Model-Based Calibration Toolbox

For Use with MATLAB®

Computation

Visualization

Programming

Model Browser User's Guide

Version 1

The MathWorks

How to Contact The MathWorks:

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

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

Model-Based Calibration Toolbox Model Browser User's Guide

Printing History: July 2002 Online only Revised for Version 1.1 (Release 13)

Contents

1 Getting Started

What Is the Model-Based Calibration Toolbox? **1-3**
About the Model Browser **1-3**
About CAGE **1-3**

How to Use This Manual **1-5**

System Requirements **1-6**
Hardware Requirements **1-6**
Operating System Requirements **1-6**
Required MathWorks Products **1-7**
Optional MathWorks Products **1-7**

2 Quickstart Tutorial

Two-Stage Models **2-3**

Starting the Toolbox **2-5**

Setting Up the Model **2-9**
Setting Up the Local Model **2-12**
Setting Up the Global Model **2-16**
Selecting Data **2-20**
Specifying the Response Model **2-23**

Verifying the Model **2-24**
Verifying the Local Model **2-24**
Verifying the Global Model **2-26**
Selecting the Two-Stage Model **2-29**
Comparing the Local Model and the Two-Stage Model **2-34**
Maximum Likelihood Estimation **2-35**
Response Node **2-38**

Exporting the Model **2-40**

3 Design of Experiment Tutorial

What Is Design of Experiment? **3-4**
Design Styles **3-5**

Starting the Design Editor **3-6**

Optimal Designs **3-9**
Start Point Tab **3-10**
Candidate Set Tab **3-11**
Algorithm Tab **3-14**

Design Displays **3-16**

Prediction Error Variance Viewer **3-19**
Improving the Design **3-21**

Classical Designs **3-24**
Classical Design Browser **3-26**
Set Up and View a Classical Design **3-27**

Design Evaluation Tool **3-30**
Improving the Design **3-32**

Space-Filling Designs **3-33**
Setting Up a Space-Filling Design **3-34**

Applying Constraints **3-35**
Saving Designs **3-40**

4 Data Editor Tutorial

Loading the Data **4-3**
Entering the Data Editor **4-3**
Loading a Data File **4-4**

Viewing and Editing the Data **4-6**
User-Defined Variables & Filtering **4-8**
Defining a New Variable **4-9**
Applying a Filter **4-10**
Sequence of Variables **4-12**
Deleting and Editing Variables and Filters **4-13**
Storage **4-14**
Test Groupings **4-15**

5 GUI Reference

Project Level: Startup View **5-3**
Project Level: Toolbar **5-7**
Project Level: Menus **5-8**

Model Tree **5-10**

Test Plans **5-15**
Creating a New Test Plan **5-15**
Creating New Test Plan Templates **5-16**

Test Plan Level **5-19**
Block Diagram Representations of Models **5-20**
Test Plan Level: Toolbar and Menus **5-23**

Setting Up Models **5-26**
Instant One-Stage Model Setup **5-27**
Instant Two-Stage Model Setup **5-28**
Setting Up Inputs **5-30**
Global Model Setup **5-32**
Global Linear Models: Polynomials and Hybrid Splines **5-33**
Global Model Class: Radial Basis Function **5-39**
Global Model Class: Hybrid RBF **5-42**
Global Model Class: Multiple Linear Models **5-43**
Global Model Class: Free Knot Spline **5-44**
Local Model Setup **5-46**
Local Model Class: Polynomials and Polynomial Splines **5-47**
Local Model Class: Truncated Power Series **5-50**
Local Model Class: Free Knot Spline **5-51**
Local Model Class: Growth Models **5-52**
Local Model Class: Linear Models **5-53**
Local Model Class: User-Defined Models **5-54**
Local Model Class: Transient Models **5-54**
Covariance Modeling **5-54**
Correlation Models **5-56**
Transforms **5-56**
Designing Experiments **5-57**

Data **5-59**
The Data Editor **5-61**
Data Loading and Merging **5-65**
User-Defined Variables and Filtering **5-69**
Variable and Filter Editors **5-71**
Storage **5-74**
Test Groupings **5-75**
Data Wizard **5-78**
Data Selection Window **5-81**

New Response Models **5-87**
Datum Models **5-88**

Local Level **5-89**
Local Special Plots **5-91**
Local Scatter Plots **5-92**
Diagnostic Statistics Pane **5-93**
Response Features List Pane **5-93**
Test Notes Pane **5-94**
Local Level: Toolbar **5-94**
Local Level: Menus **5-95**
Model Menu (Local Level) **5-96**
View Menu (Local Level) **5-99**
Outliers Menu (Local Level) **5-100**
Outlier Selection Criteria **5-100**
Data Tab **5-103**

Global Level **5-105**
Global Special Plots **5-107**
Global Scatter Plots **5-108**
Global Level: Toolbar **5-110**
Global Level: Model-Specific Tools **5-111**
Global Level: Menus **5-113**

Selecting Models **5-117**
Select **5-117**
Model Selection Window **5-118**
Tests View **5-120**
Predicted/Observed View **5-122**
Response Surface View **5-124**
Likelihood View **5-127**
RMSE View **5-129**
Residuals View **5-132**
Cross Section View **5-133**

MLE **5-137**
MLE Settings **5-139**

Response Level **5-141**
Response Level: Toolbar and Menus **5-143**

Model Evaluation Window **5-144**
Model Evaluation Using Other Data **5-146**

Exporting Models **5-148**
What Is Exported? **5-151**
Evaluating Models in the Workspace **5-152**

The Design Editor **5-153**
Design Styles **5-154**
Design Editor Displays **5-155**

Creating a Classical Design **5-159**

Creating a Space Filling Design **5-162**
Setting Up a Space Filling Design **5-166**

Creating an Optimal Design **5-168**
Optimal Design: Start Point Tab **5-170**
Optimal Design: Candidate Set Tab **5-172**
Optimal Design: Algorithm tab **5-175**
Adding Design Points **5-178**
Fixing, Deleting, and Sorting Design Points **5-181**
Saving and Importing Designs **5-183**

Applying Constraints **5-184**
Constraint Types **5-185**
Importing Constraints **5-191**

Prediction Error Variance Viewer **5-193**

6 Technical Documents

Linear Regression **6-3**
Stepwise Regression **6-3**

Definitions **6-5**

Prediction Error Variance **6-7**

High-Level Model Building Process Overview **6-9**
Univariate Model Building Process Overview **6-10**

Stepwise Regression Techniques **6-13**
Stepwise Table **6-13**

Box-Cox Transformation **6-18**

Linear Model Statistics Displays **6-22**
Summary Statistics **6-22**
ANOVA Table **6-22**
Diagnostic Statistics **6-23**
PRESS Statistic **6-23**
Pooled Statistics **6-24**

Design Evaluation Tool **6-27**
Design Matrix **6-29**
Full FX Matrix **6-29**
Model Terms **6-29**
Z_2 Matrix **6-29**
Alias Matrix **6-29**
$Z_{2.1}$ Matrix **6-30**
Regression Matrix **6-30**
Coefficient Information **6-30**
Standard Error **6-32**
Hat Matrix **6-32**
|X'X| **6-33**
Raw Residual Statistics **6-33**
Degrees of Freedom Table **6-34**
Design Evaluation Graphical Displays **6-35**
Export of Design Evaluation Information **6-35**

Maximum Likelihood Estimation **6-36**

Two-Stage Models for Engines **6-37**
Local Models **6-38**
Global Models **6-40**
Two-Stage Models **6-41**
Prediction Error Variance for Two-Stage Models **6-43**
Global Model Selection **6-45**
References **6-46**

Local Model Definitions **6-48**
Local Models and Associated Response Features **6-48**
Polynomial Models **6-48**
Polynomial Splines **6-49**
Truncated Power Series Basis (TPSBS) Splines **6-50**
Free Knot Splines **6-51**
Three Parameter Logistic Model **6-51**
Morgan-Mercer-Flodin Model **6-52**
Four-Parameter Logistic Curve **6-53**
Richards Curves **6-54**
Weibul Growth Curve **6-55**
Exponential Growth Curve **6-55**
Gompertz Growth Model **6-56**

Neural Networks **6-57**

User-Defined Models **6-58**
To Begin **6-58**
Template File **6-58**
Subfunctions **6-59**
Optional Subfunctions **6-59**
Checking into MBC **6-62**
Check In **6-62**
MBC Modeling **6-63**

Transient Models **6-64**
The Simulink Model **6-64**
Parameters **6-65**
Checking the Model **6-67**
Transient Models **6-67**
Template File **6-68**
Subfunctions **6-68**
Checking into MBC **6-70**
Check In **6-70**
MBC Modeling **6-71**

Data Loading Application Programming Interface **6-73**
Data Function Prototype **6-73**
Input Arguments **6-73**
Output Arguments **6-73**
Data Function Check In **6-74**

7 Radial Basis Functions

Guide to Radial Basis Functions for Model Building **7-3**

Types of Radial Basis Functions **7-4**
Gaussian **7-4**
Thin-Plate Spline **7-6**
Logistic Basis Function **7-7**
Wendland’s Compactly Supported Function **7-8**
Multiquadrics **7-10**
Reciprocal Multiquadrics **7-11**

Fitting Routines **7-12**

Center Selection Algorithms **7-13**
Rols **7-13**
RedErr **7-14**
WiggleCenters **7-14**
CenterExchange **7-14**

Lambda Selection Algorithms **7-16**
IterateRidge **7-16**
IterateRols **7-17**
StepItRols **7-18**

Width Selection Algorithms **7-19**
TrialWidths **7-19**
WidPerDim **7-19**

Prune Functionality **7-21**

Statistics **7-24**
GCV Criterion **7-24**
GCV for Ridge Regression **7-25**
GCV for Rols **7-25**
References **7-27**

Hybrid Radial Basis Functions **7-28**
Width Selection Algorithm: TrialWidths **7-28**
Lambda and Term Selection Algorithms: Interlace **7-28**
Lambda and Term Selection Algorithms: TwoStep **7-29**

Tips for Modeling with Radial Basis Functions **7-30**
Plan of Attack **7-30**
How Many RBFs to Use **7-31**
Width Selection Algorithms **7-32**
Which RBF to Use **7-33**
Lambda Selection Algorithms **7-33**
Center Selection Algorithms **7-34**
General Parameter Fine-Tuning **7-34**
Hybrid RBFs **7-34**

Index

Getting Started

What Is the Model-Based Calibration Toolbox? 1-3
About the Model Browser 1-3
About CAGE 1-3

How to Use This Manual 1-5

System Requirements 1-6
Hardware Requirements 1-6
Operating System Requirements 1-6
Required MathWorks Products 1-7

Optional MathWorks Products 1-7

The following sections introduce the Model-Based Calibration Toolbox.

"What Is the Model-Based Calibration Toolbox?" on page 1-3

"About the Model Browser" on page 1-3

"About CAGE" on page 1-3

"How to Use This Manual" on page 1-5

"System Requirements" on page 1-6

Starting the Model Browser

To start the application, type

```
mbcmodel
```

at the command prompt.

For information on learning and using the Model Browser, see "How to Use This Manual" on page 1-5.

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.

- "Quickstart Tutorial" on page 2-1 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.
- "Design of Experiment Tutorial" on page 3-1 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.
- "Data Editor Tutorial" on page 4-1 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:

- "GUI Reference" on page 5-1 is a complete guide to the functionality in the user interfaces of the Model Browser, Design Editor, Data Editor, Model Selection and Evaluation windows, and all related dialogs.
- "Technical Documents" on page 6-1 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.
- "Radial Basis Functions" on page 7-1 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.

The product has been tested on Microsoft Windows NT, 2000, and 98.

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

Required MathWorks Products

Model-Based Calibration requires the following other MathWorks products:

- Simulink
- Optimization Toolbox
- Statistics Toolbox
- Extended Symbolic Math Toolbox

Optional MathWorks Products

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

- Neural Network 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

Quickstart Tutorial

Two-Stage Models 2-3

Starting the Toolbox 2-5

Setting Up the Model 2-9
Setting Up the Local Model 2-12
Setting Up the Global Model 2-16
Selecting Data 2-20
Specifying the Response Model 2-23

Verifying the Model 2-24
Verifying the Local Model 2-24
Verifying the Global Model 2-26
Selecting the Two-Stage Model 2-29
Comparing the Local Model and the Two-Stage Model . . . 2-34
Maximum Likelihood Estimation 2-35
Response Node 2-38

Exporting the Model 2-40

The following sections give you a quick introduction to the modeling end of the Model-Based Calibration Toolbox. They do so by showing 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.

- “Two-Stage Models” on page 2-3
- “Starting the Toolbox” on page 2-5
- “Setting Up the Model” on page 2-9
- “Setting Up the Local Model” on page 2-12
- “Setting Up the Global Model” on page 2-16
- “Selecting Data” on page 2-20
- “Specifying the Response Model” on page 2-23
- “Verifying the Model” on page 2-24
- “Verifying the Local Model” on page 2-24
- “Verifying the Global Model” on page 2-26
- “Selecting the Two-Stage Model” on page 2-29
- “Exporting the Model” on page 2-40

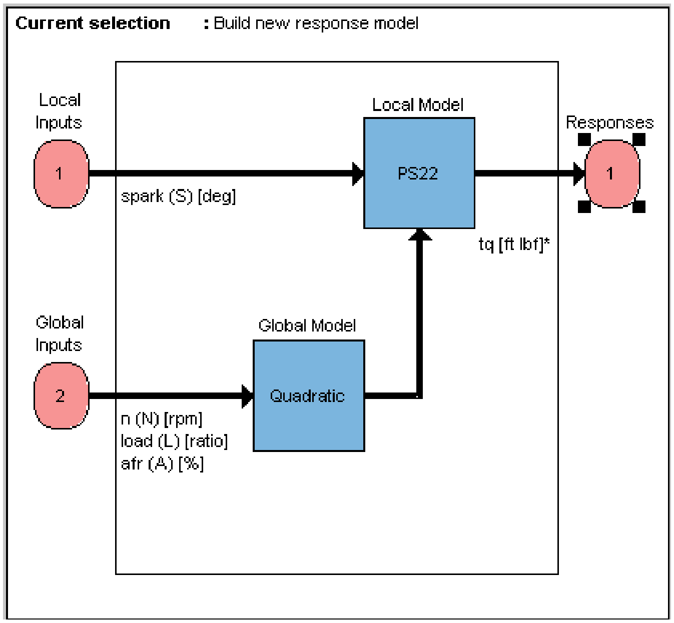
For an explanation of how two-stage models are constructed and how they differ from one-stage models, see the following section, “Two-Stage Models” on page 2-3, or see “Two-Stage Models for Engines” on page 6-37 in the Technical Documents for more depth.

This tutorial is a step-by-step guide to constructing a single two-stage model for some engine data. In the normal modeling process you might create many different models for one project and compare them to find the best solution.

Two-Stage Models

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

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



Current selection : Build new response model

Starting the Toolbox

1 Double-click the MATLAB icon to start MATLAB.

2 To start the toolbox from the launch pad, open the Model Based Calibration (MBC) Toolbox™, then double-click **Model Browser**.

   Alternatively, 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.)

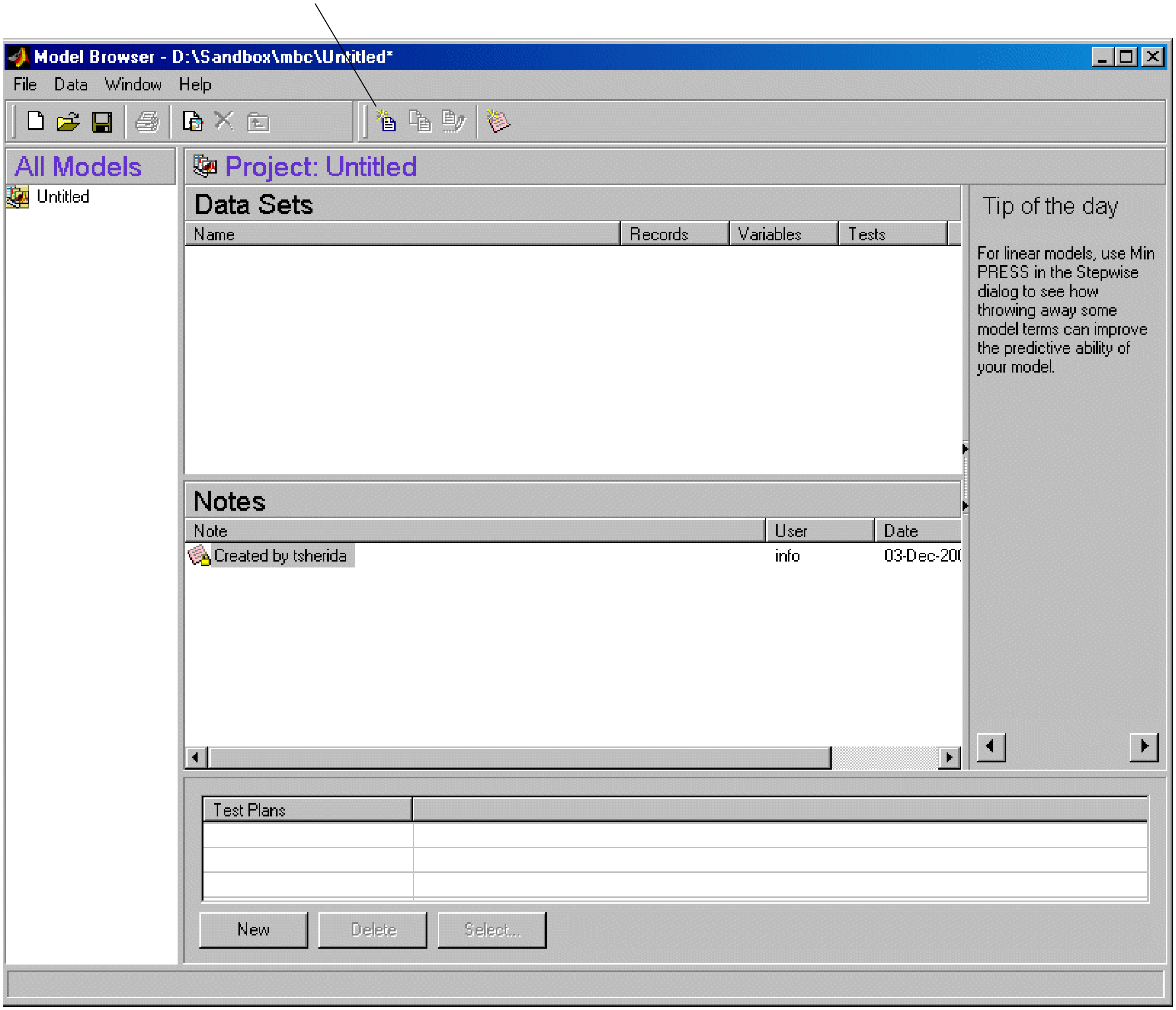
Note You can edit your user information at any time by selecting **File –> Preferences**.

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



Data Sets

Name	Records	Variables	Tests

Tip of the day

For linear models, use Min PRESS in the Stepwise dialog to see how throwing away some model terms can improve the predictive ability of your model.

Notes

Note	User	Date
Created by tsherida	info	03-Dec-20(

Test Plans	

Load the example data file `holliday.xls`.

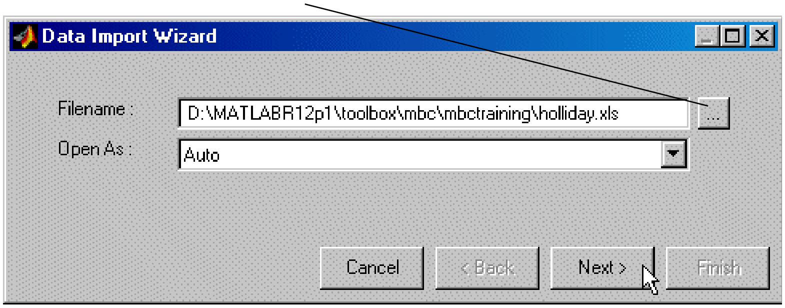
1 Click the button on the toolbar, or choose **Data –> New Data**.

   This opens the Data Editor window.

2 Click the **Open** icon on the toolbar ( ) or choose **File –> Load data from file**.

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

Click here to browse for a file



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

Open As : Auto

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

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

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

   For more details on functionality available within the Data Editor, see “The Data Editor” on page 5-61.

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.

---

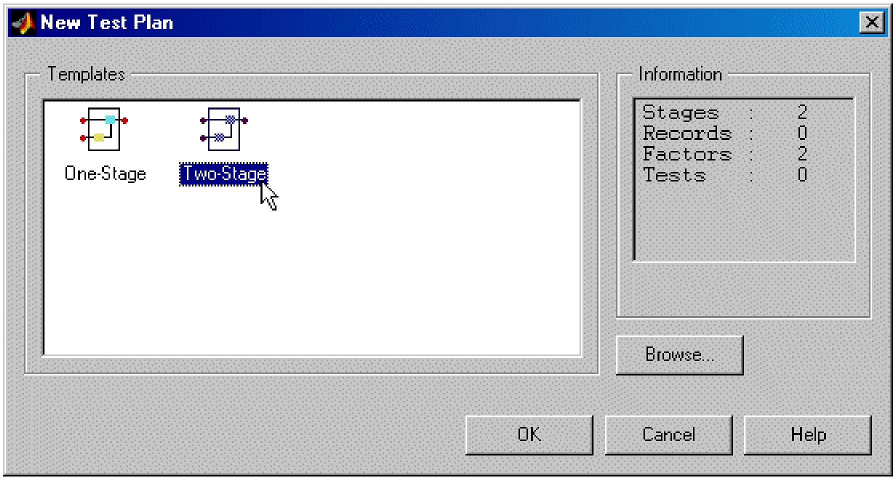
1 To create a new test plan, do one of the following:

- 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). See the model tree.

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

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



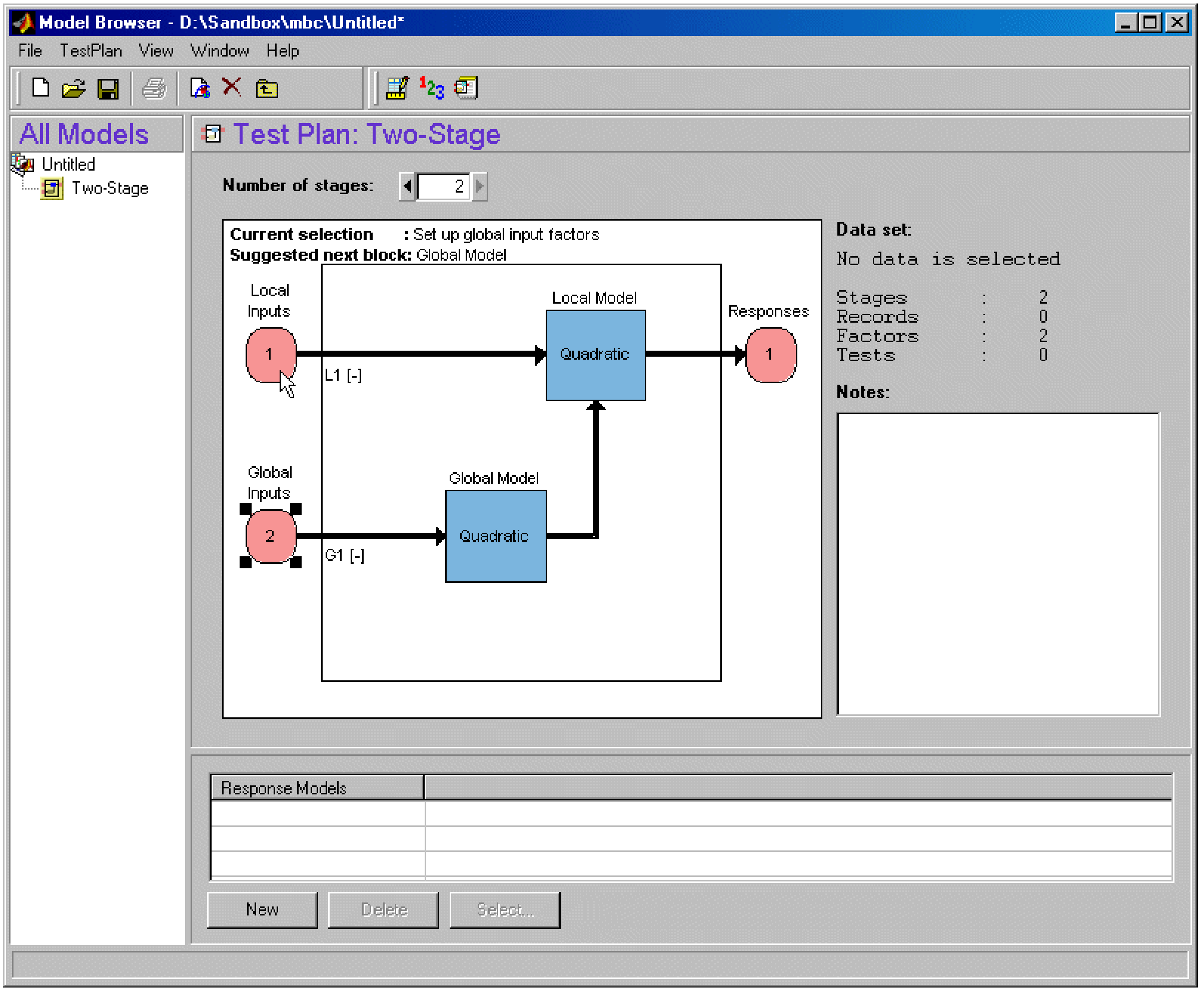
```
Stages   :   2
Records  :   0
Factors  :   2
Tests    :   0
```

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

   The default name of the model, `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 5-21.



Number of stages: 2

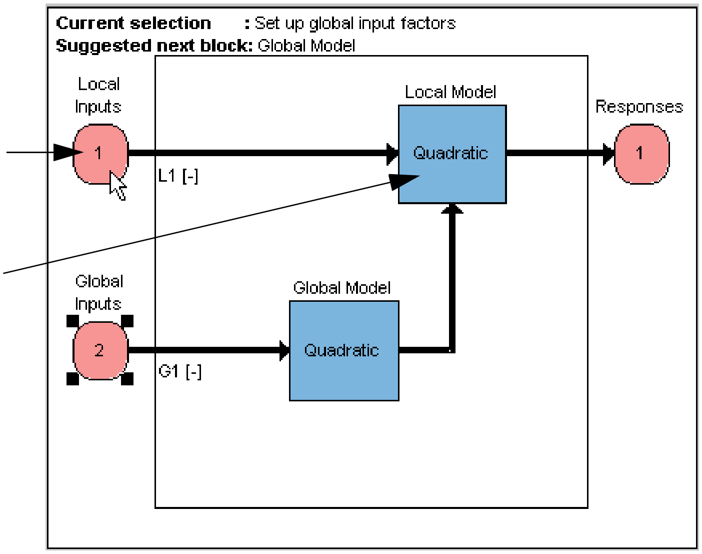
Current selection : Set up global input factors
Suggested next block: Global Model

