated response features) is used for the two-stage model. Note that this is only enabled if the local node selected has a two-stage model calculated; that is, if the local node still has a local icon (a house) you cannot use **Assign Best**. See "Model Tree" on page 7-11.

View Menu (Local Level)
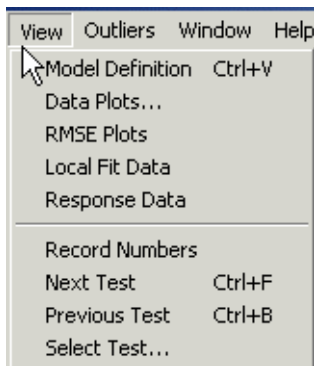
- **Model Definition** — Opens the **Model Viewer** dialog, showing the terms in the model.

- **Data Plots** — Opens the **Data** tab on the display and the **Plot Variables Setup** dialog, where you can choose to view any of the data signals in the data set for the current test (including signals not being used in modeling). Choose variables from the list on the left and use the buttons to move them into the **Y Variable(s)** list or the **X variable** edit box. You can use the **No X Data** button to plot a variable against record number only.

- **RMSE Plots** — Opens the **RMSE Explorer** where you can view plots of the standard errors of all the tests, both overall and by response feature. There is one value of standard error per test, overall and for for each response feature. It is most useful to first plot plain s.e. against test number to get an idea of how the error is distributed and locate any tests with much higher errors. Right-click to select display of test numbers. This can be useful for quickly identifying problem tests. Ideally, all the standard errors should be roughly the same value otherwise some of the statistics theory breaks down. You can also plot these standard errors against the global variables to examine the global distribution of error.

- **Local Fit Data** — Opens the Data Editor, where you can view a read-only version of the inputs and predicted values. This allows you to examine your modeling data and inputs using all the powerful display features of the Data Editor. You can also export your modeling data to the workspace or Excel. Also in the toolbar.

- **Response Data** — Opens the Data Editor, where you can view a read-only version of the values of all the response features for the local models. You can also use the Data Editor export facilities.

- **Record numbers** — toggles the display of record numbers on plots.

- **Next Test**, **Previous Test**, and **Select Test** — Duplicate the buttons for changing tests above the plots in the top left of the **Local Model** display pane.

Outliers Menu (Local Level)

Outliers

Select Multiple Outliers
Clear Outliers
Remove Outliers   Ctrl+A
Restore Removed Data Ctrl+Z
Copy Outliers
Selection Criteria
Remove All Data
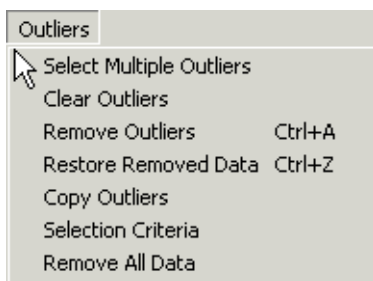
All the commands except **Remove All Data** and Copy Outliers are also available in the right-click context menus on all plots.

- **Select Multiple Outliers** — Use this item to draw a selection box around as many data points as required to select them all as outliers. This is useful for removing many data points at once.

- **Clear Outliers** — Returns all data points to the unselected state (that is, no points outlined in red) as possible outliers.

- **Remove Outliers** — Removes red-outlined data points from the fit and refits the current local fit only. Use the **Update Fit** toolbar button or **Model –> Update Fit** to refit all the global models also. This also happens automatically when another node is selected.

- **Restore Removed Data** — Opens the **Restore Removed Data** dialog, where you can choose the points to restore from the list by record number, or restore all available points. You can also press **CTRL**+Z. Select points in the left list and use the buttons to move points between the lists. When you click **OK** this refits the local model, including all data points previously removed as

outliers. Use **Update Fit** once more to refit the global models, or it happens automatically when a new node is selected.

- **Copy Outliers** — Opens the **Copy Outliers** dialog, where you can choose which model's outliers to copy. Select a model (of the same type — local or global) in the tree and click **OK**, and the current model (and other models affected) are refitted using the outlier selections for that model.

- **Selection Criteria** — Opens the **Outlier Selection Criteria** dialog where you can set the criteria for the automatic selection of outliers. This is disabled for MLE models.

- **Remove All Data** — Leaves the current local model with no data, so entirely removes the current test. This test is removed from all the global models.

Outlier Selection Criteria

You can select outliers as those satisfying a condition on the value of some statistic (for example, residual>3), or by selecting those points that fall in a region of the distribution of values of that statistic. For example, assume that residuals are normally distributed and select those with p-value>0.9. You can also select outliers using the values of model input factors.
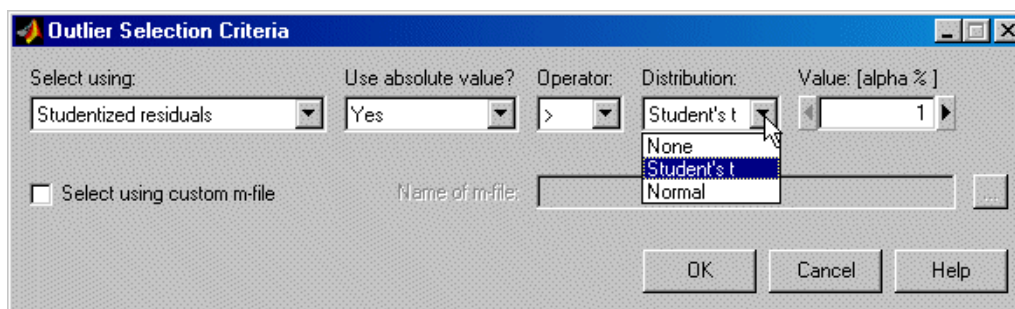
The drop-down menu labeled **Select using** contains all the available criteria, shown in the following example.
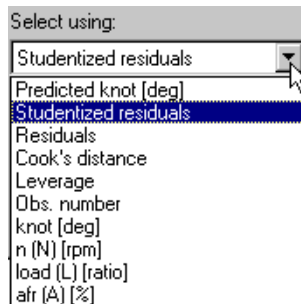
The options available in this menu change depending on the type of model currently selected. The options are exactly the same as those found in the drop-down menus for the *x*- and *y*-axis factors of the scatter plots in the Model Browser (local level and global level views).

In the preceding example, the model selected is the knot response feature, so `knot` and `Predicted knot` appear in the criteria list, plus the global input factors; and it is a linear non-MLE model, so `Cook s Distance` and `Leverage` are also available.

The range of the selected criteria (for the current data) is indicated above the **Value** edit box, to give an indication of suitable values. You can type directly in the edit box. You can also use the up/down buttons on this box to change the value (incrementing by about 10% of the range).

Distribution

You can use the **Distribution** drop-down menu to remove a proportion of the tail ends of the normal or *t* distribution. For example, to select residuals found in the tails of the distribution making up 10% of the total area:

• Select `Normal` in the **Distribution** drop-down menu.
• Select the operator >.
• Enter 10 as the α% value in the edit box.

Residuals found in the tails of the distribution that make up 10% of the total area are selected. If you had a vast data set, approximately 10% of the residuals would be selected as outliers.

As shown, residuals found beyond the value of Z_α in the distribution are selected as outliers. α is a measure of significance; that is, the probability of

finding residuals beyond Z_α is less than 10%. Absolute value is used (the modulus) so outliers are selected in both tails of the distribution.
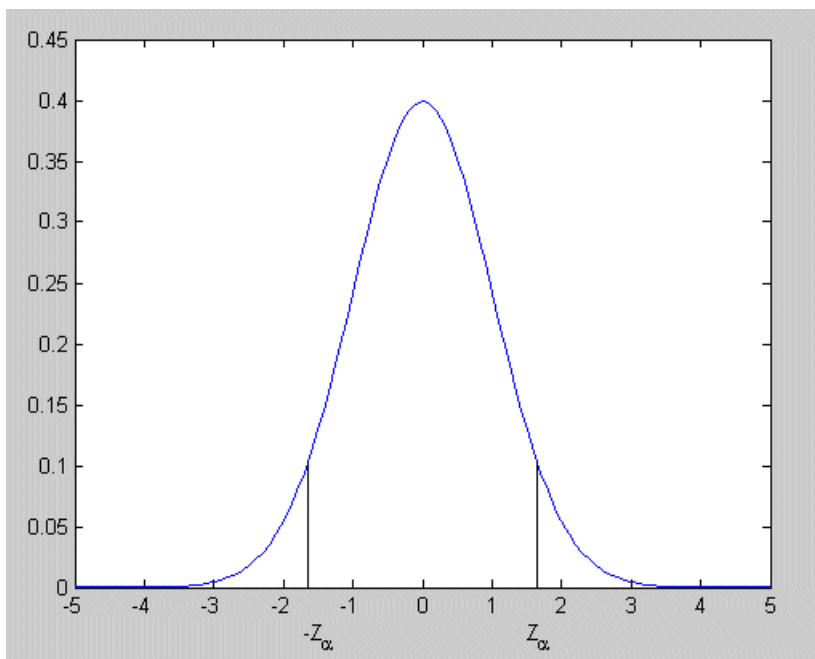
The *t* distribution is used for limited degrees of freedom.

If you select None in the **Distribution** drop-down menu, you can choose whether or not to use the absolute value. That is, you are selecting outliers using the actual values rather than a distribution. Using **absolute value** allows you to select using magnitude only without taking sign into account (for example, both plus and minus ranges). You can select No here if you are only interested in one direction: positive *or* negative values, above or below the value entered. For example, selecting only values of speed below 2000 rpm.

The **Select using custom M-file** check box enables the adjacent edit box. Here you can choose an M-file that selects outliers. Type the name of the file and path into the edit box, or use the browse button.

In this M-file you define a MATLAB function of the form:

```
function outIndices = funcname (Model, Data, Names)
```

Model is the current MBC model.

Data is the data used in the scatter plots. For example, if there are currently 10 items in the drop-down menus on the scatter plot and 70 data points, the data make up a 70 x 10 array.

Names is a cell array containing the strings from the drop-down menus on the scatter plot. These label the columns in the data (for example, spark, residuals, leverage, and so on).

The output, outIndices, must be an array of logical indices, the same size as one column in the input Data, so that it contains one index for each data point. Those points where index = 1 in outIndices are highlighted as outliers; the remainder are not highlighted.

Global Level

When you select a response feature node or one-stage model node in the model tree, this view appears. Both kinds of models have a global icon (🌐), to reflect the fact that a global model is fitted, so this is referred to as *global level*. Plots shown here are referred to as global plots. Child nodes of these models also have global icons. Any node with a global icon produces this view.

For one-stage models, this view shows the functionality available at all model nodes. For two-stage models there are other levels with different functionality for local and response models.

See "Global Level: Toolbar" on page 7-103 and "Global Level: Menus" on page 7-106 for details on these controls.

This view is similar in format to the local level view, which also contains scatter plots above special plots. The statistical information panes on the right side are different, there is a **Removed Data** pane, and there are no test number controls. There is an edit box for model comments.

The display features are described in the following sections:

- "Global Special Plots" on page 7-99
- "Global Scatter Plots" on page 7-99
- "Summary Table" on page 7-101
- "Anova Table" on page 7-101
- "Removed Data Pane" on page 7-101
- "Models Comments" on page 7-101
- "Models List" on page 7-102

Click-and-drag anywhere along this line to change the size of the Models pane

Click-and-drag or double-click here to expand plots/hide statistics panes

Global Special Plots

The lower plots in the global level view are referred to as *special plots*, as they can be different for different models (for example, none at all for neural net models).

The special plot at the global level shows a **Predicted/Observed** plot. Where there is only one input factor, the plot shows the model fit and the data against the input factor (as in most local model special plots, which often have only one input factor).

For response feature models, each data point is the value taken by this response feature for some local model fit (of this two-stage model). Note that response features are not necessarily coefficients of the local curves, but are always derived from them in some way.

When there is more than one input factor it becomes impossible to display the fit in the same way, so the data for the response feature is plotted against the values predicted by the global model. The line of **predicted=observed** is shown. With a perfect fit, each point would be exactly on this line. The distances of the points from the line (the residuals) show how well the model fits the data.

To examine the fit in more detail, double-click the arrows (indicated in the figure in "Global Level" on page 7-97) to hide the scatter plot and expand the lower plot. You can also zoom in on parts of the plot by **Shift**-click-dragging or middle-click-dragging on the place of interest on the plot. Return to full size by double-clicking.

Note Right-click a point in either the special or scatter plot to open a figure plot of that particular test (for example, torque against spark).

You can change the lower plot from the **Predicted/Observed** to a **Normal Plot** by using the drop-down menu at the top of the plot.

Global Scatter Plots

The upper plots in the global level view are referred to as *scatter plots*. They can show various scatter plots of statistics for assessing goodness-of-fit for the

current model shown. The upper scatter plots are replaced by an icon if you resize the Browser too small.
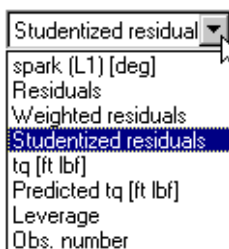
The statistics available for plotting are model dependent.
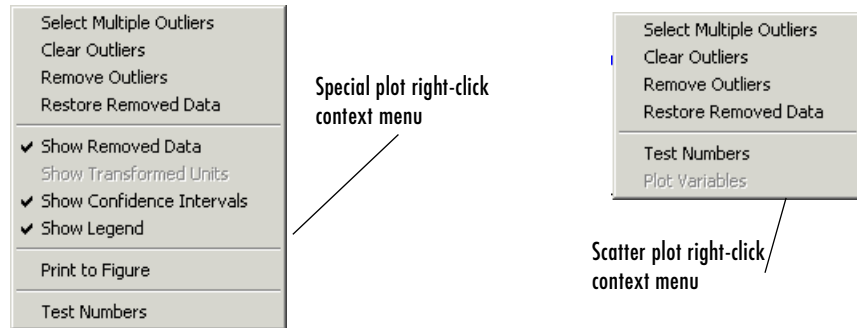
The preceding are the context menus for both plots. On both plots you can manipulate outliers with all the same commands available in the **Outliers** menu except **Outlier Selection Criteria** and **Copy Outliers**. See "Outliers Menu (Local Level)" on page 7-92 for details. For one-stage models you can select **Plot Variables** on the scatter plot.

You can choose the *x*- and *y*-axis factors using the drop-down menus. The available statistics and factors are model dependent. Following is an example.
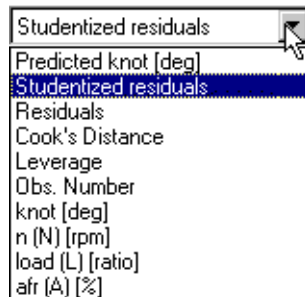
Shown is an example drop-down menu on the scatter plot for changing *x* and *y* factors. In this example knot is the response feature node selected. Therefore the model output is knot, so knot and Predicted knot are available in the menu. (For child nodes of knot, the model output is still knot.) The global

inputs, the model output, and the predicted model output are always available in these menus. The observation number is also always available.

The other options are statistics that are model dependent, and can include: Residuals, Weighted Residuals, Studentized Residuals, Leverage, and Cook's Distance. These statistics and the other factors are also used as the available criteria for selection of outliers, so the options in the **Outlier Selection Criteria** dialog are similarly model dependent.

Summary Table

For information on the displayed statistics, see "Linear Model Statistics Displays" on page 8-22 and "Definitions" on page 8-4. Here you can find explanations of the information found in the Summary table. Note that you can specify additional statistics to display in the Summary table by selecting **Model –> Summary Statistics**. Note that these settings are inherited from parent global models and from the test plan level *only if* set before the current node was created, when they also appear in the bottom list pane if there are child nodes to compare. See "Summary Statistics" on page 7–24 and "Model Menu (Global Level)" on page 7-107.

Anova Table

See "ANOVA Table" on page 8-22 and "Definitions" on page 8-4 for details of the information here. This table is not present after you have calculated MLE.

Removed Data Pane

Tests you have removed here (using the **Remove Outliers** menu item) are shown in a list. Select **Outliers –> Restore Removed Data** to select some or all of these to restore. Tests marked with an asterisk (*) are not restorable here. This means the entire test has been removed at the local level (using the **Remove All Data** menu item), or the local model could not be fitted. These can only be restored at the local level.

Double click on any removed test number to display a plot of the test in a figure window.

Models Comments

You can enter comments in the edit box.

Models List

Here is a list of any child models of the currently selected global model. This list is empty if there are no child nodes to compare. If thre are child nodes you can click the **Select** button here to enter the **Model Selection** window to compare the child models and choose the best.

Click **New** to add a new child global model (or **Delete** to remove one). For more information see the "Test Plans List Pane" on page 7-5. The contents of this pane change in different views; it always contains the child nodes of the node selected in the model tree (and the **New**, **Delete**, and **Select** buttons). At the global level it contains a list of any child models.

The list view displays the number of parameters and observations, the value of any Box-Cox transformation (1 indicates no transform), and the values of PRESS RMSE (for linear models only) and RMSE for each child model. Note that you can specify additional statistics to display here and in the Summary table by selecting **Model –> Summary Statistics**. Use these statistics to compare the fit of different child models and help you choose the best. For definitions of RMSE and PRESS RMSE, see "Linear Model Statistics Displays" on page 8-22. For other statistics see "Summary Statistics" on page 7–24 and "Model Menu (Global Level)" on page 7-107. For information on Box-Cox transforms, see "Box-Cox Transformation" on page 8-18.

Note that the Summary Statistics settings are inherited from parent global models and from the test plan level *only if* set before the current node was created, when they also appear in the bottom list pane if there are child nodes to compare.

Global Level: Toolbar

This toolbar appears when a response feature node or one-stage model node (both have a global icon) is selected in the model tree. Note that for one-stage models all model child nodes of the one-stage test plan are of this type.

Further buttons appear on the right depending on the type of model at the node selected. See "Global Level: Model-Specific Tools" on page 7-104.
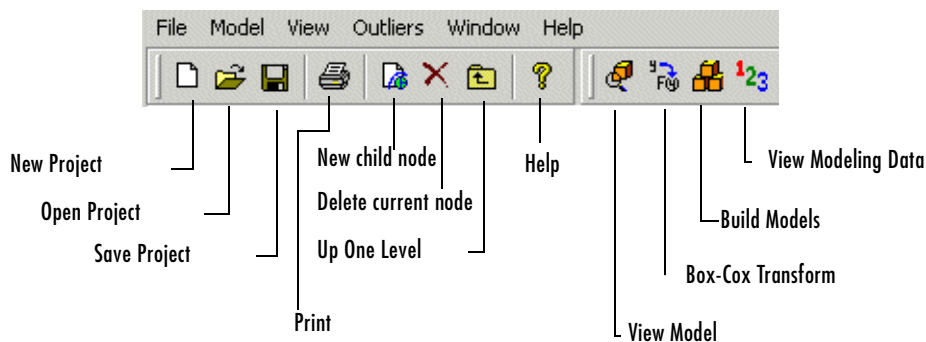
The eight left icons remain constant throughout the levels. They are for project and node management, Help, and here the print icon is enabled, as there are plots in this view. See "Project Level: Toolbar" on page 7-7 for details on these buttons.

- **View Model** — Opens a dialog displaying the terms in the current model.
- **Box-Cox Transform** — Opens the Box-Cox Transformation plots, where you can minimize SSE to try to improve the fit. See "Box-Cox Transformation" on page 8-18 for statistical details.
- **Build Models** — Opens the **Build Models** dialog, where you can choose a template for the type of models you want to build. There are predefined templates for polynomials, RBF kernels, and free knot splines. You can also save templates of whatever models you choose using the **Model –> Make Template** menu item. 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.
- **View Modeling Data** — Opens the Data Editor, where you can view a read-only version of the inputs, predicted and actual responses. This allows

you to examine and export your modeling data, using all the powerful display features of the Data Editor. Also in the **View** menu.

These four toolbar icons appear for every global model node (although Box-Cox is not enabled for neural net models). The icons that appear to the right are model specific.

Global Level: Model-Specific Tools

All twelve left buttons (up to View Modeling Data) appear for all response feature models and one-stage models. See "Global Level: Toolbar" on page 7-103 for details on these buttons. The right buttons change according to model type.

Linear Model and Multiple Linear Models
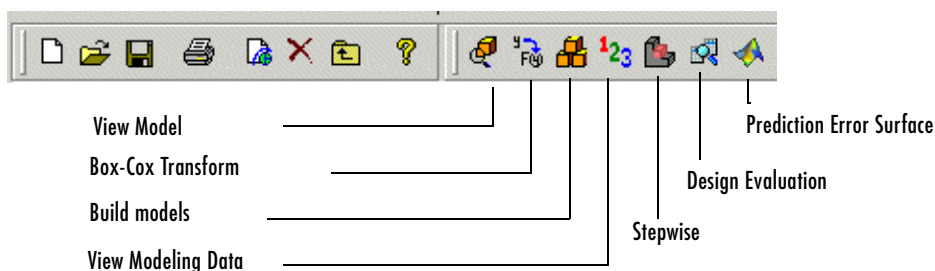
- **Stepwise** — This opens the **Stepwise Regression** window, where you can view the effects of removing and restoring model terms on the PRESS statistic (Predicted Error Sum of Squares), which is a measure of the predictive quality of a model. You can also use Min PRESS to remove all at once model terms that do not improve the predictive qualities of the model. See "Stepwise Regression Techniques" on page 8-13 for further discussion of the statistical effects of the **Stepwise** feature.

- **Design Evaluation** – Opens the Design Evaluation tool, where you can view properties of the design. See "Design Evaluation Tool" on page 8-30.

- **Prediction Error Surface** – Opens the Prediction Error Variance Viewer. See "Prediction Error Variance Viewer" on page 5-52.

Free-Knot Spline Models

Free knot spline models do not have any model-specific tools, just the standard View Model, Box-Cox Transform, Build Models and View Modeling Data.

Radial Basis Function Models
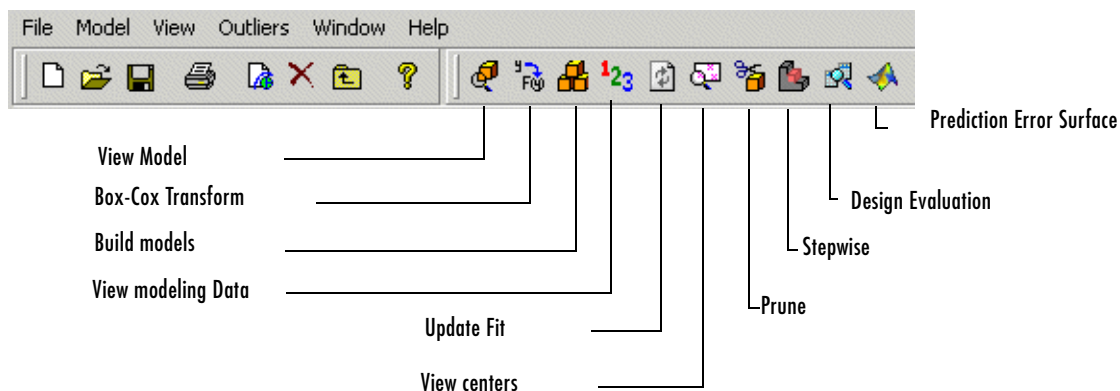
- **Update Fit** refits the RBF widths and centers. See "Tips for Modeling with Radial Basis Functions" on page 9-32 and "Fitting Routines" on page 9-10 in the Radial Basis Functions chapter.
- **View Centers** opens a dialog where you can view the position of the radial basis function's centers graphically and in table form.
- **Prune** opens the Number of Centers Selector where you can minimize various error statistics by decreasing the number of centers. See "Prune Functionality" on page 9-23.
- **Stepwise** opens the **Stepwise Regression** window.
- **Design Evaluation** – Opens the Design Evaluation tool, where you can view properties of the design. See "Design Evaluation Tool" on page 8-30.
- **Prediction Error Surface** – Opens the "Prediction Error Variance Viewer".

Hybrid RBFs have the same toolbar buttons as linear models.

MLE Models

This toolbar appears when you select any response feature that is an MLE model (purple icon). See "Global Level" on page 7-97 for other functionality in this view.

View Model

View Modeling Data
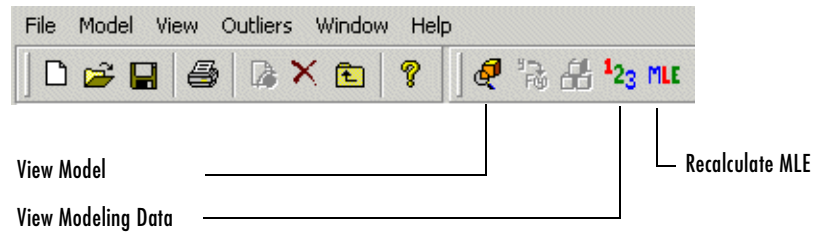
Recalculate MLE

At this point the **New child node,** Box-Cox and Build Models buttons are disabled.

- **Recalculate MLE** returns to the MLE dialog, where you can perform more iterations to try to refine the MLE model fit. See "MLE" on page 7-137 for more details.

Neural Networks

Neural net models have the **View Model**, **Build Models**, and **Update Fit** tools.

Global Level: Menus

File

Only the **New** (child node) and **Delete** (current node) functions change according to the node level currently selected. Otherwise the **File** menu remains constant.

See "File Menu" on page 7-8.

Window and Help Menus

The **Window** and **Help** menus have the same form throughout the Model Browser.

See "Window Menu" on page 7-10 and "Help Menu" on page 7-10.

Model Menu (Global Level)

```
Model
 Set Up...          Ctrl+M
 Reset
 Summary Statistics
 Evaluate...        Ctrl+E

 Box-Cox Transform
 Utilities                  ▶

 Make Template
 Build Models...
 Select...
 Assign Best
```
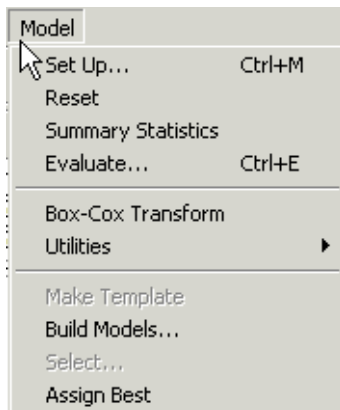
- **Set Up** opens the **Global Model Setup** dialog, where you can change the model type. See "Global Model Setup" on page 7-40.

- **Reset** returns to the global model defaults, that is, the global model specified at the test plan stage, restoring any removed outliers and removing any transforms.

- **Summary Statistics** opens the **Summary Statistics** dialog where you can select which statistics to display to help you evaluate models. In the global level view these appear in the **Summary Table** to the right of the plots, and in the **Models** list pane if there are child node models to compare. They also appear in the **Model Selection** window, and can be used to automatically select the best child node when using the **Build Models** dialog and **Local Multiple Models**. The standard summary statistics are PRESS RMSE (for linear models only) and RMSE, and these are always displayed. You can choose additional statistics for display in the **Summary Statistics** dialog by selecting the check boxes. You can also reach the dialog from the test plan. When you create child nodes the selected summary statistics are inherited. See "Summary Statistics" on page 7-24 for more information.

- **Evaluate** opens the **Model Evaluation** window.

- **Box-Cox Transform** opens the Box-Cox Transformation plots, where you can minimize SSE to try to improve the fit. See "Box-Cox Transformation" on page 8-18 for statistical details.

- **Utilities** –> opens a submenu showing the model-specific options available, duplicating the model-specific toolbar buttons (for example, Stepwise, Design Evaluation, View Centers, Prediction Error Surface, and so on).

- **Make Template** is available when child nodes exist. This opens a file browser where you can choose to save all the current child node models as a template, which you can then access using the **Build Models** menu item or toolbar button.

- **Build Models** opens the **Build Models** dialog. Here you can create a template or use a predefined template to build a selection of models as child nodes of the current node. The best model of this selection of child nodes will be automatically selected (it will have a blue icon), based on the selection criteria you choose in the following dialog (such as PRESS RMSE, RMSE, Box-Cox, Observations or Parameters). See the section "Creating Multiple Models To Compare" on page 2-42 in the Quick Start tutorial for a step by step example of using the **Build Models** dialog to quickly create many models to search for the best fit.
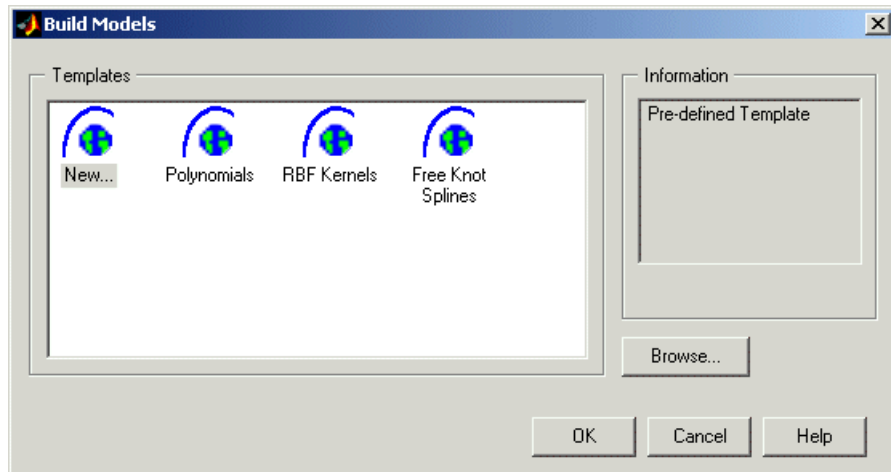
a Create a template for the type of models you want to build by clicking **New,** then clicking **OK**.

b Click **Add** in the following **Multi-Model Settings** dialog to reach the **Model Setup** dialog, where you can select any model type available for the number of inputs. Click **OK** to add the model and return to the **Multi-Model Settings** dialog.

c Click **Add** again to repeat the process to add as many different models as you like. You can click **Edit Model** to change the settings for any models in the list.

d When you are satisfied, click **OK** in the **Multi-Model Settings** dialog, and your chosen selection of model types are built as a selection of child nodes of the currently selected model node.

e The **Model Selection** dialog appears, where you can select the criterion for automatically choosing the best child node. Use the drop-down menu to select from the available criteria (such as from PRESS RMSE, RMSE, Box-Cox, Observations or Parameters). You can select additional criteria to appear here using the **Summary Statistics** options, from the **Model** menu. See "Model Menu (Global Level)" on page 7-107 and "Summary Statistics" on page 7-24. Click **OK** to accept the chosen criterion.

You can also save templates of whatever models you choose using the **Model –> Make Template** menu item from any modeling node. 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.

There are also predefined templates for polynomials, RBF kernels, and free knot splines. Note that free knot splines are only available for single inputs. When you select one of these templates, a dialog opens where you can specify the model settings:

- Polynomials – The **Model Building** dialog opens, where you can choose the initial and final order of the polynomials you want to build, and whether to use Stepwise settings. For example, if you choose 1 and 5 as the minimum and maximum polynomial order, 5 child node models are built (linear, quadratic, cubic, and so on).

- RBF Kernels – the **Radial Basis Function Options** dialog opens, where you can choose all the RBF settings. See "Types of Radial Basis Functions" on page 9-3. When you click **OK**, a family of RBF child node models are built using one of each kind of RBF kernel.

- Free Knot Splines – The **Model Building** dialog opens, where you can choose the initial and final number of knots. For example if you specify the initial and final numbers of knots as 1 and 5, five child nodes are built, one with one knot, one with two and so on. Only available for single inputs.

• **Select** is available whenever the **Select** button is also enabled in the lower right pane (when it is titled **Local Models**, **Response Features**, or **Models**).

This item opens the **Model Selection** window to allow you to choose the best model. See "Select Button" on page 7-111.

- **Assign Best** selects the current model as best. If it is one of several child node models of a global model, selecting it as best means that it is duplicated at the parent global model. See "Model Tree" on page 7-11.

View Menu (Global Level)

- **Model Definition** opens the **Model Viewer** dialog displaying the model terms.
- **Modeling Data** opens the Data Editor to display a read only version of the inputs and predicted and actual responses. Here you can view and export your modeling data. Also in the toolbar.
- **Test Numbers** turns test numbers on and off for both the special and scatter plots. Also available in the right-click plot menus.

Outliers Menu (Global Level)

This is the same as the local level **Outliers** menu, except that there is no **Remove All Data** command. All items are duplicated in the right-click context menu on the plots, except **Selection Criteria** and **Copy Outliers**. See "Outliers Menu (Local Level)" on page 7-92.

At global level, as at local level, the **Restore Removed Data** item opens the **Restore Removed Data** dialog, where you can choose the points to restore from the list, or restore all available points. Select points in the left list and use the buttons to move points between the lists. Note that entire tests removed at the local level (using the **Remove All Data** item) cannot be restored at global level.

Selecting Models

The **Model Selection** window appears when you click the **Select...** button. This window is intended to help you select a **Best Model** by comparing several candidate models. For information on how to select best models, see "Model Selection Guide" on page 7-113.

The **Model Selection** window allows visual comparison of several models in many ways, depending on the type of model:

- "Tests View" on page 7-119
- "Predicted/Observed View" on page 7-121
- "Response Surface View" on page 7-124
- "Likelihood View" on page 7-128
- "RMSE View" on page 7-130
- "Residuals View" on page 7-132
- "Cross Section View" on page 7-133

Select Button

The **Select** button is under the list view in the pane at the bottom of the Model Browser display. This pane is the **Test Plans** list pane at startup and changes title depending on the level in the model tree that is selected. The list box in this pane always contains the child nodes of whichever node in the tree is selected.

The pane also always contains three buttons: **New**, **Delete**, and **Select**.

Select is only available when the lower pane lists local models, response models, or models.

You can select among the following:

- Local models
- Response features
- Submodels of response features

But you cannot select between response models or test plans.

Note To get a two-stage model at the response node, you must use the **Select** button at the local node (when the lower pane is the **Response Features** pane) to assign a model (even if it is the only one) as best. This step then calculates the two-stage model.

Select might not be available if you are not ready to choose among the child nodes. For example, at the response node, the child nodes must have models assigned as best (using the **Select** feature at those levels) before you can select among them. Also, if a response feature has child nodes of alternate models, you must select the best, or the Browser cannot tell which to use to calculate that response feature. After calculating MLE, **Select** compares the MLE model with the previous univariate model, and you can choose the best.

The **Model Selection** window allows visual comparison of several models. From the response level you can compare several two-stage models. From the local level, if you have added new response features you can compare the different two-stage models (constructed using different combinations of response feature models). If you have added child nodes to response feature models, you can compare them all using the **Model Selection** window.

When a model is selected as best it is copied up a level in the tree together with the outliers for that model fit.

A tree node is automatically selected as best if it is the only child, except two-stage models which are never automatically selected - you must use the **Model Selection** window.

If a best model node is changed the parent node loses best model status (but the automatic selection process will reselect that best model if it is the only child node).

Note You can bypass the Model Selection window if you want to quickly assign a model as best (unless you need to calculate a two-stage model). Select the model node in the tree, then select **Model –> Assign Best**.

Model Selection Guide

Firstly it is important to point out that there is no recipe for model selection. It is not possible to cover the entire topic of Model Selection in a few paragraphs. Instead, we outline some general guidelines which should be helpful in using the Model-Based Calibration Toolbox to choose the best model for a given data set. There are many books you can go to for a fuller account of statistical modeling; see "References" on page 7-117.

Overfitting and Underfitting

When fitting a model to noisy data, we effectively make the fundamental assumption that the data have been generated from some model (the "truth") by making predictions at given values of the inputs, then adding some amount of noise to each point, where the noise is drawn from a normal distribution with an unknown variance.

Our task is to discover both this model and the width of the noise distribution. In doing so, we aim for a compromise between *bias*, where our model does not follow the right trend in the data (and so does not match well with the underlying truth), and *variance*, where our model fits the data points too closely, and so "chases" the noise rather than trying to capture the true trend. These two extremes are known as underfitting and overfitting.

An important concept in this context is the *number of parameters* in a model. As this number increases, the model can bend in more complicated ways. If the number of parameters in our model is larger than that in the truth, then we risk overfitting, and if our model contains fewer parameters than the truth, we could underfit.

RMSE

Our basic measure of how closely a model fits some data is the Root Mean Squared Error (RMSE), 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. When two-stage modeling, use the RMSE Explorer to quickly investigate the local models with highest RMSE.

The smaller the RMSE, the closer our model follows the data; if a model goes through each data point exactly, then the RMSE is zero. The illustration shows how increasing the number of parameters in the model can result in overfitting. The 9 data points (shown as black circles) are generated from a

cubic polynomial which contains 4 parameters (the "truth", shown as the black curve) by adding a known amount of noise. We can see that by selecting candidate models containing more parameters than the truth, we can reduce, and even eliminate, any mismatch between the data points and our model, causing the RMSE to vanish. This latter case occurs when the number of parameters in the model is the same as the number of data points (an 8th order polynomial has 9 parameters).

This does not mean that we have obtained a good fit - the model is overfitting, as we can see from the large difference between the model and the truth in the regions between the data points. By forcing our model to go though all the data points, we have included too much structure in the curve, which reduces the quality of the fit away from the data points. Similarly, if we use a model with fewer parameters than in the truth, we risk underfitting; our model is not flexible enough to match the truth well.This is shown in the following illustration.
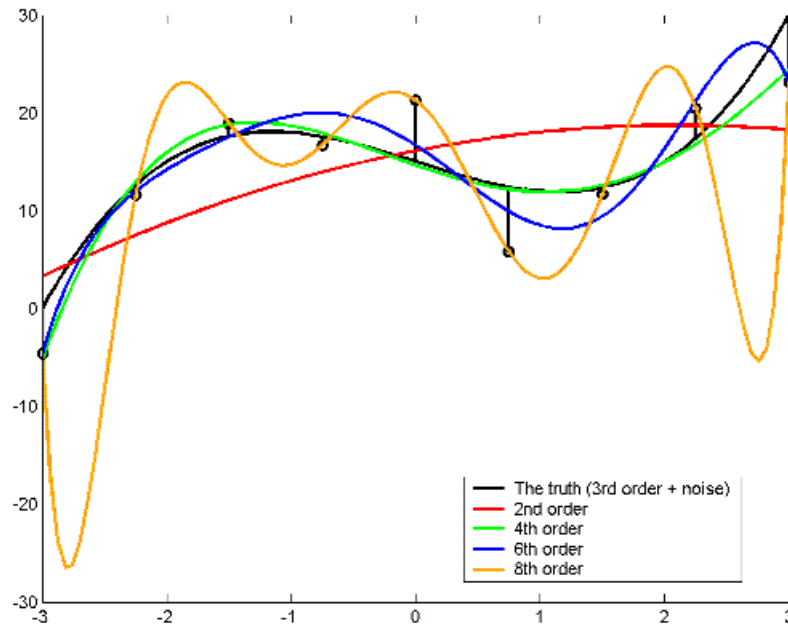
Other Statistics

As illustrated above, relying solely on RMSE can result in overfitting, which leads to poor model performance away from regions containing data points. In general, this problem is tackled by replacing RMSE with some other statistic, which also must be reduced to improve the fit, but which is designed to rise when we start overfitting. This is why you should consider RMSE *and* another tool such as the PRESS statistic to help you decide on the best model.

