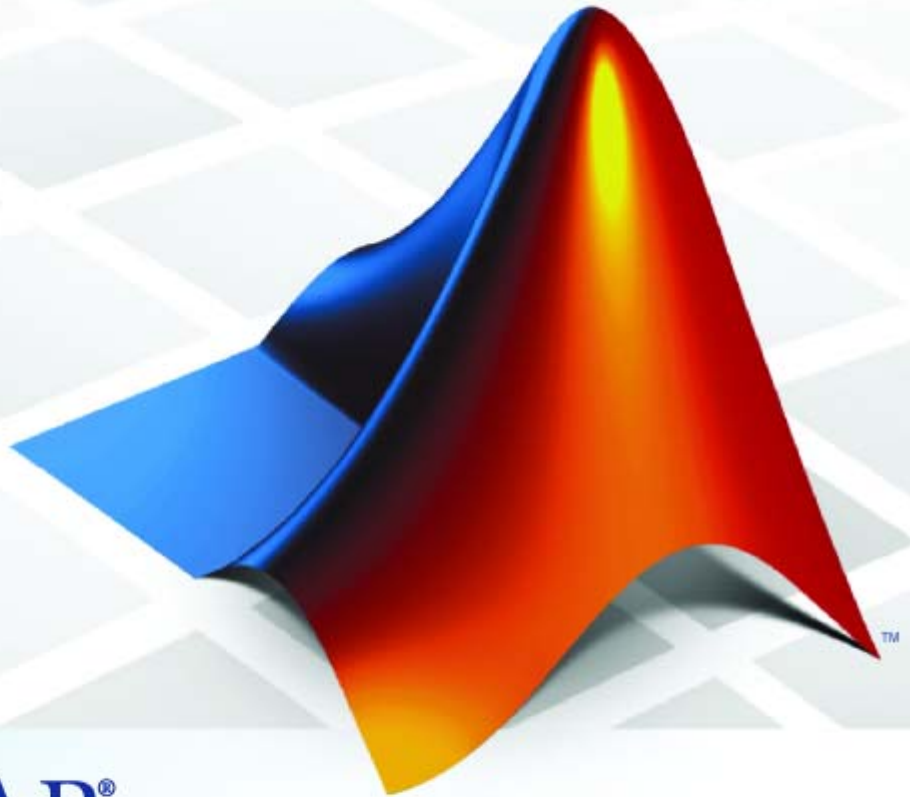


Simulink[®] Parameter Estimation[™] 1

User's Guide



MATLAB[®]
& **SIMULINK[®]**

How to Contact The MathWorks



www.mathworks.com Web
comp.soft-sys.matlab Newsgroup
www.mathworks.com/contact_TS.html 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

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

Simulink® Parameter Estimation™ User's Guide

© COPYRIGHT 2004–2008 by The MathWorks, Inc.

The software described in this document is furnished under a license agreement. The software may be used or copied only under the terms of the license agreement. No part of this manual may be photocopied or reproduced in any form without prior written consent from The MathWorks, Inc.

FEDERAL ACQUISITION: This provision applies to all acquisitions of the Program and Documentation by, for, or through the federal government of the United States. By accepting delivery of the Program or Documentation, the government hereby agrees that this software or documentation qualifies as commercial computer software or commercial computer software documentation as such terms are used or defined in FAR 12.212, DFARS Part 227.72, and DFARS 252.227-7014. Accordingly, the terms and conditions of this Agreement and only those rights specified in this Agreement, shall pertain to and govern the use, modification, reproduction, release, performance, display, and disclosure of the Program and Documentation by the federal government (or other entity acquiring for or through the federal government) and shall supersede any conflicting contractual terms or conditions. If this License fails to meet the government's needs or is inconsistent in any respect with federal procurement law, the government agrees to return the Program and Documentation, unused, to The MathWorks, Inc.

Trademarks

MATLAB and Simulink are registered trademarks of The MathWorks, Inc. See www.mathworks.com/trademarks for a list of additional trademarks. Other product or brand names may be trademarks or registered trademarks of their respective holders.

Patents

The MathWorks products are protected by one or more U.S. patents. Please see www.mathworks.com/patents for more information.

Revision History

June 2004	First printing	New for Version 1.0 (Release 14)
October 2004	Online only	Revised for Version 1.1 (Release 14SP1)
March 2005	Online only	Revised for Version 1.1.1 (Release 14SP2)
September 2005	Online only	Revised for Version 1.1.2 (Release 14SP3)
March 2006	Second printing	Revised for Version 1.1.3 (Release 2006a)
September 2006	Online only	Revised for Version 1.1.4 (Release 2006b)
March 2007	Online only	Revised for Version 1.2 (Release 2007a)
September 2007	Online only	Revised for Version 1.2.1 (Release 2007b)
March 2008	Online only	Revised for Version 1.2.2 (Release 2008a)
October 2008	Online only	Revised for Version 1.2.3 (Release 2008b)

Getting Started

1

Product Overview	1-2
What You Need to Get Started	1-3
Prerequisite Software and Optional Software	1-3
Required Knowledge	1-3
Demos	1-3
How Simulink® Parameter Estimation Software Works	1-5
Basic Steps in the Estimation Process	1-5
Structure of an Estimation Project	1-5
Adding, Deleting and Renaming an Estimation Project ...	1-7
Preparing a Model for Parameter Estimation	1-8
Setting Up the Estimation Data	1-9
Creating an Estimation Project	1-9
Importing Transient Data	1-11
Specifying Initial Conditions	1-16
Selecting Parameters for Estimation	1-17
Selecting States for Estimation	1-19
Specifying Initial Guesses and Upper/Lower Bounds	1-20
Setting Up an Estimation Project	1-23
Creating an Estimation Task	1-23
Adding Data Sets	1-23
Specifying and Setting Up Parameters	1-25
Specifying Settings for Estimation	1-27
Selecting Views for Plotting	1-29
Basic Steps for Creating Plots	1-29
Types of Plots	1-32

Running the Estimation	1-33
Model Validation	1-36
Basic Steps for Model Validation	1-36
Example: Validating the Engine Idle Speed Model	1-37
Loading and Importing the Validation Data	1-38
Performing Validation	1-39
Comparing Residuals	1-43
Setting Options for Optimization	1-46
Tuning the Results of Optimization	1-46
Selecting Optimization Methods	1-47
Selecting Optimization Termination Options	1-48
Selecting Additional Optimization Options	1-49
Specifying the Cost Function	1-49
Setting Options for the Simulation	1-51
Specifying Simulation Options	1-51
Selecting Simulation Time	1-52
Selecting Solvers	1-53
Accelerating Model Simulations During Estimation ..	1-56
About Accelerating Model Simulations During Estimation	1-56
Limitations	1-56
Setting the Accelerator Mode for Parameter Estimation ..	1-56
Estimating Independent Parameters	1-58
Basic Steps for Estimating Independent Parameters	1-58
Example: Estimating Independent Parameters	1-58

Tutorial — Preparing Data for Parameter Estimation Using the GUI

2

About This Tutorial	2-2
Objectives	2-2
About the Sample Data	2-2

Configuring a Project for Parameter Estimation	2-4
Importing Data into the GUI	2-6
Importing Input Data and Time Vector	2-6
Importing Output Data and Time Vector	2-11
Analyzing Data	2-14
Selecting Data for Estimation	2-16
Selecting Output Data	2-16
Selecting Input Data	2-22
Removing Outliers	2-25
Why Remove Outliers	2-25
How to Remove Outliers	2-25
Filtering Data	2-29
Filtering Output Data	2-29
Filtering Input Data	2-32
Interpolating Missing Data	2-34
Saving the Project	2-37