Data set:

```
No data is selected

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

Setting Up the Local Model

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



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

Double-click here to choose local model inputs

Double-click here to choose local model type

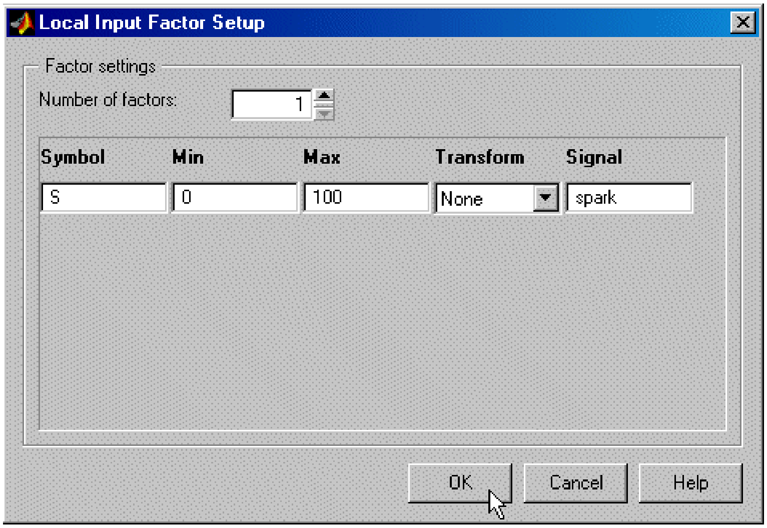
Specifying the Local Model Input

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 the Local Inputs icon on the model diagram to specify the local model input.

   The **Local Input Factor Setup** dialog box appears.



Local Input Factor Setup

Factor settings

Number of factors: 1

Symbol	Min	Max	Transform	Signal
S	0	100	None	spark

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

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

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

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

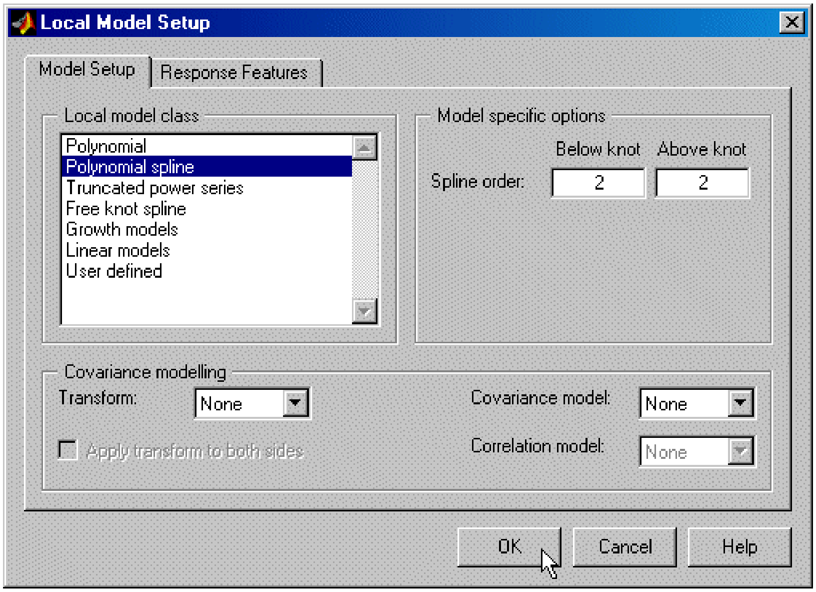
Specifying the Local Model Type

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

To specify polyspline as the model type:

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

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



Local Model Setup

Local model class: Polynomial spline

Model specific options

Spline order: Below knot 2, Above knot 2

Covariance modelling

Transform: None

□ Apply transform to both sides

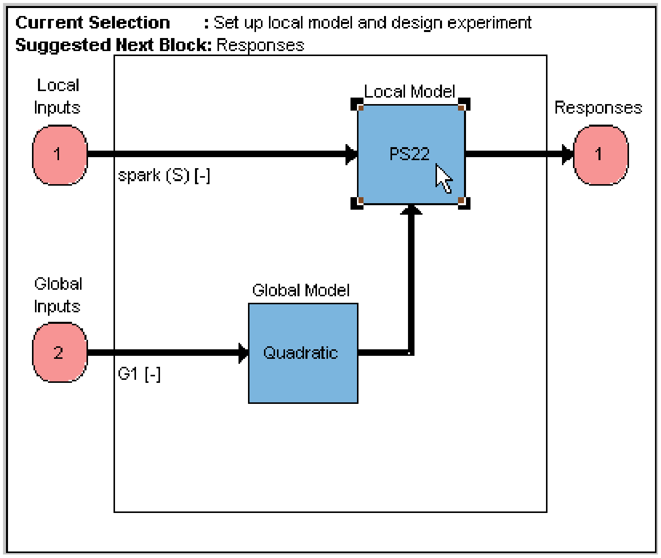
Covariance model: None

Correlation model: None

- **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.

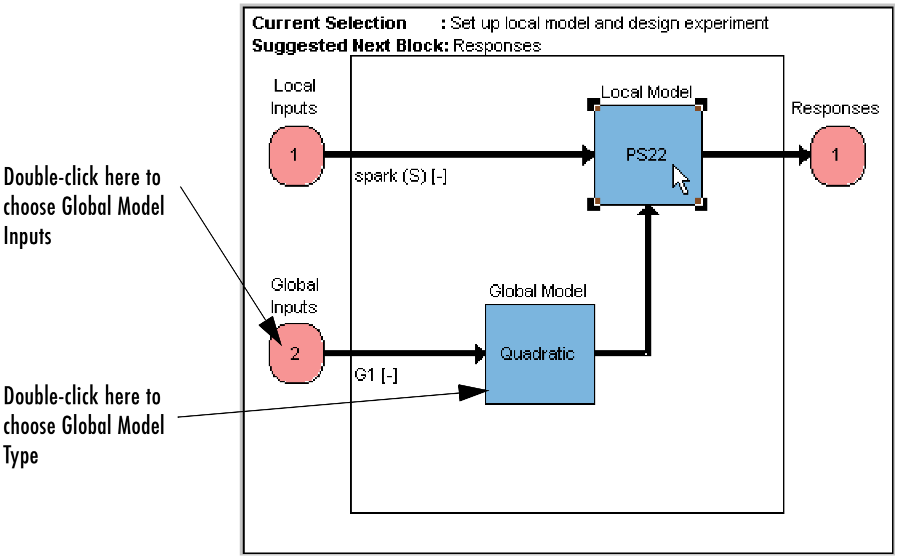
Notice that the new name of the local model class, `PS` (for polyspline) `2,2` (for spline order above and below knot) now appears on the two-stage model diagram.



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

Setting Up the Global Model

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



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

Double-click here to choose Global Model Inputs

Double-click here to choose Global Model Type

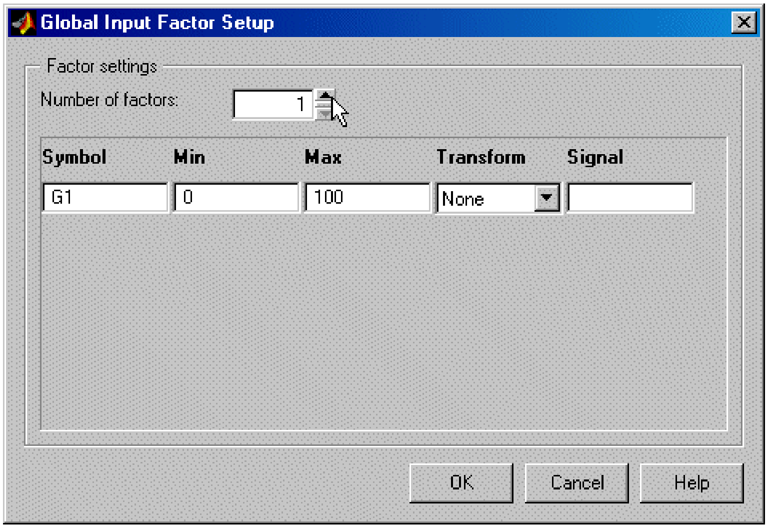
Specifying the Global Model Inputs

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

To specify these inputs:

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

   The **Global Input Factor Setup** dialog box appears.



Number of factors: 1

Symbol	Min	Max	Transform	Signal
G1	0	100	None	

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

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

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

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

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

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

Specifying the Global Model Type

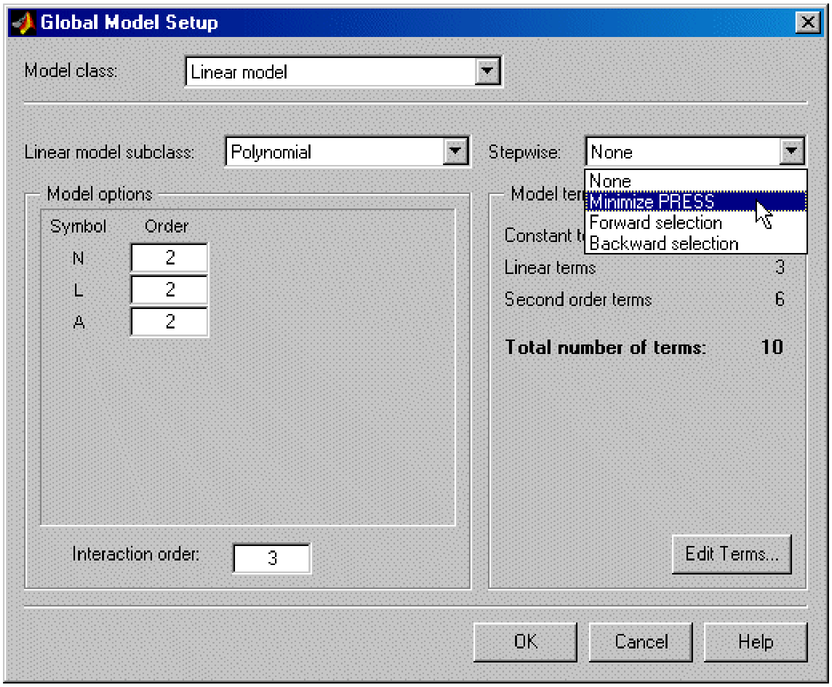
Fitting the local model finds values for each model coefficient or response feature (for example, `knot`) for each test. These coefficients then become the data to which you fit the global model.

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

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

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

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



Global Model Setup

Model class: Linear model

Linear model subclass: Polynomial

Stepwise: None

Model options

Symbol	Order
N	2
L	2
A	2

Interaction order: 3

Model ter

Constant t

Linear terms 3

Second order terms 6

Total number of terms: 10

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

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

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

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

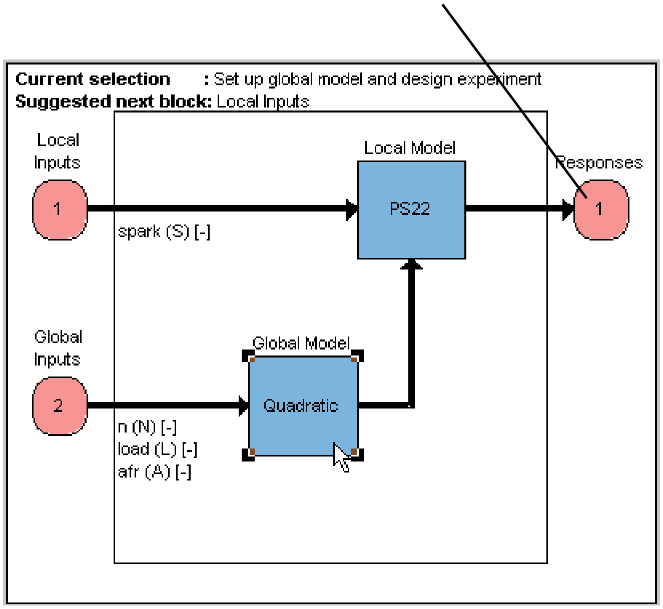
This completes the setup of the global model.

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.

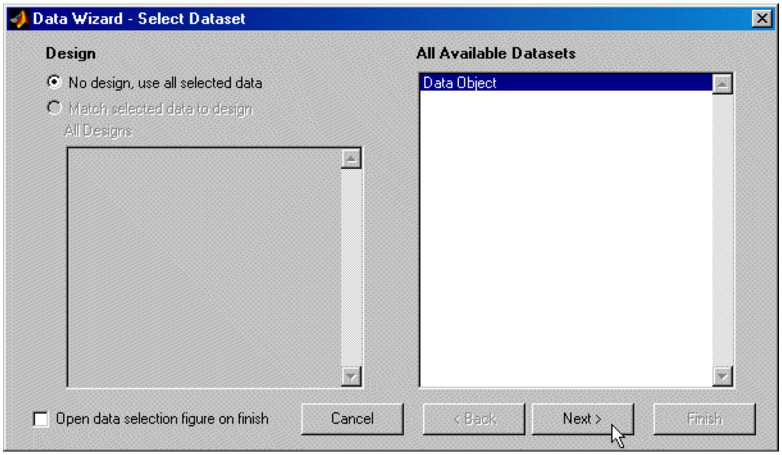
Launch Data Wizard by double-clicking here



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

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

The **Data Wizard** dialog appears.

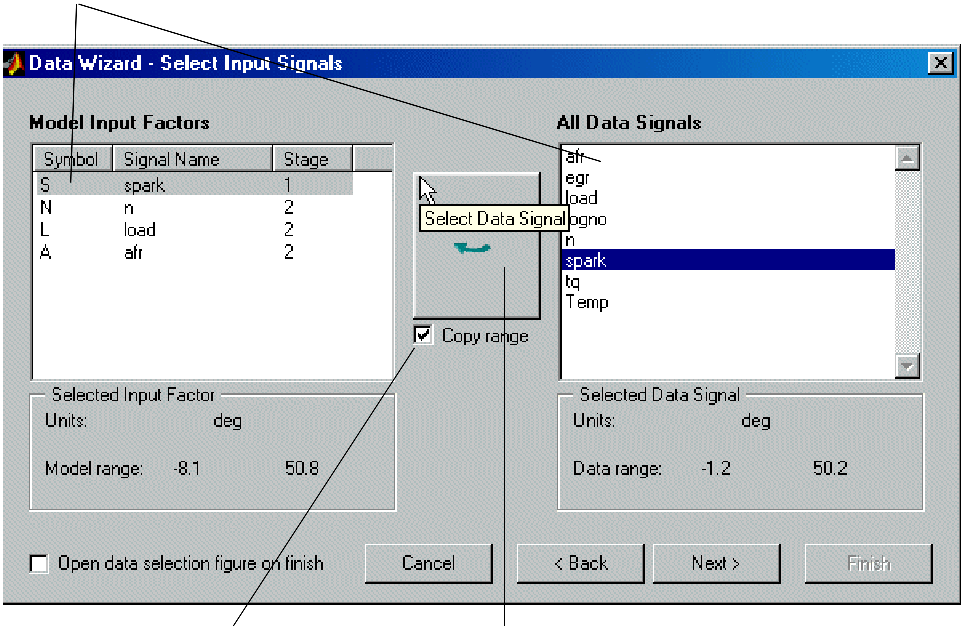

Data Wizard - Select Dataset
Design
● No design, use all selected data
○ Match selected data to design
All Designs
All Available Datasets
Data Object
□ Open data selection figure on finish
Cancel
< Back
Next >
Finish

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

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

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

Select in these lists to match input factor to data signal



Data Wizard - Select Input Signals

Model Input Factors

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

All Data Signals: afr, egr, load, logno, n, spark, tq, Temp

Selected Input Factor
Units: deg
Model range: -8.1 50.8

☑ Copy range

Selected Data Signal
Units: deg
Data range: -1.2 50.2

□ Open data selection figure on finish

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`).

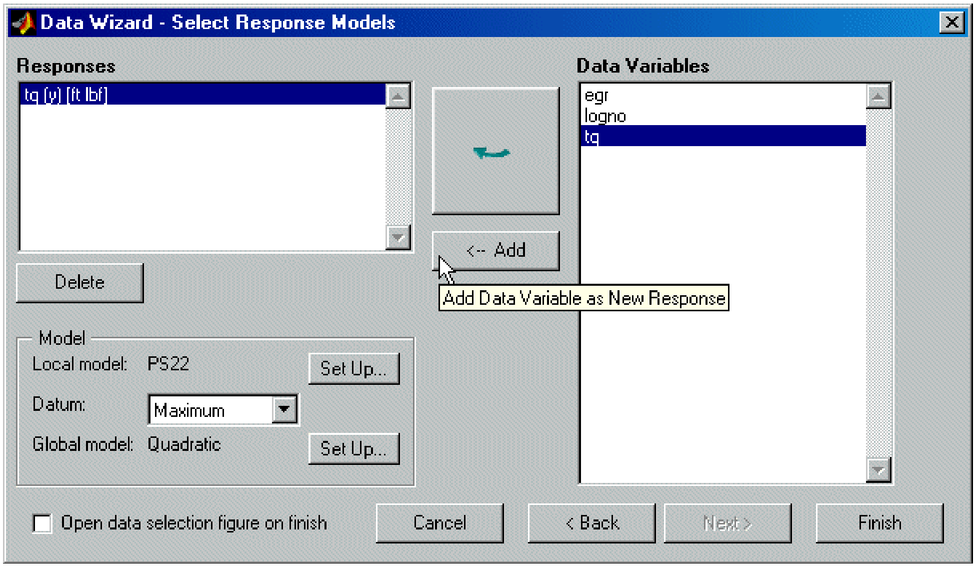
If the signal name entered during the input factor setup matches a signal name in the data set, the Wizard automatically selects the correct signal when the input factor is selected. If the name is not correct, you must select the correct signal manually by clicking. This autoselect facility can save 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.



Data Wizard - Select Response Models

Responses

tq (y) [ft lbf]

Data Variables

egr
logno
tq

Model

Local model: PS22

Datum: Maximum

Global model: Quadratic

□ Open data selection figure on finish

Add Data Variable as New Response

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.

4 Click **Finish**.

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

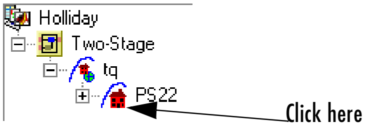
Notice that torque appears on the two-stage model diagram, and a new node appears on the tree in the **All Models** pane, called PS22.

Verifying the Model

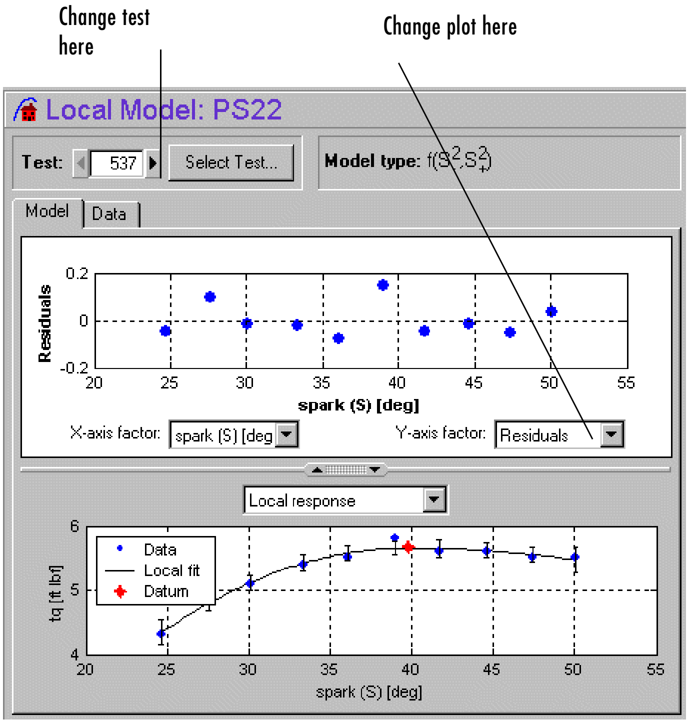
Verifying the Local Model

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.



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



Change test here

Change plot here

Local Model: PS22

Test: 537 Select Test...

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

X-axis factor: spark (S) [deg

Y-axis factor: Residuals

Local response

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**.

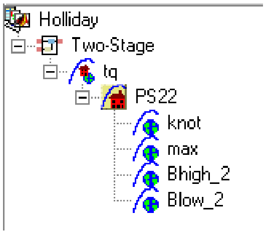
Both plots have right-click pop-up menus offering various options such as removing and restoring outliers and confidence intervals. Clicking any data point marks it in red as an outlier.

You can use the **Test Notes** pane to record information on particular tests. Each test has its own notes pane. 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.

Verifying the Global Model

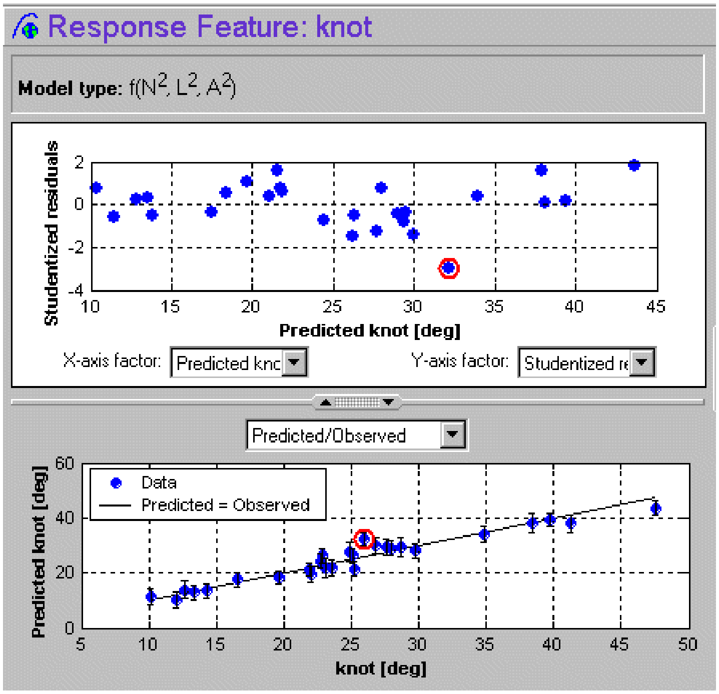
The next step is to check through the global models to see how well they fit the data.

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



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

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


Response Feature: knot

Model type: f(N2, L2, A2)

X-axis factor: Predicted knc

Y-axis factor: Studentized re

Predicted/Observed

3 The response feature `knot` has one outlier 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.

4 Select the other three response features in turn: `max`, `Bhigh 2`, 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.

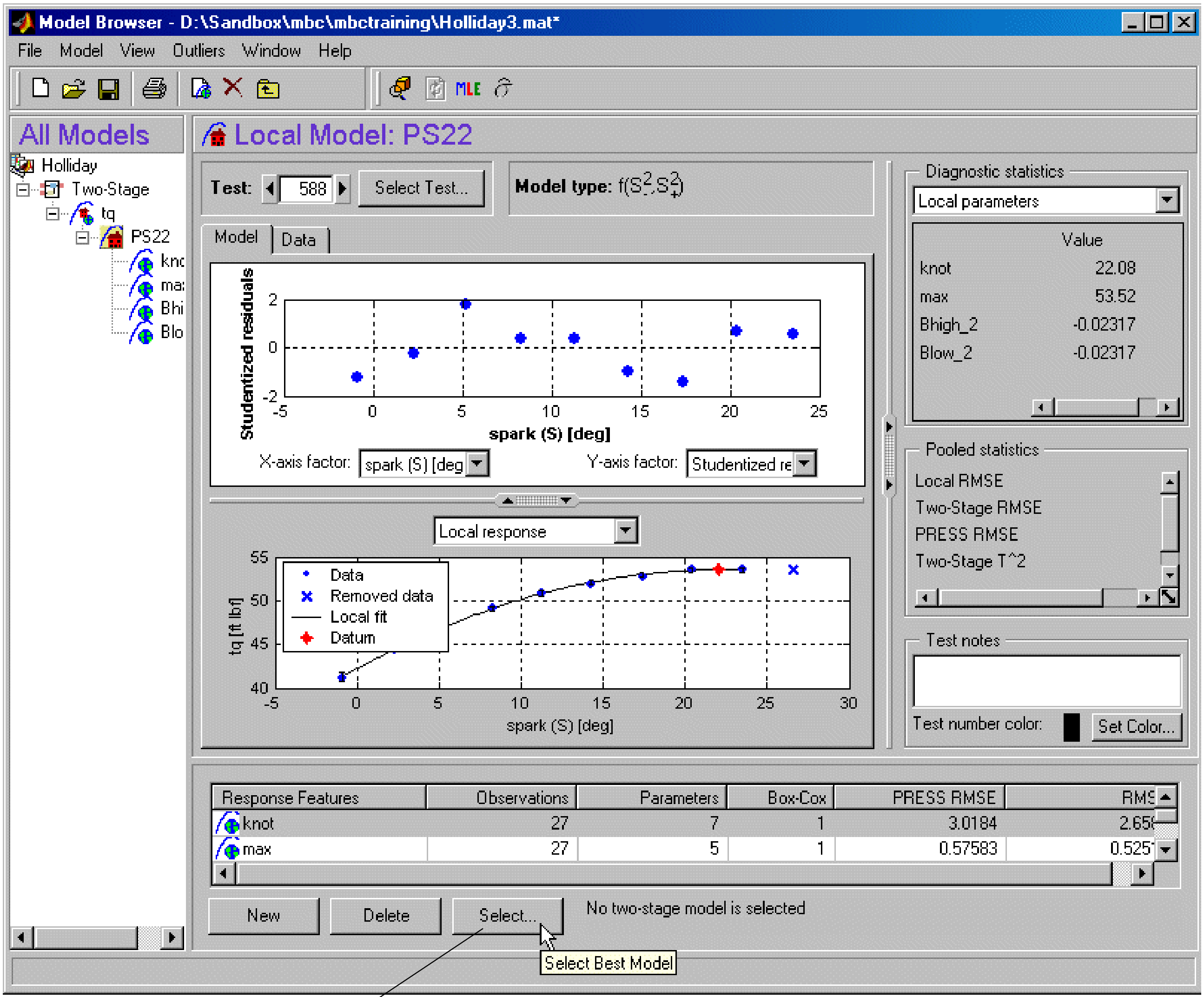
Within this tutorial, you use the following:

- "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 5-117 in the GUI reference.

Note To construct a two-stage model from the local and global models, you click the local node in the model tree (with the house icon) and click the **Select** button. This is the next step in the tutorial.

Once you are satisfied with the fit of the local and global models, it is time to construct a two-stage model from them. Return to the **Local Model** view by clicking the local node `PS22` in the Model Browser tree. The Model Browser should look like the following example.



Test: 588

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

X-axis factor: spark (S) [deg

Y-axis factor: Studentized re

Diagnostic statistics: Local parameters

	Value
knot	22.08
max	53.52
Bhigh_2	-0.02317
Blow_2	-0.02317

Pooled statistics:

Local RMSE
Two-Stage RMSE
PRESS RMSE
Two-Stage T^2

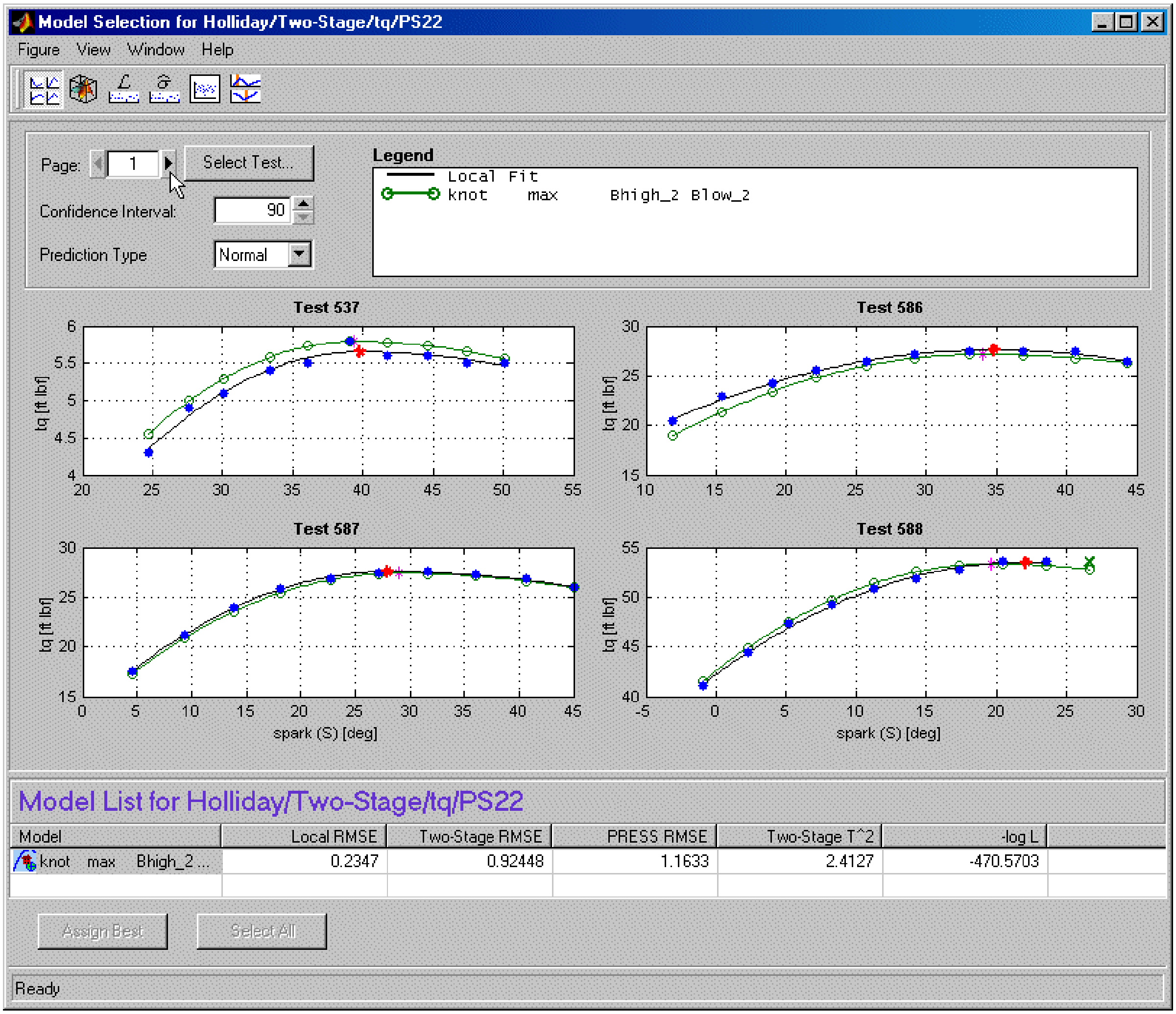
Response Features	Observations	Parameters	Box-Cox	PRESS RMSE	RMS
knot	27	7	1	3.0184	2.65
max	27	5	1	0.57583	0.525

No two-stage model is selected

Open Model Selection here

Click **Select** in the **Response Features** list pane, and the **Model Selection** window appears. This window is intended to help you select a **Best Model** by comparing several candidate models. 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



Model Selection for Holliday/Two-Stage/tq/PS22

Page: 1

Confidence Interval: 90

Prediction Type: Normal

Model List for Holliday/Two-Stage/tq/PS22

Model	Local RMSE	Two-Stage RMSE	PRESS RMSE	Two-Stage T^2	-log L
knot max Bhigh_2 ...	0.2347	0.92448	1.1633	2.4127	-470.5703

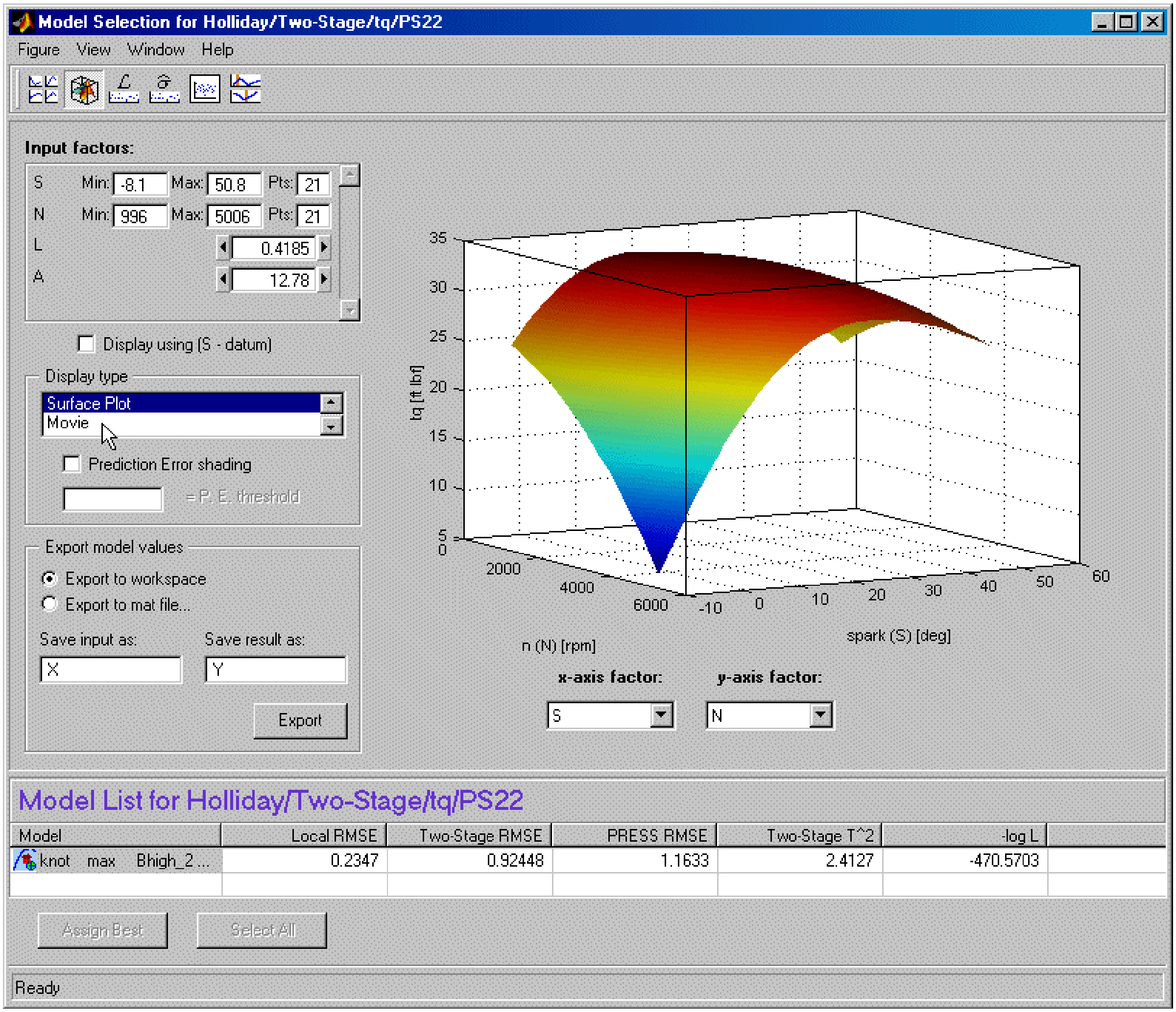
Ready

Scroll though the tests using the left/right arrows or the **Select Test** button at the top left. The plots show the fit of the two-stage model for each test (green open circles and line), compared with the fit of the local model (black line) and the data (blue dots). You can left-click (and hold) to see information on each test or zoom in on points of interest by **Shift**-click-dragging or middle-click-dragging. Double-click to return the plot to the original size.

Response Surface View

You view the model as a surface by clicking the **Response Surface** icon in the toolbar. You can rotate the plot by click-dragging it.

1 Click **Movie** in the **Display Type** list to see the surface (torque against spark and speed) vary through different values of `load`.



Input factors:

S	Min: -8.1	Max: 50.8	Pts: 21
N	Min: 996	Max: 5006	Pts: 21
L		0.4185	
A		12.78	

☐ Display using (S - datum)

Display type

Surface Plot
Movie

☐ Prediction Error shading

= P. E. threshold

Export model values

● Export to workspace
○ Export to mat file...

Save input as: X

Save result as: Y

x-axis factor: S

y-axis factor: N

Model List for Holliday/Two-Stage/tq/PS22

Model	Local RMSE	Two-Stage RMSE	PRESS RMSE	Two-Stage T^2	-log L
knot max Bhigh_2 ...	0.2347	0.92448	1.1633	2.4127	-470.5703

Ready

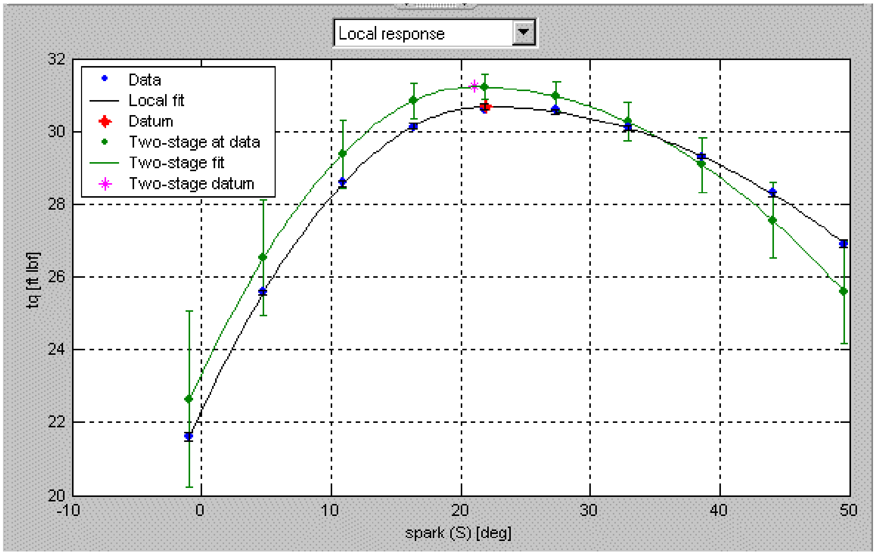
2 Dismiss the **Model Selection** pane, and accept the best model by clicking **Yes** in the **Model Selection** dialog (it is the only two-stage model so far).

3 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.



Click the MLE button in the toolbar to calculate the maximum likelihood estimate.

Maximum Likelihood Estimation

The global models were created in isolation without accounting for any correlations between the response features. Using MLE (maximum likelihood estimation) to fit the two-stage model takes account of possible correlations between response features. In cases where such correlations occur, using MLE significantly improves the two-stage model.

1 You reach the MLE dialog from the local node (PS22 in this case) by

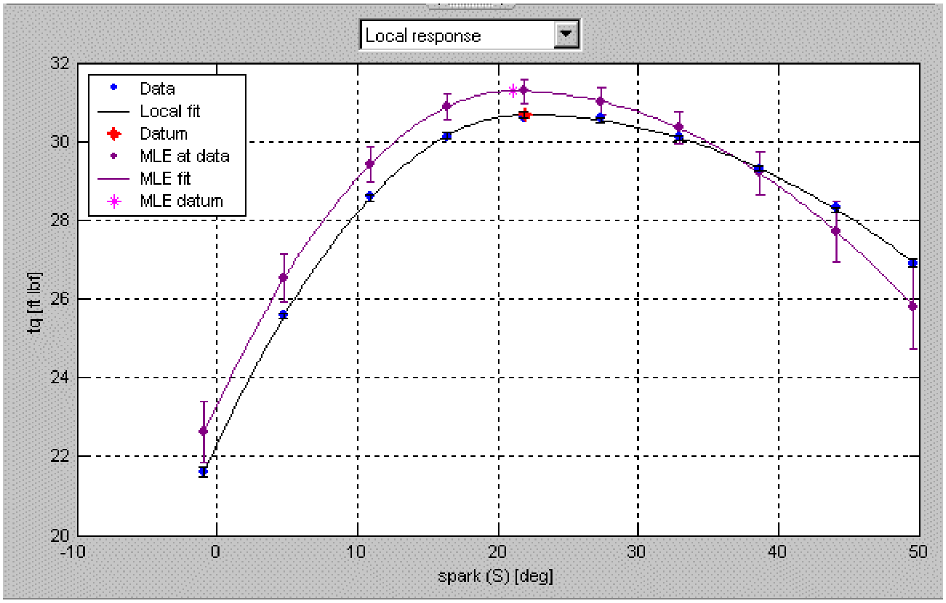
- Clicking the MLE button in the toolbar
- Or by choosing **Model –> Calculate MLE**

2 Leave the algorithm default settings and click **Start** to calculate MLE.

3 Watch the progress indicators until the process finishes and a two-stage RMSE (root mean square error) value appears.

4 Click **OK** to leave the MLE dialog.

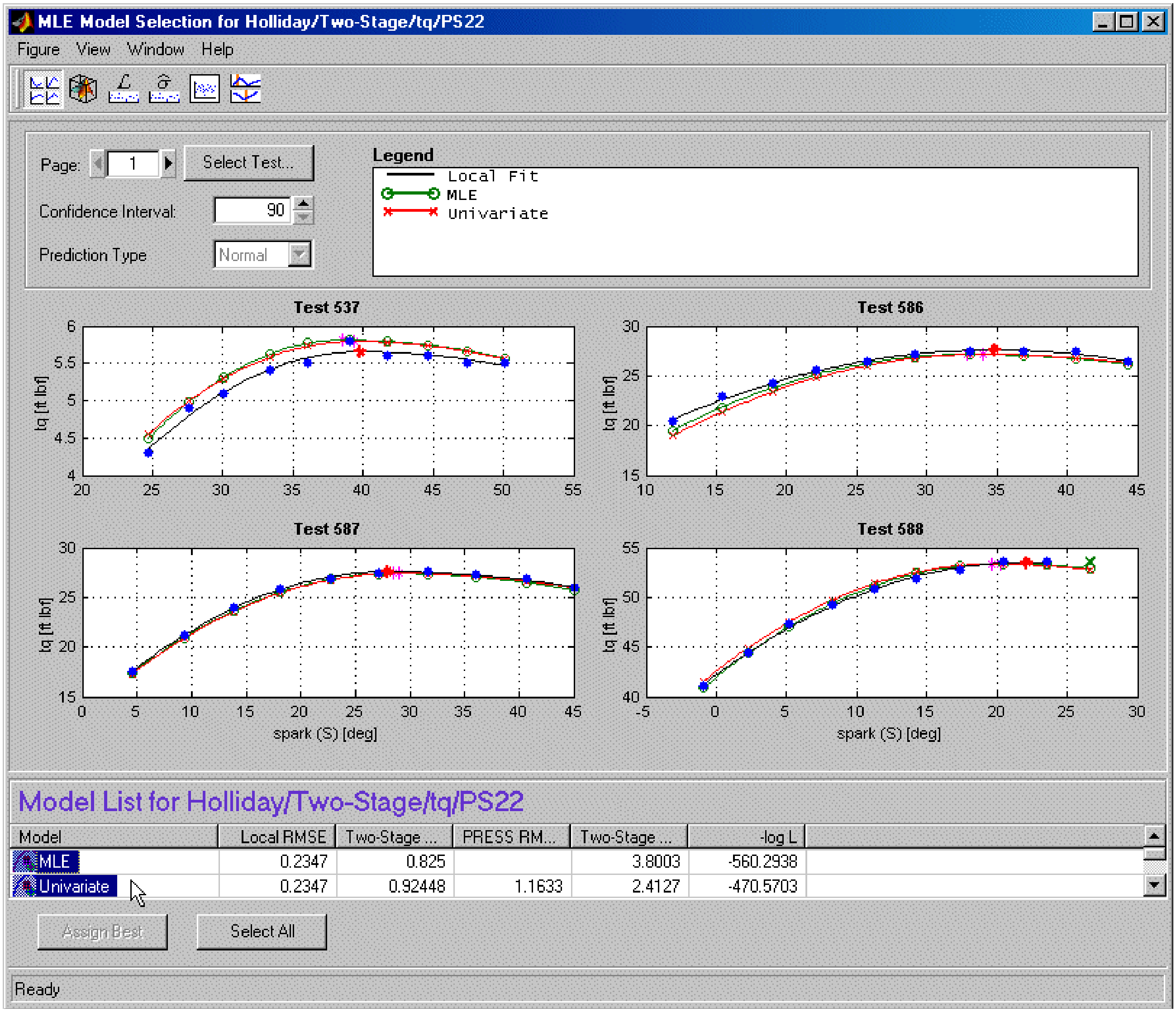
Now the plots on the **Local Model** pane all show the two-stage model in purple to indicate that it is an MLE model. This is also indicated in the legend. Notice that all the model icons in the tree (the response, the local model, and the response features) have also changed to purple to indicate that they are MLE models.



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

Here you can compare MLE with the univariate model previously constructed (without correlations). By default the local fit is plotted against the MLE model.

6 Select both MLE and the Univariate model for plotting by holding down **Shift** while you click the Univariate model in the **Model List** at the bottom of the view.



Page: 1

Confidence Interval: 90

Prediction Type: Normal

Local Fit
MLE
Univariate

Model List for Holliday/Two-Stage/tq/PS22

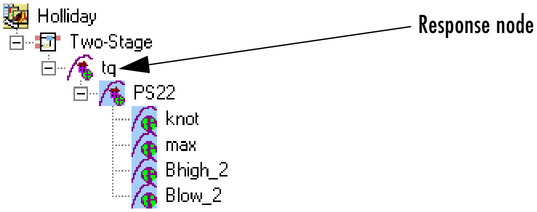
Model	Local RMSE	Two-Stage ...	PRESS RM...	Two-Stage ...	-log L
MLE	0.2347	0.825		3.8003	-560.2938
Univariate	0.2347	0.92448	1.1633	2.4127	-470.5703

Ready

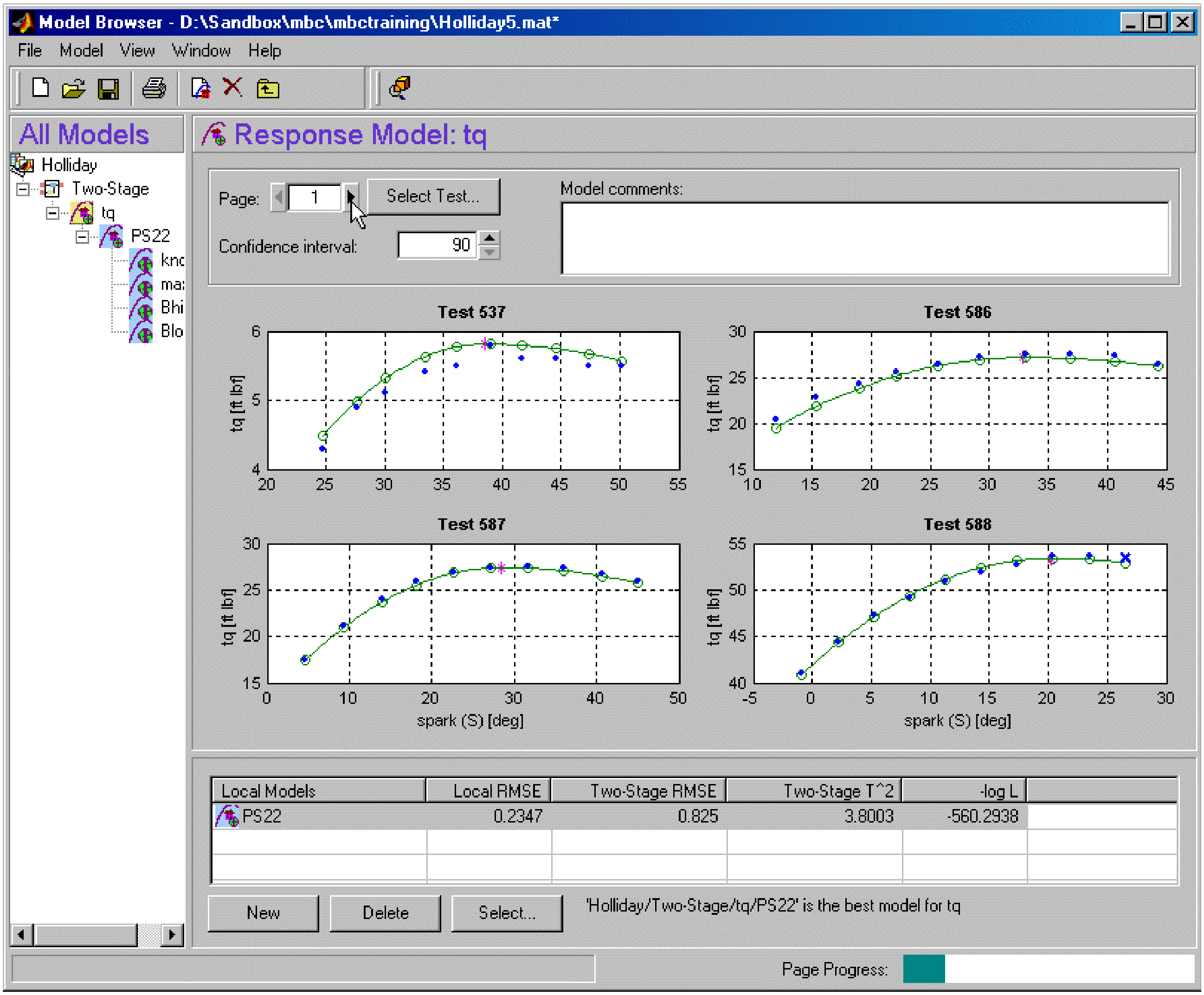
7 Close the **Model Selection** window. Click **Yes** to accept the MLE model as the best.

Response Node

Click the Response node (`tq`) in the Model Browser tree.



Now at the **Response node** in the Model Browser tree (tq), which was previously blank, you see this:



Local Models	Local RMSE	Two-Stage RMSE	Two-Stage T^2	-log L
PS22	0.2347	0.825	3.8003	-560.2938

'Holliday/Two-Stage/tq/PS22' is the best model for tq

This shows you the fit of the two-stage model to the data. You can scroll through the tests, using the arrows at top left, to view the two-stage MLE model (in green) against the data (in blue) for each test.

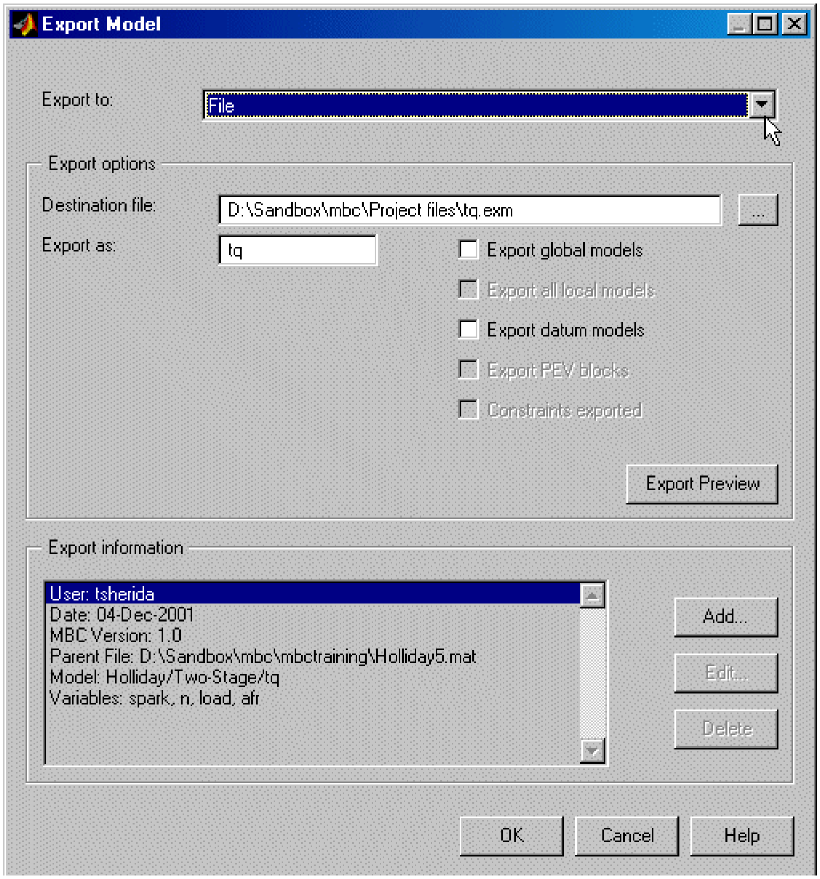
You have now completed setting up and verifying a two-stage model.

Exporting the Model

All models created in the model browser are exported using the **File** menu. A model can be exported to the MATLAB workspace, to a file, or to a Simulink model.

1 Click the `tq` node in the model tree.

2 Choose **File –> Export Models**. The **Export Model** dialog box appears.



Export to: File

Destination file: D:\Sandbox\mbc\Project files\tq.exm

Export as: tq

□ Export global models

□ Export all local models

□ Export datum models

□ Export PEV blocks

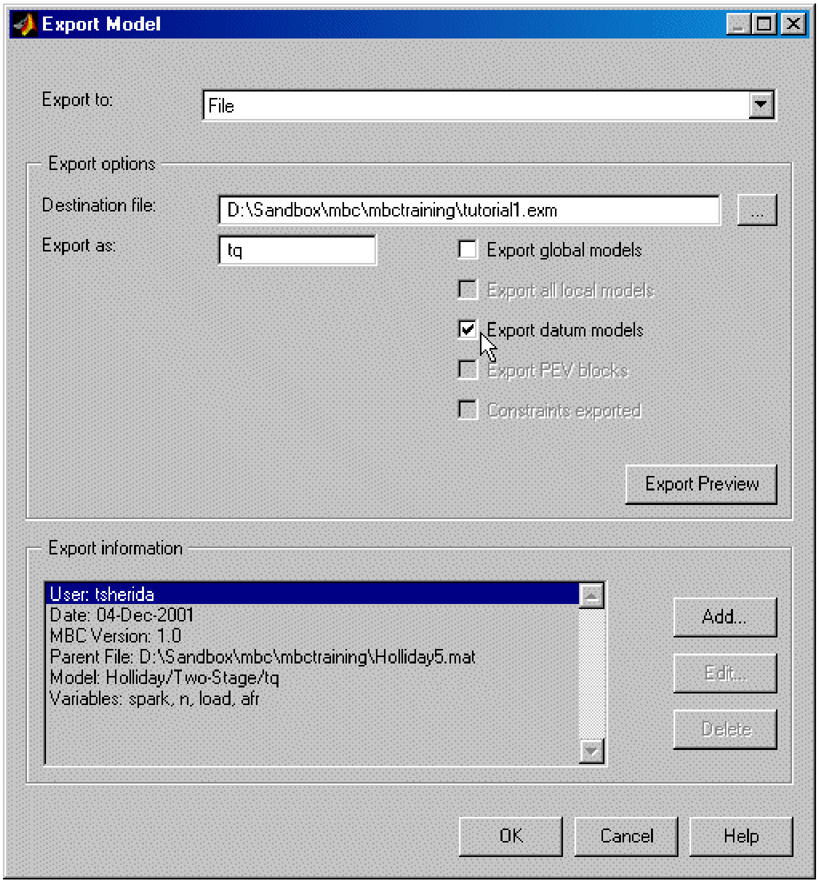
□ Constraints exported

User: tsherida
Date: 04-Dec-2001
MBC Version: 1.0
Parent File: D:\Sandbox\mbc\mbctraining\Holliday5.mat
Model: Holliday/Two-Stage/tq
Variables: spark, n, load, afr

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

4 In the **Export Options** frame, change the destination file to `mbctraining\tutorial1`. Do this by typing directly in the edit box. The **...** Browse button could be used if you want to use an existing file.

5 Ensure that **Export datum models** is selected, as this allows the datum global model to be exported.



Export to: File

Export options

Destination file: D:\Sandbox\mbc\mbctraining\tutorial1.exm

Export as: tq

□ Export global models
□ Export all local models
☑ Export datum models
□ Export PEV blocks
□ Constraints exported

Export information

User: tsherida
Date: 04-Dec-2001
MBC Version: 1.0
Parent File: D:\Sandbox\mbc\mbctraining\Holliday5.mat
Model: Holliday/Two-Stage/tq
Variables: spark, n, load, afr

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

Design of Experiment Tutorial

What Is Design of Experiment? 3-4
Design Styles 3-5

Starting the Design Editor 3-6

Optimal Designs 3-9
Start Point Tab 3-10
Candidate Set Tab 3-11
Algorithm Tab 3-14

Design Displays 3-16

Prediction Error Variance Viewer 3-19
Improving the Design 3-21

Classical Designs 3-24
Classical Design Browser 3-26
Set Up and View a Classical Design 3-27

Design Evaluation Tool 3-30
Improving the Design 3-32

Space-Filling Designs 3-33
Setting Up a Space-Filling Design 3-34

Applying Constraints 3-35
Saving Designs 3-40

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. The first section introduces the concept of design of experiment, and outlines the advantages of good experimental design for mapping modern engines. Then there is a description of the design styles available in the Model-Based Calibration Toolbox.

   - “What Is Design of Experiment?” on page 3-4
   - “Design Styles” on page 3-5

2. 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 3-6
   - “Optimal Designs” on page 3-9
   - “Design Displays” on page 3-16
   - “Prediction Error Variance Viewer” on page 3-19
   - “Saving Designs” on page 3-40
   - “Improving the Design” on page 3-21

3. 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 3-24
   - “Design Evaluation Tool” on page 3-30

4. 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 3-33
- “Applying Constraints” on page 3-35

For more details on functionality in the Design Editor, see “The Design Editor” on page 5-153.

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 3-9.

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 3-33.

Classical designs (including full factorial) are very well researched and are suitable for simple regions (hypercube or sphere). Engines have complex constraints and models (high-order polynomials and splines). See “Classical Designs” on page 3-24.

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.

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.

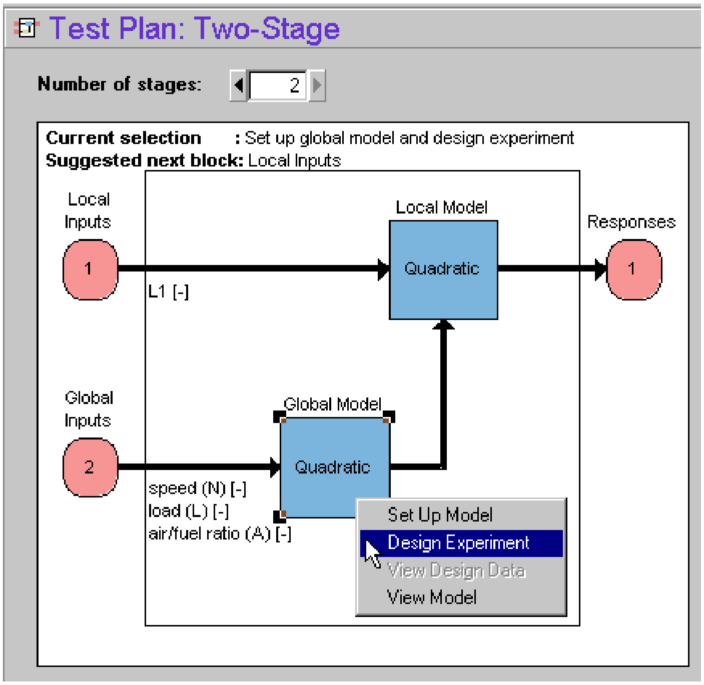
There is only one input to the global model by default. To increase the number of input factors:

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 Factor**s 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:

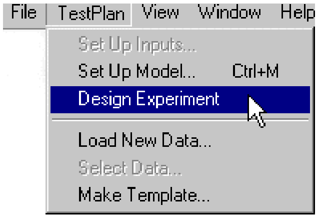
- Right-click the global model in the diagram and choose **Design Experiment**, as shown.
- You can also access the Design Editor by selecting the menu item **TestPlan –> Design Experiment**.



Test Plan: Two-Stage

Number of stages: 2

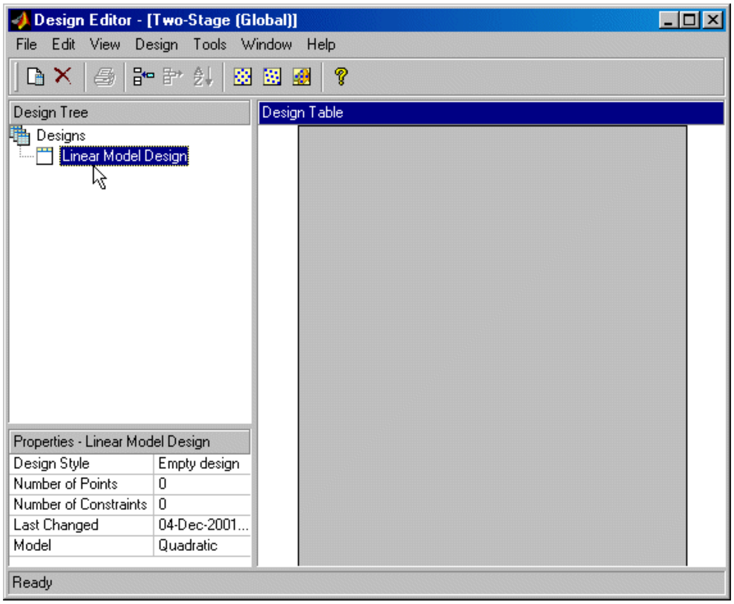
Current selection : Set up global model and design experiment
Suggested next block: Local Inputs



The **Design Editor** window appears.

Creating a New Design

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


Design Editor - [Two-Stage (Global)]
File Edit View Design Tools Window Help
Design Tree
Designs
Linear Model Design
Design Table
Ready

Properties - Linear Model Design

Design Style	Empty design
Number of Points	0
Number of Constraints	0
Last Changed	04-Dec-2001...
Model	Quadratic

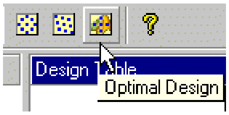
2 The new `Linear Model Design` node is automatically selected. An empty Design Table appears (see above) because you have not yet chosen a design. For this example you create an optimal design for the default global model, which is a quadratic.

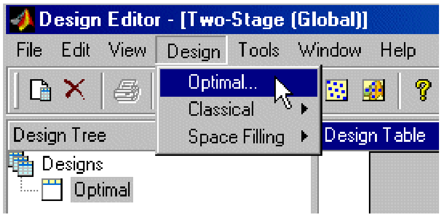
You can change the model for which you are designing an experiment from within the **Design Editor** window by selecting **Edit –> Model**.

3 Rename the new node `Optimal` (you can edit the names by clicking again on a node when it is already selected, or by pressing **F2**, as when selecting to rename in Windows Explorer).

Optimal Designs

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




Design Editor - [Two-Stage (Global)]
File Edit View Design Tools Window Help
Optimal...
Classical
Space Filling
Design Tree
Designs
Optimal
Design Table

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:

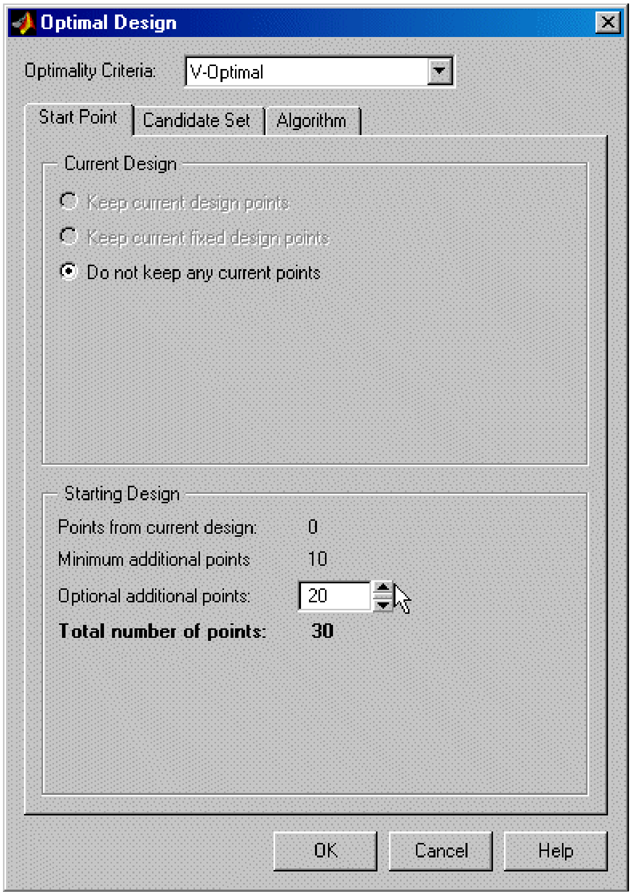
- Starting point and number of points in the design
- Candidate set of points from which the design points are chosen

- 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.



Optimality Criteria: V-Optimal

Current Design

○ Keep current design points

○ Keep current fixed design points

● Do not keep any current points

Starting Design

Points from current design: 0

Minimum additional points 10

Optional additional points: 20

Total number of points: 30

Candidate Set Tab

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

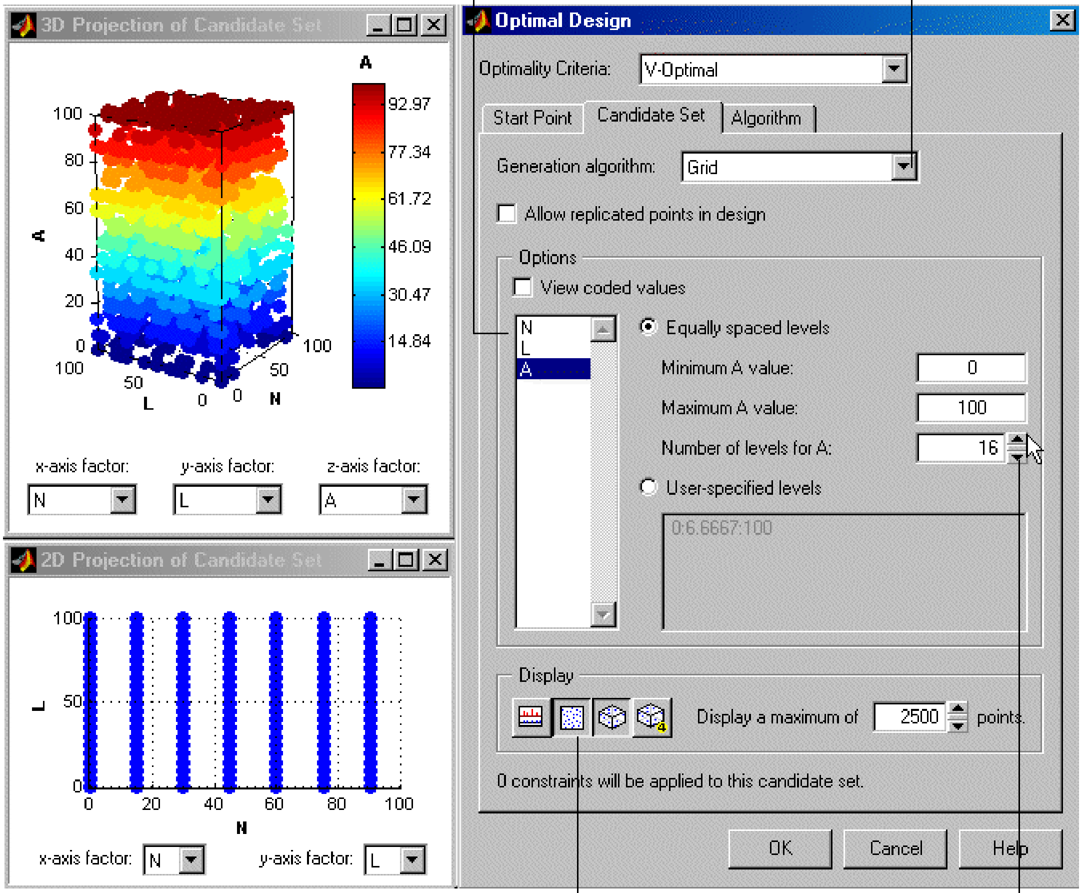
1 Choose **Grid** for this example. Note that you could choose different schemes for different factors.

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

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

Select variables in this list

Choose Grid from this list



x-axis factor: N

y-axis factor: L

z-axis factor: A

x-axis factor: N

y-axis factor: L

Optimality Criteria: V-Optimal

Generation algorithm: Grid

☐ Allow replicated points in design

Options

☐ View coded values

N
L
A

● Equally spaced levels

Minimum A value: 0

Maximum A value: 100

Number of levels for A: 16

○ User-specified levels

0:6.6667:100

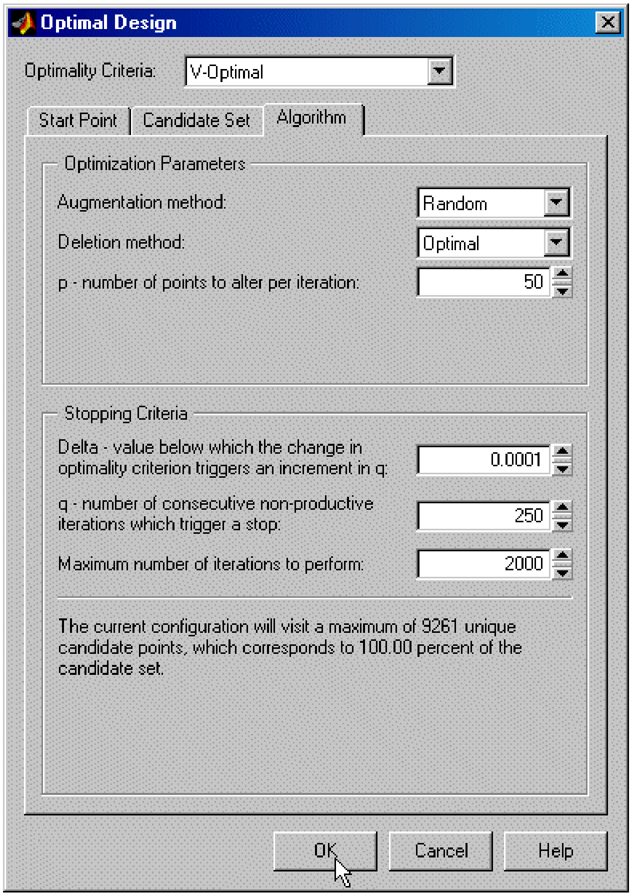
Display a maximum of 2500 points.

0 constraints will be applied to this candidate set.

Open display windows with these buttons

Change the number of levels of the selected variable here

Algorithm Tab



Optimality Criteria: V-Optimal

Optimization Parameters

Augmentation method: Random

Deletion method: Optimal

p - number of points to alter per iteration: 50

Stopping Criteria

Delta - value below which the change in optimality criterion triggers an increment in q: 0.0001

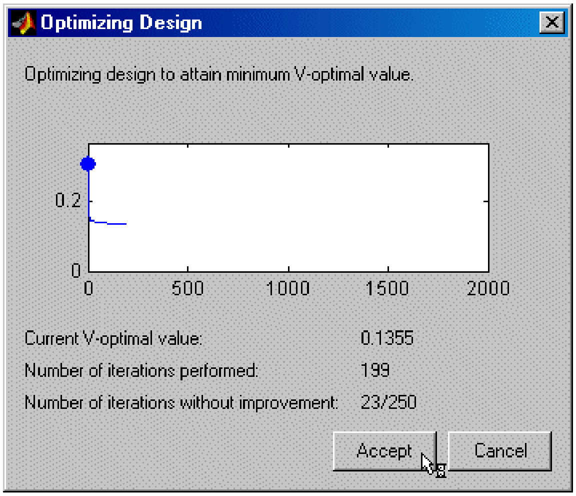
q - number of consecutive non-productive iterations which trigger a stop: 250

Maximum number of iterations to perform: 2000

The current configuration will visit a maximum of 9261 unique candidate points, which corresponds to 100.00 percent of the candidate set.

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

When you click the **OK** button on the **Optimal Design** dialog, the **Optimizing Design** dialog appears, containing a graph. This dialog shows the progress of the optimization and has two buttons: **Accept** and **Cancel**. **Accept** stops the optimization early and takes the current design from it. **Cancel** stops the optimization and reverts to the original design.



Optimizing Design

Optimizing design to attain minimum V-optimal value.

Current V-optimal value: 0.1355

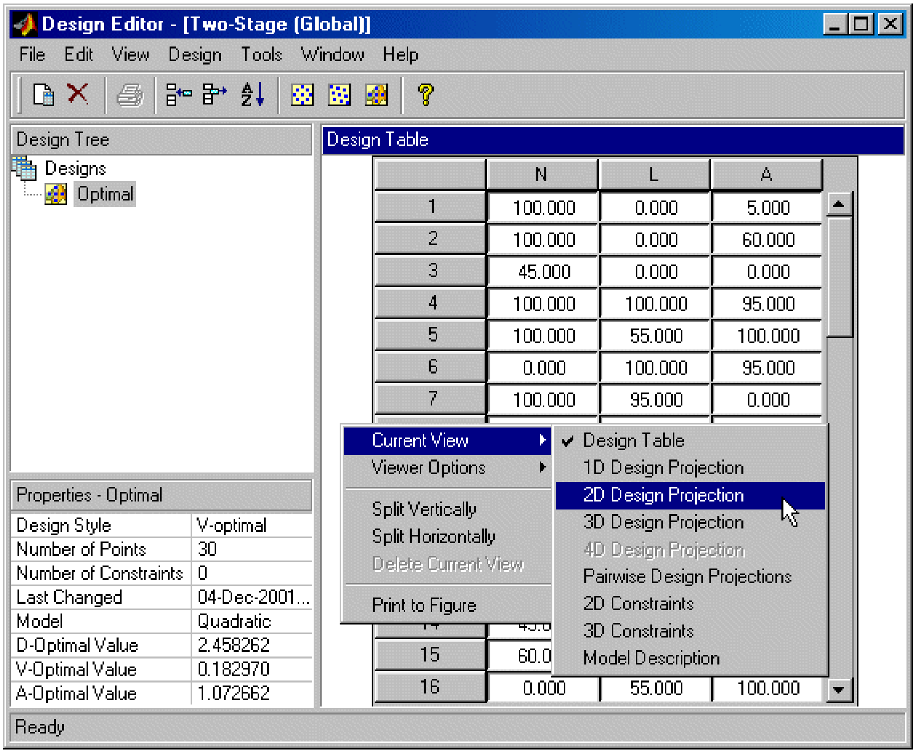
Number of iterations performed: 199

Number of iterations without improvement: 23/250

2 Click **Accept** when iterations are not producing noticeable improvements; that is, the graph becomes very flat.

Design Displays

When you press the **Accept** button, you return to the **Design Editor**.



	N	L	A
1	100.000	0.000	5.000
2	100.000	0.000	60.000
3	45.000	0.000	0.000
4	100.000	100.000	95.000
5	100.000	55.000	100.000
6	0.000	100.000	95.000
7	100.000	95.000	0.000
15	60.0		
16	0.000	55.000	100.000

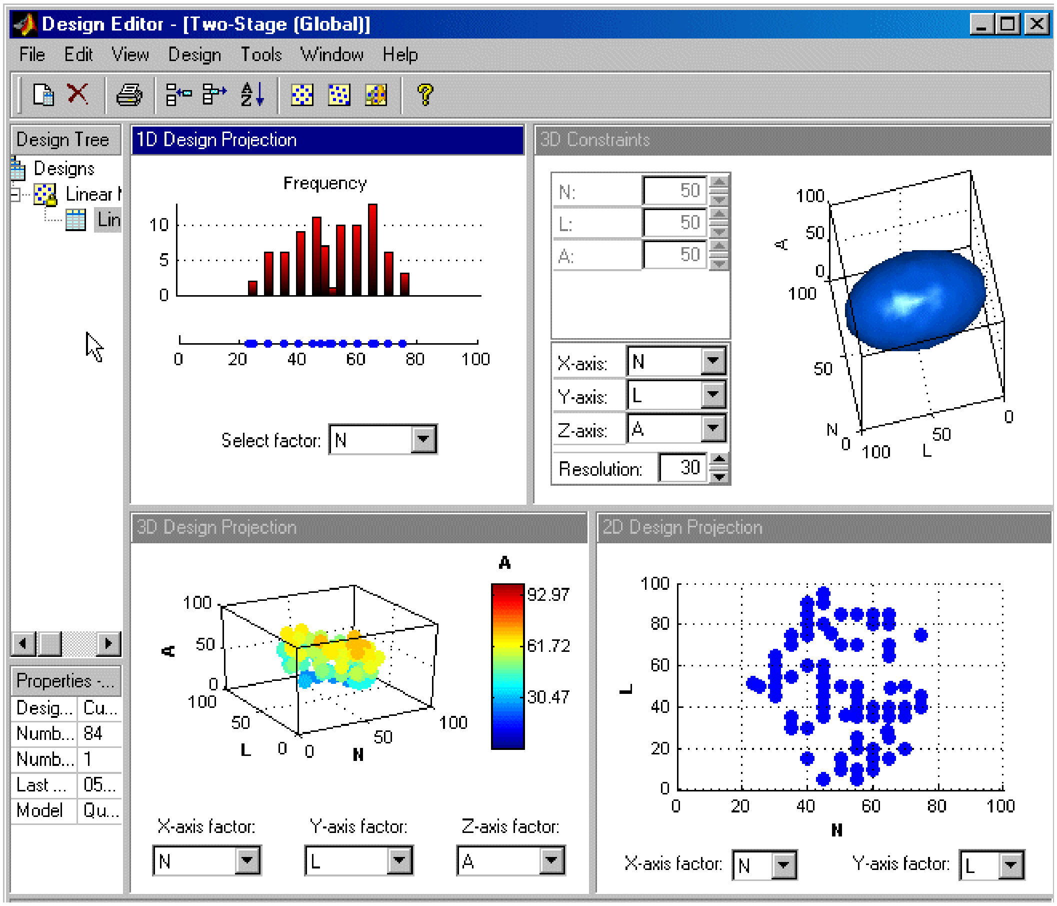
Design Style	V-optimal
Number of Points	30
Number of Constraints	0
Last Changed	04-Dec-2001...
Model	Quadratic
D-Optimal Value	2.458262
V-Optimal Value	0.182970
A-Optimal Value	1.072662

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, in which you can change the view of the design to **1-D**, **2-D**, **3-D**, and **4-D Projections** and **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. The split can also 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 5-156.

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.



Design Editor - [Two-Stage (Global)]

N: 50
L: 50
A: 50
X-axis: N
Y-axis: L
Z-axis: A
Resolution: 30

Select factor: N

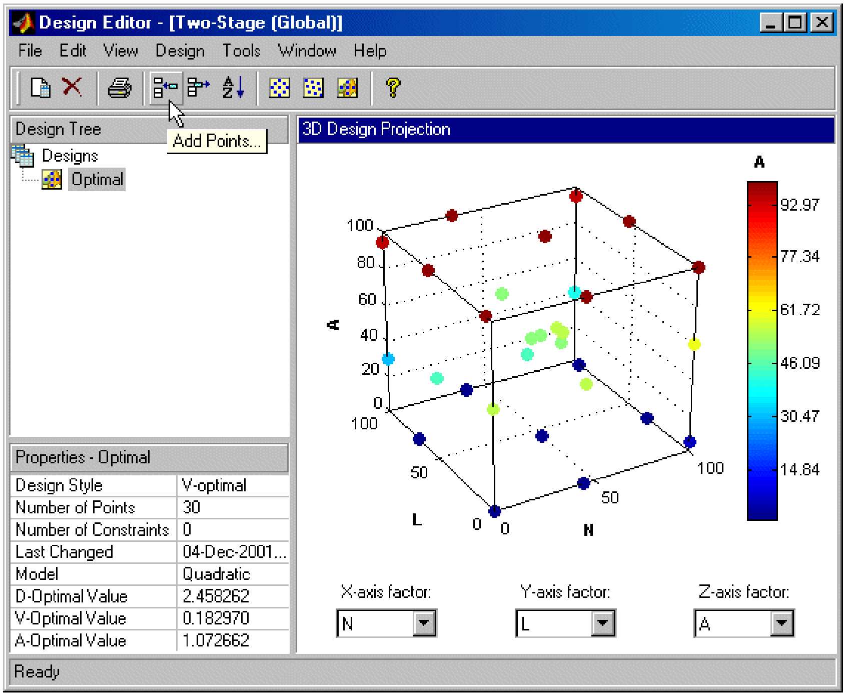
X-axis factor: N
Y-axis factor: L
Z-axis factor: A

X-axis factor: N
Y-axis factor: L

Properties -...	
Desig...	Cu...
Numb...	84
Numb...	1
Last ...	05...
Model	Qu...

The current view and options for the current view are available either through the context menu or the **View** menu on the **Design Editor** window.

1 Change the main display to 3-D Projection view.



Properties - Optimal

Design Style	V-optimal
Number of Points	30
Number of Constraints	0
Last Changed	04-Dec-2001...
Model	Quadratic
D-Optimal Value	2.458262
V-Optimal Value	0.182970
A-Optimal Value	1.072662

X-axis factor: N

Y-axis factor: L

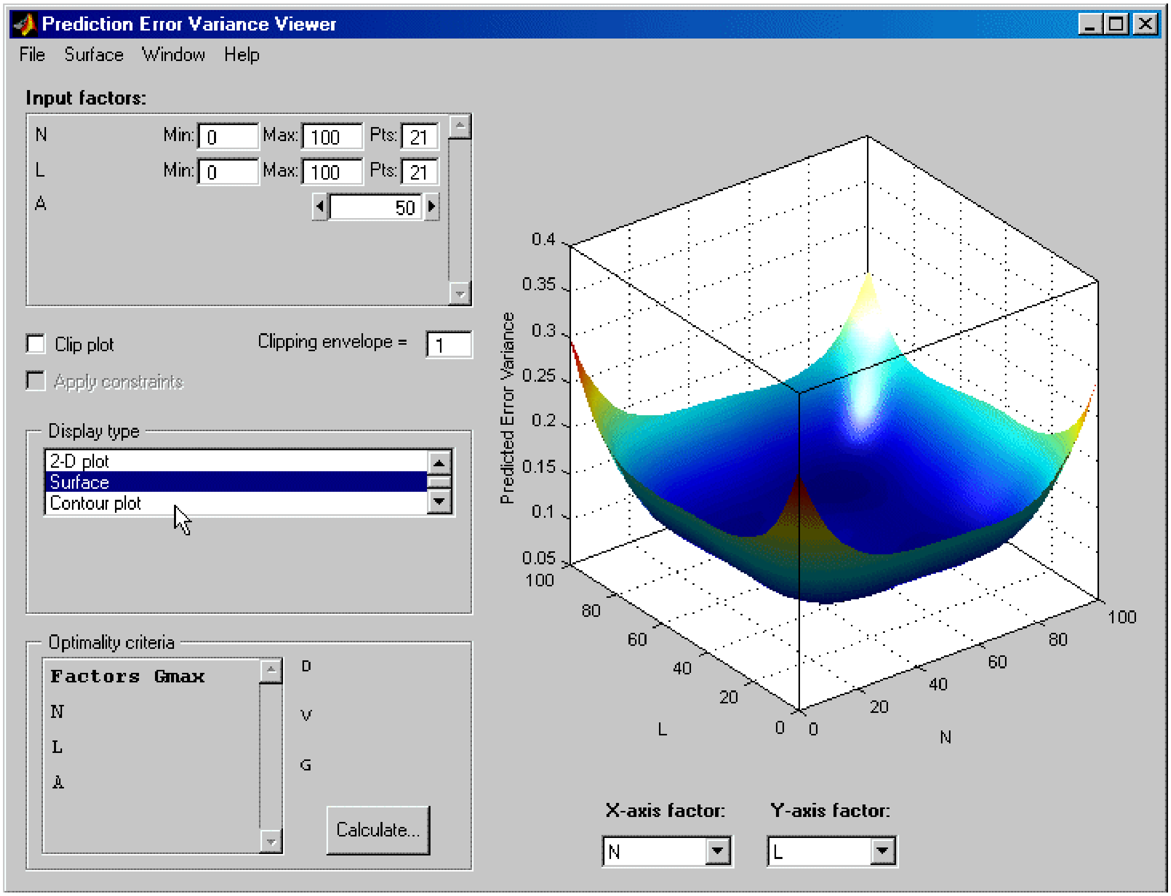
Z-axis factor: A

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

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**.



Prediction Error Variance Viewer

Input factors:

N Min: 0 Max: 100 Pts: 21

L Min: 0 Max: 100 Pts: 21

A 50

□ Clip plot   Clipping envelope = 1

□ Apply constraints

Display type: Surface

Factors Gmax

N

L

A

X-axis factor: N   Y-axis factor: L

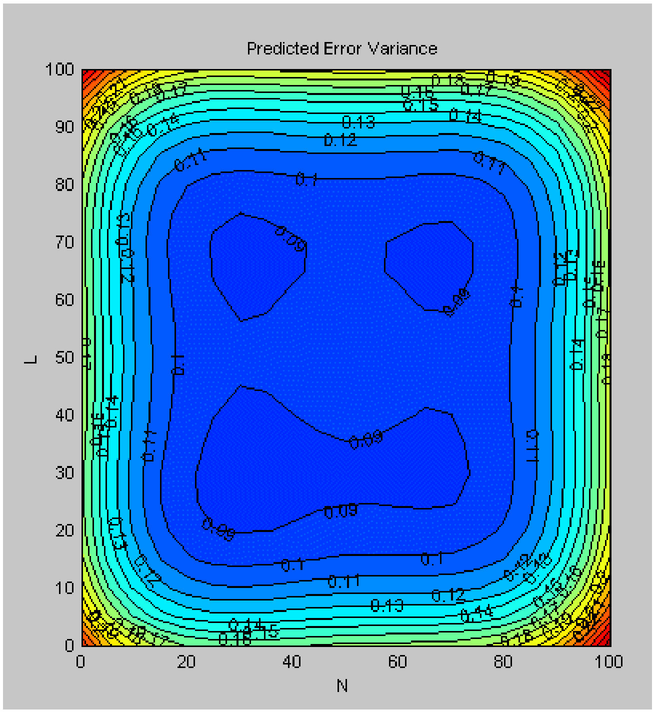
The default view is a 3-D plot of the PEV surface.

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 model has the highest error. Even in the corners the errors are small; looking at the scale, PEV values under 0.25 are not large.

Try the other display options.

- The **Surface** 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).

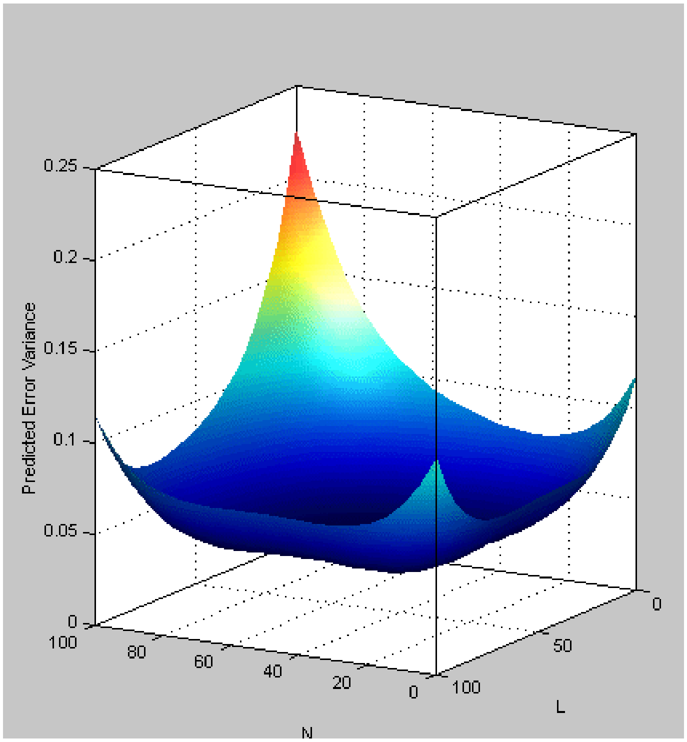
To see more dramatic changes to the design, return to the **Design Editor** window (no need to close the PEV viewer).

1 Split the display so you can see a 3-D projection at the same time as a Table view.

2 Choose **Edit –> Delete Point**.

3 Using the Table and 3-D views as a guide, in the **Delete Points** dialog, pick six points to remove along one corner; for example, pick points where N is 100 and L is 0. Add the relevant point numbers to the delete list by clicking the add (>) button.

4 Click **OK** to remove the points. See the changes in the main design displays and look at the new Surface plot in the PEV viewer (see the example following).


Predicted Error Variance
0.25
0.2
0.15
0.1
0.05
0
100
80
60
40
20
0
N
100
50
0
L

Classical Designs

1 In the **Design Editor** window, select the Optimal design in the design tree by clicking.

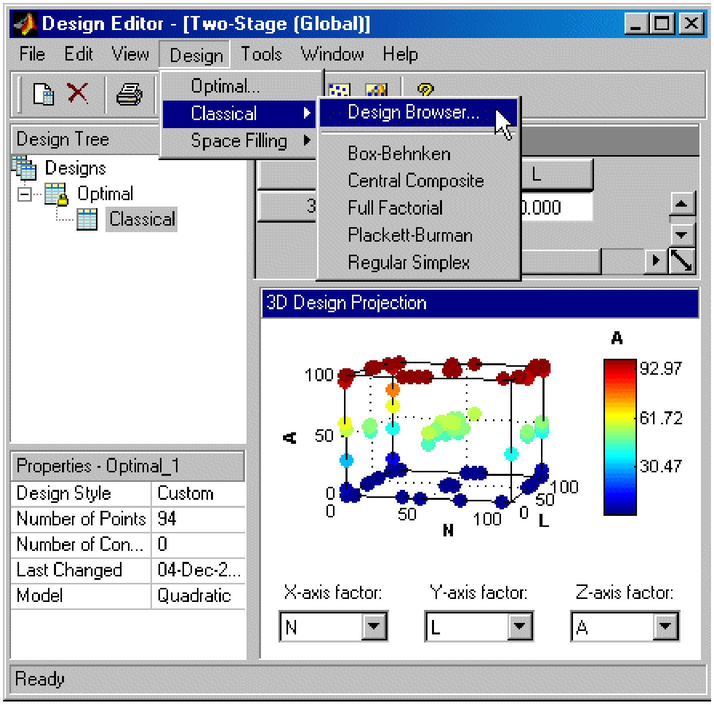
2 Add a new design. Use the first toolbar button, or select **File –> New**.

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**.



Properties - Optimal_1	
Design Style	Custom
Number of Points	94
Number of Con...	0
Last Changed	04-Dec-2...
Model	Quadratic

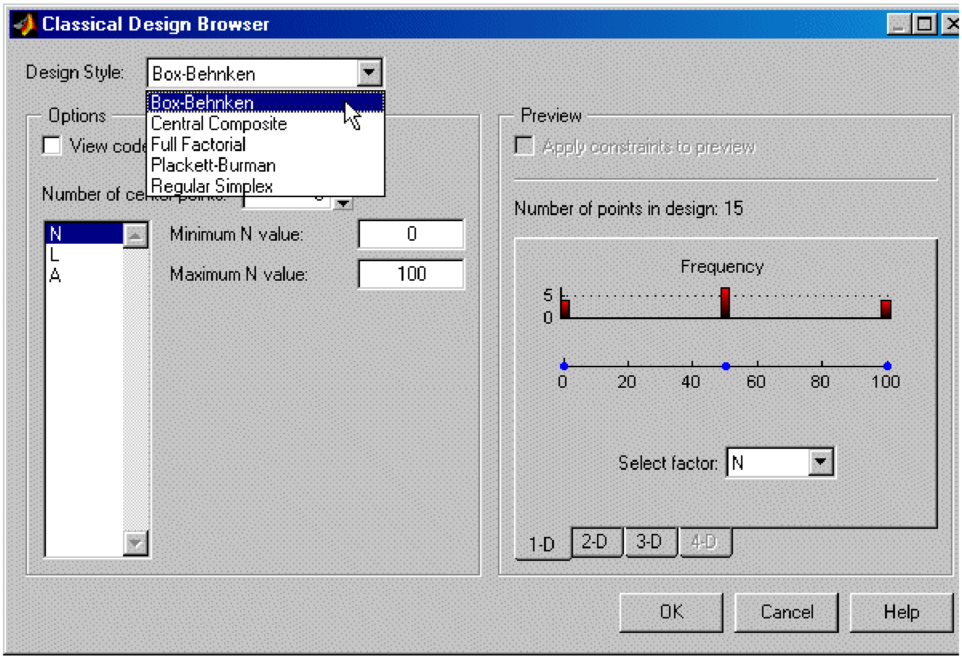
Note In cases where the preferred type of classical design is known, you can go straight to one of the five options under **Design –> Classical**. Choosing the **Design Browser** option allows you to see graphical previews of these same five options before making a choice.

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

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

The **Classical Design Browser** appears.

Classical Design Browser



Design Style: Box-Behnken

Minimum N value: 0

Maximum N value: 100

Number of points in design: 15

Select factor: N

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**

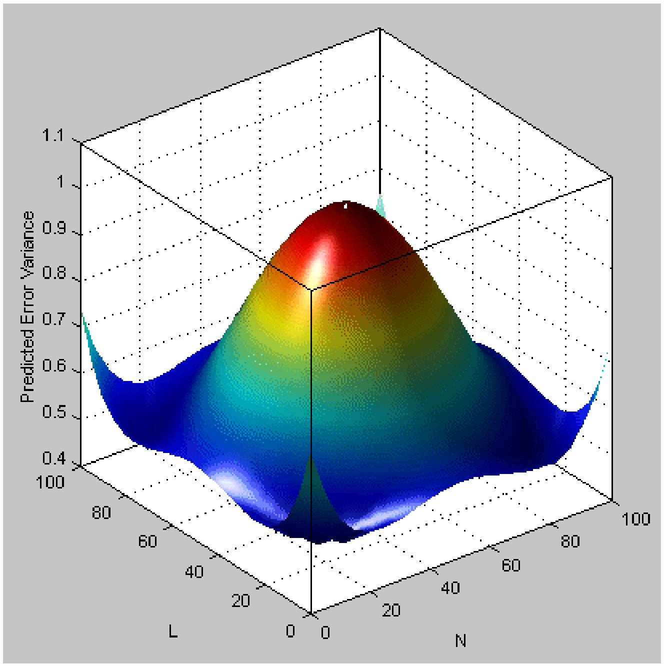
  Generates an *n*-dimensional grid of points. You can choose the number of levels for each factor and the number of additional center points to add.
- **Plackett Burman**

  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.

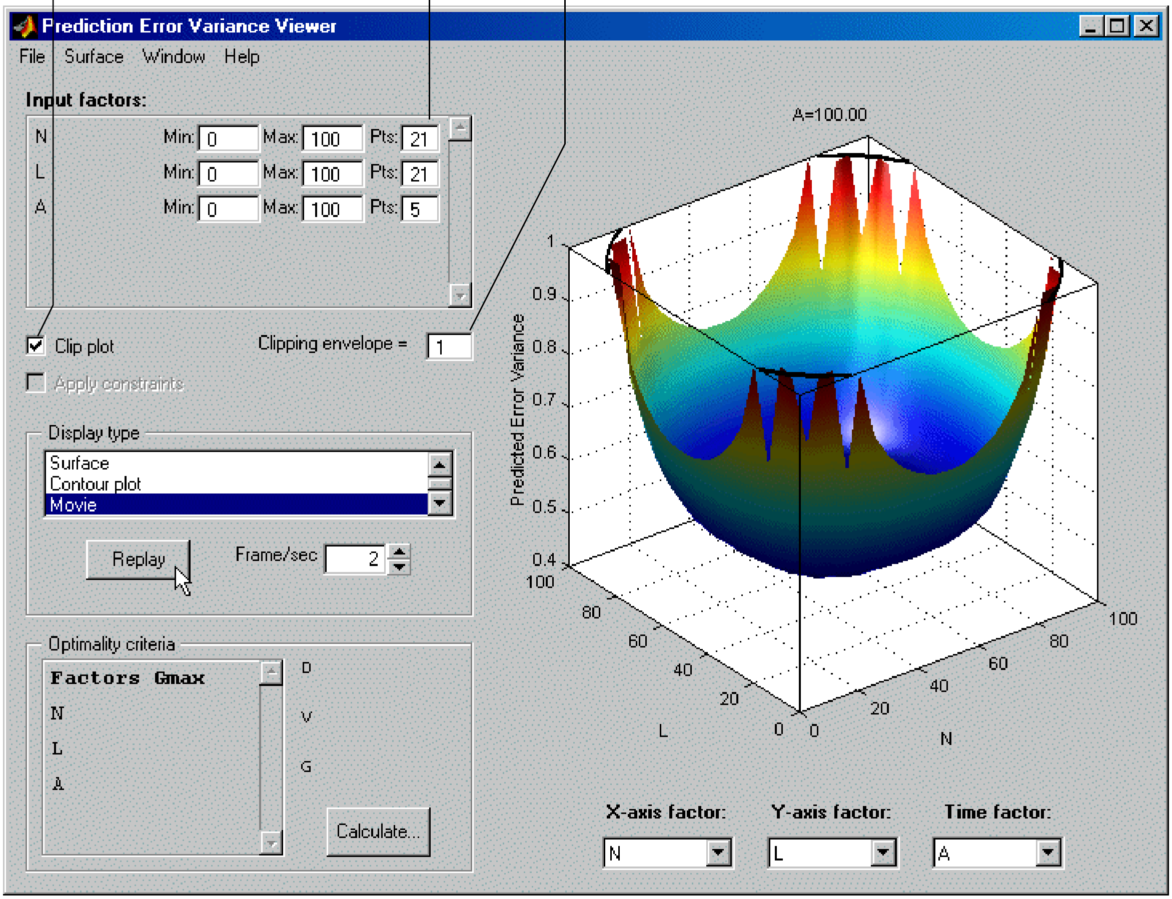
6 Select **Movie**, and you see this white contour line as the surface moves through the plane of value 1.

7 Select the **Clip Plot** check box. Areas that move above the value of 1 are removed. The edges of the clip are very jagged; you can make it smoother by increasing the numbers of points plotted for each factor. See the following example.

Turn clipping on and off here

Change number of points plotted here

Change clipping value here



Input factors:

N	Min: 0	Max: 100	Pts: 21
L	Min: 0	Max: 100	Pts: 21
A	Min: 0	Max: 100	Pts: 5

☑ Clip plot

Clipping envelope = 1

□ Apply constraints

Display type

- Surface
- Contour plot
- Movie

Frame/sec 2