PRESS RMSE (Predicted sum of squares) is calculated in a similar way to RMSE, except we remove a data point from the fit, and ask the model to predict where that point lies with no knowledge of the data in that area. To calculate PRESS RMSE, this process is repeated for each point in the data set and the results are averaged. If the value of PRESS RMSE is much bigger than the RMSE then we are overfitting. Weighted PRESS and GCV are also derived from this idea.

A different approach to solving the problem of overfitting results in statistics known as "Information Criteria", such as AIC and BIC. Here, we combine a term involving RMSE with a term that rises with the number of parameters in the model. This explicitly penalises a model for an increase in its level of structure. Both AIC and BIC are approximations, which get more accurate as the number of observations increases. In general, we do not recommend using them unless the ratio of the numbers of observations to parameters is greater than 40:1 (see Section 2.4 of Burnham and Anderson, "References" on page 7-117). AICc, however, can be used with smaller samples and is the most appropriate information criterion for most problems in engine calibration.

The absolute value of AICc for a given model includes an arbitrary constant, and so is of no direct use. However, the difference between the AICc value for two models is meaningful: one rule of thumb says that if this difference is greater than about 10, then the worse model can be neglected in the selection process (see Section 2.6 of Burnham and Anderson, "References" on page 7-117).

Validation

All of the statistics mentioned above attempt to yield a model which makes good predictions both at the data points, and in the regions in between the data points. The simplest way to confirm that this is the case, is to collect additional data and test (or "validate") the model against this new data, by evaluating the new RMSE based on these data. Comparing a validation RMSE with the RMSE based on the modeling data is a good model selection statistic. You can use the

Model Evaluation window to validate models against other data, see "Model Evaluation Window" on page 7-144.

How much validation data to collect (or whether it is feasible to collect any at all) are matters governed primarily by practical considerations.

Trends

Throughout this discussion, we have focussed on using statistics for model selection. It is advisable, however, to combine a study of the model statistics with a careful examination of the trends present in the models. It would be a mistake to underestimate the importance of using engineering knowledge as a tool for comparing models. In addition, if two or more models of a different type (e.g. two different RBF kernel functions) follow the same trend, then that lends confidence to those models, because they are likely to be picking up real structure in the data. You can use the cross-section view in the **Model Selection** window to plot multiple models on the same axes to aid this process.

Where to Find Statistics For Comparing Models

- Look for RMSE and PRESS RMSE values in the model lists at the bottom of the **Model Selection** window and the Model Browser local, global and response views. Use these to compare models.

- To use other statistics such as information criteria (e.g. AICc) to compare models, you can specify extra statistics at the test plan and global levels. From any global model node select **Model –> Summary Statistics**, and use the check boxes to select additional statistics to display. The choices you make in the **Summary Statistics** dialog determine what summary statistics are displayed in the global model views and the **Model Selection** window. These choices also determine what criteria you can use for automatically selecting best models in the Local Model Setup dialog for Multiple Models, and the Build Models dialog. Note, you must select the summary statistics before creating the child models to compare. See "Summary Statistics" on page 7-24.

- Use the **RMSE Explorer** for local models; look for tests with high RMSE and investigate. Select **View –> RMSE Plots**, or use the toolbar button.

- Use the **Model Evaluation** window for validating against other data and you will see a Fit RMSE reported alongside the validation RMSE for comparison.

References

- Classical Regression: Draper and Smith, Applied Regression Analysis (3rd edition), John Wiley & Sons 1998

- BIC Methods: Kass and Raftery, Bayes factors. Journal of the American Statistical Association, 90(1995):773-795.

- AIC Methods: Burnham and Anderson, Model Selection and Multimodel Inference: A Practical Information-Theoretic Approach (2nd edition), Springer-Verlag 2002.

Model Selection Window

The **Model Selection** window comprises several different views depending on the type of models being compared:

- "Tests View" on page 7-119
- "Predicted/Observed View" on page 7-121
- "Response Surface View" on page 7-124
- "Likelihood View" on page 7-128
- "RMSE View" on page 7-130
- "Residuals View" on page 7-132
- "Cross Section View" on page 7-133

You can change to any available view in the Model Selection window using the **View** menu or by clicking the buttons of the toolbar.

The **Assign Best** button at the bottom of the window marks the currently selected model as best. or you can double-click a model in the list.

Information about each candidate model is displayed in the list at the bottom. The information includes categories such as the number of observations and parameters, and various diagnostic statistics such as RMSE and PRESS RMSE. You can click on column headers in this list to sort models by that category — for example, clicking on the column header for PRESS RMSE sorts the models in order of increasing PRESS RMSE. As this statistic is an indication of the predictive power of the model, it is a useful diagnostic statistic to look at (the lower the better), but remember to also look at other factors.

Note The diagnostic statistics displayed depend on your choices in the **Summary Statistics** dialog for the parent model node. See "Summary Statistics" on page 7-24 for details.

To print the current view, use the **Figure/Print** menu item or its hot key equivalent **Ctrl**+P. In the Response Surface view you can also use the right-click context menu.

To close the **Model Selection** window, use the **Figure/Close** menu item or its hot key equivalent **Ctrl**+W. This window is intended to help you select a best model by comparing several candidate models. On closing the figure, you are asked to confirm the model you chose as best.

Tests View

For a two-stage model the initial view is as follows:
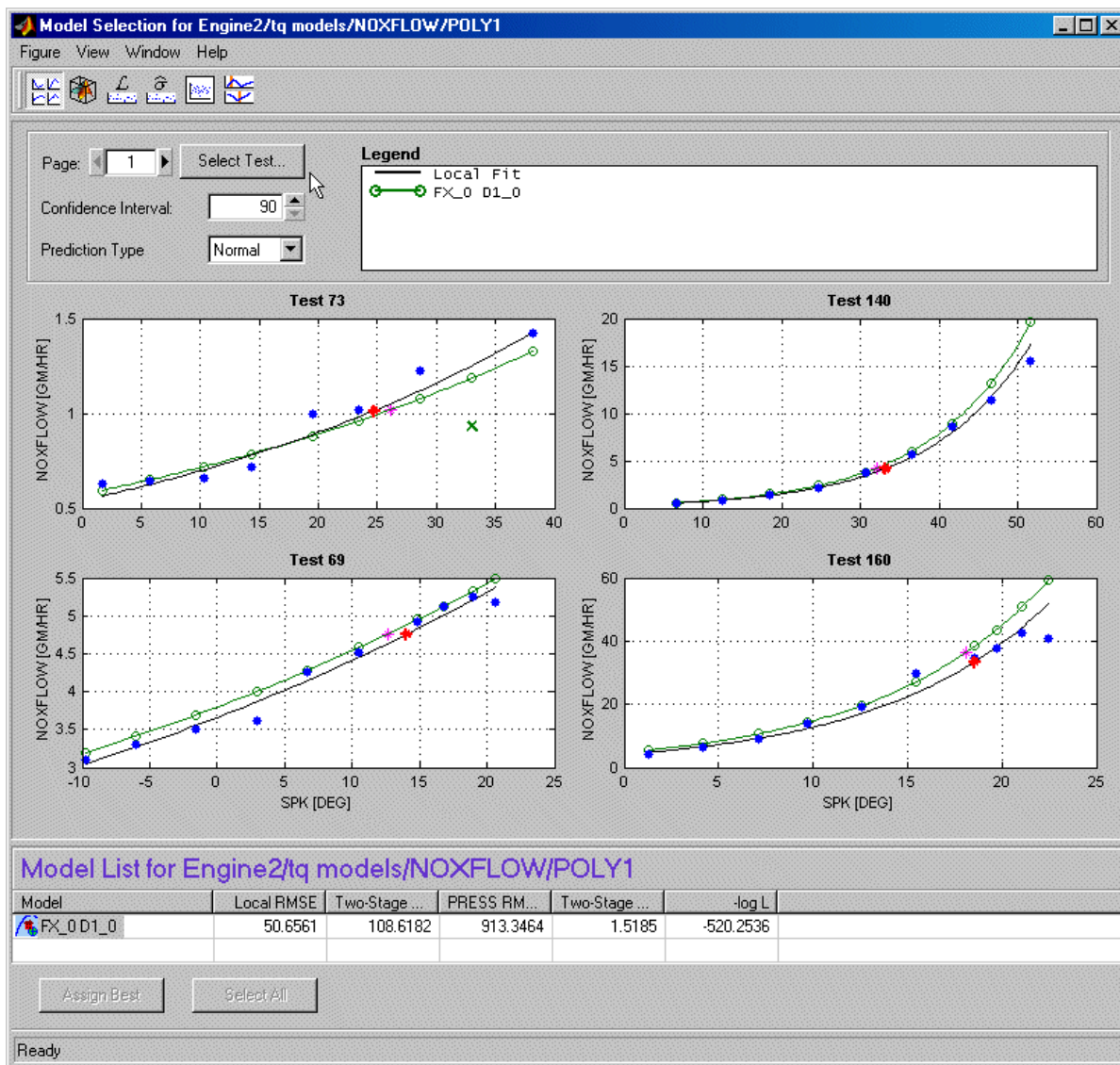
The tests view shows the data being modeled (blue dots) and two models that have been fitted to this data. The black line shows the local model that has been fitted to each test separately. The green line shows the two-stage model: you can see the local model curve reconstructed using response feature values taken from the global models.

If the local input has more than one factor, a predicted/observed plot appears.

This view allows you to compare several models simultaneously. Using standard Windows multiselect behavior (**Shift**+click and **Ctrl**+click) in the list view, or by clicking the **Select All** button, you can view several two-stage models together. A maximum of five models can be selected at once. The legend allows you to identify the different plot lines.

Clicking one of the plots (and holding the mouse button down) displays information about the data for that test. For example:
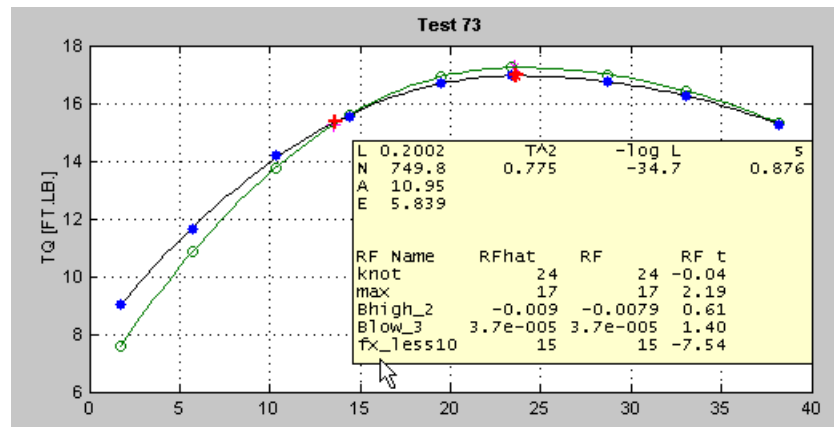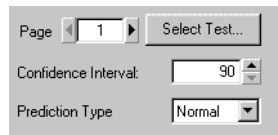
Here you see the values of the global variables for this test and some diagnostic statistics describing the model fit. Also displayed are the values (for this test) of the response features used to build this two-stage model and the two-stage model's estimation of these response features.

The controls allow navigation between tests.

You can change the size of the confidence intervals; these are displayed using a right-click menu on the plots themselves.

The prediction type allows a choice of Normal or PRESS (Predicted Error Sum of Squares) — although not if you entered this view through model evaluation (rather than model selection). PRESS predictions give an indication of the model fit if that test was not used in fitting the model. For more on PRESS see "Linear Model Statistics Displays" on page 8-22 and "Stepwise Regression Techniques" on page 8-13.

Predicted/Observed View

For a one-stage model, or when you are comparing different models for one **Response Feature,** the initial view is as follows:
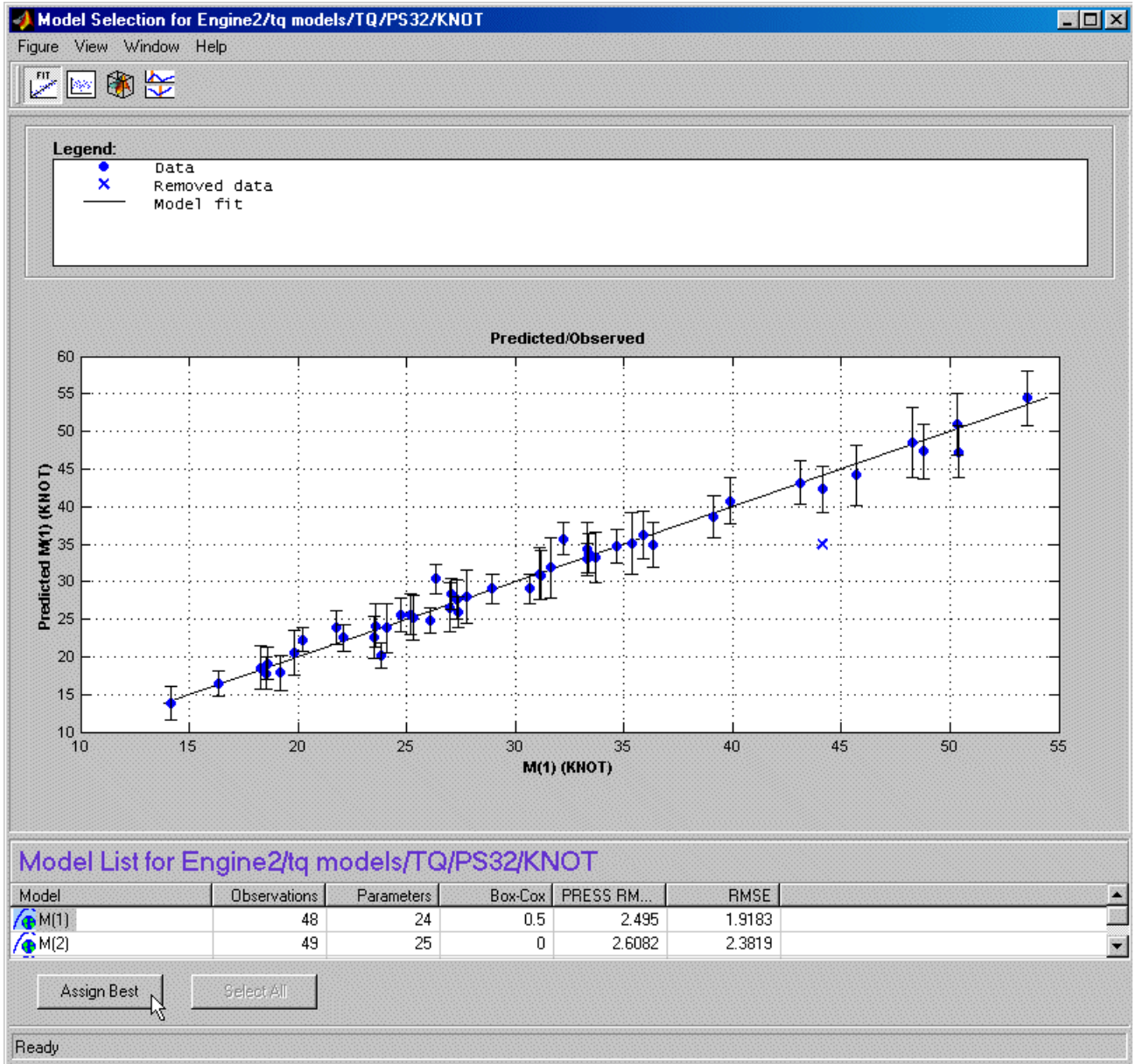
The plot shows the data used to fit this model (values of knot), against the values found by evaluating the model (here Predicted knot) at these data points. The straight black line is the plot of y=x. If the model fitted the data exactly, all the blue points would lie on this line. The error bars show the 95% confidence interval of the model fit.

For single inputs, the response is plotted directly against the input.

The Predicted/Observed view only allows single selection of models for display.

Response Surface View

This view shows the model surface in a variety of ways.

The default view is a 3-D plot of the model surface, as in the example. This model has five dependent factors; you can see these in the controls at the top left (there is a scroll bar as only four can be seen at once at this size of window).

You can choose which input factors to display by using the drop-down menus below the plot. The unselected input factors are held constant and you can change their values using the controls at the top left of the view (either by clicking the arrow buttons or by typing directly in the edit box).

Display using (S - **datum)** — If a datum model is being displayed, this check box appears. The datum variable here is spark angle, S. When you select this box, the model is displayed in terms of spark angle relative to the datum. The appropriate local variable name appears here. See "Datum Models" on page 7-79.

Display boundary constraint — If you have boundary models you can display them by selecting the check box. Areas outside the boundary are yellow, as shown in the example. Areas outside the boundary are yellow (or gray in table view). They are shown on all display types (contour, 2-D, surface, movie and table).

Display Type — Changes the model plot. Display options are available for some of these views and are described under the relevant view. The choices are as follows:

- A table showing the model evaluated at a series of input factor values.
- A 2-D plot against one input factor.
- A 2-D plot with several lines on it (called a multiline plot); this shows variation against two input factors.

- A contour plot.

  The **Contours...** button opens the **Contour Values** dialog. Here you can set the number, position, and coloring of contour lines.

  Fill Contour colors each space between contours a different color.

  Contour Labels toggles the contour value numbers on and off. Without labels a color bar is shown to give you a scale.

  Auto (the default) automatically generates contours across the model range.

  N Contour Lines opens an edit box where you can enter any number of contour lines you want.

  Specify values opens an edit box where you can enter the start and end values where you want contour lines to appear, separated by a colon. For example, entering 5:15 gives you 10 contour lines from 5 to 15. You can also enter the interval between the start and end values; for example 1:100:600 gives you contour lines between 1 and 600 at intervals of 100.

- A surface (shown in the example).

  Prediction error shading — Colors the surface in terms of the prediction error (sqrt(Prediction Error Variance)) of the model at each point. A color bar appears, to show the value associated with each color.

Note For datum models, Prediction Error shading is only available when the **Display using (***local variable*** - datum)** check box is not selected.

  P. E. Threshold — To see good color contrast in the range of PE of interest, you can set the upper limit of the coloring range. All values above this threshold are colored as maximum P.E.

- A movie: this is a sequence of surfaces as a third input factor's value changes.
  - **Replay** replays the movie.
  - **Frame/sec** selects the speed of movie replay.
  - The number of frames in the movie is defined by the number of points in the input factor control (in the array at the top left) that corresponds to the **time factor** below the plot.

Export allows the currently displayed model surface to be saved to a MAT file or to the MATLAB workspace.

Right-click on the plot to reach the context menu and change many display properties (lighting, colormap etc.) and print to figure.

Within a test plan the memory is retained of the evaluation region, plot type and the number of points resolution last displayed in the Response Surface view.

Likelihood View

The likelihood view shows two plots relating to the log likelihood function evaluated at each test. It is useful for identifying problem tests for maximum likelihood estimation (MLE).
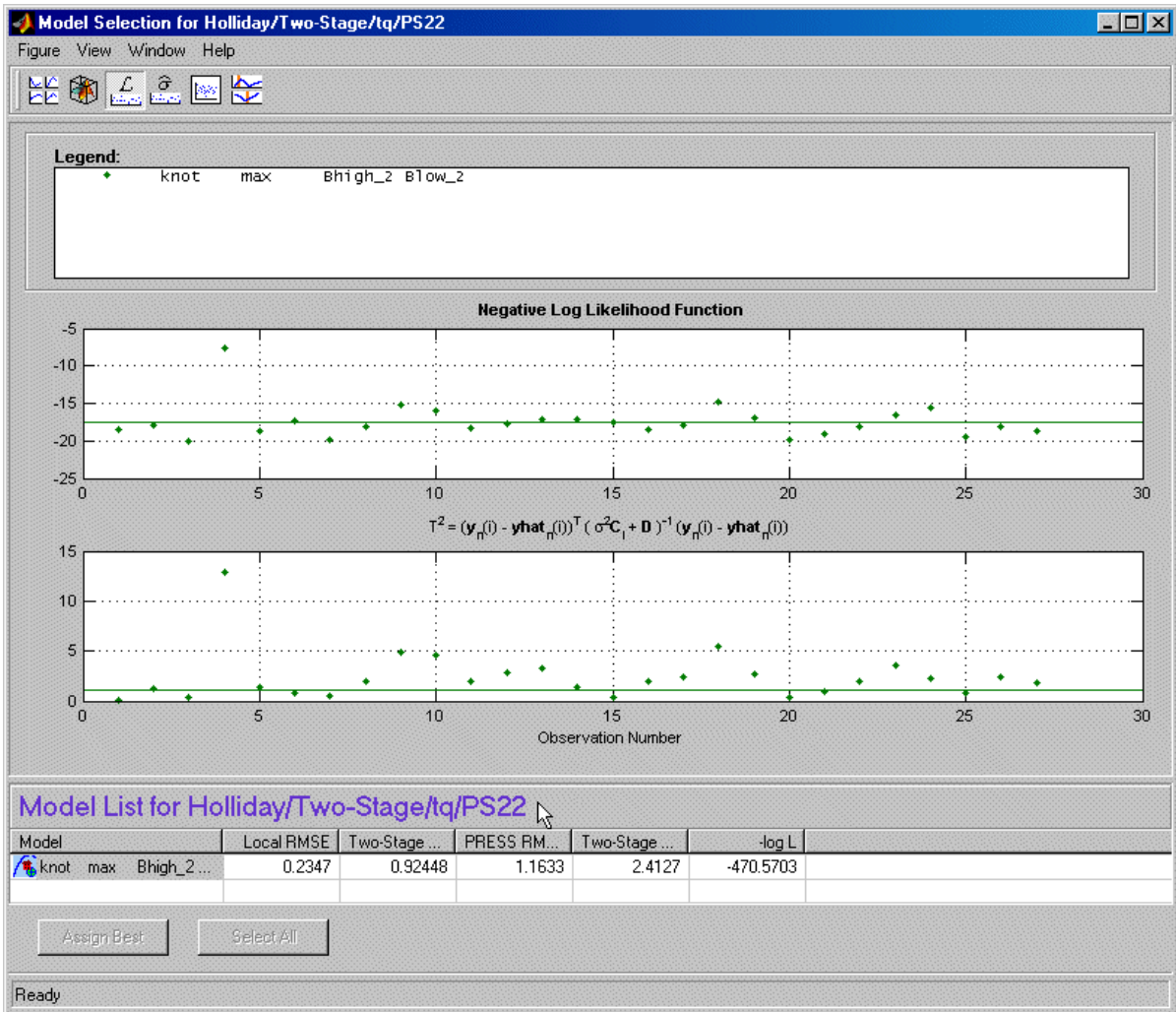
Each plot has a right-click menu that allows test numbers to be displayed on the plots and also offers autoscaling of the plots. You can also **Print to Figure**.

The likelihood view allows several models to be displayed simultaneously; click the **Select All** button at the bottom of the window or, in the model list view, **Shift**+click or **Ctrl**+click to select the models for display.

The upper plot shows values of the negative log likelihood function for each test. This shows the contribution of each test to the overall negative log likelihood function for the model, as compared with the average, as indicated by the horizontal green line.

The lower plot shows values of the T-squared statistic for each test. This is a weighted sum squared error of the response feature models for each test. As above, the purpose of this plot is to show how each test contributes to the overall T-squared statistic for this model. The horizontal line indicates the average.

RMSE View

The Root Mean Square Errors view has three different plots, each showing standard errors in the model fit for each test.
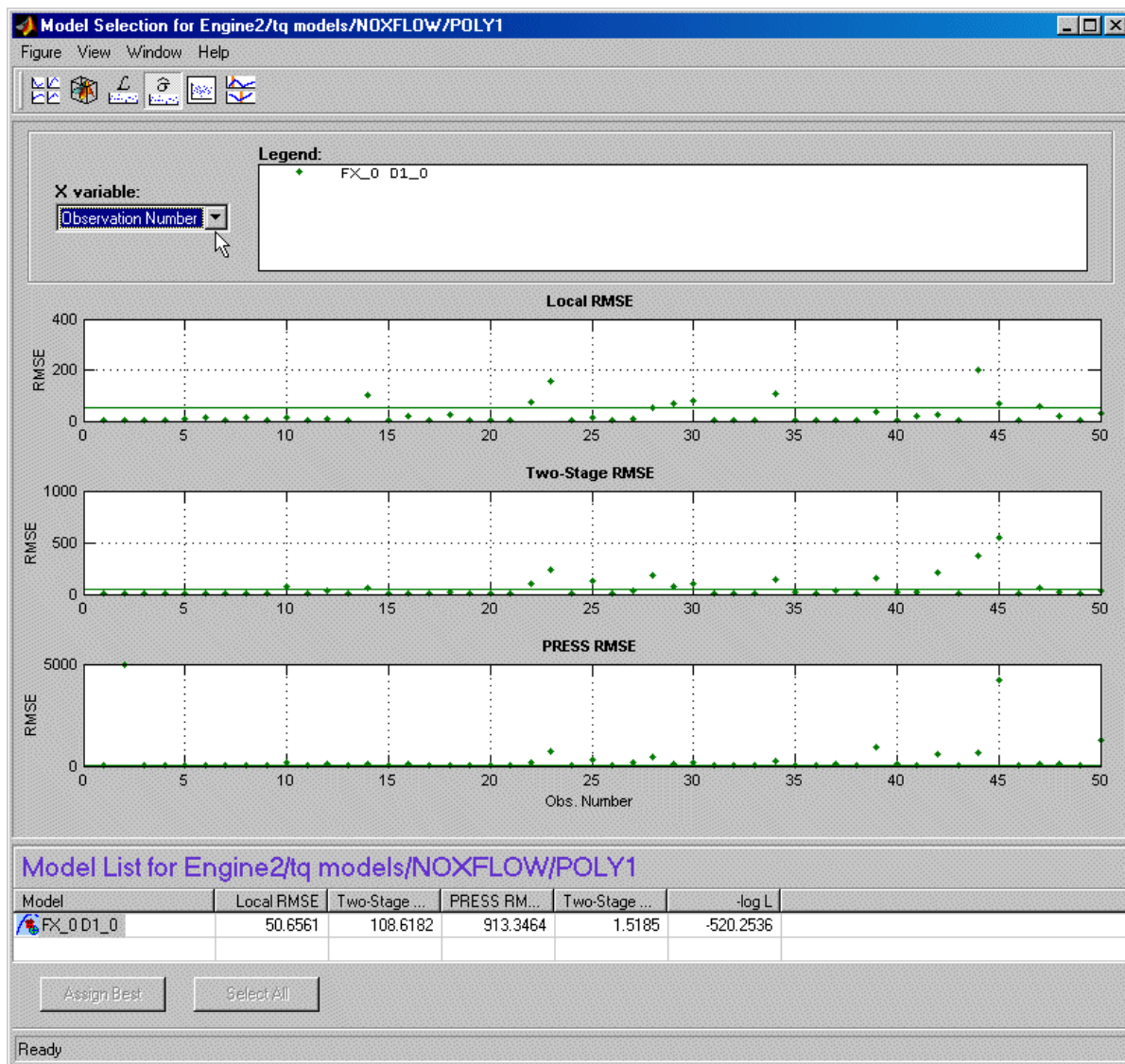
Each plot has a right-click menu that allows test numbers to be displayed on the plots, and you can **Print to Figure**.

The **X variable** menu allows you to use different variables as the *x*-axis of these plots.

The RMSE view allows several models to be displayed simultaneously; click the **Select All** button at the bottom of the window or, in the model list view, **Shift**+click or **Ctrl**+click to select the models for display.

Local RMSE shows the root mean squared error in the local model fit for each test.

Two-Stage RMSE shows the root mean squared error in the two-stage model fit to the data for each test. You should expect this to be higher than the local RMSE.

PRESS RMSE is available when all response feature models are linear. This plot shows the root mean squared error in the PRESS two-stage model fit at each test.

For information on PRESS RMSE see "Linear Model Statistics Displays" on page 8-22.

Residuals View

The residuals view shows the scatter plots of observation number, predicted and observed response, input factors, and residuals.
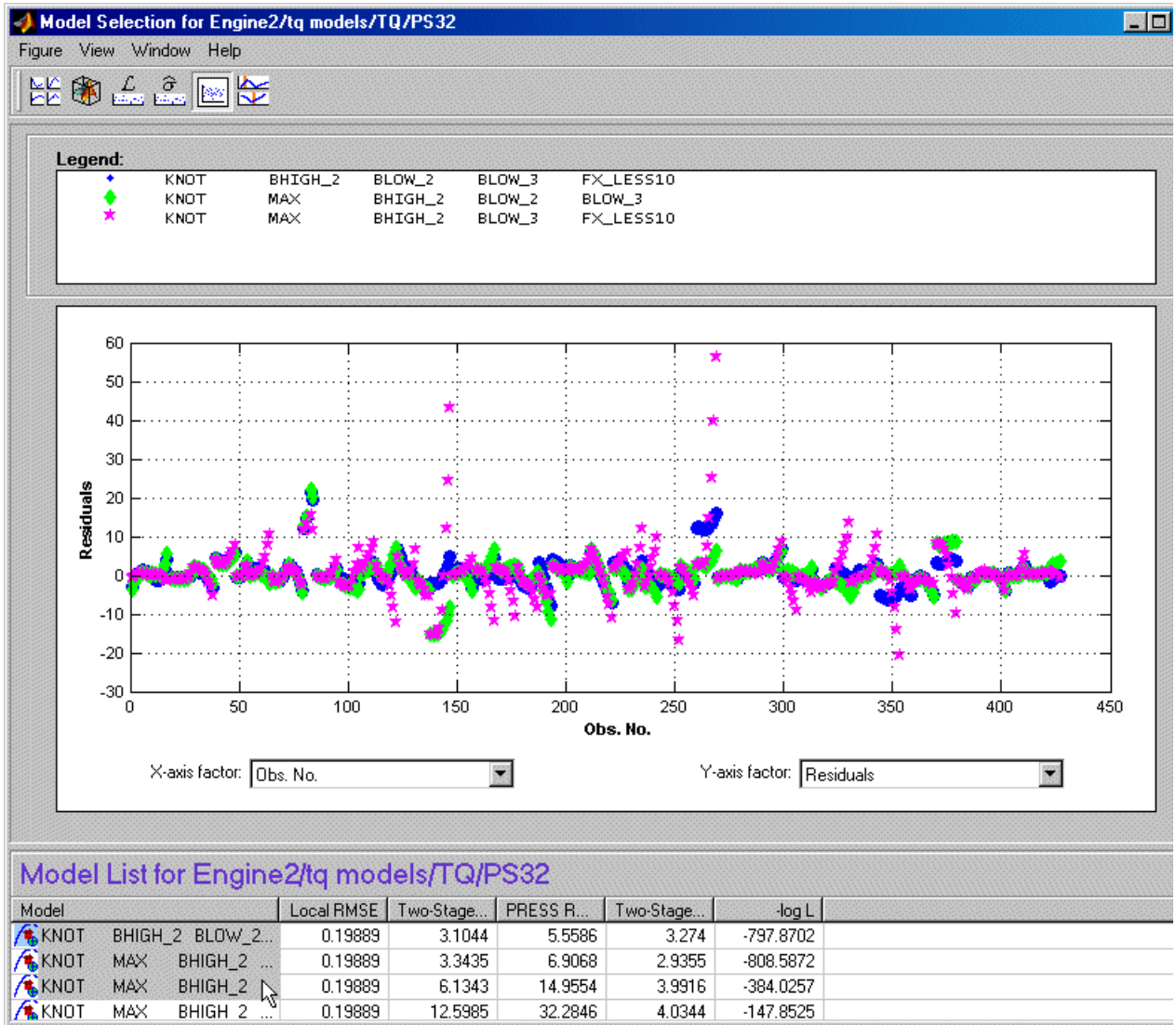
This view allows several models to be displayed simultaneously (as shown in this example); click the **Select All** button at the bottom of the window or, in the model list view, **Shift**+click or **Ctrl**+click to select the models for display.

A right-click menu allows the test number of each point to be displayed when only one model is being displayed.

The **x-axis factor** and **y-axis factor** menus allow you to display various statistics.

Cross Section View

The cross-section view shows an array of cross sections through the model surface. You can choose the point of cross section in each factor. Data points near cross sections are displayed, and you can alter the tolerances to determine how much data is shown. The only exception is when you evaluate a model without data; in this case no data points are displayed.

You can select individual data points by test number (using the **Select Data Point** button). You can double-click a data point in a graph to take the display directly to that point. You can choose to use a common Y-axis limit for all graphs using the check box.

If you have boundary models you can choose to display them here using the check box; regions outside the boundary are yellow, as shown in the example.

Within a test plan the memory is retained of the point last displayed in the Cross Section view; when you reopen the view you return to the same point.
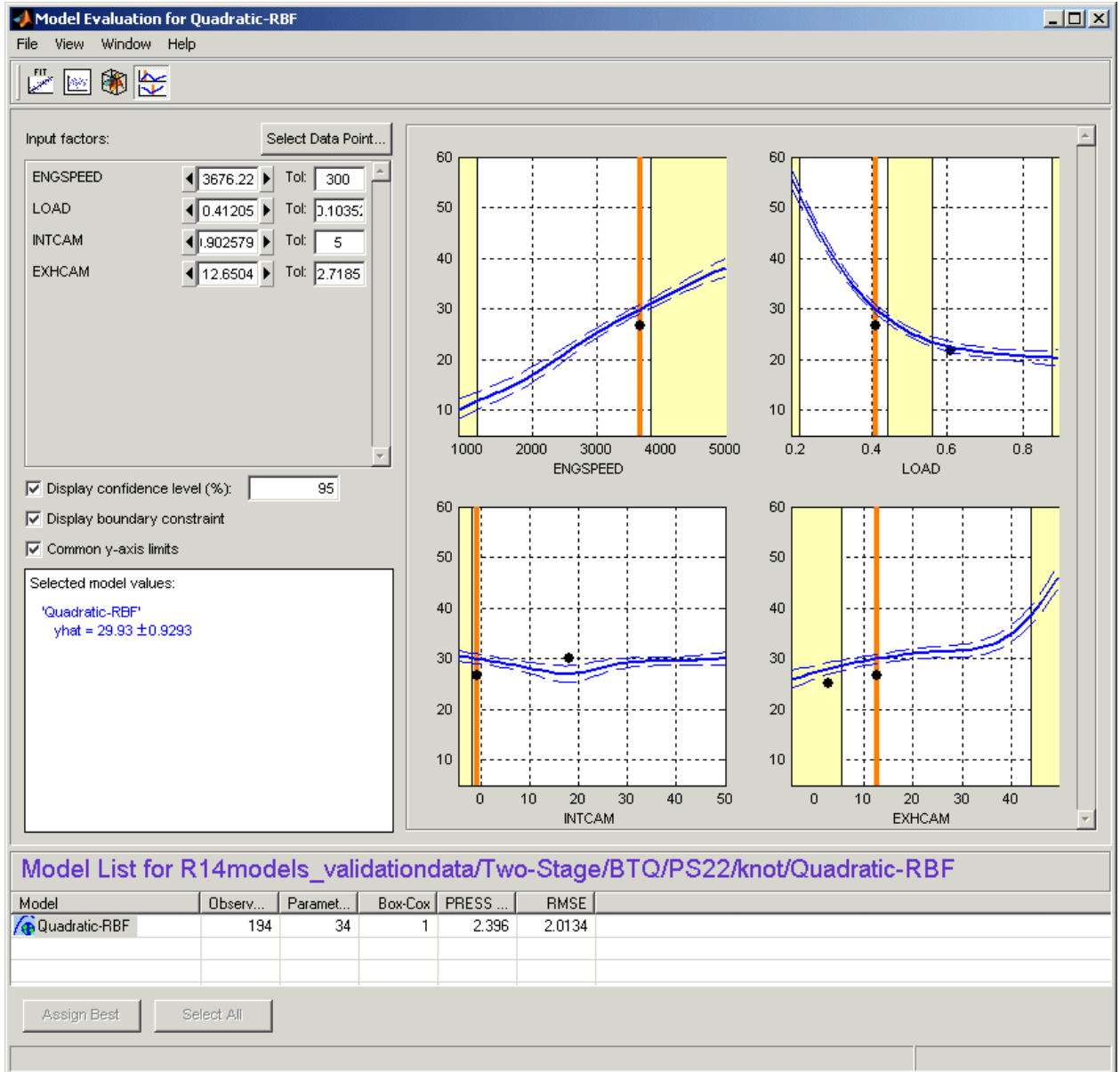
The number of plots is the same as the number of input factors to the model. The plot in **S** shows the value of the model for a range of values of **S** while the other input factors are held constant. Their values are displayed in the controls at the top left, and are indicated on the plots by the vertical orange bars.

- You can change the values of the input factors by dragging the orange bars on the plots, using the buttons on the controls, or by typing directly into the edit boxes.

- For example, changing the value of N to 1000 (in any of these ways) does nothing to the graph of N, but all the other factor plots now show cross sections through the model surface at N = 1000 (and the values of the other variables shown in the controls).

On the plots, the dotted lines indicate a confidence interval around the model. You define the confidence associated with these bounding lines using the **Plot confidence level (%)** edit box. You can toggle confidence intervals on and off using the check box on this control.

For each model displayed, the value of the model and the confidence interval around this are recorded in the legend at the lower left. The text colors match the plot colors. In the example shown, two models are selected for display, resulting in blue (PS22 model) and green (POLY2 model) legends on the left to correspond with the blue and green plots. You can select multiple models to display in the list at the bottom using **Ctrl**+click, or click **Select All**. The values of the input factors (for which the model is evaluated) can be found in the controls (in the **Input Factors** pane) and seen as the orange lines on the plots.

Data points are displayed when they fall within the tolerance limit near each cross section. You can set the tolerance in the **Tol** edit boxes.