Tutorial — Estimating Parameters from Measured Data Using the GUI

3

About This Tutorial	3-2
Objectives	3-2
About the Model	3-3
Estimating Model Parameters Using Default Estimation Settings	3-7
Overview of the Estimation Process	3-7
Specifying Parameters and Estimation Data	3-8
Validating Model Parameters	3-13

Improving Estimation Results Using Parameter	
Bounds	3-20
Strategy for Improving the Estimation Results	3-20
How to Specify Parameter Bounds	3-20
Validating Estimated Model Parameters	3-26

Tutorial — Modeling a System Using Adaptive Lookup Table

4

About This Tutorial	4-2
Objectives	4-2
About the Data	4-2
Overview of Modeling a System Using Adaptive Lookup Table	4-3
Building a Model Using Adaptive Lookup Table	
Blocks	4-4
Adapting the Lookup Table Values Using Time-Varying I/O Data	4-16

Estimating Initial Conditions

5

Why Estimate Initial Conditions?	5-2
Estimating Initial Conditions for Blocks with External Initial Conditions	5-3
Example: Estimating Initial Conditions of a Mass-Spring-Damper System	5-4
Loading the Example	5-4
Model Parameters	5-5

Setting Up the Estimation Project	5-6
Importing Transient Data and Selecting Parameters for Estimation	5-7
Selecting Parameters and Initial Conditions for Estimation	5-8
Creating the Estimation Task	5-10
Running the Estimation and Viewing Results	5-11

Preprocessing Data

6

Why Preprocess Data?	6-2
Data Preprocessing Tool	6-3
Excluding Data	6-6
Techniques for Excluding Data in the Data Preprocessing Tool	6-6
Selecting Data for Exclusion from the Data Editing Table	6-6
Selecting Data for Exclusion from a Plot of the Data	6-9
Selecting Data for Exclusion by a Rule	6-12
Detrending and Filtering	6-15
Detrending and Filtering Data in the Data Preprocessing Tool	6-15
Detrending	6-15
Filtering	6-16
Handling Missing Data	6-17
Removing Missing Data	6-17
Interpolating Missing Data	6-17
Adding Preprocessed Data Sets to an Estimation Project	6-19
Overwriting an Existing Data Set	6-19
Creating a New Data Set	6-20

Exporting Preprocessed Data to the MATLAB Workspace	6-22
--	-------------

Managing Multiple Projects

7

Multiple Projects and Tasks	7-2
Saving Control and Estimation Tools Manager Projects	7-3
Opening Control and Estimation Tools Manager Projects	7-4

Approximating System Models Using Lookup Tables

8

What Are Lookup Tables?	8-2
Static Lookup Tables	8-2
Adaptive Lookup Tables	8-3
Estimating Values of Lookup Tables	8-5
How to Estimate Values of a Lookup Table	8-5
Example — Estimating Lookup Table Values from Data ..	8-6
Example — Estimating Constrained Values of a Lookup Table	8-20
Capturing Time-Varying System Behavior Using Adaptive Lookup Tables	8-37
Building Models Using Adaptive Lookup Table Blocks ...	8-37
Setting Adaptive Lookup Table Parameters	8-40
Selecting an Adaptation Method	8-41
Example: n-D Adaptive Lookup Table	8-43
Using Adaptive Lookup Tables in Real-Time Environment	8-46

Introduction	9-2
Example: Estimating Parameters and Initial Conditions of the F14 Model	
Loading the F14 Jet Model	9-4
Baseline Simulation	9-5
Creating a Transient Experiment Object	9-7
Assigning Experimental Data to Inputs and Outputs of the Model	9-8
Creating Parameter Objects for Estimation	9-9
Creating an Estimation Object and Running the Estimation	9-10
Creating and Customizing Estimation Projects	
Creating Transient Data Objects	9-15
What is a Transient Data Object	9-15
Constructor	9-15
Properties of Transient Data Objects	9-16
Modifying Properties of Transient Data Objects	9-18
Using Class Methods	9-19
Creating State Data Objects	
What is a State Data Object	9-20
Constructor	9-20
Properties of the State Data Object	9-21
Example: Initial Condition Data	9-22
Modifying Properties	9-22
Using Class Methods	9-22
Creating Transient Experiment Objects	
What is a Transient Experiment Object	9-23
Constructor	9-23
Properties of Transient Experiment Objects	9-23
Example: Creating an F14 Experiment	9-24
Example: Creating a Van der Pol Experiment from User Objects	9-25
Modifying Properties	9-25

Using Class Methods	9-25
Creating Parameter Objects	9-26
What is a Parameter Object	9-26
Constructor	9-26
Properties of Parameter Objects	9-26
Example: F14 Model	9-28
Example: Gain Matrix	9-29
Modifying Properties	9-29
Using Class Methods	9-29
Creating State Objects	9-30
What is a State Object	9-30
Constructor	9-30
Properties of State Objects	9-30
Example: F14 Model	9-32
Modifying Properties	9-33
Using Class Methods	9-33
Creating Estimation Objects	9-34
What is an Estimation Object	9-34
Constructor	9-34
Properties of Estimation Objects	9-34
Example: F14 Model	9-36
Modifying Properties	9-36
Using Class Methods	9-36

Block Reference

10

Function Reference

11

Examples

A

Getting Started	A-2
Estimating Parameters and Initial Conditions	A-2
Estimating Lookup Table Values	A-2

Index

Getting Started

- “Product Overview” on page 1-2
- “What You Need to Get Started” on page 1-3
- “How Simulink® Parameter Estimation Software Works” on page 1-5
- “Preparing a Model for Parameter Estimation” on page 1-8
- “Setting Up the Estimation Data” on page 1-9
- “Setting Up an Estimation Project” on page 1-23
- “Selecting Views for Plotting” on page 1-29
- “Running the Estimation” on page 1-33
- “Model Validation” on page 1-36
- “Setting Options for Optimization” on page 1-46
- “Setting Options for the Simulation” on page 1-51
- “Accelerating Model Simulations During Estimation” on page 1-56
- “Estimating Independent Parameters” on page 1-58

Product Overview

Simulink® Parameter Estimation™ software is a Simulink-based product for estimating and calibrating model parameters from experimental data. This product supports the following types of estimation:

- **Transient Estimation** — Estimate parameters by comparing model output to the experimental data for a given input.
- **Initial Condition Estimation** — Estimate the initial conditions of states using experimental data.
- **Adaptive Lookup Tables** — Estimate the table values at the prescribed breakpoints by using measurements from the physical system.

Simulink Parameter Estimation software provides the tools used to:

- 1** Set up the problem.
- 2** Specify which model parameters to estimate.
- 3** Import and prepare the experimental data for parameter estimation (or *preprocess*).
- 4** View the estimation progress.
- 5** Validate the estimation results based on plots of measured versus simulated data and residuals.

What You Need to Get Started

In this section...
“Prerequisite Software and Optional Software” on page 1-3
“Required Knowledge” on page 1-3
“Demos” on page 1-3

Prerequisite Software and Optional Software

Simulink Parameter Estimation software requires MATLAB® technical computing software, Simulink® software, and Optimization Toolbox™ software.

Several of The MathWorks™ products are relevant to the kinds of task you can perform with Simulink Parameter Estimation software. For more information about any of these products, see the

- MathWorks Web site at <http://www.mathworks.com/products/simparameter/related.jsp>
- Online documentation for related products, if they are installed on your system

Required Knowledge

It is not necessary that you have a strong background in optimization theory or practice. As you gain familiarity with Simulink Parameter Estimation software, you might find it helpful to consult the Optimization Toolbox documentation for more details about optimization algorithms.