```
Factors Gmax
N
L
A
```

X-axis factor: N

Y-axis factor: L

Time factor: A

Design Evaluation Tool

The Design Evaluation Tool is only available for linear models. See also “Global Model Class: Multiple Linear Models” on page 5-43.

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.

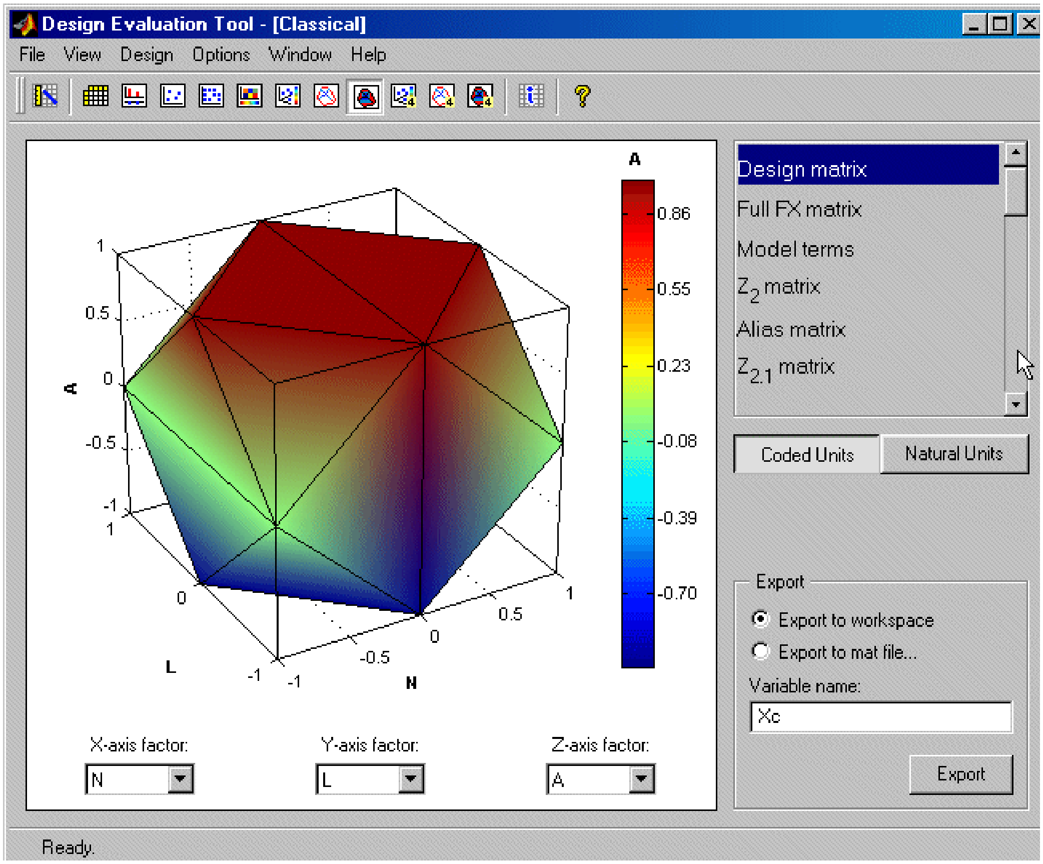
3 Select **Hat Matrix** from the list on the right.

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 6-27.

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.

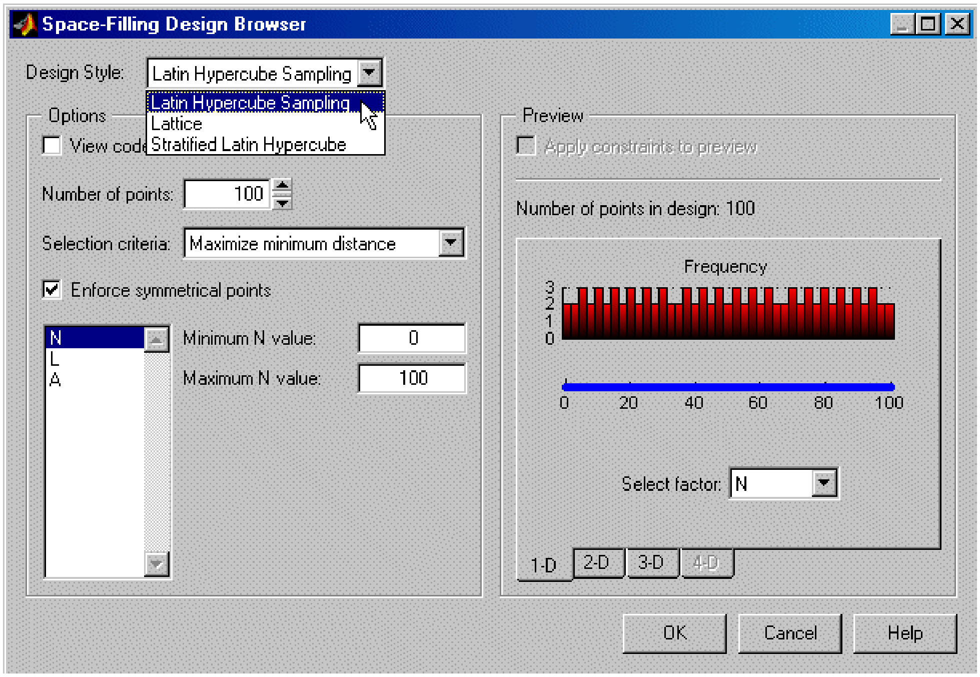
2 Rename the new node `Space Filling` (click again or press **F2**).

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.



Space-Filling Design Browser

Design Style: Latin Hypercube Sampling

Options

□ View code

Number of points: 100

Selection criteria: Maximize minimum distance

☑ Enforce symmetrical points

Minimum N value: 0

Maximum N value: 100

Preview

□ Apply constraints to preview

Number of points in design: 100

Select factor: N

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.

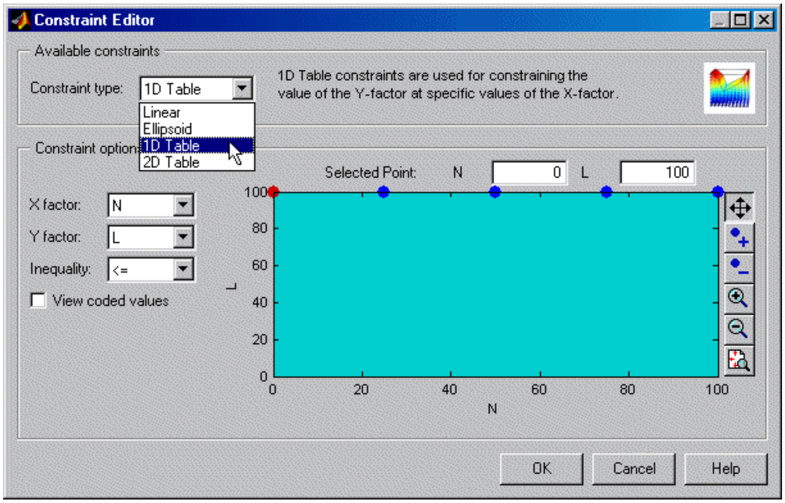
Designs can have any number of geometric constraints placed upon them. Each constraint can be one of four types: an ellipsoid, a hyperplane, a 1-D lookup table, or a 2-D lookup table. For further details, see "Applying Constraints" on page 5-184 in the "GUI Reference" for a full description of the constraint 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.



Constraint Editor

Available constraints

Constraint type: 1D Table

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

Constraint options

Selected Point: N 0 L 100

X factor: N

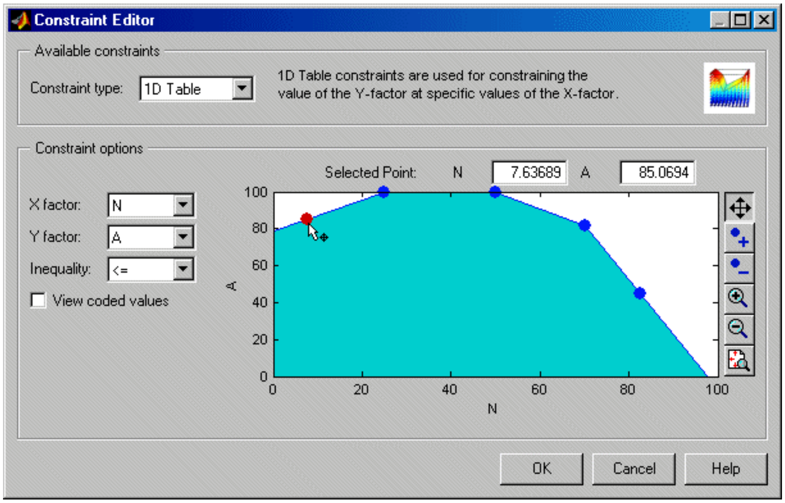
Y factor: L

Inequality: <=

□ View coded values

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

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



Constraint Editor

Available constraints

Constraint type: 1D Table

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

Constraint options

Selected Point: N 7.63689 A 85.0694

X factor: N

Y factor: A

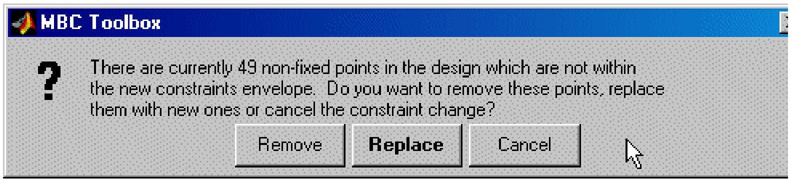
Inequality: <=

□ View coded values

6 Click **OK**.

Your new constraint appears in the **Constraint Manager** list box. Click **OK** to return to the **Design Editor**. A dialog appears because there are points in the design that fall outside your newly constrained candidate set. You can simply delete them or cancel the constraint. Note that fixed points are not deleted by this process.

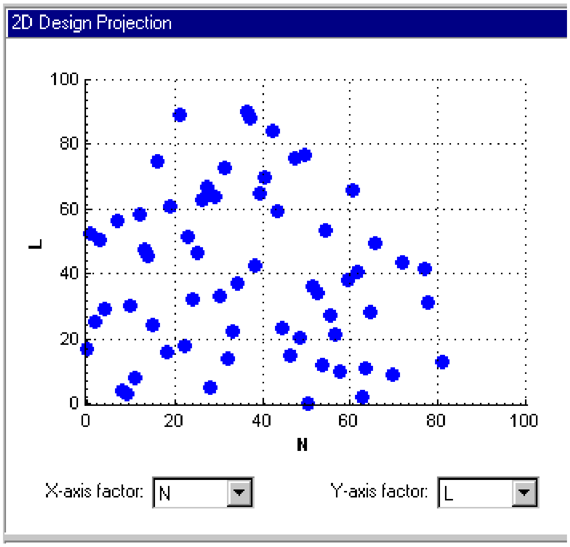
For optimal designs you get the dialog shown, where you also have the option to replace the points with new ones chosen (optimally if possible) within the new candidate set.



MBC Toolbox

There are currently 49 non-fixed points in the design which are not within the new constraints envelope. Do you want to remove these points, replace them with new ones or cancel the constraint change?

7 The default is to remove the points outside the new constraint area; choose this.



2D Design Projection

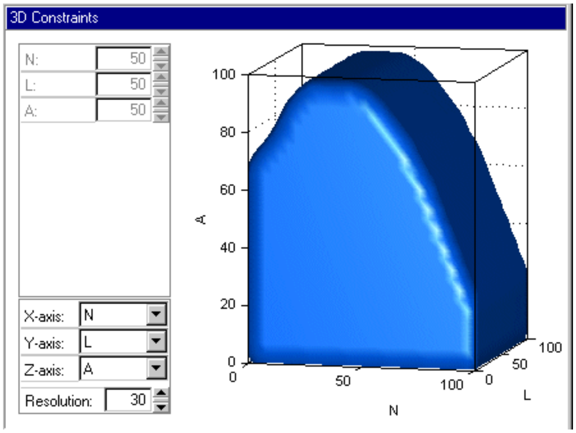
X-axis factor: N

Y-axis factor: L

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.

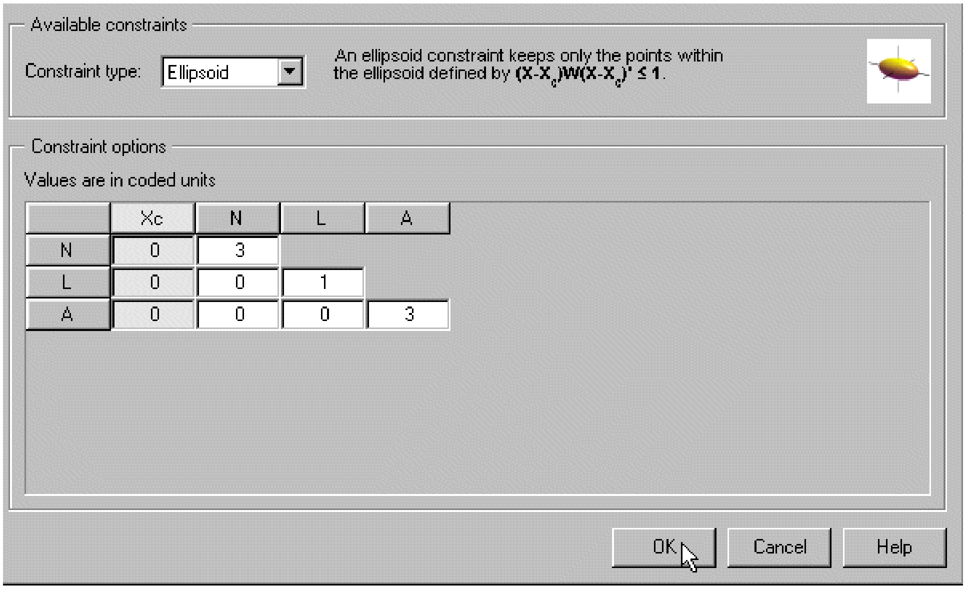
9 Select **Current View –> 3D Constraints**.



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.



Constraint type: Ellipsoid

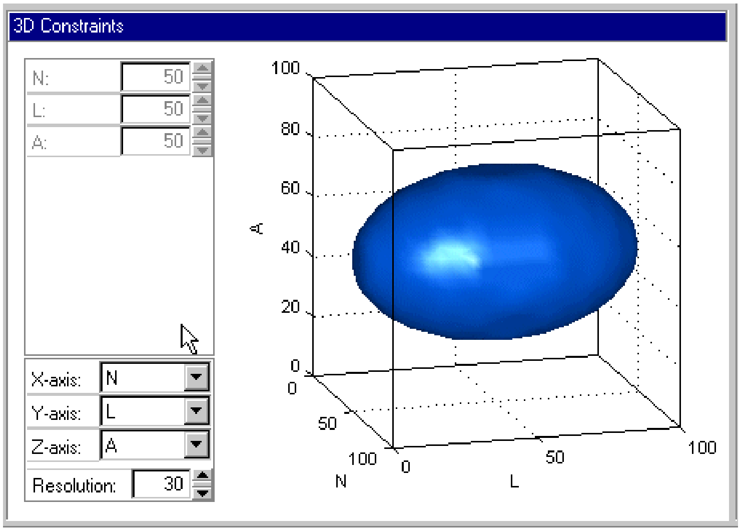
An ellipsoid constraint keeps only the points within the ellipsoid defined by $(X-X_c)W(X-X_c)' \le 1$.

Values are in coded units

	Xc	N	L	A
N	0	3		
L	0	0	1	
A	0	0	0	3

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

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.



3D Constraints

N: 50
L: 50
A: 50

X-axis: N
Y-axis: L
Z-axis: A

Resolution: 30

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.

Data Editor Tutorial

Loading the Data 4-3
Entering the Data Editor 4-3
Loading a Data File 4-4

Viewing and Editing the Data 4-6
User-Defined Variables & Filtering 4-8
Defining a New Variable 4-9
Applying a Filter 4-10
Sequence of Variables 4-12
Deleting and Editing Variables and Filters 4-13
Storage 4-14
Test Groupings 4-15

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. The Data Editor can create an Excel sheet for data input, set up in the form the Model Browser expects, and can then load this data. 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 you can store and retrieve these user-defined variables and filters for any data set. 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 6-73 in the "Technical Documents"section.

For comprehensive help on all data functions in the Model Browser, see "Data" on page 5-59 in the "GUI Reference" section.

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

Loading the Data

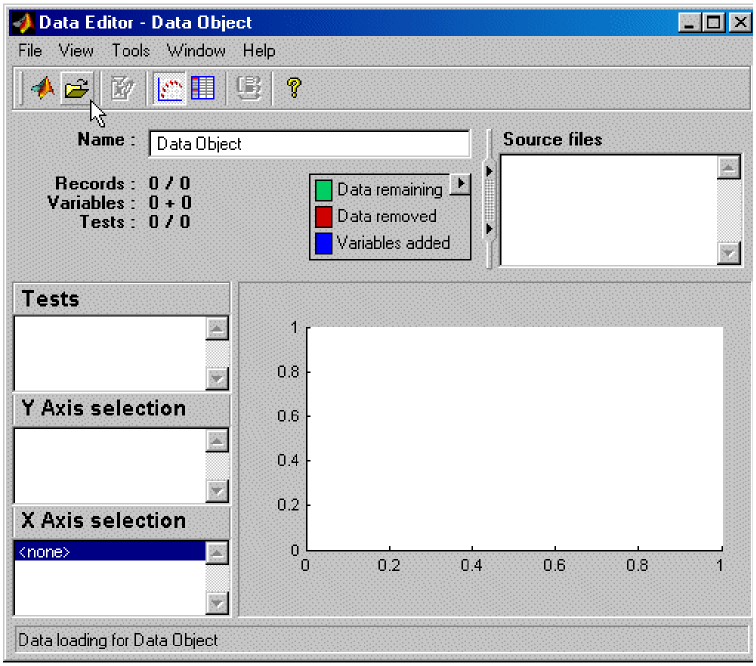
Entering the Data Editor

To enter the Data Editor and create a new data object, select one of these options:

- From the Test Plan level, choose **TestPlan –> Load New Data**.
- Alternatively, from the Project node, select **Tools –> 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.



Name : Data Object

Records : 0 / 0
Variables : 0 + 0
Tests : 0 / 0

Data loading for Data Object

There is no plot until some data has been loaded.

By default the new data object is called **Data Object**. You can change this name by typing in the **Name** edit box at the top.

You can change the names of data objects at the Project node 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.

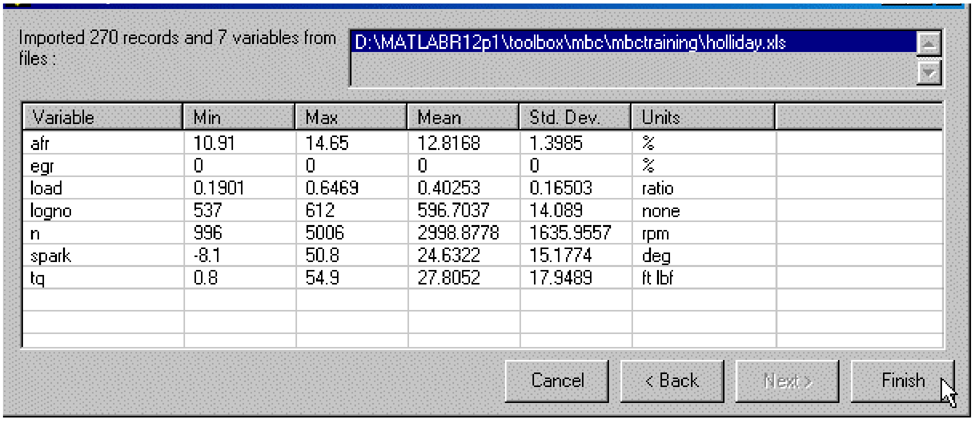
Loading a Data File

1 Click the Open File icon in the toolbar 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, concerto, `.mat`). 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**.



Imported 270 records and 7 variables from files :

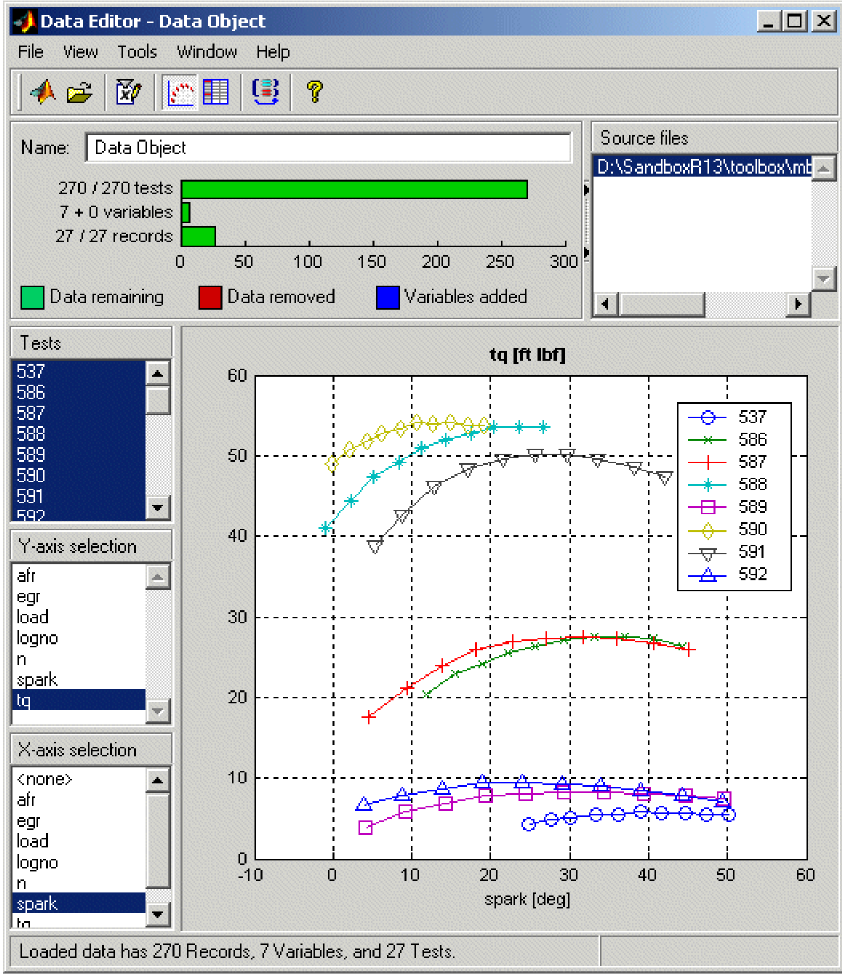
D:\MATLABR12p1\toolbox\mbc\mbctraining\holliday.xls

Variable	Min	Max	Mean	Std. Dev.	Units	
afr	10.91	14.65	12.8168	1.3985	%	
egr	0	0	0	0	%	
load	0.1901	0.6469	0.40253	0.16503	ratio	
logno	537	612	596.7037	14.089	none	
n	996	5006	2998.8778	1635.9557	rpm	
spark	-8.1	50.8	24.6322	15.1774	deg	
tq	0.8	54.9	27.8052	17.9489	ft lbf	

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



Name: Data Object

270 / 270 tests

7 + 0 variables

27 / 27 records

🟩 Data remaining 🟥 Data removed 🟦 Variables added

D:\SandboxR13\toolbox\mb

Tests: 537, 586, 587, 588, 589, 590, 591, 592

Y-axis selection: afr, egr, load, logno, n, spark, tq

X-axis selection: <none>, afr, egr, load, logno, n, spark, tq

Loaded data has 270 Records, 7 Variables, and 27 Tests.

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. You can multiple-select tests and *y*-axes to compare the data in the tests (hold down **Shift** or **Control**).

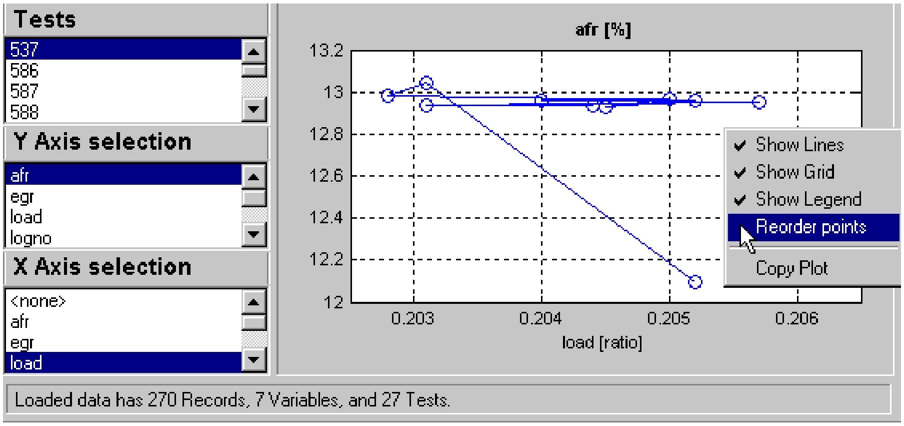
Right-click the plot to change the display: you can switch grid lines on and off and plot lines to join the data points.

Reorder Points in the right-click menu can be useful when record order does not produce a sensible line joining the data points. For an illustration of this:

1 Choose `afr` for the *y*-axis.

2 Choose `Load` for the *x*-axis.

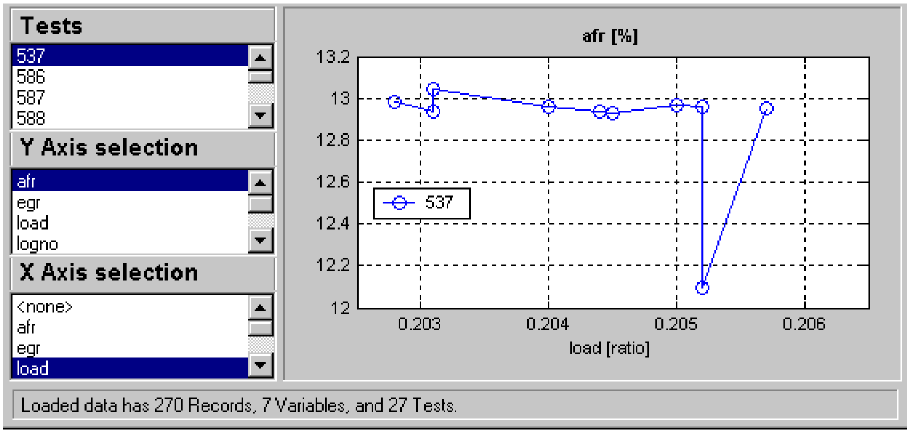
3 Make sure lines are plotted between points: choose **Show Line** from the right-click plot menu. See Test 537 plotted for these variables, following.



Loaded data has 270 Records, 7 Variables, and 27 Tests.

4 Choose **Reorder points** from the right-click menu.

This command replots the line from left to right instead of in the order of the records, as shown.



Loaded data has 270 Records, 7 Variables, and 27 Tests.

Copy Plot creates a copy of the current plot in a new figure.

The menu **View –> Table** shows this data in tabular form. You can select and edit data directly, and you can add records by right-clicking test numbers.

The menu **View –> Graph** returns the view to the plots shown. There are toolbar buttons to switch between these views.

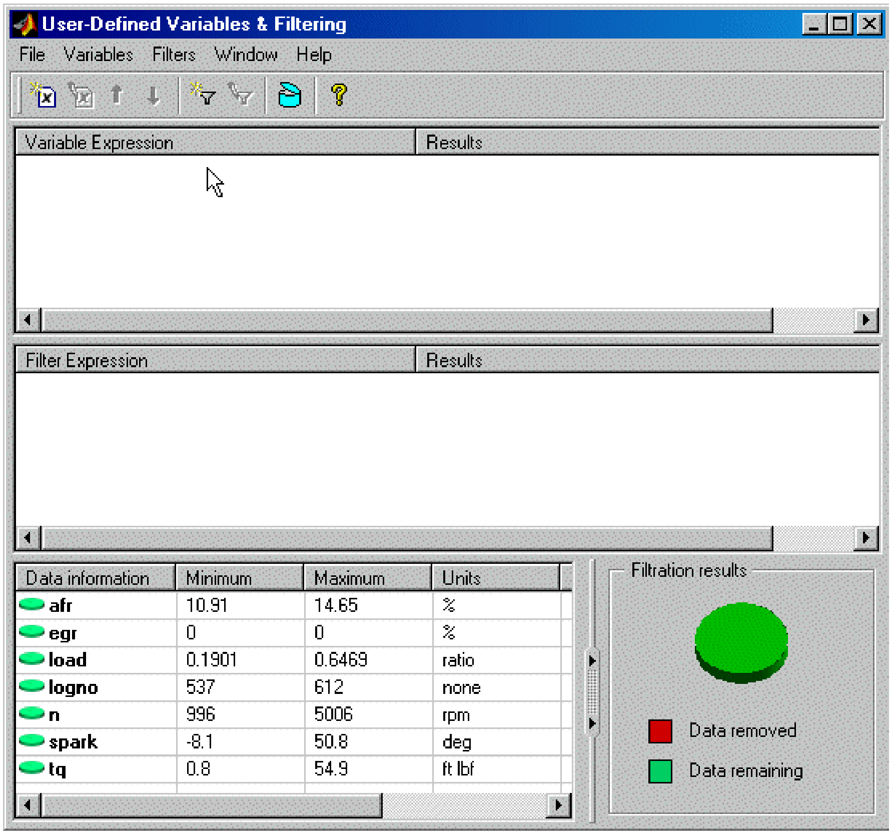
User-Defined Variables & Filtering

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

- Select **Tools –> User-Defined Variables & Filtering**.

  Alternatively, click the toolbar button.

The following window appears.



Data information	Minimum	Maximum	Units
afr	10.91	14.65	%
egr	0	0	%
load	0.1901	0.6469	ratio
logno	537	612	none
n	996	5006	rpm
spark	-8.1	50.8	deg
tq	0.8	54.9	ft lbf

Defining a New Variable

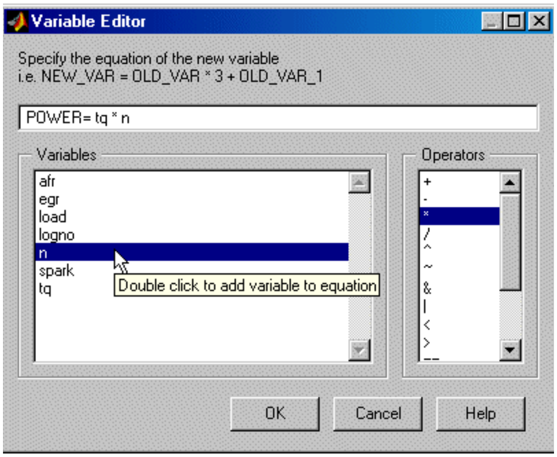
1 Select **Variables –> New Variable** in the **User-Defined Variables & Filtering** window.

The **Variable Editor** appears.

You can define new variables in terms of existing variables. You define the new variable by writing an equation in the edit box at the top of the **Variable Editor** dialog.

2 Define a new variable called **POWER** that is defined as the product of two existing variables, **tq** and **n**, by entering `POWER=tq*n`, as seen in the example following. You can also double-click variable names and operators to add them, which can be useful to avoid typing mistakes in variable names, which must be exact including case.

3 Click **OK** to add this variable to the current data set.



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

POWER= tq * n

Double click to add variable to equation

This new variable appears in the top list (variable expression) of the **User-Defined Variables & Filtering** window.

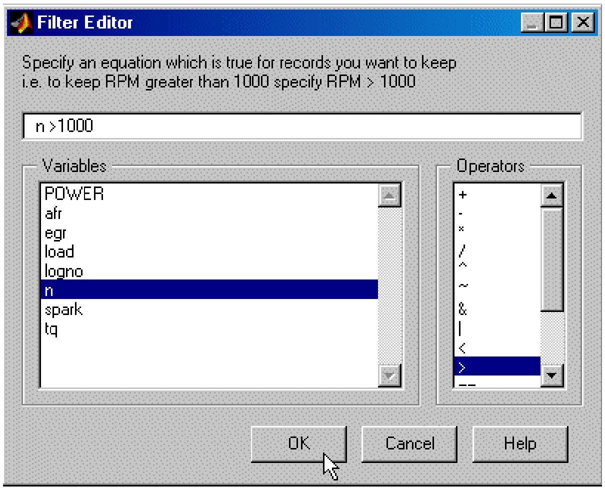
POWER also appears in the **Data Information** pane, along with the original variables, and its definition is included in the description field (to the right of the units column; you might have to scroll or resize to see it).

Applying a Filter

A filter is the name for a constraint on the data set used to exclude some records. You use the **Filter Editor** to create new filters.

1 Choose **Filters –> New Filter**, or click the button in the **User-Defined Variables & Filtering** window.

The **Filter Editor** dialog appears.



Filter Editor

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

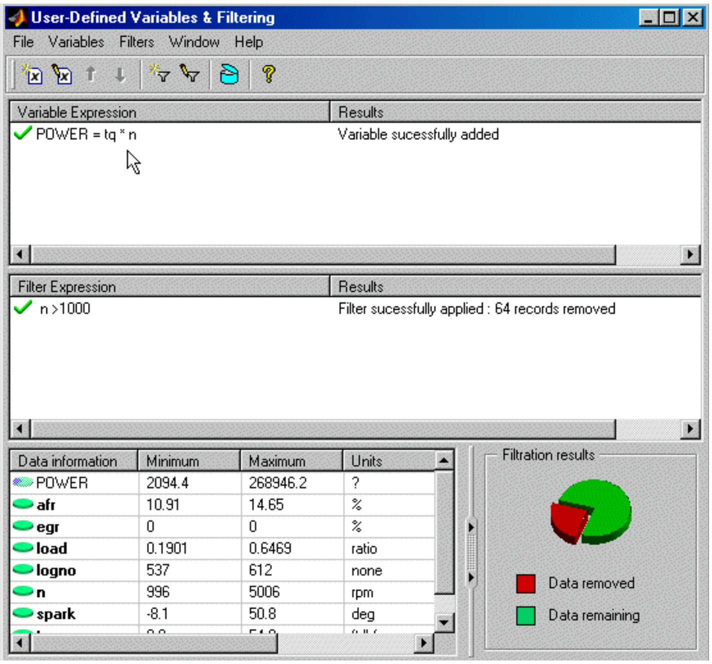
n >1000

You define the filter using logical operators on the existing variables.

2 Keep all records with speed (**n**) greater than 1000. Double-click the variable n, then the operator >, then type 1000.

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

This new filter appears in the center list of the **User-Defined Variables & Filtering** window. The **Filtration results** pie chart shows graphically how many records were removed by the imposition of this filter, and the number of records removed appears in the **Results** column of the **Filter Expression** pane. The number of records at the top of the Data Editor window also changes accordingly (for example, **Records 200/264** indicates that 64 records have been filtered out).



User-Defined Variables & Filtering
File Variables Filters Window Help

Variable Expression	Results
✓ POWER = tq * n	Variable sucessfully added

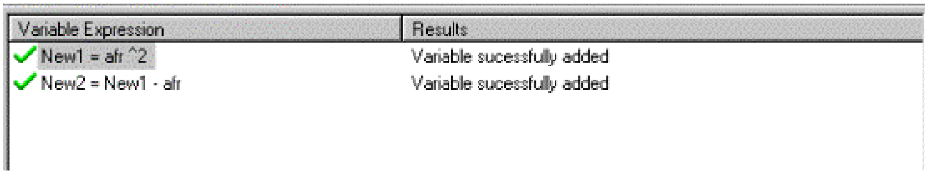
Filter Expression	Results
✓ n >1000	Filter sucessfully applied : 64 records removed

Data information	Minimum	Maximum	Units
POWER	2094.4	268946.2	?
afr	10.91	14.65	%
egr	0	0	%
load	0.1901	0.6469	ratio
logno	537	612	none
n	996	5006	rpm
spark	-8.1	50.8	deg

Filtration results
🟥 Data removed
🟩 Data remaining

Sequence of Variables

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

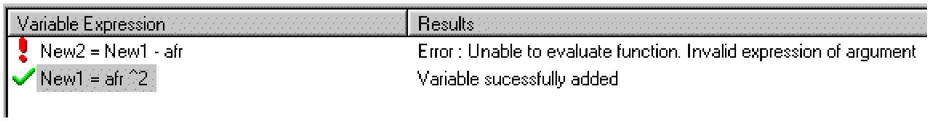


Variable Expression	Results
✓ New1 = afr ^2	Variable sucessfully added
✓ New2 = New1 - afr	Variable sucessfully added

Example:

1 Define two new variables, **New1** and **New2**. 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 Expression	Results
! New2 = New1 - afr	Error : Unable to evaluate function. Invalid expression of 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 the top list of the **User-Defined Variables & Filtering** window and press the **Delete** key.

2 You could also use the menu item **Variables –> Delete Variable**. Use this to delete **New2** (select it first!).

Similarly, you can delete filters by selecting the unwanted filter in the list and deleting via the menu or the **Delete** key.

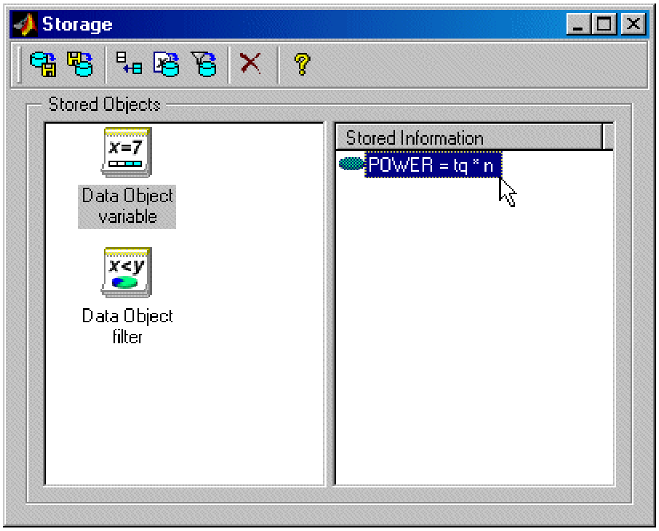
You can also edit current user-defined variables and filters using the relevant menu items or toolbar buttons.

Storage

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

You can open the Storage window from the **User-Defined Variables & Filtering** window in either of these ways:

- Using the menu **File –> Open Storage**
- Using the toolbar button



POWER = tq * n

The toolbar buttons 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 **User-Defined Variables & Filtering** window.
- **Get Current Variables** creates a **Storage Object** containing definitions of all current user-defined variables.

- **Get Current Filters** creates a **Storage Object** containing definitions of all current filters.
- **Delete Stored Object** deletes the currently selected **Storage Object**.

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).

3 This returns you to the **User-Defined Variables & Filters** window. Close this window to return to the **Data Editor**.

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 addlitional 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.

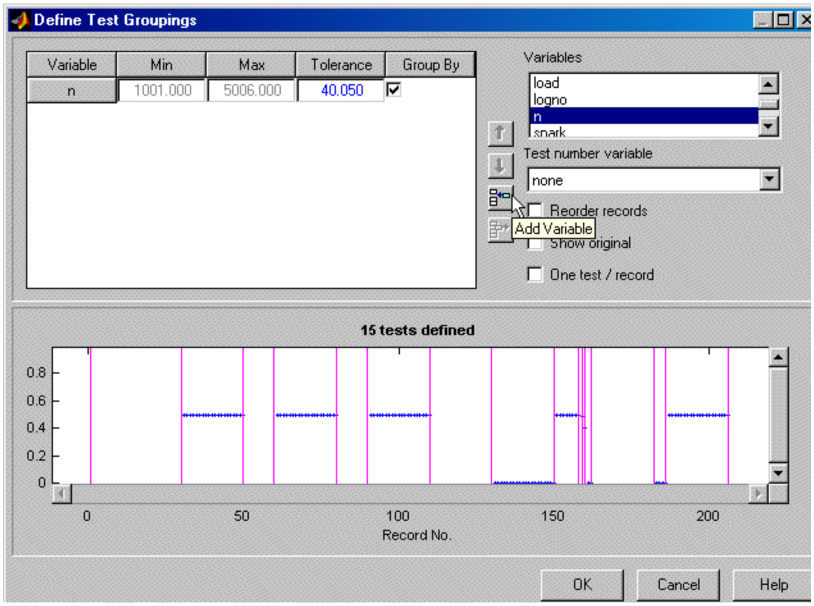
5 Clear the **Group By** check box. Notice that variables can be plotted without being used to define groups.

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.



Define Test Groupings

Variable	Min	Max	Tolerance	Group By
n	1001.000	5006.000	40.050	☑

Variables
load
logno
n
spark

Test number variable: none

□ Reorder records
□ Show original
□ One test / record

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.

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.

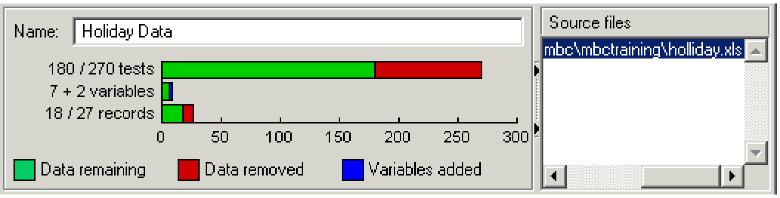
If you chose **none** from the **Test number variable** list, the tests would be numbered 1,2,3 and so on in the order in which the records appear in the data file.

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).



Name: Holiday Data

Source files

mbc\mbctraining\holliday.xls

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.

5

GUI Reference

Project Level: Startup View 5-3
Model Tree 5-10
Test Plans 5-15
Test Plan Level 5-19
Setting Up Models 5-26
Data 5-61
New Response Models 5-89
Local Level 5-91
Global Level 5-107
Selecting Models 5-119
MLE 5-140
Response Level 5-144
Model Evaluation Window 5-147
Exporting Models 5-151
The Design Editor 5-156
Creating a Classical Design 5-162
Creating a Space Filling Design 5-165
Creating an Optimal Design 5-171
Applying Constraints 5-187
Prediction Error Variance Viewer 5-196

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

For a quick guide to setting up models, see these overview pages:

"Instant One-Stage Model Setup" on page 5-27

"Instant Two-Stage Model Setup" on page 5-28

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

"Project Level: Startup View" on page 5-3

"Test Plan Level" on page 5-19

"Local Level" on page 5-89

"Global Level" on page 5-105

"Response Level" on page 5-141

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 Tree" on page 5-10

"Test Plans" on page 5-15

"Data" on page 5-59

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

"Setting Up Models" on page 5-26

"Selecting Models" on page 5-117

"Model Evaluation Window" on page 5-144

"Exporting Models" on page 5-148

Design of Experiment is described in this section:

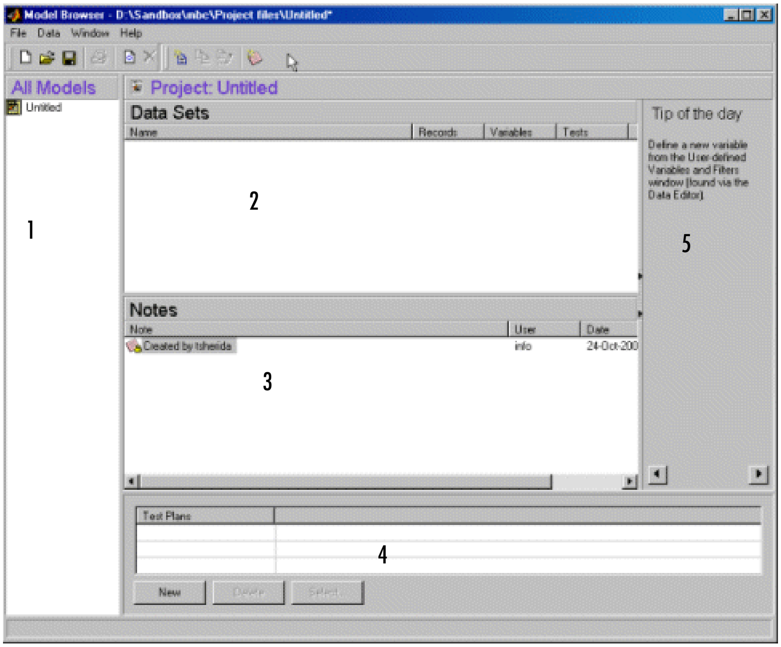
"The Design Editor" on page 5-153

Project Level: Startup View

On startup 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.

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



Name	Records	Variables	Tests

Note	User	Date
Created by tsherida	info	24-Oct-200

Define a new variable from the User-defined Variables and Filters window (found via the Data Editor)

All Models Pane

(Labeled 1). This pane contains a hierarchical structure showing all the models created within the current project. See “Model Tree” on page 5-10 for a detailed description of the information contained in this pane.

Data Sets Pane

(Labeled 2). 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.

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.

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.

Notes Pane

(Labeled 3). 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.

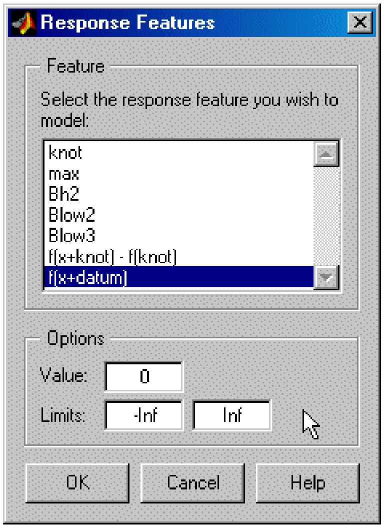
Test Plans List Pane

(Labeled 4). You generate new test plans from the **Test Plans** list pane by clicking the **New** button. See “Test Plans” on page 5-15.

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.



Response Features

Feature

Select the response feature you wish to model:

knot
max
Bh2
Blow2
Blow3
f(x+knot) - f(knot)
f(x+datum)

Options

Value: 0

Limits: -Inf Inf

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.

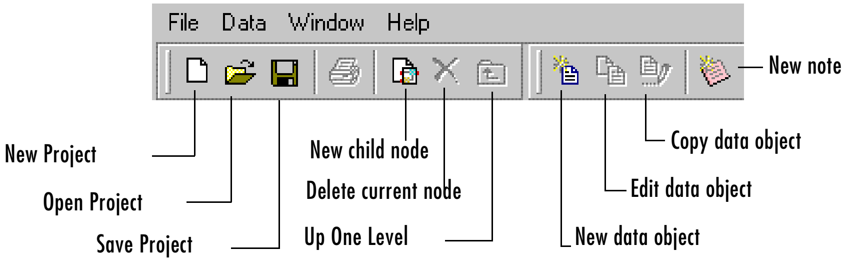
The response features available depend on the model type. For more details on which response features are available, see “Local Models and Associated Response Features” on page 6-48.

- 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 5-117.

Tip of the Day

(Labeled 5). 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).

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 5-148.

- **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.
- **Copy Data** — Copies the selected data set.
- **Edit Data** — Opens the Data Editor to enable data editing.
- **Delete Data** — Deletes the selected data set.

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

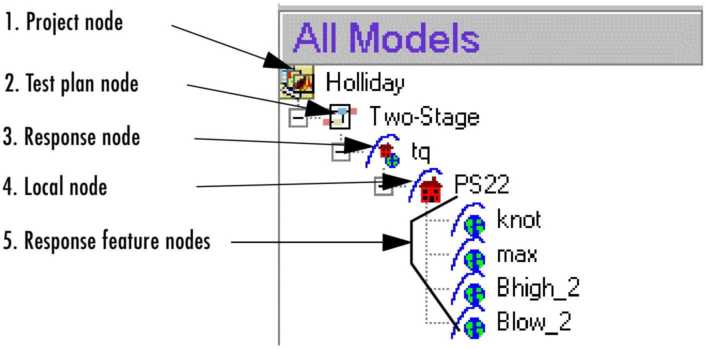
The **Help** menu remains consistent throughout the Model Browser.

- **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

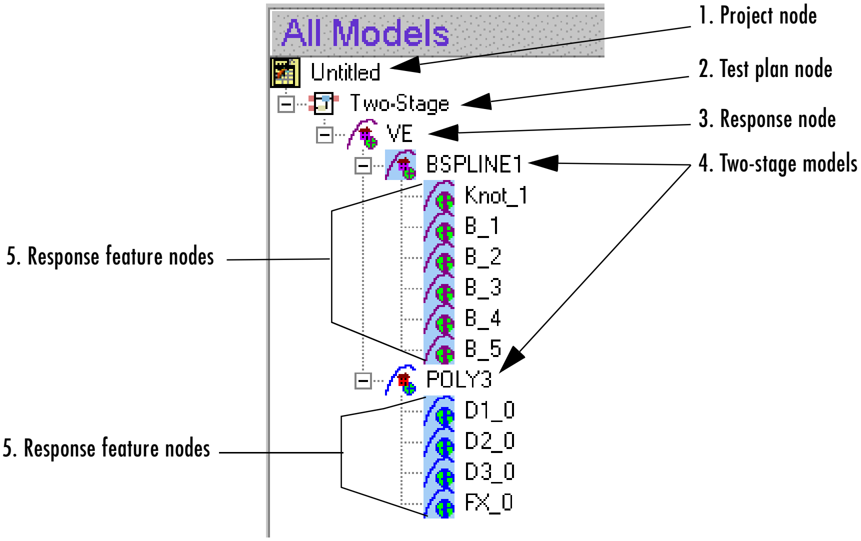
5 Response feature nodes

Note The selected node 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 response 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.

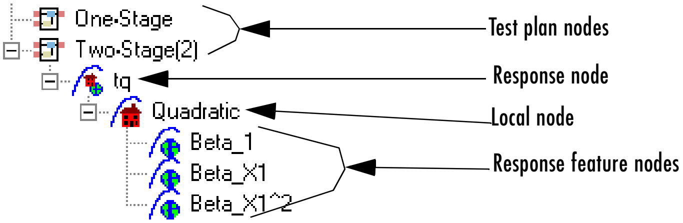
Note You can only have one project open at any one time; if you open another, you are prompted to save or discard your current project.

Response features can themselves have child nodes — several models can be tried at each response feature 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 5-13.

Icons: Curves, Worlds, and Houses

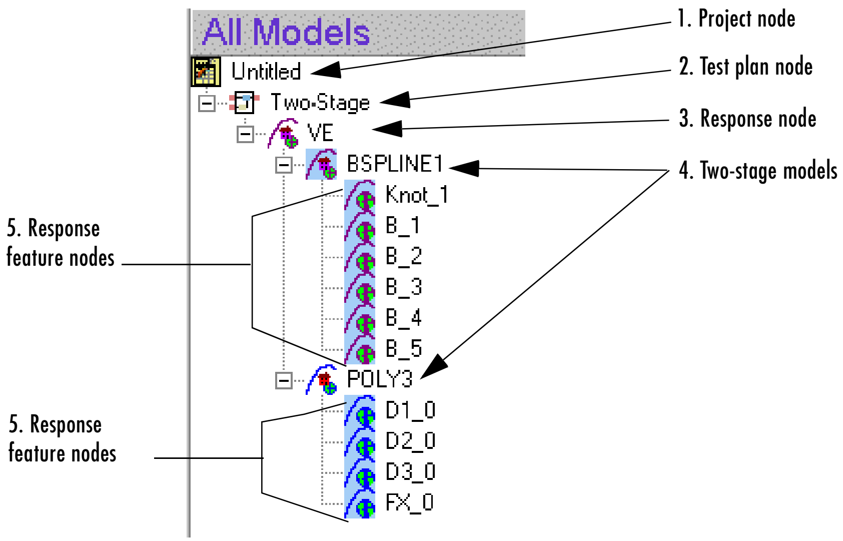
The icons are designed to give visual reminders of their function.

- 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.
- Response features have global models fitted to them, so their icons show a curve over a globe.
- The local model icon shows a curve over a house.



- 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



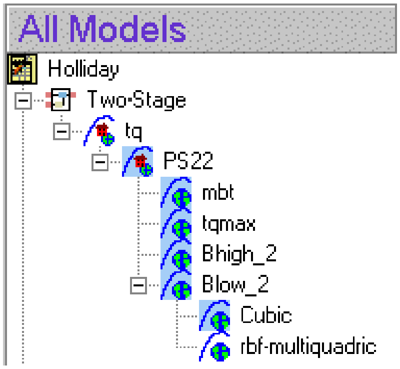
Icon changes convey information about modeling processes.

- 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.

In the following example you can see child nodes of a response feature. You can try different models within a response feature, 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` response feature.



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).

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 5-16.

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 5-89.

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 5-26
- “Setting Up Models” on page 5-26
- “Loading Data from File” on page 5-65

- “Selecting Models” on page 5-117

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)
- 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 5-17.

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.

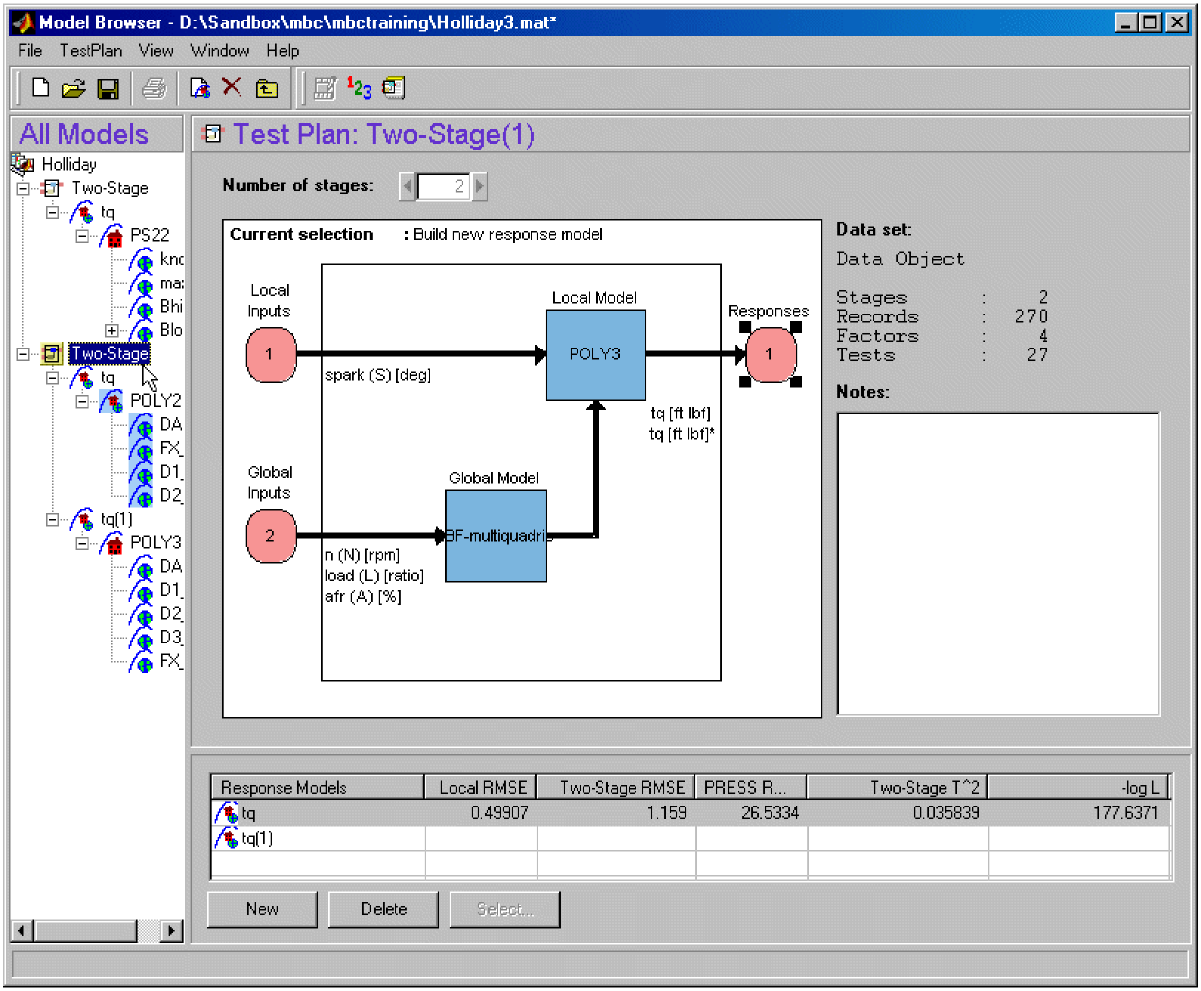
Use stored templates in exactly the same way as the default blank one- and two-stage templates. Models and input factors are already selected (including the response if that was saved with the template) so you can go straight to selecting new data to model. You can still change any settings and design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

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.



Number of stages: 2

Current selection : Build new response model