- For example, if N is set to 1000, and the tolerance for N is set to 500, all data points with values between N = 500 and N = 1500 appear on the plots of the other factors.

- This means that changing the tolerance in one factor affects the data points that appear on the plots of all the other factors. It does not affect the plots of that factor.

- You can click data points in the plots to see their values. Several points can mask each other; in this case the values of all coincident data points are displayed. Double-click to move the display directly to a data point.

The following example illustrates how the tolerance level determines which data points are displayed. The tolerance for TP_REL (500) includes all points in the data set (this is an extreme example). The plot for N therefore shows the data points for all the tests. Note that you can see the structure of the data as each test shows as a vertical line of points.

You can see that the orange line on the N plot passes through a test. This orange line shows the value of N for the cross-section plot of TP_REL. You can also read the value in the edit box (N=1753.3). The tolerance for N (200) only includes data points of this test. Data in adjacent tests fall outside this tolerance. Therefore the TP_REL plot shows the data points from one test only.

Increasing the tolerance on N will mean that more data points fall within the tolerance and so would appear on the TP_REL plot.
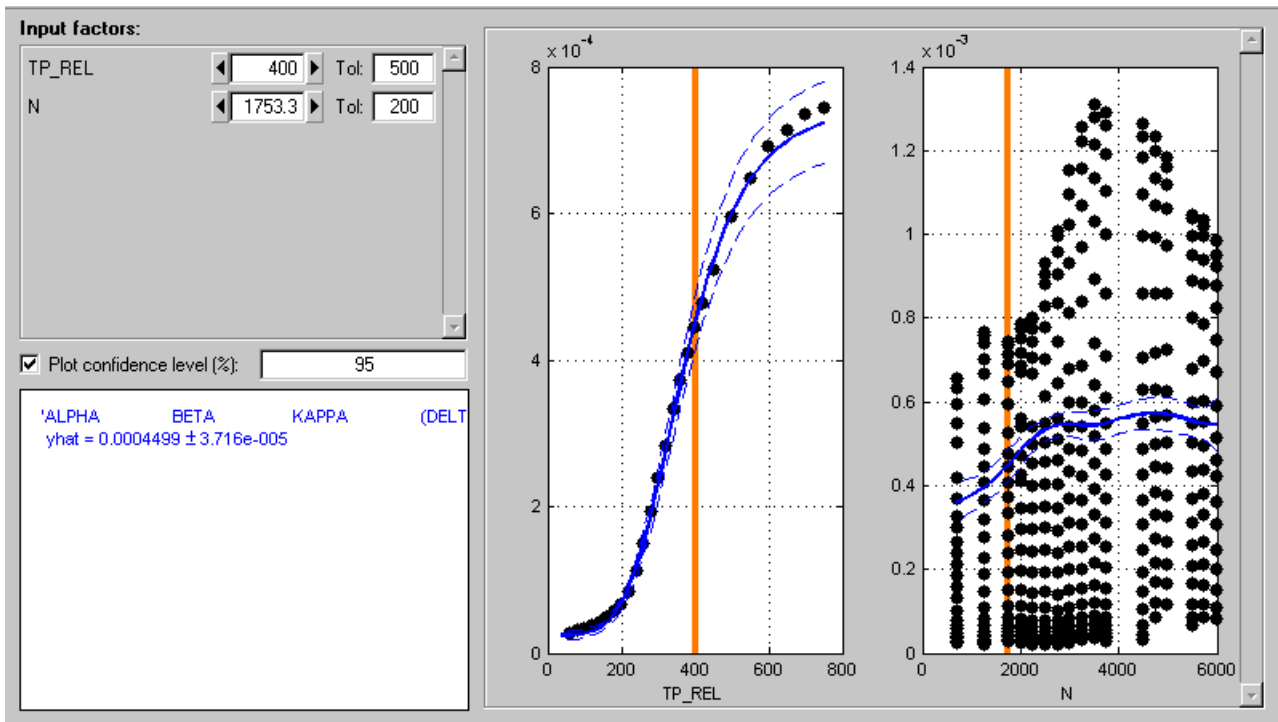
MLE

For an ordinary (univariate) two-stage model, the global models are 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.

Calculating MLE

When you close the **Model Selection** window, a dialog asks if you want to calculate MLE. If you click **Cancel** at this point, you can calculate MLE later as follows:

1 From the local node, click the **MLE** icon in the toolbar .

 Alternatively, choose **Model –> Calculate MLE**.

2 The MLE dialog appears. Click **Start**.

 You can alter various MLE settings on this dialog.

3 After you click **Start** a series of progress messages appears, and when finished a new Two-Stage RMSE (root mean square error) value is reported.

4 You can perform more iterations by clicking **Start** again to see how the RMSE value changes, or you can click **Stop** at any time.

5 Clicking **OK** returns you to the Model Browser, where you can view the new MLE model fit.

Note After calculating MLE, you will notice that the plots and the icons in the model tree for the whole two-stage model (response node, local node, and all response feature nodes) have turned purple. See "Icons: Blue Backgrounds and Purple Worlds" on page 7-14.

You can select all response features in turn to inspect their properties graphically; the plots are all purple to symbolize MLE. At the local node the plots show the purple MLE curves against the black local fit and the blue data.

- From the response feature nodes, at any time, you can use the **MLE** toolbar icon to **Recalculate MLE** and perform more iterations.

- From the local node, you can click **Select** to enter the **Model Selection** window, compare the MLE model with the previous univariate model (without correlations), and choose the best. Here you can select the univariate model and click **Assign Best** to "undo" MLE and return to the previous model.

Note If there are exactly enough response features for the model, you can go straight to MLE calculation after model setup without going through the **Select** process. The **MLE** toolbar button and the **Model –> Calculate MLE** item are both active in this case. If you add new response features, you cannot create MLE until you go through model selection to choose the response features to use.

MLE Settings
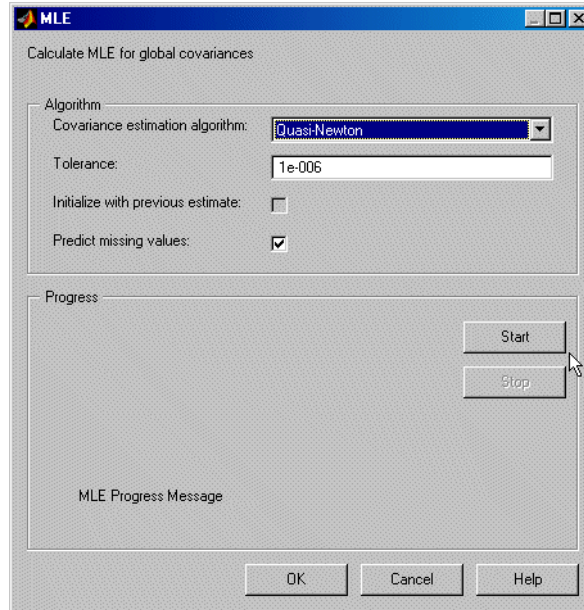
Algorithm

The algorithm drop-down menu offers a choice between two covariance estimation algorithms, Quasi-Newton and Expectation Maximization. These are algorithms for estimating the covariance matrix for the global models.

Quasi-Newton is recommended for smaller problems (< 5 response features and < 100 tests). Quasi-Newton usually produces better answers (smaller values of -logL) and hence is the default for small problems.

Expectation Maximization is an iterative method for calculating the global covariance (as described in Davidian and Giltinan (1995); see References in "Two-Stage Models for Engines" on page 8-40). This algorithm has slow convergence, so you might want to use the **Stop** button.

Tolerance

You can edit the tolerance value. Tolerance is used to specify a stopping condition for the algorithm. The default values are usually appropriate, and if calculation is taking too long you can always click **Stop**.

Initialize with previous estimate

When you recalculate MLE (that is, perform more iterations), there is a check box you can use to initialize with the previous estimate.

Predict missing values

The other check box (selected by default) predicts missing values. When it is selected, response features that are outliers for the univariate global model are replaced by the predicted value. This allows tests to be used for MLE even if one of the response features is missing. If all the response features for a particular test are missing or the check box is unselected, the whole test is removed from MLE calculation.

Response Level

> **Note** The response node remains empty until you have used the **Model Selection** window at the local level. Exiting this window then copies the best two-stage model to the response node.

Selecting a response model node (with a two-stage icon — a house *and* a globe) in the model tree produces this view.

Note that local model nodes also have the same icon *after* calculation of the two-stage model (see the model tree for clarification) but selecting them produces the local level view instead.
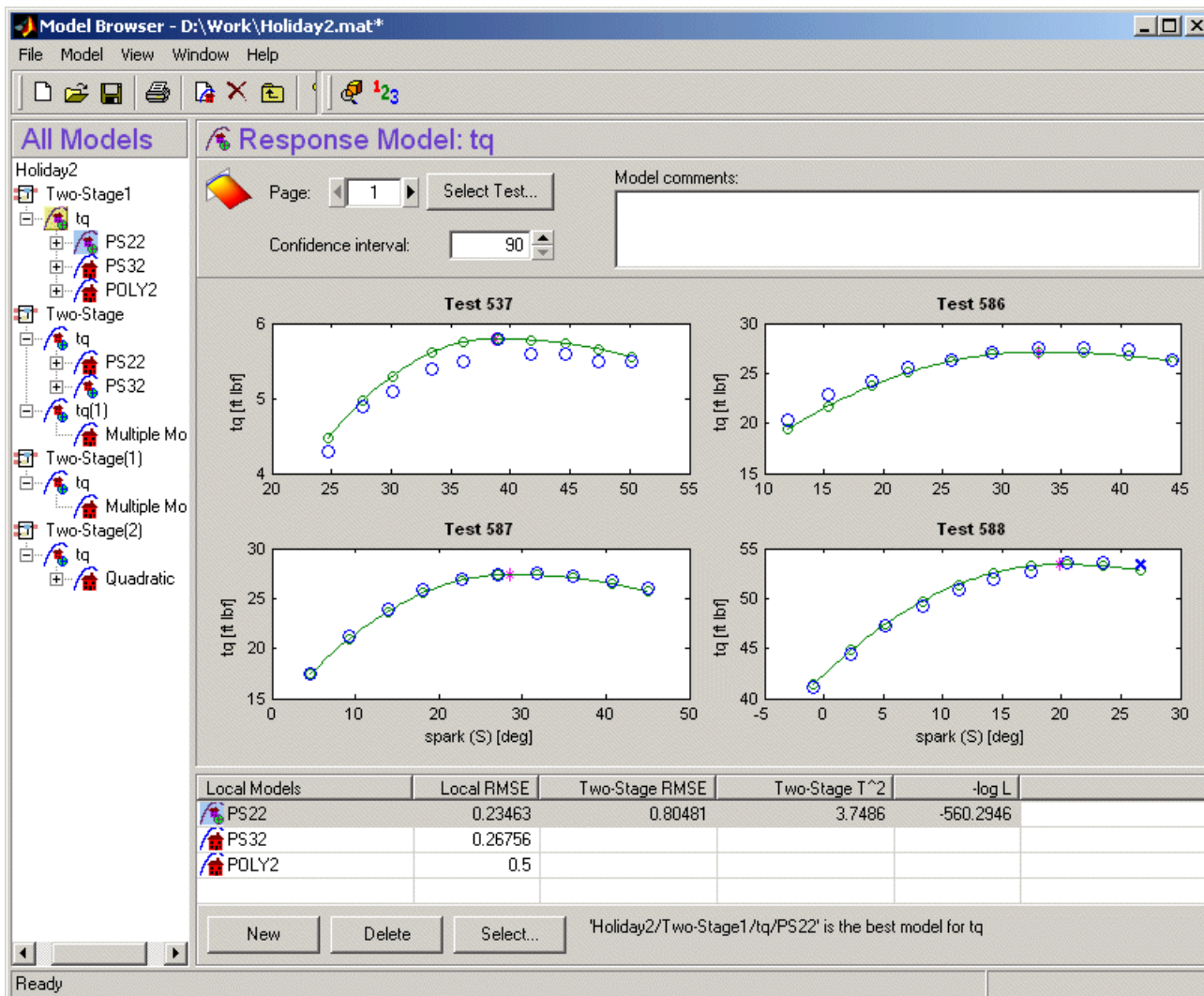
These plots show the data and the two-stage model fit. You can scroll through the tests using the test controls, as at the local level: by clicking the up/down page number buttons, typing directly in the edit box, or clicking **Select Test** to go directly to a particular test.

The Local Models list view shows all child models and summary statistics for comparison of child models. See "Pooled Statistics" on page 8-25 for information on the diagnostic statistics in the list (such as log likelihood and T^2). The statistics in this list can be seen in the Pooled Statistics table at the local model level.

Response Level: Toolbar and Menus
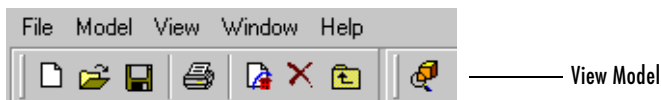
— View Model

- **View Model** is the only toolbar icon after the standard project and node management and print buttons (see "Project Level: Toolbar" on page 7-7 for details on these buttons). **View Model** opens the **Model Viewer** dialog displaying the model terms.

- **File**, **Window**, and **Help** menus remain constant throughout the Model Browser. See "Project Level: Menus" on page 7-8.

- **Model –> Evaluate** opens the **Model Evaluation** window, for examining the fit without data, against current data, or against additional data.

- **Model –> Select** is the same as the **Select** button in the **Local Models** pane and opens the **Model Selection** window. Here you can examine the fit of the two-stage model against the local fit and the data.

- **View –> View Model Definition** duplicates the toolbar button and opens the **Model Viewer** dialog.

Model Evaluation Window

You can access the **Model Evaluation** window via the menu item **Model** -> **Evaluate…** (or the hot key **Ctrl**+E) from any of the modeling nodes: the one-stage or two-stage model node, local model node, response feature nodes, or any child model nodes.

Recall that the **Model Selection** window allows you to compare different models with each other and with the data used to create these models. The **Model Evaluation** window also allows you either to examine a model without data or to validate it against data other than that used in creating the model. For any model node, model evaluation is a quick way to examine the model in more ways than those available in the main Browser views. For example, local models with more than one input factor can only be viewed in the Predicted/Observed view in the main Browser, and the **Model Selection** window only shows you the two-stage model, so you go to **Model Evaluation** to view the local model itself in more detail. For other models such as childless response feature nodes, or their child nodes, **Select** is not available, so **Model Evaluation** is the only way to view these models in detail.

The **Model Evaluation** window comprises some of the same views you see in the **Model Selection** window. The views available depend on what kind of model you are evaluating (two-stage, local, global, or one-stage) and the evaluation mode you choose.

The following example illustrates the evaluation process using other data to validate a model. From a model node, you reach the **Model Evaluation** window via the **Model –> Evaluate…** menu item. The **Data Selection** dialog appears.
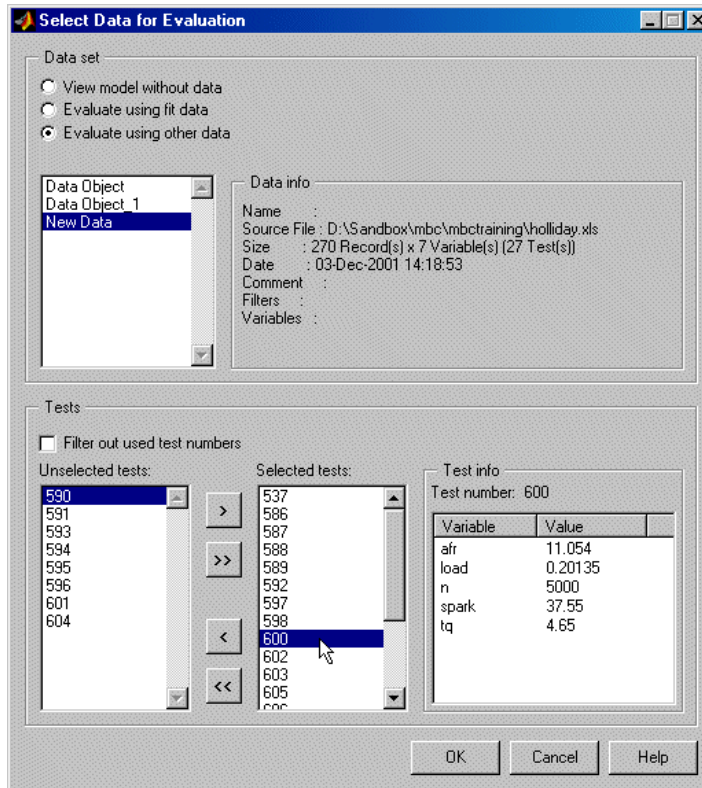
There are three modes for evaluation, each represented by a radio button choice:

- **View model without data** — The evaluation window appears with only the model surface view and the cross-section view. These do not show the model fit against data. Here you can investigate the shape of the model surface.

- **Evaluate using fit data** — The evaluation window appears, and the data shown along with the model surface is the data that was used to create this model. The views available are residuals, response surface, and cross section. If you are evaluating a two-stage model, you can also have the tests view, where the local model fit is shown for each test. If you are evaluating a one-stage model or a two-stage response feature model, the predicted/observed view is also available.

7-145

- **Evaluate using other data** — The data that is shown along with the model surface is chosen in this dialog. The evaluation window then appears with the residuals, response surface, and cross-section view. If you are evaluating a two-stage model, the tests view is also available, although the local fit is not shown, as the local model was fitted to different data.

In the tests view (two-stage models only) you can click and hold on a plot to see the values of the global variables for that test.

Model Evaluation Using Other Data

Prediction performance creates a resource dilemma for the experimenter. You could keep some data back to use later on to get a measure of the predictive capacity of the model, but why withhold data from the model building process if its addition is likely to improve the predictive capability of the model?

There is always a tradeoff in statistics when you decide how much of your data to use for model fitting and whether to leave some data to test your model against.

The Model Evaluation window is intended to help you validate your model against other data, although you can also evaluate the fit against the original data or without any data. The rest of the **Model Evaluation** dialog is only enabled when you choose **Evaluate using other data**, the last of three choices in the **Data Selection** dialog.

When you select the **Evaluate using other data** choice, if the input factors required to evaluate the model do not appear in the selected data set, another dialog appears to match signal names in the selected data set to those required to evaluate the model.
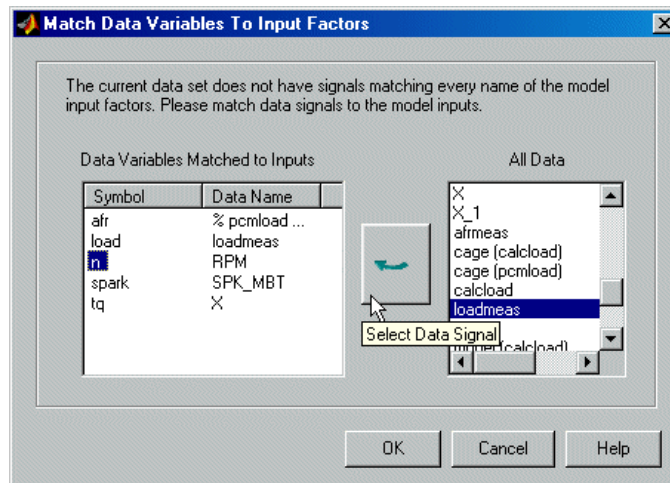
Match signal names from the data set (on the right) to the symbols required to evaluate the model (on the left) and click **OK** to accept this assignment.

In the example shown in "Model Evaluation Window" on page 7-144, the **Data Selection** dialog shows three data sets that you can use to evaluate this model: **Data Object**, **Data Object_1**, and **New Data**. Traceability information about the data set currently selected in the list box is always displayed in the **Data info** pane on the right.

In the lower **Tests** frame you can choose the tests from this data set to use. Certain test numbers might have been used to create the model (as in this case) and you can eliminate these test numbers from the evaluation data set if you want, by selecting the **Filter out used log numbers** check box.

Of the remaining tests, you can select some and reject others using the **Selected Tests/Unselected Tests** list boxes. You move tests between them using the arrow buttons. For the currently selected test, the mean test values of all variables in this data set are displayed on the right.

Click **OK** to use the selected tests to evaluate this model. The **Model Evaluation** window appears.

For information about the available views, see "Selecting Models" on page 7-111.

Exporting Models

You can export all models created in MBC using the menu item **File** ->**Export Models**.

The **Export Model** dialog appears.
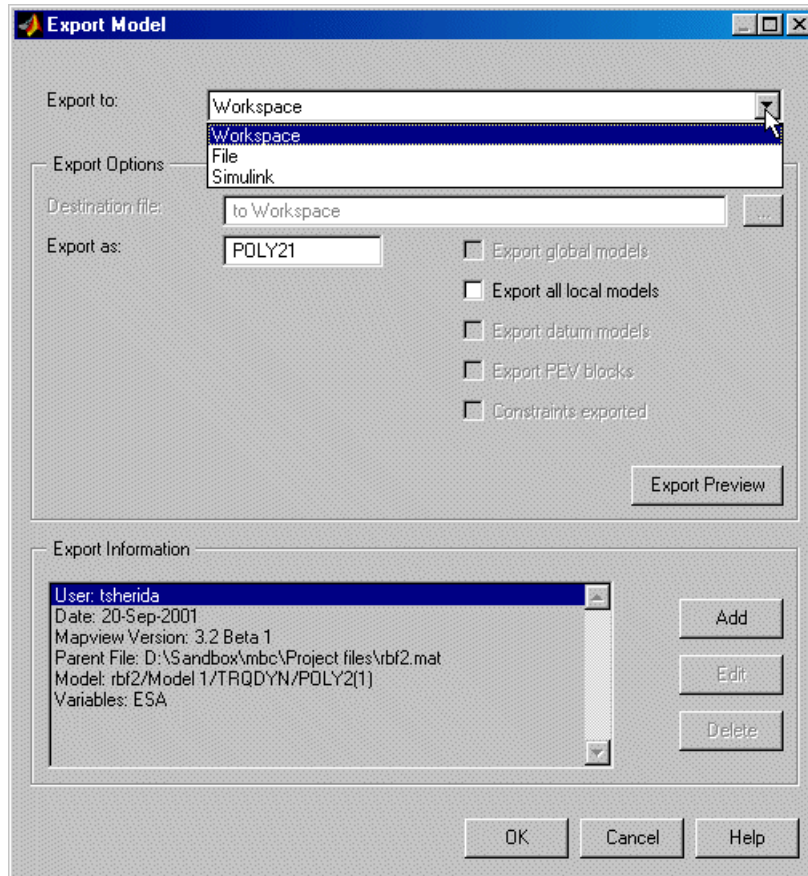
You can export a model to the MATLAB workspace, to an EXM file for CAGE, to a Simulink model (an MDL file), or directly to CAGE (only if CAGE is open). You choose the export format using the **Export to** drop-down menu. If a file format is chosen (export to file or to Simulink), the **Destination file** controls

are enabled, and the "**...**" button allows you to locate a destination file using a file browser.

The EXM file format is specifically designed for loading into the CAGE part of MBC Toolbox, for example, to create calibrations.

Export As — Defines the name that the model has when loaded. For example, if the model is exported to the MATLAB workspace, a workspace variable appears called PS22 (in the preceding example figure).

Export global models — When a two-stage model is being exported (from the response node) the constituent response feature models can also be exported. Multiple models are exported to the workspace as a cell array.

Export all local models — When exporting at the local node, the single local model for the current test is exported. Selecting this control exports the local models for all tests (to the workspace as a cell array).

Export datum models — When exporting a two-stage model that has a datum defined, this control allows you to export the datum global model (without exporting all other response feature models).

Export constraints — Where design constraints and boundary constraints exist, you can choose to export them.

Export PEV blocks — When exporting to Simulink, you can create a PEV block as part of the Simulink diagram so that the prediction error variance can be evaluated along with the model. This is not available for models where PEV cannot be calculated.

Export Preview — Displays the models that are exported with the current choice of options.

For example:
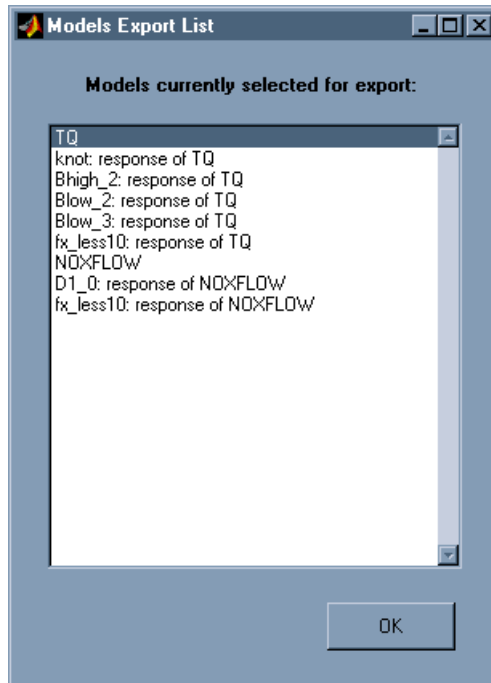
See "What Is Exported?" on page 7-151 for details on what to expect here.

Export information — Traceability information is exported with the models. You can add/edit/delete further comments using the buttons on the right.

Tradeoff... — This button appears when you are exporting from the test plan node. This button exports all local models in a .mat file that can then be loaded into the Multimodel Tradeoff tool within CAGE.

Note The **Tradeoff** button is only enabled when the current test plan has at least one two-stage model available for export, and when this model has exactly two global input factors.

Clicking **Tradeoff...** creates a file browser. When you click **Save**, all local models are saved to the specified file, regardless of the main dialog's **OK** or **Cancel** buttons.

Click **OK** to export the current selection of models and close the dialog.

If you export directly to CAGE (only if CAGE is open) you will see a dialog after you click **OK** to export. In the dialog you can see a list of the models to export. Double click to edit the settings for any model — you can edit the name, replace an existing CAGE model or skip the selected model.

What Is Exported?

Note At any point you can use the **Export Preview** button to check the models that have been selected for export. This displays the **Models Export List** dialog.

At the test plan node:

• You cannot export from the test plan node unless all response models within that test plan have a best model chosen (that is, you used the **Select** process at response level to assign the best model and calculate the two-stage model). All models within the test plan are exported.

• If the datum model check box is selected, the datum model is exported.

• If the checkbox for constraints is selected, boundary constraints are also exported.

• If the check box for global models is selected, all the response features are also exported.

At the response node:

• The response model is exported. You cannot export from a response node until it contains a best model (it is empty before that).

• If the check box for global models is selected, all the response features are also exported. Note that the datum model is not necessarily a response feature.

7-151

- There is also a datum model check box. As at the test plan node, this exports the datum model along with the two-stage model (without exporting all other response feature models).

At the local node:

- The local model *for the current test only* is exported.
- If the check box for all local models is selected, all the local models are exported for all the tests.
- If the node is purely a local model (with a house icon, that is; no two-stage model has yet been calculated), the model is exported under its own name (for example, PS2,3); if a two-stage model has been calculated (that is, the local node has a two-stage house-and-globe icon), the local model is exported under the name of the response node.

At response feature nodes and all child nodes from here:

- The current response feature (or other selected subnode) only is exported.

Note When you are exporting to Simulink, linear models support PEV, so the **Export PEV blocks** check box is active. This is only the case when you are exporting from response features that are linear functions of the local parameters. See Export PEV blocks under "Exporting Models" on page 7-148.

Evaluating Models in the Workspace

If a model is exported to the workspace as MyModel and has four input factors, it can be evaluated at a point as in this example:

```
Y = MyModel([3.7, 89.55, -0.005, 1]);
```

If column vectors p1,p2,p3,p4 (of equal length) are created for each input factor, the model can be evaluated to give a column vector output

```
Y = MyModel([p1,p2,p3,p4]);
```

Left to right argument order corresponds exactly to top-down input order in the Test Plan view in the Model Browser.

If you export a group of models, the group is exported as a cell array of models. The argument order in the curly braces {1 to n} cell array reference corresponds exactly to top-down model order in the model tree in the Model Browser.

The inputs and outputs for MATLAB model evaluation are in natural engineering units, not coded units.

It is also possible to evaluate the PEV (prediction error variance) for the model using the commmand

```
[pev, y] = pev(MyModel, [x1 x2 x3])
```

You can use one or two arguments, as follows:

[p]=pev(x) gives pev at x

[p,y]=pev(x) gives pev at x and model eval at x

Confidence Intervals. The confidence interval is given by:

upperbound = y + t*sqrt(pev)

lowerbound = y - t*sqrt(pev)

Where y is the model prediction, and t is the appropriate percentile of the t-statistic, with df = nObs-1 degrees of freedom. This is calculated using the function tinv from the Statistics Toolbox as follows:

t = tinv(p,v)

p=confidence level e.g. 95%

v= degrees of freedom (n-1)

t = tinv(1-alpha/2, df)

where alpha = 0.05 for 95% confidence intervals.

Boundary Constraint Modeling

From the test plan level, if you select **TestPlan –> Boundary Constraints...** the **Boundary Constraint Editor** appears.

Start Here — Fit Boundary Constraint
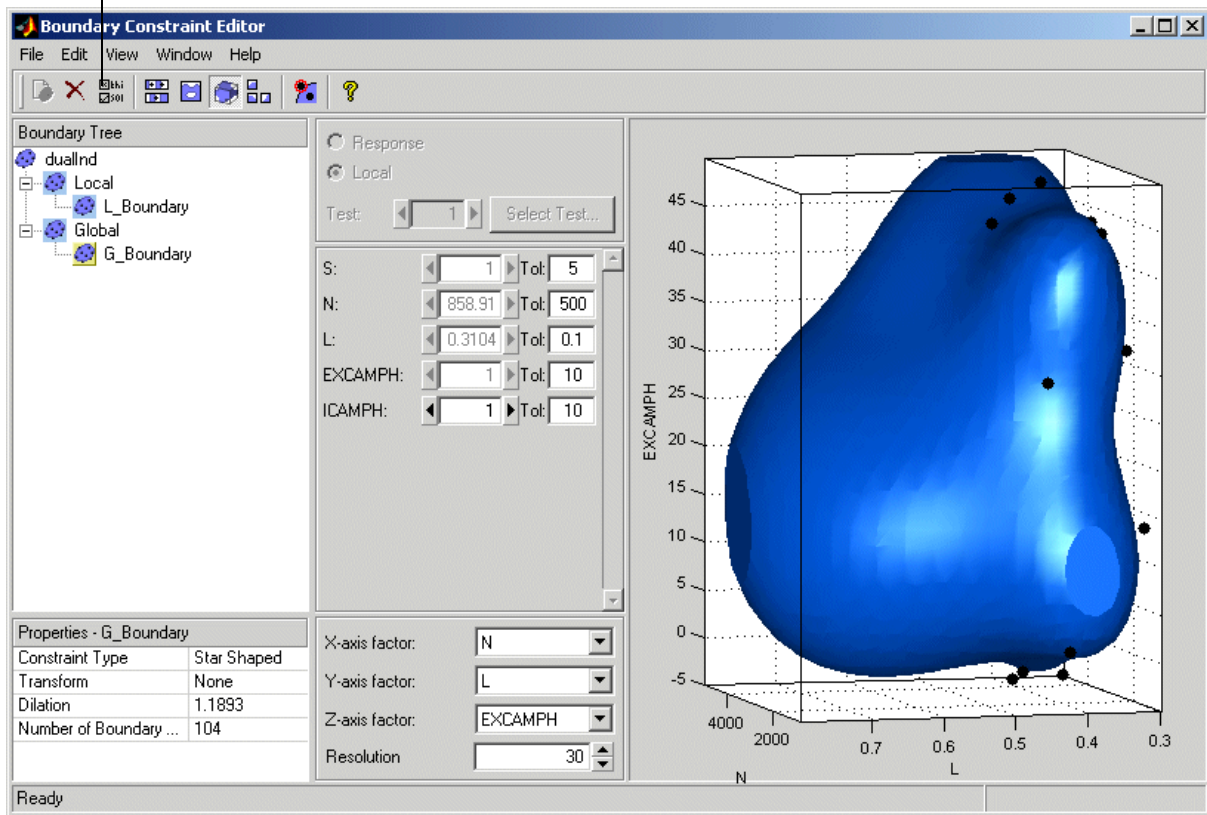
Here you can construct boundary constraint models from your data. You can construct nonparametric surfaces as a visual aid to understanding complex operating envelopes.

These boundary constraints are integrated with the rest of the toolbox functionality. The boundary constraints can be put to use throughout. You can view them in the Model Selection window (in the cross section view and response surface view) and in optimizations and tradeoff in CAGE. This is useful for clipping models to view only the area of interest, constraining modeling to realistic engine operating envelopes, and for designating the most valid areas for optimization, tradeoff and calibration.

In the Model Browser you can export constraints with your models from the test plan level. Use the **File** menu to export either to the Workspace or as an exported model file (`.exm`). See "Exporting Models" on page 7–148. You can use the function `ceval` to evaluate a boundary constraint exported to the Workspace. For example, if your exported model is `M`, then `ceval(M, X)` evaluates the boundary constraint attached to `M` at the points given by the matrix `X` (values less than zero are inside the boundary).

Note Get started by clicking **Fit Boundary Constraint** in the toolbar.

Boundary Constraint Toolbar

Help

Highlight Boundary Points On/Off

Pairwise View

3-D Slice

2-D Slice

1-D Slice

New Boundary Constraint

Delete Boundary Constraint

Fit Boundary Constraint
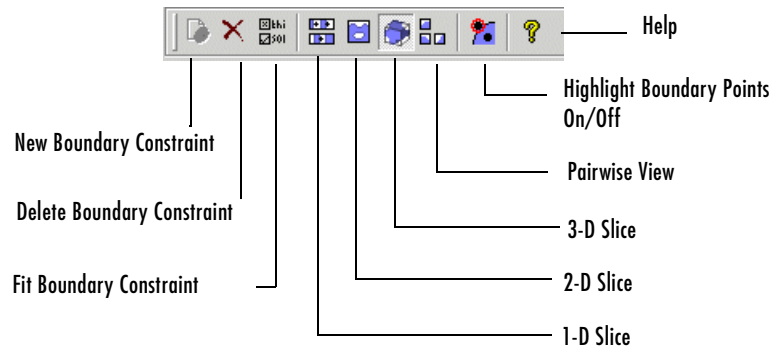
- **New Boundary Constraint**— Adds a new child node to the current node. As for the model tree in the Model Browser, these child nodes are different in different places.

  a   For one-stage test plans, new child nodes of the root (top) node are boundary constraints (leaf nodes).

7-155

    a For two-stage test plans, new child nodes differ depending on the parent node. From the top or root node, a new child node is a response, local or global node (you can only have one of each). From local, global or response nodes, new child nodes are boundary constraints of the same type as their parent node - local, global or response. You can add as many boundary constraints of each type as you want.

    New Boundary Constraint is not available at boundary constraint nodes (these leaf nodes are the lowest level in the tree). Also in the **File** menu.

- **Delete Boundary Constraint**— Deletes the current node. Not available for the root node.

- **Fit Boundary Constraint** — click here to set up a new boundary constraint, or change settings on an existing constraint. The **Boundary Constraint Settings** dialog opens where you can change the constraint type and parameters. Also in the **File** menu.

- **1-D Slice** — shows a 1-D slice through your model and data. Also in the **View** menu. You can select values for variables in the edit boxes. These values determine the point at which the slice through the constraint is plotted. You can change the tolerance values in the **Tol** edit boxes next to each variable. This determines the 'width' of the slice. Data points within the tolerances will be displayed with the slice. This is very similar to the display in the **Cross Section** model selection view, where the Tolerance either side of the displayed model slice determines how near data points must be to the model slice to be displayed. See "Cross Section View" on page 7-133 for comparison.

- **2-D View** — shows a 2-D slice through your model and data. Also in the **View** menu. You can choose which variables to plot on the X and Y drop-down menus. Similarly to the 1-D Slice view, you can change the values and tolerances of the other variables in the edit boxes to determine where the constraint slice is plotted and how much data is also displayed.

- **3-D Slice** — shows a 3-D slice of your model and data. Also in the **View** menu. You can choose which variables to use for the three axes using the drop-down menus, and set the resolution of the grid to display in the edit box. You can set the value of other variables in the edit boxes as for the 2-D view. In the example shown in "Boundary Constraint Modeling" on page 7-154 you can see that the variable A is set by the value in the edit box, while the other three variables are plotted on the X,Y and Z axes.

- **Pairwise View** — shows a pairwise projection of your constraint and data. Also in the **View** menu. There is a plot of the entire range for each pair of

variables in turn. You can click and drag to select a region. Do not click on points, click in the blue or white regions and then drag. The same region is then highlighted in yellow in each projection, so you can see how your data is distributed in each dimension. The resolution you choose affects how closely the selected region fits into the rectangle you have drawn. At low resolutions the grid defined may be well inside the rectangle you have drawn. For many pairs of variables a large resolution can take a long time to draw. Start with a low resolution (less than 10).

- **Highlight Boundary Points**— this highlights in red all data points that are on the boundary surface. This is available for star-shaped constraints and applies to all views. In the 1-D and 2-D view you can also click on any data point to see the values of the variables at that point, and the model evaluation at that point. The model value is a distance to the boundary - if it is negative the point is inside the boundary; if it is positive it is outside. The points highlighted red (on the boundary) should have a model evaluation of zero. Due to rounding errors the value may not be exactly zero, just very small. Also in the **View** menu

Boundary Constraint Menus

- In the **File** menu:

New Boundary Constraint and **Fit Boundary Constraint** — see toolbar.

Close — this closes the **Boundary Constraint Editor**. Your constraint models are saved with the test plan.

- In the **Edit** menu:

Duplicate and **Delete** copy or remove the currently selected node. You cannot duplicate or delete the root node, and you cannot duplicate the Local or Global nodes. Use **Duplicate** to make a selection of leaf nodes in order to make changes and compare many constraint models.

Assign Best — you can choose a best constraint. In the same way as in the model tree in the Model Browser, this best constraint is copied up the tree. For example, if you have a selection of constraints under the root node for a one-stage model and select one as best, this constraint is copied to the root node above. Similarly, for two-stage models, if you assign a best constraint from a variety of leaf nodes under the Global node, that best model is copied up to the Global node. You can see which model is currently best for the selected node by looking under Best Model in the bottom left **Properties** pane.