Demos

Simulink Parameter Estimation demonstration files show you how to use the software to perform control design tasks in various settings. To run these demos, type

```
demo
```

at the MATLAB prompt. This opens the Demos pane in the Help browser. Select **Simulink > Simulink Parameter Estimation** to list the available demos. Alternatively, if you have the Help browser open, you can select the Demos pane in the Help browser and then select **Simulink > Simulink Parameter Estimation**.

How Simulink Parameter Estimation Software Works

In this section...

“Basic Steps in the Estimation Process” on page 1-5

“Structure of an Estimation Project” on page 1-5

“Adding, Deleting and Renaming an Estimation Project” on page 1-7

Basic Steps in the Estimation Process

Simulink Parameter Estimation software compares empirical data with data generated by a Simulink model. Using optimization techniques, the software estimates the parameter and (optionally) initial conditions of states such that a user-selected cost function is minimized. The cost function typically calculates a least-square error between the empirical and model data signals.

After you build a Simulink model, follow these steps to configure and run a parameter estimation:

- 1 Select **Tools > Parameter Estimation** in your Simulink model window.

This opens the Control and Estimation Tools Manager, creates a new project, and adds an **Estimation** node to the workspace directory tree.

- 2 Import the input and output data set for estimating parameters of your Simulink model.
- 3 Select the parameters and initial conditions you want to estimate.
- 4 Configure the estimation itself, including cost functions and data views.
- 5 Run the estimation.
- 6 Check the results by examining either the cost-function values, plots, or parameter values.

Structure of an Estimation Project

The Control and Estimation Tools Manager, which is a graphical user interface (GUI) for performing parameter estimation, stores and organizes all

data from a given Simulink model inside a *project*. To open the Control and Estimation Tools Manager GUI, select **Tools > Parameter Estimation** in the Simulink model window. See Control and Estimation Tools Manager GUI on page 1-11 for a picture showing this GUI.

When using the Control and Estimation Tools Manager for parameter estimation, you can

- Manage estimation projects.
- Select parameters and initial conditions to configure the estimation.
- Specify cost functions.
- Import experimental data (to be matched by the output of your Simulink model).
- Specify the initial conditions of your model.

Each estimation task can include

- One or more data sets
- Parameter information
- One or more sets of estimation settings, or configurations

The default project name is the same as the Simulink model name. The project name is shown in the *workspace directory tree* of the Control and Estimation Tools Manager.

You can also add tasks from Simulink® Control Design™ and Model Predictive Control Toolbox™ software to the current project, if these products are installed on your system.

Adding, Deleting and Renaming an Estimation Project

To add, delete, or rename the project or task:

- 1** Right-click the project or task node in the workspace directory tree.
- 2** Select the appropriate command from the shortcut menu.

Preparing a Model for Parameter Estimation

To import estimation data, verify that your Simulink model contains one of the following elements:

- Top-level Inport block

Note You do not need an Inport block if your model already contains a fixed input block, such as a Step block.

- Top-level Outport block
- Logged signal. The logged signal can be a top-level signal in the model or a signal in the model subsystem.

For more information about the blocks and logged signals, see the Inport and Outport block reference pages and “Logging Signals” in the Simulink documentation.

In the Control and Estimation Tools Manager GUI, the rows in the **Input Data** tab correspond to the model’s top-level Inport blocks. Similarly, the rows in the **Output Data** tab correspond to either the top-level Outport blocks or logged signals in the model.

Adding an Inport or Outport block or marking a signal for logging creates a new row in the corresponding **Input Data** or **Output Data** tab. You can use the new row to import estimation data for the corresponding signal. To view the new row, click **Update Task** in the **Estimation Task** node of the Control and Estimation Tools Manager GUI.

The `engine_idle_speed` model used in the following example contains the Inport block `BPAV` and Outport block `Engine Speed` for importing input and output data, respectively.

Setting Up the Estimation Data

In this section...

“Creating an Estimation Project” on page 1-9

“Importing Transient Data” on page 1-11

“Specifying Initial Conditions” on page 1-16

“Selecting Parameters for Estimation” on page 1-17

“Selecting States for Estimation” on page 1-19

“Specifying Initial Guesses and Upper/Lower Bounds” on page 1-20

Creating an Estimation Project

Before beginning the estimation process, you must create an estimation project and set up the problem by configuring the appropriate parameters, solvers, and cost functions.

Simulink Parameter Estimation software provides a Graphical User Interface (GUI) that makes setting up the estimation project quick and easy. This section describes how to use the GUI to estimate parameters.

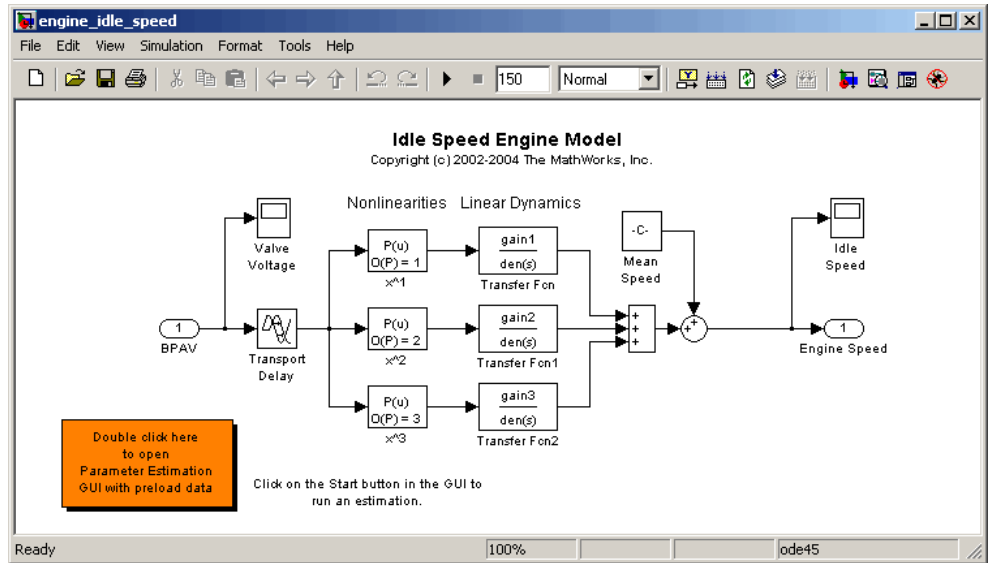
To perform the setup:

- 1 Open the nonlinear idle speed model of an automotive engine by typing :

```
engine_idle_speed
```

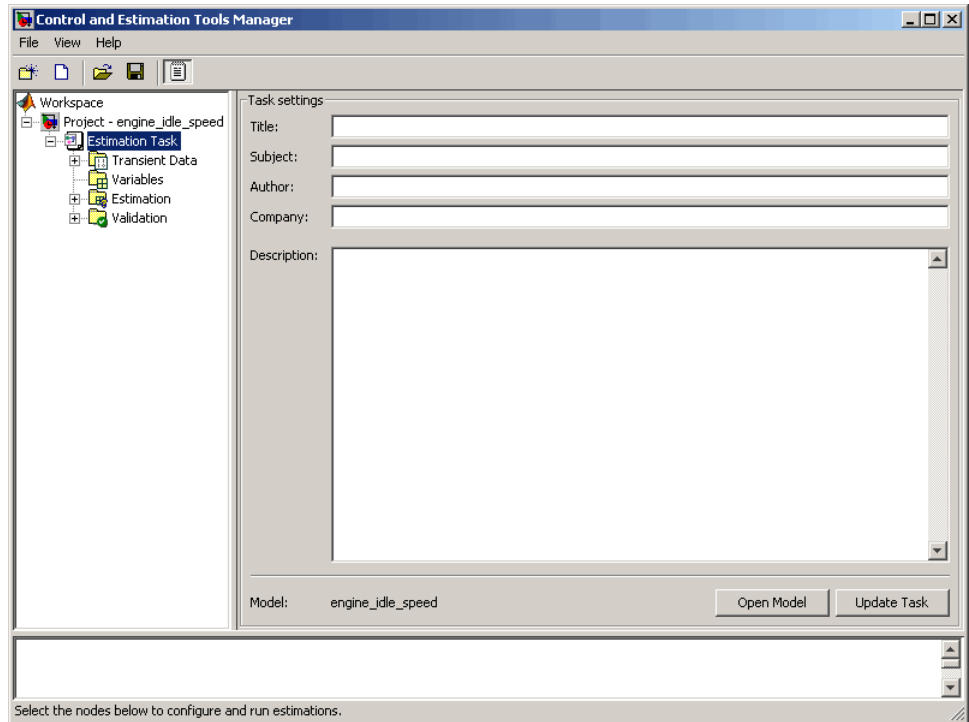
at the MATLAB prompt.