```
Data set:
Data Object

Stages    :    2
Records   :  270
Factors   :    4
Tests     :   27

Notes:
```

Response Models	Local RMSE	Two-Stage RMSE	PRESS R...	Two-Stage T^2	-log L
tq	0.49907	1.159	26.5334	0.035839	177.6371
tq(1)					

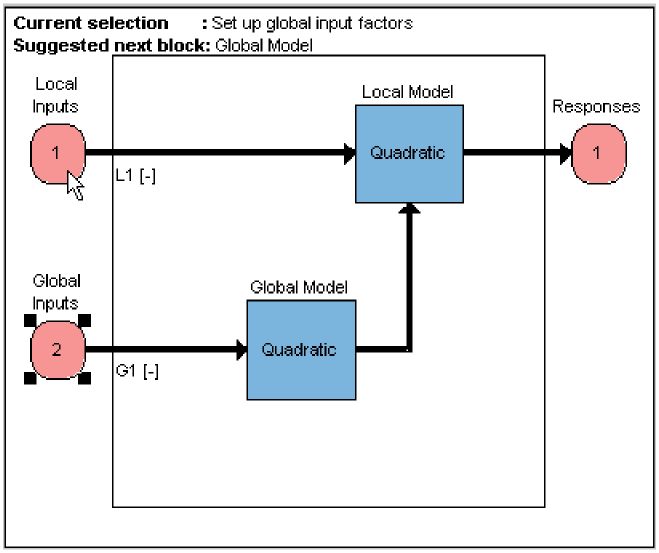
This example is a two-stage model, the same model used in the Quick Start tutorial. All test plan nodes (one- and two-stage) bring up this view with a block diagram of the test plan and the functionality described below.

- 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.
- You can attach data to a new test plan by choosing **TestPlan –> Load New 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** window using the **Select Data** toolbar button 123 or the **TestPlan** menu item. Here you can match data to a design. For example, after the design changes, new data matching might be necessary. See “Data Selection Window” on page 5-81 for details.
- You can save the current test plan as a template using the **TestPlan –> Make Template** command or the toolbar button . See “Local Level” on page 5-89.

See also “Test Plan Level: Toolbar and Menus” on page 5-23

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.



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

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 Models” on page 5-26 — Setting the number of inputs for each stage of the model hierarchy.

2 “Setting Up Models” on page 5-26 — Setting up the new default models for each stage in the model hierarchy.

3 “Designing Experiments” on page 5-57 —Using the Design Editor.

4 “Loading Data from File” on page 5-65 and “New Response Models” —

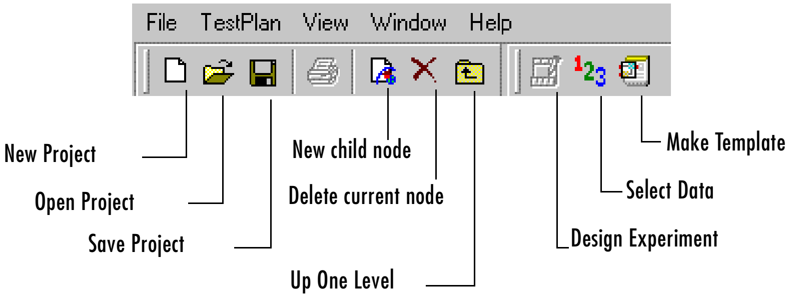
5 “Viewing Designs” on page 5-57 — Viewing the statistical properties of the collected data.

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

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

Test Plan Level: Toolbar and Menus



The seven buttons on the left (project and node management, plus the print button) appear in every view level. See "Project Level: Toolbar" on page 5-7 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.
- **Select Data** opens the Data Wizard, or opens the **Data Selection** window if data sets have already been selected.
- **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 5-89.

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 5-8.

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 5-9.

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 5-9.

TestPlan Menu

- **Set Up Inputs** — See "Setting Up Models" on page 5-26.
- **Set Up Model** — See "Setting Up Models" on page 5-26.

  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 5-153.

  This is also available in the toolbar and in the right-click context menu on the blocks in the test plan diagram.
- **Load New Data** — Opens the Data Editor to load new data.
- **Select Data** — Opens the **Data Selection** window.

  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 to the **Data Selection** window.
- **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 5-89.

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).

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:

- "Instant One-Stage Model Setup" on page 5-27
- "Instant Two-Stage Model Setup" on page 5-28

These sections cover setting up inputs and models in detail:

- "Setting Up Inputs" on page 5-30
- "Global Model Setup" on page 5-32
  - "Global Linear Models: Polynomials and Hybrid Splines" on page 5-33
  - "Global Model Class: Radial Basis Function" on page 5-39
  - "Global Model Class: Hybrid RBF" on page 5-42
  - "Global Model Class: Multiple Linear Models" on page 5-43
  - "Global Model Class: Free Knot Spline" on page 5-44
- "Local Model Setup" on page 5-46
  - "Local Model Class: Polynomials and Polynomial Splines" on page 5-47
  - "Local Model Class: Truncated Power Series" on page 5-50
  - "Local Model Class: Free Knot Spline" on page 5-51
  - "Local Model Class: Growth Models" on page 5-52
  - "Local Model Class: Linear Models" on page 5-53
  - "Local Model Class: User-Defined Models" on page 5-54
  - "Local Model Class: Transient Models" on page 5-54
  - "Covariance Modeling" on page 5-54
  - "Correlation Models" on page 5-56
  - "Transforms" on page 5-56
- "Designing Experiments" on page 5-57 tells you how to proceed to experimental design after setting up models.

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 5-15.

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 Models” on page 5-26 and “Setting Up Models” on page 5-26.

4 At this point, you might want to design an experiment. See “The Design Editor” on page 5-153.

5 From the test plan node, load a new data set to use. Choose **TestPlan –> Load New Data**, which opens the Data Editor. See “Loading Data from File” on page 5-65.

Note Data for one-stage models must be set to one record per test using the test grouping utility. Select **Tools –> Change Test Groupings** in the Data Editor, and select the **one test/record** radio button. See “Test Groupings” on page 5-75.

6 Dismissing the Data Editor opens the Data Wizard, where you match data signals to model variables and then set up the response model.

On completion of these steps 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 5-105 and “Selecting Models” on page 5-117.

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 5-15.

2 From the test plan node, set up the inputs and models at the local and global stages. See "Setting Up Models" on page 5-26 and "Setting Up Models" on page 5-26.

3 At this point, you might want to design an experiment. See "The Design Editor" on page 5-153.

4 From the test plan node, load the data set you want to use. See "Loading Data from File" on page 5-65.

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

Note On completing the Data Wizard, the local and global models are calculated.

---

5 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 (optional).

6 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 5-117.

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 5-12.

---

7 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 5-137 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 6-37 for a detailed explanation of two-stage models.

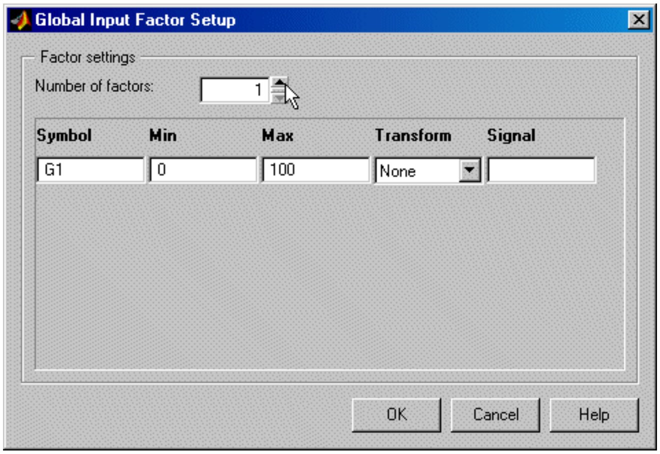
Double-click the Model blocks of the block diagram or select the **TestPlan –> Set Up Model** menu item. MBC supports a wide range of models.

For model descriptions, see “Global Model Setup” on page 5-32 and “Local Model Setup” on page 5-46.

For further statistical details, see “Technical Documents” on page 6-1.

Setting Up Inputs

You set up the number and definition of model input factors for each stage by double-clicking the inports of the block diagram. See “Functions Implemented in the Block Diagram” on page 5-21.



Number of factors: 1

Symbol	Min	Max	Transform	Signal
G1	0	100	None	

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

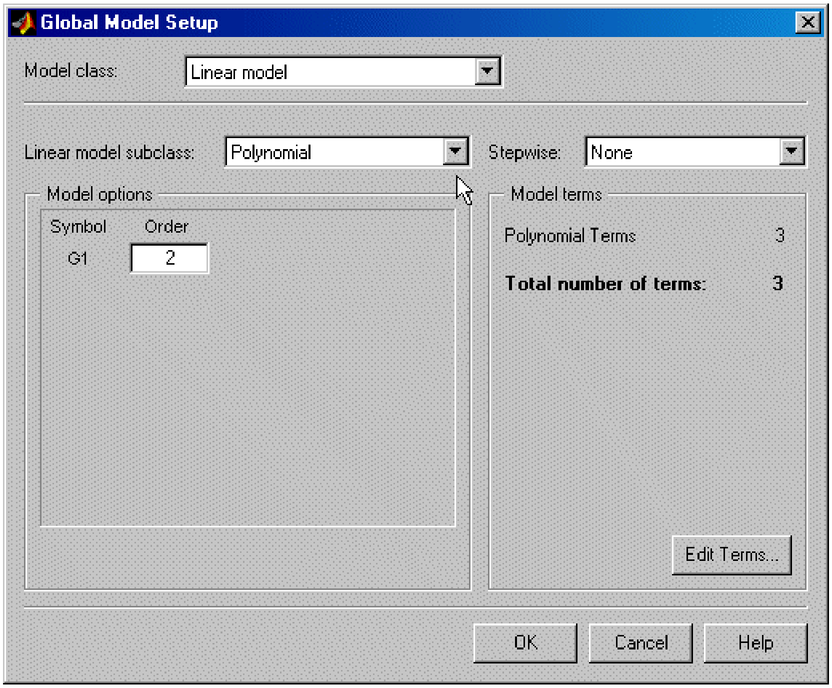
You can use input transformations to change the distribution of the input factor space for designing experiments. The available input transformations are `1/x`, `sqrt(x)`, `log10(x)`, `x^2`, `log(x)`. See "Transforms" on page 5-56 for a discussion of the uses of transformations.

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 Setup

Model class: Linear model

Linear model subclass: Polynomial

Stepwise: None

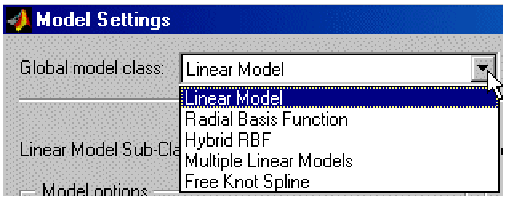
Model options

Symbol	Order
G1	2

Model terms

Polynomial Terms	3
Total number of terms:	**3**

Global Model Classes



Model Settings

Global model class: Linear Model

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 5-33
- "Global Model Class: Radial Basis Function" on page 5-39

- “Global Model Class: Hybrid RBF” on page 5-42
- “Global Model Class: Multiple Linear Models” on page 5-43
- “Global Model Class: Free Knot Spline” on page 5-44

“Linear Model Statistics Displays” on page 6-22 has a description of the statistics displayed when this type of model is analyzed.

See “Setting Up Models” on page 5-26 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

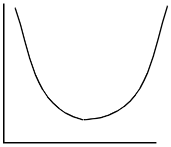
- Polynomial
- Hybrid Spline

Polynomials

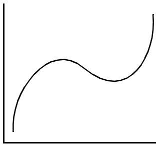
Polynomials of order n are of the form

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

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



Quadratic curve

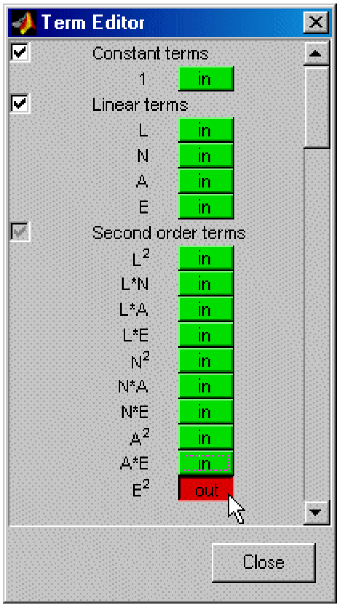


Cubic curve

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-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).



☑ Constant terms

1	in

☑ Linear terms

L	in
N	in
A	in
E	in

☑ Second order terms

L^2	in
L*N	in
L*A	in
L*E	in
N^2	in
N*A	in
N*E	in
A^2	in
A*E	in
E^2	out

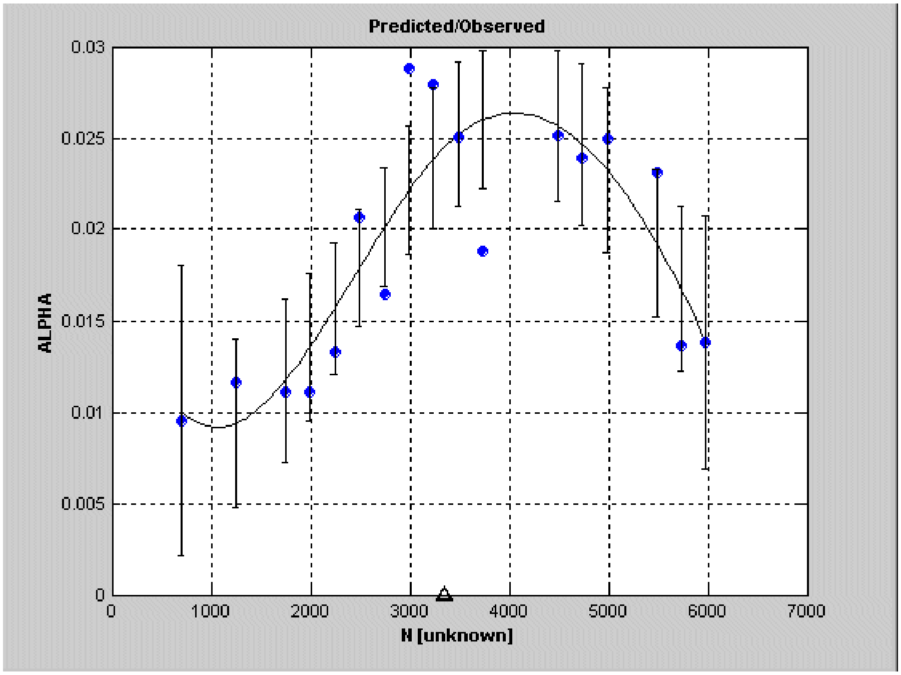
See also “Interaction” on page 5-37.

Hybrid Splines

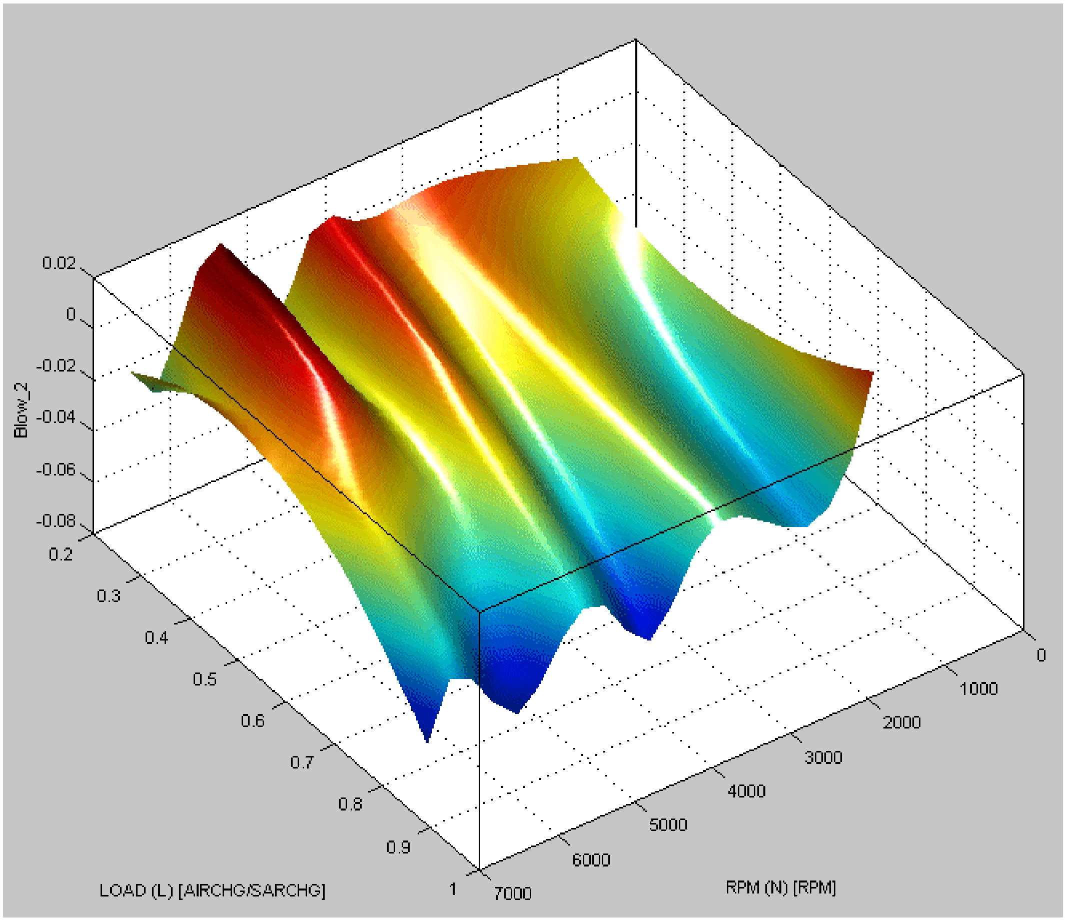
You can use the Hybrid Spline model to fit a spline to one factor and polynomials to all other factors.

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.

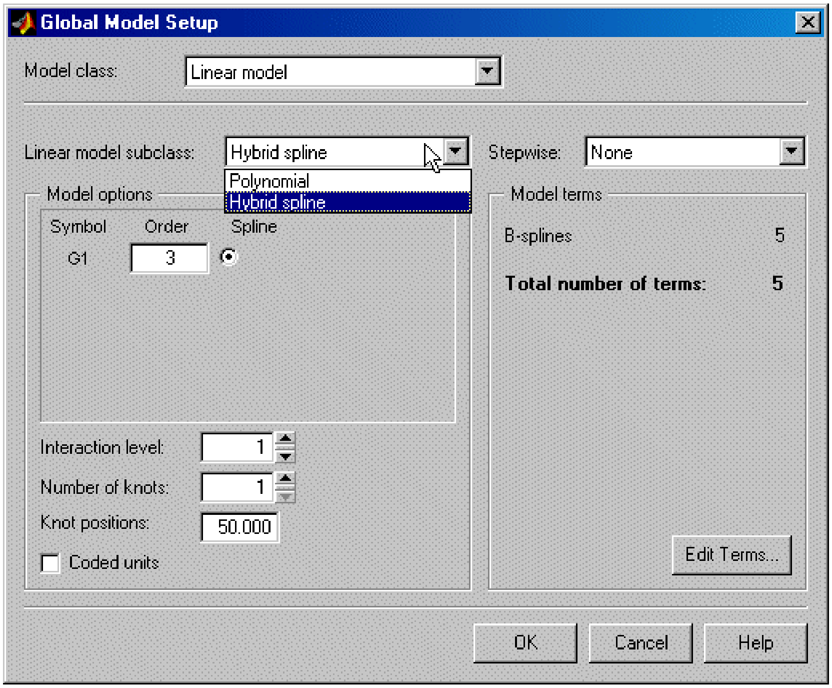


You can fit more complicated curves using splines, so they can be useful for the factor you expect to behave in the most complex way. This allows you to model detailed fluctuations in the response for one factor, while simpler models are sufficient to describe the other factors. The following example 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.



Global Model Setup

Model class: Linear model

Linear model subclass: Hybrid spline

Stepwise: None

Model options

Symbol	Order	Spline
G1	3	●

Interaction level: 1

Number of knots: 1

Knot positions: 50.000

□ Coded units

Model terms

B-splines	5
Total number of terms:	**5**

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 5-33.

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.

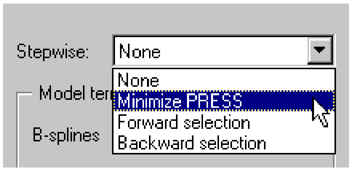
If you increase the interaction level to 2, under second-order terms you see L^2 and also L multiplied by each of the other factors; that is, second-order cross-terms (for example, LN, LA, and LS).

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



Stepwise: None

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.

Predicted sum of squares error(PRESS) is a measure of the predictive quality of a model.

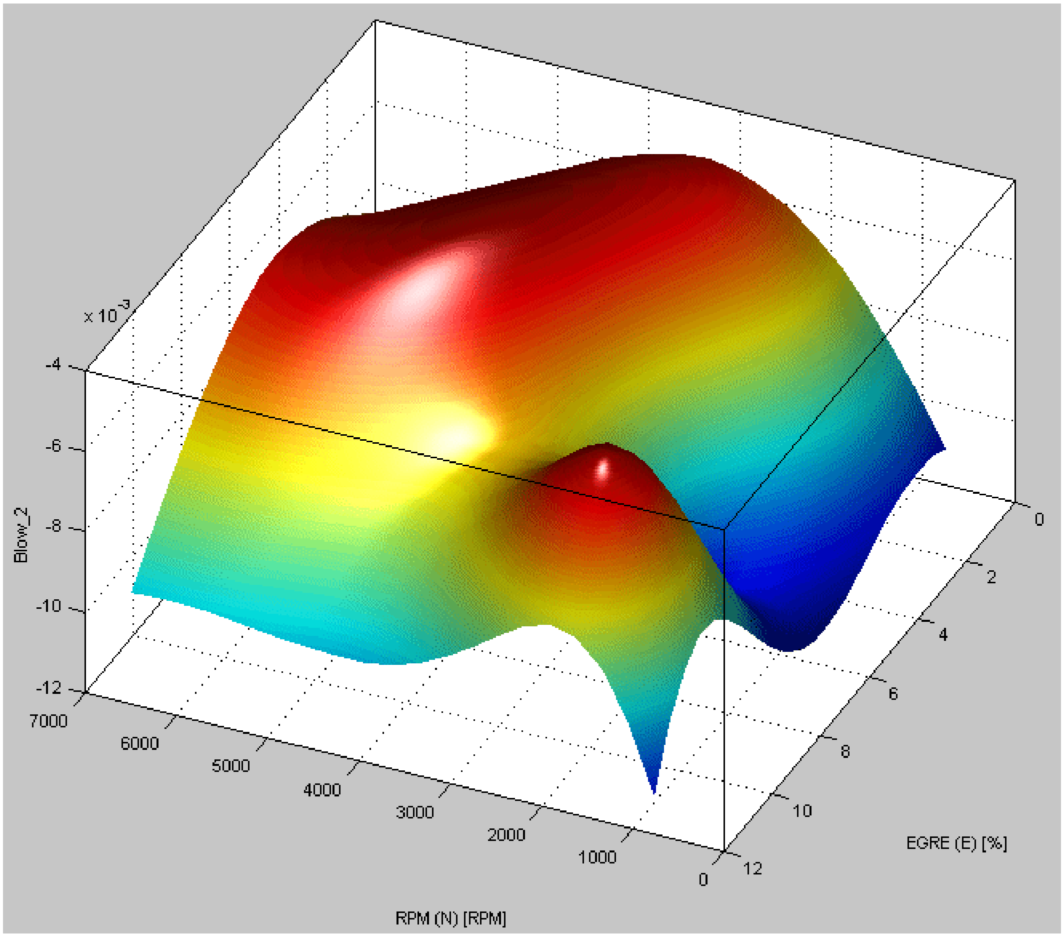
`Min PRESS` throws away terms in the model to improve its predictive quality, removing those terms that reduce the PRESS of the model.

For a further discussion of the statistical effects of all the menu items in the **Stepwise** feature, see “Stepwise Regression Techniques” on page 6-13 and “Linear Model Statistics Displays” on page 6-22.

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.

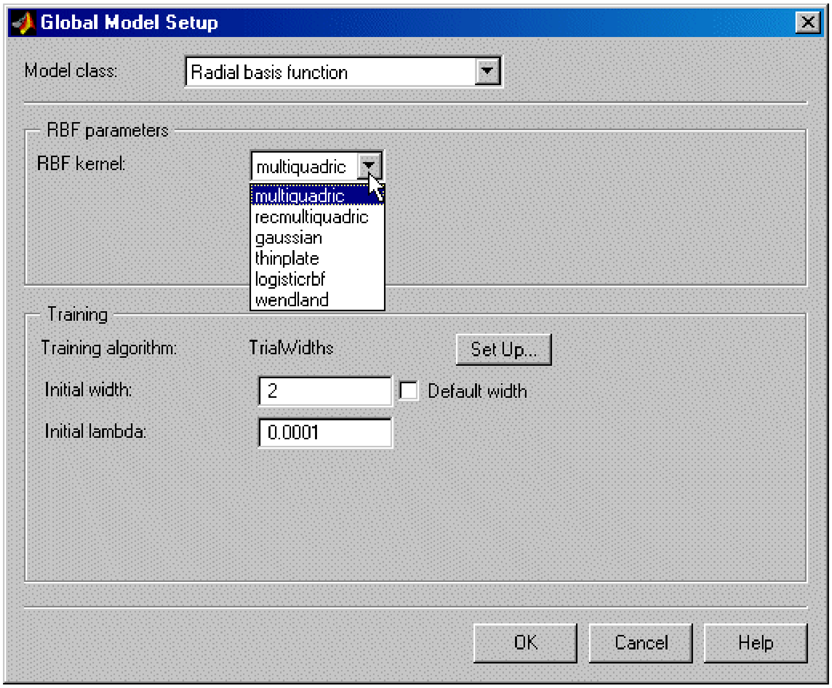
Networks of RBFs can model a wide variety of surfaces. You can optimize on the number of centers and their position, height and width. You can have different widths of centers in different factors. RBFs can be very useful for investigating the shapes of surfaces when system knowledge is low. Combining 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 7-1. See especially “Tips for Modeling with Radial Basis Functions” on page 7-30 and “Types of Radial Basis Functions” on page 7-4.

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 7-3.

The following example illustrates the types of RBF available.



Global Model Setup

Model class: Radial basis function

RBF parameters

RBF kernel: multiquadric

Training

Training algorithm: TrialWidths

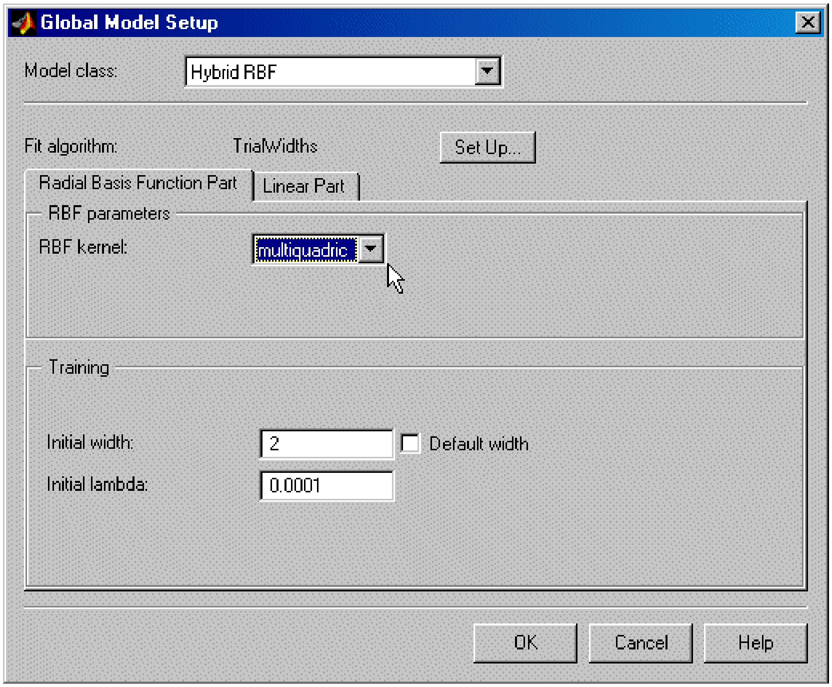
Initial width: 2

□ Default width

Initial lambda: 0.0001

Global Model Class: Hybrid RBF

This option combines an RBF model with a linear model.



Model class: Hybrid RBF

Fit algorithm: TrialWidths

RBF kernel: multiquadric

Initial width: 2 ☐ Default width

Initial lambda: 0.0001

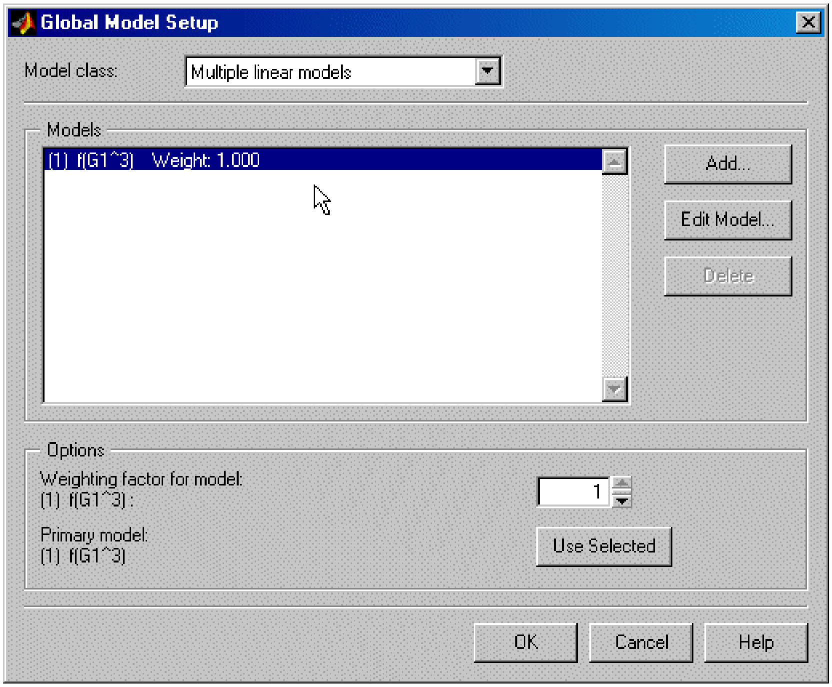
The **RBF kernel** drop-down menu offers the same options as for normal RBF.

The **Linear Part** tab contains the same options as the other global linear models; see “Global Linear Models: Polynomials and Hybrid Splines” on page 5-33.

See “Hybrid Radial Basis Functions” on page 7-28.

See also “Radial Basis Functions” on page 7-1 for a detailed guide to the use of all the available RBFs in modeling.

Global Model Class: Multiple Linear Models



Model class: Multiple linear models

(1) f(G1^3) Weight: 1.000

Weighting factor for model:
(1) f(G1^3) : 1

Primary model:
(1) f(G1^3)

Use Selected

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 three 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.

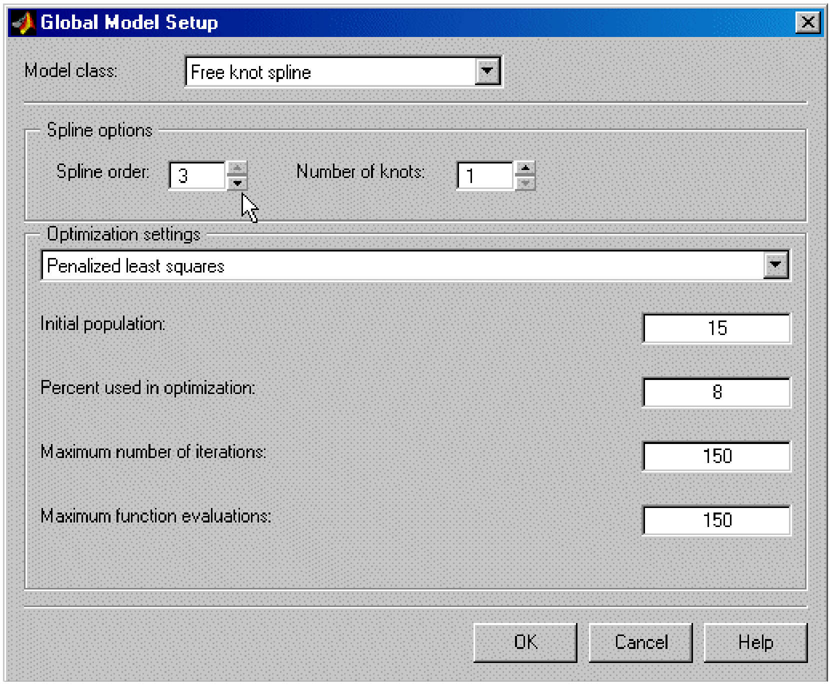
The model that appears in the model tree is the one you select, listed as **Primary model**. Click the model in the list, then click **Use Selected**. The

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 5-33.

Global Model Class: Free Knot Spline



Model class: Free knot spline

Spline order: 3

Number of knots: 1

Penalized least squares

Initial population: 15

Percent used in optimization: 8

Maximum number of iterations: 150

Maximum function evaluations: 150

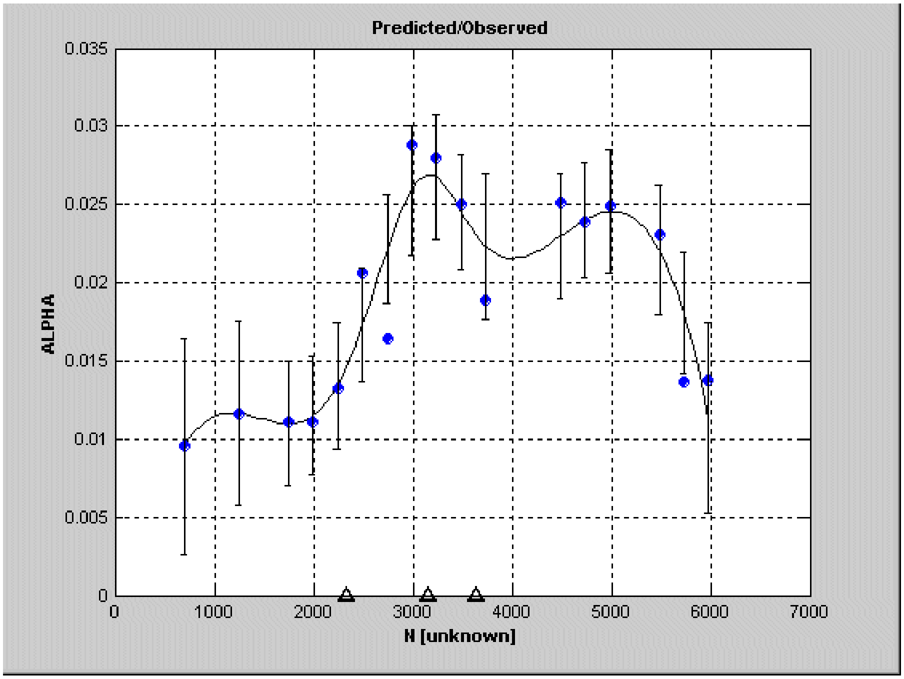
This option is only available for global models with only one input factor. See also “Hybrid Splines” on page 5-34 for a description of splines. The major difference is that you choose the position of the knots for hybrid splines; here the optimal knot positions are calculated as part of the fitting 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.

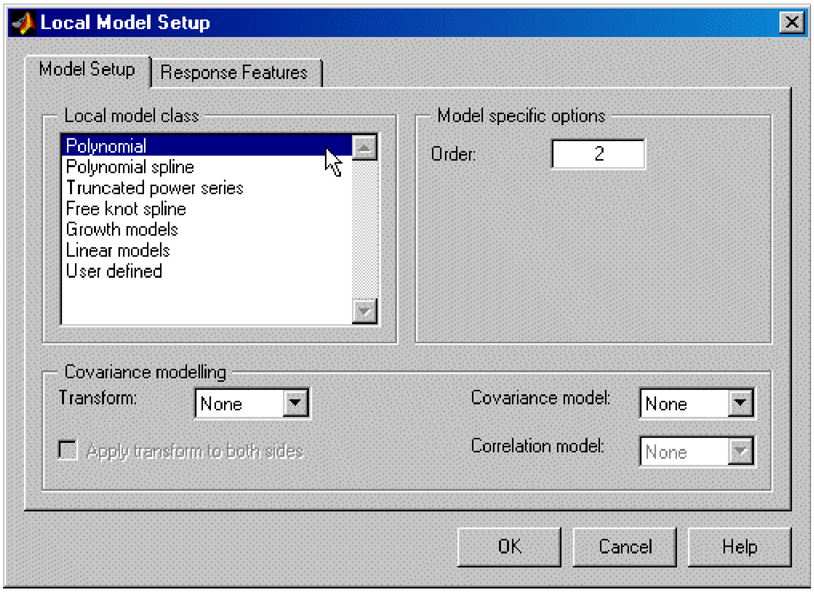
The example following shows a free knot spline model with three knots. The position of the knots is marked on the `N` axis.



See also the local models involving splines:

- "Polynomial Spline" on page 5-48
- "Local Model Class: Truncated Power Series" on page 5-50
- "Local Model Class: Free Knot Spline" on page 5-51

Local Model Setup



Order: 2

Transform: None

Covariance model: None

□ Apply transform to both sides

Correlation model: None

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
- 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 5-54

- “Correlation Models” on page 5-56
- “Transforms” on page 5-56

See “Local Model Definitions” on page 6-48 for statistical details of all local models.

See “Setting Up Models” on page 5-26 for a list of all information about setting up local and global models.

Local Model Class: Polynomials and Polynomial Splines

Polynomials

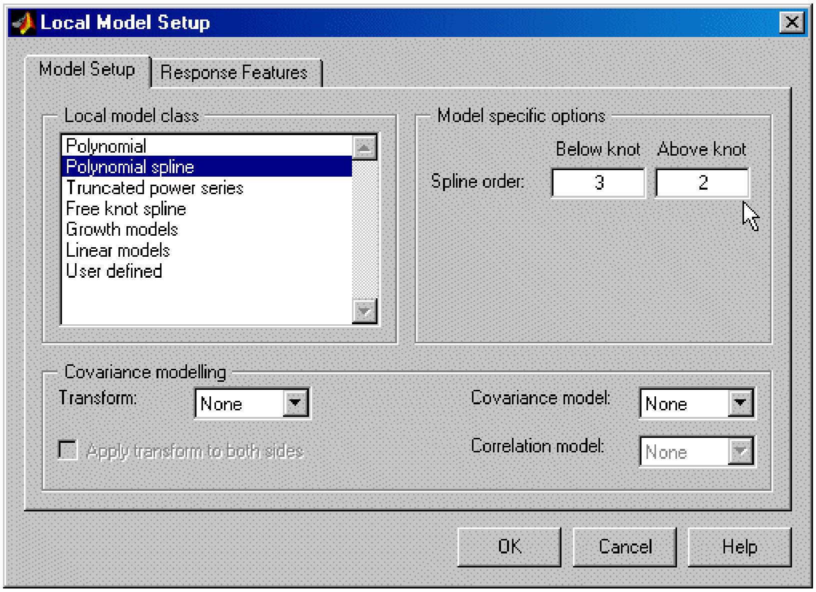
At the local level, if you have one input factor, you can choose Polynomial directly from the list of local model classes. Here you can choose the order of polynomials used, and you can define a datum model for this kind of local model (see below).

If there is more than one input factor, you can only choose Linear Models 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 5-53.

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 5-88

See “Polynomials” on page 5-33 for a general description of polynomial models.

Polynomial Spline



Spline order: Below knot 3, Above knot 2

Transform: None

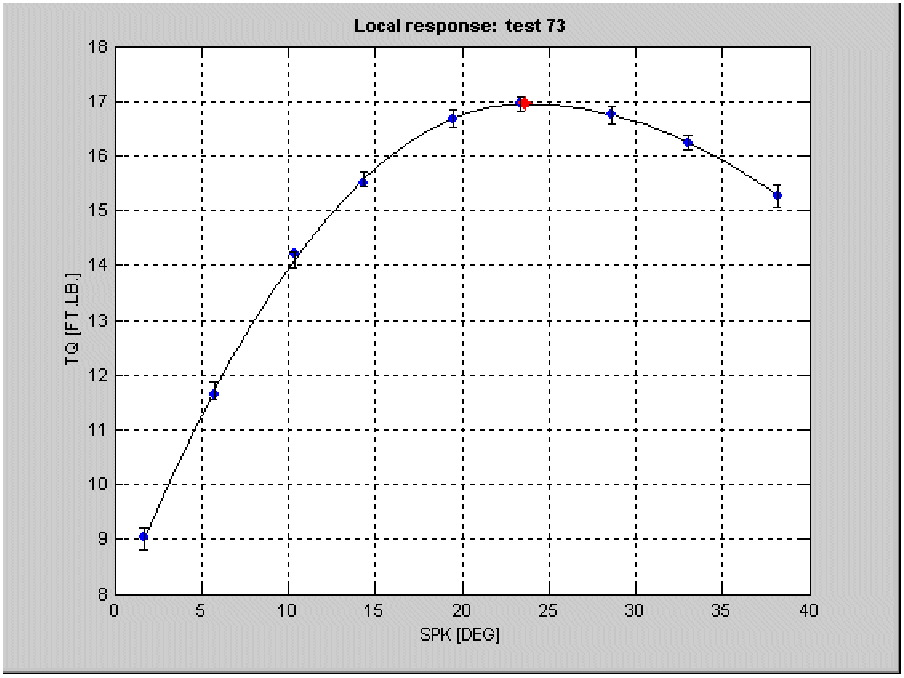
Covariance model: None

Correlation model: None

A spline is a piecewise polynomial, where different sections of polynomial are fitted smoothly together. The location of each break is called a knot.

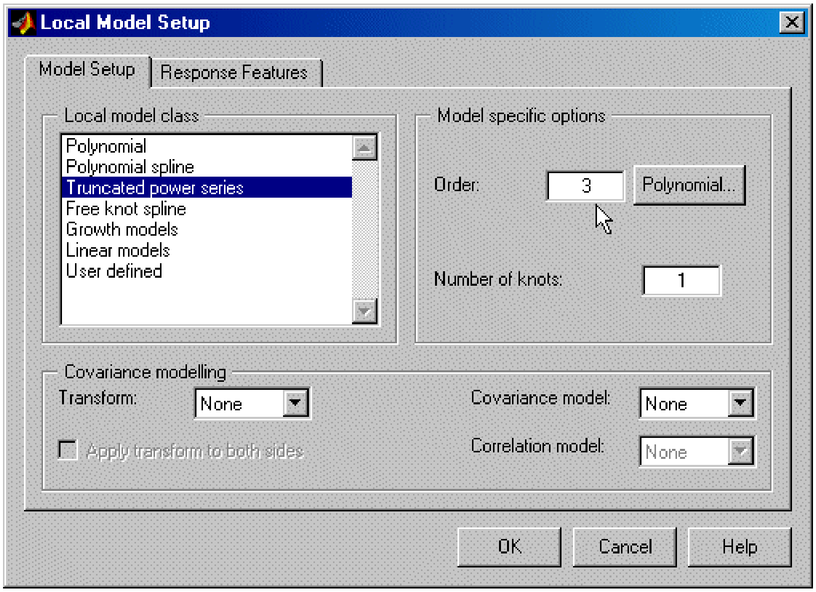
This model has only one knot. You can choose the orders of the polynomials above and below the knot. See also “Hybrid Splines” on page 5-34. These global models also use splines, but use the same order polynomial throughout.

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 6-49 for statistical details.

Local Model Class: Truncated Power Series



Order: 3

Number of knots: 1

Transform: None

Covariance model: None

Correlation model: None

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.

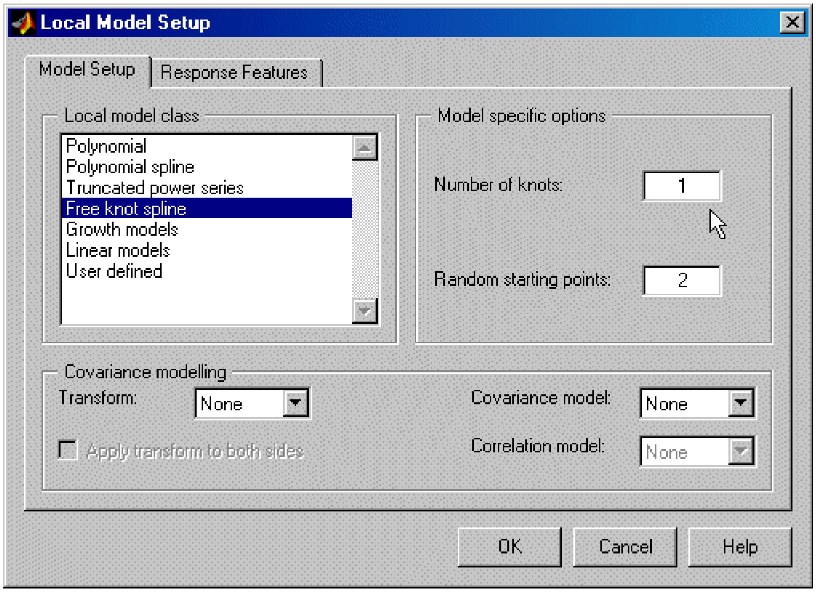
Click **Polynomial** to see (and remove, if you want) the polynomial terms. One use of the remove polynomial term function is to make the function linear until the knot, and then quadratic above the knot. In this case we remove the quadratic coefficient.

See also

- "Polynomial Spline" on page 5-48, where you can choose different order basis functions either side of the knot
- "Hybrid Splines" on page 5-34, 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 5-51, free knot splines with cubic basis functions, where you can choose the number of knots

Local Model Class: Free Knot Spline

These are the same as the "Global Model Class: Free Knot Spline" on page 5-44 (which is also only available for one input factor). See the global free knot splines for an example curve shape.



Number of knots: 1

Random starting points: 2

Transform: None

Covariance model: None

☐ Apply transform to both sides

Correlation model: None

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. The order of polynomial fitted (in all curve sections) is cubic. The "B" in B-spline stands for *Basis*, after the polynomial basis function used for the curve sections.

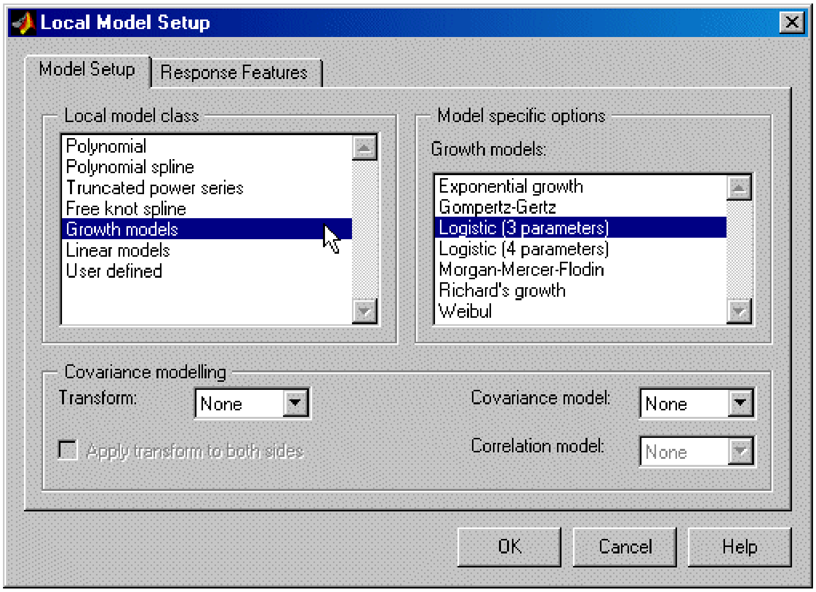
You can set the number of **Random starting points**. These are the number of initial guesses at the knot positions.

See also

- "Polynomial Spline" on page 5-48, where you can choose different order basis functions for either side of the knot
- "Local Model Class: Truncated Power Series" on page 5-50, where you can choose the order of the basis function
- "Hybrid Splines" on page 5-34, a global model where you can choose a spline order for one of the factors (the rest have polynomials fitted)

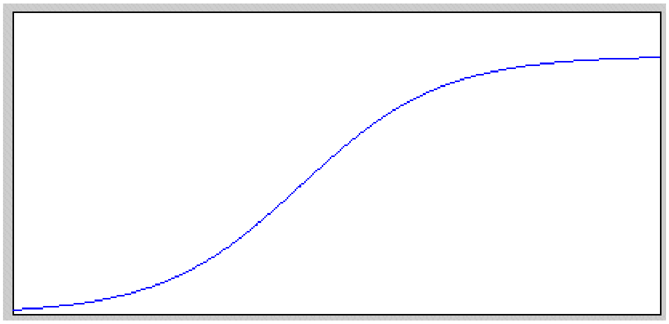
Local Model Class: Growth Models

Growth models have a number of varieties available, as shown. They are only available for single input factors.



Local model class: Growth models
Growth models: Logistic (3 parameters)
Transform: None
Covariance model: None
Correlation model: None

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 6-48 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 5-47.

If there is more than one input factor, you can only choose Linear 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 5-33 for details.

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 5-88.

These linear models are labelled `Quadratic`, `Cubic` etc on the test plan block diagram, while the single input type of polynomials are labelled `Poly2`, `Poly3` etc.For higher orders both types are labelled `Poly` *n*.

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`.

See "User-Defined Models" on page 6-58 of the Technical Documents for detailed instructions on this.

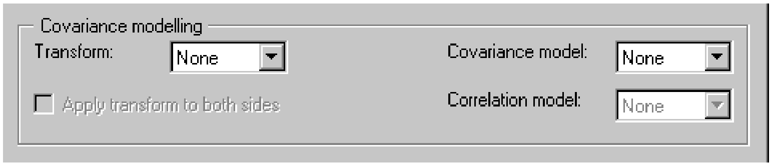
Local Model Class: Transient Models

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. You must check these into the Model-Based Calibration Toolbox before you can use them for modeling.

See "Transient Models" on page 6-64 of the Technical Documents for detailed instructions and an example.

Covariance Modeling

This frame is visible no matter what form of local model is selected in the list.



Covariance modelling

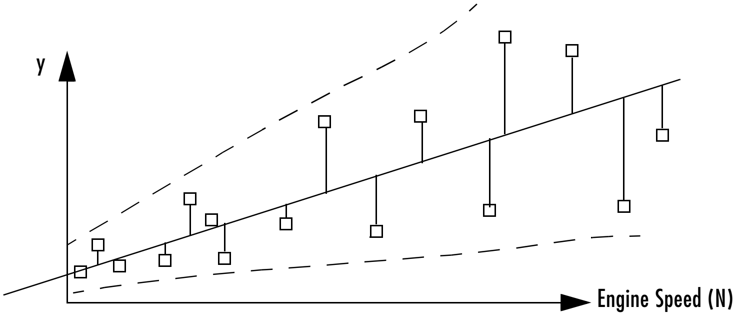
Transform: None

□ Apply transform to both sides

Covariance model: None

Correlation model: None

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 $\varepsilon_n = \alpha_1 \xi_{n-1} + \xi_n$.

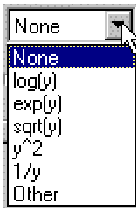
AR(1) – The Auto Regressive method has the form $\varepsilon_n = \alpha_1 \varepsilon_{n-1} + \xi_n$.

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 “Design of Experiment Tutorial” 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 5-153 or the step-by-step guide in the “Design of Experiment Tutorial” on page 3-1.

Data

This section describes all aspects of loading, manipulating, and selecting data in the Model Browser. There are three main graphical interfaces for dealing with data:

- You use the Data Editor for loading, filtering, grouping, and editing data, and you can define new variables. The Data Editor contains various graphical interfaces for these tasks: the Data Import Wizard for loading and merging data; the **User-Defined Variables and Filters** window (with the Variable Editor, Filters Editor, and **Storage** dialog) for creating and storing new variables and data filters; and the **Test Groupings** dialog for plotting and manipulating data groups. You can reach the Data Editor from project or test plan level.
- You use the Data Wizard to select data for modeling and set up matching data to designs by setting tolerances and opening the **Data Selection** window. You reach the Data Wizard from test plan level.
- You use the **Data Selection** window 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 reach the **Data Selection** window from test plan level.

You can load and merge data from the following:

- From files (Excel, Concerto, MATLAB)
- From the workspace
- From tailor-made Excel sheets

See “Data Loading and Merging” on page 5-65.

Within the Data Editor, you can do the following:

- View plots, edit and add data records.
- Define new variables. See “User-Defined Variables and Filtering” on page 5-69 and “Variable Editor” on page 5-71.
- Apply filters to remove unwanted records. See “Filter Editor” on page 5-72.
- Store and retrieve user-defined variables and filters. See “Storage” on page 5-74.
- Define test groupings to collect data into groups.

You use the Data Wizard to do the following:

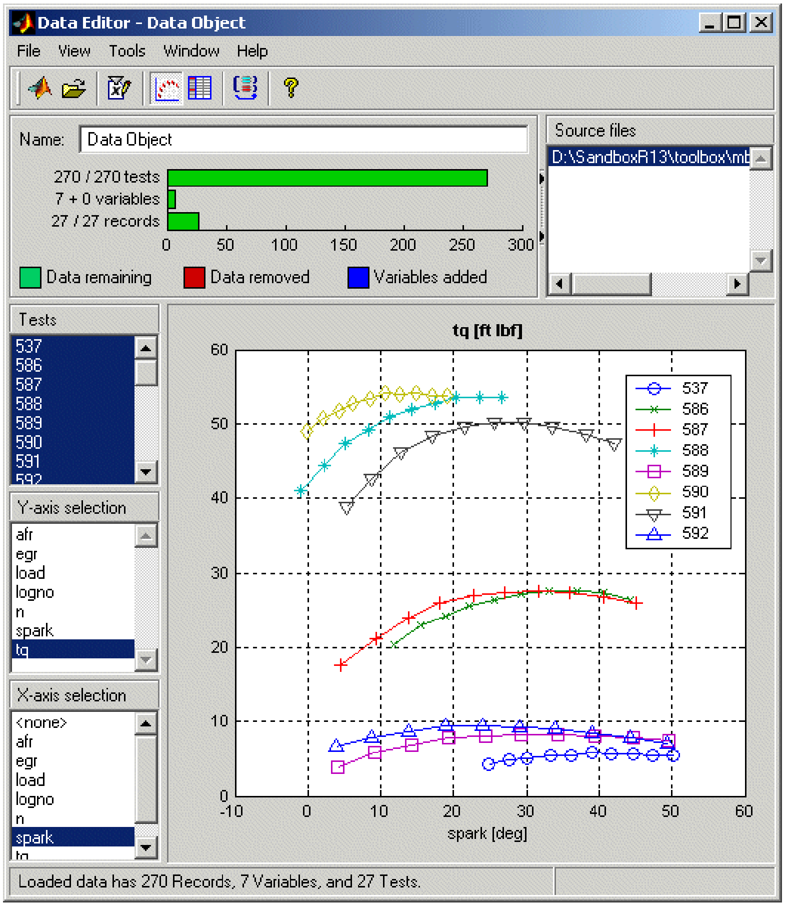
- Select the data set and design to use for modeling.
- Select the data signals to use for model input factors (one-stage, or local and global for two-stage).
- Select matching tolerances (if matching data to a design).
- Select data signals for response model input factors.

The **Data Selection** window matches data to experimental designs.

The Data Editor

To reach the Data Editor:

- From the test plan node, choose **TestPlan –> Load New 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.

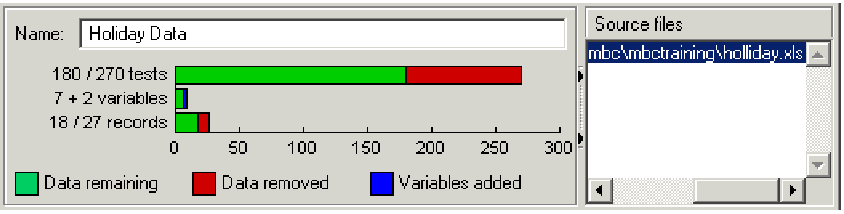


Name: Data Object

D:\SandboxR13\toolbox\mb

Loaded data has 270 Records, 7 Variables, and 27 Tests.

As can be seen in the preceding example, in the default Graph view you can select combinations of variables from the list boxes on the left and view all tests. Multiple selection of tests and *y*-axis variables is possible — in the example, multiple tests are selected to view several tests simultaneously. In the Table view (toolbar button and **View** menu), you can edit and add records. See "Data Editor Toolbar and Menus" on page 5-63.


Name: Holiday Data

Source files

mbc\mbctraining\holliday.xls

The list box at the top right contains the source file information for the data, and other information is displayed on the left: the name of the data set and the numbers of records, variables, and tests it contains. See the preceding example.

The bars and figures on the left 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 variable added to a data set originally containing seven variables, you see '**Variables 7 + 2**'.

By default the new data set is called Data Object. You can change this name in the **Name** edit box at the top of the window.

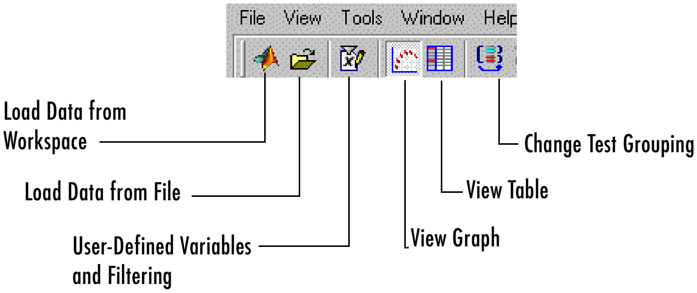
See "Data Editor Toolbar and Menus" on page 5-63 for other controls.

You can also 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).

There is a right-click menu on the plot, to show the legend (only applicable for multiple selections), the grid, and the line. This line joins the data points. **Reorder points** 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



- **Load Data from Workspace** — See “Loading Data from the Workspace” on page 5-66; also in the **File** menu.
- **Load Data from File** — See “Loading Data from File” on page 5-65; also in the **File** menu.
- **User-Defined Variables and Filtering** — Opens the **User-Defined Variables and Filtering** window; also in the **Tools** menu.
- **View Graph** — The default display in the Data Editor is graphical; also in the **View** menu.
- **View Table** — Changes the main display to a table of the data; also in the **View** menu. In the Table view, you can edit data records, and you can access the **Add Record** command by right-clicking a test number.
- **Change Test Grouping** — Opens the **Test Groupings** dialog; also in the **Tools** menu.

The **Window** and **Help** menus are the same as everywhere else in the Model Browser. See “Window Menu” on page 5-9 and “Help Menu” on page 5-9.

File Menu

- **Load Data from File** — See “Loading Data from File” on page 5-65.
- **Load Data from Workspace** — See “Loading Data from the Workspace” on page 5-66.
- **Export Data** — Exports data to Excel or to the workspace.

- **Close** — Closes the Data Editor.

View Menu

- **Graph**
- **Table**

See Toolbar above.

Tools Menu

- **User-Defined Variables & Filtering** — You can add new variables to the data set and remove records by imposing constraints on the data. Also in the toolbar.
- **Change Test Groupings** — Opens the **Test Groupings** dialog; also in the toolbar.
- **Make Excel Sheet for Import** — Opens an Excel sheet, prepared for data input in the form the Model Browser expects, with Name, Unit, and Data the headings for the first three rows (variables are expected to be column headings with data records in each row).
- **Import Excel sheet** — Only active after the **Make Excel Sheet for Import** command. This command closes the Excel sheet and loads the data into the Data Editor (unless there is already data loaded, in which case it first opens the Data Merging Wizard).

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.

Loading Data from File

1. From the test plan node, choose **TestPlan –> Load New Data**.

   Alternatively, from the project node choose **Data –> New Data**.

   The Data Editor appears.

2. Click the **Open File** button in the toolbar 📂 to load data from a file. See "Data Editor Toolbar and Menus" on page 5-63.

3. Choose **File –> Load data from file**.

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

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 decide 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. 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.

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 test plan node, choose **TestPlan –> Load New Data**.

   Alternatively, from the project node choose **Data –> New Data**.

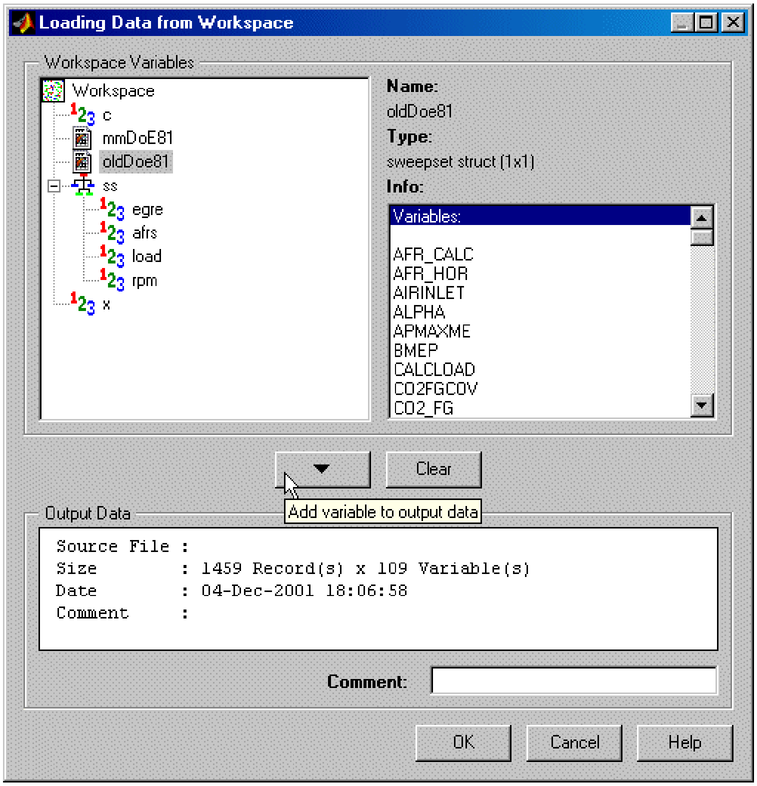
   The Data Editor appears.

   You can import variables from the MATLAB workspace by choosing **File -> Load data from the MATLAB workspace**.

   Alternatively, click the equivalent button in the toolbar (the left button with the MATLAB membrane on it — look for the tooltip).

   See "Data Editor Toolbar and Menus" on page 5-63.

The **Loading Data from Workspace** dialog appears.



Name:
oldDoe81
Type:
sweepset struct (1x1)
Info:

```
Source File :
Size        : 1459 Record(s) x 109 Variable(s)
Date        : 04-Dec-2001 18:06:58
Comment     :
```

Comment:

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 variable to output data** 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 add comments in the edit box; pressing Return adds them. Note that Return does not automatically click **OK**.

3 Click **Clear** to remove all data from the **Output Data** pane and start again.

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)

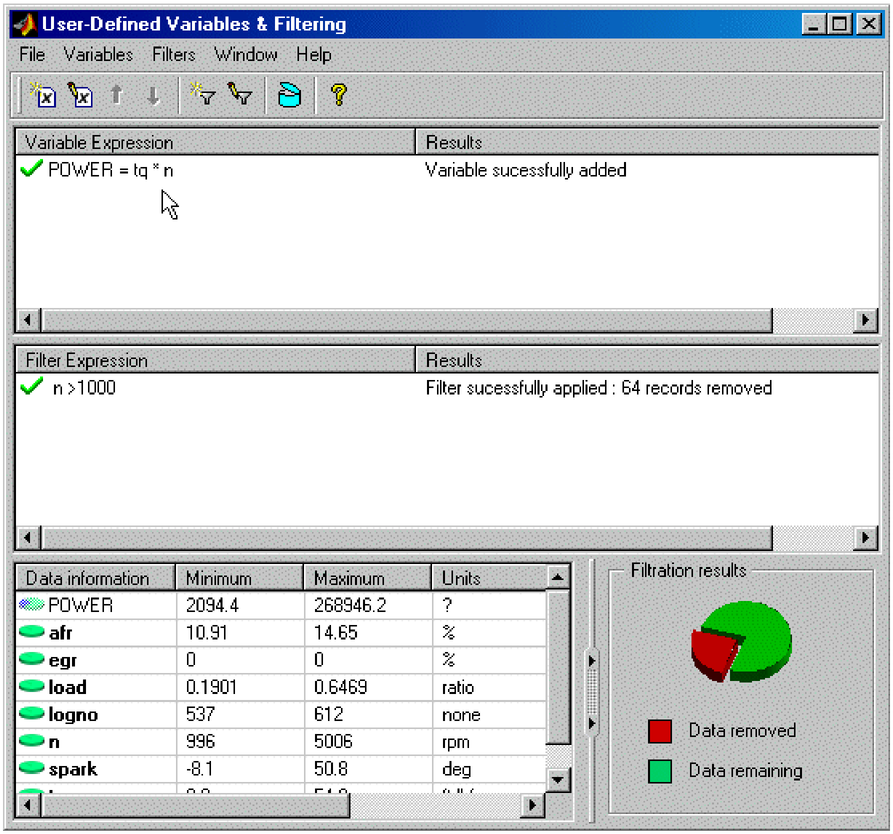
Click **Next**. The Wizard summary screen appears, showing the results of the merge. You can click **<Prev** to go back and change merging options.

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.

User-Defined Variables and Filtering

- These functions are available via the menu **Tools –> User-Defined Variables & Filtering** in the Data Editor.
- There is an equivalent toolbar button .



Variable Expression	Results
✓ POWER = tq * n	Variable sucessfully added

Filter Expression	Results
✓ n >1000	Filter sucessfully applied : 64 records removed

Data information	Minimum	Maximum	Units
POWER	2094.4	268946.2	?
afr	10.91	14.65	%
egr	0	0	%
load	0.1901	0.6469	ratio
logno	537	612	none
n	996	5006	rpm
spark	-8.1	50.8	deg

🟥 Data removed

🟩 Data remaining

The preceding example shows the **User-Defined Variables and Filtering** window after the creation of a new variable and a new filter (as in the “Data Editor Tutorial” on page 4-1).

New Variables

New variables you create in the Variable Editor appear in the **Variable Expression** pane and the **Data Information** pane.

- You create variables by doing the following:
  - Clicking the toolbar button
  - Selecting **Variable –> New Variable**
  - Alternatively, by selecting the **Variable Expression** pane by clicking in it, then pressing **Insert**

  The Variable Editor appears.

You can also load user-defined variables from Storage.

- You can edit 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 **Variables –> Edit Variable**
- You can also delete variables:
  - By selecting them in the **Variable Expression pane** and pressing **Delete**
  - By selecting **Variables –> Delete Variable**

New Filters

New filters you create in the Filter Editor appear in the **Filter Expression** pane, and their effects are shown graphically in the pie chart in the **Filtration Results** pane.

- You can create filters:
  - By clicking the toolbar button
  - By selecting **Filters –> New Filter**
  - Alternatively, by selecting the **Filter Expression** pane by clicking in it, then pressing **Insert**

  The Filter Editor appears.

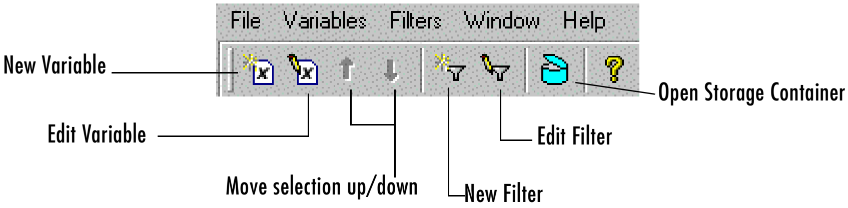
You can also load user-defined filters from Storage.

- Filters can be edited in the same way as variables:
  - Directly, after you select-click them in the **Filters Expression** pane, or by pressing **F2**
  - Using **Filters –> Edit Filter**, which opens the Filter Editor

- By double-clicking, which also opens the Filter Editor

- Delete filters by selecting them and pressing **Delete**, or choose **Filters –> Delete Filter**.

User-Defined Variables and Filters Toolbar



You can change the order of user variables in the list using the arrow buttons in the toolbar. You can define new variables with reference to each other, and they are added to the data in turn, in which case you can use the up/down arrows to reorder variables in legitimate sequence. See "Sequence of Variables" on page 4-12 in the "Data Editor Tutorial" for an example.

Variable and Filter Editors

Variable Editor

You can define new variables in terms of existing variables:

1 Click the toolbar button in the **User-Defined Variables and Filtering** window.

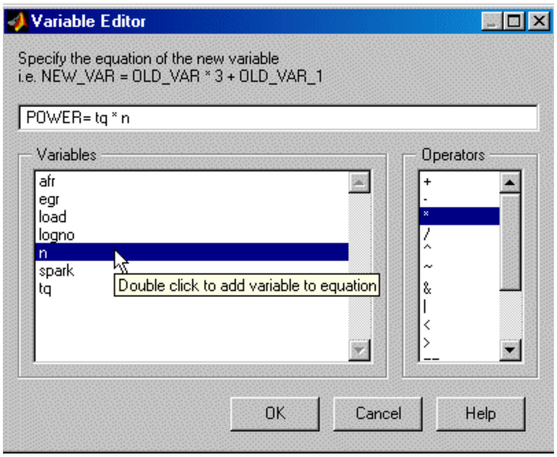
2 Choose the menu item **Variable –> New Variable**.

Alternatively, select the **Variable Expression** pane by clicking in it, then press **Insert**.

The **Variable Editor** dialog appears.

- 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 and operators by double-clicking them. In the case of variable names especially, this latter method avoids typing mistakes. Variable names are case sensitive.



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

POWER= tq * n

Double click to add variable to equation

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.

New variables you create in the Variable Editor then appear in the **Variable Expression** pane of the **User-Defined Variables and Filtering** window.

New variables, along with the original variables, appear in the list view (in the **Data Information** pane) and the new variable definition is included in the description field (to the right of the units column; you might have to scroll or resize to see it).

Filter Editor

A filter is the name for a constraint on the data set used to exclude some records. To reach the Filter Editor:

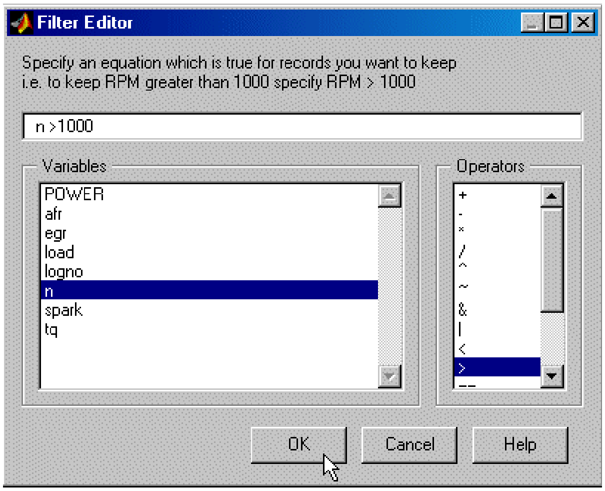
1 Click the toolbar button in the **User-Defined Variables and Filtering** window.

2 Select the menu item **Filters –> New Filter**.

Alternatively, select the **Filter Expression** pane by clicking in it, then press **Insert**.

The **Filter Editor** dialog appears.

You define the filter using logical operators on the existing variables.



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

n >1000

In the preceding example, n>1000, the effect of this filter is to keep all records with speed (**n**) greater than 1000.

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

This new filter appears in the **Filter Expression** pane of the **User-Defined Variables and Filtering** window.

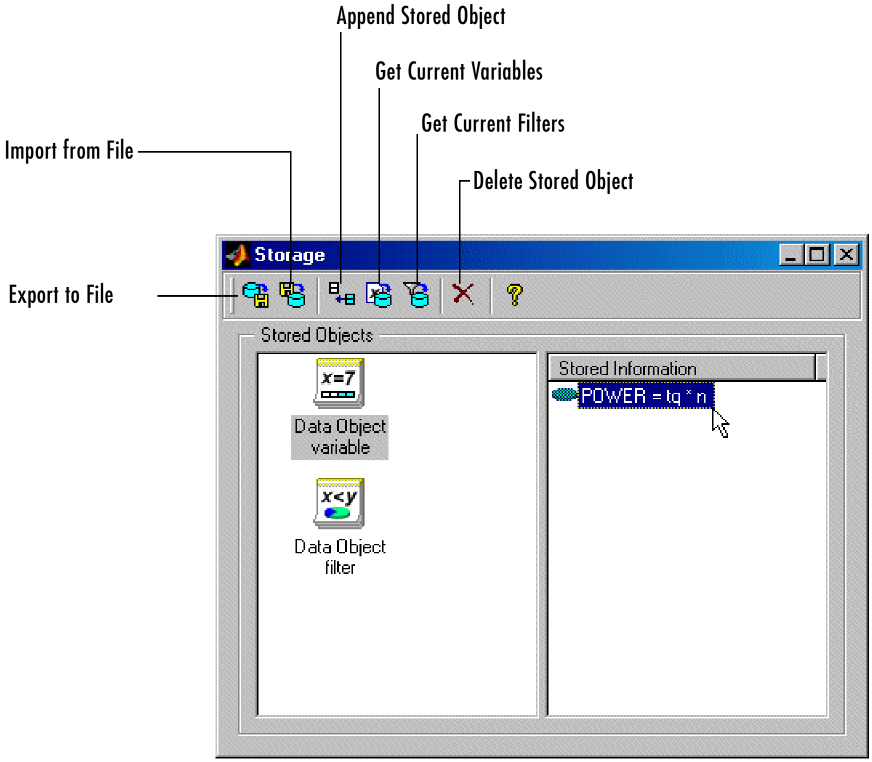
The **Filtration results** pie chart shows how many records have been removed by the imposition of this filter, and the number of records changes accordingly (for example, **Records 200/264** indicates that 64 records have been filtered out).

Storage

Storage allows user-defined variables and filters to be stored so that you can apply them to other data sets at any time.

You can open the **Storage** window from the **User-Defined Variables and Filtering** window by either

- Using the menu item **File –> Open Storage**
- Using the toolbar button



The button **Append Stored Object** adds the selected user-defined variable or filter in Storage to the current session. It appears in the **Variables and Filters** window. Double-clicking an object in Storage appends it to the current session.

The two buttons **Get Current Variables** and **Get Current Filters** 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 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.

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

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.

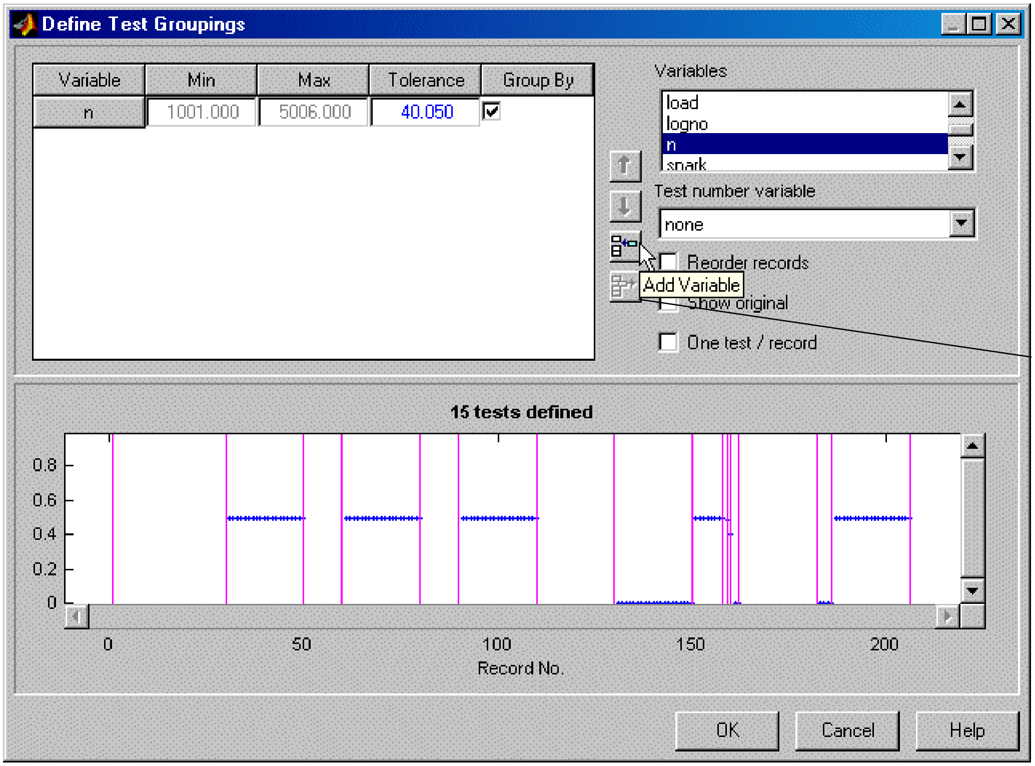
In the following example, the variable **n** is being used to define groups. The maximum and minimum values of **n** are displayed.

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 .


Define Test Groupings

Variable	Min	Max	Tolerance	Group By
n	1001.000	5006.000	40.050	☑

Variables

- load
- logno
- n
- spark

Test number variable: none

□ Reorder records

□ Show original

□ One test / record

15 tests defined

Remove variable

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

Dismissing the Data Editor after loading data from the test plan node automatically brings up the Data Wizard.

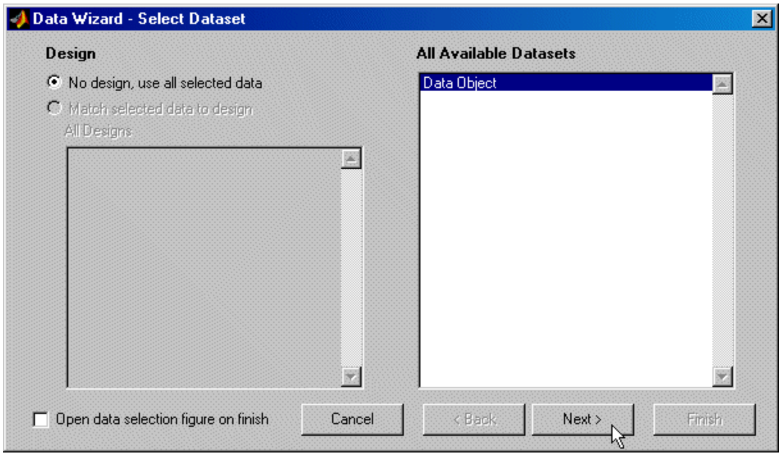
Alternatively, after setting up a new test plan, *if no data has been selected in that test plan*, then either of the following also brings up the Data Wizard.

- 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.

Note The check box at bottom left opens the **Data Selection** window on closing the Data Wizard, for matching data to a design. This check box appears on each screen of the Data Wizard.


Data Wizard - Select Dataset
Design
● No design, use all selected data
○ Match selected data to design
All Designs
All Available Datasets
Data Object
□ Open data selection figure on finish
Cancel
< Back
Next >
Finish

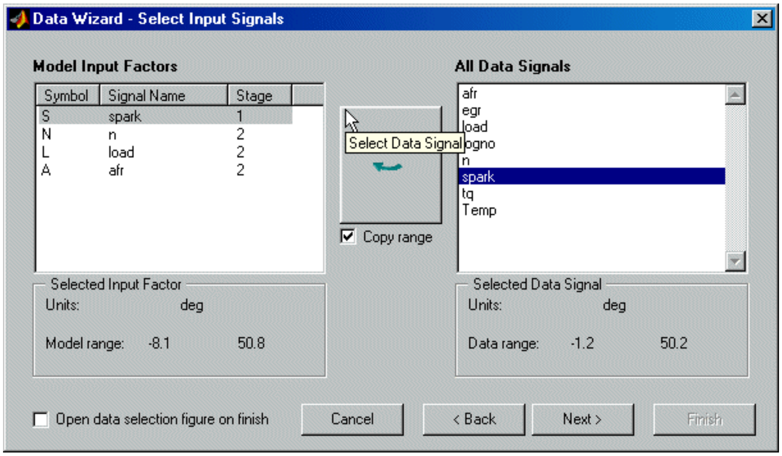
Step 2: Select Input Signals

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.

These options reappear when you select **Data –> Input Signals** in the Data Selection window.

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.



Data Wizard - Select Input Signals

Model Input Factors

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

All Data Signals

afr
egr
load
ogno
n
spark
tq
Temp

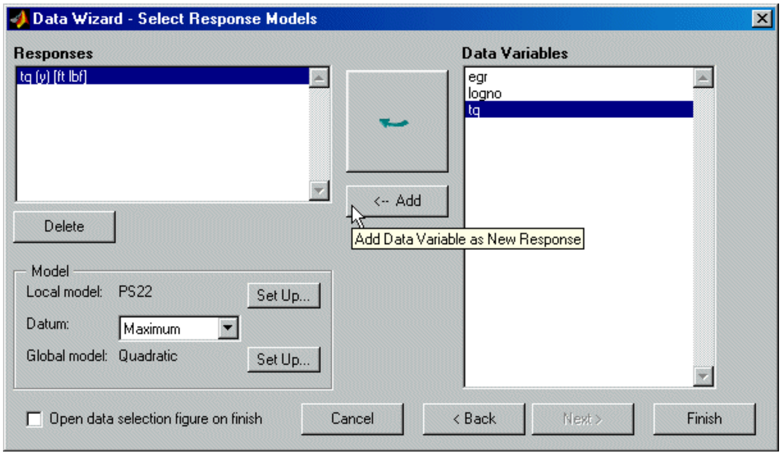
☑ Copy range

Selected Input Factor
Units: deg
Model range: -8.1 50.8

Selected Data Signal
Units: deg
Data range: -1.2 50.2

□ Open data selection figure on finish

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.

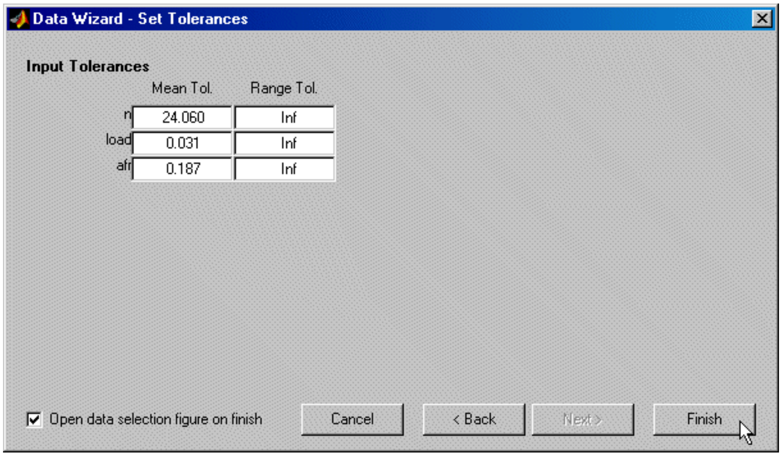
You can use **Delete** to remove selected responses. You can select datum models (if the local model supports them), and you can change the local and global models by using the **Set Up** buttons. See “Global Model Setup” on page 5-32 and “Local Model Setup” on page 5-46 for details.

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 also set tolerances later using the **Data** menu in the **Data Selection** window.

The **Data Selection** window appears by default when you close the Data Wizard while you are matching data to designs. If you are not matching data, it does not appear unless you select the check box.

The range tolerance sets the matching tolerance on signal range within a particular test. Setting the range tolerance to `Inf`, the default, causes matching to be performed using test means only.



Data Wizard - Set Tolerances

Input Tolerances

	Mean Tol.	Range Tol.
n	24.060	Inf
load	0.031	Inf
afr	0.187	Inf

☑ Open data selection figure on finish

Data Selection Window

The **Data Selection** window is primarily for matching data to designs.

To reach the **Data Selection** window:

- From the test plan node, click the **Select Data** 123 button in the toolbar.
- Alternatively, choose **TestPlan –> Select Data**.

The **Data Selection** window appears.

Note If the current test plan has no data loaded (in which case you can see ‘`No Data is selected` in the top right **Data Sets** pane) these actions open the Data Wizard instead.

You can also select the check box **Open data selection figure on finish** in the Data Wizard. This check box is selected by default when you select **Matching selected data to design**.

There is an option to match data to a design or select all available data from a particular data set.

Data Menu

- **New** — Opens the Data Wizard to select a new data set to match to the design.
- **Augment** — If there are data left over (that is, not matched to design points using current tolerances), this command opens a dialog where you can select tests to add. More data might be useful despite not matching original design points.
- **Tolerances** — Opens the **Matching Tolerances** dialog, where you can set limits for matching data points to design points. See also the relevant Data Wizard help: "Step 4: Set Tolerances" on page 5-80.
- **Input Signals** — Returns to the input signal setup (step 2 of the Data Wizard).
- **Setup Plots** — Opens a dialog where you can select variables to plot (including any variables in the data set, not just those used in modeling).

Matching to designs always occurs at the outer level.

Matching to Design (Two-Stage)

All matching is performed at the second stage.

- Tests with fewer observations than the number of local model parameters are not selected.
- The ten closest matches are displayed in the data points list.
- You can select data points with the check mark button and clear them with the cross button.

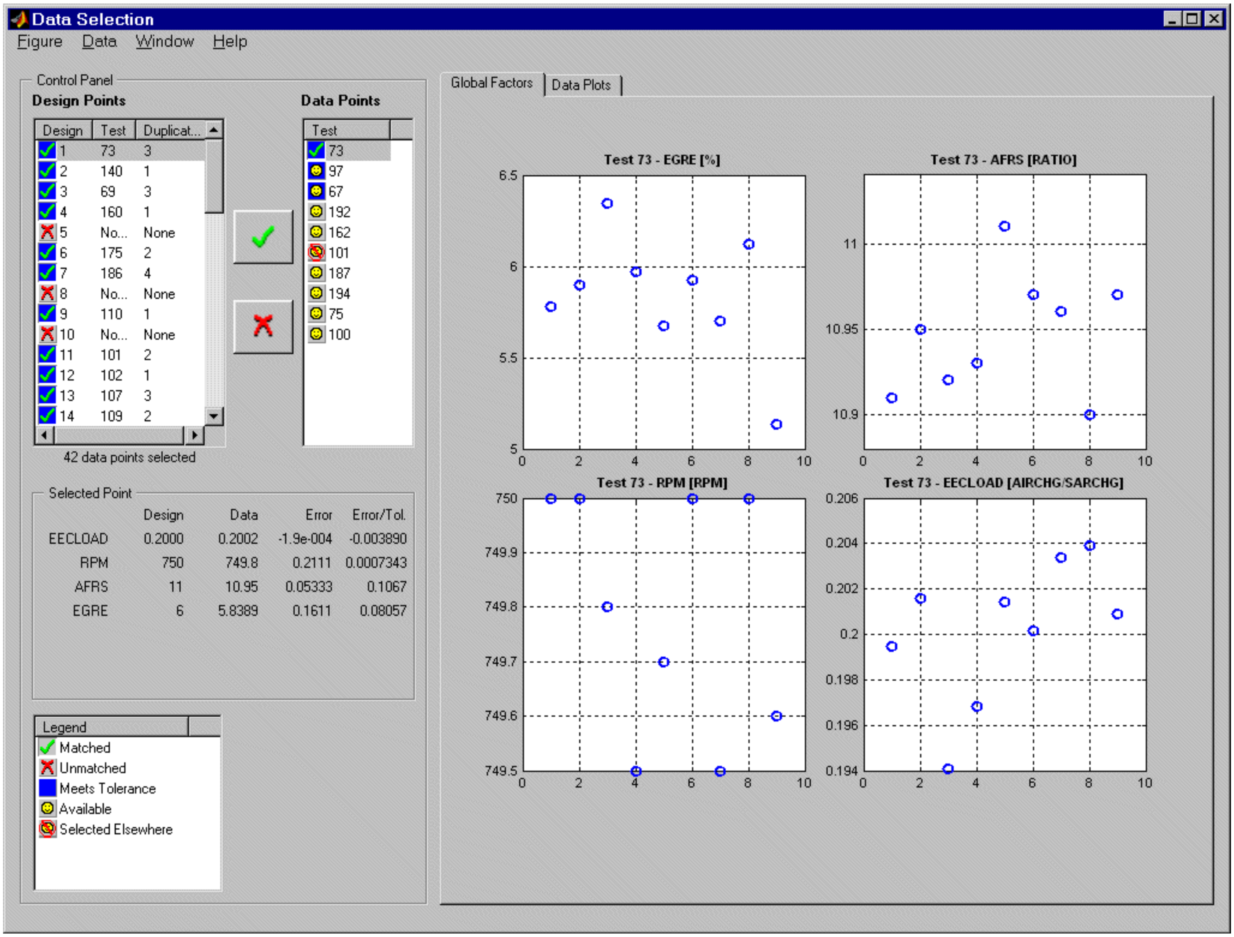
Icon information

- A selected data or design point has a check mark icon.
- If a point is within the matching tolerance it is colored blue.

- If a point is available to be matched it has a smiley face icon.
- If a point is not available to be matched (already matched) it has a smiley face icon with a line through it.
- If a point has not been matched it has a cross icon.

Plots

- A plot of the second stage inputs for the selected data point is shown on the first tab.
- The **Data Plots** tab allows the user to screen the data using other variables (for example, response or other unused variables).



Design Points

Design	Test	Duplicat...
✓ 1	73	3
✓ 2	140	1
✓ 3	69	3
✓ 4	160	1
✗ 5	No...	None
✓ 6	175	2
✓ 7	186	4
✗ 8	No...	None
✓ 9	110	1
✗ 10	No...	None
✓ 11	101	2
✓ 12	102	1
✓ 13	107	3
✓ 14	109	2

42 data points selected

Data Points

Test
✓ 73
☺ 97
☺ 67
☺ 192
☺ 162
⊘ 101
☺ 187
☺ 194
☺ 75
☺ 100

Selected Point

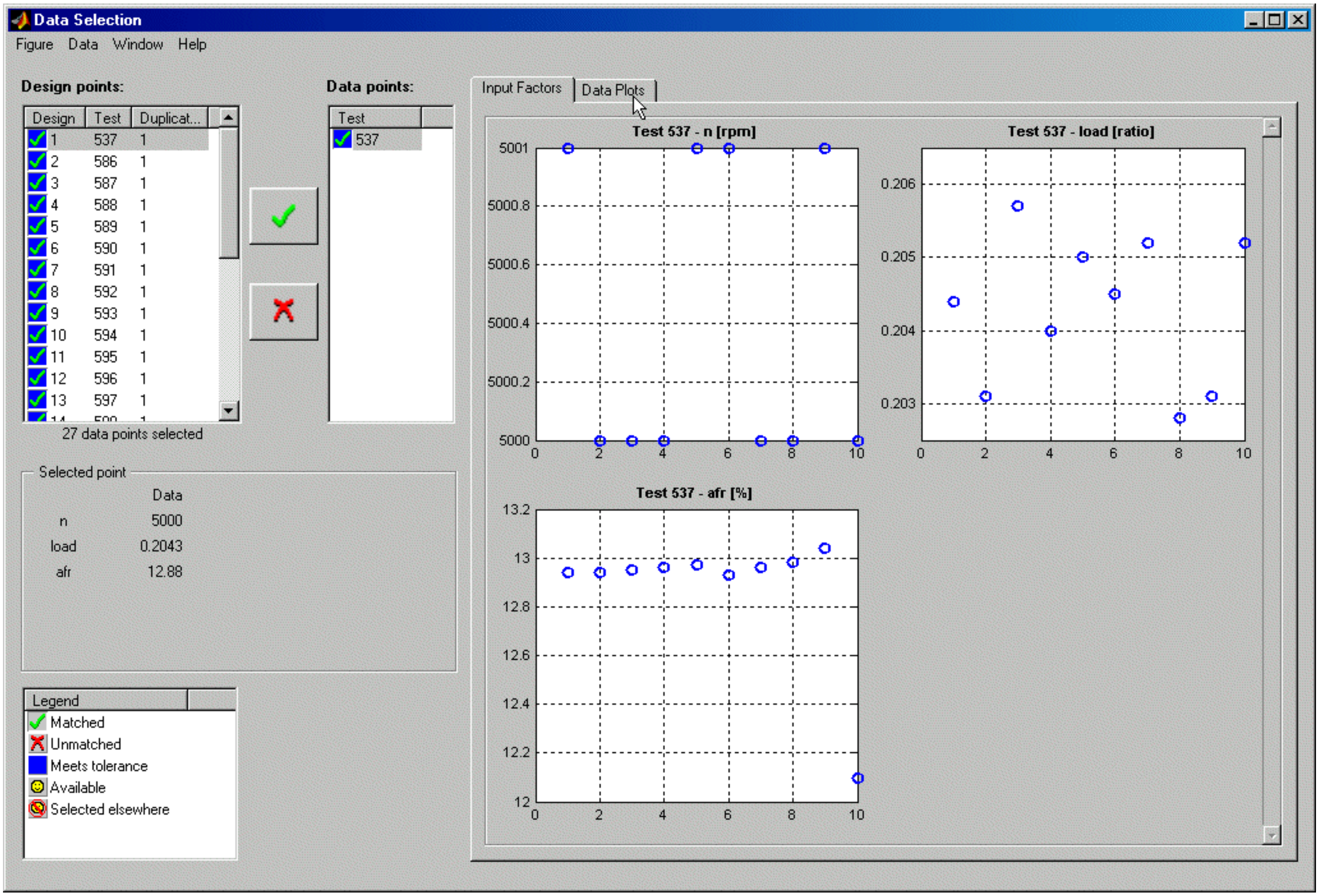
	Design	Data	Error	Error/Tol.
EECLOAD	0.2000	0.2002	-1.9e-004	-0.003890
RPM	750	749.8	0.2111	0.0007343
AFRS	11	10.95	0.05333	0.1067
EGRE	6	5.8389	0.1611	0.08057

Legend
✓ Matched
✗ Unmatched
■ Meets Tolerance
☺ Available
⊘ Selected Elsewhere

Selecting All the Data Set

Only one data point is shown in the data point list.

Tests with fewer observations than the number of stage 1 model parameters are not selected.



Design points:

Design	Test	Duplicat...
✓ 1	537	1
✓ 2	586	1
✓ 3	587	1
✓ 4	588	1
✓ 5	589	1
✓ 6	590	1
✓ 7	591	1
✓ 8	592	1
✓ 9	593	1
✓ 10	594	1
✓ 11	595	1
✓ 12	596	1
✓ 13	597	1

27 data points selected

Data points:

Test
✓ 537

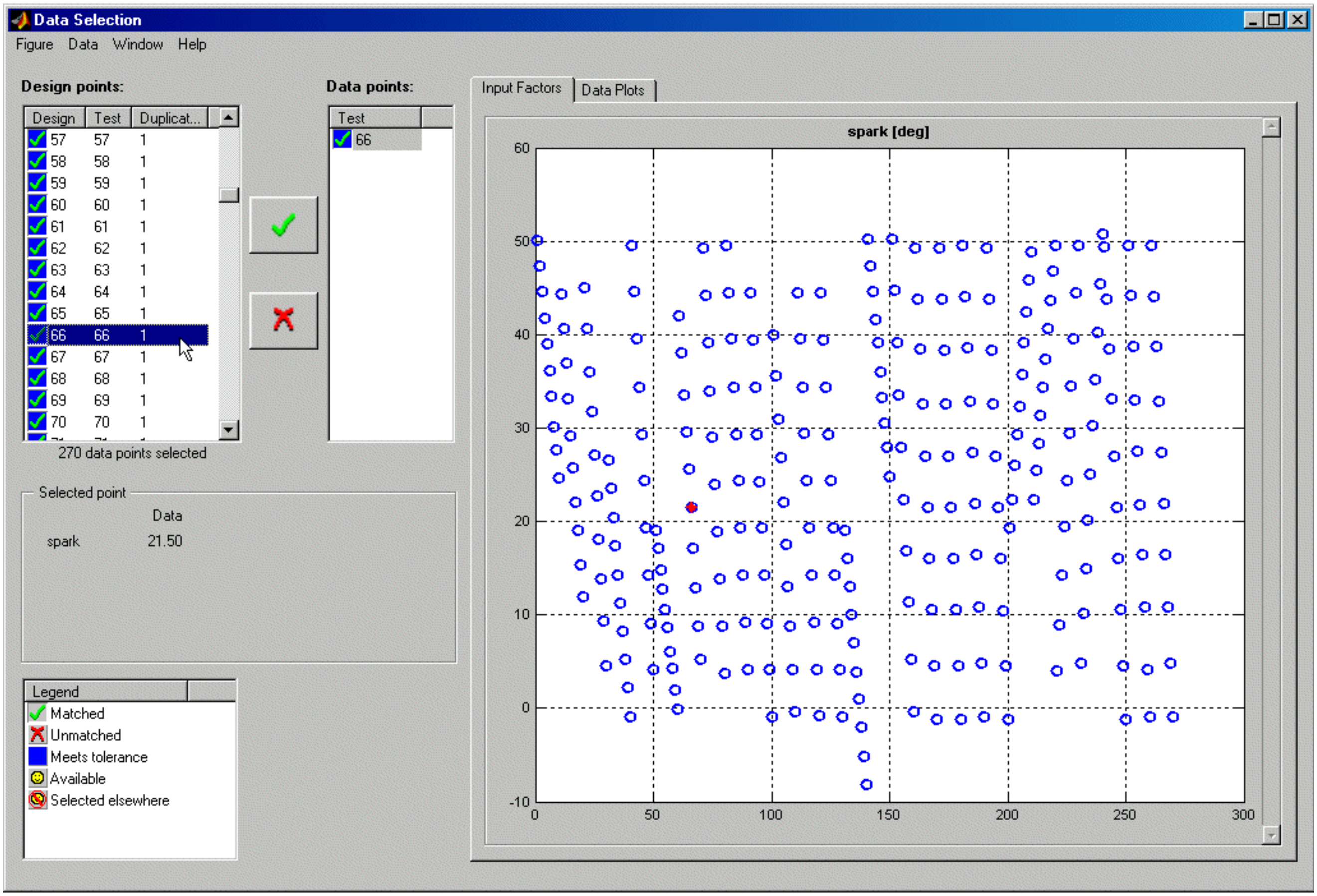
Selected point

	Data
n	5000
load	0.2043
afr	12.88

Legend
✓ Matched
☒ Unmatched
🟦 Meets tolerance
☺ Available
⊘ Selected elsewhere

Selecting Data for One-Stage Models

This shows plots of all the tests at once, with one point per test (to plot each test individually would only plot one point). The selected test is highlighted red in the plot.



Design points:

Design	Test	Duplicat...
✓ 57	57	1
✓ 58	58	1
✓ 59	59	1
✓ 60	60	1
✓ 61	61	1
✓ 62	62	1
✓ 63	63	1
✓ 64	64	1
✓ 65	65	1
✓ 66	66	1
✓ 67	67	1
✓ 68	68	1
✓ 69	69	1
✓ 70	70	1

270 data points selected

Data points:

Test
✓ 66

Selected point

	Data
spark	21.50

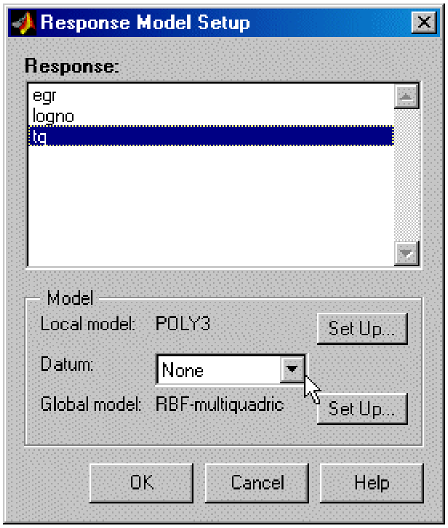
Legend
✓ Matched
☒ Unmatched
🟦 Meets tolerance
☺ Available
⊘ Selected elsewhere

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.)



Response Model Setup

Response:

egr
logno
tg

Model

Local model: POLY3

Datum: None

Global model: RBF-multiquadric

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 model below). 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 5-148.

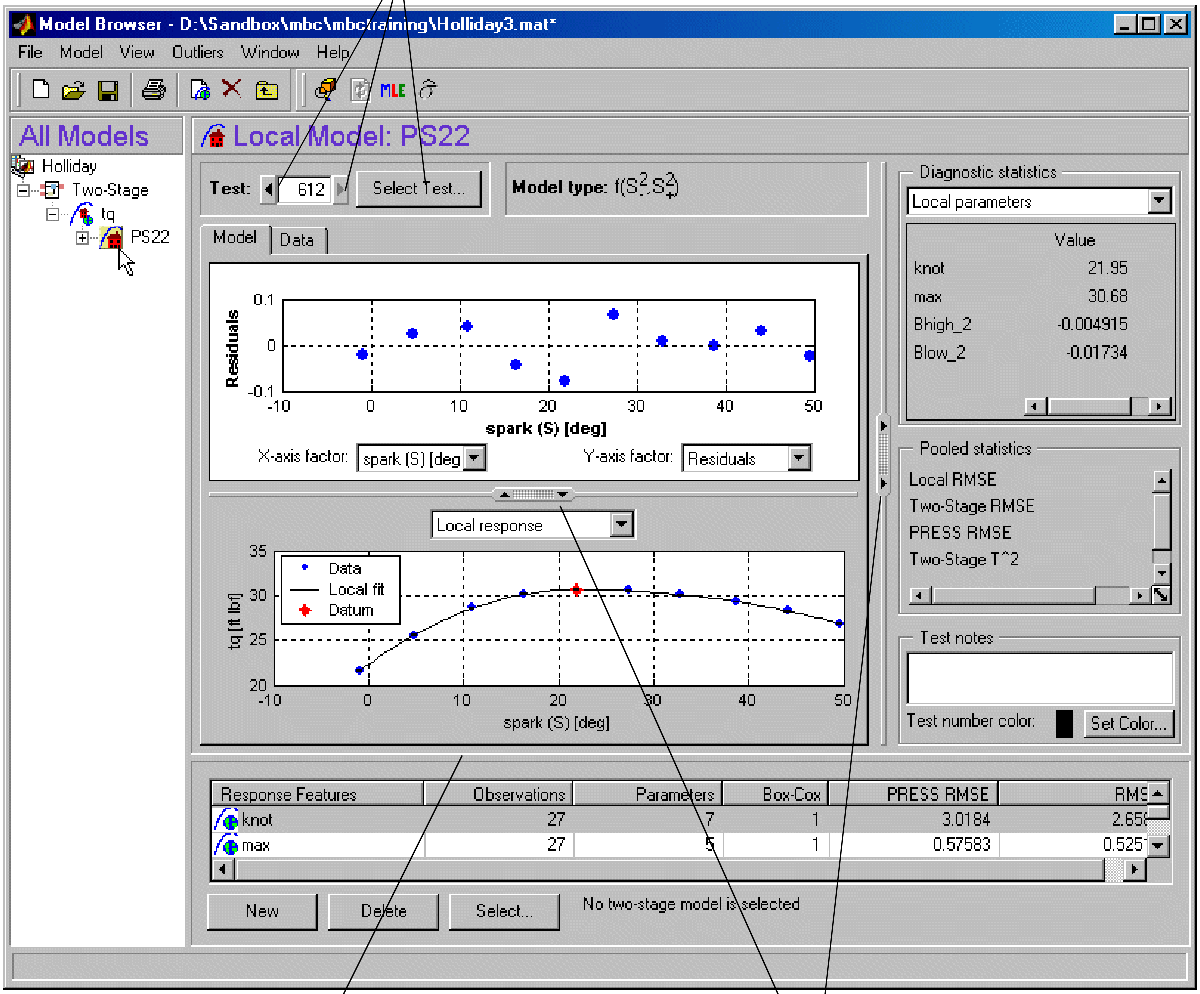
Local Level

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.

The following example shows a local model of torque/spark curves.

Click here to change test



Test: 612

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

X-axis factor: spark (S) [deg

Y-axis factor: Residuals

Diagnostic statistics: Local parameters

	Value
knot	21.95
max	30.68
Bhigh_2	-0.004915
Blow_2	-0.01734

Test number color: ■

Response Features	Observations	Parameters	Box-Cox	PRESS RMSE	RMS
knot	27	7	1	3.0184	2.65
max	27	5	1	0.57583	0.525

No two-stage model is selected

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

See “Local Level: Toolbar” on page 5-94 and “Local Level: Menus” on page 5-95 for details on these controls.

The default view is the **Model** tab, described below. See also “Data Tab” on page 5-103.

Upper scatter plots are replaced by an icon if you resize the Browser too small.

Local Special Plots

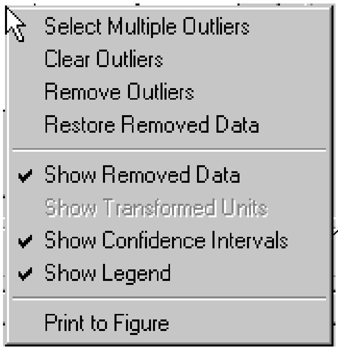
The lower plots are referred to as *special plots* as they can be different for different models.

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 5-144.

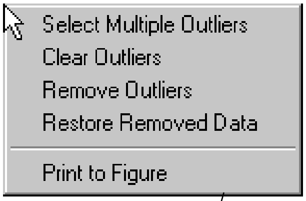
You can scroll through all the local models by using the up/down test buttons, type directly in the edit box, or go directly to test numbers by clicking **Select Test**.

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

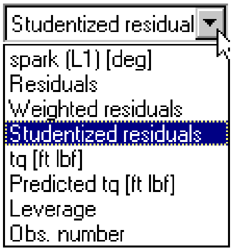
Above are the right-click context menus for both plots. On both plots you can manipulate outliers with all the same commands available in the **Outliers** menu. See “Outliers Menu (Local Level)” on page 5-100 for details.

On both plots 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.

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.

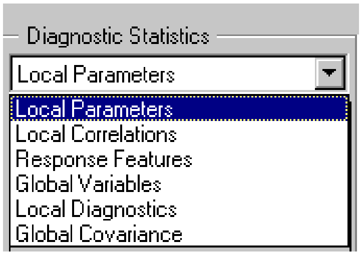
The statistics available for plotting are model dependent.



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 test), number of observations, degrees of freedom on the error, R squared, Cond(J *or* Sigma): the condition index for the Jacobian matrix *or* 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 “Definitions” on page 6-5.

Response Features List Pane

See the “Test Plans List Pane” on page 5-5 for information on the lower pane, which here contains a list of response features (with **New**, **Delete**, and **Select** buttons). The contents of this pane change in different views; it always contains the child nodes of the node selected in the model tree.

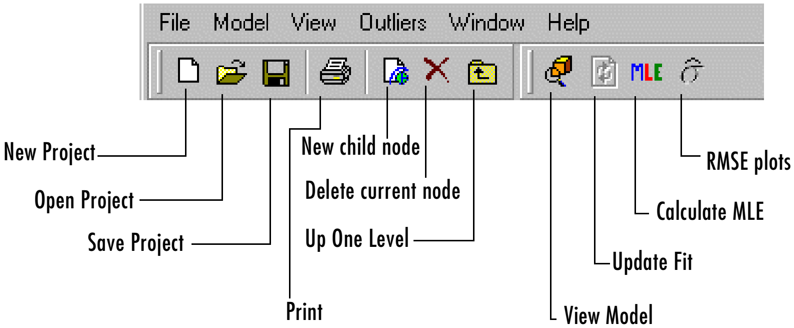
Here is a list of all the response features calculated for the local model. A two-stage model using the local and global models is formed by using **Select**. Click the **Select** button here to enter the **Model Selection** window. This step is required before two-stage models can be calculated.

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.

Local Level: Toolbar

This toolbar appears when a local node is selected in the model tree.



The seven left icons remain constant throughout the levels. They are for project and node management, and here the print icon is enabled as there are plots in this view. See “Project Level: Toolbar” on page 5-7 for details on these buttons.

- **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 5-137 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.

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 5-8.

Window and Help Menus

- The **Window** and **Help** menus remain the same throughout the Model Browser.

  See “Window Menu” on page 5-9 and “Help Menu” on page 5-9.

See also

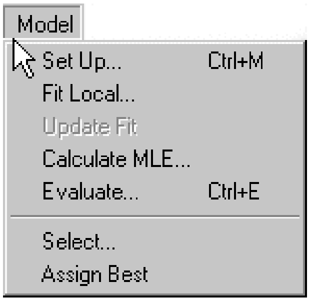
“Model Menu (Local Level)” on page 5-96

“View Menu (Local Level)” on page 5-99

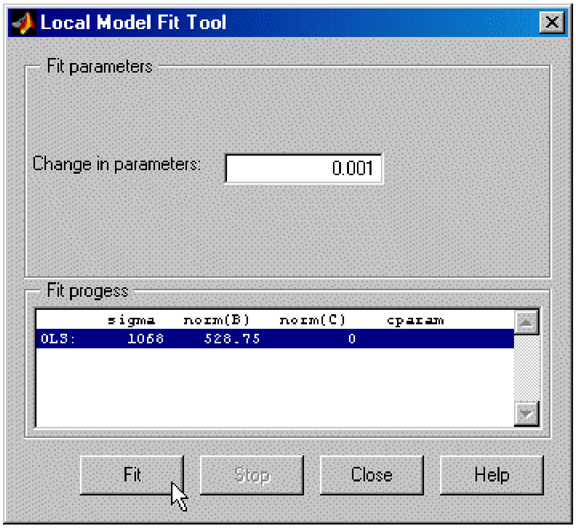
“Outliers Menu (Local Level)” on page 5-100

“Outlier Selection Criteria” on page 5-100

Model Menu (Local Level)



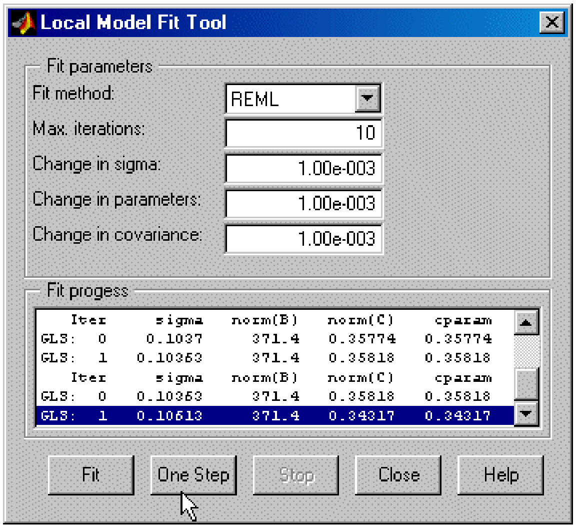
- **Set Up** — Opens the **Local Model Setup** dialog where you can change the model type. See “Local Model Setup” on page 5-46.
- **Fit Local**— Opens the **Local Model Fit Tool** dialog. Without covariance modeling, you see the following controls. This example shows the results after clicking **Fit** 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.



Change in parameters: 0.001

```
          sigma    norm(B)    norm(C)     cparam
OLS:       1068     528.75          0
```

- 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.



Fit method: REML

Max. iterations: 10

Change in sigma: 1.00e-003

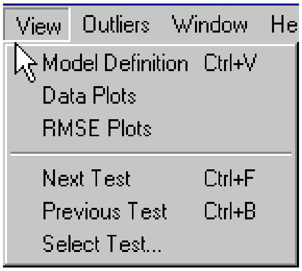
Change in parameters: 1.00e-003

Change in covariance: 1.00e-003

```
      Iter       sigma    norm(B)    norm(C)      cparam
GLS:   0      0.1037      371.4    0.35774     0.35774
GLS:   1     0.10363      371.4    0.35818     0.35818
      Iter       sigma    norm(B)    norm(C)      cparam
GLS:   0     0.10363      371.4    0.35818     0.35818
GLS:   1     0.10613      371.4    0.34317     0.34317
```

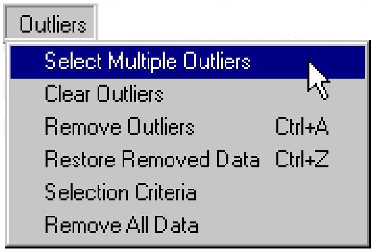
- **Update Fit** — Only enabled when outliers have been removed. The **Update Fit** option updates the global models to take these changes into account. This happens automatically when you select a different node. Duplicated in the toolbar. This can affect many models; see “Local Level: Toolbar” on page 5-94.
- **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 5-137 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” on page 5-117.
- **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 5-10.

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 response features. There is one value of standard error per sweep for each response feature. You can also plot these standard errors against the global variables to examine the global distribution of error.
- **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)

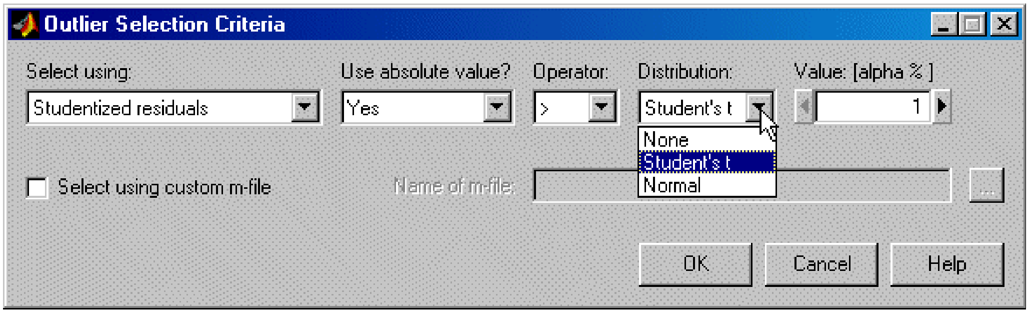


All the commands except **Remove All Data** are also available in the right-click context menus on all plots.

- **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** — 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.
- **Select 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.



Outlier Selection Criteria

Select using: Studentized residuals

Use absolute value? Yes

Operator: >

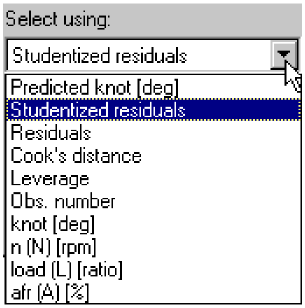
Distribution: Student's t

Value: [alpha %] 1

☐ Select using custom m-file

Name of m-file:

The drop-down menu labeled **Select using** contains all the available criteria, shown in the following example.



Select using: Studentized residuals

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).

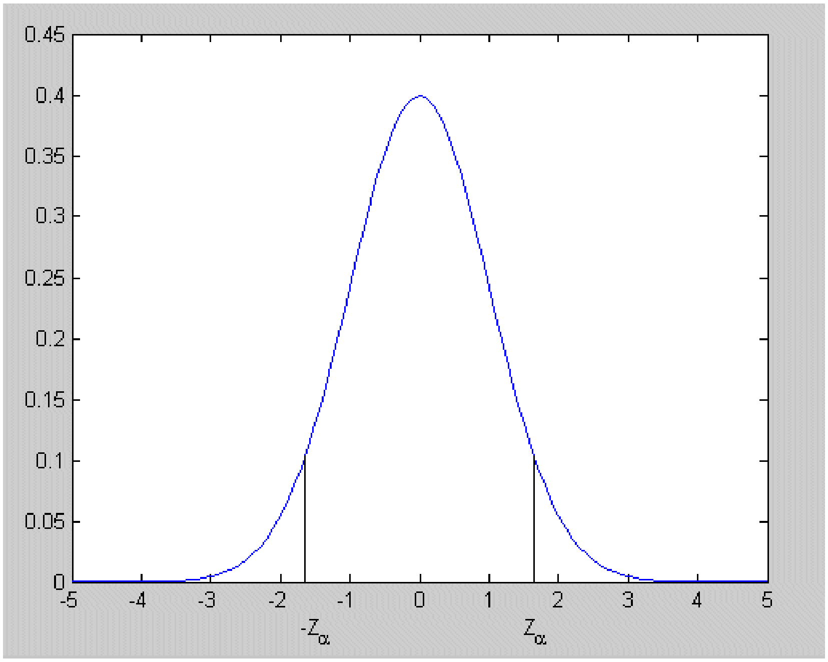
Distribution

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.

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.

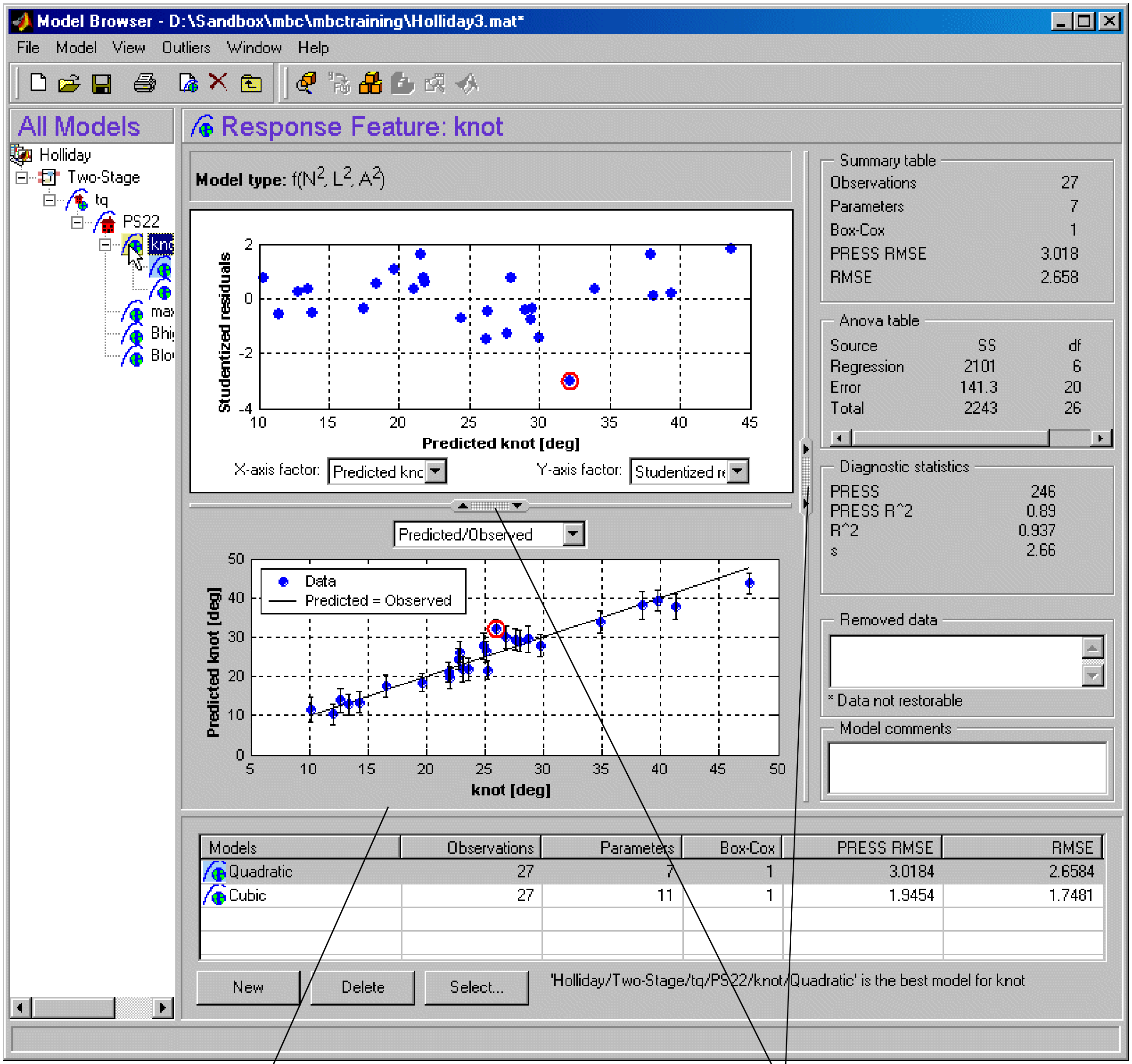
Here you can choose to view any of the data signals in the data set for the current test (including signals not being used in modeling). Choose variables from the list on the left and use the buttons to move them into the **Y**

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.

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 ( ), 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.



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

Summary table

Observations	27
Parameters	7
Box-Cox	1
PRESS RMSE	3.018
RMSE	2.658

Anova table

Source	SS	df
Regression	2101	6
Error	141.3	20
Total	2243	26

Diagnostic statistics

PRESS	246
PRESS R^2	0.89
R^2	0.937
s	2.66

Removed data

* Data not restorable

Model comments

Models	Observations	Parameters	Box-Cox	PRESS RMSE	RMSE
Quadratic	27	7	1	3.0184	2.6584
Cubic	27	11	1	1.9454	1.7481

'Holliday/Two-Stage/tq/PS22/knot/Quadratic' is the best model for knot

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

See “Global Level: Toolbar” on page 5-110 and “Global Level: Menus” on page 5-113 for details on these controls.

The upper scatter plots are replaced by an icon if you resize the Browser too small.

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.

For information on the displayed statistics, see “Linear Model Statistics Displays” on page 6-22 and “Definitions” on page 6-5. Here you can find explanations of the information found in the Summary table, ANOVA table, Diagnostic Statistics pane and (for two-stage models) the Pooled Statistics pane.

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 5-105) 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.

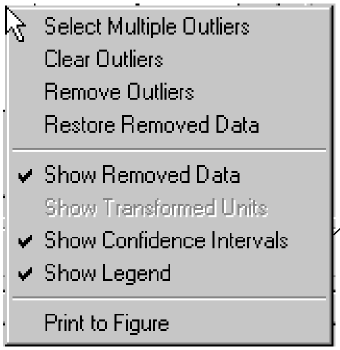
Note Right-click a point in either the special or scatter plot to open a figure plot of that particular test (for example, torque against spark).

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.

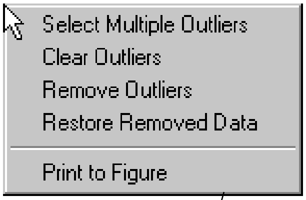
Global Scatter Plots

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 statistics available for plotting are model dependent.



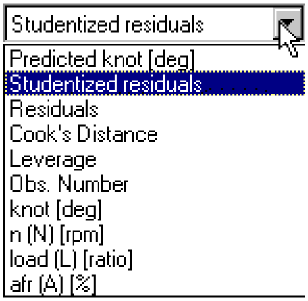
Special plot right-click context menu



Scatter plot right-click context menu

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**. See “Outliers Menu (Local Level)” on page 5-100 for details.

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.



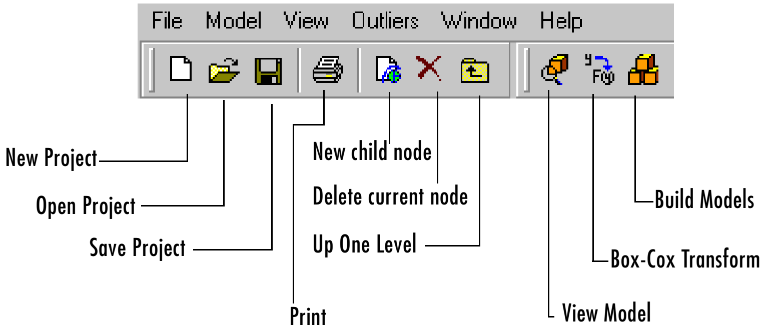
Shown is an example drop-down menu on the scatter plot for changing *x* and *y* factors. In this example `knot` is the response feature node selected. Therefore the model output is knot, so `knot` and `Predicted knot` are available in the menu. (For child nodes of `knot`, the model output is still `knot`.) 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.

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 5-111.



The seven left icons remain constant throughout the levels. They are for project and node management, and here the print icon is enabled, as there are plots in this view. See "Project Level: Toolbar" on page 5-7 for details on these buttons.

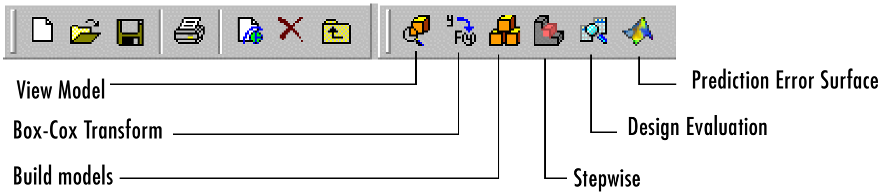
- **View Model** — Opens a dialog displaying the terms in the current model.
- **Box-Cox Transform** — Opens the Box-Cox Transformation plots, where you can minimize SSE to try to improve the fit. See "Box-Cox Transformation" on page 6-18 for statistical details.
- **Build Models** — Opens the **Build 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.

These three 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 nine left buttons (up to Box-Cox Transform) appear for all response feature models and one-stage models. See “Global Level: Toolbar” on page 5-110 for details on these buttons. The right buttons change according to model type.

Linear Model and Multiple Linear Models

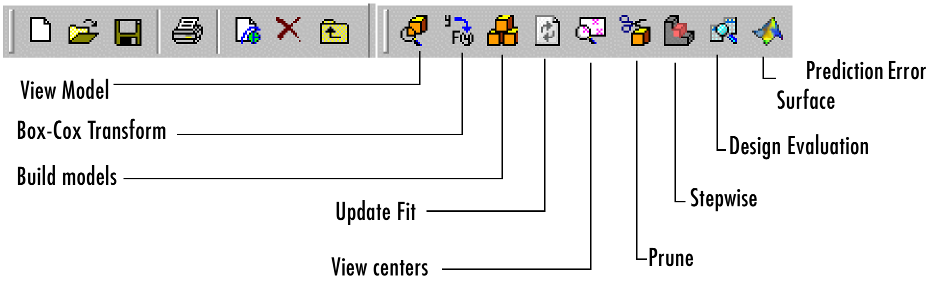


- **Stepwise** — This opens the **Stepwise Regression** window, where you can view the effects of removing and restoring model terms on the PRESS statistic (Predicted Error Sum of Squares), which is a measure of the predictive quality of a model. You can also use Min PRESS to remove all at once model terms that do not improve the predictive qualities of the model. See “Stepwise Regression Techniques” on page 6-13 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 6-27.
- **Prediction Error Surface** – Opens the Prediction Error Variance Viewer. See “Prediction Error Variance Viewer” on page 5-193.

Free-Knot Spline Models

Free knot spline models do not have any model-specific tools, just the standard View Model, Box-Cox Transform, and Build Models.

Radial Basis Function Models

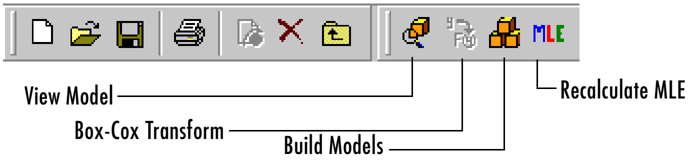


- **Update Fit** refits the RBF widths and centers. See “Fitting Routines” on page 7-12 in the Radial Basis Functions chapter.
- **View Centers** opens a dialog where you can view the position of the radial basis function’s centers graphically and in table form.
- **Prune** opens the Number of Centers Selector where you can minimize various error statistics by decreasing the number of centers. See “Prune Functionality” on page 7-21.
- **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 6-27.
- **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 5-105 for other functionality in this view.



At this point the **New child node** and Box-Cox 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 5-137 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.

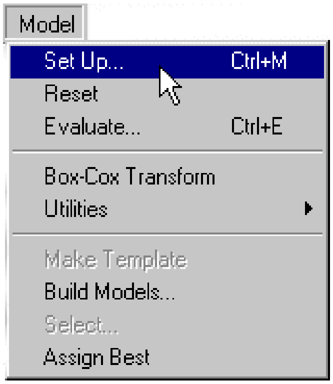
See “File Menu” on page 5-8.

Window and Help Menus

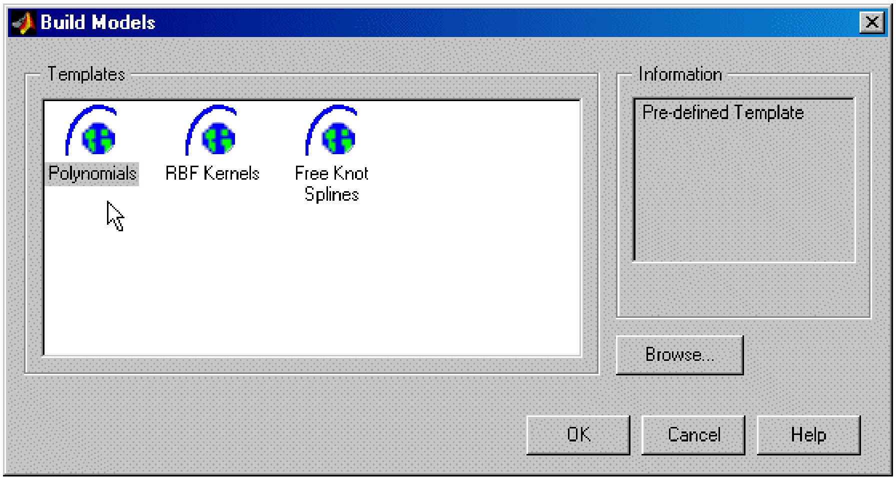
The **Window** and **Help** menus have the same form throughout the Model Browser.

See “Window Menu” on page 5-9 and “Help Menu” on page 5-9.

Model Menu (Global Level)



- **Set Up** opens the **Global Model Setup** dialog, where you can change the model type. See "Global Model Setup" on page 5-32.
- **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.
- **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 6-18 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 select a template and build a selection of models as child nodes of the current node.



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.

When you select a template, 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 7-4. 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.

- **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” on page 5-117.
- **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 5-10.

View Menu (Global Level)

At this level there are only two items:

- **Model Definition** opens the **Model Viewer** dialog displaying the model terms.
- **Test Numbers** turns test numbers on and off on 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**. See "Outliers Menu (Local Level)" on page 5-100.

At global 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.

The **Model Selection** window allows visual comparison of several models in many ways, depending on the type of model:

- "Tests View" on page 5-120
- "Predicted/Observed View" on page 5-122
- "Response Surface View" on page 5-124
- "Likelihood View" on page 5-127
- "RMSE View" on page 5-129
- "Residuals View" on page 5-132
- "Cross Section View" on page 5-133

Select

The **Select** button is under the list view in the pane at the bottom of the 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).

Model Selection Window

The **Model Selection** window comprises several different views depending on the type of models being compared:

- “Tests View” on page 5-120
- “Predicted/Observed View” on page 5-122
- “Response Surface View” on page 5-124

- “Likelihood View” on page 5-127
- “RMSE View” on page 5-129
- “Residuals View” on page 5-132
- “Cross Section View” on page 5-133

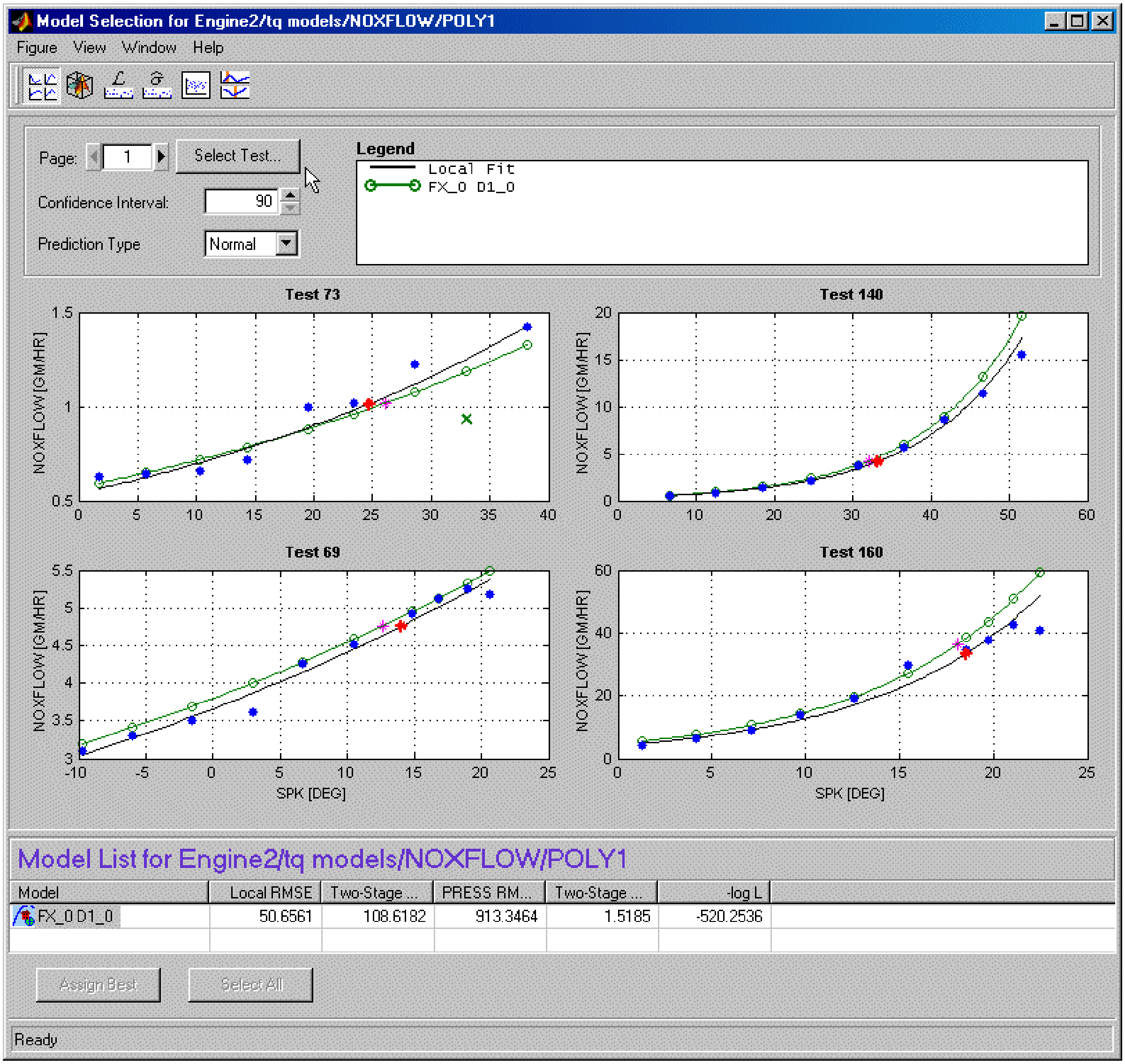
The **Assign Best** button at the bottom of the window marks the currently selected model as best.

To print the current view, use the **Figure/Print** menu item or its hot key equivalent **Ctrl**+P.

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.



Page: 1

Confidence Interval: 90

Prediction Type: Normal

Model List for Engine2/tq models/NOXFLOW/POLY1

Model	Local RMSE	Two-Stage ...	PRESS RM...	Two-Stage ...	-log L
FX_0 D1_0	50.6561	108.6182	913.3464	1.5185	-520.2536

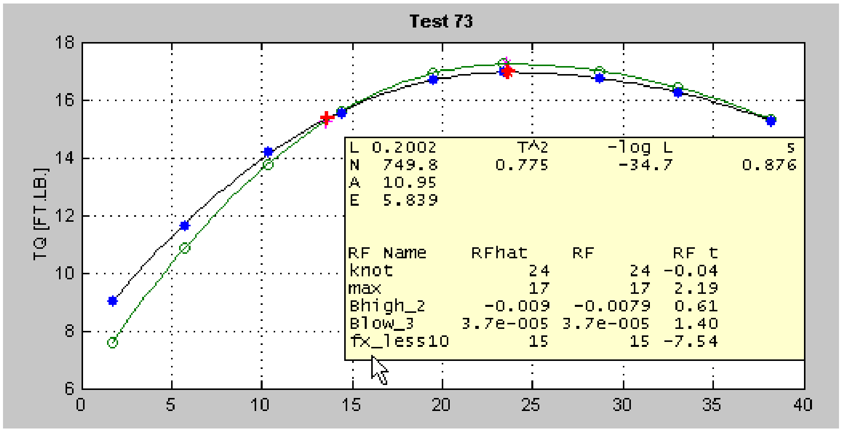
Ready

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:



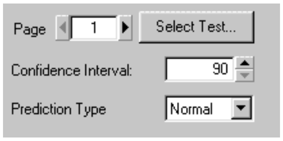
```
L  0.2002        T^2      -log L          5
N  749.8        0.775      -34.7       0.876
A  10.95
E  5.839