Add to Best — you can also combine constraints. For example, for a one-stage model, if you pick two leaf nodes and select **Add to Best** for each, the root node will show a combination of the two constraints. For example you may want your final constraint to combine the boundaries of a star shaped and range constraint. You can view the results: the combined model is clipped to fit within the ranges defined by both constraints. You can combine as many of the same type of leaf nodes as you like.

Similarly you can combine best models for both local and global leaf nodes. Note you can only combine local leaf nodes with other local leaf nodes, and global leaf nodes with other global leaf nodes, because you can only have one best model at the Local node, and one best model at the Global node. However you can choose to combine the Local and Global constraints for the root node. You can always see which constraints have been combined at the currently selected node in the **Properties** pane.

Remove From Best — use this to reverse your selections of best and combined best models.

• In the **View** menu:

You can select any of the views available in the toolbar — 1-D, 2-D, 3-D slices, or the pairwise view. You can also toggle boundary point highlighting on and off.

Constraint Information— this opens the **Constraint Information** dialog. This shows you how many points you have, and the number of points on the boundary, interior and exterior of your model.

Constraint Description — this opens the **Constraint Description** dialog. This shows the constraint type and a description of all constraints that you have combined to make the currently selected constraint.

Boundary Constraint Settings Dialog
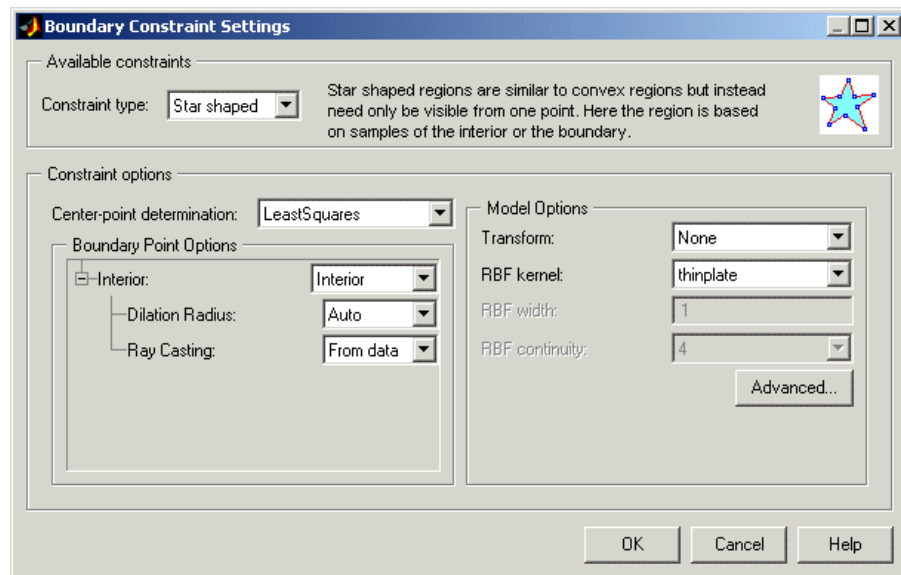
You reach this dialog by clicking **Fit Boundary Constraint** in the **Boundary Constraint Editor** toolbar.

You can choose Star Shaped, Ellipsoid or Range constraints from the **Constraint type** drop-down menu.

- The Range constraint finds the furthest extent of points for each variable and draws a hyper-rectangle to enclose all points.
- The Ellipsoid constraint forms an ellipse to enclose all points.
- The Star Shaped constraint is a more complex constraint with various settings that determine how your boundary model is calculated. This is done in three stages: determining the center of the data; deciding which points are on the boundary, and interpolating between those points. The Star Shaped constraint settings are detailed below.

For all constraint types, click **OK** to fit the boundary constraint.

Star Shaped Constraint Settings

Center Selection — This is not the same as the Center Selection settings for RBFs, this is the method for determining the center of the boundary model sphere. The boundary model can be thought of as a deformed sphere. You can choose **Least Squares** or **MinEllipse**; we recommend Least Squares for most problems as it is faster and there is usually minimal benefit obtained by choosing MinEllipse.

Boundary Point Options

These settings determine how to decide which points are on the boundary.

- **Interior**
  - **Boundary Only** — places all points on the boundary. If this is suitable for your data then this saves time on this stage.
  - **Interior** — choose this if you don't want all your points to be on the boundary.
- **Dilation Radius** —If you choose **Interior** points, then **Dilation Radius** is used to determine which points are on the boundary. Each point is expanded to a sphere until a boundary shape is formed by the intersection of all those spheres. See the illustration below.

Data Points   Each point dilated by a given radius   Dilated data
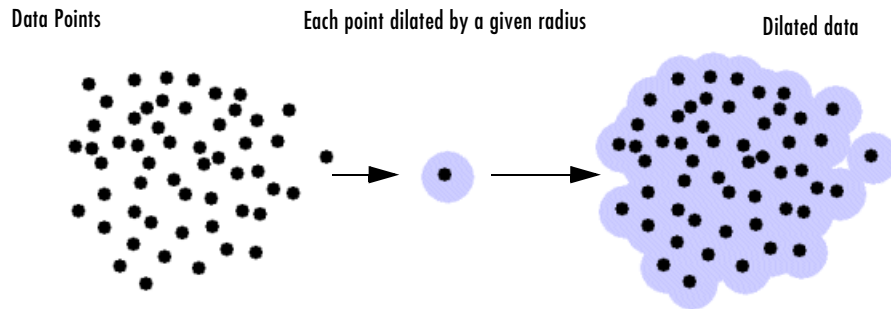
Dilation Radius settings:

- **Auto** — This setting selects the dilation radius (how much to expand each point) by checking all the minimum distances between points, then choosing the largest of those.

- **Manual** — you can manually set the dilation radius in the edit box. The default is 1. This may seem large as model range is between -1 and 1, but all points are expanded equally so you will still detect the points on the edge. However, very large spheres will intersect and obscure points that should be detected as boundary points.

• **Ray Casting** — Rays are drawn from the centre of the boundary model to determine which points are on the edge. The last point intersected on each ray is a boundary point (remember that the ray actually intersects a sphere, given that each point has been expanded by the dilation radius). See the following illustration.

RayCasting

Center of data   Repeat for many rays   Boundary points

Last intersection with sphere,
therefore this point is on the
boundary
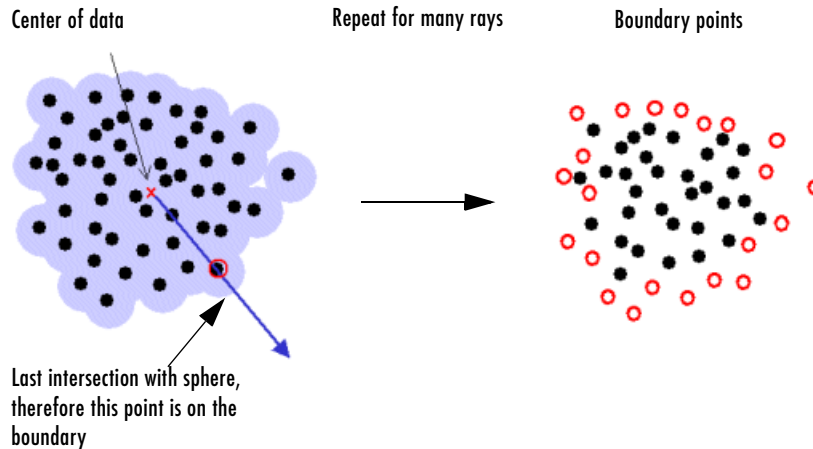
Ray casting settings:

- **From Data** — This uses the same number of rays as there are data points and sends one ray in the direction of each point. If you have very dense data or a very large number of points it might be better to use the Manual setting to choose a smaller number of rays.

- **Manual** — You can set a value in the **Number of Rays** edit box. This number of rays will then be used in random directions. A good guide is about twice the number of data points, though if you have a large number (many hundreds) it will become very slow and you may run out of memory. In most situations more than 1000 is too many.

Model Options

- **Transforms** — **None**, **Log**, or **McCallum**. The default is **None**. Depending on the shape of your boundary, you might need to use a transform to prevent self intersections near the center of the model.

- **Radial Basis Function** (RBF) settings — you can choose RBF kernels as normal. See "Global Model Class: Radial Basis Function" on page 7-49 for more information. Once the boundary points have been determined, each of

those points is used as an RBF center and the boundary surface is obtained by interpolating radial basis functions between all those centers. The width and continuity settings depend on which kernel you choose, and the **Advanced** button gives access to the interpolating RBF model settings.

8

Technical Documents

This section discusses the following topics:

Linear Regression (p. 8-3)

Introducing linear regression in the Model-Based Calibration Toolbox.

Definitions (p. 8-4)

A table defining terms and statistics used in the toolbox and these technical documents.

Prediction Error Variance (p. 8-7)

The meaning of Prediction Error Variance and why it is useful for evaluating the usefulness of models.

High-Level Model Building Process Overview (p. 8-9)

A flowchart and guidelines for the modeling process.

Stepwise Regression Techniques (p. 8-13)

How to use the Stepwise window to improve your models.

Box-Cox Transformation (p. 8-18)

How to use Box-Cox transformation and decide if this improves your models.

Linear Model Statistics Displays (p. 8-22)

A description of the displayed diagnostic statistics, including the meaning of the PRESS statistic and guidelines on what to look for in the diagnostics when comparing models to search for a good fit.

Design Evaluation Tool (p. 8-30)

Descriptions and definitions of all the information available in the Design Evaluation Tool.

Maximum Likelihood Estimation (p. 8-39)

This section contains an overview of the mathematics of Maximum Likelihood Estimation two-stage models and links to model definitions.

Two-Stage Models for Engines (p. 8-40)

An explanation of the reasons why using two-stage models is an effective way to model data containing repeated measures.

Linear Regression

Building a regression model that includes only a subset of the total number of available terms involves a tradeoff between two conflicting objectives:

- Increasing the number of model terms always reduces the Sum Squared Error.
- However, you do not want so many model terms that you overfit by chasing points and trying to fit the model to signal noise. This reduces the predictive value of your model.

The best regression equation is the one that provides a satisfactory tradeoff between these conflicting goals, at least in the mind of the analyst. It is well known that there is no unique definition of *best*. Different model building criteria (for example, forward selection, backward selection, PRESS search, stepwise search, Mallows C_p Statistic…) yield different models. In addition, even if the optimal value of the model building statistic is found, there is no guarantee that the resulting model will be optimal in any other of the accepted senses.

Principally the purpose of building the regression model for calibration is for predicting future observations of the mean value of the response feature. Therefore the aim is to select the subset of regression terms such that PRESS, defined below, is minimized. Minimizing PRESS is consistent with the goal of obtaining a regression model that provides good predictive capability over the experimental factor space. This approach can be applied to both polynomial and spline models. In either case the model building process is identical.

1 The regression matrix can be viewed in the Design Evaluation Tool. Terms in this matrix define the *full model*. In general, the stepwise model is a subset of this full term set.

2 All regressions are carried out with the factors represented on their coded scales (-1,1).

3 All factors and response features are identified by appropriate symbols defined by the test plan. These symbols are automatically carried across to all parts of the model building tool.

Definitions

Symbol	Definition
N	Number of data points
p	Number of terms currently included in the model
q	Total number of possible model parameters (q=p+r)
r	Number of terms not currently included from the model
y	(Nx1) response vector
X	Regression matrix. X has dimensions (Nxq)
X_p	(Nxp) model matrix corresponding to terms currently included in the model
X_r	(Nxr) matrix corresponding to terms currently excluded from the model
β_p	(px1) vector of model coefficients $\beta_p = \{\beta_1, \beta_2, ..., \beta_p\}$ $$\hat{\beta} = (X^T X)^{-1} X^T y$$ $$\text{var}\hat{\beta} = (X^T X)^{-1} \text{MSE}$$
PEV	Prediction Error Variance $$\text{PEV}(x) = \text{var} y(\hat{y}) = x(X^T X)^{-1} x^T \text{MSE}$$
α	User-defined threshold criteria for automatically rejecting terms
$\hat{y}$	(Nx1) vector of predicted responses. $\hat{y} = X_p \beta_p$
e	(Nx1) residual vector. $e = (y - \hat{y})$
$\mathbf{e}_{(i)}$	(Nx1) vector of PRESS residuals. $e_{(i)} = e_i/(1 - H_{ii})$
H	Hat matrix. $X(X'X)^{-1}X$

L	(Nx1) vector of leverage values. $L = \{l_1, l_2, ..., l_N\}' = \{H_{11}, H_{22}, ..., H_{NN}\}$
VIF	Variance Inflation Factors
SSE	Error Sum of Squares. SSE = **e'e**
SSR	Regression Sum of Squares. SSE = $\sum e_i^2$
SST	Total Sum of Squares. SST = y'y - N$\bar{y}^2$
MSE	Mean Square Error. MSE = SSE/(N-p)
MSR	Mean Square of Regression. MSR = SSR/P
F	F-statistic. F = MSR/MSE
MSE$_{(i)}$	MSE calculated with ith point removed from the data set. $$MSE_{(i)} = \frac{(N-p)MSE - e_i/(1-H_{ii})}{N-p-1}$$
RMSE	Root Mean Squared Error: the standard deviation of regression. $RMSE = \sqrt{MSE}$
s$_i$	ith R-Student or Externally Scaled Studentized Residual. $$s_i = \frac{e_i}{\sqrt{MSE_{(i)}(1-H_{ii})}}$$
r$_i$	ith Standardized or Internally Scaled Studentized Residual. $$r_i = \frac{e_i}{\sqrt{MSE(1-H_{ii})}}$$
D	Cook's D Influence Diagnostic. $$D_i = \frac{r_i^2 H_{ii}}{p(1-H_{ii})}$$

SEBETA	(px1) vector of model coefficient standard errors. $$SEBETA = MSE\{\sqrt{c_{11}}, \sqrt{c_{22}}, ..., \sqrt{c_{pp}}\}' \text{ where } c = (\mathbf{X}^\mathrm{T}\mathbf{X})^{-1}$$
PRESS	Predicted Error Sum of Squares. PRESS = $\mathbf{e'}_{(i)}\mathbf{e}_{(i)}$

For more on PRESS and other displayed statistics, see "Linear Model Statistics Displays" on page 8-22.

Prediction Error Variance

Prediction Error Variance (PEV) is a very useful way to investigate the predictive capability of your model. It gives a measure of the precision of a model's predictions.

You can examine PEV for designs and for models. It is useful to remember that:

PEV (model) = PEV (design) * MSE

So the accuracy of your model's predictions is dependent on the design PEV and the mean square errors in the data. You should try to make PEV for your design as low as possible, as it is multiplied by the error on your model to give the overall PEV for your model. A low PEV (close to zero) means that good predictions are obtained at that point.

You can think of the design PEV as multiplying the errors in the data. If the design PEV < 1, then the errors are reduced by the model fitting process. If design PEV >1, then any errors in the data measurements are multiplied. Overall the predictive power of the model will be more accurate if PEV is closer to zero.

You start with the regression (or design) matrix, for example:

$$\mathbf{X} = \begin{bmatrix} 1 & L_1 & N_1 & L^2_1 & L_1N_1 & N^2_1 \\ 1 & L_2 & N_2 & L^2_2 & L_2N_2 & N^2_2 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 1 & L_n & N_n & L^2_n & L_nN_n & N^2_n \end{bmatrix}$$

If you knew the actual model, you would know the actual model coefficients β. In this case the observations would be:

$$y = \mathbf{X}\beta + \varepsilon$$

where ε is the measurement error with variance

$$\mathrm{var}(\varepsilon) = \mathrm{MSE}$$

However you can only ever know the predicted coefficients:

$$\hat{\beta} = (X^T X)^{-1} X^T y$$

which have variance

$$\mathrm{var}\hat{\beta} = (X^{\mathrm{T}}X)^{-1}\mathrm{MSE}$$

Let x be the regression matrix for some new point where you want to evaluate the model, for example:

$$x = \begin{bmatrix} 1 & \mathrm{L}_{new} & \mathrm{N}_{new} & \mathrm{L}^2{}_{new} & \mathrm{L}_{new}\mathrm{N}_{new} & \mathrm{N}^2{}_{new} \end{bmatrix}$$

Then the model prediction for this point is:

$$\hat{y} = x\hat{\beta} = x(X^{\mathrm{T}}X)^{-1}X^{\mathrm{T}}y$$

Now you can calculate PEV as follows:

$$\mathrm{PEV}(x) = \mathrm{var}(\hat{y}) = (x(X^{\mathrm{T}}X)^{-1}X^{\mathrm{T}})(X(X^{\mathrm{T}}X)^{-1}x^{\mathrm{T}})\mathrm{MSE}$$

$$\mathrm{PEV}(x) = x(X^{\mathrm{T}}X)^{-1}x^{\mathrm{T}}\mathrm{MSE}$$

Note the only dependence on the observed values is in the variance (MSE) of the measurement error. You can look at the PEV(x) for a design (without MSE, as you don't yet have any observations) and see what effect it will have on the measurement error - if it is greater than 1 it will magnify the error, and the closer it is to 0 the more it will reduce the error.

You can examine PEV for designs or global models using the Prediction Error Variance viewer. When you open it from the Design Editor, you can see how well the underlying model predicts over the design region. When you open it from a global model, you can view how well the current global model predicts. A low PEV (tending to zero) means that good predictions are obtained at that point. See "Prediction Error Variance Viewer" on page 5-52.

For information on the calculation of PEV for two-stage models, see "Prediction Error Variance for Two-Stage Models" on page 8-46.

High-Level Model Building Process Overview

The recommended overall process is best viewed graphically, as shown in the following flow chart.
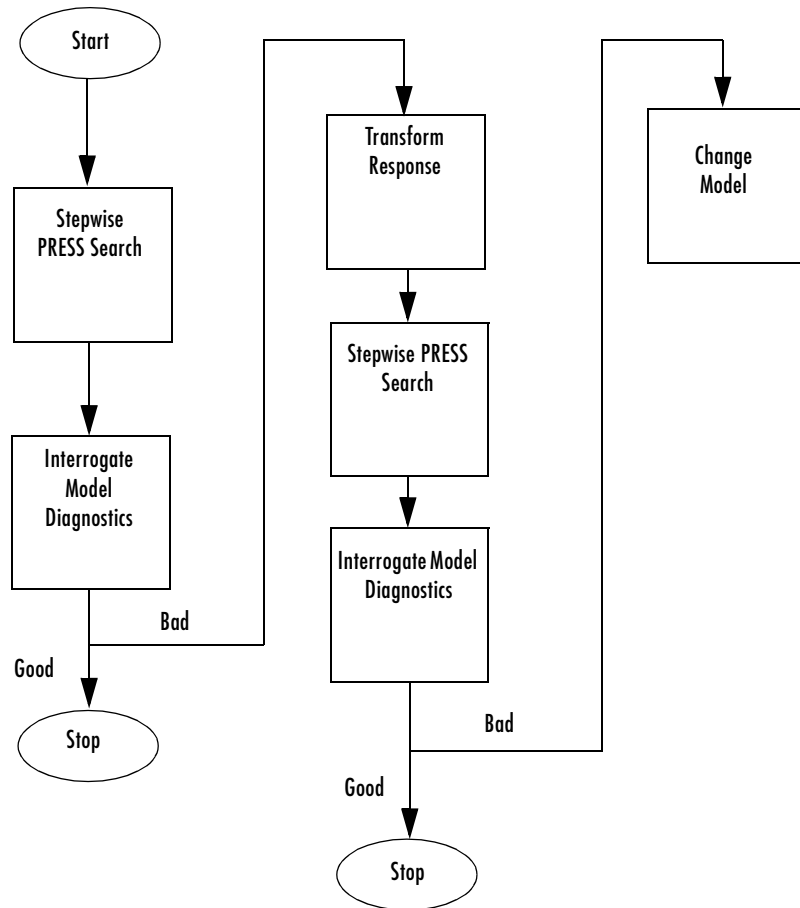
Note that the process depicted in the preceding diagram should be carried out for each member of the set of response features associated with a given response and then repeated for the remaining responses.

Univariate Model Building Process Overview

Once you have set up a model, you should create several alternative models, use the Stepwise functions and examine the diagnostic statistics to search for a good model fit. For each response feature,

1 Begin by conducting a stepwise search.

   You can do this automatically or by using the Stepwise window.

   The goal of the stepwise search is to minimize PRESS. The precise nature of this process is discussed in future sections. What is important to appreciate about the output from this step is that usually not one but several candidate models per response features arise, each with a very similar PRESS R^2. The fact is that the predictive capability of a model with a PRESS R^2 of 0.91 cannot be assumed superior in any meaningful engineering sense to a model with a PRESS R^2 of 0.909. Further, the nature of the model building process is that the "improvement" in PRESS R^2 offered by the last few terms is often very small. Consequently, several candidate models can arise. You can store each of the candidate models and associated diagnostic information separately for subsequent review. Do this by making a selection of child nodes for the response feature.

   However, experience has shown that a model with a PRESS R^2 of less than 0.8, say, is of little use as a predictive tool for engine mapping purposes. This criterion must be viewed with caution. Low PRESS R^2 values can result from a poor choice of the original factors but also from the presence of outlying or influential points in the data set. Rather than relying on PRESS R^2 alone, a safer strategy is to study the model diagnostic information in order to discern the nature of any fundamental issues and then take appropriate corrective action.

2 Once the stepwise process is complete, the diagnostic data should be reviewed for each candidate model.

   It might be that these data alone are sufficient to provide a means of selecting a single model. This would be the case given that one model clearly exhibited more ideal behavior than the others. Remember that the interpretation of diagnostic plots is subjective.

3 You should also remove outlying data at this stage, using the mouse to select the offending point. You can set criteria for detecting outlying data. The default criterion is any case where the absolute value of the external studentized residual is greater than 3.

4 Given that outlying data has been removed, you might want to continue the model building process in an attempt to remove further terms.

This seems reasonable because high-order terms might have been retained in the model in an attempt to follow the outlying data. Even after removing outlying data, there is no guarantee that the diagnostic data will suggest that a suitable candidate model has been found. Under these circumstances,

5 A transform of the response feature might prove beneficial.

A useful set of transformations is provided by the Box and Cox family, which is discussed in the next section. Note that the Box-Cox algorithm is model dependent and as such is always carried out using the (Nxq) regression matrix **X**.

6 After you select a transform, you should repeat the stepwise PRESS search and select a suitable subset of candidate models.

7 After this you should analyze the respective diagnostic data for each model in the usual manner.

At this juncture it might not be apparent why the original stepwise search was carried out in the natural metric. Why not proceed directly to taking a transformation? This seems sensible when it is appreciated that the Box-Cox algorithm often, but not always, suggests that a contractive transform such as the square root or log be applied. There are two main reasons for this:

- The primary reason for selecting response features is that they possess a natural engineering interpretation. It is unlikely that the behavior of a transformed version of a response feature is as intuitively easy to understand.
- Outlying data can strongly influence the type of transformation selected. Applying a transformation to allow the model to fit bad data well does not seem like a prudent strategy. By "bad" data it is assumed that the data is

truly abnormal and a reason has been discovered as to why the data is outlying; for example, "The emission analyzer was purging while the results were taken."

Finally, if you cannot find a suitable candidate model on completion of the stepwise search with the transformed metric, then a serious problem exists either with the data or with the current level of engineering knowledge of the system. Model augmentation or an alternative experimental or modeling strategy should be applied in these circumstances.

After these steps it is most useful to validate your model against other data (if any is available). See "Model Evaluation Window" on page 7-144.

See also these guideline pages with links to information about each of the steps involved in creating one and two-stage models and then searching for the best fit:

- "Instant One-Stage Model Setup" on page 7-33
- "Instant Two-Stage Model Setup" on page 7-34
- "Creating Multiple Models To Compare" on page 7-36

Stepwise Regression Techniques

You can open the stepwise regression window through the ⚙ toolbar icon when you are in the global level view (that is, with a response feature selected in the "Model Tree"). The Stepwise tool provides a number of methods of selecting the model terms that should be included. You can use the **Stepwise** menu in the **Model Setup** dialogs to run stepwise automatically when building linear models.

Minimizing Predicted Error Sum of Squares (PRESS) is a good method for working toward a regression model that provides good predictive capability over the experimental factor space.

The use of PRESS is a key indicator of the predictive quality of a model. The predicted error uses predictions calculated without using the observed value for that observation. PRESS is known as Delete 1 statistics in the Statistics Toolbox. See also "PRESS statistic" on page 8-23.

The next section describes the displays and controls in the Stepwise window.

Stepwise Table

Term	Label for Coefficient
Status	Always. Stepwise does not remove this term. Never. Stepwise does not add this term. Step. Stepwise considers this term for addition or removal.
B	Value of coefficient. When the term is not in the model the value of the coefficient if it is added to the model is displayed in red.
stdB	Standard error of coefficient.

Term	Label for Coefficient
t	*t* value to test whether the coefficient is statistically different from zero. The *t* value is highlighted in blue if it is less than the critical value specified in the α % edit box (at bottom right).
Next PRESS	The value of PRESS if the status of this term is changed at the next iteration. The smallest Next PRESS is highlighted with a yellow background, The column header is also yellow if there is a model change that results in a smaller PRESS value.

The preceding table describes the meanings of the column headings in the **Stepwise Regression** window, shown in the following example.

Stepwise Regression for FX_LESS10

Figure Regression Window Help

Term	Status	B	std B	t	Next PRESS
ϕ_1	Step	51.70	1.9093	27.08	53543
ϕ_2	Step	70.17	3.5734	19.64	28604
ϕ_3	Step	51.63	3.0650	16.85	22320
ϕ_4	Step	56.61	4.5539	12.43	10087
ϕ_5	Step	22.22	10.13	2.1946	910.3
ϕ_6	Step	10.32	17.27	0.5976	1348
$\phi_1{}^*L$	Step	23.96	4.5705	5.2430	2765
$\phi_2{}^*L$	Step	54.35	4.4833	12.12	14673
$\phi_3{}^*L$	Step	31.58	4.6286	6.8224	3605
$\phi_4{}^*L$	Step	45.52	4.2623	10.68	9229
$\phi_5{}^*L$	Step	34.17	6.1988	5.5126	2595
$\phi_6{}^*L$	Step	39.33	3.8513	10.37	8807
$\phi_1{}^*A$	Step	6.8805	3.1501	2.1842	915.0
$\phi_2{}^*A$	Step	-5.8827	3.8804	-1.5160	610.0
$\phi_3{}^*A$	Step	16.15	4.5622	3.5395	1071
$\phi_4{}^*A$	Step	-4.4407	3.2830	-1.3526	582.1
$\phi_5{}^*A$	Step	20.83	5.7326	3.6332	1348
$\phi_6{}^*A$	Step	3.3713	3.5562	0.9480	564.7
$\phi_1{}^*E$	Step	-0.4376	3.8411	-0.1139	610.7
$\phi_2{}^*E$	Step	2.6371	3.0196	0.8733	694.9
$\phi_3{}^*E$	Step	-6.5601	2.7516	-2.3841	767.4
$\phi_4{}^*E$	Step	7.9927	3.6425	2.1943	612.2
$\phi_5{}^*E$	Step	-26.11	11.05	-2.3633	762.6
$\phi_6{}^*E$	Step	-46.02	3.6867	-12.48	17826
L^2	Step	-3.0598	1.0366	-2.9518	958.3
L^*A	Step	-0.4536	0.8736	-0.5193	750.2
L^*E	Step	0.1342	1.6439	0.08166	697.0
A^2	Step	-5.5984	0.8611	-6.5017	3617
A^*E	Step	0.3533	0.9792	0.3608	633.1
E^2	Step	-3.9152	1.4880	-2.6312	1285
L^3	Step	-2.7789	2.2646	-1.2271	645.1
L^2A	Step	-1.4268	1.2494	-1.1420	663.1
L^*A^2	Step	-0.4439	1.5772	-0.2814	686.4
A^3	Step	-7.4471	2.4574	-3.0305	796.9
N^*L^2	Step	12.33	2.1199	5.8186	3810

Coefficients with Errorbars

-100   -50   0   50   100

Stepwise PRESS History

7000
6000
5000
4000
3000
2000
1000

1 2 3 4 5 6 7 8 9 10
Model Number

No. Observations	No. Parameters		Box-Cox	PRESS
48	40	1	11.95	2.98
48	39	1	9.195	2.871
48	38	1	6.516	2.842
48	37	1	5.485	2.71
48	36	1	4.928	2.621
48	35	1	4.332	2.52
48	34	1	3.806	2.428
48	33	1	3.567	2.356
48	32	1	3.394	2.282

Anova table

Source	SS	df	MS
Regression	9.822e+004	32	3069
Error	83.31	16	5.207
Total	9.831e+004	48	

Diagnostic statistics

PRESS   553
PRESS R^2   0.994
R^2   0.999
s   2.28

α(%) 5   t(0.025,16) = 2.120   t(0.025,15) = 2.131

Min. PRESS | Include All | Remove All | Forward | Backwards

1 The confidence intervals for all the coefficients are shown to the right of the table. Note that the interval for the constant term is not displayed, as the value of this coefficient is often significantly larger than other coefficients.

2 Terms that are currently not included in the model are displayed in red.

3 Terms can be included or removed from the model by clicking on the Term, Next PRESS, or coefficient error bar line.

4 A history of the PRESS and summary statistics is shown on the right of the stepwise figure. You can return to a previous model by selecting an item in the list box or a point on the stepwise plot.

5 The ANOVA table and diagnostic statistics for the current model are shown on the right side of the stepwise figure.

6 The critical values for testing whether a coefficient is statistically different from zero at the α% level are displayed at the bottom right side of the stepwise figure. The value of α can be entered in the edit box to the left of the critical values. The default is 5%.

7 A number of further stepwise commands are provided through the buttons at the bottom of the figure (and duplicated in the **Regression** menu):

- **Min. PRESS** includes or remove terms to minimize PRESS. This procedure provides a model with improved predictive capability.
- **Include All** terms in the model (except the terms flagged with **Status** as **Never**). This option is useful in conjunction with **Min. PRESS** and backward selection. For example, first click **Include All**, then **Min. PRESS**. Then you can click **Include All** again, then **Backwards**, to compare which gives the best result.
- **Remove All** terms in the model (except the terms flagged with **Status** as **Always**). This option is useful in conjunction with forward selection (click **Remove All**, then **Forwards**).
- **Forwards** selection adds all terms to the model that would result in statistically significant terms at the α% level. The addition of terms is repeated until all the terms in the model are statistically significant.

- **Backwards** selection removes all terms from the model that are not statistically significant at the α % level. The removal of terms is repeated until all the terms in the model are statistically significant.

Any changes made in the stepwise figure automatically update the diagnostic plots in the Model Browser.

You can revert to the starting model when closing the Stepwise window. When you exit the Stepwise window, the **Confirm Stepwise Exit** dialog asks `Do you want to update regression results?` You can click **Yes** (the default), **No** (to revert to the starting model), or **Cancel** (to return to the Stepwise window).

You can set the **Minimize PRESS**, **Forward**, and **Backward** selection routines to run automatically without the need to enter the stepwise figure. These options are selected through the **Global Model Setup** dialog.

From the global level, select **Model –> Set Up**. The **Global Model Setup** dialog has a drop-down menu **Stepwise**, with the options **None**, **Minimize PRESS**, **Forward selection**, and **Backward selection**. You can set these options when you initially set up your test plan.

Box-Cox Transformation

You might want to transform a response feature either to correct for nonnormality and/or a heteroscedastic variance structure. A useful class of transformations for this purpose is the power transform y^λ, where λ is a parameter to be determined. Box and Cox (1964) showed how λ and the regression coefficients themselves could be estimated simultaneously using the method of maximum likelihood. The procedure consists of conducting a standard least squares fit using

$$y^{(\lambda)} = \frac{y^\lambda - 1}{\lambda y^{\lambda-1}} \text{ for } \lambda \neq 0 \, \lambda$$

$$y^{(\lambda)} = y \ln(y) \text{ for } \lambda = 0$$

where the so called geometric mean of the observations is given by

$$y = \exp\left[\frac{\displaystyle\sum_{i=1}^{N} \ln(y_i)}{N}\right]$$

The maximum likelihood estimate of λ corresponds to the value for which the SSE(λ) from the fitted model is a minimum. This value of λ is determined by fitting a model (assumed throughout to be defined by the regression matrix for the full model - X) for various levels of λ and choosing the value corresponding to the minimum SSE(λ). A plot of SSE(λ) versus λ is often used to facilitate this choice.

The parameter λ is swept between the range of -3 to 3 in increments of 0.5.

• You can enter a value for lambda in the edit box that approaches the point on the plot with the smallest SSE.

Although SSE(λ) is a continuous function of λ, simple choices for λ are recommended. This is because the practical difference between 0.5 and 0.593, say, is likely to be very small but a simple transform like 0.5 is much easier to interpret.

You can also find an approximate 100(1-α) confidence interval on l by computing

$$SS^* = SSE(\lambda)\left[1 + \frac{t_{\alpha/(2,\upsilon)}}{\upsilon}\right]$$

where υ is the number of residual degrees of freedom equal to (N-q).

In this formula λ is understood to be the value that minimizes SSE(λ). Note that this confidence interval might encompass more than one incremental value for λ. In this case, any of these values is as valid as any other and you can select any of these transformations from which to develop trial models.

• You should always look at the residuals plots at the top to see the effect of different transforms.

• You can create several child nodes of a single model and choose different transforms for each in order to compare them using the rest of the Model Browser tools.

For the sake of clarity, consider the example following, which illustrates the outcome of applying the Box-Cox method.
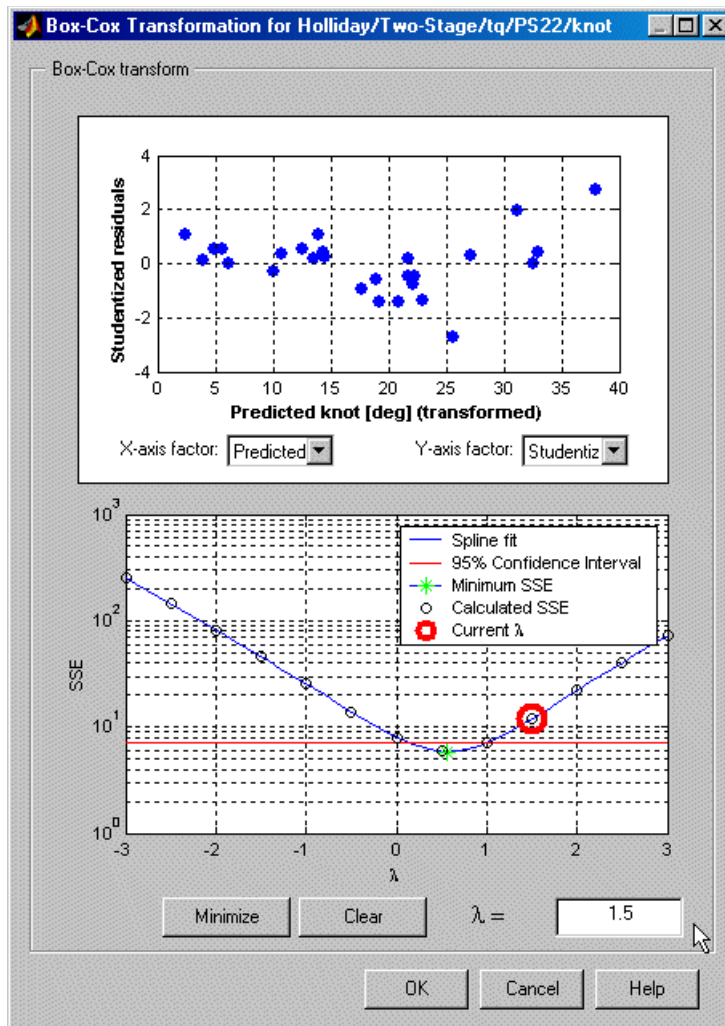
The preceding example shows the results of applying the Box-Cox algorithm to a polyspline torque model.

In this example the minimum value of SSE(λ) occurs near to λ=0. The minimum is marked in green. The 95% confidence limit has been calculated and drawn on the figure as a red solid line. It is apparent in this example that, after rounding to the nearest incremental value contained within the

confidence interval, any λ in the range $0 \le \lambda \le 1$ is appropriate. Of the three possible increments, 0, 0.5, and 1, $\lambda = 0.5$ is the closest to the minimum SSE.

You can select any point on the plot by clicking. The chosen point (current lambda) is then outlined in red. You can also enter values of lambda directly in the edit box and press **Return**.

Linear Model Statistics Displays

Summary Statistics

These are seen for global models.

Observations	Number of observations used to estimate model
Parameters	Number of parameters in model
Box-Cox	Power transform used for box-cox transformation. A value of zero means a log transform is used. A value of 1 means there is no transformation.
PRESS RMSE	Root mean squared error of predicted errors. The divisor used for PRESS RMSE is the number of observations. The residuals are in untransformed values to enable comparison between alternative models with different Box-Cox transformations.
RMSE	Root mean squared error. The divisor used for RMSE is the number of observations minus the number of parameters. The residuals are in untransformed values, to enable comparison between alternative models with different Box-Cox transformations.

Note that additional statistics can be shown here. You can choose from a selection in the **Summary Statistics** dialog by selecting **Model –> Summary Statistics**, or they can be set at the test plan. These settings are inherited by child node models. Names and formulae are contained in the dialog. See "Summary Statistics" on page 7-24.

ANOVA Table

	SS	df	MS
Regression	SSR	p-1	SSR/(p-1)
Error	SSE	n-p	SSE/(N-p)
Total	SST	n-1	

PRESS statistic

With n runs in the data set, the model equation is fitted to n-1 runs and a prediction taken from this model for the remaining one. The difference between the recorded data value and the value given by the model (at the value of the omitted run) is called a prediction residual. PRESS is the sum of squares of the prediction residuals. The square root of PRESS/n is PRESS RMSE (root mean square prediction error).

Note that the prediction residual is different from the ordinary residual, which is the difference between the recorded value and the value of the model when fitted to the whole data set.

The PRESS statistic gives a good indication of the predictive power of your model, which is why minimizing PRESS is desirable. It is useful to compare PRESS RMSE with RMSE as this may indicate problems with overfitting. RMSE is minimized when the model gets very 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.

For more information see "Stepwise Regression Techniques" on page 8-13 and "Definitions" on page 8-4.