The model appears as shown next.



- 2 Open the Control and Estimation Tools Manager GUI by selecting **Tools** > **Parameter Estimation** in the Simulink model window.

The project tree displays the project name **Project - engine_idle_speed**. Estimation tasks are organized inside the **Estimation Task** node.

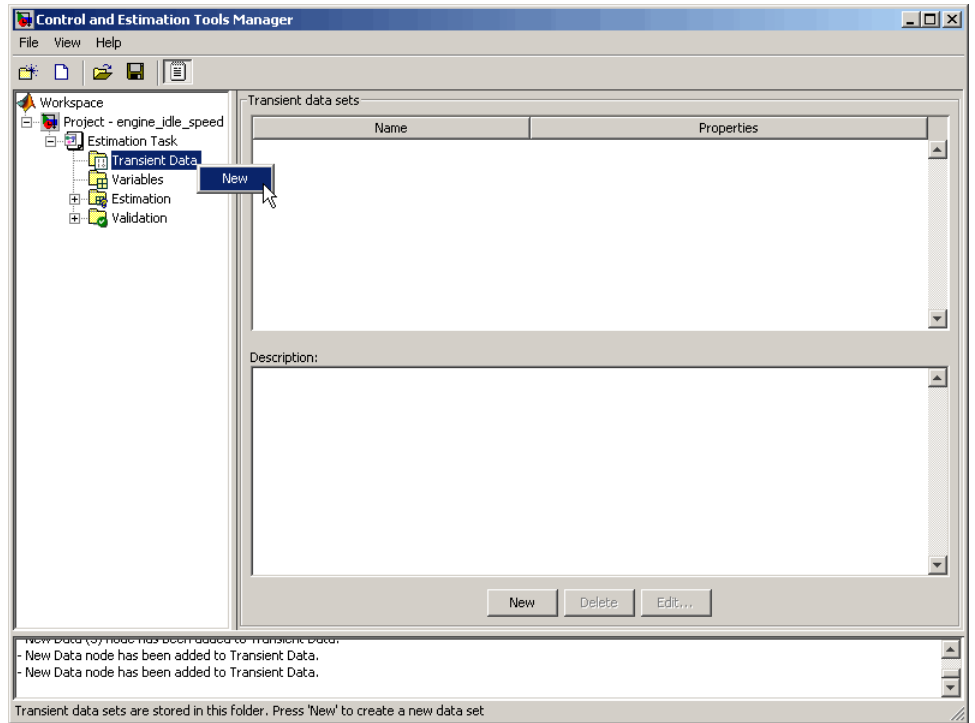


Control and Estimation Tools Manager GUI

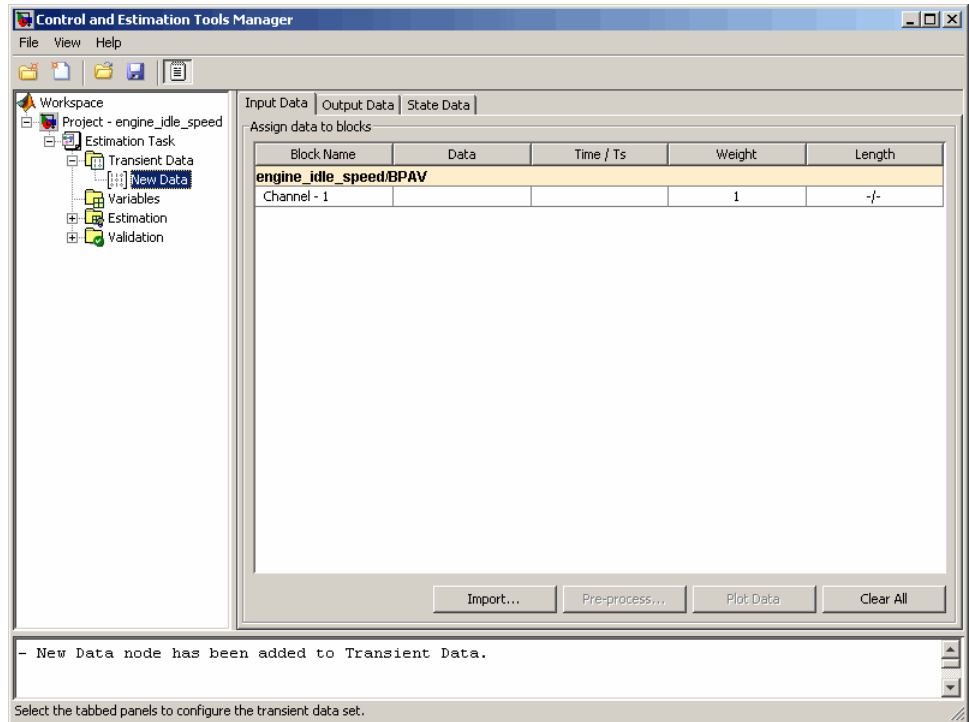
Importing Transient Data

To import *transient* (measured) data for your dynamic system:

- 1 In the Control and Estimation Tools Manager, select **Estimation Task** > **Transient Data** in the workspace directory tree.
- 2 Right-click **Transient Data** and select **New** to create a **New Data** node. Alternatively, you can use the **New** button to create this node.



3 Select the **New Data** node in the workspace directory tree.



Import Data into the Control and Estimation Tools Manager

The table rows in the **Input Data** and **Output Data** tabs correspond to the Inport block BPAV and Outport block Engine Speed, respectively.

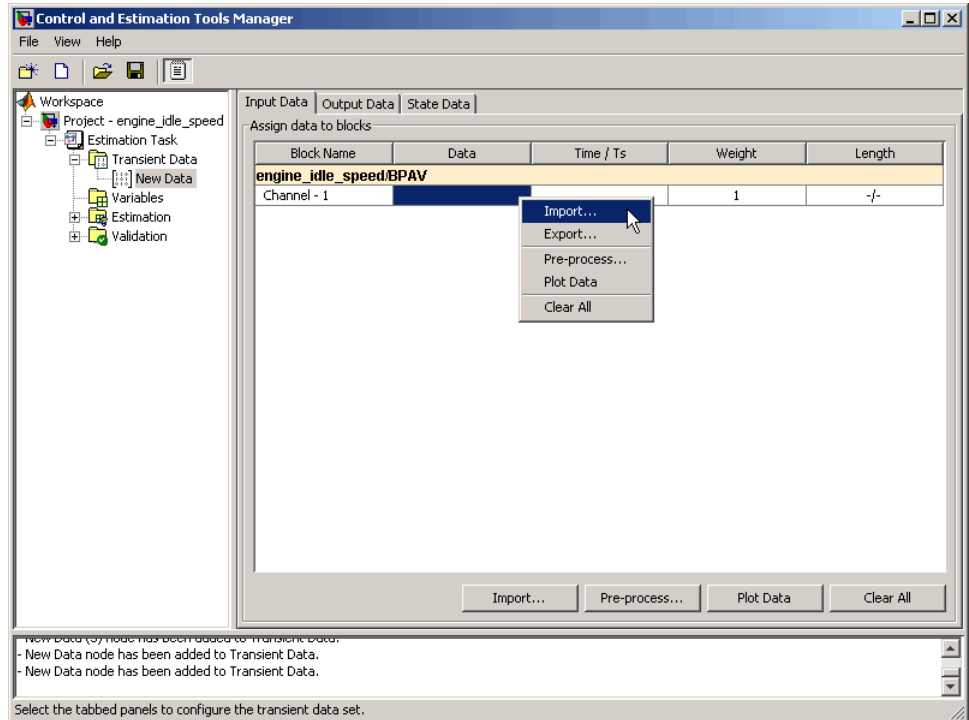
Note The Simulink model must contain an Inport or Outport block or logged signals to enable importing data. For more information, see “Preparing a Model for Parameter Estimation” on page 1-8.