RF Name      RFhat     RF        RF t
knot            24          24  -0.04
max             17          17   2.19
Bhigh_2     -0.009   -0.0079   0.61
Blow_3    3.7e-005  3.7e-005   1.40
fx_less10       15          15  -7.54
```

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 6-22 and “Stepwise Regression Techniques” on page 6-13.



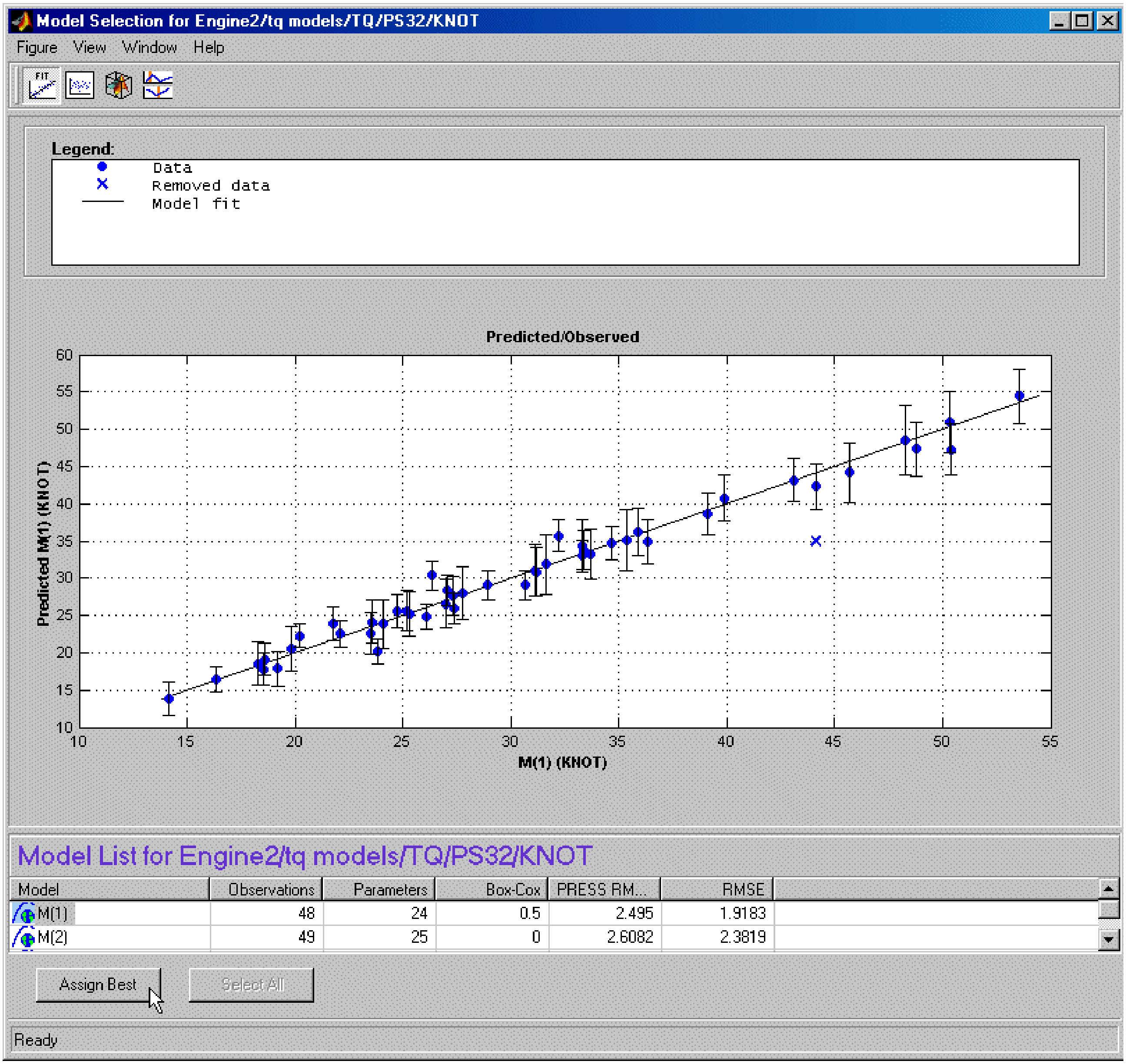
Page 1

Confidence Interval: 90

Prediction Type Normal

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:



Model Selection for Engine2/tq models/TQ/PS32/KNOT

Model List for Engine2/tq models/TQ/PS32/KNOT

Model	Observations	Parameters	Box-Cox	PRESS RM...	RMSE
M(1)	48	24	0.5	2.495	1.9183
M(2)	49	25	0	2.6082	2.3819

Ready

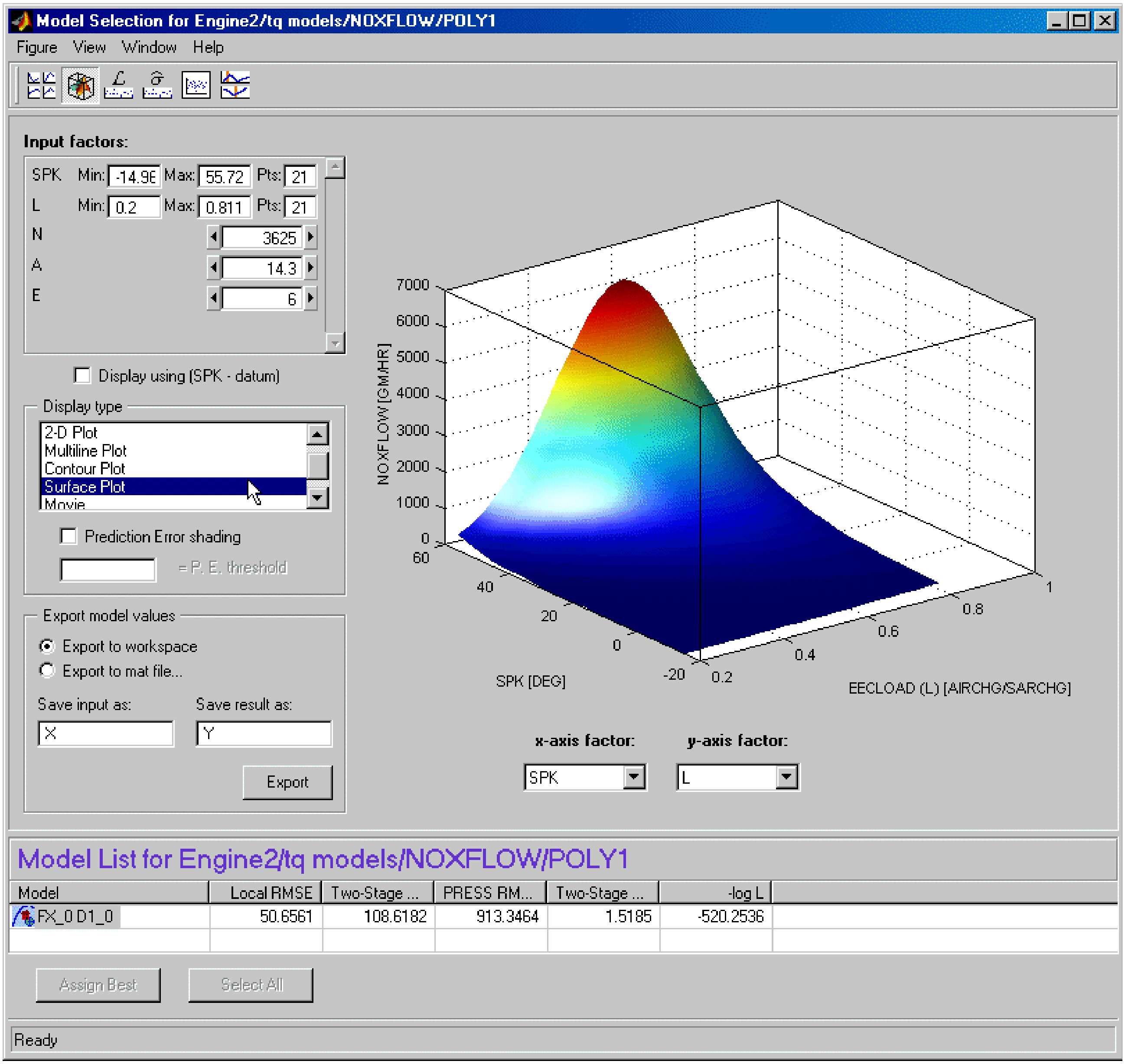
The plot shows the data used to fit this model (values of knot), against the values found by evaluating the model (here Predicted knot) at these data points. The straight black line is the plot of y=x. If the model fitted the data exactly, all the blue points would lie on this line. The error bars show 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

You can change the view using the **View** menu or by clicking the buttons of the toolbar. The next view along the toolbar shows the model surface in a variety of ways.



Model Selection for Engine2/tq models/NOXFLOW/POLY1

Input factors:

SPK Min: -14.96 Max: 55.72 Pts: 21

L Min: 0.2 Max: 0.811 Pts: 21

N 3625

A 14.3

E 6

☐ Display using (SPK - datum)

Display type: Surface Plot

☐ Prediction Error shading

= P. E. threshold

● Export to workspace

○ Export to mat file...

Save input as: X

Save result as: Y

x-axis factor: SPK

y-axis factor: L

Model List for Engine2/tq models/NOXFLOW/POLY1

Model	Local RMSE	Two-Stage ...	PRESS RM...	Two-Stage ...	-log L
FX_0 D1_0	50.6561	108.6182	913.3464	1.5185	-520.2536

Ready

The default view is a 3-D plot of the model surface. This model has four dependent factors; you can see these in the controls at the top left and in the menus below the plot.

You use the drop-down menus to select the input factors for 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 (SPK - datum) — If a datum model is being displayed, this check box appears. The datum variable here is spark angle, `SPK`. 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 5-88.

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 above).

  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.

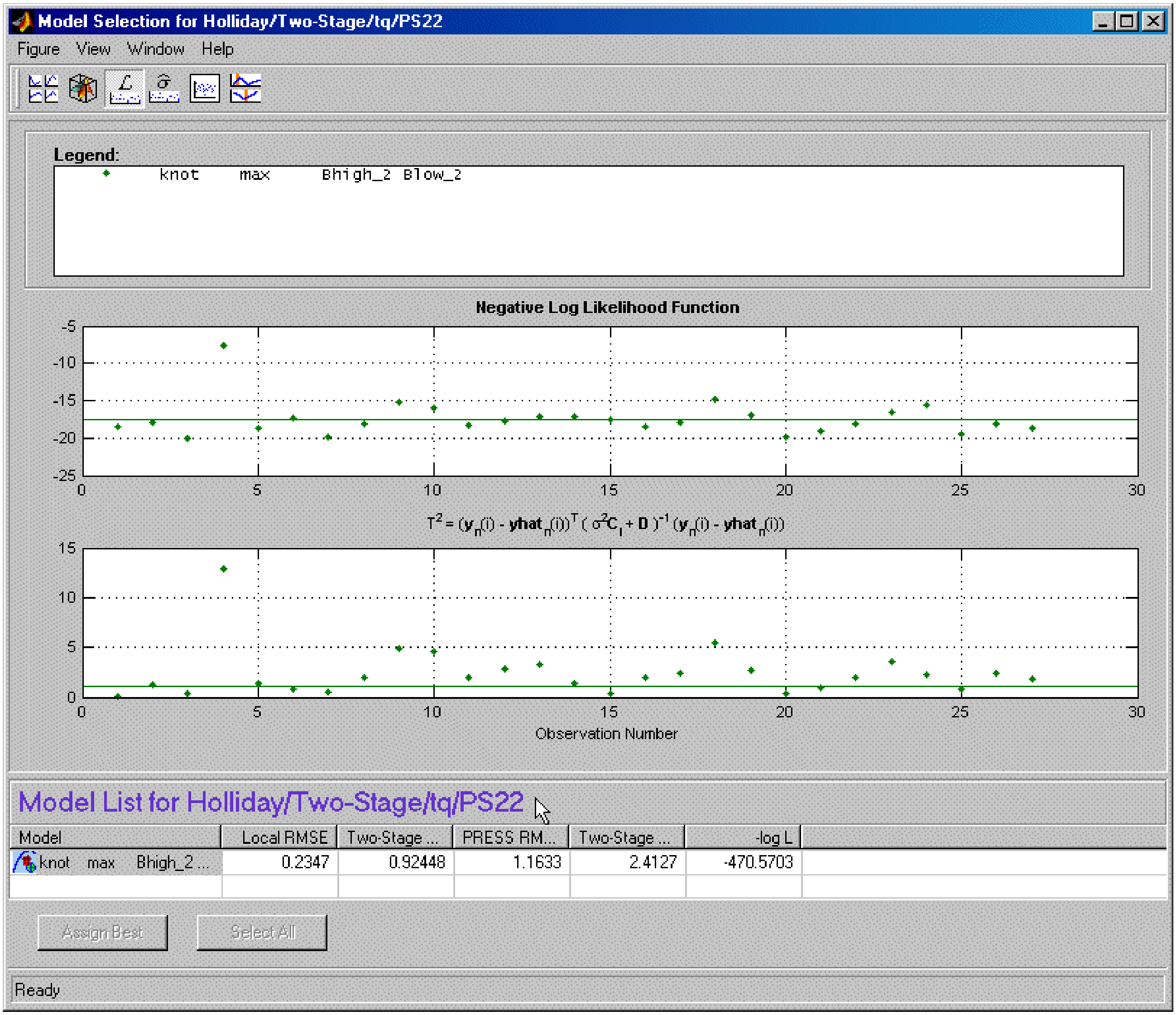
  P. E. Threshold — To see good color contrast in the range of PE of interest, you can set the upper limit of the coloring range. All values above this threshold are colored as `maximum P.E.`

- 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.

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).


Model Selection for Holliday/Two-Stage/tq/PS22
Figure View Window Help
Legend:
knot max Bhigh_2 Blow_2
Negative Log Likelihood Function
-5
-10
-15
-20
-25
0
5
10
15
20
25
30
$T^2 = (y_n(i) - yhat_n(i))^T (\sigma^2 C_i + D)^{-1} (y_n(i) - yhat_n(i))$
15
10
5
0
0
5
10
15
20
25
30
Observation Number
Model List for Holliday/Two-Stage/tq/PS22
Assign Best
Select All
Ready

Model	Local RMSE	Two-Stage ...	PRESS RM...	Two-Stage ...	-log L
knot max Bhigh_2 ...	0.2347	0.92448	1.1633	2.4127	-470.5703

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**.

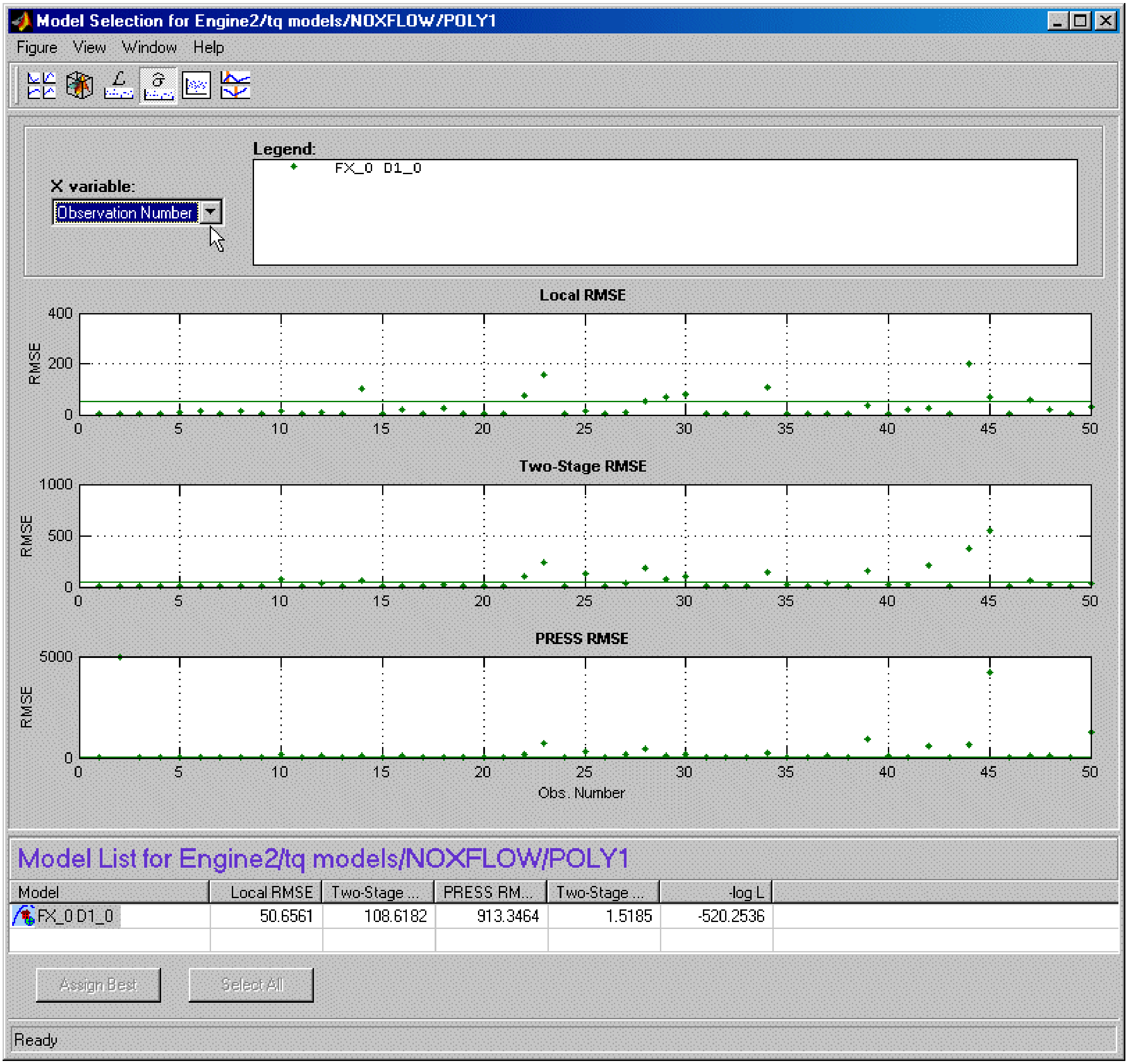
The likelihood view allows several models to be displayed simultaneously; click the **Select All** button at the bottom of the window or, in the model list view, **Shift**+click or **Ctrl**+click to select the models for display.

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.



Model Selection for Engine2/tq models/NOXFLOW/POLY1

X variable: Observation Number

Legend:

● FX_0 D1_0

Model List for Engine2/tq models/NOXFLOW/POLY1

Model	Local RMSE	Two-Stage ...	PRESS RM...	Two-Stage ...	-log L
FX_0 D1_0	50.6561	108.6182	913.3464	1.5185	-520.2536

Ready

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.

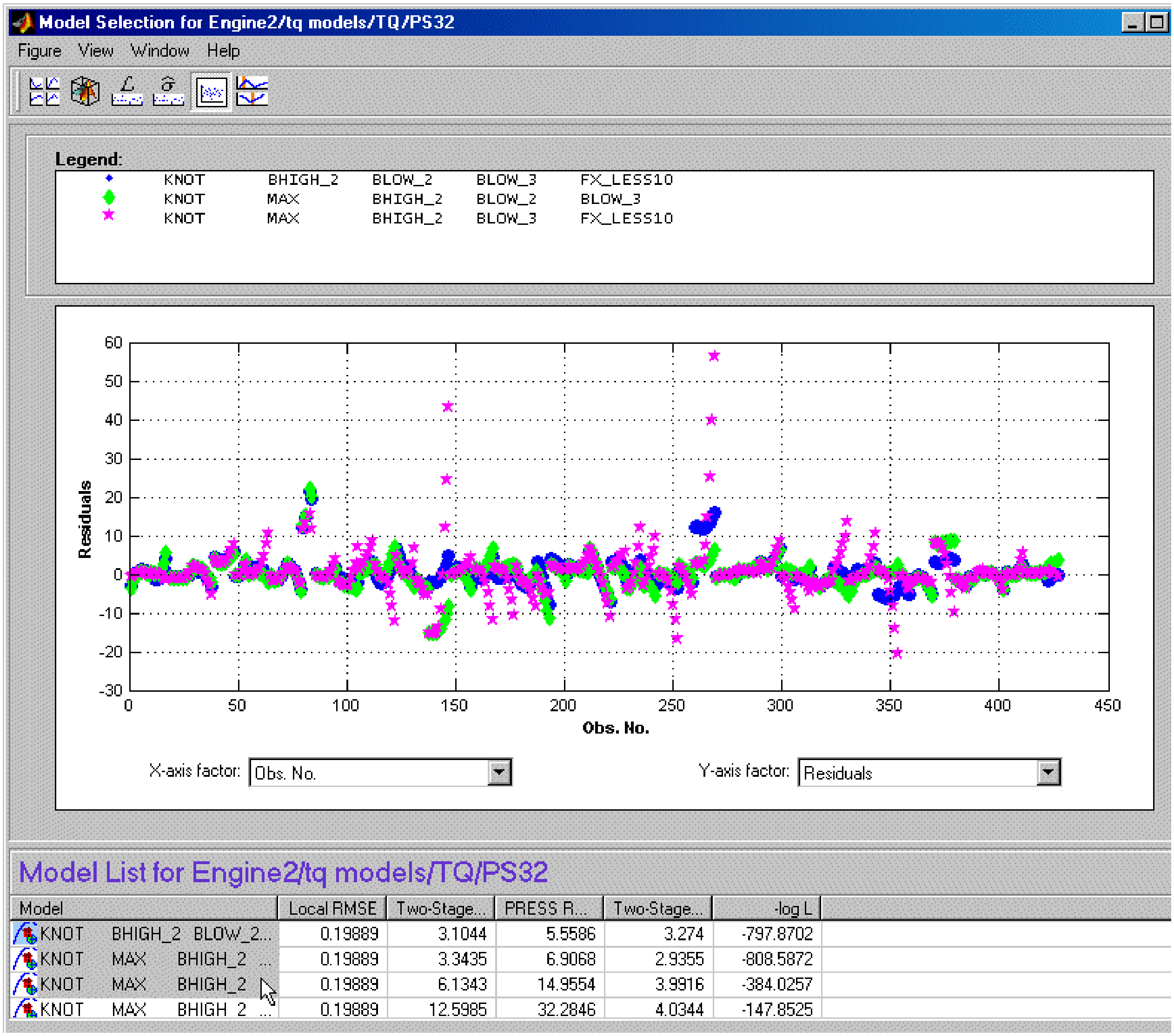
Two-Stage RMSE shows the root mean squared error in the two-stage model fit to the data for each test. You should expect this to be higher than the local RMSE.

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 6-22.

Residuals View

The residuals view shows the scatter plots of observation number, predicted and observed response, input factors, and residuals.



Legend:

●	KNOT	BHIGH_2	BLOW_2	BLOW_3	FX_LESS10
◆	KNOT	MAX	BHIGH_2	BLOW_2	BLOW_3
★	KNOT	MAX	BHIGH_2	BLOW_3	FX_LESS10

X-axis factor: Obs. No.

Y-axis factor: Residuals

Model List for Engine2/tq models/TQ/PS32

Model	Local RMSE	Two-Stage...	PRESS R...	Two-Stage...	-log L
KNOT BHIGH_2 BLOW_2...	0.19889	3.1044	5.5586	3.274	-797.8702
KNOT MAX BHIGH_2 ...	0.19889	3.3435	6.9068	2.9355	-808.5872
KNOT MAX BHIGH_2 ...	0.19889	6.1343	14.9554	3.9916	-384.0257
KNOT MAX BHIGH_2 ...	0.19889	12.5985	32.2846	4.0344	-147.8525

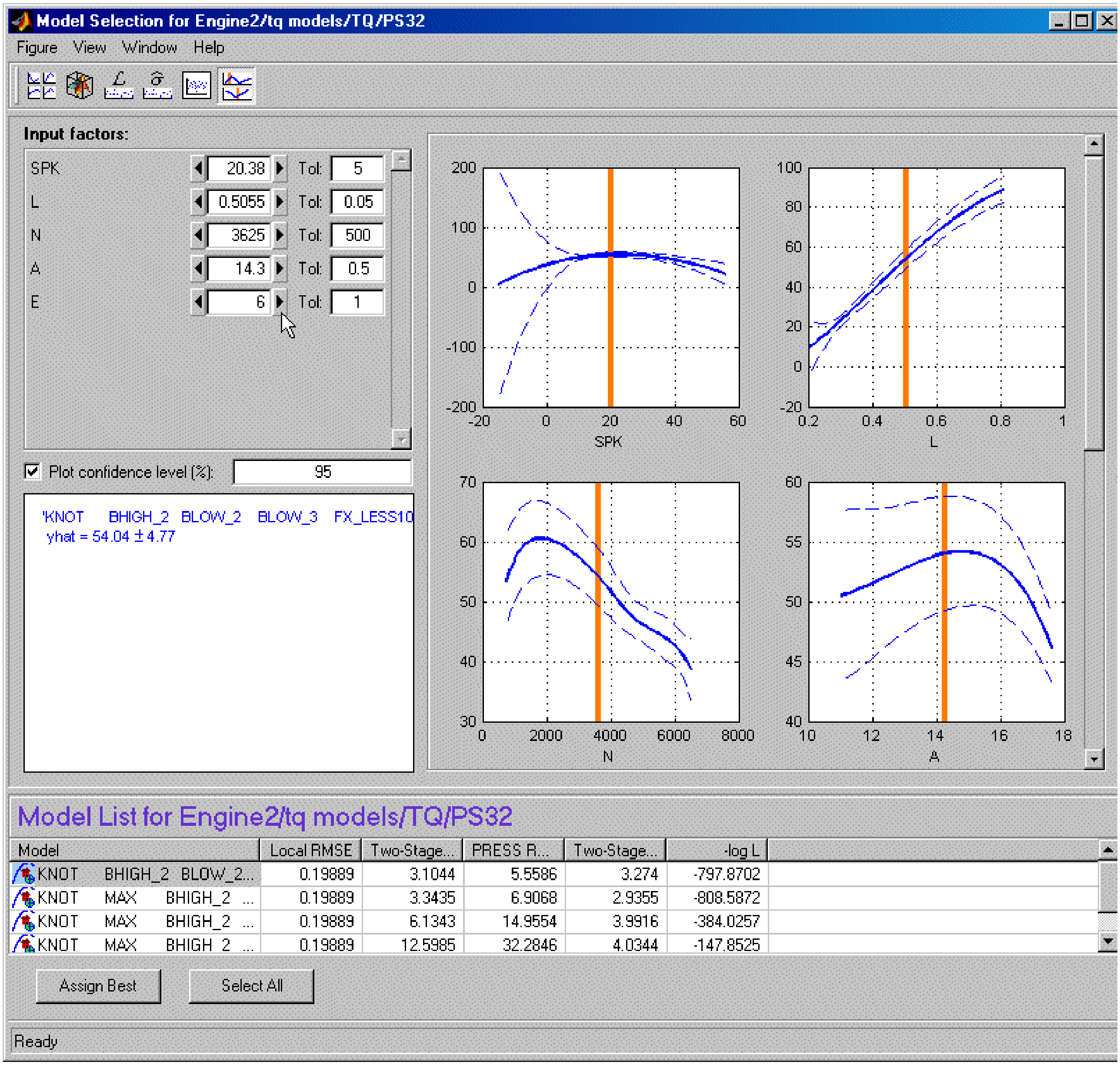
This view allows several models to be displayed simultaneously (as shown in this example); click the **Select All** button at the bottom of the window or, in the model list view, **Shift**+click or **Ctrl**+click to select the models for display.

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.



Model Selection for Engine2/tq models/TQ/PS32

Input factors:

SPK	20.38	Tol:	5
L	0.5055	Tol:	0.05
N	3625	Tol:	500
A	14.3	Tol:	0.5
E	6	Tol:	1

☑ Plot confidence level (%): 95

'KNOT BHIGH_2 BLOW_2 BLOW_3 FX_LESS10
yhat = 54.04 ± 4.77

Model List for Engine2/tq models/TQ/PS32

Model	Local RMSE	Two-Stage...	PRESS R...	Two-Stage...	-log L
KNOT BHIGH_2 BLOW_2...	0.19889	3.1044	5.5586	3.274	-797.8702
KNOT MAX BHIGH_2 ...	0.19889	3.3435	6.9068	2.9355	-808.5872
KNOT MAX BHIGH_2 ...	0.19889	6.1343	14.9554	3.9916	-384.0257
KNOT MAX BHIGH_2 ...	0.19889	12.5985	32.2846	4.0344	-147.8525

Ready

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 (**L**, **N**, **A**) 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. 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.

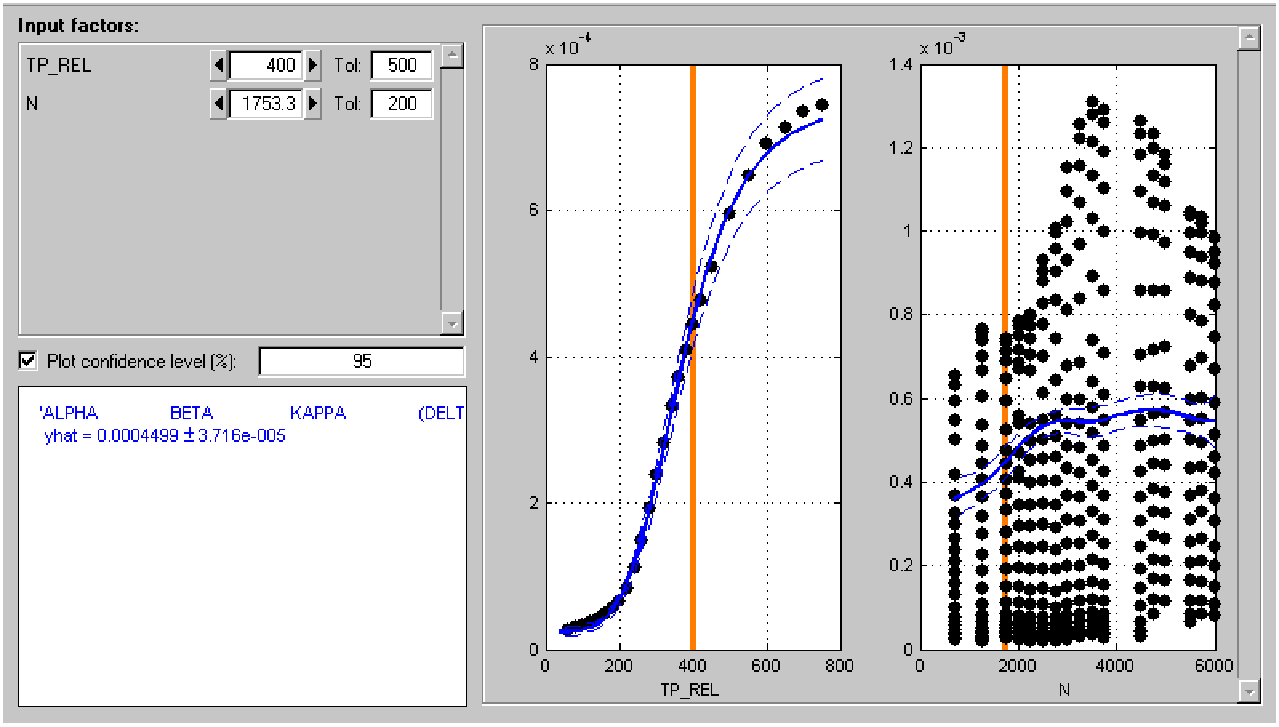
Data points are displayed when they fall within the tolerance limit near each cross section. You can set the tolerance in the **Tol** edit boxes.

- 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.

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.



Input factors:

TP_REL 400 Tol: 500

N 1753.3 Tol: 200

☑ Plot confidence level (%): 95

'ALPHA BETA KAPPA (DELT

yhat = 0.0004499 ± 3.716e-005

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 MLE.

   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.

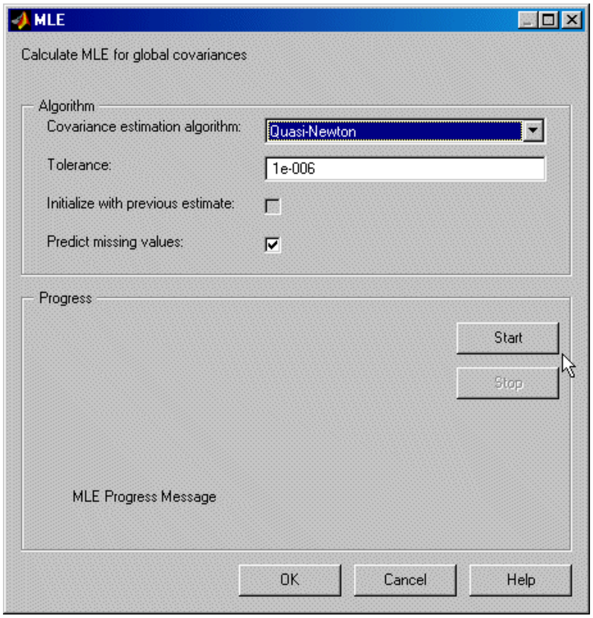
Note After calculating MLE, you will notice that the plots and the icons in the model tree for the whole two-stage model (response node, local node, and all response feature nodes) have turned purple. See "Icons: Blue Backgrounds and Purple Worlds" on page 5-13.

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



MLE

Calculate MLE for global covariances

Algorithm

Covariance estimation algorithm: Quasi-Newton

Tolerance: 1e-006

Initialize with previous estimate: □

Predict missing values: ☑

Progress

MLE Progress Message

Algorithm

The algorithm drop-down menu offers a choice between two covariance estimation algorithms, Quasi-Newton (the default) 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 6-37). 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

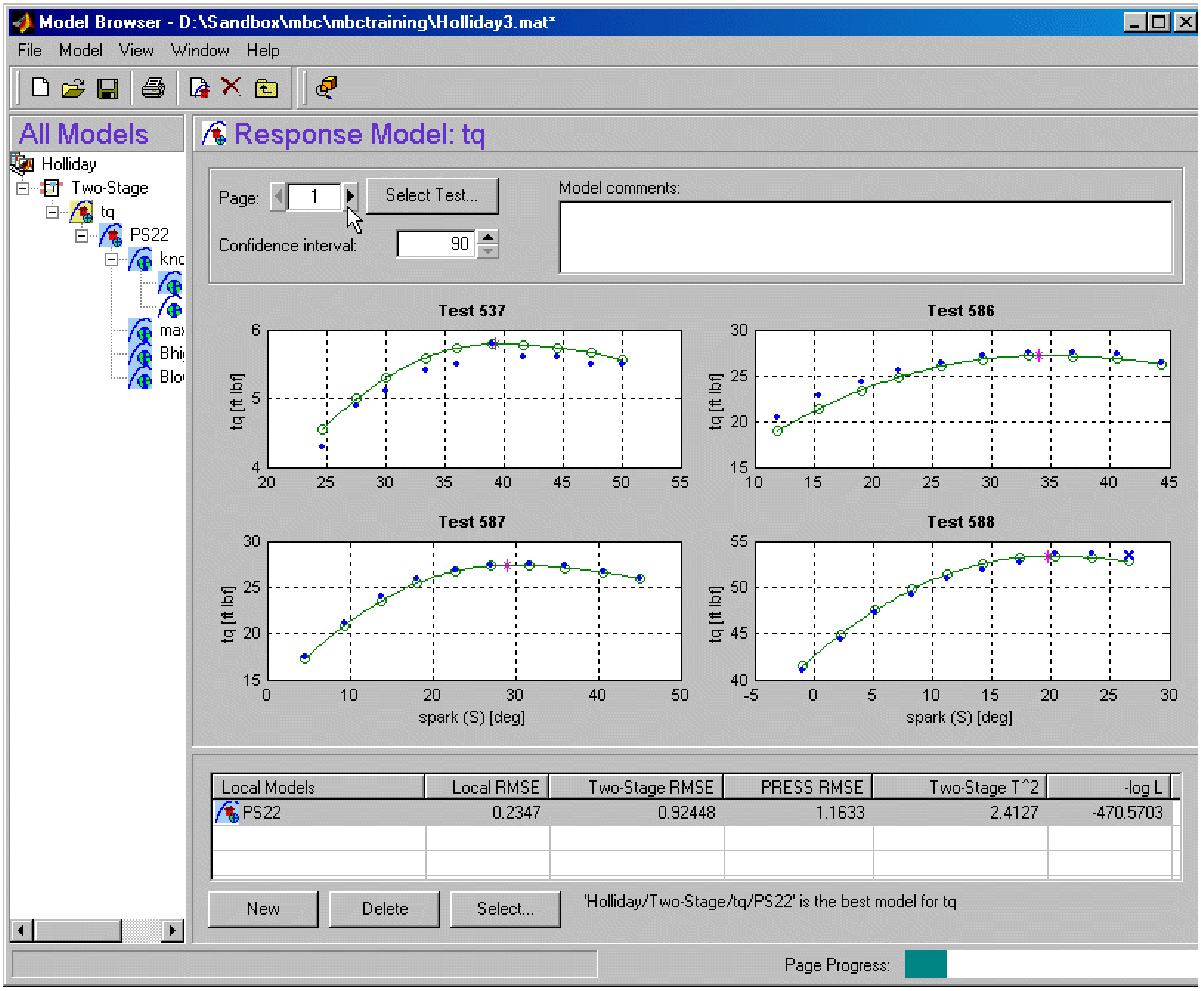
The other check box (selected by default) predicts missing values. When it is selected, response features that are outliers for the univariate global model are replaced by the predicted value. This allows tests to be used for MLE even if one of the response features is missing. If all the response features 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.



Page: 1

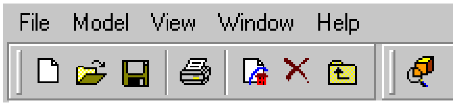
Confidence interval: 90

Local Models	Local RMSE	Two-Stage RMSE	PRESS RMSE	Two-Stage T^2	-log L
PS22	0.2347	0.92448	1.1633	2.4127	-470.5703

'Holliday/Two-Stage/tq/PS22' is the best model for tq

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.

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 5-7 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 5-8.
- **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.

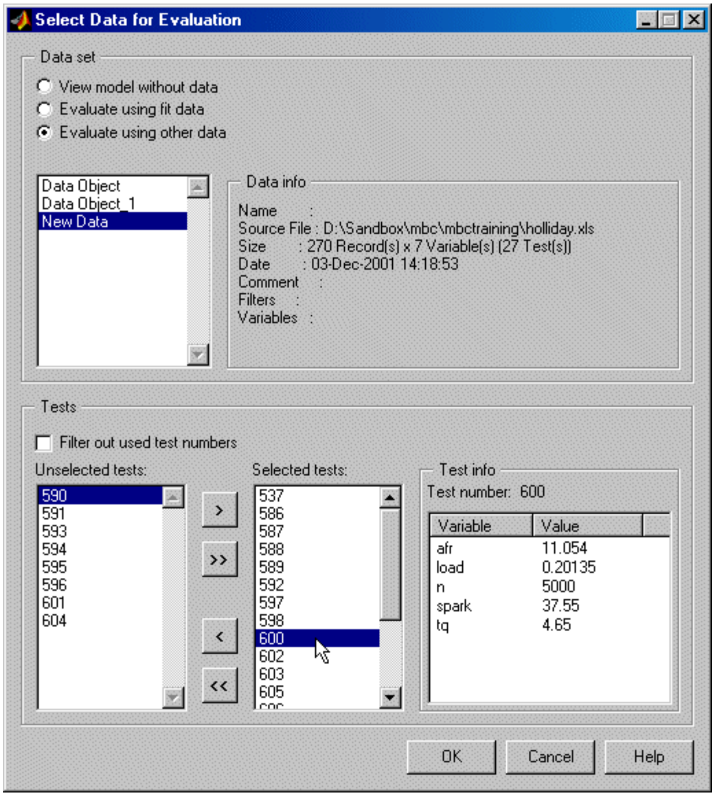
Model Evaluation Window

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.



Data set

○ View model without data
○ Evaluate using fit data
● Evaluate using other data

Data Object
Data Object_1
New Data

Data info

Name :
Source File : D:\Sandbox\mbc\mbctraining\holliday.xls
Size : 270 Record(s) x 7 Variable(s) (27 Test(s))
Date : 03-Dec-2001 14:18:53
Comment :
Filters :
Variables :

Tests

□ Filter out used test numbers

Unselected tests:
590
591
593
594
595
596
601
604

Selected tests:
537
586
587
588
589
592
597
598
600
602
603
605

Test info

Test number: 600

Variable	Value
afr	11.054
load	0.20135
n	5000
spark	37.55
tq	4.65

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.

- **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.

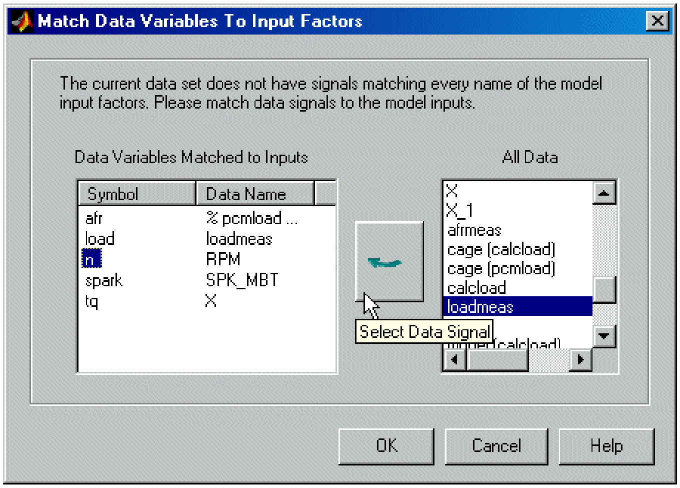
Model Evaluation Using Other Data

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.



The current data set does not have signals matching every name of the model input factors. Please match data signals to the model inputs.

Symbol	Data Name
afr	% pcmload ...
load	loadmeas
n	RPM
spark	SPK_MBT
tq	X

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 5-144, 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.

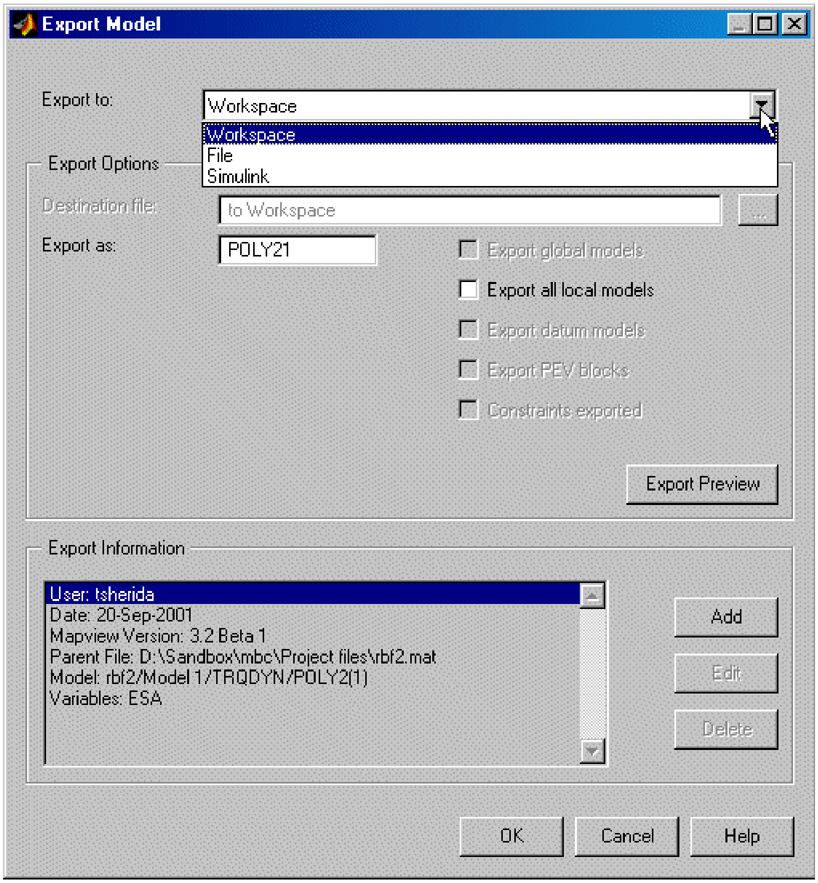
Click **OK** to use the selected tests to evaluate this model. The **Model Evaluation** window appears.

For information about the available views, see “Selecting Models” on page 5-117.

Exporting Models

You can export all models created in MBC using the menu item **File ->Export Models**.

The **Export Model** dialog appears.



Export to: Workspace

Destination file: to Workspace

Export as: POLY21

□ Export global models
□ Export all local models
□ Export datum models
□ Export PEV blocks
□ Constraints exported

User: tsherida
Date: 20-Sep-2001
Mapview Version: 3.2 Beta 1
Parent File: D:\Sandbox\mbc\Project files\rbf2.mat
Model: rbf2/Model 1/TRQDYN/POLY2(1)
Variables: ESA

You can export a model to the MATLAB workspace, to an EXM file for CAGE, or to a Simulink model (an MDL file). 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.

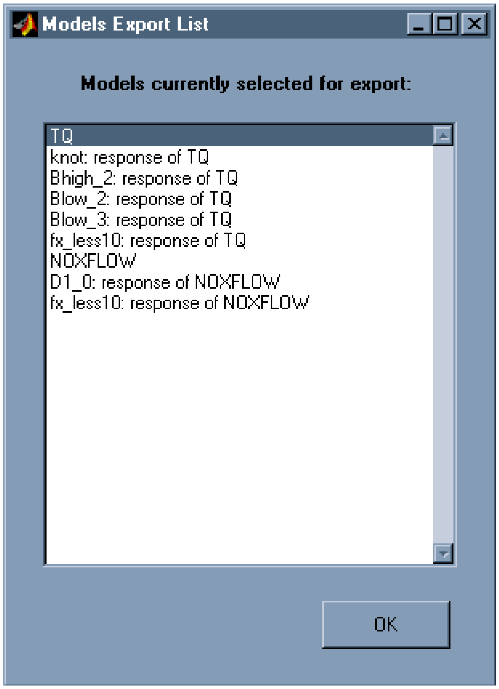
Export all local models — When exporting at the local node, the single local model for the current test is exported. Selecting this control exports the local models for all tests (to the workspace as a cell array).

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).

Constraints exported — Where design constraints exist, they are always exported.

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:



Models currently selected for export:

TQ
knot: response of TQ
Bhigh_2: response of TQ
Blow_2: response of TQ
Blow_3: response of TQ
fx_less10: response of TQ
NOXFLOW
D1_0: response of NOXFLOW
fx_less10: response of NOXFLOW

See “What Is Exported?” on page 5-151 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.

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 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.
- 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 5-148.

Evaluating Models in the Workspace

If a model is exported to the workspace as `MyMod` and has four input factors, it can be evaluated at a point as follows:

```
Y = MyMod([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 = MyMod([p1,p2,p3,p4]);
```

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 "Design of Experiment Tutorial" on page 3-1.

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

"Design Styles" on page 5-154

"Design Editor Displays" on page 5-155

"The Design Tree" on page 5-156

"Display Options" on page 5-156

"Adding Design Points" on page 5-178

"Fixing, Deleting, and Sorting Design Points" on page 5-181

"Saving and Importing Designs" on page 5-183

"Creating a Classical Design" on page 5-159

"Creating a Space Filling Design" on page 5-162

"Creating an Optimal Design" on page 5-168

"Applying Constraints" on page 5-184

You can design experiments at both stages, local and global. 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 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.

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 5-168.

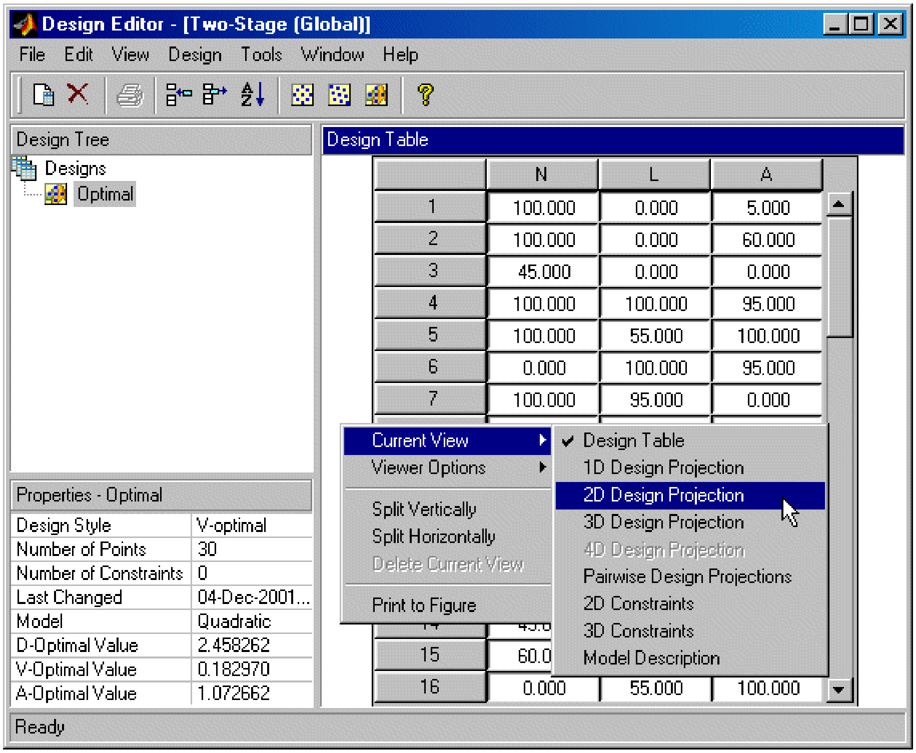
Space-filling designs are better when there is low system knowledge. In cases where you are not sure what type of model is appropriate, and the constraints are uncertain, space-filling designs collect data in such as a way as to maximize coverage of the factors’ ranges as quickly as possible. See “Creating a Space Filling Design” on page 5-162.

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 5-159.

Any design can be augmented 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 5-178.

Design Editor Displays

The following example shows the display after creating an optimal design.



	N	L	A
1	100.000	0.000	5.000
2	100.000	0.000	60.000
3	45.000	0.000	0.000
4	100.000	100.000	95.000
5	100.000	55.000	100.000
6	0.000	100.000	95.000
7	100.000	95.000	0.000
15	60.0		
16	0.000	55.000	100.000

Properties - Optimal

Design Style	V-optimal
Number of Points	30
Number of Constraints	0
Last Changed	04-Dec-2001...
Model	Quadratic
D-Optimal Value	2.458262
V-Optimal Value	0.182970
A-Optimal Value	1.072662

When you first see the main display area, it shows the default **Design Table** view of the design (see example above). There is a context menu, available by right-clicking, in which you can change the view of the design to **1-D**, **2-D**, **3-D**, and **4-D Projections** and **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. 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.

The Design Tree

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

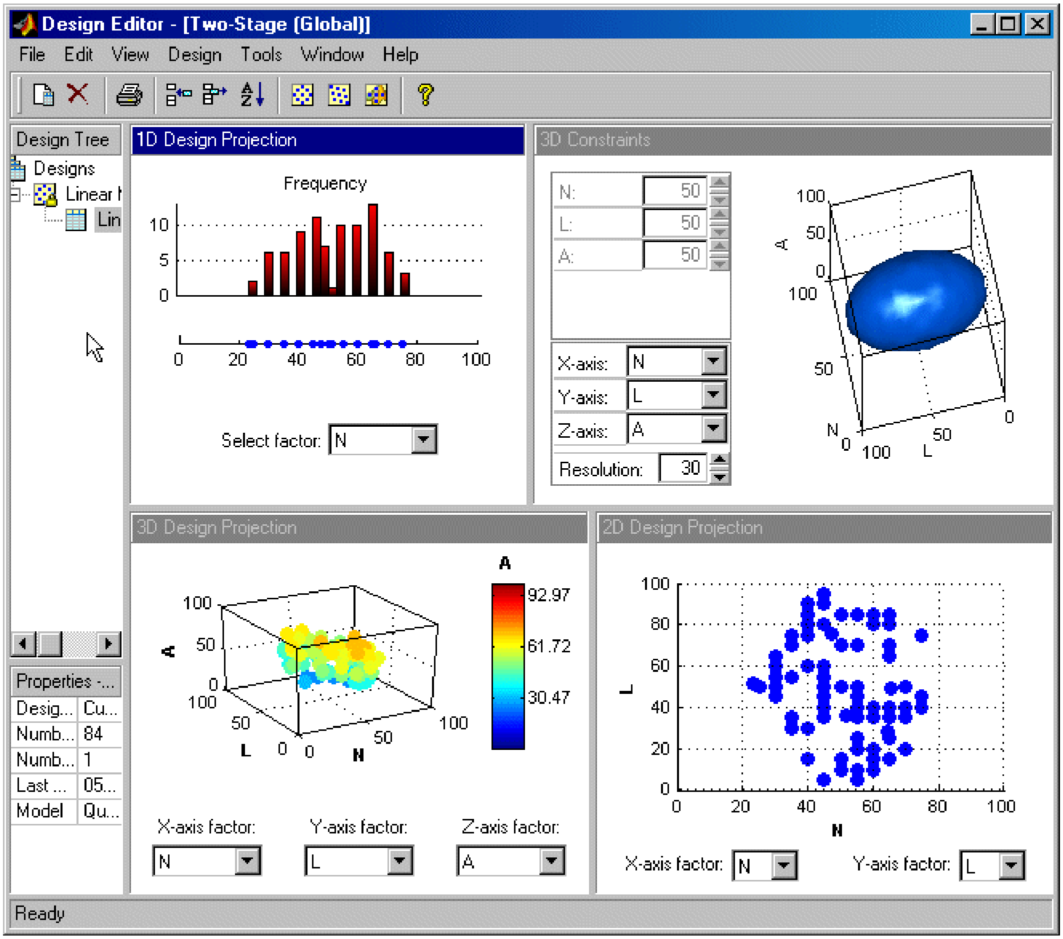
The tree displays three pieces of information:

- 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.

The information pane, bottom left, displays pieces of information for the current design. 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.

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.



Design Editor - [Two-Stage (Global)]

N: 50
L: 50
A: 50

X-axis: N
Y-axis: L
Z-axis: A

Resolution: 30

Select factor: N

X-axis factor: N
Y-axis factor: L
Z-axis factor: A

X-axis factor: N
Y-axis factor: L

Properties -...	
Desig...	Cu...
Numb...	84
Numb...	1
Last ...	05...
Model	Qu...

Ready

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. Also in these menus is the **Print to Figure** command. 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.

The **Viewer Options** item in the **View** menu opens windows for configuring details of the current display. You can change basic properties such as color on all the projections (1-D, 2-D, 3-D, and 4-D). For the table view, you can alter the precision used for displaying data and set up a filter to selectively display certain ranges of values. You can rotate all 3-D views as usual. You can double-click the color bar to edit the colormap.

You can also view projections of constraints; see “Applying Constraints” on page 5-184.

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 of its initial settings from the currently selected design and becomes a child node of that design.

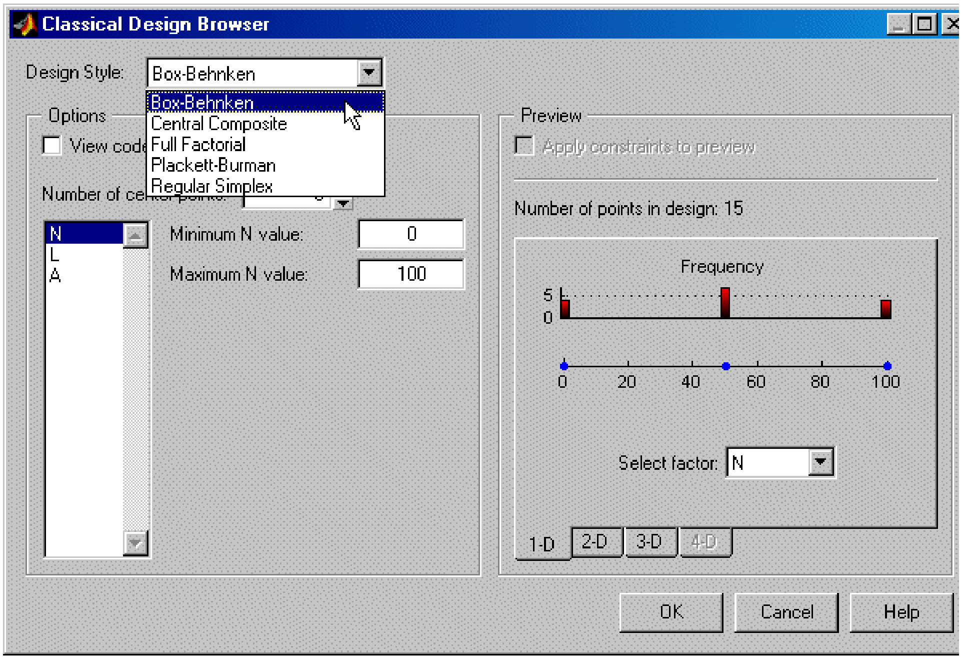
3 Click the button in the toolbar or select **Design –> Classical –> Design Browser**.

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



Classical Design Browser

Design Style: Box-Behnken

Number of points in design: 15

Minimum N value: 0

Maximum N value: 100

Select factor: N

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**.

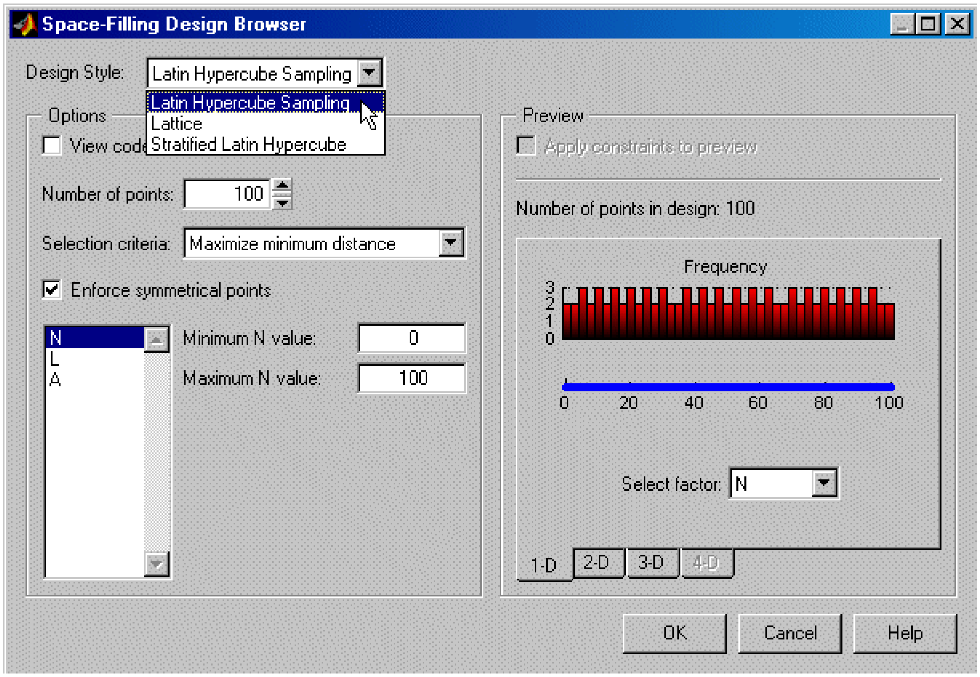
2 Select the node in the tree by clicking. An empty Design Table appears if you have not yet chosen a design. Otherwise, if this is a new child node the display remains the same, because child nodes inherit all the parent design's properties.

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.



Design Style: Latin Hypercube Sampling

□ View code

Number of points: 100

Selection criteria: Maximize minimum distance

☑ Enforce symmetrical points

Minimum N value: 0

Maximum N value: 100

□ Apply constraints to preview

Number of points in design: 100

Select factor: N

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) 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.

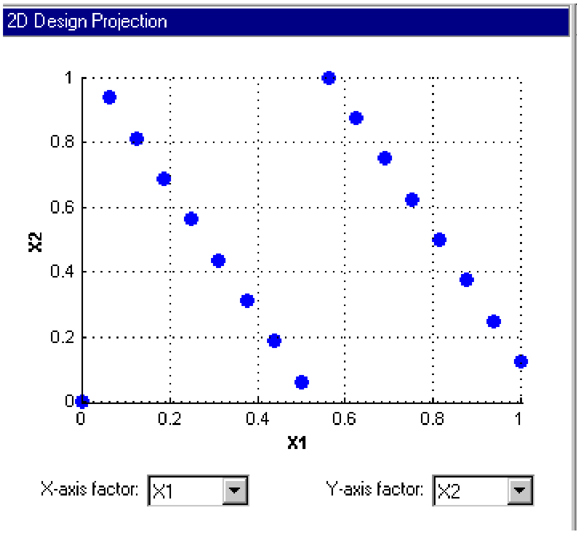
Lattice

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.

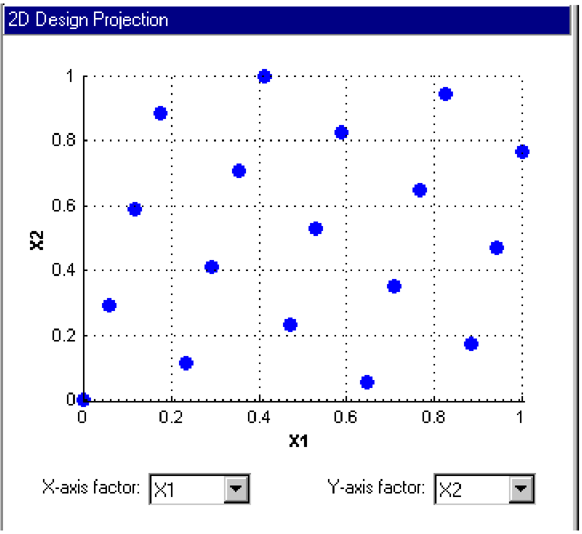
Stratified Latin Hypercube

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.



X-axis factor: X1

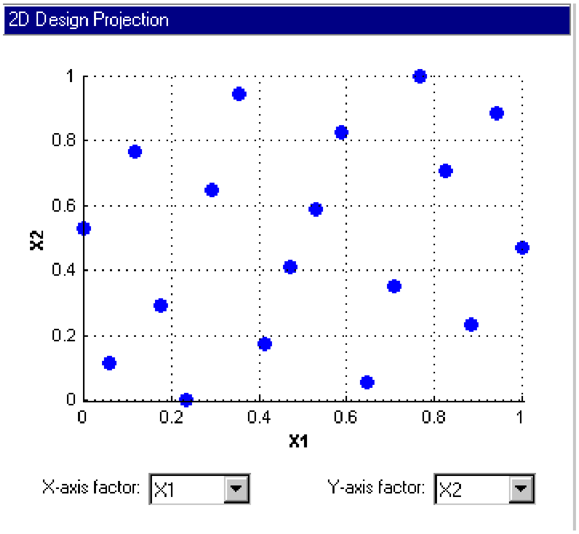
Y-axis factor: X2



X-axis factor: X1

Y-axis factor: X2

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.



X-axis factor: X1

Y-axis factor: X2

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 5-43.

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 5-184.

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.

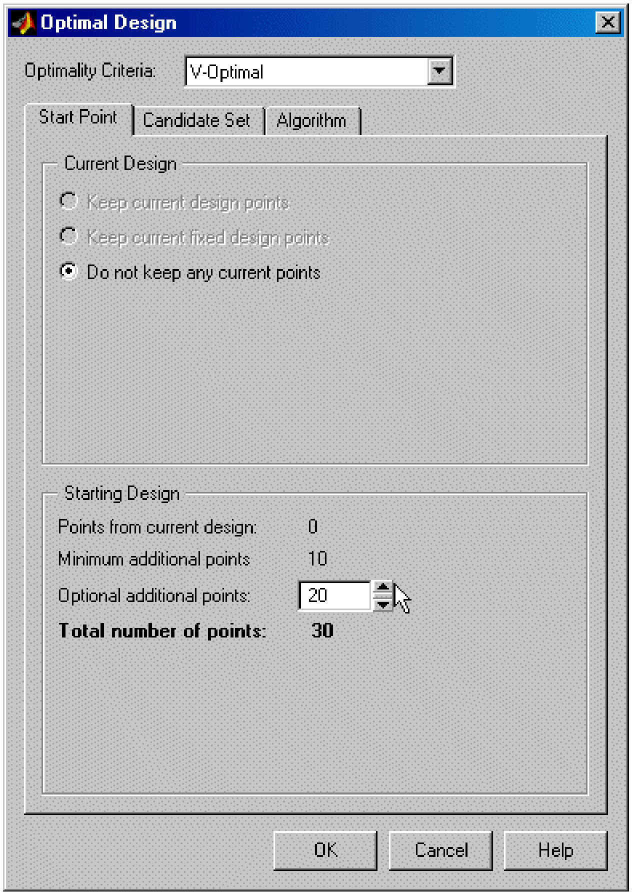
- m additional points are added to the design, either optimally or at random. These points are chosen from the candidate set.
- m points are deleted from the design, either optimally or at random.
- If the resulting design is better than the original, it is kept.

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



Optimality Criteria: V-Optimal

Current Design

○ Keep current design points

○ Keep current fixed design points

● Do not keep any current points

Starting Design

Points from current design: 0

Minimum additional points 10

Optional additional points: 20

Total number of points: 30

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 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

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

  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

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

  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

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

  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 Choose the number of additional points to add by clicking the **Optional Additional Points** up/down buttons or by typing directly into the edit box.

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 5-162.
- **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.

There is an edit box where you can choose the maximum number of points to display. This is a limiter, as candidate sets with several factors soon become very large if each factor has, say, 21 levels, and the computation and display could be very slow.

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**).

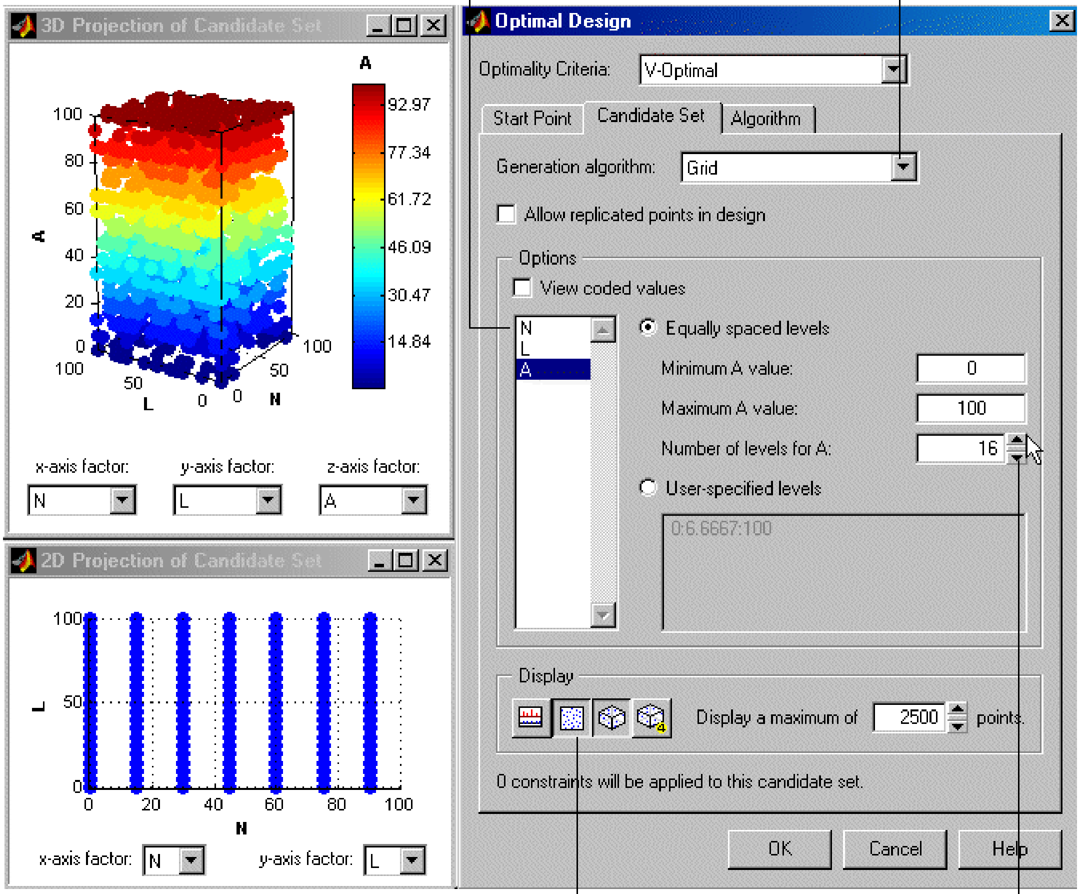
2 This tab also has buttons for creating plots of the candidate sets. Try them to preview your design. If you have added custom points, you can check them here.

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



x-axis factor: N

y-axis factor: L

z-axis factor: A

x-axis factor: N

y-axis factor: L

Optimality Criteria: V-Optimal

Generation algorithm: Grid

□ Allow replicated points in design

□ View coded values

● Equally spaced levels

Minimum A value: 0

Maximum A value: 100

Number of levels for A: 16

○ User-specified levels

0:6.6667:100

Display a maximum of 2500 points.

0 constraints will be applied to this candidate set.

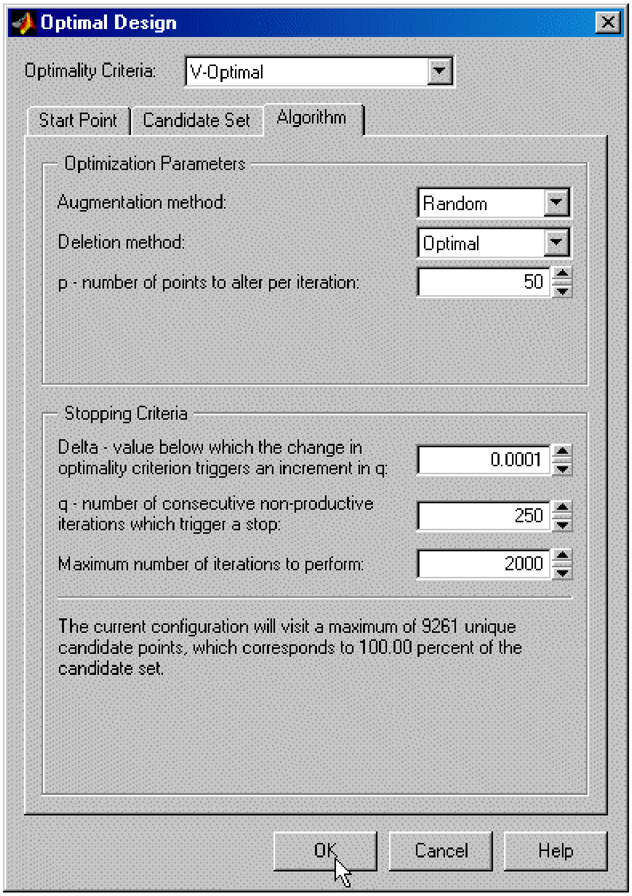
Open display windows with these buttons

Change the number of levels of the selected variable here

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 effect if random augmentation or deletion is chosen.
- **Maximum number of iterations to perform** — Overall maximum number of iterations.



Optimality Criteria: V-Optimal

Augmentation method: Random

Deletion method: Optimal

p - number of points to alter per iteration: 50

Delta - value below which the change in optimality criterion triggers an increment in q: 0.0001

q - number of consecutive non-productive iterations which trigger a stop: 250

Maximum number of iterations to perform: 2000

The current configuration will visit a maximum of 9261 unique candidate points, which corresponds to 100.00 percent of the candidate set.

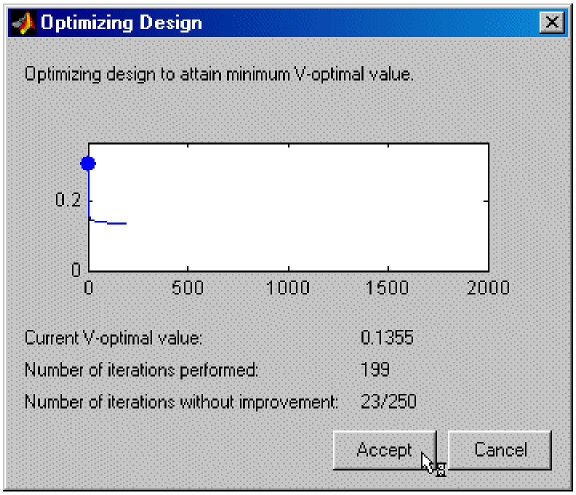
1 Choose the augmentation and deletion methods from the drop-down menus (or leave at the defaults).

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.



Optimizing design to attain minimum V-optimal value.

Current V-optimal value: 0.1355

Number of iterations performed: 199

Number of iterations without improvement: 23/250

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.

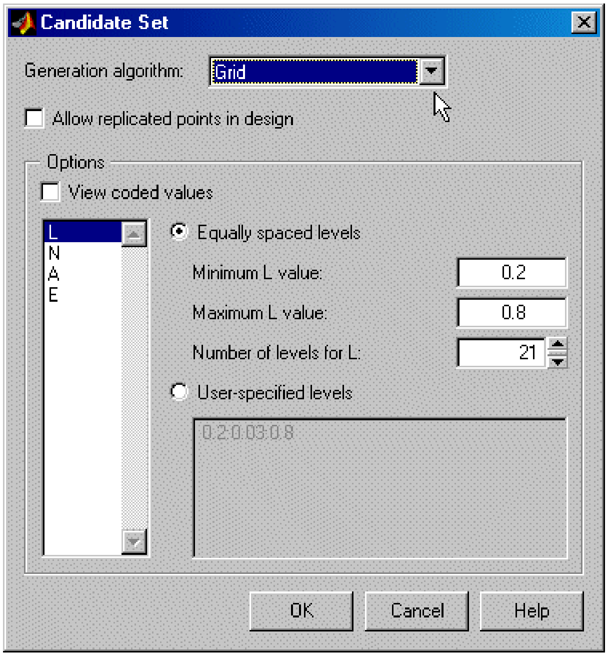
2 Choose an augmentation method from the drop-down menu: optimal (D,V, or A), random, or user-specified.

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.



Candidate Set

Generation algorithm: Grid

□ Allow replicated points in design

Options

□ View coded values

L
N
A
E

● Equally spaced levels

Minimum L value: 0.2

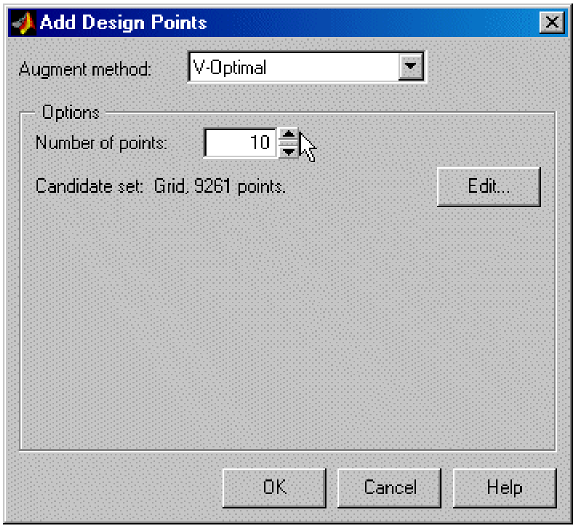
Maximum L value: 0.8

Number of levels for L: 21

○ User-specified levels

0.2:0.03:0.8

5 Click **OK** to add the points and return to the main display.



Add Design Points

Augment method: V-Optimal

Options

Number of points: 10

Candidate set: Grid, 9261 points.

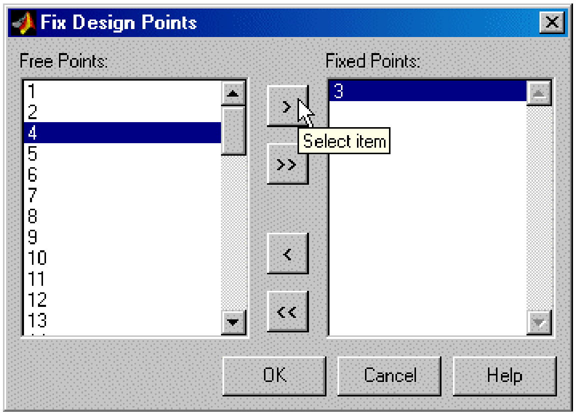
Fixing, Deleting, and Sorting Design Points

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. Once fixed, 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.

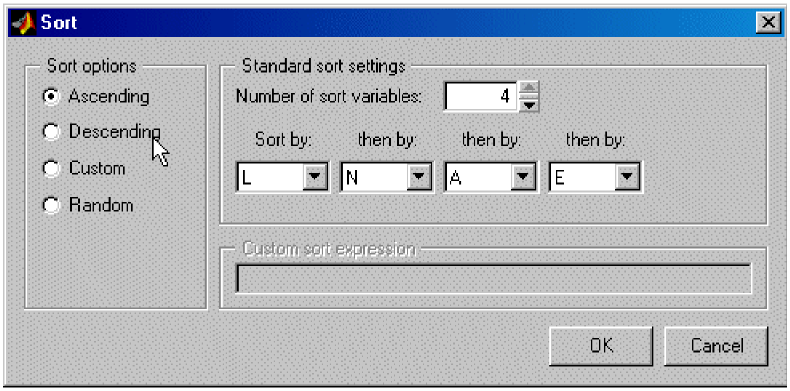
2 In order to see where these points are in the design, you must change the main Design Editor display pane to the **Table** view. This gives a numbered list of every point in the design.

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.



Sort options
● Ascending
○ Descending
○ Custom
○ Random

Standard sort settings
Number of sort variables: 4
Sort by: L
then by: N
then by: A
then by: E

Custom sort 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 centre of your design space will be near zero in coded units, and those near the edge will tend to 1 or -1.

Saving and Importing Designs

You can save 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. and/or 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 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

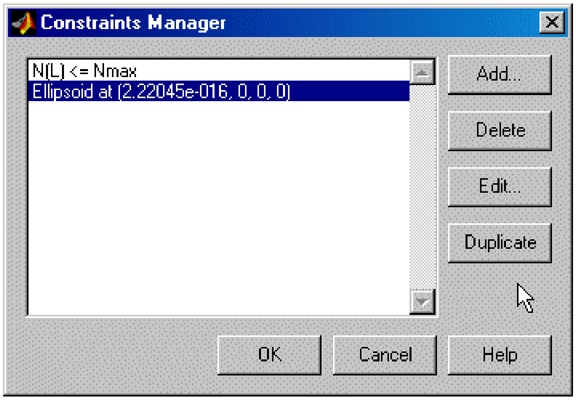
In many cases designs might not coincide with the operating region of the system to be tested. For example, an automobile engine normally does not operate in a region of low speed (n) and high exhaust gas recirculation (EGR). You cannot run 15% EGR at 1000 RPM. There is no point selecting design points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

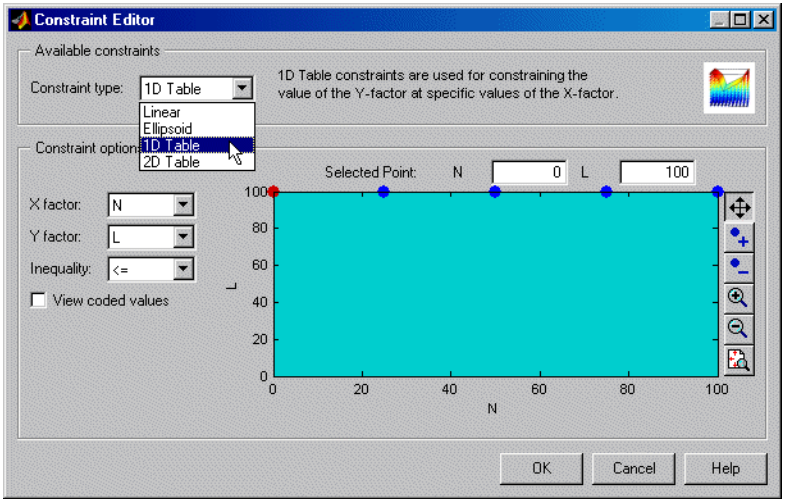
2 The **Constraints Manager** dialog appears.


N(L) <= Nmax
Ellipsoid at (2.22045e-016, 0, 0, 0)

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 Editor

Available constraints

Constraint type: 1D Table

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

Constraint options

Selected Point: N 0 L 100

X factor: N

Y factor: L

Inequality: <=

□ View coded values

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

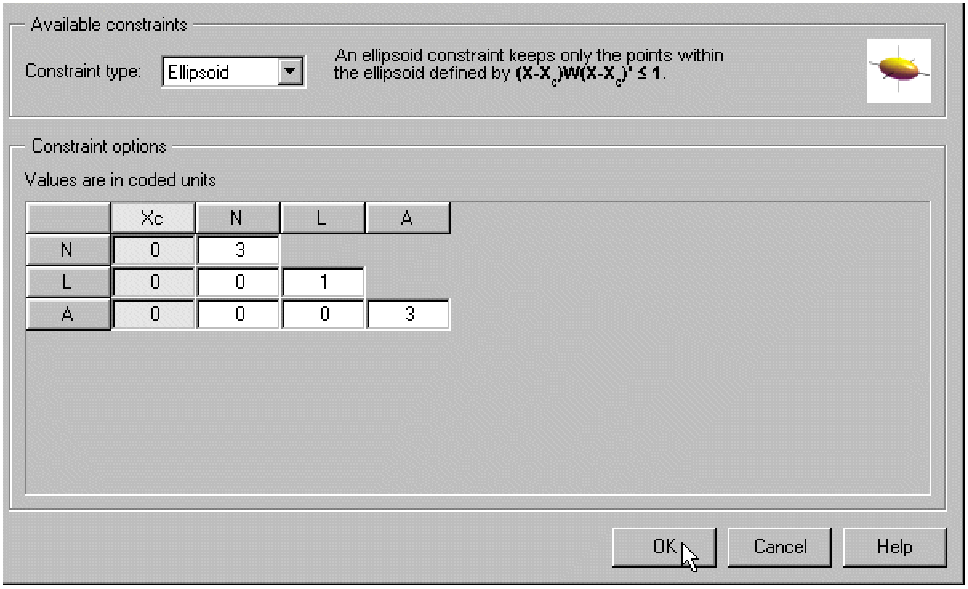
The ellipsoid constraint allows you to define an N-dimensional ellipsoid. You can specify the center of the ellipsoid, the length of each axis, and the rotation of the ellipsoid.

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.



Available constraints

Constraint type: Ellipsoid

An ellipsoid constraint keeps only the points within the ellipsoid defined by **$(X-X_c)W(X-X_c)' \le 1$**.

Constraint options

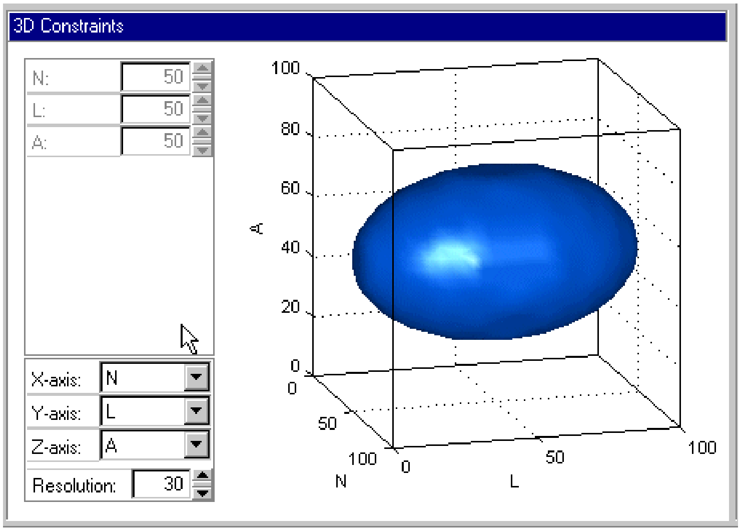
Values are in coded units

	Xc	N	L	A
N	0	3		
L	0	0	1	
A	0	0	0	3

- 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.



3D Constraints

N: 50

L: 50

A: 50

X-axis: N

Y-axis: L

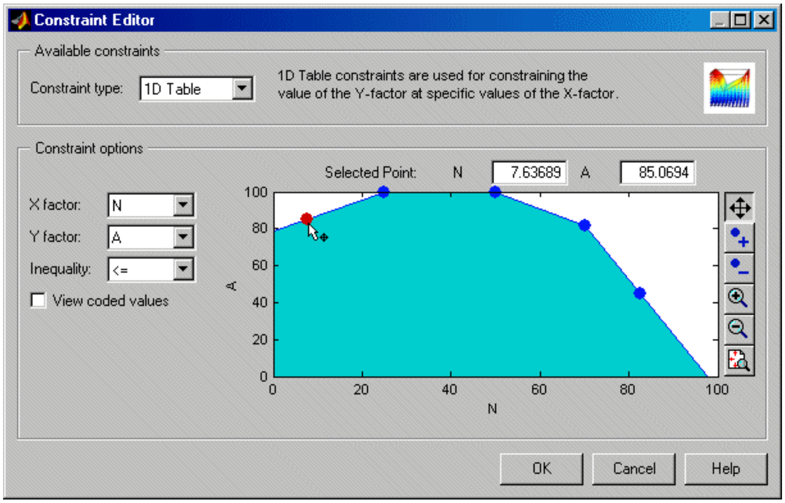
Z-axis: A

Resolution: 30

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.



Constraint type: 1D Table

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

X factor: N

Y factor: A

Inequality: <=

□ View coded values

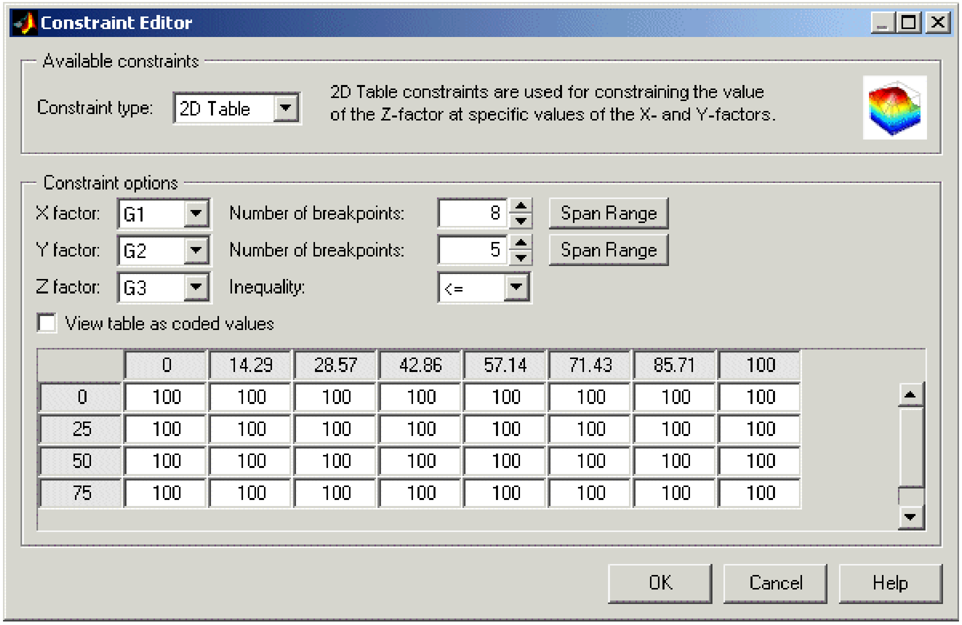
Selected Point: N 7.63689 A 85.0694

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 decribed 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.



Available constraints

Constraint type: 2D Table

2D Table constraints are used for constraining the value of the Z-factor at specific values of the X- and Y-factors.

Constraint options

X factor: G1 Number of breakpoints: 8

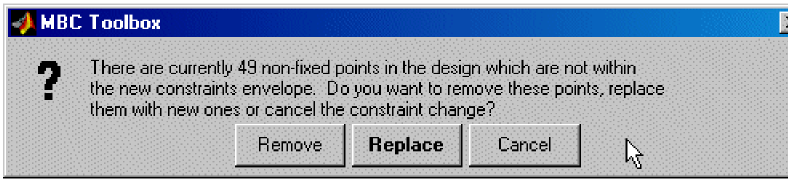
Y factor: G2 Number of breakpoints: 5

Z factor: G3 Inequality: <=

□ View table as coded values

	0	14.29	28.57	42.86	57.14	71.43	85.71	100
0	100	100	100	100	100	100	100	100
25	100	100	100	100	100	100	100	100
50	100	100	100	100	100	100	100	100
75	100	100	100	100	100	100	100	100

- 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.



There are currently 49 non-fixed points in the design which are not within the new constraints envelope. Do you want to remove these points, replace them with new ones or cancel the constraint change?

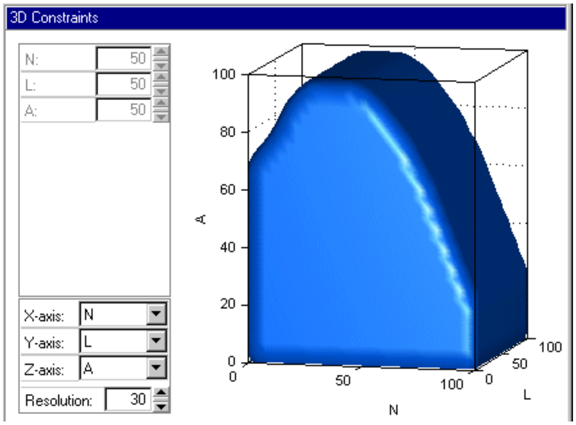
A dialog appears if there are points in the design that fall outside your newly constrained candidate set. You can simply continue (delete them) or cancel the constraint. Fixed points are *not* removed by this process. For optimal designs you can also replace them with new random points within the new 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 5-178.

To view constraints:

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

2 Select **Current View –> 3D Constraints**. An example is shown.

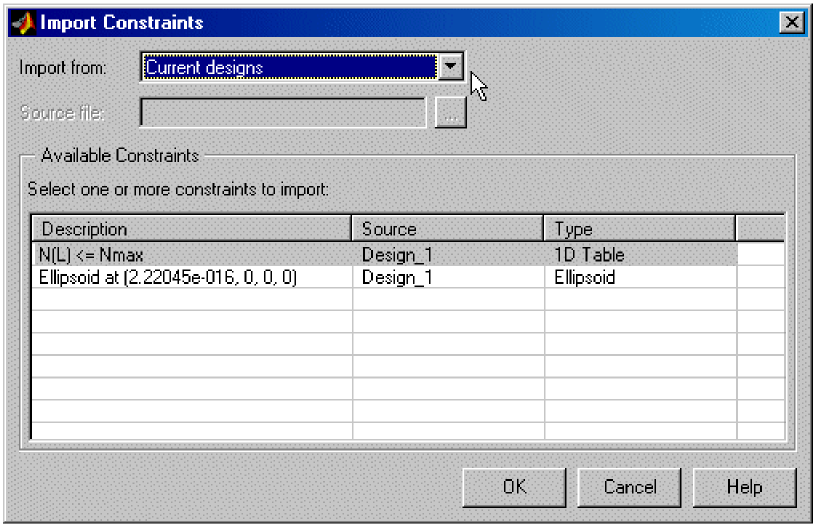


N: 50
L: 50
A: 50
X-axis: N
Y-axis: L
Z-axis: A
Resolution: 30

These views are intended to give some idea of the region of space that is currently available within the constraint boundaries.

Importing Constraints

Select **File –> Import Constraints**. The **Import Constraints** dialog appears, as shown in the following example.



Import Constraints

Import from: Current designs

Source file:

Available Constraints

Select one or more constraints to import:

Description	Source	Type
N(L) <= Nmax	Design_1	1D Table
Ellipsoid at (2.22045e-016, 0, 0, 0)	Design_1	Ellipsoid

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 file name 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.

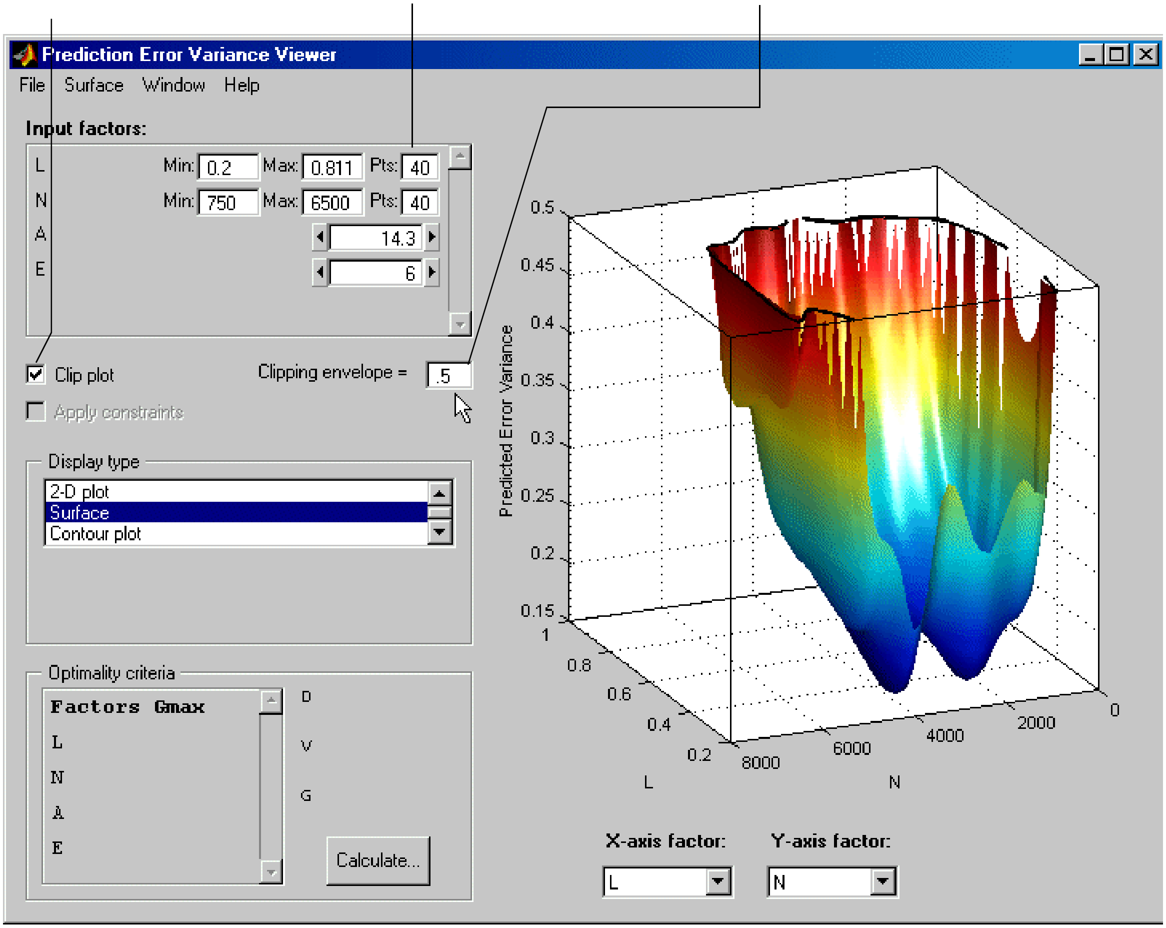
When designs are rank deficient the PEV Viewer appears but is empty; that is, the PEV values cannot be evaluated because there are not enough points to fit the model.

- 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:
  - You can click the toolbar button to open the PEV viewer.
  - Alternatively, you can select **Model –> Utilities –> Prediction Error Surface**.

Turn clipping on and off here

Change number of points plotted here

Change clipping value here



Prediction Error Variance Viewer

Input factors:

L Min: 0.2 Max: 0.811 Pts: 40

N Min: 750 Max: 6500 Pts: 40

A 14.3

E 6

☑ Clip plot

Clipping envelope = .5

□ Apply constraints

Display type: Surface

X-axis factor: L

Y-axis factor: N

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 **Surface** 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 the Response Surface view of model selection and evaluation) 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. The edges of the clip are very jagged; you can make it smoother by increasing the numbers of points plotted for each factor (enter values in the **Pts** edit boxes in the top frame).

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 5-124.

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 6-7 and "Prediction Error Variance for Two-Stage Models" on page 6-43 in the Technical Documents section.

Technical Documents

Linear Regression 6-3

Definitions 6-5

Prediction Error Variance 6-8

High-Level Model Building Process Overview 6-10

Stepwise Regression Techniques 6-14

Box-Cox Transformation 6-19

Linear Model Statistics Displays 6-23

Design Evaluation Tool 6-29

Maximum Likelihood Estimation 6-38

Two-Stage Models for Engines 6-39

Local Model Definitions 6-50

Neural Networks 6-60

User-Defined Models 6-61

Transient Models 6-67

Data Loading Application Programming Interface . . 6-76

The technical documents are divided into the following sections:

- “Linear Regression” on page 6-3
- “Definitions” on page 6-5
- “High-Level Model Building Process Overview” on page 6-9
- “Stepwise Regression Techniques” on page 6-13
- “Box-Cox Transformation” on page 6-18
- “Linear Model Statistics Displays” on page 6-22
- “Design Evaluation Tool” on page 6-27
- “Maximum Likelihood Estimation” on page 6-36
- “Two-Stage Models for Engines” on page 6-37
- “Local Model Definitions” on page 6-48
- “Neural Networks” on page 6-57
- “User-Defined Models” on page 6-58
- “Transient Models” on page 6-64
- “Data Loading Application Programming Interface” on page 6-73

Linear Regression

This introduction to linear regression in the Model-Based Calibration Toolbox is divided into the following sections:

- "Stepwise Regression" on page 6-3
- "Definitions" on page 6-5
- "Prediction Error Variance" on page 6-7

Stepwise 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 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 excluded from the model
X_r	(Nxr) matrix corresponding to terms currently excluded from the model
β_p	(px1) vector of model coefficients $\beta_p = \{\beta_1, \beta_2, \ldots, \beta_p\}$ ""$\hat{\beta} = (X^T X)^{-1} X^T y$ $\mathrm{var}\hat{\beta} = (X^T X)^{-1}\mathrm{MSE}$
PEV	Prediction Error Variance $\mathrm{PEV}(x) = \mathrm{var}y(\hat{y}) = x(X^T X)^{-1} x^T \mathrm{MSE}$
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. $L = \{l_1, l_2, \ldots, l_N\}' = \{H_{11}, H_{22}, \ldots, H_{NN}\}$
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
$\mathrm{MSE}_{(i)}$	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. $RMSE = \sqrt{MSE}$
$\mathrm{s_i}$	ith R-Student or Externally Scaled Studentized Residual. $s_i = \frac{e_i}{\sqrt{MSE_{(i)}(1-H_{ii})}}$
$\mathrm{r_i}$	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. $SEBETA = MSE\{\sqrt{c_{11}}, \sqrt{c_{22}}, \ldots, \sqrt{c_{pp}}\}'$ where $c = (\mathrm{X^T X})^{-1}$
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 6-22.

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 start with the regression (or design) matrix, for example:

$$X = \begin{bmatrix} 1 & L_1 & N_1 & L^2{}_1 & L_1N_1 & N^2{}_1 \\ 1 & L_2 & N_2 & L^2{}_2 & L_2N_2 & N^2{}_2 \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ 1 & L_n & N_n & L^2{}_n & L_nN_n & N^2{}_n \end{bmatrix}$$

If you knew the actual model, you would know the actual model coefficients β. In this case the observations would be

$$y = X\beta + \varepsilon$$

where ε is the measurement error with variance

$$\mathrm{var}(\varepsilon) = (X^{\mathrm{T}}X)^{-1}\mathrm{MSE}$$

However you can only ever know the predicted coefficients:

$$\hat{\beta} = (X^{\mathrm{T}}X)^{-1}X^{\mathrm{T}}y$$

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:

$$x = \begin{bmatrix} 1 & L_{new} & N_{new} & L^2{}_{new} & L_{new}N_{new} & N^2{}_{new} \end{bmatrix}$$

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:

$$\mathrm{PEV}(x) = \mathrm{var}(\hat{y}) = (x(X^{\mathrm{T}}X)^{-1}X^{\mathrm{T}})(\mathrm{X}(X^{\mathrm{T}}X)^{-1}x^{\mathrm{T}})\mathrm{MSE}$$

$$\mathrm{PEV}(x) = x(X^{\mathrm{T}}X)^{-1}x^{\mathrm{T}}\mathrm{MSE}$$

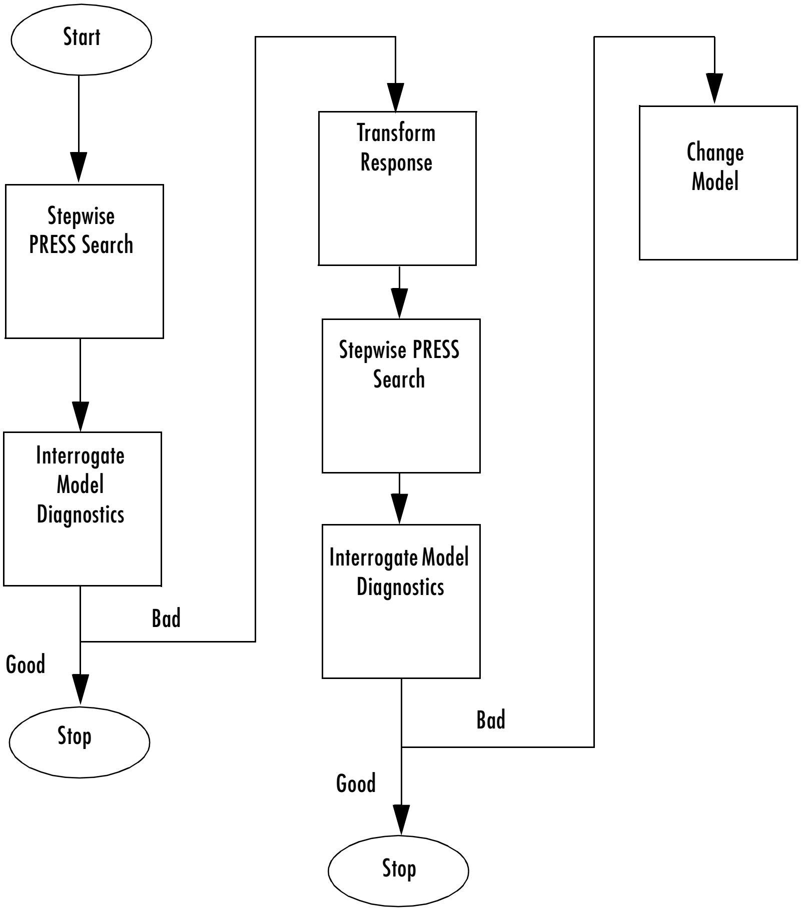
Note the only dependence on the observed values is in the variance (MSE) of the measurement error. You can look at the PEV(x) for a design (without MSE, as you don’t yet have any observations) and see what effect it will have on the measurement error - if it is greater than 1 it will magnify the error, 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 5-193.

For information on the calculation of PEV for two-stage models, see “Prediction Error Variance for Two-Stage Models” on page 6-43.

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

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 criteria 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 are 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 analyser 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 5-144.

Stepwise Regression Techniques

You can open the stepwise regression window through the toolbar icon, when 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.

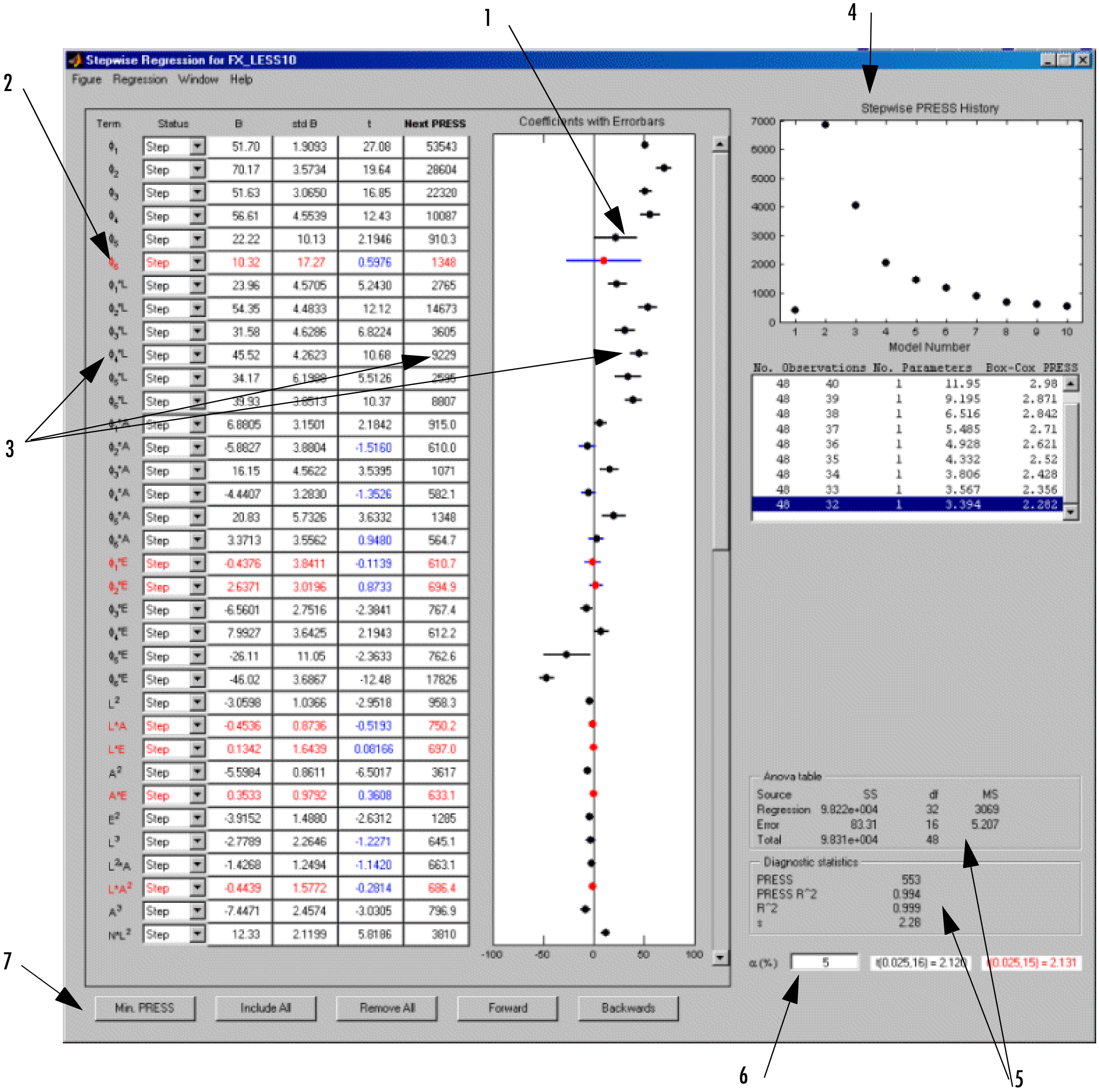
Minimizing Predicted Error Sum of Squares (PRESS) is a good method for working toward a regression model that provides good predictive capability over the experimental factor space.

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 “Linear Model Statistics Displays” on page 6-22.

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.
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.



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
ϕ_1*L	Step	23.96	4.5705	5.2430	2765
ϕ_2*L	Step	54.35	4.4833	12.12	14673
ϕ_3*L	Step	31.58	4.6286	6.8224	3605
ϕ_4*L	Step	45.52	4.2623	10.68	9229
ϕ_5*L	Step	34.17	6.1988	5.5126	2595
ϕ_6*L	Step	39.93	3.8513	10.37	8807
ϕ_1*A	Step	6.8805	3.1501	2.1842	915.0
ϕ_2*A	Step	-5.8827	3.8804	-1.5160	610.0
ϕ_3*A	Step	16.15	4.5622	3.5395	1071
ϕ_4*A	Step	-4.4407	3.2830	-1.3526	582.1
ϕ_5*A	Step	20.83	5.7326	3.6332	1348
ϕ_6*A	Step	3.3713	3.5562	0.9480	564.7
ϕ_1*E	Step	-0.4376	3.8411	-0.1139	610.7
ϕ_2*E	Step	2.6371	3.0196	0.8733	694.9
ϕ_3*E	Step	-6.5601	2.7516	-2.3841	767.4
ϕ_4*E	Step	7.9927	3.6425	2.1943	612.2
ϕ_5*E	Step	-26.11	11.05	-2.3633	762.6
ϕ_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

```
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

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 $\alpha\%$ 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 $\alpha\%$ 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

$$y^{(\lambda)} = \frac{y^{\lambda}-1}{\lambda\dot{y}^{\lambda-1}} \text{ for } \lambda \neq 0\,\lambda$$

$$y^{(\lambda)} = \dot{y}\ln(y) \text{ for } \lambda = 0$$

where the so called geometric mean of the observations is given by

$$\dot{y} = \exp\left[\frac{\sum_{i=1}^{N} \ln(y_i)}{N}\right]$$

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

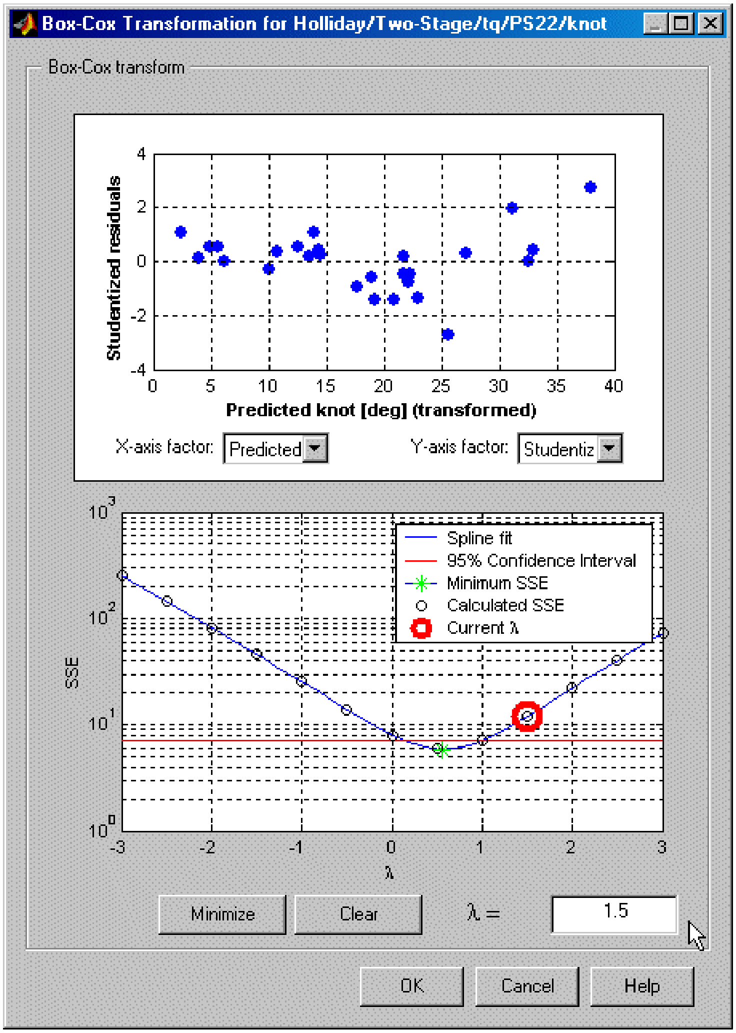
$$SS^{*} = SSE(\lambda)\left[1 + \frac{t_{\alpha/(2,\upsilon)}}{\upsilon}\right]$$

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

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.