Note that calculating PRESS for the two-stage model applies the same principle (fitting the model to n-1 runs and taking a prediction from this model for the remaining one) but in this case the predicted values are first found for response features instead of data points. The predicted value, omitting each test in turn, for each response feature is estimated. The predicted response features are then used to reconstruct the local curve for the test and this curve is used to obtain the two-stage predictions. This is applied as follows:

To calculate two stage PRESS:

1 For each test, S, do the following steps:

- For each of the response features, calculate what the response feature predictions would be for S (with the response features for S removed from the calculation).
- This gives a local prediction curve C based on all tests except S.
- For each data point in the test, calculate the difference between the observed value and the value predicted by C.

2 Repeat for all tests.

3 Sum the square of all of the differences found and divide by the total number of data points.

Pooled Statistics

These are seen at the local node (when two-stage modeling) in the Pooled Statistics table, and at the response node in the list of local models. If you have a selection of local models, use these statistics to help you choose which is best.

Local RMSE	Root mean squared error, using the local model fit to the data for the displayed test. The divisor used for RMSE is the number of observations minus the number of parameters.
Two-Stage RMSE	Root mean squared error, using the two-stage model fit to the data for the displayed test. You want this error to be small for a good model fit.
PRESS RMSE	Root mean squared error of predicted errors, useful for indicating overfitting; see "PRESS statistic" on page 8-23 above. The divisor used for PRESS RMSE is the number of observations. Not displayed for MLE models because the simple univariate formula cannot be used.
Two-Stage T^2	T^2 is a normalized sum of squared errors for all the response features models. You can see the basic formula on the Likelihood view of the Model Selection window. $$T^2 = (y_{rf} - \hat{y})^T \Sigma^{-1} (y_{rf} - \hat{y})$$ Where $\Sigma = \text{blockdiag}(C_i + D)$, where C_i is the local covariance for test i. See *blockdiag* diagram following. A large T^2 value indicates that there is a problem with the response feature models.

-log L	Log-likelihood function: the probability of a set of observations given the value of some parameters. You want the likelihood to be large, tending towards -infinity, so large negative is good.
	For n observations $x_1, x_2, \ldots x_n$, with probability distribution $f(x, \theta)$, the likelihood is:
	$$L = \prod_{i=1}^{n} f(x_i, \theta)$$
	This is the basis of "Maximum Likelihood Estimation" on page 8-39.
	$$\log L = \log(\det(\Sigma)) + (y_{rf} - \hat{y})^T \Sigma^{-1} (y_{rf} - \hat{y})$$
	which is the same as:
	$$\log L = \log(\det(\Sigma)) + T^2$$
	This assumes a normal distribution.
	You can view plots of -log L in the Model Selection window, see "Likelihood View" on page 7-128.

To explain blockdiag as it appears under T^2 in the Pooled statistics table: $\Sigma = \text{blockdiag}(C_i + D)$, where C_i is the local covariance for test i, is calculated as shown below.

$$blockdiag(C_i + D) = \begin{bmatrix} C_1 + D & & & \\ & C_2 + D & & \\ & & C_3 + D & \\ & & & C_i + D \end{bmatrix}$$

Information Criteria

There are information criteria available as additional summary statistics for comparison of models. See "Summary Statistics" on page 7-24 for information on how to display these. This section provides some statistical background to help you compare the Akaike Information Criteria (AIC and AICc) and the Bayes Information Criterion (BIC). See also "Model Selection Guide" on page 7-113 for practical guidelines on using AIC and BIC.

AIC-type criteria are based on the difference in Kullback-Leibler information between two models, or their K-L distance. K-L distance is an appealing measure because it essentially compares the information content of two curves, by calculating the entropy in each. Akaike and others found ways to estimate K-L distance based on the results of a maximum likelihood estimate of the parameters of a model, given some data. These estimates are the information criteria, and become more accurate as the sample size increases.

BIC is derived from Bayes' theorem, and essentially just applies the Occam effect to select a preferred model; the idea that if two models provide an equally good fit with some data then the simpler model is the likelier. This can be understood in the following sense: for models with greater complexity (both in terms of the number of parameters and the set of values those parameters can take) it is less remarkable that they are able to fit a given data set well. Conversely, for a simple model, if you happen to encounter a data set for which the model provides an acceptable fit, it would seem a lucky coincidence. Therefore, for data matching both models well, the odds are that the simpler one is closer to the truth.

Quantifying these ideas leads to Bayes factors (evidence ratios) which measure the relative probabilities of two models. In the context of MBC, BIC is an estimate of Bayes factors based on the results of a maximum likelihood estimate, and, like AIC, increases in accuracy in the limit of large sample size. Although priors often spring to mind in the context of Bayes theorem, all of the above can proceed with uniform priors on everything, and the Occam effect still applies.

There is a degree of controversy over which approach gives the best results. Copious literature exists on the subject of Bayesian model selection, a smaller amount on K-L distance based techniques and a still smaller amount comparing the two approaches. Bayesian authors consistently find that BIC performs better in Monte Carlo simulations (e.g. Leonard and Hsu 1999)

whereas Burnham and Anderson 2002 (the main proponents of K-L distance techniques) reach conclusions which favour AIC.

Such tests can be set up to favour either criterion, and there are two main effects relevant to understanding this. Differences arise due to the assumptions made about the truth in each case (relevant to choosing Bayes factors or K-L distance), and due to the number of samples relative to the number of parameters in the candidate models.

Regarding the former effect: Bayes factors always seek the simplest model consistent with the data. K-L distance also has this tendency, although not as strongly as the Bayesian approach. As a result, the simulations in which BIC does well tend to be based on simple models with few parameters (Leonard and Hsu choose a simple quadratic, then consider polynomials of order 1 to 7 in their candidate set). Although both approaches choose the correct model more often than any other, AIC gives slightly more weight to the higher order models than does BIC. For this reason, Bayesians often accuse AIC of overfitting.

Burnham and Anderson, however, are biologists and as such they abandon all hope of actually finding the true model in their candidate set - they simply attempt to find the best approximation to the truth. A typical simulation of theirs considers linear models with up to 13 possible variables in the context of predicting body fat. They are not concerned with the subtle shape of curves, only with which variables they can safely throw away. In this scenario, they find that BIC favours too simple a model and hence underfits the data.

Although for BIC most authors assume that the true model is contained within the candidate set, this is not necessary for model comparison -- it just concerns the normalisation of the probabilities, and hence not the ratios that form the Bayes factors.

AIC and BIC both improve as estimators of their respective statistical measures as the sample size increases, with relative errors of O(n-1), where n is the sample size. AIC is obtained from a first order Taylor expansion, and AICc is a second order correction to that for the special case of Gaussian Likelihood (there is no general second order correction) and should be used when the ratio of data samples to model parameters (in the largest model for nested sets) is less than about 40:1. For very small sample sizes, even Bayesian authors do not seem to trust BIC, but do consider AICc.

In terms of the complexity of the truth, most problems in MBC probably lie in between the two extremes described above: internal combustion engines are

not so simple that we assume that our model set really contains the precise, closed-form solutions to the relevant dynamical equations, but we are dealing with a mechanical system, not trying to predict, for example, characteristics of the human body. In terms of the number of samples per model parameter, AIC is seldom likely to be a reliable statistic; AICc should be used instead. But if you have reason to prefer a more conservative estimate of the complexity of the model, BIC should be considered.

For a discussion of Bayes factors, see:

Kass and Raftery (1995). Bayes factors. Journal of the American Statistical Association 90, 773-795

See also Chapter 28 from the following book: Information Theory, Inference, and Learning Algorithms, available from
`http://www.inference.phy.cam.ac.uk/mackay/itila/book.html`
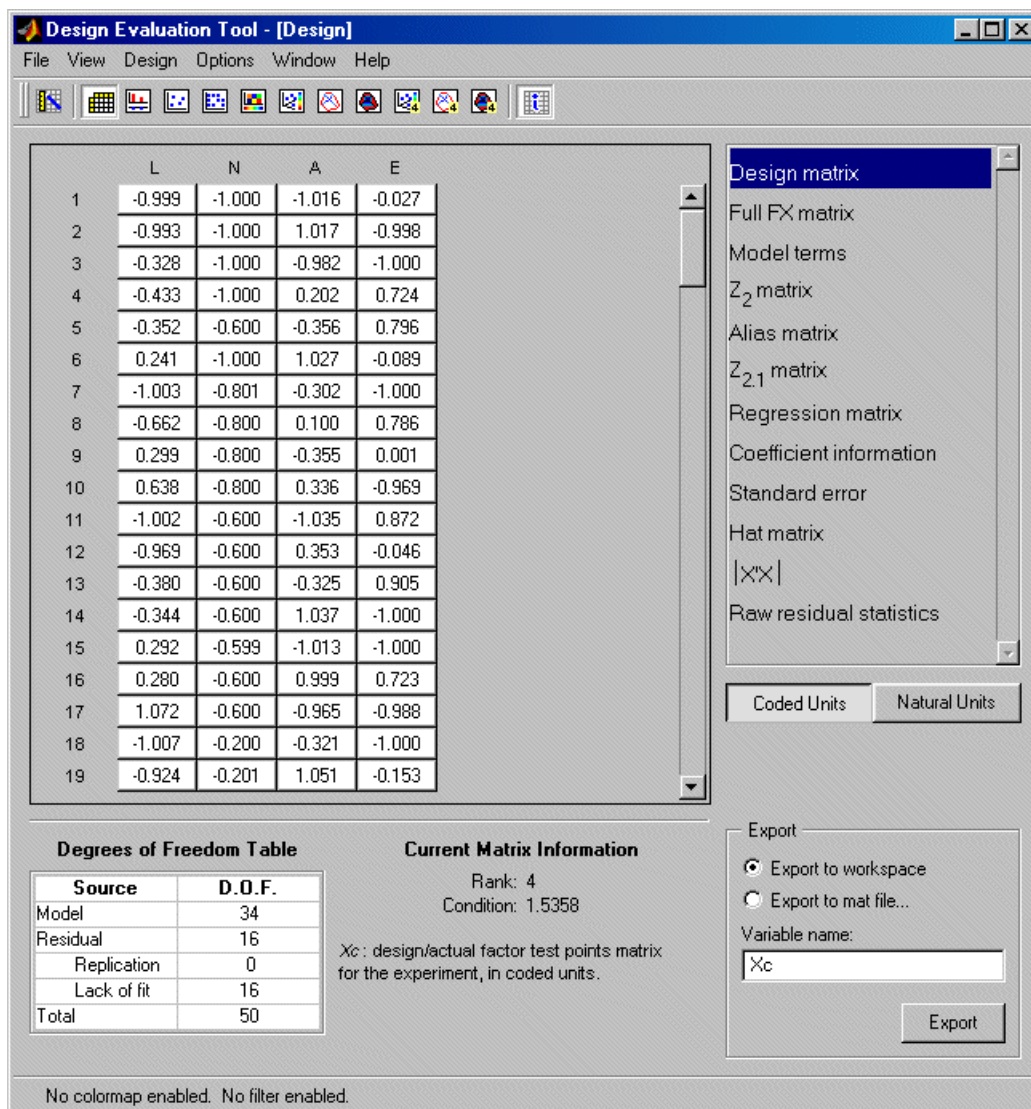
Bayesian Methods, Leonard and Hsu, Cambridge University Press 1999

Model Selection and Multimodel Inference, Burnham and Anderson, Second Edition, Springer-Verlag 2002

Design Evaluation Tool

The Design Evaluation tool is only available for linear models.

You can open the Design Evaluation tool from the Design Editor or from the Model Browser windows. From the Design Editor select **Tools –> Evaluate Designs** and choose the design you want to evaluate. From the Model Browser global view, you can click the [icon] button.

In the Design Evaluation tool you can view all the information on correlations, covariance, confounding, and variance inflation factors (VIFs). You can investigate the effects of including or excluding model terms aided by this information (you must remove them in the Stepwise window). Interpretation is aided by color-coded values based on the magnitude of the numbers. You can specify changes to these criteria.

When you open the Design Evaluation tool, the default view is a table, as shown in the preceding example. You choose the elements to display from the list on the right. Click any of the items in the list described below to change the display. Some of the items have a choice of buttons that appear underneath the list box.

To see information about each display, click the [icon] toolbar button or select **View –> Matrix Information**.

Table Options

You can apply color maps and filters to any items in a table view, and change the precision of the display.

To apply a color map or edit an existing one:

1 Select **Options –> Table –> Colors**. The **Table Colors** dialog appears.

2 Select the check box **Use a colormap for rendering matrix values**.

3 Click the **Define colormap** button. The **Colormap** dialog appears, where you can choose how many levels to color map, and the colors and values to use to define the levels. Some tables have default color maps to aid analysis of the information, described below.

You can also use the **Options –> Table** menu to change the precision (number of significant figures displayed) and to apply filters that remove specific values or values above or below a specific value from the display.

The status bar at bottom left displays whether color maps and filters are active in the current view.

When evaluating several designs, you can switch between them with the **Next design** toolbar button or the **Design** menu.

Design Matrix

Xn/Xc: design/actual factor test points matrix for the experiment, in natural or coded units. You can toggle between natural and coded units with the buttons on the right.

Full FX Matrix

Full model matrix, showing all possible terms in the model. You can include and exclude terms from the model here, by clicking on the green column headings. When you click one to remove a term, the column heading becomes red and the whole column is grayed.

Model Terms

You can select terms for inclusion in or exclusion from the model here by clicking. You can toggle the button for each term by clicking. This changes the button from in (green) to out (red) and vice versa. You can then view the effect of these changes in the other displays.

Note Removal of model terms only affects displays within the Design Evaluation tool. If you decide the proposed changes would be beneficial to your model, you must return to the Stepwise window and make the changes there to fit the new model.

Z_2 Matrix

Z_2 : Matrix of terms that have been removed from the model. If you haven't removed any terms, the main display is blank apart from the message "All terms are currently included in the model."

Alias Matrix

Like the Z_2 matrix, the alias matrix also displays terms that are not included in the model (and is therefore not available if all terms are included in the

model). The purpose of the alias matrix is to show the pattern of confounding in the design.

A zero in a box indicates that the row term (currently included in the model) is not confounded with the column term (currently not in the model). A complete row of zeros indicates that the term in the model is not confounded with any of the terms excluded from the model. A column of zeros also indicates that the column term (currently not in the model) could be included (but at the cost of a reduction in the residual degrees of freedom).

A: the alias matrix is defined by the expression

$$A = (X'X)^{-1}X'Z_2$$

$Z_{2.1}$ Matrix

As this matrix also uses the terms not included in the model, it is not available if all terms are included.

$Z_{2.1}$: Matrix defined by the expression $Z_{2.1} = Z_2 - XA$

Regression Matrix

Regression matrix. Consists of terms included in the model. $n \times p$ matrix where n is the number of test points in the design and p is the number of terms in the model.

Coefficient Information

When you select **Coefficient information**, six buttons appear below the list box. Covariance is displayed by default; click the buttons to select any of the others for display.

Covariance

Cov(b): variance-covariance matrix for the regression coefficient vector *b*.

$$Cov(b) = (X'X)^{-1}$$

Correlation

Corr(b): correlation matrix for the regression coefficient vector b.

$$Corr(b)_{ij} = \frac{Cov(b)_{ij}}{\sqrt{(Cov(b)_{ii})}\sqrt{(Cov-(b)_{jj})}}$$

By default **Correlation** has an active color map to aid analysis. Values below `-0.9` are red, `-0.9` to `-0.7` are orange, `-0.7` to `0.707` are black, `0.707` to `0.9` are orange, and greater than `0.9` are red. You can view and edit the color map using **Options** –> **Table** –> **Colors**.

Partial VIFs

Variance Inflation Factors (VIFs) are a measure of the nonorthogonality of the design with respect to the selected model. A fully orthogonal design has all VIFs equal to unity.

The Partial VIFs are calculated from the off-diagonal elements of Corr(b) as

$$VIF_{ij} = \frac{1}{(1 - Corr(b)_{ij}^2)} \text{ for } p \geq i > j > 1$$

Partial VIFs also has a default color map active (`<1.2` black, `>1.2<1.4` orange, `>1.4` red). A filter is also applied, removing all values within `0.1` of `1`. In regular designs such as Box-Behnken, many of the elements are exactly 1 and so need not be displayed; this plus the color coding makes it easier for you to see the important VIF values. You can always edit or remove color maps and filters.

Multiple VIFs

Measure of the nonorthogonality of the design. The Multiple VIFs are defined as the diagonal elements of Corr(b):

$$VIF_i = \{Corr(b)^{-1}\}_{ii}$$

Multiple VIFs also has a default color map active (`<8` black, `8><10` orange, `>10` red). A filter is also applied, removing all values within `0.1` of `1`. Once again this makes it easier to see values of interest.

2 Column Corr.

Corr(X); correlation for two columns of X.

$$w_{ij} = \frac{x_{ij} - \bar{x}_j}{\sqrt{\sum_{i=1}^{N} (x_{ij} - \bar{x})^2}}$$

Let W denote the matrix of w_{ij} values. Then the correlation matrix for the columns of X (excluding column 1) is Corr(X), defined as

```
Corr(X) = W W
```

2 Column Correlation has the same default color map active as **Correlation**.

Single Term VIFs

Measure of the nonorthogonality of the design. The Single Term VIFs are defined as

$$VIF_{ij} = \frac{1}{(1 - Corr(X)_{ij}^2)} \text{ for } p \geq i > j > 1$$

Single term VIFs have a default color map active (<2 black, 2>red) and values within 0.1 of 1 are filtered out, to highlight values of interest.

Standard Error

σ_j : Standard error of the jth coefficient relative to the RMSE.

Hat Matrix

Full Hat matrix

H: The Hat matrix.

```
H = QQ
```

where Q results from a QR decomposition of X. Q is an $n \times n$ orthonormal matrix and R is an $n \times p$ matrix.

Leverage values

The leverage values are the terms on the leading diagonal of H (the Hat matrix). Leverage values have a color map active (<0.8 black, 0.8>orange<0.9, >0.9 red).

|X'X|

D; determinant of X'X.

D can be calculated from the QR decomposition of X as follows:

$$D = \left[\prod_{i=1}^{p} (R_1)_{ii} \right]^2,$$

where p is the number of terms in the currently selected model.

This can be displayed in three forms:

$|X'X|$

$\log(|X'X|)$

$|X'X|^{(1/p)},$

Raw Residual Statistic

Covariance

Cov(e): Variance-covariance matrix for the residuals.

```
Cov(e) = (I-H)
```

Correlation

Corr(e) : Correlation matrix for the residuals.

$$Corr(e)_{ij} = \frac{Cov(e)_{ij}}{\sqrt{(Cov(e)_{ii})}\sqrt{(Cov(e)_{jj})}}$$

Degrees of Freedom Table

To see the **Degrees of Freedom** table (and the information about each display), click the ![icon] toolbar button or select **View –> Matrix Information**.

Degrees of Freedom Table	
Source	**D.O.F.**
Model	5
Residual	14
Replication	0
Lack of fit	14
Total	19

Source	D.F.
Model	p
Residual	n-p
Replication	by calculation
Lack of fit	by calculation
Total	n

Replication is defined as follows:

Let there be n_j (>1) replications at the jth replicated point. Then the degrees of freedom for replication are

$$\sum_j (n_j - 1)$$

and **Lack of fit** is given by n - p - degrees of freedom for replication.

Note: replication exists where two rows of X are identical. In regular designs the factor levels are clearly spaced and the concept of replication is unambiguous. However, in some situations spacing can be less clear, so a tolerance is imposed of 0.005 (coded units) in all factors. Points must fall within this tolerance to be considered replicated.

Design Evaluation Graphical Displays
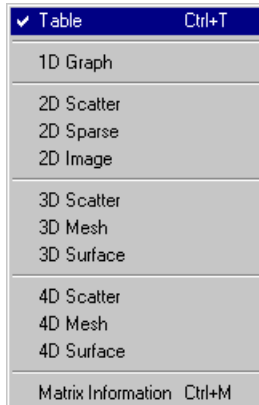
The Design Evaluation tool has options for 1-D, 2-D, 3-D, and 4-D displays. You can switch to these by clicking the toolbar buttons or using the **View** menu.

Which displays are available depends on the information category selected in the list box. For the Design matrix, (with sufficient inputs) all options are available. For the Model terms, there are no display options.

You can edit the properties of all displays using the **Options** menu. You can configure the grid lines and background colors. In the 2-D image display you can click points in the image to see their values. All 3-D displays can be rotated as usual. You can edit all color map bars by double-clicking.

Export of Design Evaluation Information

All information displayed in the Design Evaluation tool can be exported to the workspace or to a .mat file using the radio buttons and **Export** button at the bottom right. You can enter a variable name in the edit box.

Maximum Likelihood Estimation

Maximum Likelihood Estimation for Nonlinear Repeated Measures

This section contains an overview of the mathematics of two-stage models. A comprehensive reference for two-stage modeling is Davidian and Giltinan [3]. The information is divided into the following sections:

- "Two-Stage Models for Engines" on page 8-40
- "Local Models" on page 8-41
  - "Local Covariance Modeling" on page 8-41
  - "Response Features" on page 8-43
- "Global Models" on page 8-43
- "Two-Stage Models" on page 8-44
  - "Prediction Error Variance for Two-Stage Models" on page 8-46
  - "Global Model Selection" on page 8-48
  - "Initial Values for Covariances" on page 8-48
  - "Quasi-Newton Algorithm" on page 8-48
  - "Expectation Maximization Algorithm" on page 8-48
- "References" on page 8-49

Two-Stage Models for Engines

Lindstrom and Bates [6] define repeated measurements as data generated by observing a number of individuals repeatedly under various experimental conditions, where the individuals are assumed to constitute a random sample from a population of interest. An important class of repeated measurements is longitudinal data where the observations are ordered by time or position in space. More generally, longitudinal data is defined as repeated measurements where the observations on a single individual are not, or cannot be, randomly assigned to the levels of a treatment of interest.

Modeling data of this kind usually involves the characterization of the relationship between the measured response, y, and the repeated measurement factor, or covariate x. Frequently, the underlying systematic relationship between y and x is nonlinear. In some cases the relevant nonlinear model can be derived on physical or mechanistic grounds. However, in other contexts a nonlinear relationship might be imposed simply to provide a convenient empirical description for the data. The presence of repeated observations on an individual requires particular care in characterizing the variation in the experimental data. In particular, it is important to represent two sources of variation explicitly: random variation among measurements within a given individual (*intraindividual*) and random variation among individuals (*interindividual*). Inferential procedures accommodate these different variance components within the framework of an appropriate hierarchical statistical model. This is the fundamental idea behind the analysis of repeated measurement data.

Holliday [1,2] was perhaps the first to apply nonlinear repeated measurements analysis procedures to spark injection engine data. The focus of Holliday's work was the modeling of data taken from engine mapping experiments. In these experiments, engine speed, load, and air/fuel ratio were held constant while spark was varied. Various engine response characteristics, for example, torque or emission quantities, were measured at each spark setting. Holliday modeled the response characteristics for each sweep as a function of spark advance. Variations in the individual sweep parameters were then modeled as a function of the global engine operating variables speed, load, and air/fuel ratio. Conceptually, variations in the measurements taken within a sweep represent the intraindividual component of variance. Similarly, variation in the sweep-specific parameters between sweeps represents the interindividual component of variance. You can generalize these principles to other steady-state engine modeling exercises where the nature of data collection

usually involves sweeping a single engine control variable while the remainder are held at fixed values. These points suggest that nonlinear repeated measurements analysis represents a general approach to the parameterization of mean value engines models for controls-oriented development.

Another application for models of this form is the flow equations for a throttle body. Assuming the flow equations are based upon the usual one-dimensional isentropic flow principle, then they must be modified by an effective area term, A_e, which accounts for the fact that the true flow is multidimensional and irreversible. You can map the throttle flow characteristics by sweeping the throttle position at fixed engine speed. This data collection methodology naturally imposes a hierarchy the analysis of which is consistent with the application of nonlinear repeated measures. Experience in modeling effective area suggests that free knot spline or biological growth models provide good local predictions. The global phase of the modeling procedure is concerned with predicting the systematic variation in the response features across engine speed. A free knot spline model has proven useful for this purpose.

Local Models

Modeling responses locally within a sweep as a function of the independent variable only. That is,

$$y_i^j = f_i(s_i^j, \theta_i) + \varepsilon_i^j \text{ for } j = 1, 2, \dots m_i \tag{8-1}$$

where the subscript i refers to individual tests and j to data within a test, s_i^j is the jth independent value, θ_i is a (rx1) parameter vector, y_i^j is the jth response, and ε_i^j is a normally distributed random variable with zero mean and variance σ^2. Note that Equation 8-1 can be either a linear or a nonlinear function of the curve fit parameters. The assumption of independently normally distributed errors implies that the least squares estimates of θ are also maximum likelihood parameters.

Local Covariance Modeling

The local model describes both the systematic and random variation associated with measurements taken during the ith test. Systematic variation is characterized through the function f while variation is characterized via the distributional assumptions made on the vector of random errors e_i. Hence, specification of a model for the distribution of e_i completes the description of the

intratest model. The Model-Based Calibration Toolbox allows a very general specification of the local covariance:

$$e_i \sim N(0, \sigma^2 C_i(\beta_i, \xi_i)) \tag{8-2}$$

where C_i is an (n_i x n_i) covariance matrix, σ^2 is the coefficient of variation, and ξ_i is a (q-by-1) vector of dispersion parameters that account for heterogeneity of variance and the possibility of serially correlated data. The specification is very general and affords considerable flexibility in terms of specifying a covariance model to adequately describe the random component of the intratest variation.

The Model-Based Calibration Toolbox supports the following covariance models:

• Power Variance Model:

$$C_i = diag\left\{ f(x_i, \beta_i)^{\xi_1} \right\} \tag{8-3}$$

• Exponential Variance Model:

$$C_i = diag\{ \exp(f(x_i, \beta_i)\xi_1) \} \tag{8-4}$$

• Mixed Variance Model:

$$C_i = diag\left\{ \xi_1 + f(x_i, \theta_i)^{\xi_2} \right\} \tag{8-5}$$

where diag{x} is a diagonal matrix.

Correlation models are only available for equispaced data in the Model-Based Calibration Toolbox. It is possible to combine correlation models with models with the variance models such as power.

One of the simplest structures that can be used to account for serially correlated errors is the AR(m) model (autoregressive model with lag m). The general form of the AR(m) model is

$$e_j = \phi_1 e_{j-1} + \phi_2 e_{j-2} + \dots + \phi_m e_{j-m} + v_j \tag{8-6}$$

where ϕ_k is the kth lag coefficient and v_j is an exogenous stochastic input identically and independently distributed as $N(0,\sigma_v^2)$. First- and second-order autoregressive models are implemented in the Model-Based Calibration Toolbox.

Another possibility is a moving average model (MA). The general structure is

$$e_j = \phi_1 v_{j-1} + \phi_2 v_{j-2} + \dots + \phi_m v_{j-m} + v_j \qquad \textbf{(8-7)}$$

where ϕ_k is the kth lag coefficient and v_j is an exogenous stochastic input identically and independently distributed as $N(0,\sigma_v^2)$. Only a first-order moving average model is implemented in the Model-Based Calibration Toolbox.

Response Features

From an engineering perspective, the curve fit parameters do not usually have any intuitive interpretation. Rather characteristic geometric features of the curve are of interest. The terminology "response features" of Crowder and Hand [7] is used to describe these geometric features of interest. In general, the response feature vector $\mathbf{p}_i$ for the ith sweep is a nonlinear function ($\mathbf{g}$) of the corresponding curve fit parameter vector θ_i, such that

$$\mathbf{p}_i = g(\theta_i) \qquad \textbf{(8-8)}$$

Global Models

Modeling the variation in the response features as a function of the global variables. The response features are carried through to the second stage of the modeling procedure rather than the curve fit parameters because they have an engineering interpretation. This ensures that the second stage of the modeling process remains relatively intuitive. It is much more likely that an engineer will have better knowledge of how a response feature such as MBT behaves throughout the engine operating range (at least on a main effects basis) as opposed to an esoteric curve fit parameter estimate.

The global relationship is represented by one of the global models available in the Model-Based Calibration Toolbox. In this section we only consider linear models that can be represented as

$$\mathbf{p}_i = X_i\beta + \gamma_i \ \textbf{ for } \ i = 1, 2, \dots, r \qquad \textbf{(8-9)}$$

where the X_i contains the information about the engine operating conditions at the ith spark sweep, β is the vector of global parameter estimates that must be estimated by the fitting procedure, and γ_i is a vector of normally distributed random errors. It is necessary to make some assumption about the error distribution for γ, and this is typically a normal distribution with

$$\gamma_i \sim N_r(0, D) \tag{8-10}$$

where r is the number of response features. The dimensions of D are (rxr) and, being a variance-covariance matrix, D is both symmetric and positive definite. Terms on the leading diagonal of D represent the test-to-test variance associated with the estimate of the individual response features. Off-diagonal terms represent the covariance between pairs of response features. The estimation of these additional covariance terms in a multivariate analysis improves the precision of the parameter estimates.

Two-Stage Models

To unite the two models, it is first necessary to review the distributional assumptions pertaining to the response feature vector p_i. The variance of p_i ($Var(p_i)$) is given by

$$Var(\mathbf{p}_i) = \left[\frac{\partial g(\theta_i)}{\partial \theta}\right]\sigma^2 C_i\left[\frac{\partial g(\theta_i)}{\partial \theta}\right]^T \tag{8-11}$$

For the sake of simplicity, the notation $\sigma^2 C_i$ is to denote $Var(p_i)$. Thus, p_{ii} is distributed as

$$\mathbf{p}_i \sim N_r(\mathbf{p}_i, \sigma^2 C_i) \tag{8-12}$$

where C_i depends on f_i through the variance of θ_i and also on g_i through the conversion of θ_i to the response features p_i. Two standard assumptions are used in determining C_i: the asymptotic approximation for the variance of maximum likelihood estimates and the approximation for the variance of functions of maximum likelihood estimates, which is based on a Taylor series expansion of g_i. In addition, for nonlinear f_i or g_i, C_i depends on the unknown θ_i; therefore, we will use the estimate $\hat{\theta}_i$ in its place. These approximations are likely to be good in the case where σ^2 is small or the number of points per sweep (m_i) is large. In either case we assume that these approximations are valid throughout.

We now return to the issue of parameter estimation. Assume that the γ_i are independent of the ε_i^j. Then, allowing for the additive replication error in response features, the response features are distributed as

$$p_i \sim N(X_i\beta, \sigma^2 C_i + D) \tag{8-13}$$

When all the tests are considered simultaneously, equation (6-13) can be written in the compact form

$$P \sim N(Z\beta, W(\varpi)) \tag{8-14}$$

where P is the vector formed by stacking the n vectors p_i on top of each other, Z is the matrix formed by stacking the n X_i matrices, W is the block diagonal weighting matrix with the matrices on the diagonal being $\sigma^2 C_i + D$, and ω is a vector of dispersion parameters. For the multivariate normal distribution (6-14) the negative log likelihood function can be written:

$$\log L(\beta, \varpi) = \log|W| + (P - Z\beta)' W^{-1} (P - Z\beta) \tag{8-15}$$

Thus, the maximum likelihood estimates are the vectors β_{ML} and ω_{ML} that minimize $\log L(\beta,\omega)$. Usually there are many more fit parameters than dispersion parameters; that is, the dimension of β is much larger than ω. As such, it is advantageous to reduce the number of parameters involved in the minimization of $\log L(\beta,\omega)$. The key is to realize that equation (6-15) is conditionally linear with respect to β. Hence, given estimates of ω, equation (6-15) can be differentiated directly with respect to β and the resulting expression set to zero. This equation can be solved directly for β as follows:

$$\beta = (Z'W^{-1}Z)^{-1}(Z'W^{-1}P) \tag{8-16}$$

The key point is that now the likelihood depends only upon the dispersion parameter vector ω, which as already discussed has only modest dimensions. Once the likelihood is minimized to yield ω_{ML}, then, since $W(\omega_{ML})$ is then known, equation (6-16) can subsequently be used to determine β_{ML}.

Prediction Error Variance for Two-Stage Models

It is very useful to evaluate a measure of the precision of the model's predictions. You can do this by looking at Prediction Error Variance (PEV). Prediction error variance will tend to grow rapidly in areas outside the original design space. The following section describes how PEV is calculated for two-stage models.

For linear global models applying the variance operator to Equation 8-15 yields:

$$\text{Var}(\beta) = (Z^T W^{-1} Z)^{-1} Z^T W^{-1} \text{Var}(P) W^{-1} Z (Z^T W^{-1} Z)^{-1} \text{ so}$$

$$\text{Var}(\beta) = (Z^T W^{-1} Z)^{-1}, \tag{8-17}$$

since Var(P) = W. Assume that it is required to calculate both the response features and their associated prediction error variance for the ith test. the predicted response features are given by:

$$\hat{p}_i = z_i \hat{\beta} \tag{8-18}$$

where z_i is an appropriate global covariate matrix. Applying the variance operator to Equation 8-18 yields:

$$\text{Var}(\hat{p}_i) = z_i \text{Var}(\hat{\beta}) z_i^T = z_i (Z^T W^{-1} Z)^{-1} z_i^T \tag{8-19}$$

In general, the response features are non-linear functions of the local fit coefficients. Let g denote the non-linear function mapping θ_i onto p_i. Similarly let h denote the inverse mapping.

$$\hat{\theta}_i = h(\hat{p}_i) \tag{8-20}$$

Approximating h using a first order Taylor series expanded about p_i (the true and unknown fixed population value) and after applying the variance operator to the result:

$$\text{Var}(\hat{\theta}_i) = \dot{h} \text{Var}(\hat{p}_i) \dot{h}^T \tag{8-21}$$

where the dot notation denotes the Jacobian matrix with respect to the response features, p_i. This implies that $\dot{h}$ is of dimension (pxp). Finally the predicted response values are calculated from:

$$\hat{\mathbf{y}}_i = \mathbf{f}(\theta_i) \tag{8-22}$$

Again, after approximating f by a first order Taylor series and applying the variance operator to the result:

$$\text{Var}(\hat{\mathbf{y}}_i) = \left[\frac{\partial \mathbf{f}}{\partial \theta}\right]\bigg|_{\hat{\theta}} \dot{\mathbf{h}}\text{Var}(\hat{\mathbf{p}}_i)\dot{\mathbf{h}}^{\mathrm{T}}\left[\frac{\partial \mathbf{f}}{\partial \theta}\right]\bigg|_{\hat{\theta}}^{\mathrm{T}} \tag{8-23}$$

After substituting Equation 8-19 into Equation 8-23 the desired result is obtained:

$$\text{Var}(\hat{\mathbf{y}}_i) = \left[\frac{\partial \mathbf{f}}{\partial \theta}\right]\bigg|_{\hat{\theta}} \dot{\mathbf{h}}\mathbf{z}_i(\mathbf{Z}^{\mathrm{T}}\mathbf{W}^{-1}\mathbf{Z})^{-1}\mathbf{z}_i^{\mathrm{T}}\dot{\mathbf{h}}^{\mathrm{T}}\left[\frac{\partial \mathbf{f}}{\partial \theta}\right]\bigg|_{\hat{\theta}}^{\mathrm{T}} \tag{8-24}$$

This equation gives the value of Prediction Error Variance.

See also the introduction to "Prediction Error Variance" on page 8-7 for details about PEV for one-stage models.

Global Model Selection

Before undertaking the minimization of Equation 8-15 (see "Two-Stage Models" on page 8-44) it is first necessary to establish the form of the $\mathbf{X}_i$ matrix. This is equivalent to establishing a global expression for each of the response features a priori. Univariate stepwise regression is used to select the form of the global model for each response feature. Minimization of the appropriate PRESS statistic is used as a model building principle, as specified in "High-Level Model Building Process Overview" on page 8-9. The underlying principle is that having used univariate methods to establish possible models, maximum likelihood methods are subsequently used to estimate their parameters.

Initial Values for Covariances

An initial estimate of the global covariance is obtained using the standard two-stage estimate of Steimer *et al.* [10],

$$\mathbf{D}_{\text{STS}} = \frac{1}{r-1} \sum_{i=1}^{r} (\mathbf{p}_i - \mathbf{X}_i\beta)(\mathbf{p}_i - \mathbf{X}_i\beta)^{\text{T}} \tag{8-25}$$

where β are the estimates from all the univariate global models. This estimate is biased.

Quasi-Newton Algorithm

Implicit to the minimization of equation (6-17) is that $\mathbf{D}$ is positive definite. It is a simple matter to ensure this by noting that $\mathbf{D}$ is positive definite if and only if there is an upper triangular matrix, $\mathbf{G}$, say, such that

$$\mathbf{D} = \mathbf{G}'\mathbf{G} \tag{8-26}$$

This factorization is used in the Quasi-Newton algorithm. Primarily, the advantage of this approach is that the resulting search in $\mathbf{G}$, as opposed to $\mathbf{D}$, is unconstrained.

Expectation Maximization Algorithm

The expectation maximization algorithm is an iterative method that converges toward the maximal solution of the likelihood function. Each iteration has two steps:

1 Expectation Step — Produce refined estimates of the response features given the current parameter estimates.

2 Maximization Step — Obtain new estimates of the parameters (global model parameters and covariance matrix) for the new response features.

These steps are repeated until the improvement in value of the log likelihood function is less than the tolerance. Details of the algorithm can be found in [3, Chapter 5].

References

1 Holliday, T., *The Design and Analysis of Engine Mapping Experiments: A Two-Stage Approach*, Ph.D. thesis, University of Birmingham, 1995.

2 Holliday, T., Lawrance, A. J., Davis, T. P., Engine-Mapping Experiments: A Two-Stage Regression Approach, *Technometrics*, 1998, Vol. 40, pp 120-126.

3 Davidian, M., Giltinan, D. M., *Nonlinear Models for Repeated Measurement Data*, Chapman & Hall, First Edition, 1995.

4 Davidian, M., Giltinan, D. M., Analysis of repeated measurement data using the nonlinear mixed effects model, *Chemometrics and Intelligent Laboratory Systems*, 1993, Vol. 20, pp 1-24.

5 Davidian, M., Giltinan, D. M., Analysis of repeated measurement data using the nonlinear mixed effects model, *Journal of Biopharmaceutical Statistics*, 1993, Vol. 3, part 1, pp 23-55.

6 Lindstrom, M. J., Bates, D. M., Nonlinear Mixed Effects Models for Repeated Measures Data, *Biometrics*, 1990, Vol. 46, pp 673-687.