Importing Input Data

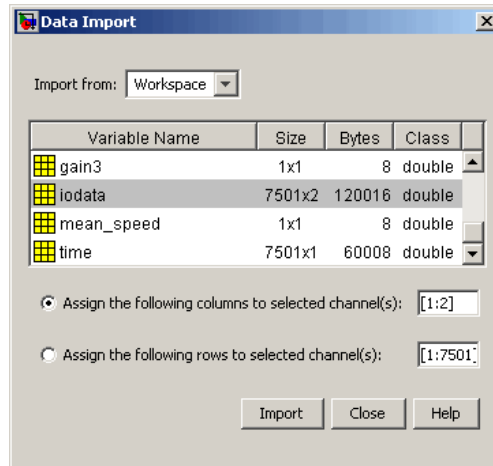
To import the model input data for the input port BPAV:

- 1 In the **New Data** node, click the **Input Data** tab.

- 2 Right-click the **Data** cell and select **Import** to open the Data Import dialog box. Alternatively, you can use the **Import** button to open this dialog box.



- 3 In the Data Import dialog box, select **iodata** from the list of variables.



The idle-speed model of an automotive engine contains the measured data stored in the `iodata` array. The array contains two columns: the first for model input data, and the second for model output data.

- 4** Enter 1 in the **Assign the following columns to selected channel(s)** field, and then click **Import**.
- 5** Click the **Time/Ts** cell, and select `time` in the Data Import dialog box.
- 6** Click **Import** to import the time vector for the input data.
- 7** Click **Close** to close the Data Import dialog box.

Importing Output Data

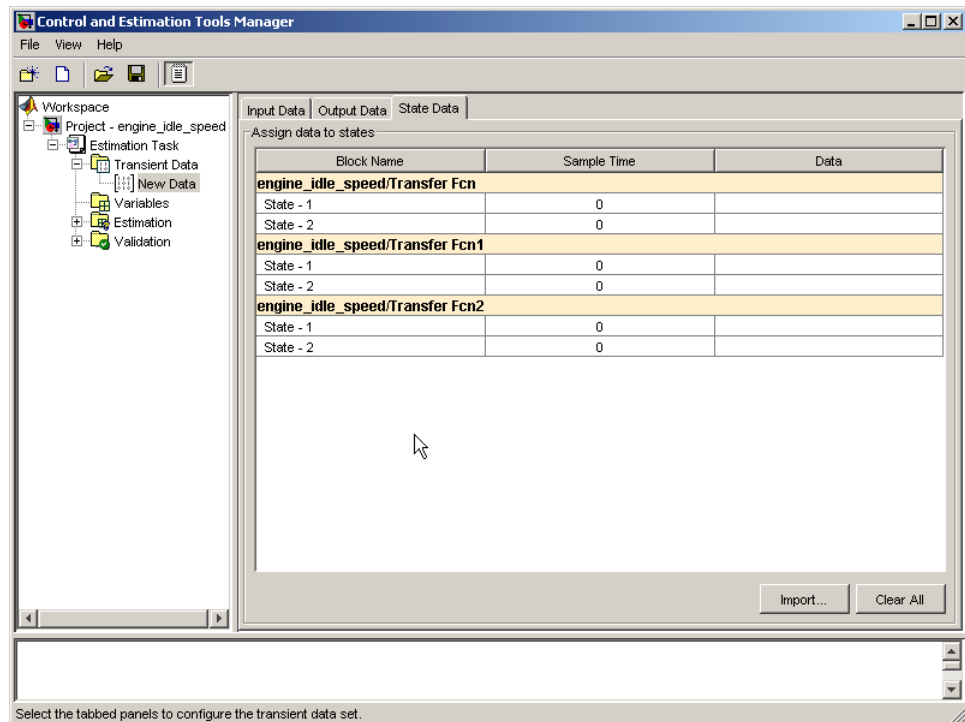
To import the model output data for the output port `Engine Speed`:

- 1** In the **New Data** node, click the **Output Data** tab.
- 2** Right-click the **Data** cell and select **Import** to open the Data Import dialog box.
- 3** Select `iodata` from the list of variables.
- 4** Enter 2 in the **Assign the following columns to selected channel(s)** field to use the second column of `iodata`, and then click **Import**.

- 5 Click the **Time/Ts** cell, and select `time` in the Data Import dialog box.
- 6 Click **Import** to import the `time` vector for the output data.
- 7 Click **Close** to close the Data Import dialog box.

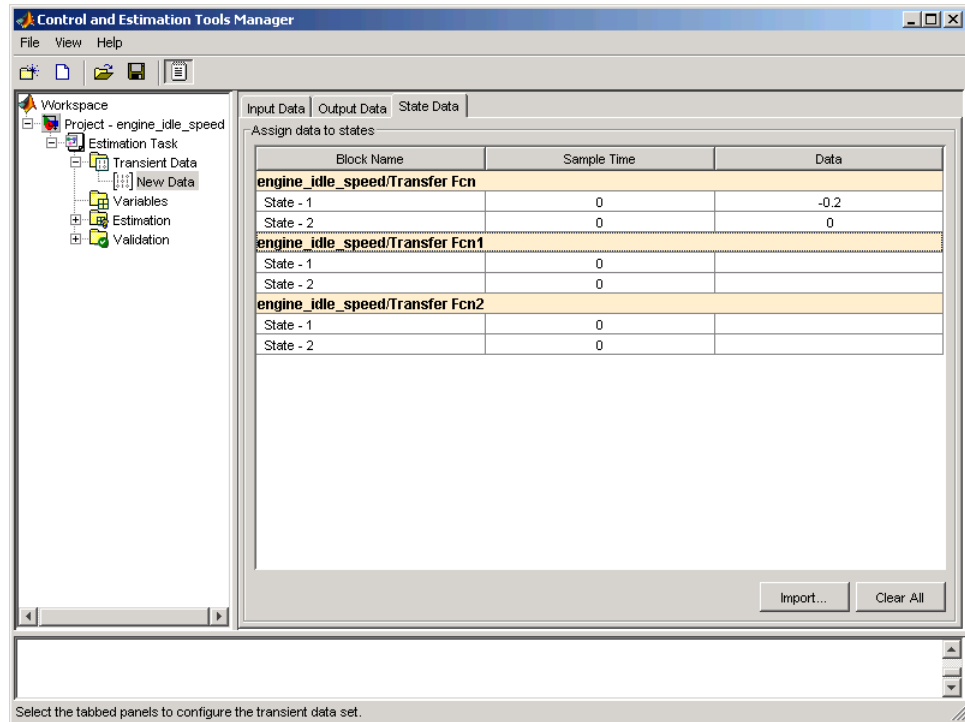
Specifying Initial Conditions

By default, the estimation uses initial conditions specified in the Simulink model. If you want to specify initial conditions other than the defaults, use the State Data tab. You can open it by selecting **Transient Data > New Data** in the workspace directory tree, and then clicking the **State Data** tab.