ANOVA Table

	SS	**df**	**MS**
Regression	SSR	p-1	SSR/(p-1)
Error	SSE	n-p	SSE/(N-p)
Total	SST	n-1	

Diagnostic Statistics

PRESS	Predicted Error Sum of Squares.
PRESS R^2	R^2 value calculated using PRESS = (SST-PRESS)/SST. Note that this figure is sometimes negative.
R^2	R^2= (SST-SSE)/SST.
s	Root Mean Squares Error for regression = sqrt(SSE/(N-p)).

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 can 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 does 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 6-13 and “Definitions” on page 6-5.

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

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.

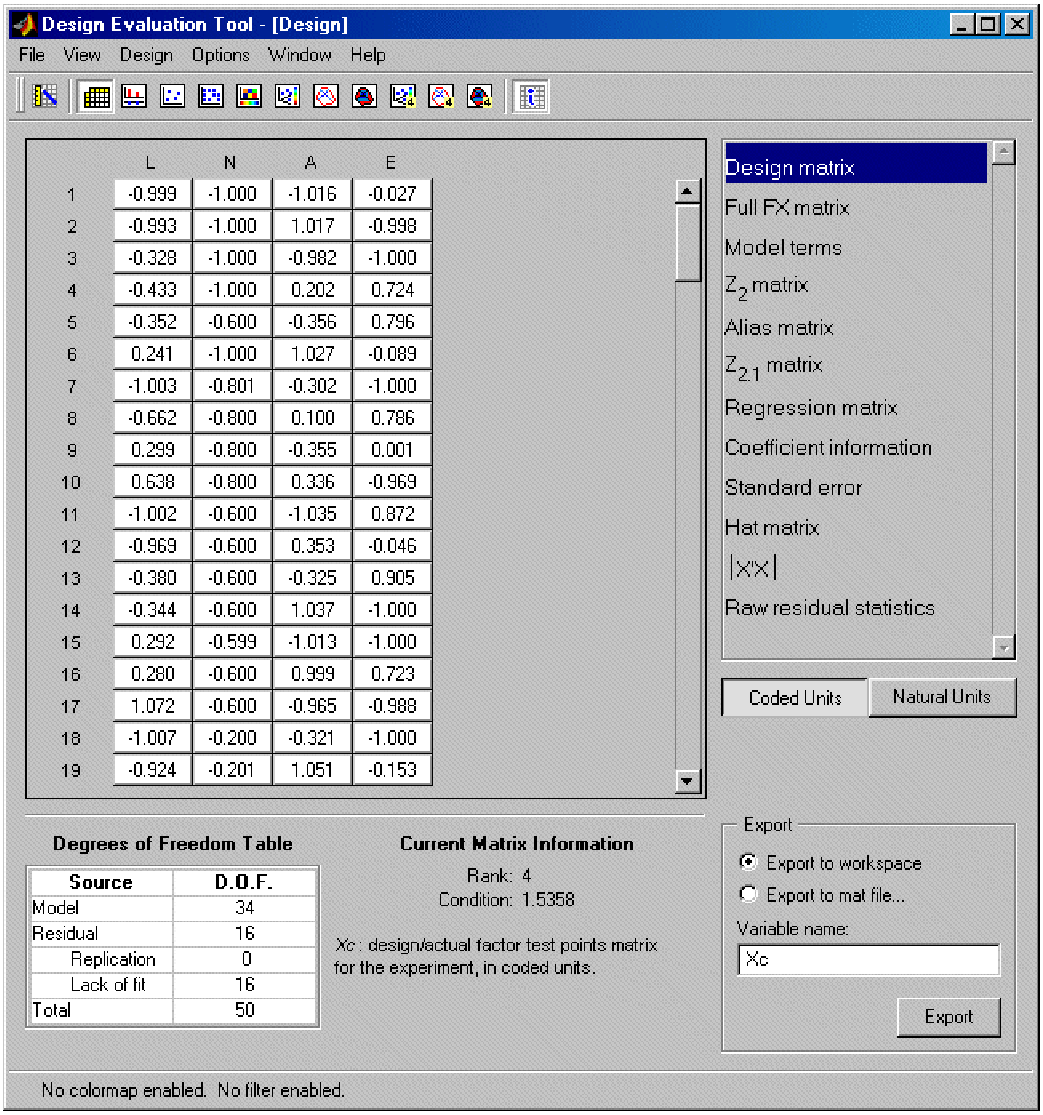
PRESS RMSE	Root mean squared error of predicted errors, see “PRESS Statistic” on page 6-23 above. The divisor used for PRESS RMSE is the number of observations. Not displayed for MLE models because the simple univariate formula cannot be used.
Two-Stage T^2	T^2 is a normalized sum of squared errors for all the response features models. You can see the basic formula on the Likelihood view of the Model Selection window. $\mathrm{T}^2 = (y_{rf} - \hat{y})^{\mathrm{T}} \Sigma^{-1} (y_{rf} - \hat{y})$ Where $\Sigma = \mathrm{blockdiag}(\mathrm{C}_i + \mathrm{D})$, where $\mathrm{C_i}$ 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 $x_1, x_2, .. x_\mathrm{n}$, with probability distribution $f(x, \theta)$, the likelihood is $L = \prod_{i=1} n \quad f(x_i, \theta)$ This is the basis of “Maximum Likelihood Estimation” on page 6-36. $\log \mathrm{L} = \log(\det(\Sigma)) + (y_{rf} - \hat{y})^{\mathrm{T}} \Sigma^{-1} (y_{rf} - \hat{y})$ which is the same as $\log \mathrm{L} = \log(\det(\Sigma)) + \mathrm{T}^2$ This assumes a normal distribution. You can view plots of -log L in the Model Selection window, see “Likelihood View” on page 5-127.

To explain blockdiag as it appears under T^2 in the Pooled statistics table: $\Sigma = \mathrm{blockdiag}(C_i + D)$, where C_i is the local covariance for test *i,* is calculated as shown below.

$$blockdiag(C_i + D) = \begin{bmatrix} C_1 + D & & & \\ & C_2 + D & & \\ & & C_3 + D & \\ & & & C_i + D \end{bmatrix}$$

Design Evaluation Tool

The Design Evaluation tool is only available for linear models.



	L	N	A	E
1	-0.999	-1.000	-1.016	-0.027
2	-0.993	-1.000	1.017	-0.998
3	-0.328	-1.000	-0.982	-1.000
4	-0.433	-1.000	0.202	0.724
5	-0.352	-0.600	-0.356	0.796
6	0.241	-1.000	1.027	-0.089
7	-1.003	-0.801	-0.302	-1.000
8	-0.662	-0.800	0.100	0.786
9	0.299	-0.800	-0.355	0.001
10	0.638	-0.800	0.336	-0.969
11	-1.002	-0.600	-1.035	0.872
12	-0.969	-0.600	0.353	-0.046
13	-0.380	-0.600	-0.325	0.905
14	-0.344	-0.600	1.037	-1.000
15	0.292	-0.599	-1.013	-1.000
16	0.280	-0.600	0.999	0.723
17	1.072	-0.600	-0.965	-0.988
18	-1.007	-0.200	-0.321	-1.000
19	-0.924	-0.201	1.051	-0.153

Degrees of Freedom Table

Source	D.O.F.
Model	34
Residual	16
Replication	0
Lack of fit	16
Total	50

Current Matrix Information

Rank: 4
Condition: 1.5358

Xc : design/actual factor test points matrix for the experiment, in coded units.

Export

● Export to workspace
○ Export to mat file...

Variable name: Xc

No colormap enabled. No filter enabled.

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 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 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

$$VIF_{ij} = \frac{1}{(1-Corr(b)_{ij}^{2})} \text{ for } p \ge i > j > 1$$

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 Statistics

Covariance

Cov(e): variance-covariance matrix for the residuals.

```
Cov(e) = (I-H)
```

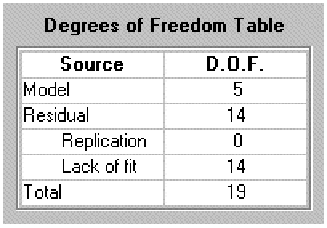
Correlation

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 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:

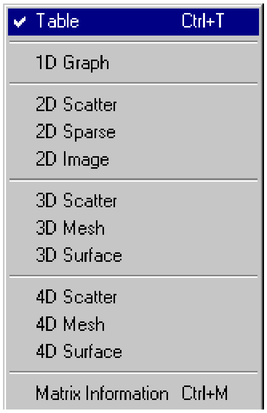
Let there be n_j (>1) replications at the j^{th} replicated point. Then the degrees of freedom for replication are

$$\sum_j (n_j - 1)$$

and **Lack of fit** is given by n - p - degrees of freedom for replication.

Note that 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 6-37
- “Local Models” on page 6-38
  - “Local Covariance Modeling” on page 6-38
  - “Response Features” on page 6-40
- “Global Models” on page 6-40
- “Two-Stage Models” on page 6-41
  - “Prediction Error Variance for Two-Stage Models” on page 6-43
  - “Global Model Selection” on page 6-45
  - “Initial Values for Covariances” on page 6-45
  - “Quasi-Newton Algorithm” on page 6-45
  - “Expectation Maximization Algorithm” on page 6-45
- “References” on page 6-46

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,

$$y_i^j = f_i(s_i^j, \theta_i) + \varepsilon_i^j \textbf{ for } j = 1, 2, \ldots m_i \tag{6-1}$$

where the subscript i refers to individual tests and j to data within a test, s_i^j is the j^{th} independent value, θ_i is a (rx1) parameter vector, y_i^j is the j^{th} response, and ε_i^j is a normally distributed random variable with zero mean and variance σ^2. Note that Equation 6-1 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 i^{th} 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,

$$e_i \sim N(0, \sigma^2 C_i(\beta_i, \xi_i)) \tag{6-2}$$

where C_i is an $(n_i \text{ x } n_i)$ 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

$$C_i = diag\left\{f(x_i, \beta_i)^{\xi_1}\right\} \tag{6-3}$$

- Exponential Variance Model

$$C_i = diag\{\exp(f(x_i, \beta_i)\xi_1)\} \tag{6-4}$$

- Mixed Variance Model

$$C_i = diag\left\{\xi_1 + f(x_i, \theta_i)^{\xi_2}\right\} \tag{6-5}$$

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

$$e_j = \phi_1 e_{j-1} + \phi_2 e_{j-2} + \ldots + \phi_m e_{j-m} + v_j \tag{6-6}$$

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

$$e_j = \phi_1 v_{j-1} + \phi_2 v_{j-2} + \ldots + \phi_m v_{j-m} + v_j \tag{6-7}$$

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

$$p_i = g(\theta_i) \tag{6-8}$$

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

$$p_i = X_i\beta + \gamma_i \textbf{ for } i = 1, 2, \ldots, r \tag{6-9}$$

where the X_i contains the information about the engine operating conditions at the i^{th} 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

$$\gamma_i \sim N_r(0,D) \tag{6-10}$$

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

$$Var(p_i) = \left[\frac{\partial g(\theta_i)}{\partial \theta}\right]\sigma^2 C_i \left[\frac{\partial g(\theta_i)}{\partial \theta}\right]^T \tag{6-11}$$

For the sake of simplicity, the notation $\sigma^2 C_i$ is to denote $Var(p_i)$. Thus, p_{ii} is distributed as

$$p_i \sim N_r(p_i, \sigma^2 C_i) \tag{6-12}$$

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

$$p_i \sim N(X_i\beta, \sigma^2 C_i + D) \qquad \textbf{(6-13)}$$

When all the tests are considered simultaneously, equation (6-13) can be written in the compact form

$$P \sim N(Z\beta, W(\varpi)) \qquad \textbf{(6-14)}$$

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

$$\log L(\beta, \varpi) = \log|W| + (P - Z\beta)'W^{-1}(P - Z\beta) \qquad \textbf{(6-15)}$$

Thus, the maximum likelihood estimates are the vectors β_{ML} and ω_{ML} that minimize logL(β,ω).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 logL(β,ω). 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:

$$\beta = (Z'W^{-1}Z)^{-1}(Z'W^{-1}P) \qquad \textbf{(6-16)}$$

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 6-15 yields

$$\mathrm{Var}(\beta) = (Z^T W^{-1} Z)^{-1} Z^T W^{-1} \mathrm{Var}(P) W^{-1} Z (Z^T W^{-1} Z)^{-1} \text{ so}$$

$$\mathrm{Var}(\beta) = (Z^T W^{-1} Z)^{-1}, \qquad \textbf{(6-17)}$$

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

$$\hat{p}_i = z_i \hat{\beta} \qquad \textbf{(6-18)}$$

where z_i is an appropriate global covariate matrix. Applying the variance operator to Equation 6-18 yields

$$\mathrm{Var}(\hat{p}_i) = z_i \mathrm{Var}(\hat{\beta}) z_i^T = z_i (Z^T W^{-1} Z)^{-1} z_i^T \qquad \textbf{(6-19)}$$

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.

$$\hat{\theta}_i = h(\hat{p}_i) \qquad \textbf{(6-20)}$$

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,

$$\mathrm{Var}(\hat{\theta}_i) = \dot{h} \mathrm{Var}(\hat{p}_i) \dot{h}^T \qquad \textbf{(6-21)}$$

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

$$\hat{y}_i = f(\theta_i) \tag{6-22}$$

Again, after approximating `f` by a first order Taylor series and applying the variance operator to the result,

$$\mathrm{Var}(\hat{y}_i) = \left[\frac{\partial f}{\partial \theta}\right]\Bigg|_{\hat{\theta}} \dot{h}\mathrm{Var}(\hat{p}_i)\dot{h}^T \left[\frac{\partial f}{\partial \theta}\right]\Bigg|_{\hat{\theta}}^T \tag{6-23}$$

After subsituting Equation 6-19 into Equation 6-23 the desired result is obtained:

$$\mathrm{Var}(\hat{y}_i) = \left[\frac{\partial f}{\partial \theta}\right]\Bigg|_{\hat{\theta}} \dot{h}z_i(Z^T W^{-1} Z)^{-1} z_i^T \dot{h}^T \left[\frac{\partial f}{\partial \theta}\right]\Bigg|_{\hat{\theta}}^T \tag{6-24}$$

This equation gives the value of Prediction Error Variance.

See also the introduction to “Prediction Error Variance” on page 6-7 for details about PEV for one-stage models.

Global Model Selection

Before undertaking the minimization of Equation 6-15 (see "Two-Stage Models" on page 6-41) it is first necessary to establish the form of the $\mathbf{X}_\mathrm{i}$ 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 6-9. 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],

$$D_{STS} = \frac{1}{r-1}\sum_{i=1}^{r}(p_i - X_i\beta)(p_i - X_i\beta)^T \qquad \textbf{(6-25)}$$

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 **D** is positive definite. It is a simple matter to ensure this by noting that **D** is positive definite if and only if there is an upper triangular matrix, **G**, say, such that

$$D = G'G \qquad \textbf{(6-26)}$$

This factorization is used in the Quasi-Newton algorithm. Primarily, the advantage of this approach is that the resulting search in **G**, as opposed to **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, ch. 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 6-48

"Polynomial Splines" on page 6-49

"Truncated Power Series Basis (TPSBS) Splines" on page 6-50

"Free Knot Splines" on page 6-51

"Three Parameter Logistic Model" on page 6-51

"Morgan-Mercer-Flodin Model" on page 6-52

"Four-Parameter Logistic Curve" on page 6-53

"Richards Curves" on page 6-54

"Weibul Growth Curve" on page 6-55

"Exponential Growth Curve" on page 6-55

"Gompertz Growth Model" on page 6-56

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 n^{th} 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,

$$y_{ij} = \beta_o + \sum_{a=2}^{c} \beta_{Low\_a} \left(x_j - k\right)_+^a + \sum_{b=2}^{h} \beta_{High\_b} \left(x_j - k\right)_+^b$$

where k is the knot location, β denotes a regression coefficient,

$$\left(\mathrm{x}_j - \mathrm{k}\right)_- = \min\left\{0, \left(\mathrm{x}_j - \mathrm{k}\right)\right\}, \quad \left(\mathrm{x}_j - \mathrm{k}\right)_+ = \max\left\{0, \left(\mathrm{x}_j - \mathrm{k}\right)\right\},$$

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 $\left(\mathrm{x}_j - \mathrm{k}\right)_-$ and $\left(\mathrm{x}_j - \mathrm{k}\right)_+$ 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 $\{\mathrm{k}\}$
- Value of the fit function at a user-specified delta

  $$\left\{\Delta \mathrm{a}_j = \mathrm{x}_j - \mathrm{k}\right\}$$

  from the knot position $\left\{\mathrm{f}\left(\pm\Delta \mathrm{a}_j\right)\right\}$ 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

  $$\left\{\mathrm{f}\left(\pm\Delta \mathrm{a}_j\right) - \mathrm{f}(\mathrm{k})\right\}$$

Truncated Power Series Basis (TPSBS) Splines

A very general class of spline functions with arbitrary (but strictly increasing) knot sequence:

$$\mathbf{k} = \left\{k_1, k_2, \ldots, k_k\right\}^T :$$

$$f\left(x_j\right) \cong \sum_{i=0}^{m-1} \beta_i x_j^{m-1} + \sum_{i=1}^{k} \beta_{m-1+i} \left(x_j - k_i\right)_+^{m-1}$$

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 $\{\mathbf{k}\}$
- Value of the fit function $\left\{f\left(a_j\right)\right\}$ at a user-specified value $\left\{a_j\right\}$
- Value of the n^th^ derivative of the fit function with respect to x_j $\left\{f\left(a_j\right)^n\right\}$ 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 evaluatıon, as tne 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 $\left\{f\left(a_j\right)\right\}$ 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 wnen $x_j = \gamma$. 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:

$$g = \left[\alpha \ \beta \ \kappa \ \gamma \ \frac{\left(\kappa\alpha - \log\left(\frac{1+\kappa}{\kappa-1}\right)\right)}{\kappa}\right]^T$$

Richards Curves

The Richards curves family of growth models is defined by the equation

$$y_j = \alpha\left[1 + (\delta - 1)\exp\left(-\kappa\left(x_j - \gamma\right)\right)\right]^{1/(1-\delta)} \quad \delta \neq 1$$

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

$$g = \left[\alpha \ \gamma \ \kappa \ \delta \ \frac{\delta\kappa}{\delta 2\delta + 1}\right]^T$$

Weibul Growth Curve

The Weibul growth curve is defined by the equation

$$y_j = \alpha - (\alpha - \beta)\exp\left(-(\kappa x_j)^{\delta}\right)$$

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

$$g = \left[\alpha\ \beta\ \kappa\ \delta\ \left(\frac{1}{\kappa}\right)\left(\frac{\delta - 1}{\delta}\right)^{1/\delta}\right]^T$$

Exponential Growth Curve

The exponential growth model is defined by

$$y_j = \alpha - (\alpha - \beta)\exp(-\kappa x_j)$$

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

$$\mathbf{g} = [\alpha \quad \beta \quad \kappa]^T$$

Gompertz Growth Model

Another useful formulation that does not exhibit a symmetric point of inflection is the Gompertz growth model. The defining equation is

$$y_j = \alpha \exp\left(-e^{-\kappa(x_j-\gamma)}\right)$$

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 MBC Toolbox, see the documentation in the Neural Network Toolbox. At the MATLAB command line, enter

```
doc nnet
```

The training algorithms available in MBC are `traingdm`, `trainlm`, and `trainbr`.

These algorithms are a subset of the ones available in the Neural Network 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 MBC and hence should not be changed.

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 into 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');
```

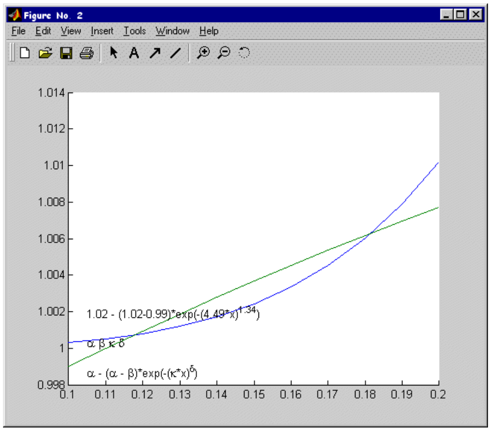
Call `checkin` with the transient model, its name, and some appropriate data.

```
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
```

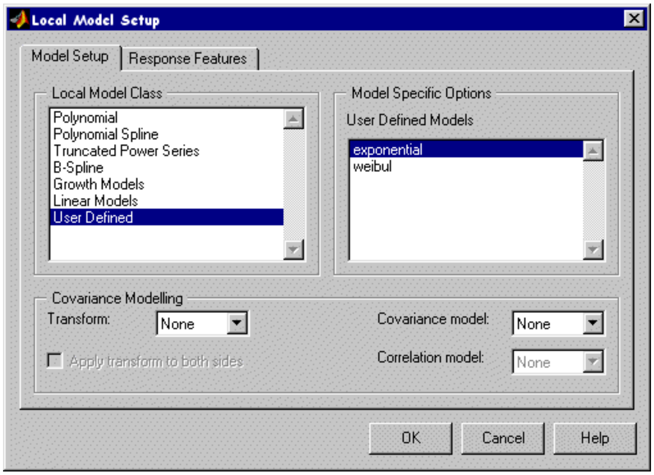
The figure displayed for `weibul` is



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:



Local Model Class

- Polynomial
- Polynomial Spline
- Truncated Power Series
- B-Spline
- Growth Models
- Linear Models
- User Defined

Model Specific Options

User Defined Models

- exponential
- weibul

Covariance Modelling

Transform: None

□ Apply transform to both sides

Covariance model: None

Correlation model: None

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. You can define a dynamic model using Simulink and a template 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.

The following sections describe the process using an example called `fuelpuddle`.

Throughout this section you use this Simulink model:

```
<MATLAB root>\toolbox\mbc\mbcsimulink\fuelPuddle.mdl
```

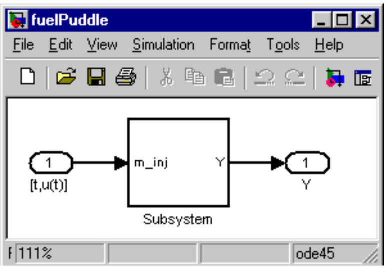
and the template file

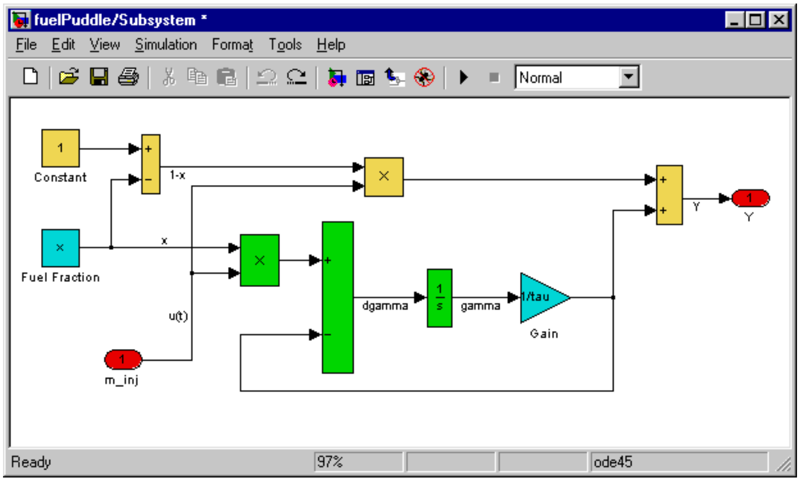
```
<MATLAB root>\toolbox\mbc\mbcmodel\@xregtransient\fuelPuddle.m
```

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 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.

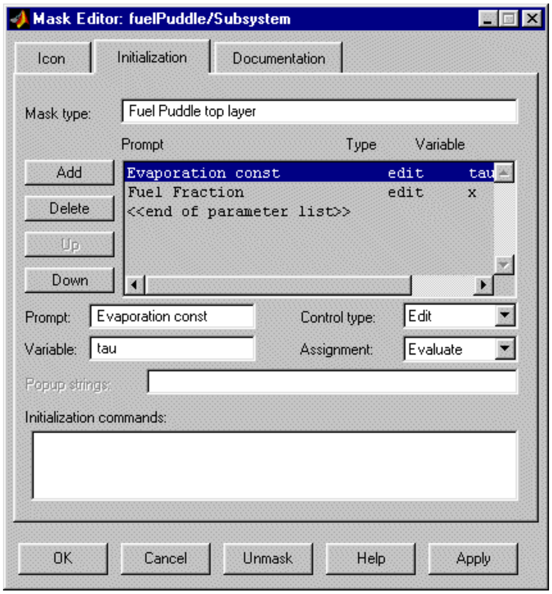
The model returns a single vector output (here labeled "Y").

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 we require two parameters. 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.



Mask Editor: fuelPuddle/Subsystem

Mask type: Fuel Puddle top layer

```
Prompt                    Type      Variable
Evaporation const         edit      tau
Fuel Fraction             edit      x
<<end of parameter list>>
```

Prompt: Evaporation const

Variable: tau

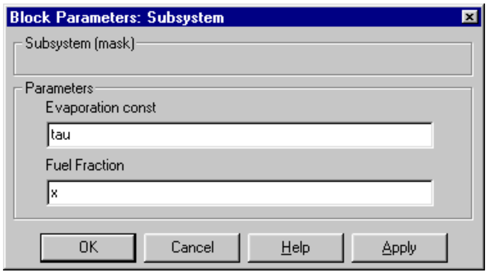
Control type: Edit

Assignment: Evaluate

Popup strings:

Initialization commands:

2 On double-clicking on 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.



Block Parameters: Subsystem

Subsystem (mask)

Parameters

Evaporation const: tau

Fuel Fraction: x

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 fuel `Puddle.m`.

Checking the Model

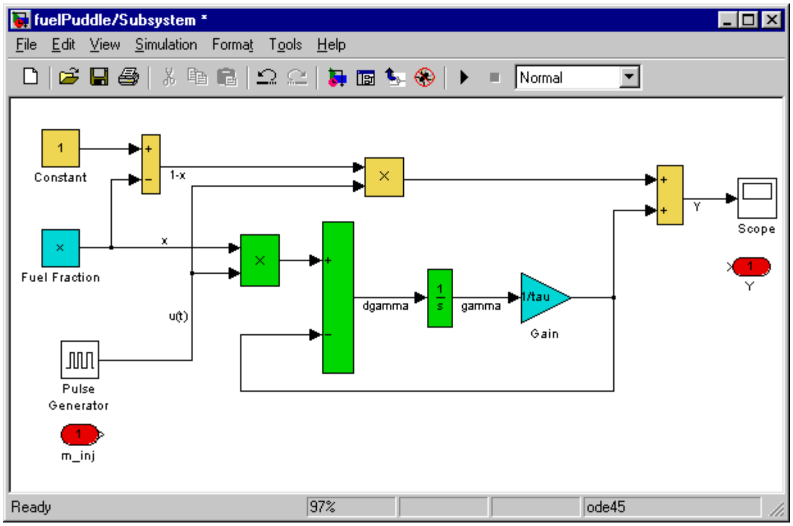
Open your Simulink model and clear the MATLAB workspace.

Define the masked block parameters in the workspace. Here, for example, we 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 (press 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.



Transient Models

You can create a transient model at the command line:

```
m = xregtransient
m = functemplate([X1,X2],b=[0.5,0.5])
```

The default transient model is created. This model can be evaluated and fitted, but it uses the default Simulink model, probably not a model that you are interested in! In order to create a transient model that uses your 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. Several sections of the template file are often not actively used but must be filled in with sensible defaults.

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. The m-file is called using

```
varargout= fuelPuddle(U,X,varargin)
U = the transient model
X = input data e.g. X = [[0:100] , sin([0:100] )]
```

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 6-65).

```
function [vars,vals]= i_simconstants(U,b,varargin);
vars = {};
vals = [];
```

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. The steady state can be passed in and you can calculate this from supplied data.

```
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,varargin);
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. 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 into MBC

Having created a Simulink model and the corresponding template file, save each somewhere on the path. One place to put these files is

```
<MATLAB root>\toolbox\mbc\mbcsimulink\
```

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`. For `fuelPuddle` the procedure would be

1 Create 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 called `fuelPuddle`:

```
m = xregtransient('name','fuelPuddle');
```

3 Call `checkin` with the transient model, its name, and the data.

```
checkin(m, 'fuelpuddle', X);
```

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 `fuelPuddle.`

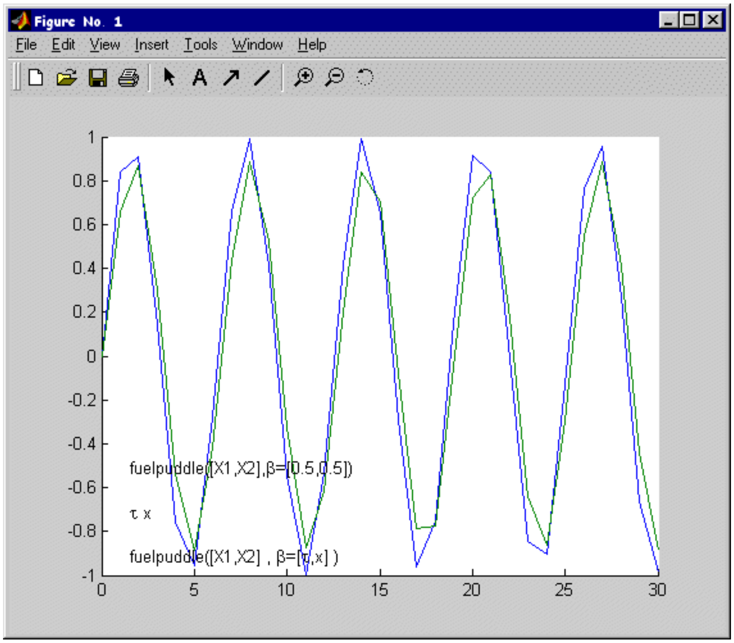
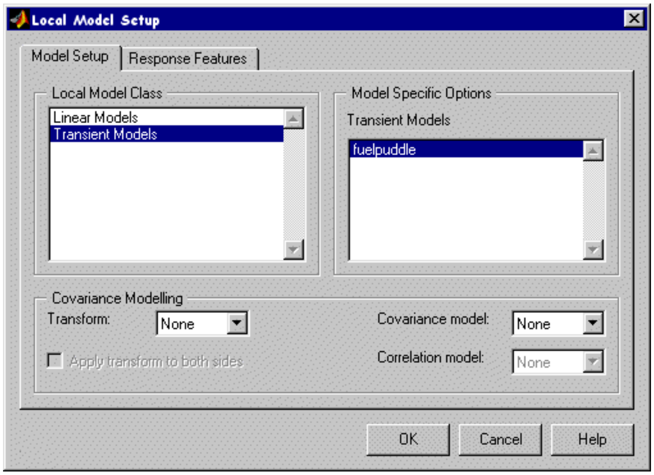


Figure No. 1
File Edit View Insert Tools Window Help
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
5
10
15
20
25
30
fuelpuddle([X1,X2],β=[0.5,0.5])
τ x
fuelpuddle([X1,X2] , β=[τ,x])

MBC Modeling

With the `fuelPuddle` model successfully checked in, start up MBC modeling and load data that has the necessary format to evaluate this transient model.

Set up a **Test Plan** with two `Stage1` input factors (the same number of input factors required by the `fuelPuddle` model). The **Local Model Setup** now offers the following options:



Local Model Class: Linear Models, Transient Models

Model Specific Options

Transient Models: fuelpuddle

Covariance Modelling

Transform: None

□ Apply transform to both sides

Covariance model: None

Correlation model: None

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.

Notes

In `@xregtransient` there is a template for `fuelPuddleDelay`, and the Simulink model can be found in `mbc\mbcsimulink`. You can check this model in using

```
t = [0:30]'; u = sin(t);
X = [t,u];
m = xregtransient('name','fuelPuddleDelay');
checkin(m, 'fuelpuddledelay', X);
```

And now at the MBC **Local Model Setup** dialog, this model is also available.

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-by-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`'.

Radial Basis Functions

Guide to Radial Basis Functions for Model Building . . 7-3

Types of Radial Basis Functions 7-4

Fitting Routines 7-12

Center Selection Algorithms 7-13

Lambda Selection Algorithms 7-16

Width Selection Algorithms 7-19

Prune Functionality 7-21

Statistics 7-24

Hybrid Radial Basis Functions 7-28

Tips for Modeling with Radial Basis Functions 7-30

This Radial Basis Functions chapter is divided into the following sections:

- "Guide to Radial Basis Functions for Model Building" on page 7-3
- "Types of Radial Basis Functions" on page 7-4
- "Fitting Routines" on page 7-12
- "Center Selection Algorithms" on page 7-13
- "Lambda Selection Algorithms" on page 7-16
- "Width Selection Algorithms" on page 7-19
- "Prune Functionality" on page 7-21
- "Statistics" on page 7-24
- "Hybrid Radial Basis Functions" on page 7-28
- "Tips for Modeling with Radial Basis Functions" on page 7-30

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 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 7-27) 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).

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

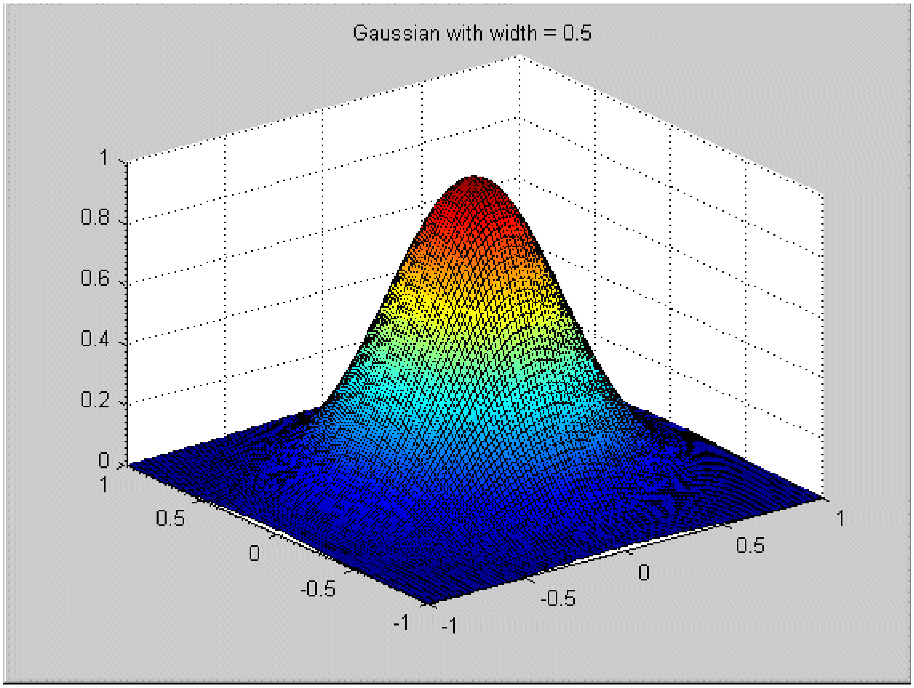
This is the radial basis function most commonly used in the neural network community. Its profile function is

$$\Phi(r) = e^{(-r^2/\sigma^2)}$$

This leads to the radial basis function

$$z(x) = \exp\left(\frac{\|x-\mu\|^2}{\sigma^2}\right)$$

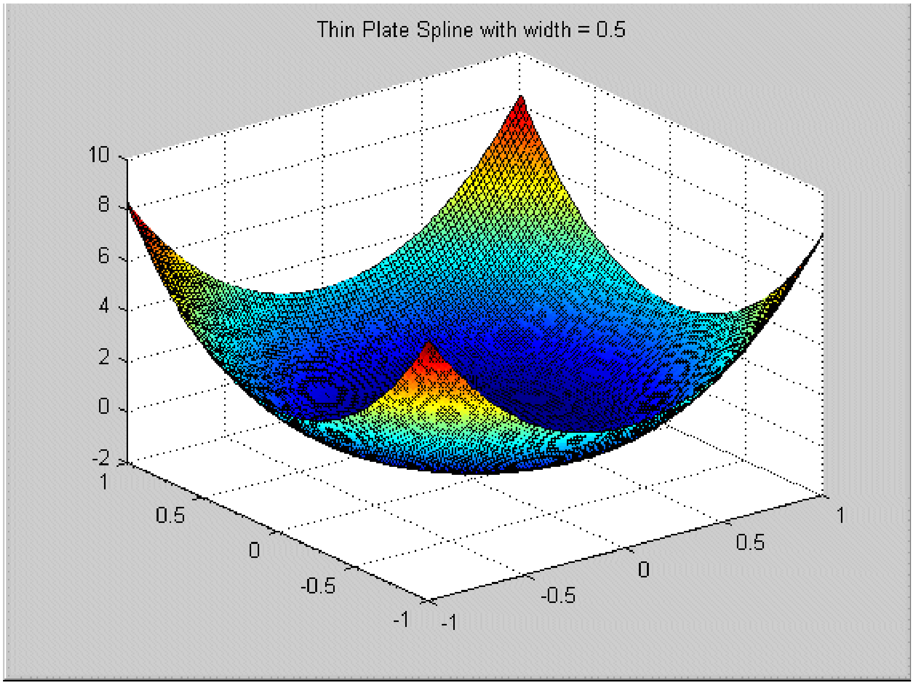
In this case, the width parameter is the same as the standard deviation of the Gaussian function.


Gaussian with width = 0.5
1
0.8
0.6
0.4
0.2
0
1
0.5
0
-0.5
-1
-1
-0.5
0
0.5
1

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

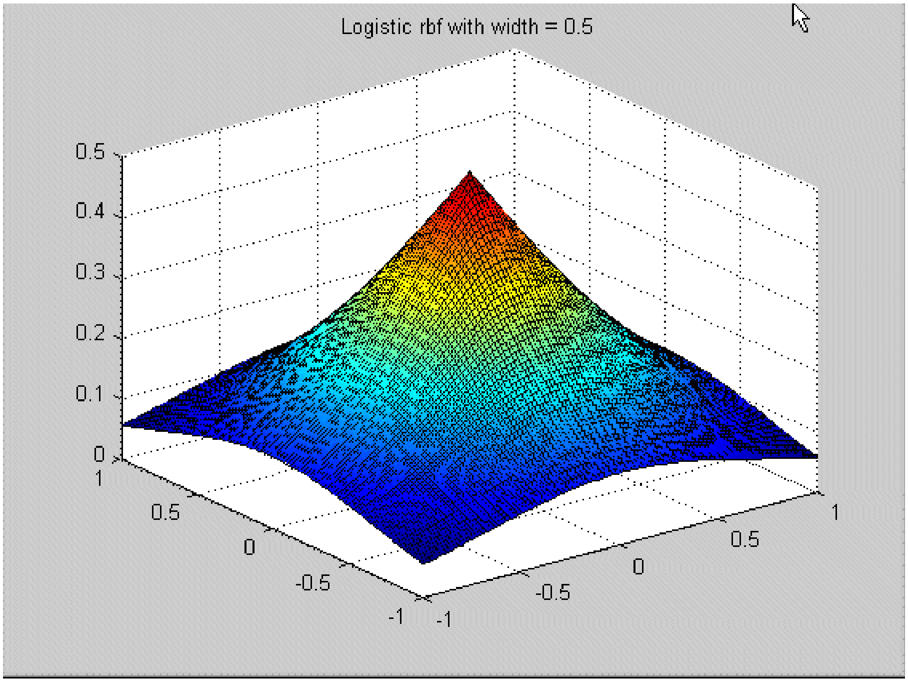
$$\Phi(r) = (r/\sigma)^2 \log(r/\sigma)$$



Logistic Basis Function

These radial basis functions are mentioned in Hassoun (see “References” on page 7-27). They have the profile function

$$\Phi(r) = \frac{1}{1 + \exp(\frac{r}{\sigma^2})}$$



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 7-27]. 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.

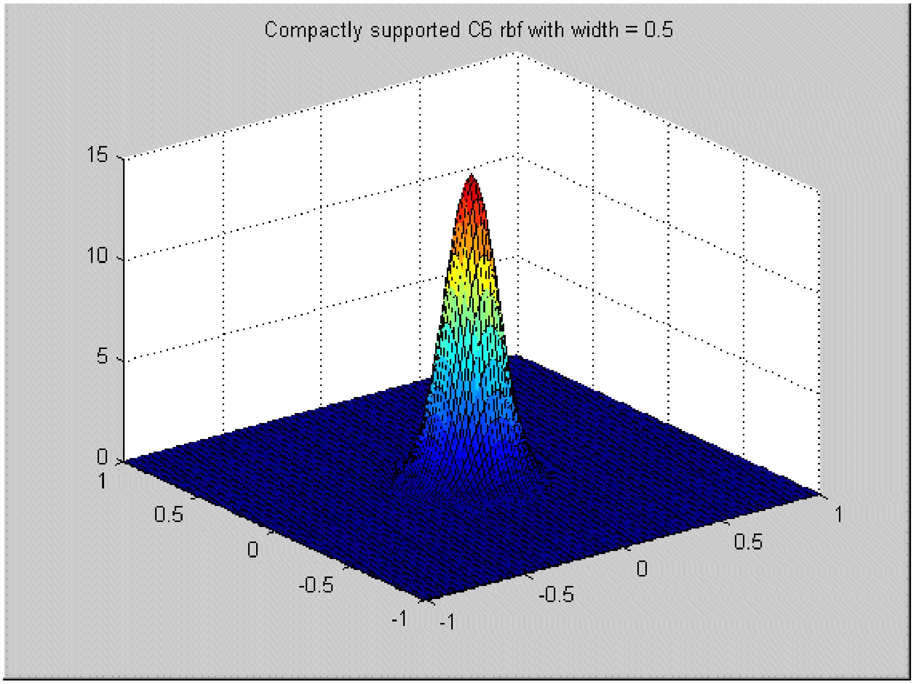
When n is even, the radial basis function corresponding to dimension n+1 is used.

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 $[0,\sigma]$ 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

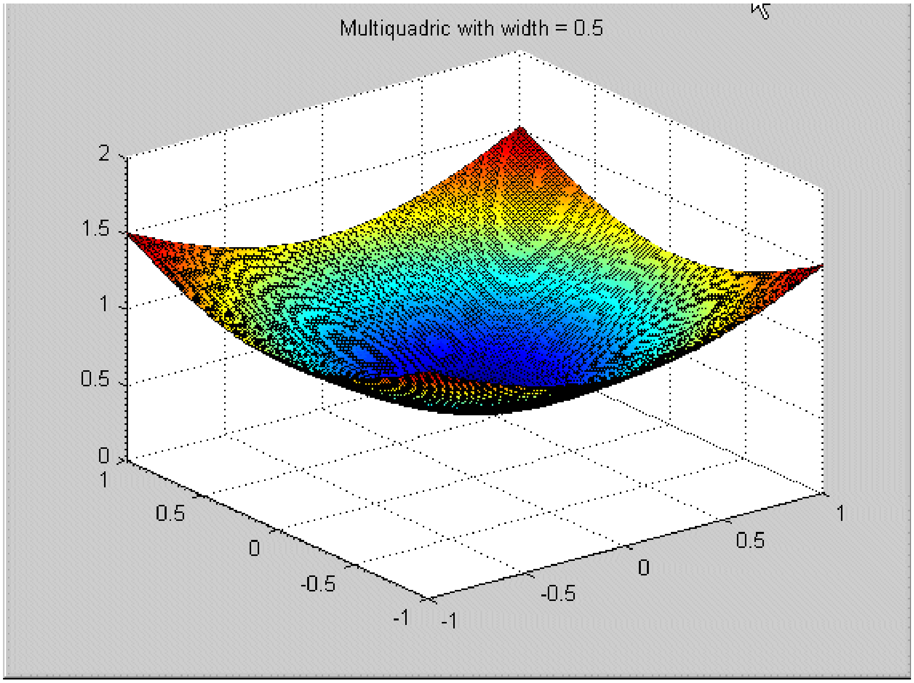
- Better approximation properties are usually associated with higher continuity.
- For a given data set the width parameter for Wendland’s functions should be larger than the width chosen for the Gaussian.



Multiquadrics

These are a popular tool for scattered data fitting. They have the profile function

$$\Phi(r) = \sqrt{r^2 + \sigma^2}$$

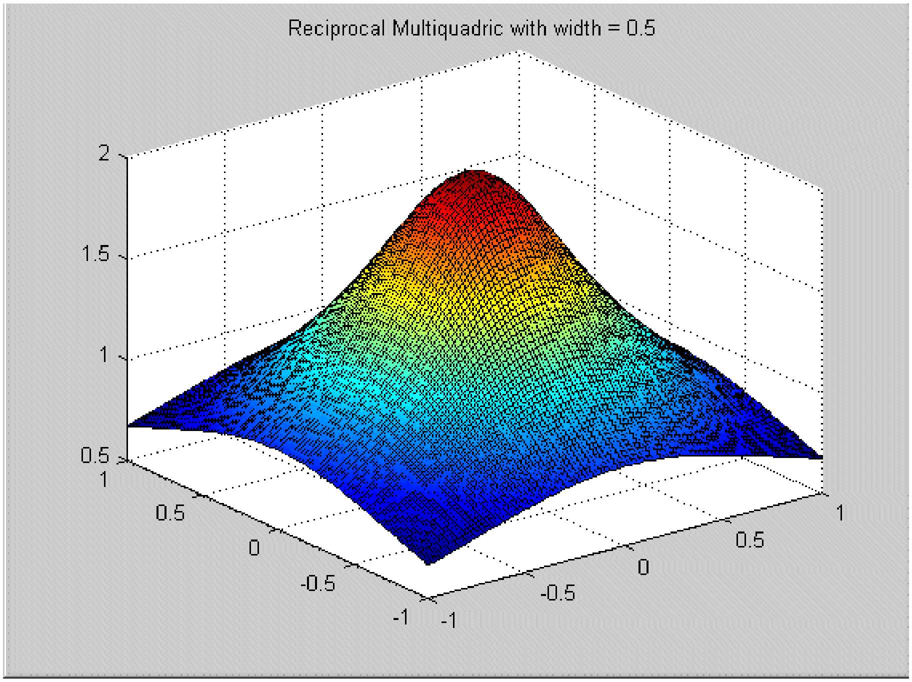


Reciprocal Multiquadrics

These have the profile function

$$\Phi(r) = 1/\sqrt{r^2 + \sigma^2}$$

Note that a width σ of zero is invalid.



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.

Center Selection Algorithms

Rols

This is the basic algorithm as described in Chen, Chng, and Alkadhimi [See "References" on page 7-27]. 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 ten 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 7-27] 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**.

This is the only algorithm that permits centers that are not located at the data points. The 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 are selected either 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 $\mathrm{GCV} - \lambda$ 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.

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.

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 trialing 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: Mid level 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.

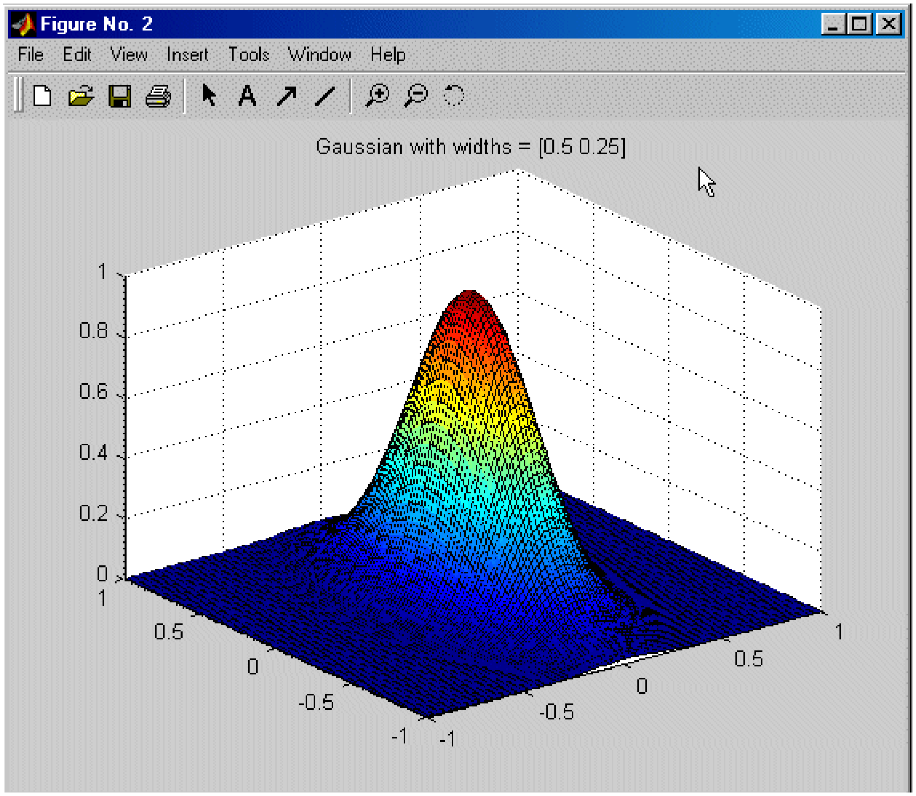


Figure No. 2
File Edit View Insert Tools Window Help
Gaussian with widths = [0.5 0.25]
1
0.8
0.6
0.4
0.2
0
1
0.5
0
-0.5
-1
-1
-0.5
0
0.5
1

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.

Fit Parameters

As for the **TrialWidths** algorithm.

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.

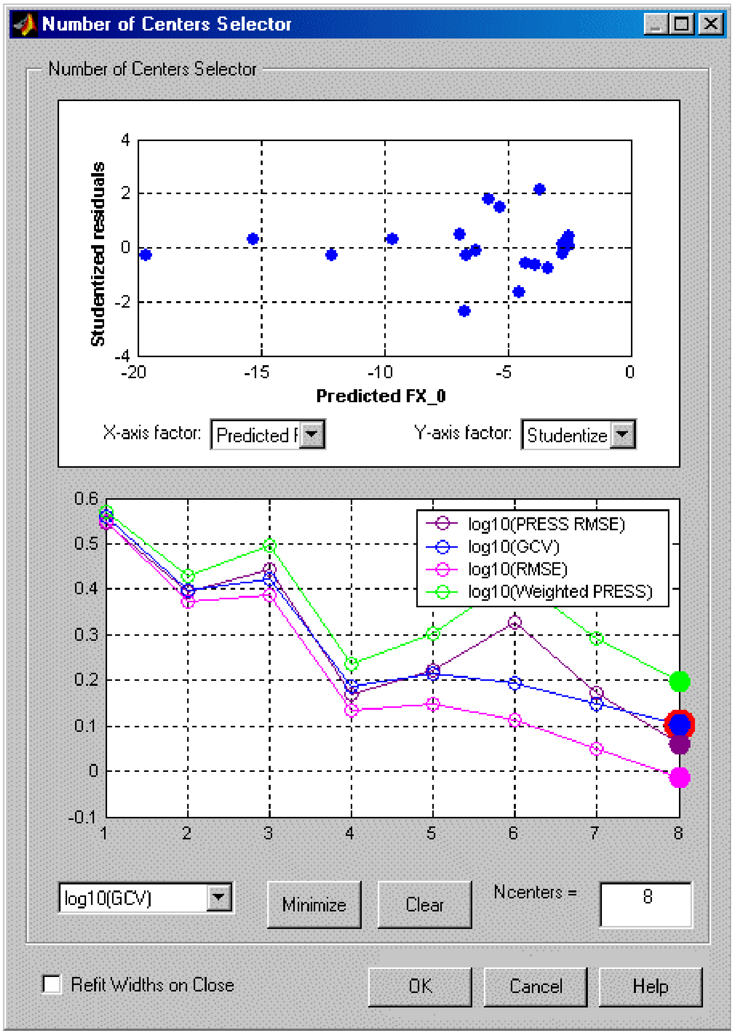
To use the Prune facility:

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 four fit criteria are the logarithms of PRESS, GCV, RMSE, and Weighted PRESS. 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 + λ I.

The most complicated situation is for the Rols algorithm. Recall that 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.

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 7-27).

$$\mathrm{GCV} = \frac{p(y'P^2y)}{(trace(P))^2}$$

where y is the target vector, and P is the projection matrix, given by I - $XA^{-1}X^T$. 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 7-27)

$$y'P\frac{\partial(Py)}{\partial\lambda}trace(P) = (y'P^2y)\frac{\partial(trace(P))}{\partial\lambda}$$

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 7-27) that for the case of ridge regression GCV can be written as

$$GCV = \frac{p(e'e)}{(p-\gamma)^2}$$

where γ is the “effective number of parameters” that is given by

$$\gamma = N - \lambda trace(A^{-1})$$

The formula for updating λ is given by

$$\lambda = \frac{\eta}{p-\gamma}\frac{(e'e)}{(\beta'A^{-1}\beta)^2}$$

where $\eta = tr(A^{-1} - \lambda A^{-2})$

In practice, the preceding formulas are not used explicitly in Orr (5, see “References” on page 7-27). 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.

GCV for Rols

In the case of Rols, the components for the formula

$$\text{GCV} = \frac{p(y'P^2y)}{(trace(P))^2}$$

are computed using the formulas given in Orr [6, see “References” on page 7-27]. 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

$$y'P^2y = y'y - \sum_{j=1}^{p} \frac{(2\lambda + w_j'w_j)(y'w_j)^2}{(\lambda + w_j'w_j)^2}$$

and

$$Trace(P) = N - \sum_{j=1}^{p} \frac{w_j'w_j}{(\lambda + w_j'w_j)}$$

The reestimation formula for λ is given by

$$\lambda = \frac{\eta}{Trace(P)} \frac{y'P^2y}{(\beta'A^{-1}\beta)^2} \text{ where additionally}$$

$$\eta = \sum_{j=1}^{p} \frac{w_j'w_j}{(\lambda + w_j'w_j)^2} \text{ and } \beta'A^{-1}\beta = \sum_{j=1}^{p} \frac{(y'w_j)^2}{(\lambda + w_j'w_j)^3}$$

Note that these formulas for Rols do not require the explicit inversion of A.

General Points

1 You can view the location of the centers in graphical and table format by using the (View Centers) toolbar button. If a center coincides with a data point, it is marked with a magenta asterisk on the Predicted/Observed plot.

2 You can alter the parameters in the fit by clicking the **Set Up** button in the **Model Selection** dialog.

3 An estimation of the time for each of the top-level algorithms 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.

4 If too many graphs are likely to be produced, because of the Display check box's being activated on a lower level algorithm, a warning is generated, and you have the option to stop execution.

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., Optimising 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 pressing 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.

3 Try with more than one kernel.

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 whether this would affect the path you have chosen.

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 are 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).

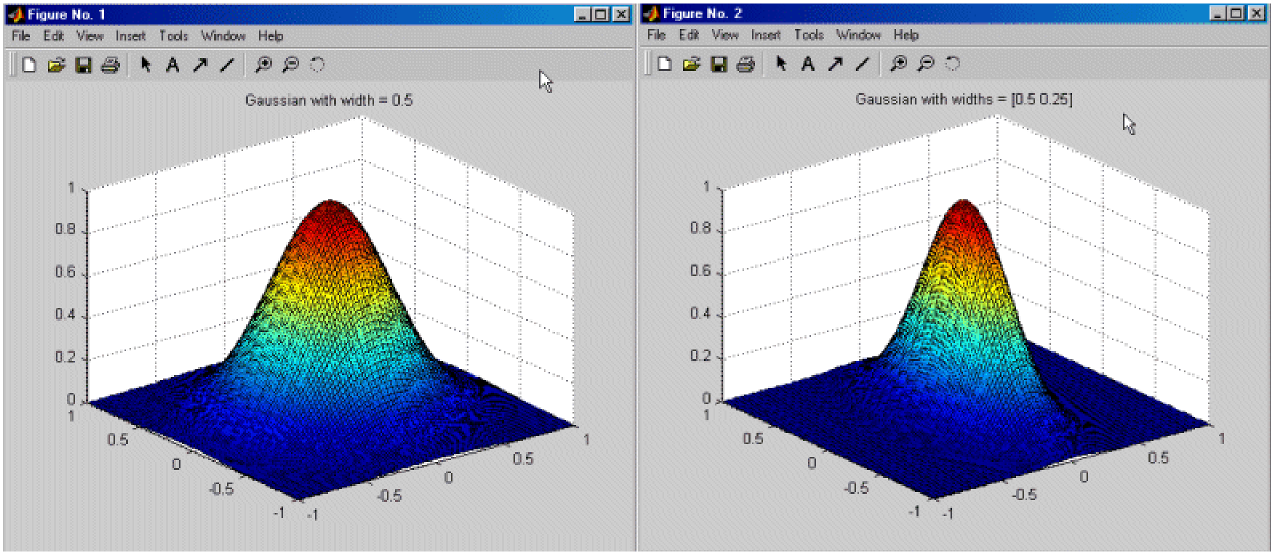


Figure No. 1
File Edit View Insert Tools Window Help
Gaussian with width = 0.5
1
0.8
0.6
0.4
0.2
0
1
0.5
0
-0.5
-1
-1
-0.5
0
0.5
1
Figure No. 2
File Edit View Insert Tools Window Help
Gaussian with widths = [0.5 0.25]
1
0.8
0.6
0.4
0.2
0
1
0.5
0
-0.5
-1
-1
-0.5
0
0.5
1

- 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** than **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 trialed; 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 7-28.
- 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 fewer 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.

Index

B

- Box-Cox transformation 6-18

C

- constraints manager 5-184
- correlation models 5-56
- covariance modeling 5-54

D

- data 5-59
  - import wizard 5-65
  - loading from file 5-65
  - loading from the Workspace 5-66
  - loading user-defined functions 6-73
  - matching to designs 5-81
  - merging 5-66, 5-68
  - wizard 5-78
- data editor 5-61
- data selection window 5-81
- datum models 5-88
- definitions 6-5
- design
  - adding points 5-178
  - classical 5-159
  - constraints 5-184
  - deleting points 5-181
  - fixing points 5-181
  - importing 5-183
  - optimal 5-168
  - saving 5-183
  - sorting points 5-181
  - space filling 5-162
  - styles 5-154
  - tree 5-156
  - tutorial 3-1
- Design Editor 5-153
  - display options 5-156
  - displays 5-155
- designing experiments 5-57

F

- filter editor 5-72
- filters 5-69

G

- global level 5-105
- global models
  - free knot spline 5-44
  - hybrid RBF 5-42
  - linear
    - hybrid spline 5-34
    - multiple 5-43
    - polynomial 5-33
  - radial basis functions 5-39
  - specifying inputs to 2-16
  - specifying type of 2-18
- global regression 6-3

H

- hierarchical models 6-37

I

- inputs
  - setting up 5-30
- interaction 5-37

L

linear model statistics 6-22
local level 5-89
local model definitions 6-48
local model setup 5-46
local models
  B-spline 5-51
  growth models 5-52
  linear 5-53
  polynomial 5-47
  polynomial spline 5-48
  transient 5-54, 6-64
  truncated power series 5-50
  user-defined 5-54, 6-58

M

Maximum Likelihood Estimation 6-36
MLE 5-137
model building process overview 6-9
model evaluation 5-144
model selection 5-117
model setup
  global 5-32
  local 5-46
  response 5-87
model tree 5-10
models
  setting up 5-26

N

neural nets 6-57
new test plan templates 5-16

O

one-stage model
  instant setup 5-27

P

Prediction Error Variance (PEV) viewer 5-193
project level 5-3

R

radial basis functions
  fitting routines 7-12
  guide 7-3
  hybrid RBFs 7-28
  prune functionality 7-21
  statistics 7-24
  tips for modeling 7-30
requirements, system 1-6
response level 5-141

S

stepwise 5-38
stepwise regression techniques 6-13
storage 5-74
system requirements 1-6

T

test groupings 5-75

test plan level 5-19

test plans 5-15

transforms 5-56

tutorial

- data editor 4-1
- design of experiment 3-1
- quick start 2-1

two-stage model

- instant setup 5-28

V

variable editor 5-71

variables

- user-defined 5-69

view

- cross-section 5-133
- likelihood 5-127
- predicted/observed 5-122
- residuals 5-132
- response surface 5-124
- RMSE 5-129
- tests 5-120

viewing designs 5-57