7 Davidian, M., Giltinan, D. M., Some Simple Methods for Estimating Intraindividual Variability in Nonlinear Mixed Effects Models, *Biometrics*, 1993, Vol. 49, pp 59-73.

8 Hand, D. J., Crowder, M. J., *Practical Longitudinal Data Analysis*, Chapman and Hall, First Edition, 1996.

9 Franklin, G.F., Powell, J.D., Workman, M.L., *Digital Control of Dynamic Systems*, Addison-Wesley, Second Edition, 1990.

10 Steimer, J.-L., Mallet, A., Golmard, J.L., and Boisvieux, J.F., Alternative approaches to estimation of population pharmacokinetic parameters: Comparison with the nonlinear mixed effect model. *Drug Metabolism Reviews*, 1984, **15**, 265-292.

Local Model Definitions

Local Models and Associated Response Features

In the following sections are listed the model classes and their associated response features available for modeling at the local node in MBC.

"Polynomial Models" on page 8-51

"Polynomial Splines" on page 8-52

"Truncated Power Series Basis (TPSBS) Splines" on page 8-53

"Free Knot Splines" on page 8-53

"Three Parameter Logistic Model" on page 8-54

"Morgan-Mercer-Flodin Model" on page 8-55

"Four-Parameter Logistic Curve" on page 8-56

"Richards Curves" on page 8-57

"Weibul Growth Curve" on page 8-58

"Exponential Growth Curve" on page 8-58

"Gompertz Growth Model" on page 8-59

Polynomial Models

MBC includes extensive capabilities for using polynomials of arbitrary order to model local phenomena. The following response features are permitted for the polynomial model class:

- Location of the maximum or minimum value (when using datum models; note that the datum model is not used in reconstructing).
- Value of the fit function at a user-specified x-ordinate. When datum models are used, the value is relative to the datum (for example, mbt - x).
- The nth derivative at a user-specified x-ordinate, for n = 1, 2, …, d where d is the degree of the polynomial.

Polynomial Splines

These are essential for modeling torque/spark curves. To model responses that are characterized in appearance by a single and well defined stationary point with asymmetrical curvature either side of the local minimum or maximum, we define the following spline class,

$$y_{ij} = \beta_o + \sum_{a=2}^{c} \beta_{Low\_a} (x_j - k)_+^a + \sum_{b=2}^{h} \beta_{High\_b} (x_j - k)_+^b$$

where k is the knot location, β denotes a regression coefficient,

$$(x_j - k)_- = \min\{0, (x_j - k)\}, \quad (x_j - k)_+ = \max\{0, (x_j - k)\},$$

where c is the user-specified degree for the left polynomial, h is the user-specified degree for the right polynomial, and the subscripts Low and High denote to the left (below) and right of (above) the knot, respectively.

Note that by excluding terms in $(x_j - k)_-$ and $(x_j - k)_+$ we ensure that the first derivative at the knot position is continuous. In addition, by definition the constant β_o must be equal to the value of the fit function at the knot, that is, the value at the stationary point.

For this model class, response features can be chosen as

- Fit constants $\{\beta_o, \beta_{Low\_2}, \ldots, \beta_{Low\_p}, \beta_{High\_2}, \beta_{High\_q}\}$
- Knot position $\{k\}$
- Value of the fit function at a user-specified delta

   $$\{\Delta a_j = x_j - k\}$$

   from the knot position $\{f(\pm \Delta a_j)\}$ if the datum is defined, otherwise the value is absolute.

- Difference between the value of the fit function at a user-specified delta from the knot position and the value of the fit function at the knot

   $$\{f(\pm \Delta a_j) - f(k)\}$$

Truncated Power Series Basis (TPSBS) Splines

A very general class of spline functions with arbitrary (but strictly increasing) knot sequence:

$$\mathbf{k} = \left\{ k_1, k_2, \ldots, k_k \right\}^T :$$

$$f(x) = \sum_{i=0}^{m} \beta_i x^i + \sum_{i=0}^{k} \beta_{m+i}(x - k_i)_+^m$$

This defines a spline of order *m* with knot sequence

$$\mathbf{k} = \left\{ k_1, k_2, \ldots, k_k \right\}^T$$

For this model class, response features can be chosen as

- Fit constants $\left\{ \beta_o, \beta_1, \ldots, \beta_{m-1+k} \right\}$
- Knot position vector $\left\{ \mathbf{k} \right\}$
- Value of the fit function $\left\{ f(a_j) \right\}$ at a user-specified value $\left\{ a_j \right\}$
- Value of the nth derivative of the fit function with respect to x_j $\left\{ f(a_j)^n \right\}$ at a user-specified value $\left\{ a_j \right\}$, with n = 1, 2, …, m-2

Any of the polynomial terms can be removed from the model.

Free Knot Splines

The $\left(x_j - k_i \right)_+^{m-1}$ basis is not the best suited for the purposes of estimation and evaluation, as the design matrix might be poorly conditioned. In addition, the number of arithmetic operations required to evaluate the spline function depends on the location of x_j relative to the knots. These properties can lead to numeric inaccuracies, especially when the number of knots is large. You can reduce such problems by employing *B-splines*.

The most important aspect is that for moderate m the design matrix expressed in terms of B-splines is relatively well conditioned.

For this model class, response features can be chosen as

- Fit constants $\left\{\beta_{-(m-1)}, \beta_{-m}, \ldots, \beta_k\right\}$
- Knot position vector $\{\mathbf{k}\}$
- Value of the fit function $\left\{f(a_j)\right\}$ at a user specified value $\left\{a_j\right\}$

Three Parameter Logistic Model

The three parameter logistic curve is defined by the equation

$$y_j = \frac{\alpha}{1 + \exp\left(-\kappa\left(x_j - \gamma\right)\right)} r$$

where α is the final size achieved, κ is a scale parameter, and γ is the x-ordinate of the point of inflection of the curve.

The curve has asymptotes $y_j = 0$ as $x_j \to -\infty$ and $y_j = \alpha$ as $x_j \to \infty$. Growth rate is at a maximum when $y_j = \alpha/2$, which occurs when $x_j = \gamma$. Maximum growth rate corresponds to

$$\frac{\kappa\alpha}{4}$$

The following constraints apply to the fit coefficients:

$$\alpha > 0, \, \kappa > 0, \, \gamma > 0$$

The response feature vector **g** for the 3 parameter logistic function is defined as

$$g = \left[\alpha \; \gamma \; \kappa \; \frac{\kappa\alpha}{4}\right]^T$$

Morgan-Mercer-Flodin Model

The Morgan-Mercer-Flodin (MMF) growth model is defined by

$$y_j = \alpha - \frac{\alpha - \beta}{\left(1 + \left(\kappa x_j\right)^\delta\right)}$$

where α is the value of the upper asymptote, β is the value of the lower asymptote, κ is a scaling parameter, and δ is a parameter that controls the location of the point of inflection for the curve. The point of inflection is located at

$$x = \left[\frac{\delta - 1}{\delta + 1}\right]^{1/\delta}$$

$$y = \frac{\delta - 2}{2\delta}$$

for $\delta \geq 1$

There is no point of inflection for $\delta < 1$. All the MMF curves are sublogistic, in the sense that the point of inflection is always located below 50% growth (0.5α). The following constraints apply to the fit coefficient values:

$\alpha > 0$, $\beta > 0$, $\kappa > 0$, $\delta > 0$

$\alpha > \beta$

The response feature vector **g** is given by

$$g = \left[\alpha \ \ \beta \ \ \kappa \ \ \delta \ \ \frac{\delta - 1}{2\delta}\right]^{T}$$

Four-Parameter Logistic Curve

The four-parameter logistic model is defined by

$$y_j = \beta + \frac{\alpha - \beta}{1 + \exp\left(-\kappa\left(\log(x_j) - \gamma\right)\right)}$$

with constraints $\alpha, \beta, \kappa, \gamma > 0$, $\beta < \alpha$ and $\beta < \gamma < \alpha$. Again, α is the value of the upper asymptote, κ is a scaling factor, and γ is a factor that locates the x-ordinate of the point of inflection at

$$\exp\left(\frac{\kappa\gamma - \log\left(\frac{1 + \kappa}{\kappa - 1}\right)}{k}\right)$$

The following constraints apply to the fit coefficient values:

- $\alpha > 0$, $\beta > 0$, $\kappa > 0$, $\gamma > 0$
- $\alpha > \beta$
- $\alpha > \gamma > \beta$

This the available response feature vector:

$$g = \left[\alpha \ \beta \ \kappa \ \gamma \ \frac{\left(\kappa\gamma - \log\left(\frac{1 + \kappa}{\kappa - 1}\right)\right)}{\kappa}\right]^T$$

Richards Curves

The Richards curves family of growth models is defined by the equation

$$y_j = \alpha \Big[1 + (\delta - 1) \exp\big(-\kappa(x_j - \gamma)\big) \Big]^{\frac{1}{(1-\delta)}} \quad \delta \neq 1$$

where α is the upper asymptote, γ is the location of the point of inflection on the x axis, κ is a scaling factor, and δ is a parameter that indirectly locates the point of inflection. The y-ordinate of the point of inflection is determined from

$$\frac{\alpha}{\delta^{1/(\delta - 1)}} \quad \delta > 0$$

Richards also derived the average normalized growth rate for the curve as

$$\frac{\kappa}{2(\delta + 1)}$$

The following constraints apply to the fit coefficient values:

- $\alpha > 0$, $\gamma > 0$, $\kappa > 0$, $\delta > 0$
- $\alpha > \gamma$
- $\delta \neq 1$

Finally, the response feature vector **g** for Richards family of growth curves is defined as

$$g = \left[\alpha \ \gamma \ \kappa \ \delta \ \frac{\delta \kappa}{\delta 2 \delta + 1} \right]^{\mathrm{T}}$$

Weibul Growth Curve

The Weibul growth curve is defined by the equation

$$y_j = \alpha - (\alpha - \beta) \exp\left(-\left(\kappa x_j\right)^\delta\right)$$

where α is the value of the upper curve asymptote, β is the value of the lower curve asymptote, κ is a scaling parameter, and δ is a parameter that controls the x-ordinate for the point of inflection for the curve at

$$\left(\frac{1}{\kappa}\right)\left(\frac{\delta - 1}{\delta}\right)^{1/\delta}$$

The following constraints apply to the curve fit parameters:

- $\alpha > 0, \beta > 0, \kappa > 0, \delta > 0$
- $\alpha > \beta$

The associated response feature vector is

$$g = \left[\alpha \ \beta \ \kappa \ \delta \ \left(\frac{1}{\kappa}\right)\left(\frac{\delta - 1}{\delta}\right)^{1/\delta}\right]^{\mathrm{T}}$$

Exponential Growth Curve

The exponential growth model is defined by

$$y_j = \alpha - (\alpha - \beta) \exp\left(-\kappa x_j\right)$$

where α is the value of the upper asymptote, β is the initial size, and κ is a scale parameter (time constant controlling the growth rate). The following constraints apply to the fit coefficients:

- $\alpha > 0, \beta > 0, \kappa > 0$
- $\alpha > \beta$

The response feature vector **g** for the exponential growth model is defined as

$$\mathbf{g} = \begin{bmatrix} \alpha & \beta & \kappa \end{bmatrix}^{\mathrm{T}}$$

Gompertz Growth Model

Another useful formulation that does not exhibit a symmetric point of inflection is the Gompertz growth model. The defining equation is

$$y_j = \alpha \exp\left(-e^{-\kappa\left(x_j - \gamma\right)}\right)$$

where α is the final size achieved, κ is a scaling factor, and γ is the x-ordinate of the point of inflection. The corresponding y-ordinate of the point of inflection occurs at

$$\frac{\alpha}{e}$$

With maximum growth rate

$$\frac{\kappa\alpha}{e}$$

The following constraints apply to the selection of parameter values for the Gompertz model:

$$\alpha > 0, \ \kappa > 0, \ \gamma > 0$$

The response feature vector **g** for the Gompertz growth model is defined as

$$g = \left[\alpha\gamma\kappa\frac{\kappa\alpha}{e}\right]^{T}$$

Neural Networks

For help on the neural net models implemented in the Model-Based Calibration Toolbox, see the documentation in the Neural Nets Toolbox ™. At the MATLAB command line, enter

```
>>doc nnet
```

The training algorithms available in the Model-Based Calibration Toolbox are `traingdm, trainlm, trainbr`.

These algorithms are a subset of the ones available in the neural nets toolbox. (The names indicate the type: gradient with momentum, named after the two authors, and bayesian reduction). Neural networks are inspired by biology, and attempt to emulate learning processes in the brain.

Neural nets contain no preconceptions of what the model shape will be, so they are ideal for cases with low system knowledge. They are useful for functional prediction and system modeling where the physical processes are not understood or are highly complex.

The disadvantage of neural nets is that they require a lot of data to give good confidence in the results, so they are not suitable for small data sets. Also, with higher numbers of inputs, the number of connections and hence the complexity increase rapidly.

MBC provides an interface to some of the neural network capability of the Neural Network Toolbox. Therefore these functions are only available if the Neural Network Toolbox is installed. See the Neural Network Toolbox documentation for more help.

User-Defined Models

Throughout this section we refer to the template file

```
<MATLAB root>\toolbox\mbc\mbcmodel\@xregusermod\weibul.m
```

This file defines the local model Weibul in the Model-Based Calibration Toolbox and hence should not be changed. Make sure you do not overwrite the file by saving under another name if you alter it.

The xregusermod class allows the user to define new local models. When the data being modeled at the local level matches the profile of (checked-in) user-defined local models, these models appear as options on the **Local Model Setup** dialog. At the end of the following section MBC offers the user-defined model weibul under the category User-defined in the **Local Model Setup** dialog.

To Begin

Decide on the function that you want to use. We will be using the Weibul function:

```
y = alpha - (alpha - beta).*exp(-(kappa.*x).^delta)
```

Template File

Open the file

```
<MATLAB root>\toolbox\mbc\mbcmodel\@xregusermod\weibul.m
```

The M-file is called using

```
varargout= weibul(U,X,varargin)
```

Where the variables are given by

```
U = the xregusermod object
X = input data as a column vector for fast eval -OR-
X = specifies what sub-fcn to evaluate (not usually called
directly)
```

The first function in the template file is a vectorized evaluation of the function.

First the model parameters are extracted:

```
b= double(U);
```

Then the evaluation occurs:

```
y = b(1) - (b(1)-b(2)).*exp(-(b(3).*x).^b(4));
```

Note that the parameters are always referred to and declared in the same order.

Subfunctions

Those subfunctions that must be edited are as follows:

```
function n= i_nfactors(U,b,varargin);
n= 1;
```

This is the number of input factors. For functions $y = f(x)$ this is 1.

```
function n= i_numparams(U,b,varargin);
n= 4;
```

This is the number of fitted parameters. In this example there are four parameters in the Weibul model.

```
function [param,OK]= i_initial(U,b,X,Y)
param= [2 1 1 1]';
OK=1;
```

This subfunction returns a column vector of initial values for the parameters to be fitted. The initial values can be defined to be data dependent; hence there is a flag to signal if the data is not worth fitting. In the template file weibul.m there is a routine for calculating a data-dependent initial parameter estimate. If no data is supplied, the default parameter values are used.

Optional Subfunctions

```
function [LB,UB,A,c,nlcon,optparams]=i_constraints(U,b,varargin)

LB=[0 0 0 0]';
UB=[1e10 1e10 1e10 1e10]';

A= [-1 1 0 0];
c= [0];

nlcon= 0;
```

```
optparams= [];
```

Lower and upper bounds are stated for the model parameters. These are in the same order that the parameters are declared and used throughout the template file.

A linear constraint is defined:

b = the vector of model parameters, then the constraint is A*b <c
We define A and c in the subfunction above

The number of nonlinear constraints is declared to be zero. If the number of nonlinear constraints is not zero, the nonlinear constraints are calculated in `i_nlconstraints`.

No optional parameters are declared for the cost function.

```
function fopts= i_foptions(U,b,fopts)

    fopts= optimset(fopts,'Display','iter');
```

The fit options are always based on the input `fopts`. See MATLAB help on the function `optimset` for more information on fit options. When there are no constraints the fitting is done using the MATLAB function `lsqnonlin`, otherwise `fmincon` is used.

```
function J= i_jacobian(U,b,x)

    x = x(:);
    J= zeros(length(x),4);

    a=b(1); beta=b(2); k=b(3); d=b(4);

    ekd= exp(-(k.*x).^d);
    j2= (a-beta).*(k.*x).^d.*ekd;

    J(:,1)= 1-ekd;
    J(:,2)= ekd;
    J(:,3)= j2.*d./k;
    J(:,4)= j2.*log(k.*x);
```

To speed up the fitting algorithm an analytic Jacobian can be supplied, as it is here.

```
function c= i_labels(U,b)
c={'\alpha','\beta','\kappa','\delta'};
```

These labels are used on plots and so on. Latex notation can be used and is formatted.

```
function str= i_char(U,b,fopts)

s= get(U,'symbol');
str=sprintf('%.3g - (%.3g-%.3g)*exp(-(%.3g*x)^{%.3g})',...
       b([1 1 2 3 4]));
```

This is the display equation string and can contain Latex expressions. The current values of model parameters appear.

```
function str= i_str_func(U,b)

s= get(U,'symbol');
lab= labels(U);
str= sprintf('%s - (%s - %s)*exp(-(%s*x)^{%s})',...
       lab{1},lab{1},lab{2},lab{3},lab{4});
```

This displays the function definition with labels appearing in place of the parameters (not numerical values).

```
function rname= i_rfnames(U,b)
rname= {'INFLEX'};
```

This does not need to be defined (can return an empty array). Here we define a response feature that is not one of the parameters (here it is also nonlinear).

```
function [rf,dG]= i_rfvals(U,b)

% response feature definition
rf= (1/b(3))*((b(4)-1)/b(4))^(1/b(4))

if nargout>1
   % delrf/delbi
   dG= [0, 0, -((b(4)-1)/b(4))^(1/b(4))/b(3)^2,...
       1/b(3)*((b(4)-1)/b(4))^(1/b(4))*...
       (-1/b(4)^2*log((b(4)-1)/b(4))+(1/b(4)-...
       (b(4)-1)/b(4)^2)/(b(4)-1))];
   end
```

The response feature (labeled as INFLEX above) is defined. The jacobian is also defined here as dG.

```
function p= i_reconstruct(U,b,Yrf,dG,rfuser)
p= Yrf/dG';

f= find(rfuser>size(p,2));
if any(rfuser==4)
   % need to use delta
   p(:,3)= ((p(:,4)-1)./p(:,4)).^(1./p(:,4))./Yrf(:,f);
end
```

If all response features are linear in the parameters this function does not need to be defined. Here we must first find out which response features (if any) are user-defined. This subfunction allows the model parameters to be reconstructed from the response features we have been given.

Checking In To MBC

Having created a model template file, save it somewhere on the path.

To ensure that the model you have defined provides an interface that allows MBC to evaluate and fit it, we *check in* the model. If this procedure succeeds, the model is registered by MBC and is thereafter available for fitting at the local level whenever appropriate input data is being used.

Check In

At the command line, with the template file on the path, create a model and some input data, then call checkin. For the user-defined Weibul function, the procedure is as follows:

Create an xregusermod using the template model file called weibul.

```
m = xregusermod('name','weibul');
```

Call checkin with the transient model, its name, and some appropriate data.

```
checkin(m, 'weibul', [0.1:0.01:0.2]');
```

This creates some command line output, and a figure appears with the model name and variable names displayed over graphs of input and model evaluation output. The final command line output (if checkin is called with a semicolon as above) is

```
Model successfully registered for MBC
```

The figure displayed for weibul is
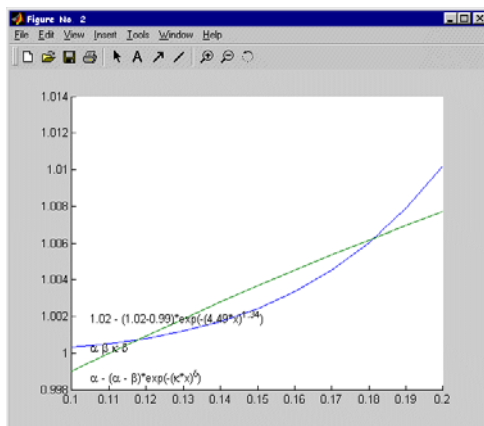
MBC Modeling

Start up MBC and load data that has the necessary format to evaluate your user-defined model.

Set up a **Test Plan** with one Stage1 input factor. The **Local Model Setup** now offers the following options:
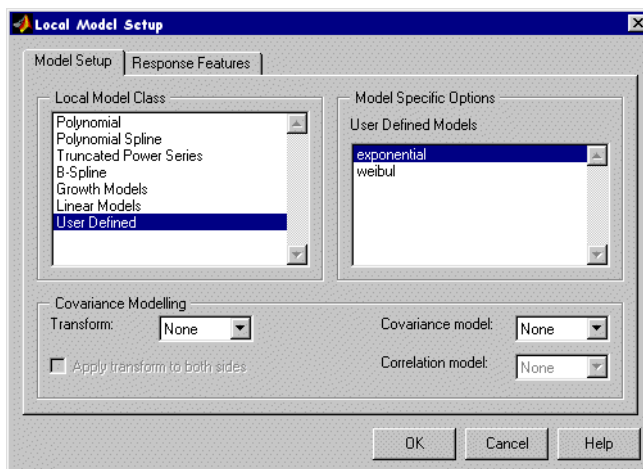
Here two user-defined models exist: **exponential** and **weibul**, as checked in.

Transient Models

Transient models are supported in the Model-Based Calibration Toolbox, for multiple input factors where time is one of the factors. Note that the example porovided erquires two input factors. You can define a dynamic model using a Simulink model (`.mdl` file)and a template M-file that describes parameters to be fitted in this model. You must check these into the Model-Based Calibration Toolbox before you can use them for modeling.

We provide an example called `fuelpuddle` which is already checked in to the toolbox ready for modeling. The process of creating and checking in your own transient models is described in the following sections using this example.

Throughout this section you use this Simulink model:

```
<MATLAB root>\toolbox\mbc\mbcsimulink\fuelPuddle.mdl
```

and the template file

```
<MATLAB root>\toolbox\mbc\mbcmodel\@xregtransient\fuelPuddle.m
```

The Simulink Model

Create a Simulink model to represent your transient system. This has some free parameters that MBC fits by regression. There are three places to declare these variables and they all need to match (see Parameters section below).

Any subsystems can be masked so that, at the highest level, the Simulink model looks like this:
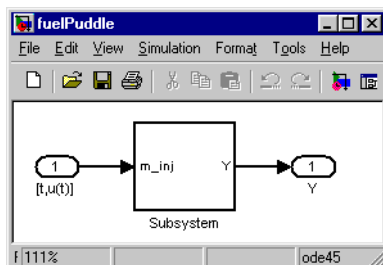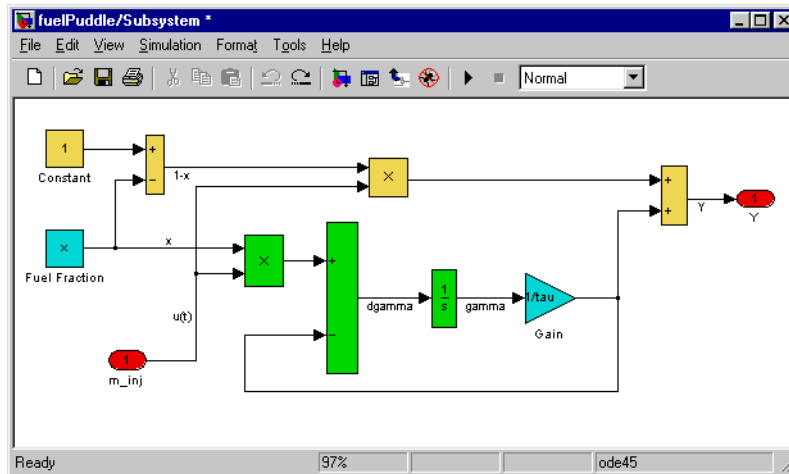
The block labels are not important. The model inputs are represented by Simulink inports at the top level. The output is represented by a Simulink output. Only one output is supported in MBC.

The model returns a single vector output (here labeled "Y").

Parameters

Inside the subsystem, some blocks depend on parameters that have not been quantified. In fuelPuddle.mdl there is a constant block whose value is 'x' and a gain block with the value '1/tau'.

These are the parameters we will be fitting by regression using MBC. The following setup needs to be followed:

1 Having masked the subsystem, you need to pass the unknown parameters in to the subsystem. We need to pass them in from the workspace. This is done via a **Block Parameters** dialog. To create such a dialog, right-click the masked subsystem and select **Edit mask** to see the **Mask Editor**. Use the **Add** button to add fields. Here two parameters are required . The **Variable** field must match the name of the corresponding parameter in the subsystem. Having declared the necessary parameters (all the unknowns of the subsystem), click **OK** to close and save.
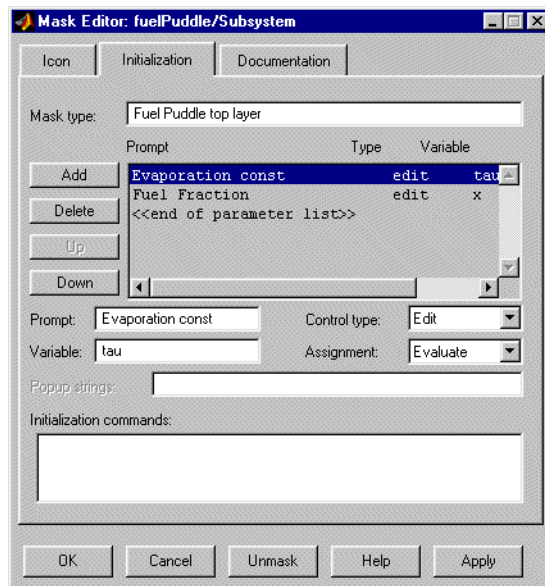
2 On double-clicking the subsystem mask, the **Block Parameters** dialog appears with the Prompt and variable names declared in the **Mask Editor**. By default the variable names appear in the edit boxes, and these should not be changed. It is these names that the Simulink model tries to find in the workspace.
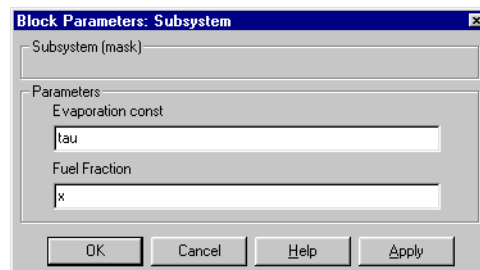
3 The same variable names must also be used in the template M-file that we shall discuss later. For this example the template M-file is fuelPuddle.m.

Testing the Simulink Model

Open your Simulink model and clear the MATLAB workspace.

Define the masked block parameters in the workspace. Here, for example, you could enter at the command line

```
tau = 0.5;
x = 0.3;
```

In the subsystem, connect sources to the input factors (for example, use a pulse generator at the input) and run the model (click Play). If the model is taking parameters from the workspace it should run okay.

Clear the workspace again and, without defining tau and x, try to run the model; it should fail to run.
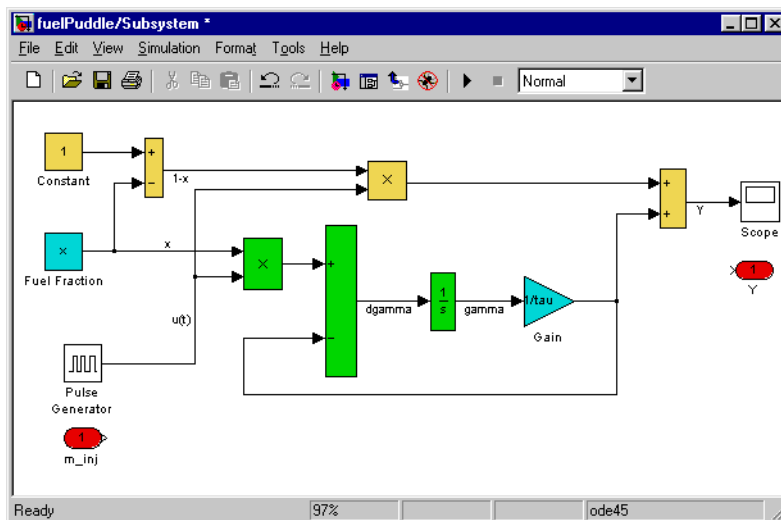
Creating Transient Models

You can create a transient model by typing the following at the command line:

```
m = xregtransient
```

You will see the following output at the command line:

```
m = functemplate([X1,X2],b=[0.5,0.5])
```

The default transient model is created. This model can be evaluated and fitted, but remember it uses the default Simulink model. In order to create a transient model that uses your own Simulink model, you must first define a template file for your Simulink model and this template file must have the same name as your Simulink model. The template file for `fuelPuddle.mdl` is therefore called `fuelPuddle.m`.

Template File

Note that the template file has a common layout for transient models and for user-defined models. It is not compulsory to modify every section. It is not a problem if you don't want to use several sections of the template file. The sections not actively read must have sensible defaults, so if you do not wish to use a section, leave it unmodified and it will be fine.

Open the file

```
<MATLAB root>\toolbox\mbc\mbcmodel\@xregtransient\fuelPuddle.m
```

The fast `eval` at the top of the file is not used for transient models and can be left as is.

The next section of commented code gives a summary of the functions in the template file. The function definitions follow this.

Subfunctions

Those subfunctions that must be edited are as follows:

```
function vars= i_simvars(U,b,varargin);
vars = {'tau','x'};
```

These are the parameters of the Simulink model that will be fitted.

This subfunction must return a cell array of strings. These are the parameter names that the Simulink model requires from the workspace. These strings *must* match the parameter names declared in the Simulink model (see "Parameters" on page 8-69).

```
function [vars,vals]= i_simconstants(U,b,varargin);
vars = {};
vals = [];
```

If your Simulink model requires constant parameters to be defined you do so here. These are constant parameters required by the Simulink model from the workspace. They are not fitted. These parameters must be the same as those in the Simulink model and all names must match. Here `fuelPuddle` requires no such parameters, and hence we return an empty call array and empty matrix.

```
function [ic]= i_initcond(U,b,X);
ic=[];
```

Initial conditions (for the integrators) are based on the current parameters, and inputs could be calculated here.

```
function n= i_numparams(U,b,varargin);
n= 2;
```

This is the number of fitted parameters. For `fuelPuddle` we have two parameters, `x` and `tau`.

```
function n= i_nfactors(U,b,t,u);
n= 2;
```

This is the number of input factors, including time. For `fuelPuddle` we input `X` = `[t, u(t)]` and hence the number of input factors is 2.

```
function [param,OK]= i_initial(U,b,X,Y)
param= [.5 .5]';
OK=1;
```

This subfunction returns a column vector of initial values for the parameters that are to be fitted. You could do some calculations here to estimate good initial values from the X, Y data. The initial values can be defined to be data dependent, hence there is a flag to signal if the data is not worth fitting. Here we simply define some default initial values for `x` and `tau`.

Optional Subfunctions

The remaining subfunctions need not be edited unless they are required. The comments in the code describe the role of each function. Mostly these functions are used when you are creating a template for user-defined models. There is only one of these subfunctions relevant here.

```
function c= i_labels(U,b)
b= {'\tau','x'};
```

These labels are used on plots and so on. You can use LaTeX notation, and it is correctly formatted.

'Checking In' To MBC

Having created a Simulink model and the corresponding template file, you must save each somewhere on the path. One place to put these files is

```
<MATLAB root>\work
```

Note that you must create the folder @xregtransient somewhere on the path (such as <MATLAB root>\work) and save your template.m file inside this new folder.

To ensure that the transient model you have defined provides an interface that allows MBC to evaluate and fit it, we *check in* the model. If this procedure succeeds, the model is registered by MBC and is thereafter available for fitting at the local level whenever appropriate input data is being used.

Check In. At the command line, with both template file and Simulink models on the path, create a model and some input data, then call checkin. The procedure follows.

1 Create some appropriate input data. The data is not important; it is to check if the model can be evaluated.

```
t = [0:0.1:4*pi]'; u = sin(t); %% creates a sine wave input factor
X = [t,u];
```

2 Create a transient model using the template/Simulink model by typing the following at the command line

```
m = xregtransient('name','MyTransient');
```

where MyTransient is the name you choose for your new transient model. For fuelPuddle you would type

```
m = xregtransient('name','fuelPuddle');
```

3 Call `checkin` with the transient model, its name, and the data.

```
checkin(m, 'MyTransient', X);
```

Note that the string you use for your new transient model name is the string that will appear in the **Model Setup** dialog when you set up transient models using the appropriate number of inputs.

When the check in is successful, this creates some command-line output, and a figure appears with the model name and variable names displayed over graphs of input and model evaluation output. The final command line output (if `checkin` is called with a semicolon as above) is

```
Model successfully registered for MBC
```

A figure is displayed for the model. Shown is the `fuelPuddle` example.
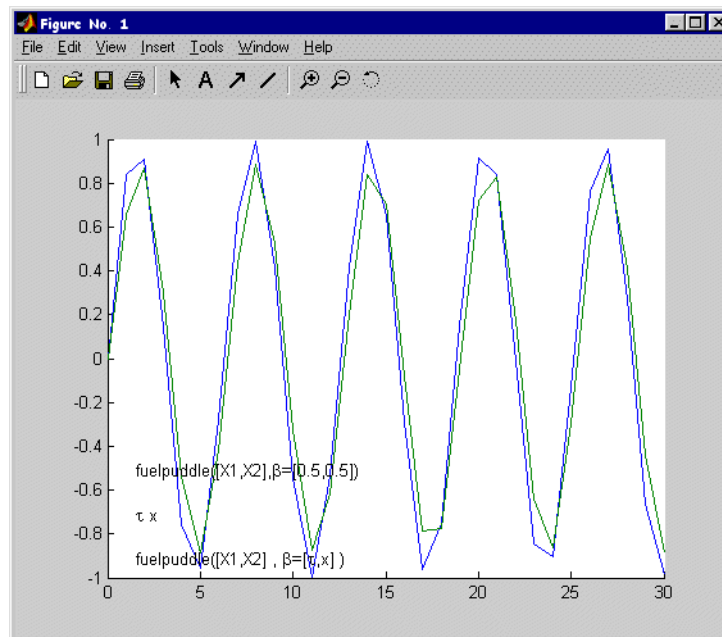
Using and Removing Transient Models

We provide the fuelPuddle model already checked in for you to use. Start up MBC modeling and load data that has the necessary format to evaluate this transient model — two local inputs.

Set up a **Test Plan** with two local input factors (the same number of input factors required by the fuelPuddle model). The **Local Model Setup** now offers the following options:
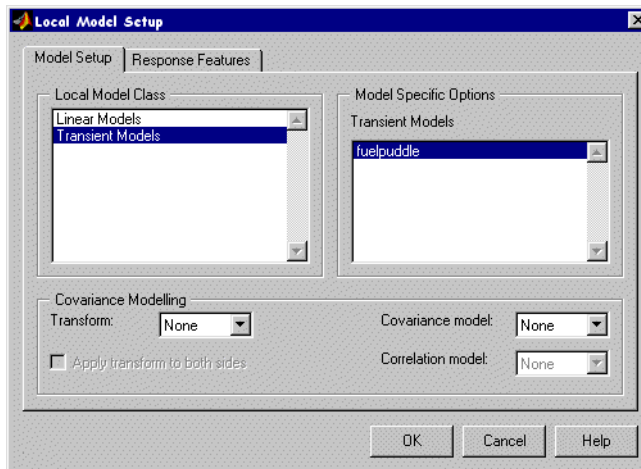
Select the fuelpuddle model and click **OK**.

On building a response model with fuelPuddle as the local model, MBC fits the two parameters tau and x across all the tests.

Removing Checked In Models

If you wish to remove checked in models for some reason, close the toolbox and type the following at the command line:

```
mbcdeleteprefs
```

When you restart the toolbox you will see that this removes all checked in transient models. Note that this will also delete all your saved preferences for file locations.

Data Loading Application Programming Interface

Data Loading API Specification

You can use the data loading API (application programming interface) to write your own data loading function, plug these functions into the toolbox, and subsequently use data loaded by these functions within the toolbox. To allow this, there are several stages that need to be followed as described below. For an example, see xregReadConcerto.m (in the **mbctools** directory).

Data Function Prototype

A function to successfully load data has the following prototype:

```
[OK, msg, out] = dataLoadingFcn(filename, protoOut)
```

Input Arguments

filename is the full path to the file to be loaded.

protoOut is an empty structure with the fields expected in the return argument out. This allows the data loading API to be easily extended without the need for data loading functions to change when MBC changes.

Output Arguments

The first return argument, OK, allows the function to signal that it has successfully loaded the data file. A value of 1 signals success, and 0 failure. If the function fails, it can return a message, msg, to indicate the reason for the failure. This message is displayed in a warning dialog box if the function fails. If the function is successful, the return argument out contains the data necessary for MBC.

out.varNames is a cell array of strings that hold the names of the variables in the data (1 x n or n x 1).

out.varUnits is a cell array of strings that hold the units associated with the variables in varNames (1 x n or n x 1). This array can be empty, in which case no units are defined.

out.data is an array that holds the values of the variables (m x n).

out.comment is an optional string holding comment information about the data.

Data Function Check In

Once you have written the function, you need to check it into the MBC Toolbox, using the `xregCheckinDataLoadingFcn` function. This function has the following prototype:

```
OK= xregCheckinDataLoadingFcn(fcn, filterSpec, fileType)
```

`fcn` is either a function handle or a string that calls the data loading function via `feval`. This function must be on the MATLAB path.

`filterSpec` is a `1 x 2` element cell array that contains the extensions that this function loads and the descriptions of those files. This cell array is used in the `uigetfile` function, for example, `{'*.m;*.fig;*.mat;*.mdl', 'All MATLAB Files'}`. MBC attempts to decide automatically which sort of file is loaded, based on the extension. In the case of duplicate extensions, the first in the list is selected; however, it is always possible to override the automatic selection with a user selection. You will see a warning message if there is any ambiguity.

`fileType` is a string that describes the file type, for example, 'MATLAB file' or 'Excel file'.