To specify an initial condition of a state:

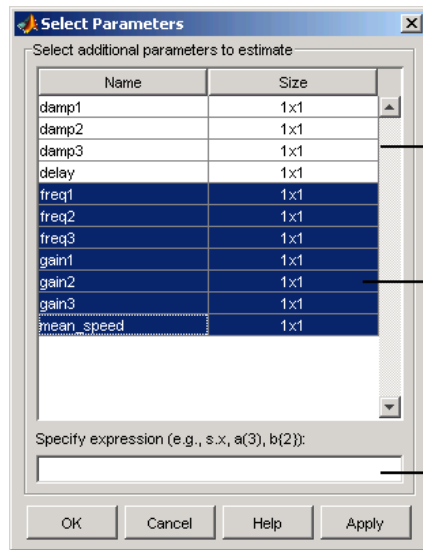
- 1 Select the **Data** cell associated with the state.
- 2 Enter the initial conditions. In this example, enter **-0.2** for **State - 1** of the **engine_idle_speed/Transfer Fcn**. For **State - 2**, enter **0**.



Selecting Parameters for Estimation

To select parameters for estimation:

- 1 In the Control and Estimation Tools Manager, select the **Variables** node in the workspace directory tree to open the **Estimated Parameters** pane.
- 2 In the **Estimated Parameters** pane, click **Add** to open the Select Parameters dialog box.



By default, the Select Parameters dialog box looks at all variables in the model workspace and the MATLAB workspace that are used by the model.

List of parameters

Use your mouse to select data. To select adjacent parameters, hold down the Shift key while clicking the first and last parameter in the selection. To select nonadjacent parameters, hold down the Ctrl key while clicking each parameter.

Use the text field to get the parameters contained in either a Simulink parameter object, MATLAB array, structure, or cell array. Note that you cannot use mathematical expressions such as $x + 5$.

3 Select the last seven parameters: freq1, freq2, freq3, gain1, gain2, gain3, and mean_speed, and then click **OK**.

In general, you can enter parameters, (separated by commas, in the **Specify expression** field. The parameters can be stored in one of the following:

- Simulink software parameter object
Example: For a Simulink parameter object k, type `k.value`.
- Structure
Example: For a structure S, type `S.fieldname` (where *fieldname* represents the name of the field that contains the parameter).
- Cell array
Example: Type `C{1}` to select the first element of the C cell array.
- MATLAB array
Example: Type `a(1:2)` to select the first column of a 2-by-2 array called a.

Note You need not estimate the parameters selected here all at once. You can first select all the parameters that you are interested in, and then later decide which ones to estimate in a particular estimation.

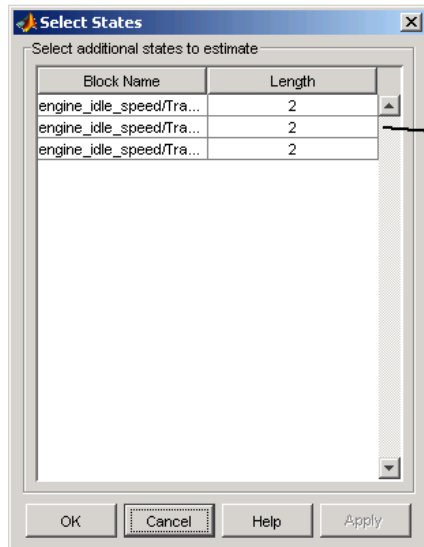
Often, it is more practical to estimate a small group of parameters and use the final estimated values as a starting point for further estimation of parameters that are trickier. Making these sorts of choices involves experience, intuition, and a solid understanding of the strengths and limitations of your Simulink model.

Sometimes models have parameters that are not explicitly defined in the model itself. For example, a gain k could be defined in the MATLAB workspace as $k=a+b$, where a and b are not defined in the model but k is used. To add these independent parameters to the Select Parameters dialog box, see “Estimating Independent Parameters” on page 1-58.

Selecting States for Estimation

In general, you choose to estimate only those states that are not already in the model. To estimate initial conditions (or initial states) if they are not known:

- 1** In the Control and Estimation Tools Manager, select the **Variables** node in the workspace directory tree.
- 2** Click the **Estimated States** tab.
- 3** Click **Add** to open the Select States dialog box.



By default, the Select States dialog box looks at all states of all blocks in the model.

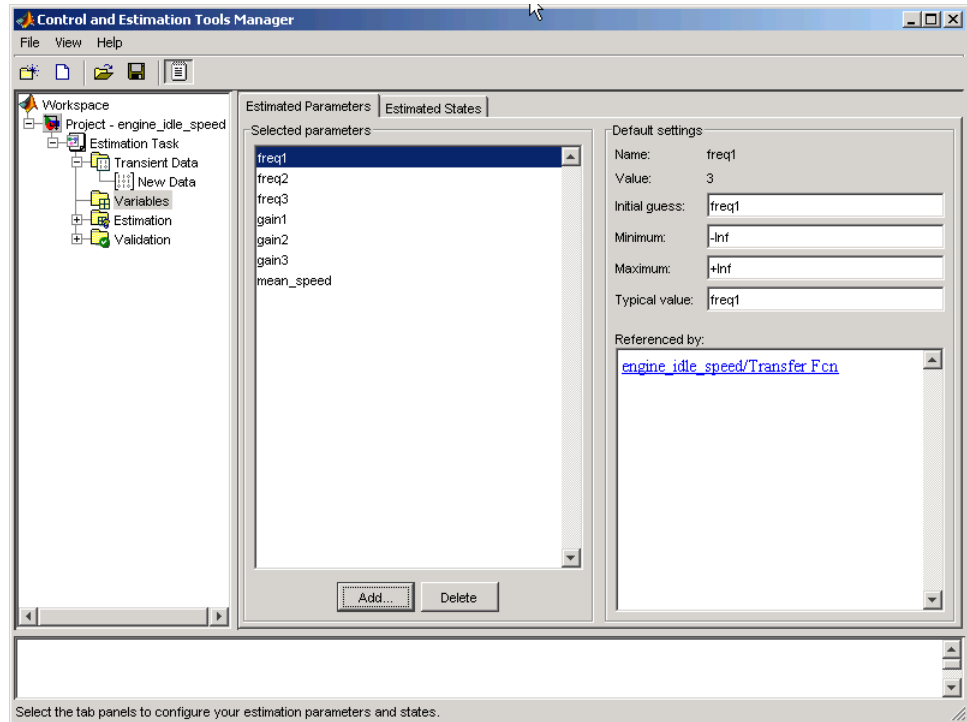
List of states

Use your mouse to select data. To select adjacent states, hold down the Shift key while clicking the first and last state in the selection. To select nonadjacent states, hold down the Ctrl key while clicking each state.

4 Examine the available states but do not select any for this example.

Specifying Initial Guesses and Upper/Lower Bounds

After you select parameters for estimation, the Control and Estimation Tools Manager looks like the following figure.



For each parameter, use the **Default settings** pane to specify the following:

- **Initial guess** — The value the estimation uses to start the process.
- **Minimum** — The smallest allowable parameter value. The default is $-\text{Inf}$.
- **Maximum** — The largest allowable parameter value. The default is $+\text{Inf}$.
- **Typical value** — The average order of magnitude. If you expect your parameter to vary over several orders of magnitude, enter the number that specified the average order of magnitude you expect. For example, if your initial guess is 10, but you expect the parameter to vary between 10 and 1000, enter 100 (the average of the order of magnitudes) for the typical value.