9

Radial Basis Functions

This section discusses the following topics:

Guide to Radial Basis Functions for Model Building

A radial basis function has the form

$$z(x) = \Phi(\|x - \mu\|)$$

where x is a n-dimensional vector, μ is an n-dimensional vector called the center of the radial basis function, $\|.\|$ denotes Euclidean distance, and Φ is a univariate function, defined for positive input values, that we shall refer to as the profile function.

The model is built up as a linear combination of N radial basis functions with N distinct centers. Given an input vector x, the output of the RBF network is the activity vector $\hat{y}$ given by

$$\hat{y}(x) = \sum_{j=1}^{N} \beta_j z_j(x),$$

where β_j is the weight associated with the jth radial basis function, centered at μ_j, and $z_j = \Phi(\|x - \mu_j\|)$. The output $\hat{y}$ approximates a target set of values denoted by y.

A variety of radial basis functions are available in MBC, each characterized by the form of Φ. All of the radial basis functions also have an associated width parameter σ, which is related to the spread of the function around its center. Selecting the box in the model setup provides a default setting for the width. The default width is the average over the centers of the distance of each center to its nearest neighbor. This is a heuristic given in Hassoun (see "References" on page 9-29) for Gaussians, but it is only a rough guide that provides a starting point for the width selection algorithm.

Another parameter associated with the radial basis functions is the regularization parameter λ. This (usually small) positive parameter is used in most of the fitting algorithms. The parameter λ penalizes large weights, which tends to produce smoother approximations of y and to reduce the tendency of the network to overfit (that is, to fit the target values y well, but to have poor predictive capability).

The following sections explain the different parameters for the radial basis functions available in the Model-Based Calibration Toolbox, and how to use them for modeling.

Types of Radial Basis Functions

Within the model setup, you can choose which RBF kernel to use. Kernels are the types of RBF (multiquadric, gaussian, thinplate, and so on).

Gaussian

This is the radial basis function most commonly used in the neural network community. Its profile function is

$$\Phi(r) = e^{(-r^2/\sigma^2)}$$

This leads to the radial basis function $z(x) = \exp\left(\dfrac{\|x-\mu\|^2}{\sigma^2}\right)$.

In this case, the width parameter is the same as the standard deviation of the gaussian function.

Gaussian with width = 0.5
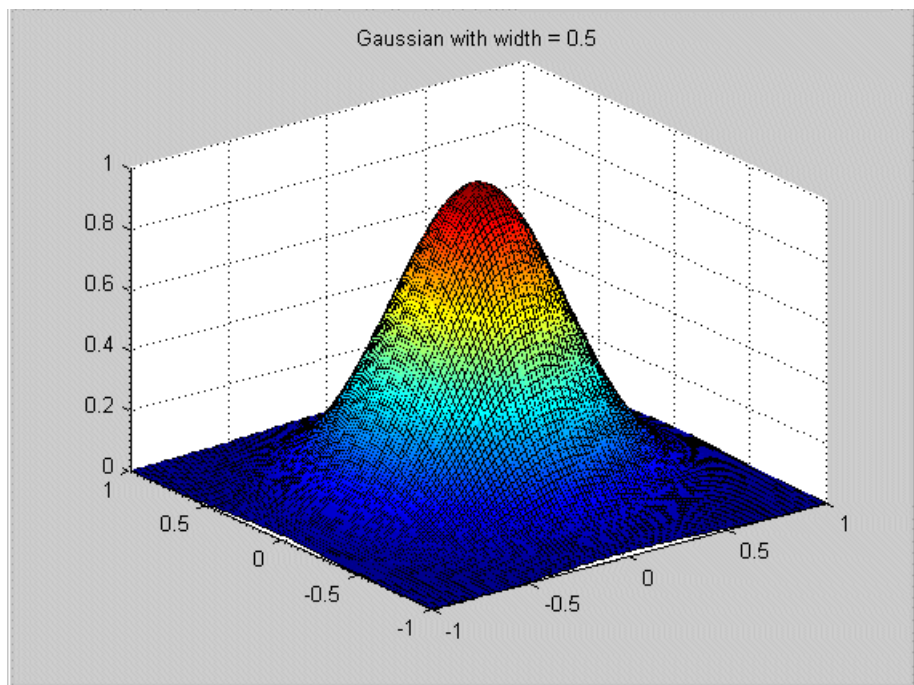
Thin-Plate Spline

This radial basis function is an example of a smoothing spline, as popularized by Grace Wahba (http://www.stat.wisc.edu/~wahba/). They are usually supplemented by low-order polynomial terms. Its profile function is

$$\Phi(r) = (r/\sigma)^2 \log(r/\sigma)$$
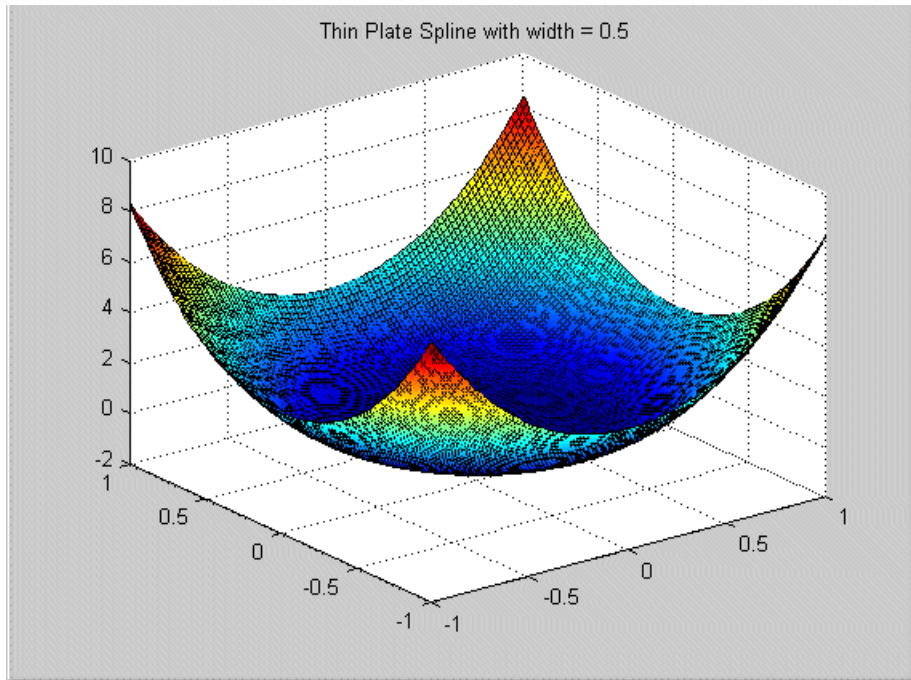
Logistic Basis Function

These radial basis functions are mentioned in Hassoun (see "References" on page 9-29). They have the profile function

$$\Phi(r) = \frac{1}{1 + \exp(\frac{r}{\sigma})}$$
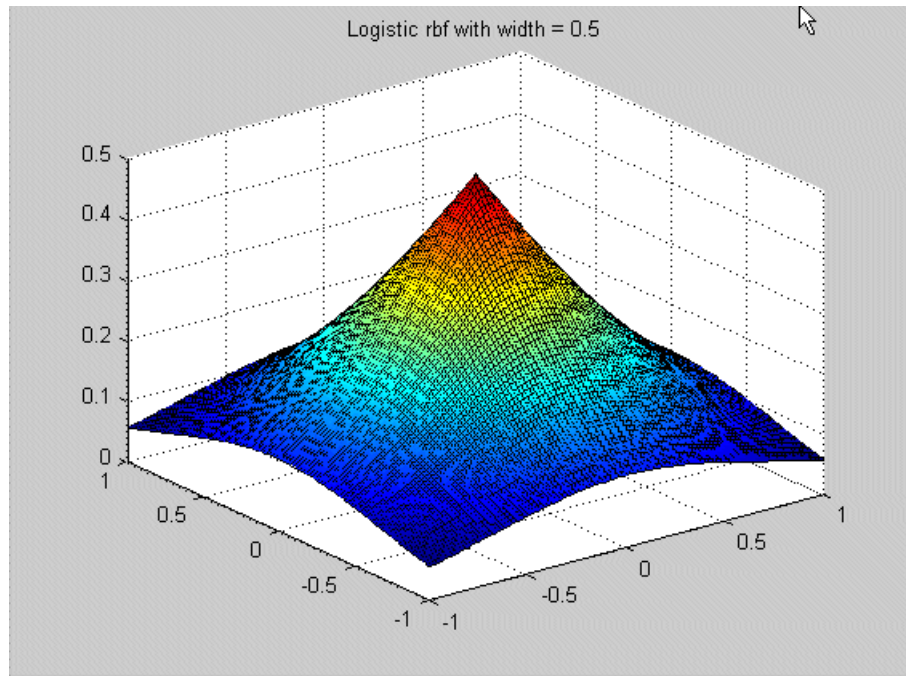
Wendland's Compactly Supported Function

These form a family of radial basis functions that have a piecewise polynomial profile function and compact support [Wendland, see "References" on page 9-29]. The member of the family to choose depends on the dimension of the space (n) from which the data is drawn and the desired amount of continuity of the polynomials.

Dimension	Continuity	Profile
n=1	0	$\Phi(r) = (1-r)_+$
	2	$\Phi(r) = (1-r)_+^3 (3r+1)$
	4	$\Phi(r) = (1-r)_+^5 (8r^2 + 5r + 1)$
n=3	0	$\Phi(r) = (1-r)_+^2$
	2	$\Phi(r) = (1-r)_+^4 (4r+1)$
	4	$\Phi(r) = (1-r)_+^6 (35r^2 + 18r + 3)$
n=5	0	$\Phi(r) = (1-r)_+^3$
	2	$\Phi(r) = (1-r)_+^5 (5r+1)$
	4	$\Phi(r) = (1-r)_+^7 (16r^2 + 7r + 1)$

We have used the notation $a_+ := \begin{cases} a, \, a > 0 \\ 0, \, a \le 0 \end{cases}$ for the positive part of a.

When n is even, the radial basis function corresponding to dimension n+1 is used.

Note that each of the radial basis functions is nonzero when r is in [0,1]. It is possible to change the support to be [0,σ] by replacing r by r/σ in the preceding formula. The parameter σ is still referred to as the width of the radial basis function.

Similar formulas for the profile functions exist for n>5, and for even continuity > 4. Wendland's functions are available up to an even continuity of 6, and in any space dimension n.

Notes on Use

- Better approximation properties are usually associated with higher continuity.
- For a given data set the width parameter for Wendland's functions should be larger than the width chosen for the Gaussian.

Compactly supported C6 rbf with width = 0.5
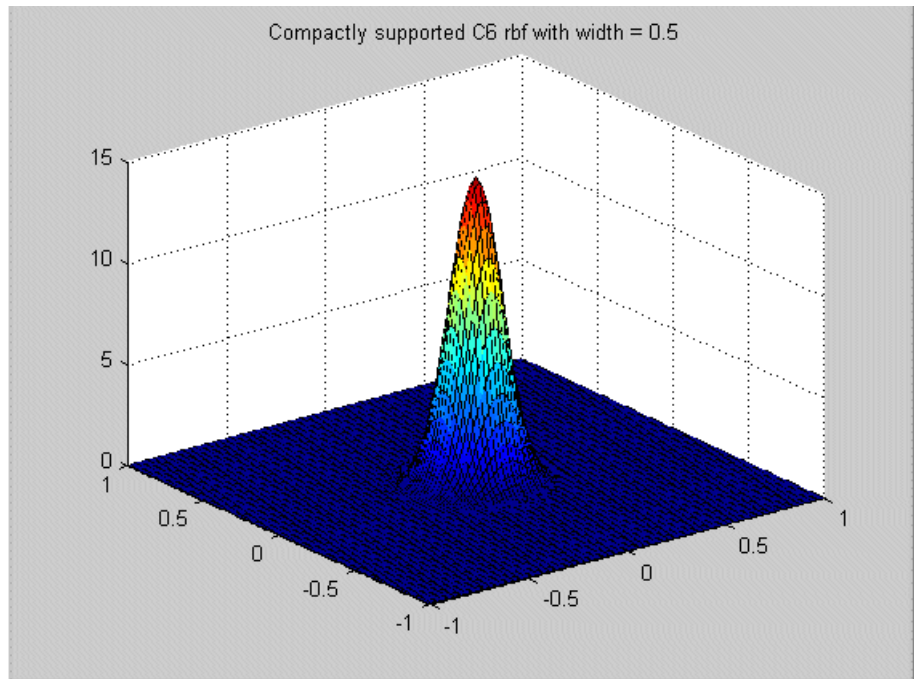
Multiquadrics

These are a popular tool for scattered data fitting. They have the profile function

$$\Phi(r) = \sqrt{r^2 + \sigma^2}$$
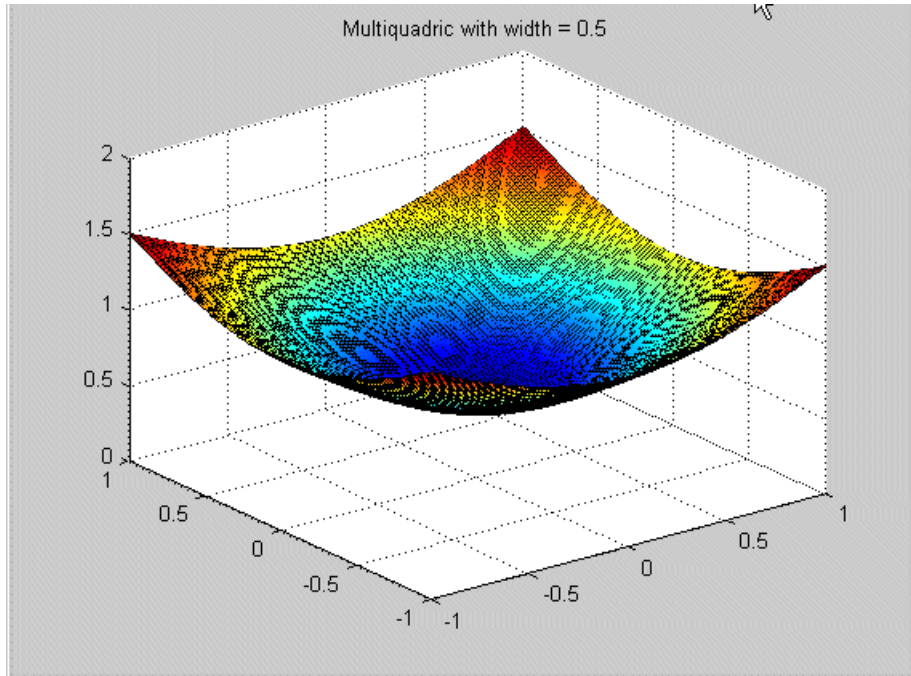
Reciprocal Multiquadrics

These have the profile function

$$\Phi(r) \;=\; 1/\sqrt{r^2 + \sigma^2}$$

Note that a width σ of zero is invalid.

Reciprocal Multiquadric with width = 0.5
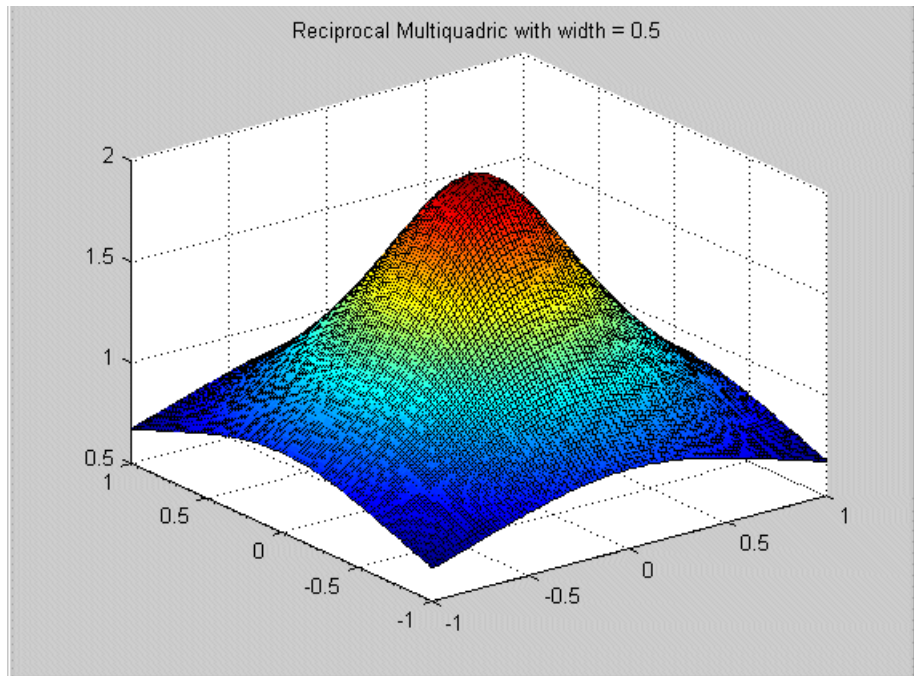
Fitting Routines

There are four characteristics of the RBF that need to be decided: weights, centers, width, and λ. Each of these can have significant impact on the quality of the resulting fit, and good values for each of them need to be determined. The weights are always determined by specifying the centers, width, and λ, and then solving an appropriate linear system of equations. However, the problem of determining good centers, width, and λ in the first place is far from simple, and is complicated by the strong dependencies among the parameters. For example, the optimal λ varies considerably as the width parameter changes. A global search over all possible center locations, width, and λ is computationally prohibitive in all but the simplest of situations.

To try to combat this problem, the fitting routines come in three different levels.

At the lowest level are the algorithms that choose appropriate centers for given values of width and λ. The centers are chosen one at a time from a candidate set (usually the set of data points or a subset of them). The resulting centers are therefore ranked in a rough order of importance.

At the middle level are the algorithms that choose appropriate values for λ and the centers, given a specified width.

At the top level are the algorithms that aim to find good values for each of the centers, width, and λ. These top-level algorithms test different width values. For each value of width, one of the middle-level algorithms is called that determines good centers and values for λ.

These algorithms and their fit parameters are described in the following sections:

1 "Center Selection Algorithms" on page 9-11

2 "Lambda Selection Algorithms" on page 9-14

3 "Width Selection Algorithms" on page 9-17

Center Selection Algorithms

Rols

This is the basic algorithm as described in Chen, Chng, and Alkadhimi [See "References" on page 9-29]. In **Rols** (Regularized Orthogonal Least Squares) the centers are chosen one at a time from a candidate set consisting of all the data points or a subset thereof. It picks new centers in a forward selection procedure. Starting from zero centers, at each step the center that reduces the regularized error the most is selected. At each step the regression matrix X is decomposed using the Gram-Schmidt algorithm into a product X = WB where W has orthogonal columns and B is upper triangular with ones on the diagonal. This is similar in nature to a QR decomposition. Regularized error is given by $e'e + \lambda g'g$ where g = Bw and e is the residual, given by $e = y - \hat{y}$. Minimizing regularized error makes the sum square error $e'e$ small, while at the same time not letting $g'g$ get too large. As g is related to the weights by g = Bw, this has the effect of keeping the weights under control and reducing overfit. The term $g'g$ rather than the sum of the squares of the weights $w'w$ is used to improve efficiency.

The algorithm terminates either when the maximum number of centers is reached, or adding new centers does not decrease the regularized error ratio significantly (controlled by a user-defined tolerance).

Fit Parameters

Maximum number of centers: The maximum number of centers that the algorithm can select. The default is the smaller of 25 centers or ¼ of the number of data points. The format is min(nObs/4, 25). You can enter a value (for example, entering 10 produces ten centers) or edit the existing formula (for example, (nObs/2, 25) produces half the number of data points or 25, whichever is smaller).

Percentage of data to be candidate centers: The percentage of the data points that should be used as candidate centers. This determines the subset of the data points that form the pool to select the centers from. The default is 100%, that is, to consider all the data points as possible new centers. This can be reduced to speed up the execution time.

Regularized error tolerance: Controls how many centers are selected before the algorithm stops. See Chen, Chng, and Alkadhimi ["References" on page 9-29] for details. This parameter should be a positive number between 0

and 1. Larger tolerances mean that fewer centers are selected. The default is 0.0001. If less than the maximum number of centers is being chosen, and you want to force the selection of the maximum number, then reduce the tolerance to epsilon (eps).

RedErr

RedErr stands for Reduced Error. This algorithm also starts from zero centers, and selects centers in a forward selection procedure. The algorithm finds (among the data points not yet selected) the data point with the largest residual, and chooses that data point as the next center. This process is repeated until the maximum number of centers is reached.

Fit Parameters

Only has **Number of centers**.

WiggleCenters

This algorithm is based on a heuristic that you should put more centers in a region where there is more variation in the residual. For each data point, a set of neighbors is identified as the data points within a distance of sqrt(nf) divided by the maximum number of centers, where nf is the number of factors. The average residuals within the set of neighbors is computed, then the amount of wiggle of the residual in the region of that data point is defined to be the sum of the squares of the differences between the residual at each neighbor and the average residuals of the neighbors. The data point with the most wiggle is selected to be the next center.

Fit Parameters

Almost as in the **Rols** algorithm, except no **Regularized error**.

CenterExchange

This algorithm takes a concept from optimal Design of Experiments and applies it to the center selection problem in radial basis functions. A candidate set of centers is generated by a Latin hypercube, a method that provides a quasi-uniform distribution of points. From this candidate set, n centers are chosen at random. This set is augmented by p new centers, then this set of n+p centers is reduced to n by iteratively removing the center that yields the best

PRESS statistic (as in stepwise). This process is repeated the number of times specified in **Number of augment/reduce cycles**.

CentreExchange and Tree Regression (see "Tree Regression" on page 9-19) are the only algorithms that permit centers that are not located at the data points. This means that you do not see centers on model plots. The CentreExchange algorithm has the potential to be more flexible than the other center selection algorithms that choose the centers to be a subset of the data points; however, it is significantly more time consuming and not recommended on larger problems.

Fit Parameters

Number of centers: The number of centers that will be chosen

Number of augment/reduce cycles: The number of times that the center set is augmented, then reduced

Number of centers to augment by: How many centers to augment by

Lambda Selection Algorithms

Lambda is the regularization parameter.

IterateRidge

For a specified width, this algorithm optimizes the regularization parameter with respect to the GCV criterion (generalized cross-validation; see the discussion under GCV criterion).

The initial centers either are selected by one of the low-level center selection algorithms or the previous choice of centers is used (see discussion under the parameter **Do not reselect centers**). You can select an initial start value for λ by testing an initial number of values for lambda (set by the user) that are equally spaced on a logarithmic scale between 10^{-10} and 10 and choosing the one with the best GCV score. This helps avoid falling into local minima on the GCV $- \lambda$ curve. The parameter λ is then iterated to try to minimize GCV using the formulas given in the GCV criterion section. The iteration stops when either the maximum number of updates is reached or the log10(GCV) value changes by less than the tolerance.

Fit Parameters

Center selection algorithm: The center selection algorithm to use.

Maximum number of updates: Maximum number of times that the update of λ is made. The default is 10.

Minimum change in log10(GCV): Tolerance. This defines the stopping criterion for iterating λ; the update stops when the difference in the log10(GCV) value is less than the tolerance. The default is 0.005.

Number of initial test values for lambda: Number of test values of λ to determine a starting value for λ. Setting this parameter to 0 means that the best λ so far is used.

Do not reselect centers for new width: This check box determines whether the centers are reselected for the new width value, and after each lambda update, or if the best centers to date are to be used. It is cheaper to keep the best centers found so far, and often this is sufficient, but it can cause premature convergence to a particular set of centers.

Display: When you select this check box, this algorithm plots the results of the algorithm. The starting point for λ is marked with a black circle. As λ is updated, the new values are plotted as red crosses connected with red lines. The best λ found is marked with a green asterisk.

If too many graphs are likely to be produced, because of the **Display** check box being activated here, a warning is generated, and you have the option to stop execution.

A lower bound of 10^{-12} is placed on λ, and an upper bound of 10.

IterateRols

For a specified width, this algorithm optimizes the regularization parameter in the **Rols** algorithm with respect to the GCV criterion. An initial fit and the centers are selected by **Rols** using the user-supplied λ. As in **IterateRidge**, you select an initial start value for λ by testing an initial number of start values for lambda that are equally spaced on a logarithmic scale between 10^{-10} and 10, and choosing the one with the best GCV score.

λ is then iterated to improve GCV. Each time that λ is updated, the center selection process is repeated. This means that **IterateRols** is much more computationally expensive than **IterateRidge**.

A lower bound of 10^{-12} is placed on λ, and an upper bound of 10.

Fit Parameters

Center selection algorithm: The center selection algorithm to use. For **IterateRols** the only center selection algorithm available is **Rols**.

Maximum number of updates: The same as for **IterateRidge**.

Minimum change in log10(GCV): The same as for **IterateRidge**.

Number of initial test values for lambda: The same as for **IterateRidge**.

Do not reselect centers for new width: This check box determines whether the centers are reselected for the new width value or if the best centers to date are to be used.

Display: When you select this check box, this algorithm plots the results of the algorithm. The starting point for λ is marked with a black circle. As λ is updated, the new values are plotted as red crosses connected with red lines. The best λ found is marked with a green asterisk.

If too many graphs are likely to be produced, because of the **Display** check box being activated here, a warning is generated, and you have the option to stop execution.

StepItRols

This algorithm combines the center-selection and lambda-selection processes. Rather than waiting until all centers are selected before λ is updated (as with the other lambda-selection algorithms), this algorithm offers the ability to update λ after each center is selected. It is a forward selection algorithm that, like **Rols**, selects centers on the basis of regularized error reduction. The stopping criterion for **StepItRols** is on the basis of log10(GCV) changing by less than the tolerance more than a specified number of times in a row (given in the parameter **Maximum number of times log10(GCV) change is minimal**). Once the addition of centers has stopped, the intermediate fit with the smallest log10(GCV) is selected. This can involve removing some of the centers that entered late in the algorithm.

Fit Parameters

Maximum number of centers: As in the **Rols** algorithm.

Percentage of data to candidate centers: As in the **Rols** algorithm.

Number of centers to add before updating: How many centers are selected before iterating λ begins.

Minimum change in log10(GCV): Tolerance. It should be a positive number between 0 and 1. The default is 0.005.

Maximum number of times log10(GCV) change is minimal: Controls how many centers are selected before the algorithm stops. The default is 5. Left at the default, the center selection stops when the log10(GCV) values change by less than the tolerance five times in a row.

Width Selection Algorithms

TrialWidths

This routine tests several width values by trying different widths. A set of trial widths equally spaced between specified initial upper and lower bounds is selected. The width with the lowest value of log10(GCV) is selected. The area around the best width is then tested in more detail — this is referred to as a zoom. Specifically, the new range of trial widths is centered on the best width found at the previous range, and the length of the interval from which the widths are selected is reduced to 2/5 of the length of the interval at the previous zoom. Before the new set of trial widths is tested, the center selection is updated to reflect the best width and λ found so far. This can mean that the location of the optimum width changes between zooms because of the new center locations.

Fit Parameters

Lambda selection algorithm: Midlevel fit algorithm that you test with the various trial values of λ. The default is **IterateRidge**.

Number of trial widths in each zoom: Number of trials made at each zoom. The widths tested are equally spaced between the initial upper and lower bounds. Default is 10.

Number of zooms: Number of times you zoom in. Default is 5.

Initial lower bound on width: Lower bound on the width for the first zoom. Default is 0.01.

Initial upper bound on width: Upper bound on the width for the first zoom. Default is 20.

Display: If you select this check box, a stem plot of log10(GCV) against width is plotted. The best width is marked by a green asterisk.

WidPerDim

In the **WidPerDim** algorithm (Width Per Dimension), the radial basis
functions are generalized. Rather than having a single width parameter, a
different width in each input factor can be used; that is, the level curves are
elliptical rather than circular (or spherical, with more factors). The basis
functions are no longer radially symmetric.
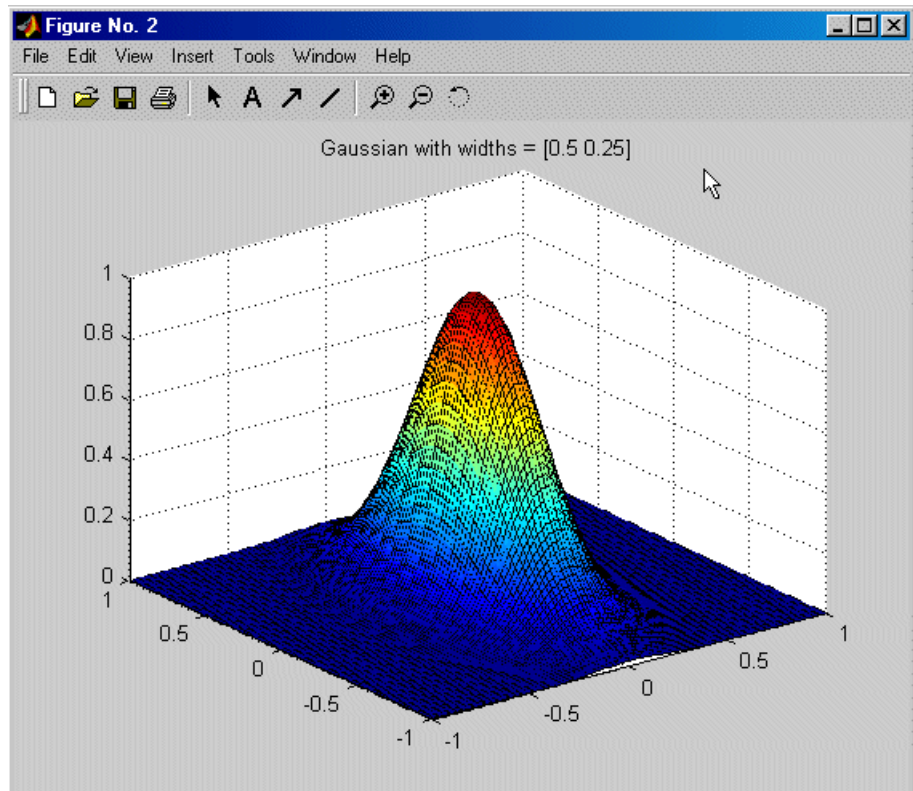
This can be especially helpful when the amount of variability varies
considerably in each input direction. This algorithm offers more flexibility than
TrialWidths, but is more computationally expensive.

An estimation of the time for the width per dimension algorithm is computed.
This is given as a number of time units (as it depends on the machine). A time
estimate of over 10 but less than 100 generates a warning. A time estimate of

over 100 might take a prohibitively long amount of time (probably over five minutes on most machines). You have the option to stop execution and change some of the parameters to reduce the run time.

Fit Parameters

As for the **TrialWidths** algorithm.

Tree Regression

There are three parts to the tree regression algorithm for RBFs:

- Tree building
- Alpha selection
- Center selection.

Tree Building. The tree regression algorithm builds a regression tree from the data and uses the nodes (or panels) of this tree to infer candidate centers and widths for the RBF. The root panel of the tree corresponds to a hypercube that contains all of the data points. This panel is divided into two child panels such that each child contains the same amount of variation, as much as is possible. The child panel with the most variation is then split in a similar way. This process continues until there are no panels left to split, i.e., no childless panel has more than the minimum number of data points, or until the maximum number of panels has been reached. Each panel in the tree corresponds to a candidate center and the size of the panel determines the width that goes with that vector.

The size of the child panels can be based solely on the size of the parent panel or can be determined by shrinking the child panel onto the data that it contains.

Once you have selected Radial Basis Function in the **Global Model Setup** dialog, you can choose Tree Regression from the **Width Selection Algorithm** drop-down menu.

Click **Advanced** to open the **Radial Basis Functions Options** dialog to reach settings such as maximum number of panels and minimum number of data points per panel. To shrink child panels to fit the data, select the check box **Shrink panels to data**.

Alpha Selection Algorithm. The size for the candidate widths are not taken directly from the panel sizes: we need to scale the panel sizes to get the corresponding widths. This scaling factor is called alpha. The same scaling factor needs to be applied to every panel in the tree and to determine the optimal value of alpha we use an alpha selection algorithm.

You can choose the parameter `Specify Alpha` to specify the exact value of alpha to use, or you can select `Trial Alpha`. `Trial Alpha` is very similar to the Trial Widths algorithm. The only difference is that the trial alpha algorithm can specify how to space the values to search. `Linear` is the same as used by trial widths but `Logarithmic` searches more values near the lower range.

Click **Advanced** to open the **Radial Basis Functions Options** dialog to reach further settings such as bounds on alpha, number of zooms and number of trial alphas. Here you can select the **Display** check box to see the progress of the algorithm and the values of alpha trailed.

Center Selection Algorithm. The tree building generates candidate centers and the alpha selection generates candidate widths for these centers. The center selection chooses which of those centers to use.

`Generic Center Selection` is a center selection algorithm that knows nothing about the tree structure to be used. It uses `Rols`, which is very fast way to choose centers and works in this case as well as the usual RBF cases. However, in this case the candidates for centers are not the data by the centers from the regression tree.

`Tree-based center selection` uses the regression tree. It is natural to use the regression tree to select centers because of the way it is built. In particular, the panel corresponding to the root node should be considered for selection before any of its children as it captures coarse detail while nodes at the leaves of the tree capture fine detail. This is what the `Tree-based center selection` does. You can also set the maximum number of centers.
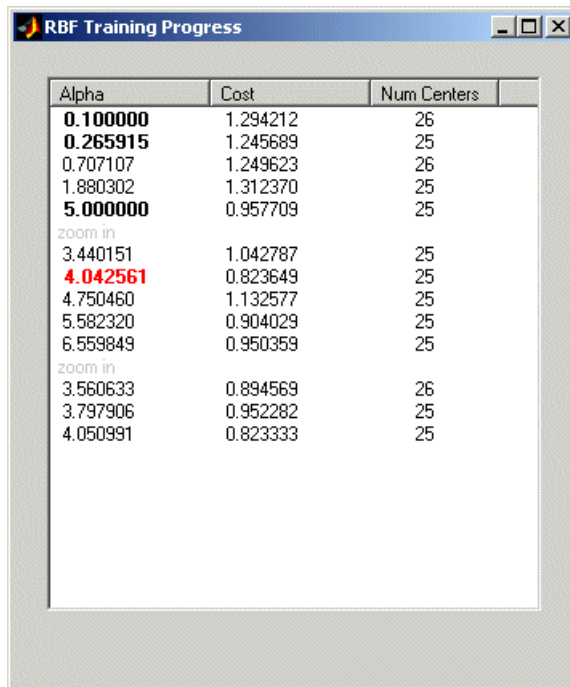
Click **Advanced** to open the **Radial Basis Functions Options** dialog to reach the **Model selection criteria** setting. The **Model selection criteria** determines what function should be used as a measure of how good a model is. This can be 'BIC' (Bayesian information criterion) or 'GCV' (generalized cross-validation). BIC is usually less susceptible to over fitting.

The Stepwise menu is the same for all RBFs, see "Global Model Class: Radial Basis Function" on page 7-49.

Tree Regression and CentreExchange are the only algorithms that permit centers that are not located at the data points. This means that you do not see centers on model plots.

If you leave the **Alpha selection algorithm** at the default, Trial Alpha, you will see a progress dialog when you click **OK** to begin modeling. An example is shown.

This is an example progress dialog of a Tree Regression RBF model fitting in progress. Here you can see each trial value of alpha with its calculated cost and the best number of centers with that value of alpha. The alpha value in red is the best so far. Alpha values no longer red, but in bold, are previous best values. You can then refine your model by zooming in on the best values for alpha and number of centers.

Reference: M. Orr, J. Hallam, K. Takezawa, A. Murray, S. Ninomiya, M. Oide, T. Leonard, "Combining Regression Trees and Radial Basis Function

Networks" , International Journal of Neural Systems, Vol. 10, No. 6 (2000) 453-465.

`http://www.anc.ed.ac.uk/~mjo/rbf.html`

Prune Functionality

You can use the Prune function to reduce the number of centers in a radial basis function network. This helps you decide how many centers are needed.

To use the Prune facility:

1 Select an RBF global model in the model tree.

2 Either click the ⬚ toolbar button or select the menu item **Model –> Utilities – > Prune**.

The **Number of Centers Selector** dialog appears.

The graphs show how the fit quality of the network builds up as more RBFs are added. It makes use of the fact that most of the center selection algorithms are greedy in nature, and so the order in which the centers were selected roughly reflects the order of importance of the basis functions.

The default fit criteria are the logarithms of PRESS, GCV, RMSE, and Weighted PRESS. Additional options are determined by your selections in Summary Statistics. See "Model Menu (Global Level)" on page 7-107. Weighted PRESS penalizes having more centers, and choosing number of centers to minimize weighted PRESS is often a good option.
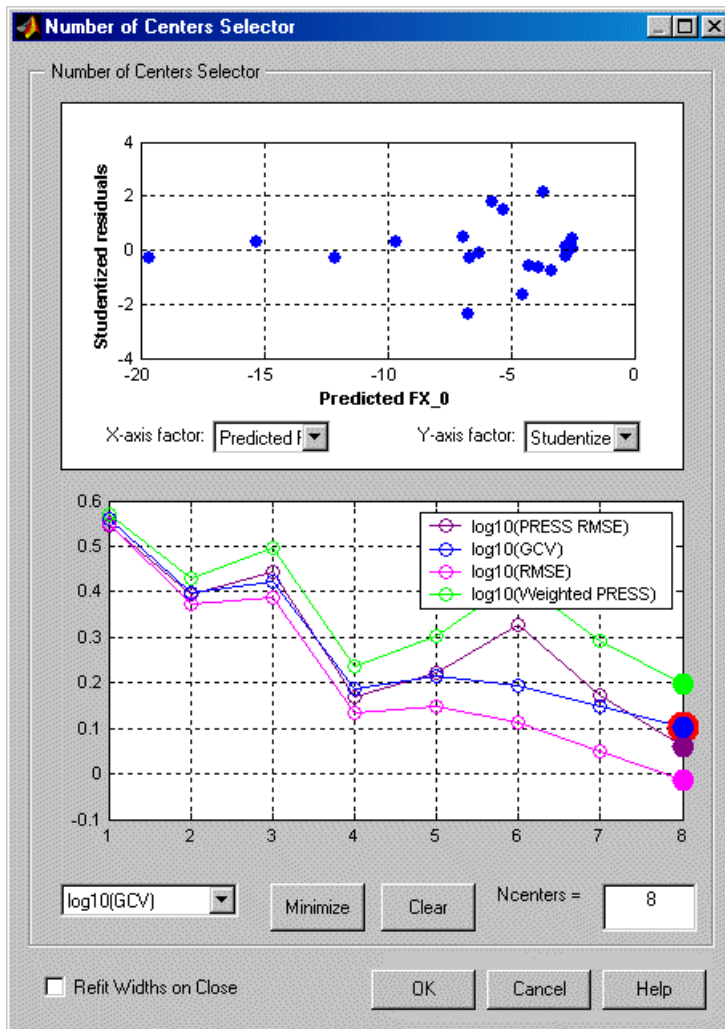
All four criteria in this typical example indicate the same minimum at eight centers.

If the graphs all decrease, as in the preceding example, this suggests that the maximum number of centers is too small, and the number of centers should be increased.

Clicking the **Minimize** button selects the number of centers that minimizes the criterion selected in the drop-down menu on the left. It is good if this value also minimizes all the other criteria. The **Clear** button returns to the previous selection.

Note that reducing the number of centers using **Prune** only refits the linear parameters (RBF weights). The nonlinear parameters (center locations, width, and lambda) are not adjusted. You can perform a cheap width refit on exiting the dialog by selecting the **Refit widths on close** check box. If a network has been pruned significantly, you should use the Update Model Fit toolbar button. This performs a full refit of all the parameters.

Statistics

Let A be the matrix such that the weights are given by $\beta = A^{-1}X'y$ where X is the regression matrix. The form of A varies depending on the basic fit algorithm employed.

In the case of ordinary least squares, we have A = X'X.

For ridge regression (with regularization parameter λ), A is given by

$$A = X'X + \lambda I$$

Next is the Rols algorithm. During the Rols algorithm X is decomposed using the Gram-Schmidt algorithm to give X = WB, where W has orthogonal columns and B is upper triangular. The corresponding matrix A for Rols is then $A = X'X + \lambda B'B$.

The matrix $H := XA^{-1}X'$ is called the hat matrix, and the leverage of the ith data point h_i is given by the ith diagonal element of H. All the statistics derived from the hat matrix, for example, PRESS, studentized residuals, confidence intervals, and Cook's distance, are computed using the hat matrix appropriate to the particular fit algorithm.

Similarly PEV, given in the "Definitions" on page 8-4 as

$$PEV(x) = \mathrm{var}\, y(\hat{y}) = x(X^TX)^{-1}x^T MSE$$

becomes

$$PEV(x) = \mathrm{var}\, y(\hat{y}) = xA^{-1}x^T MSE$$

PEV is computed using the form of A appropriate to the particular fit algorithm (ordinary least squares, ridge or rols).

GCV Criterion

Generalized cross-validation(GCV) is a measure of the goodness of fit of a model to the data that is minimized when the residuals are small, but not so small that the network has overfitted the data. It is easy to compute, and networks with small GCV values should have good predictive capability. It is related to the PRESS statistic.

The definition of GCV is given by Orr (4, see "References" on page 9-29).

$$GCV = \frac{N(y'P^2y)}{(trace(P))^2}$$

where y is the target vector, N is the number of observations, and P is the projection matrix, given by I - $XA^{-1}X^T$. See "Statistics" on page 9-26 for definition of A.

An important feature of using GCV as a criterion for determining the optimal network in our fit algorithms is the existence of update formulas for the regularization parameter λ. These update formulas are obtained by differentiating GCV with respect to λ and setting the result to zero. That is, they are based on gradient-descent.

This gives the general equation (from Orr, 6, "References" on page 9-29)

$$y'P\frac{\partial(Py)}{\partial\lambda}trace(P) = (y'P^2y)\frac{\partial(trace(P))}{\partial\lambda}$$

We now specialize these formulas to the case of ridge regression and to the Rols algorithm.

GCV for Ridge Regression

It is shown in Orr (4), and stated in Orr (5, see "References" on page 9-29) that for the case of ridge regression GCV can be written as

$$GCV = \frac{N(e'e)}{(N-p)^2}$$

where p is the "effective number of parameters" that is given by

$$p = NumTerms - \lambda\,trace(A^{-1})$$

where NumTerms is the number of terms included in the model.

For RBFs, 'p' is the effective number of parameters, that is, the number of terms minus an adjustment to take into account the smoothing effect of lambda in the fitting algorithm. When lambda = 0, the effective number of parameters is the same as the number of terms.

The formula for updating λ is given by

$$\lambda = \frac{\eta}{N-p}\frac{(e'e)}{(\beta'A^{-1}\beta)^2} \text{ where } \eta = tr(A^{-1} - \lambda A^{-2})$$

In practice, the preceding formulas are not used explicitly in Orr (5, see "References" on page 9-29). Instead a singular value decomposition of X is made, and the formulas are rewritten in terms of the eigenvalues and eigenvectors of the matrix XX'. This avoids taking the inverse of the matrix A, and it can be used to cheaply compute GCV for many values of λ. See "Statistics" on page 9-26 for definition of A.

GCV for Rols

In the case of Rols, the components for the formula

$$\text{GCV} = \frac{N(y'P^2 y)}{(trace(P))^2}$$

are computed using the formulas given in Orr [6; see "References" on page 9-29]. Recall that the regression matrix is factored during the Rols algorithm into the product X = WB. Let w_j denote the jth column of W, then we have

$$y'P^2 y = y'y - \sum_{j=1}^{N} \frac{(2\lambda + w_j'w_j)(y'w_j)^2}{(\lambda + w_j'w_j)^2}$$

and the "effective number of parameters" is given by

$$Trace(P) = NumTerms - \sum_{j=1}^{N} \frac{w_j'w_j}{(\lambda + w_j'w_j)}$$

This is equivalent to 'p' (the effective number of parameters) defined in "GCV for Ridge Regression" on page 9-27.

The reestimation formula for λ is given by

$$\lambda = \frac{\eta}{Trace(P)} \frac{y'P^2 y}{(\beta'A^{-1}\beta)^2} \text{ where additionally}$$

$$\eta = \sum_{j=1}^{N} \frac{w_j'w_j}{(\lambda + w_j'w_j)^2} \text{ and } \beta'A^{-1}\beta = \sum_{j=1}^{N} \frac{(y'w_j)^2}{(\lambda + w_j'w_j)^3}$$

Note that these formulas for Rols do not require the explicit inversion of A. See "Statistics" on page 9-26 for definition of A.

References

1 Chen, S., Chng, E.S., Alkadhimi, Regularized Orthogonal Least Squares Algorithm for Constructing Radial Basis Function Networks, Int J. Control, 1996, Vol. 64, No. 5, pp. 829-837.

2 Hassoun, M., Fundamentals of Artificial Neural Networks, MIT, 1995.

3 Orr, M., Introduction to Radial Basis Function Networks, available from `http://www.anc.ed.ac.uk/~mjo/rbf.html`.

4 Orr, M., Optimizing the Widths of Radial Basis Functions, available from `http://www.anc.ed.ac.uk/~mjo/rbf.html`.

5 Orr, M., Regularisation in the Selection of Radial Basis Function Centers, available from `http://www.anc.ed.ac.uk/~mjo/rbf.html`.

6 Wendland, H., Piecewise Polynomials, Positive Definite and Compactly Supported Radial Basis Functions of Minimal Degree, Advances in Computational Mathematics 4 (1995), pp. 389-396.

Hybrid Radial Basis Functions

Hybrid RBFs combine a radial basis function model with more standard linear models such as polynomials or hybrid splines. The two parts are added together to form the overall model. This approach offers the ability to combine a priori knowledge, such as the expectation of quadratic behavior in one of the variables, with the nonparametric nature of RBFs.

The model setup GUI for hybrid RBFs has a top **Set Up** button, where you can set the fitting algorithm and options. The interface also has two tabs, one to specify the radial basis function part, and one for the linear model part.

Width Selection Algorithm: TrialWidths

This is the same algorithm as is used in ordinary RBFs, that is, a guided search for the best width parameter.

Lambda and Term Selection Algorithms: Interlace

This algorithm is a generalization of **StepItRols** for RBFs. The algorithm chooses radial basis functions and linear model terms in an interlaced way, rather than in two steps. At each step a forward search procedure is performed to select the radial basis function (with a center chosen from within the set of data points) or the linear model term (chosen from the ones specified in the linear model setup pane) that decreases the regularized error the most. This process continues until the maximum number of terms is chosen. The first few terms are added using the stored value of lambda. After **StartLamUpdate** terms have been added, lambda is iterated after each center is added to improve GCV.

The fit options for this algorithm are as follows:

- **Maximum number of terms:** Maximum number of terms that will be chosen. The default is a quarter of the data points, or 25, whichever is smaller.
- **Maximum number of centers:** Maximum number of terms that can be radial basis functions. The default is the same as the maximum number of terms.
- **Percentage of data to be candidate centers:** Percentage of the data points that are available to be chosen as centers. The default is 100% when the number of data points is <=200.

- **Number of terms to add before updating:** How many terms to add before updating lambda begins.

- **Minimum change in log10(GCV):** Tolerance.

- **Maximum no. times log10(GCV) change is minimal:** Number of steps in a row that the change in log10(GCV) can be less than the tolerance before the algorithm terminates.

Lambda and Term Selection Algorithms: TwoStep

This algorithm starts by fitting the linear model specified in the linear model pane, and then fits a radial basis function network to the residual. You can specify the linear model terms to include in the usual way using the term selector. If desired, you can activate the stepwise options. In this case, after the linear model part is fitted, some of the terms are automatically added or removed before the RBF part is fitted. You can choose the algorithm and options that are used to fit the nonlinear parameters of the RBF by clicking the **Set Up** button in the RBF training options.

Tips for Modeling with Radial Basis Functions

Plan of Attack

Determine which parameters have the most impact on the fit by following these steps:

1 Fit the default RBF. Remove any obvious outliers.

2 Get a rough idea of how many RBFs are going to be needed. If a center coincides with a data point, it is marked with a magenta asterisk on the Predicted/Observed plot. You can view the location of the centers in graphical and table format by using the (View Centers) toolbar button. If you remove an outlier which coincided with a center (marked with an asterisk), refit by clicking Update Fit in the toolbar.

3 Try with more than one kernel. You can alter the parameters in the fit by clicking the **Set Up** button in the **Model Selection** dialog.

4 Decide on the main width selection algorithm. Try with both **TrialWidths** and **WidPerDim** algorithms.

5 Determine which types of kernel look most hopeful.

6 Narrow the corresponding width range to search over.

7 Decide on the center selection algorithm.

8 Decide on the lambda-selection algorithm.

9 Try changing the parameters in the algorithms.

10 If any points appear to be possible outliers, try fitting the model both with and without those points.

If at any stage you decide on a change that has a big impact (such as removal of an outlier), then you should repeat the previous steps to determine if this would affect the path you have chosen.

See "Fitting Routines" on page 9-10 for details on all the fit parameters.

The Model Browser has a quick option for comparing all the different RBF kernels.

1 After fitting the default RBF, select the RBF global model in the model tree.

2 Click the ⬚ (Build Models) toolbar icon.

3 Select the **RBF Kernels** icon in the **Build Models** dialog that appears and click **OK**.

One of each kernel is built as a selection of child nodes of the current RBF model.

How Many RBFs to Use

- The main parameter that you must adjust in order to get a good fit with an RBF is the maximum number of centers. This is a parameter of the center selection algorithm, and is the maximum number of centers/RBFs that is chosen.

- Usually the maximum number of centers is the number of RBFs that are actually selected. However, sometimes fewer RBFs are chosen because the (regularized) error has fallen below the tolerance before the maximum was reached.

- You should use a number of RBFs that is significantly less than the number of data points, otherwise there are not enough degrees of freedom in the error to estimate the predictive quality of the model. That is, you cannot tell if the model is useful if you use too many RBFs. We would recommend an upper bound of 60% on the ratio of number of RBFs to number of data points. Having 80 centers when there are only 100 data points might seem to give a good value of PRESS, but when it comes to validation, it can sometimes become clear that the data has been overfitted, and the predictive capability is not as good as PRESS would suggest.

- One strategy for choosing the number of RBFs is to fit more centers than you think is needed (say 70 out of 100), then use the ✂ (Prune) toolbar button to reduce the number of centers in the model. After pruning the network, make a note of the reduced number of RBFs. Try fitting the model again with the maximum number of centers set to this reduced number. This

recalculates the values of the nonlinear parameters (width and lambda) to be optimal for the reduced number of RBFs.

- One strategy for the use of Stepwise is to use it to minimize PRESS as a final fine-tuning for the network, once pruning has been done. Whereas Prune only allows the last RBF introduced to be removed, Stepwise allows any RBF to be taken out.

- Do not focus solely on PRESS as a measure of goodness of fit, especially at large ratios of RBFs to data points. Take log10(GCV) into account also.

Width Selection Algorithms

- Try both **TrialWidths** and **WidPerDim**. The second algorithm offers more flexibility, but is more computationally expensive. View the width values in each direction to see if there is significant difference, to see whether it is worth focusing effort on elliptical basis functions (use the 🔍 View Model toolbar button).
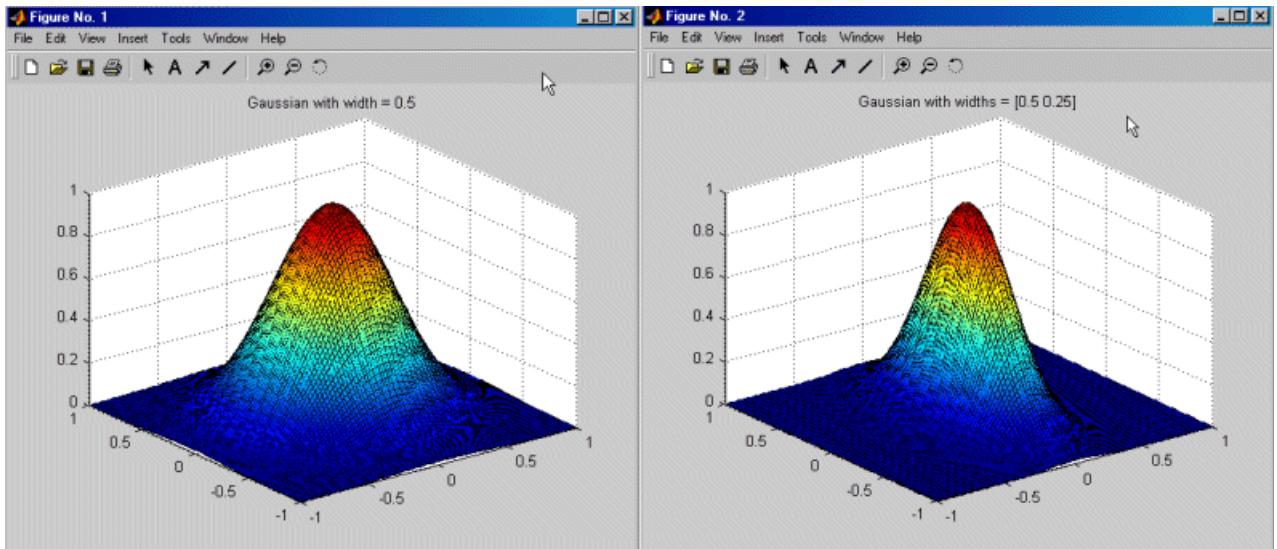
- If with a variety of basis functions the widths do not vary significantly between the dimensions, and the PRESS/GCV values are not significantly

improved using **WidPerDim** over **TrialWidths**, then focus on **TrialWidths**, and just return to **WidPerDim** to fine-tune in the final stages.

- Turn the **Display** option on in **TrialWidths** to see the progress of the algorithm. Watch for alternative regions within the width range that have been prematurely neglected. The output log10(GCV) in the final zoom should be similar for each of the widths tried; that is, the output should be approximately flat. If this is not the case, try increasing the number of zooms.

- In **TrialWidths**, for each type of RBF, try to narrow the initial range of widths to search over. This might allow the number of zooms to be reduced.

Which RBF to Use

- It is hard to give rules of thumb on how to select the best RBF, as the best choice is highly data-dependent. The best guideline is to try all of them with both top-level algorithms (**TrialWidths** and **WidPerDim**) and with a sensible number of centers, compare the PRESS/GCV values, then focus on the ones that look most hopeful.

- If multiquadrics and thin-plate splines give poor results, it is worth trying them in combination with low-order polynomials as a hybrid spline. Try supplementing multiquadrics with a constant term and thin-plate splines with linear (order 1) terms. See "Hybrid Radial Basis Functions" on page 9-30.

- Watch out for conditioning problems with Gaussian kernels (say condition number $> 10^8$).

- Watch out for strange results with Wendland's functions when the ratio of the number of parameters to the number of observations is high. When these functions have a very small width, each basis function only contributes to the fit at one data point. This is because its support only encompasses the one basis function that is its center. The residuals will be zero at each of the data points chosen as a center, and large at the other data points. This scenario can indicate good RMSE values, but the predictive quality of the network will be poor.

Lambda Selection Algorithms

Lambda is the regularization parameter.

- **IterateRols** updates the centers after each update of lambda. This makes it more computationally intensive, but potentially leads to a better combination of lambda and centers.
- **StepItRols** is sensitive to the setting of **Number of centers to add before updating**. Switch the **Display** option on to view how log10(GCV) reduces as the number of centers builds up.
- Examine the plots produced from the lambda selection algorithm, ignoring the warning "An excessive number of plots will be produced." Would increasing the tolerance or the number of initial test values for lambda lead to a better choice of lambda?

Center Selection Algorithms

- On most problems, **Rols** seems to be the most effective.
- If less than the maximum number of centers are being chosen, and you want to force the selection of the maximum number, reduce the tolerance to epsilon (eps).
- **CenterExchange** is very expensive, and you should not use this on large problems. In this case, the other center selection algorithms that restrict the centers to be a subset of the data points might not offer sufficient flexibility.

General Parameter Fine-Tuning

- Try Stepwise after pruning, then update the model fit with the new maximum number of centers set to the number of terms left after Stepwise.
- Update the model fit after removal of outliers; use the toolbar button.

Hybrid RBFs

- Go to the linear part pane and specify the polynomial or spline terms that you expect to see in the model.

Fitting too many non-RBF terms is made evident by a large value of lambda, indicating that the underlying trends are being taken care of by the linear part. In this case, you should reset the starting value of lambda (to say 0.001) before the next fit.

How to Find RBF Model Formula

With any model you can use the View Model toolbar button or **View –>Model Definition** (or keyboard shortcut **CTRL**+V) to see the details of the current model. The **Model Viewer** dialog appears. Here for any RBF model you can see the kernel type, number of centers, width and regularization parameter.

However to specify the formula of an RBF model completely, you also need to give the locations of the centers, and the height of each basis function. The center location information is available in the "View Centers" dialog and the coefficients can be found in the "Stepwise" window. Note these values are all in coded units.

Model Browser
Case Studies

This section discusses the following topics:

Gasoline Example Study

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 "Design the Experiment" on page 10-4.

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 10-11.

3 Make statistical models based on the data.

    a "How Is a Two-Stage Model Constructed?" on page 10-14

    b "Building the Models" on page 10-18

    c "Model Selection" on page 10-20

    d "Adding a New Response Model" on page 10-22

    e "Boundary Modeling" on page 10-23

4 Export these models to the CAGE part of the toolbox in order to generate optimal calibration tables — see "Exporting the Models" on page 10-25.

This part of the case study covers the Model Browser section, involving design of experiment, data handling, and model construction and export. The CAGE browser section of the case study continues in the CAGE User's Guide, where you use the models to complete the optimization of the calibration tables.

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 <=1200C

In order 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 — in each test, spark is swept (the local variable) while the other variables are held constant (such as speed and load — 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.

Design 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 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.

Start the Toolbox and Create 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.

For more information on the test plan diagram, see "Functions Implemented in the Block Diagram" on page 7-23.

Specify the Model Inputs and Model Type

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.

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 the **Max** to 50 (and leave the **Min** at 0).

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 increase the input factors as follows:

4 Click the up arrow button indicated by the cursor above to increase the number of factors to four.

5 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.

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

7 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 only vital for optimal designs, 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 will 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 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.

8 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.

9 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. Min 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" on page 8-23. You can also open the **Stepwise** window after model fitting to try to improve the fit with the **Stepwise** tools.

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

Create Designs

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**. 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 Click the Remove Point button, and then click the center points to remove all but the two endpoints. You must click inside the frame to select points.

   c 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.

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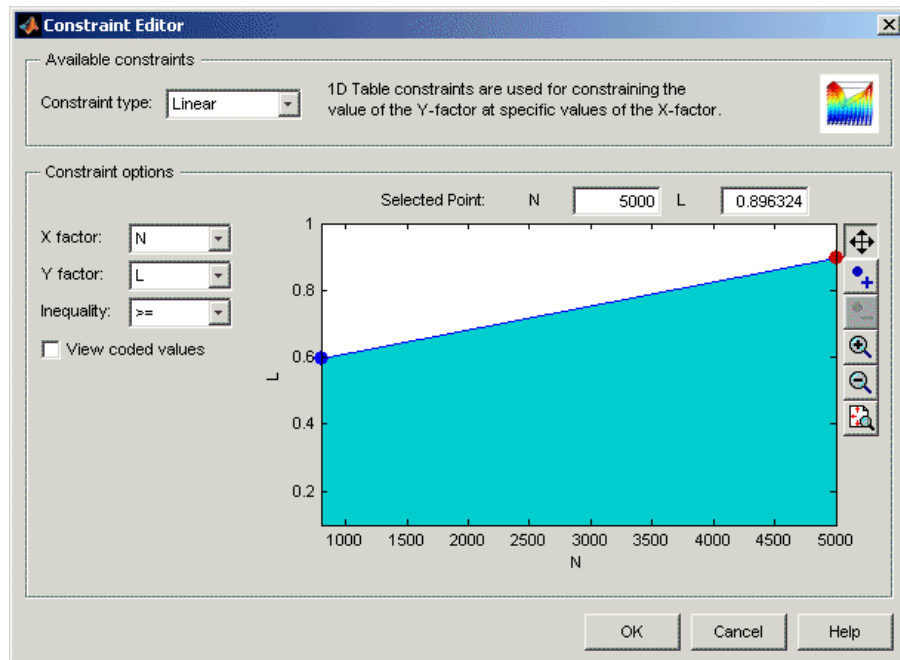
- **e** Click **OK** to add the new constraint, then click **OK** to dismiss the **Constraint Manager**.

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

10-9

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, then click **OK**.

A space-filling design is constructed, using the latin hypercube sampling method. 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 later if required to compensate for this.

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 we provide the example design that was used for this case study: 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.

4 Rename the imported design (click and press **F2**) Constrained_VVT.

5 Choose Constrained_VVT as the preferred design 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 800MHz 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. 10 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.

1 In the Model Browser, click the project node to go 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 ![open file icon] 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.

   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 on 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 Notes** editor appears.

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

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

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

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

12 If you decide certain operating points are unsuitable for modeling, for instance, unstable points on the very 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 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 which are unrealistically far from 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 max torque and 16 degrees after:

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

Click **OK**.

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

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

13 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.

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

14 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 AFR held at a constant value for each sweep. The following illustrates a single sweep with a local model fitted.
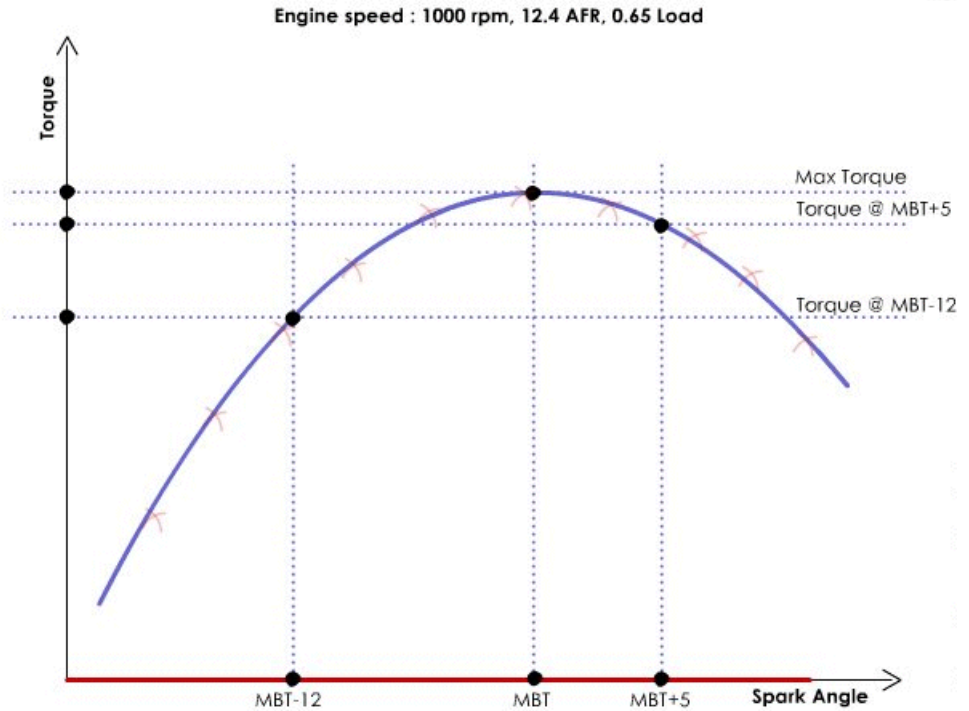
The local models provide the coefficients to generate global models. The equations describing those local model curves have certain coefficients such as max and knot, which for this data are peak torque and MBT spark (the spark angle that generates maximum brake torque).

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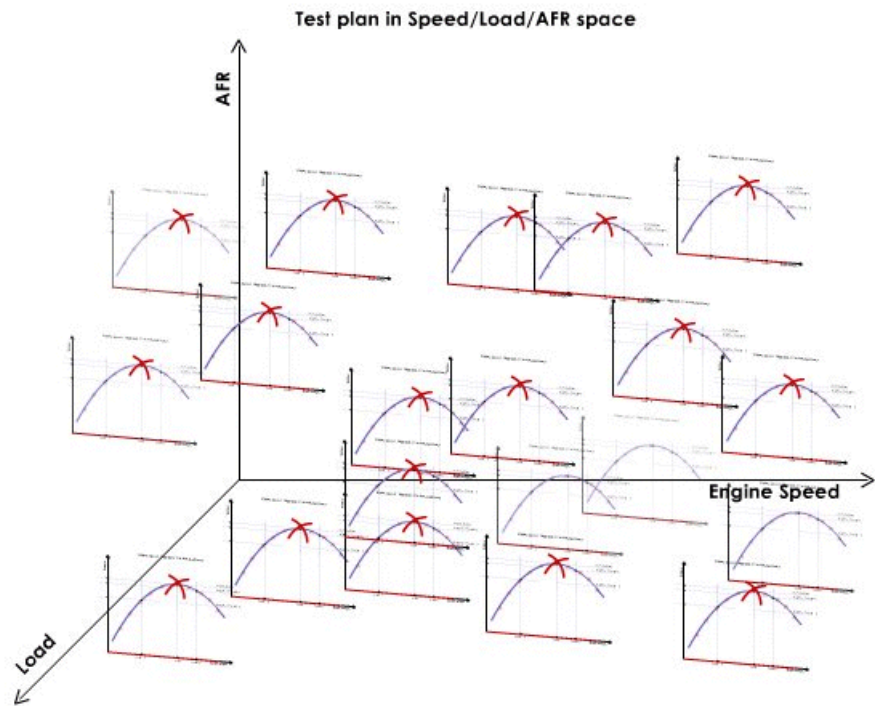
Local models are fitted to each test, in different places across the global space, as illustrated above. Each local model has coefficients for MBT and peak torque etc. These coefficients become the data to which the global models are fitted. Coefficients such as peak torque and MBT are used to make the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

Global models are the best fit of a curve to the values of, for example, MBT for each test. This is repeated for each coefficient, producing several global models fitted to different coefficients of the local models. These coefficients are referred

to as response features of the local models. The following example shows a global model for maximum torque across the speed/load global space.
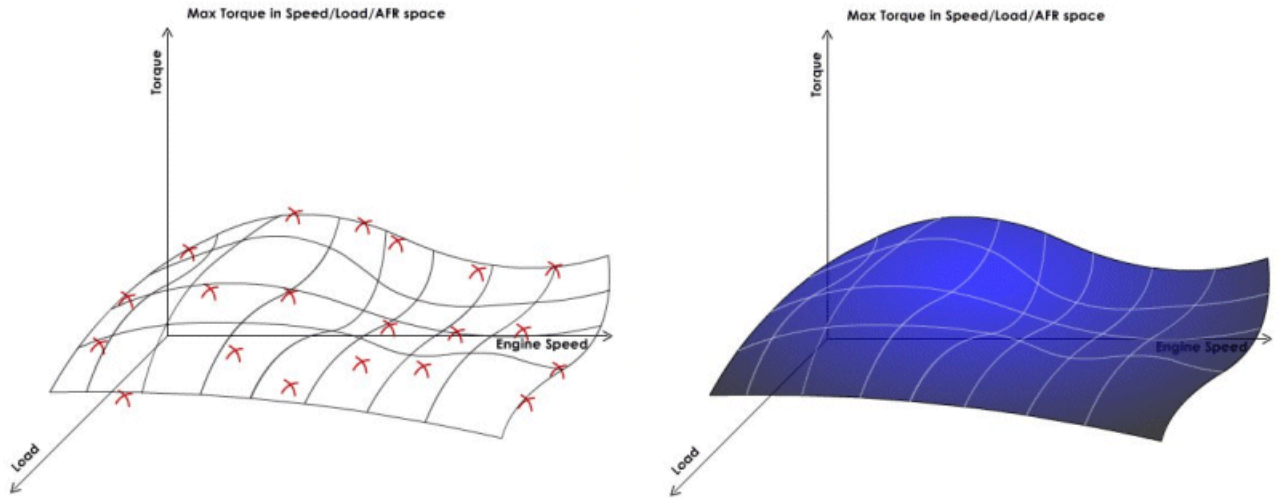
The two-stage model is a surface fitted across all the global models, to describe the behavior across all global variables.

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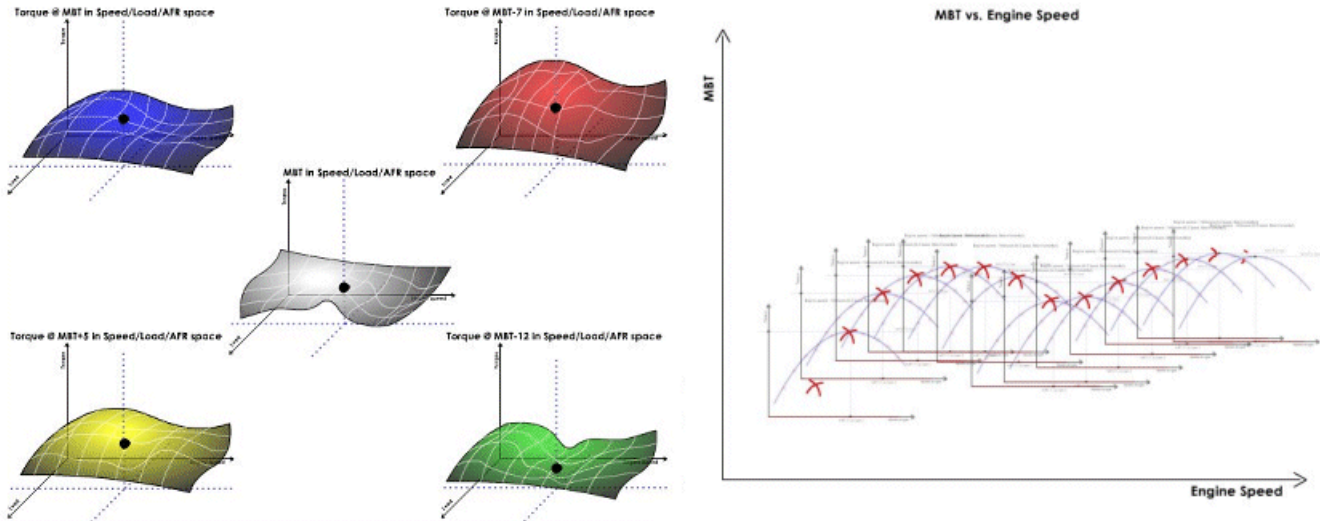
The two-stage model can take values of each coefficient at a certain value of, say, speed, to generate a new curve of torque against spark. This is a slice through the two-stage model surface.

In other words, you can test your two-stage model by comparing it with the local fit and with the data — e.g., 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 2-2, and see "Two-Stage Models for Engines" on page 8-40 for more statistical depth.

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 a function of Load and RPM, subject to constraints of exhaust temperature

In order 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 very 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.

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.

Select 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.

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. Click **Next**.

3 Match the model inputs to the correct data signals. Select SPK in the left and right lists and click the button to select, and so on. Do not select the copy range check box. Click **Next**.

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

5 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. Click **Next**.

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

  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**.

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

8 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.

9 Look at excluded data points (plotted as crosses). 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.

10 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.

Model Selection

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

2 Look through the tests to inspect the fits.

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.

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.

5 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.

6 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.

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

8 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**.

9 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.

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

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

12 Inspect the child models in turn. If you remove outliers that have an RBF centre (marked with a star) be sure to refit the model (click the toolbar button Update Model Fit) to reselect widths and centers.

13 Return to the knot model node and click **Select** in the **Models** pane at the bottom to open the **Model Selection** window and compare the child models simultaneously.

14 Select one of the child models as 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.

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

16 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. 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.

17 You can choose to calculate MLE at this point. This process refits, taking proper account of the correlation between different response features. Try calculating MLE and return to the **Model Selection** window to compare the MLE model with the univariate model and select the best.

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.

   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 the local node and enter -10 to use MBT minus 10 degrees of spark as a new response feature.

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.

Boundary Modeling

Create a boundary model at the test plan node. A model describing the limits of the operating envelope can be very 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 In this case you are only interested in making a model of the global boundary. You are not interested in the Local boundary (you know the range of spark already) or the Response boundary. You can delete these nodes if you want (select the node in the boundary tree and click **Delete**).

4 Select the G_Boundary child node under the Global node.

5 Click **Fit Boundary Constraint** in the toolbar. The **Boundary Constraint Settings** dialog opens.

6 Leave the **Constraint type** set to Star shaped.

7 Under **Model Options**,

  a Select Wendland from the **RBF kernel** drop-down menu.

  b Enter 2 in the **RBF width** edit box.

  c Set the **RBF continuity** to 0.

8 Under **Boundary Point Options**,

  a Select Manual from the **Dilation radius** drop-down menu.

  b Enter 1000 in the **Radius** edit box.

  Increasing the dilation radius like this is a good quick method for constructing a boundary model. You can fine-tune the settings later if you want more detail in the surface. The boundary edge points are determined by expanding a balloon centered on each data point — making the balloon radius large quickly finds edge points for a smooth boundary.

9 Click **OK** and the boundary constraint is calculated.

10 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.

11 Some points appear to be outside the boundary — in this case this is due to the resolution of the plot. You can alter the number of points plotted in the Resolution edit box, but high resolutions can be time-consuming to plot. Instead, select **View –> 2D Slice**.

  Here (and in the **1D Slice**) you can click (and hold) points that appear to be plotted outside the boundary 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.

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

13 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.

14 To examine how the settings change the detail in the surface, click **Fit Boundary Constraint** in the toolbar to return to the **Boundary Constraint Settings** dialog.

15 Enter 0.001 in the **Radius** edit box and click **OK** to refit the model.

16 Switch to the **3D Slice** view to see how this change has made the surface much less smooth. You can see that many points that should be in the interior of the boundary are now selected as edge points, resulting in holes in the operating envelope. Boundary modeling is a tradeoff because you want to capture the detail of the surface without making the model too complex.

17 Click **Fit Boundary Constraint** in the toolbar to return to the **Boundary Constraint Settings** dialog. Try increasing the value for the dilation radius until you are satisfied with the shape of the boundary constraint model.

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.

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

Exporting the Models

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

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. 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.

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 CAGE case studies for instructions.

Diesel Example Study

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. 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, NOx, 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 example takes you through the following steps:

1 "Design of Experiments" on page 10-28

2 "Modeling" on page 10-33

This part of the case study covers the Model Browser section, involving design of experiment, data handling, and model construction and export. The CAGE browser section of the case study continues in the CAGE User's Guide, where you use the models to complete the optimization of the calibration tables.

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 Experiments

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 "constraints" 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 part of the toolbox 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 test plan node in the model tree to see the test plan diagram.

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

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.

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 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 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. For this exercise leave the model type at the default.

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 **New Design** in the **Design Editor** toolbar to create a new design.

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

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

5 The **Constraint Editor** dialog with available constraints appears. Select the default **1D Table** 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 speed (RPM), `basefuelmass`, and the inequality <= from the menus.

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

7 Remove the center points by clicking to select and then clicking **Remove** until the table has only two points, one on each side. Note that you must click *inside* the box to select points. Points become red when selected.

8 Click to select the left point. Enter 1600 in the **RPM** edit box and 200 in the **basefuelmass** edit box.

9 Click to select the right point. Enter 2200 in the **RPM** edit box and 175 in the **basefuelmass** edit box.

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

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

11 Repeat the process 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. 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

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**.

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

Data Source

Data was generated by a Ricardo WAVE model using the experimental design. Simulation tools in MATLAB and Simulink control WAVE. Simulation tools support multiple WAVEs 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.

Data

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 10-11, and for comprehensive information on data handling in the toolbox, see "Data" on page 6-2.

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 10-4 and "How Is a Two-Stage Model Constructed?" on page 10-14 for information about two-stage models.

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, 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 Match up model input names with data signal names one pair at a time, select the check box to copy the range of the signals, and click the button to associate each pair. 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:

- tq
- vtgrpm
- eqrexh
- pkpress
- exhtemp
- NOX
- egrmf

5 Click **Next**, and you are not matching data to a design so you can click **Finish**. The models are calculated.

Building and Evaluating Alternative Models

1 Review the local fits.

Look for trends in the residuals. 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.

The **RMSE Explorer** (use the toolbar button at local level) can be very useful for quickly identifying problem tests with high errors. Strongly outlying residuals should be investigated. 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.

2 Use the same principles to review the global fits.

3 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. Pick different model types from the **Model Setup** dialog, or you can change any existing model by selecting **Model –> Set Up**.

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** 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. 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 Validation tool.

Make use of the **Stepwise** tool (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.

4 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.

5 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

Try exporting the models directly to the CAGE browser part of the MBC 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 10-25 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 CAGE Case Studies in the CAGE User's Guide.

Index