You use the typical value in two ways:

- To scale parameters with radically different orders of magnitude for equal emphasis during the estimation. For example, try to select the typical values so that

$$\frac{\text{anticipated value}}{\text{typical value}} \cong 1$$

or

$$\frac{\text{initial value}}{\text{typical value}} \cong 1$$

- To put more or less emphasis on specific parameters. Use a larger typical value to put more emphasis on a parameter during estimation.

Setting Up an Estimation Project

In this section...

“Creating an Estimation Task” on page 1-23

“Adding Data Sets” on page 1-23

“Specifying and Setting Up Parameters” on page 1-25

“Specifying Settings for Estimation” on page 1-27

Creating an Estimation Task

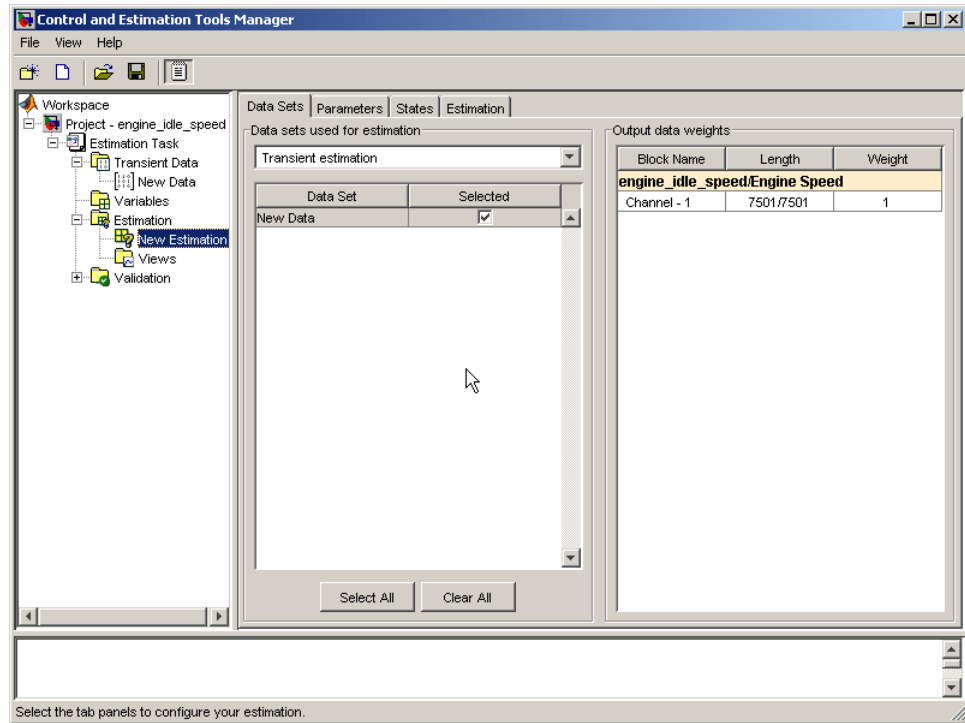
After you import the transient data and select the parameters and any initial conditions (or states) to estimate, you must create an estimation task and configure the estimation settings. To create a container that stores the estimation settings:

- 1 In the Control and Estimation Tools Manager, right-click **Estimation** in the workspace directory tree and select **New**.
- 2 Click the **New Estimation** node.

Adding Data Sets

After you select the **New Estimation** node, the **Data Sets** tab appears. Here you choose the output data from the Simulink model that you want to use in the estimation.

In this example, select the check box to the right of the **New Data** data set.



Note If you imported multiple data sets, you can select them for estimation by selecting the check box to the right of each desired data set. When using several data sets, you increase the estimation precision. However, you also increase the number of required simulations: for N parameters and M data sets, there are $M \cdot (2N + 1)$ simulations per iteration.

Then, specify the weight of each output from this model by setting the **Weight** column in the **Output data weights** table.

The relative weights are used to place more or less emphasis on specific output variables. The following are a few guidelines for specifying weights:

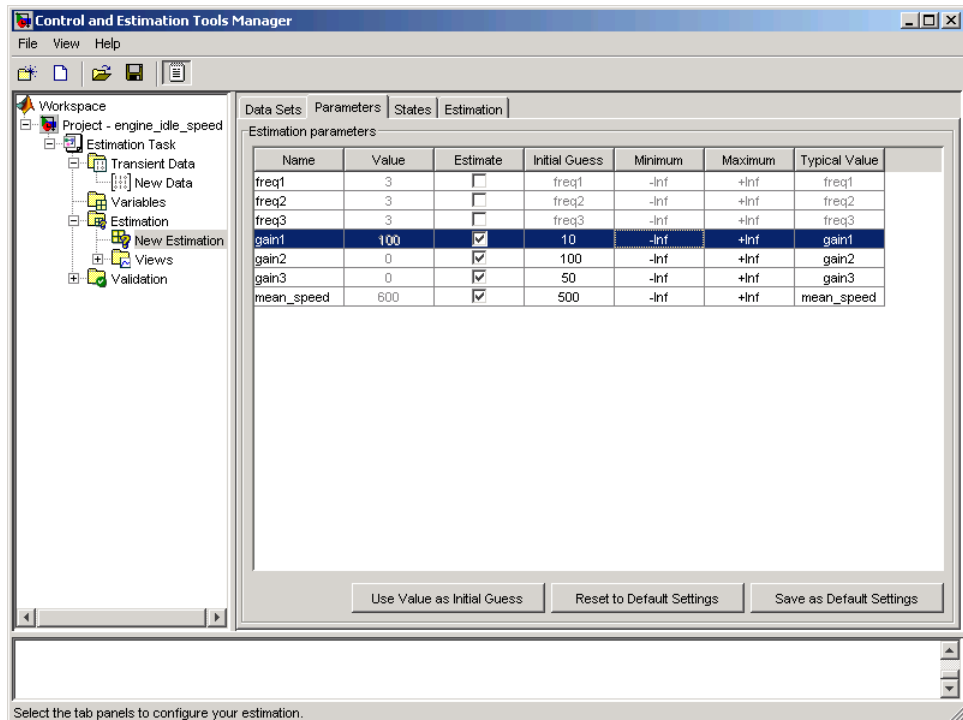
- Use less weight when an output is noisy.
- Use more weight when an output strongly affects parameters.

- Use more weight when it is more important to accurately match this model output to the data.

Specifying and Setting Up Parameters

In the **New Estimation** node in the workspace directory tree, click the **Parameters** tab in the Control and Estimation Tools Manager. In this pane, you select which parameters to estimate and the range of values for the estimation.

Note When you specify the **Minimum** and **Maximum** values for the parameters here, it does not affect your settings in the **Variables** node. You make these choices on a per estimation basis. You can move data to and from the **Variables** node into the **Estimation** node.



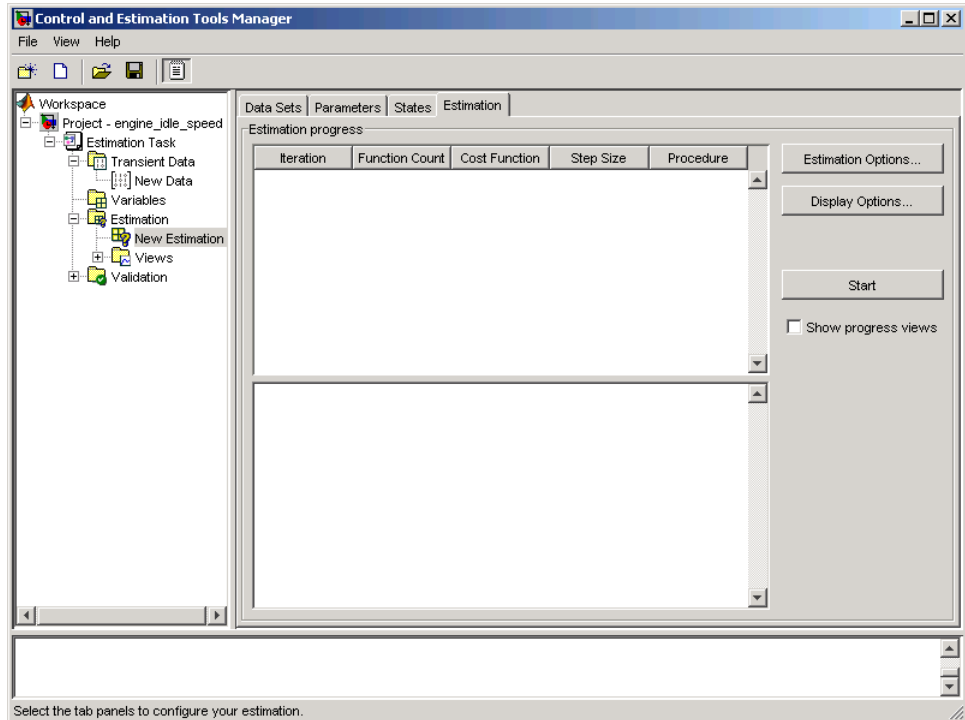
You can select the parameters you want to estimate by selecting the check box in the **Estimate** column. Enter initial values for your parameters in the **Initial Guess** column. The default values in the **Minimum** and **Maximum** columns are `-Inf` and `+Inf`, respectively, but you can select any range you want.

For this example, set `gain1` to 10, `gain2` to 100, `gain3` to 50, and `mean_speed` to 500. Alternatively, use any initial values you like.

If you have good reason to believe a parameter lies within a finite range, it is usually best not to use the default minimum and maximum values. Often, there are computational advantages in specifying finite bounds if you can. It can be very important to specify lower and upper bounds. For example, if a parameter specifies the weight of a part, be sure to specify 0 as the absolute lower bound if better knowledge is unavailable.

Specifying Settings for Estimation

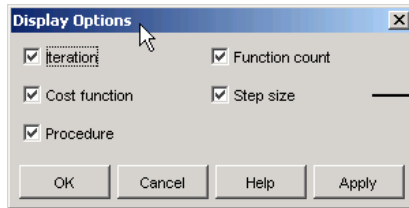
In the **New Estimation** node in the workspace directory tree, click the **Estimation** tab to specify the settings for estimation.



Before you start, you can click **Estimation Options** to specify various algorithms and simulation features. See “Setting Options for Optimization” on page 1-46 for more information.

Display Options

You can specify the display options by clicking **Display Options**. This opens the following dialog box.



By default, all boxes are checked. Uncheck any feature that you don't want to view during the estimation process.

Clearing a check box implies that feature will not appear in the display table as the estimation progresses.

Selecting Views for Plotting

In this section...
“Basic Steps for Creating Plots” on page 1-29
“Types of Plots” on page 1-32

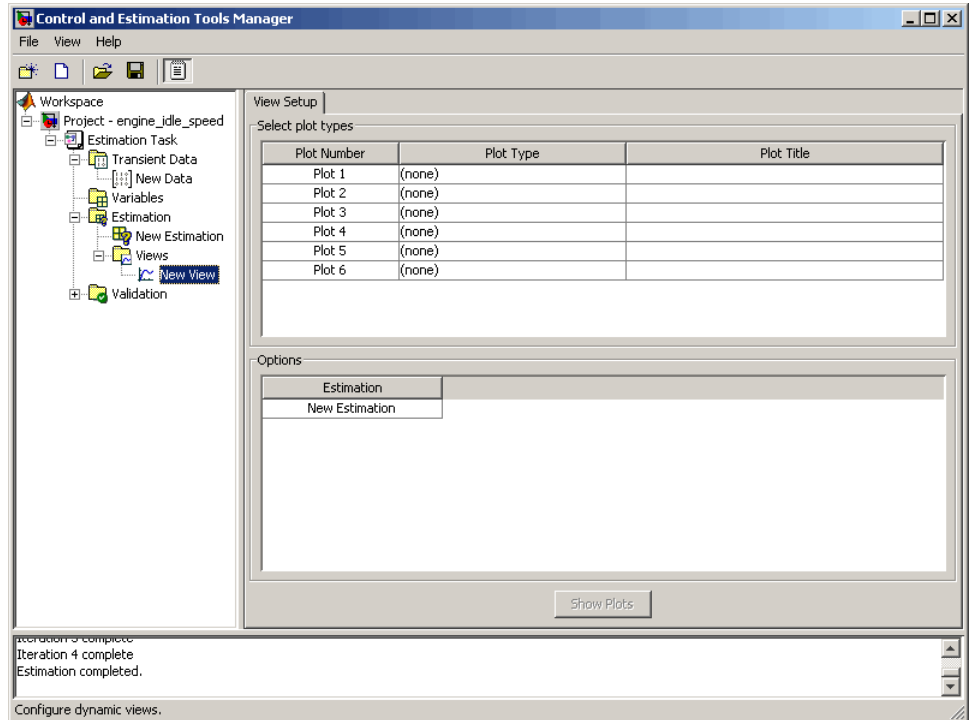
Basic Steps for Creating Plots

Note An estimation must be created before creating views. Otherwise, the **Options** table will be empty.

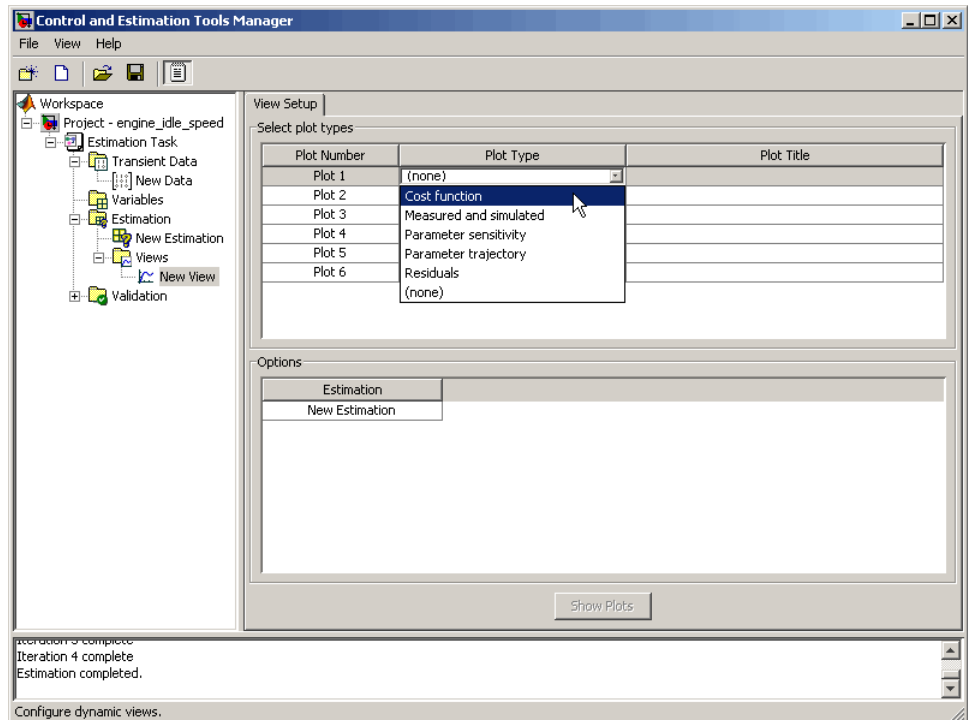
To create plots for viewing the estimation progress, follow the steps below:

- 1 Right-click the **Views** node in the Control and Estimation Tools Manager and select **New**.

- 2 In the workspace directory tree, select **New View** to open the **View Setup** pane.

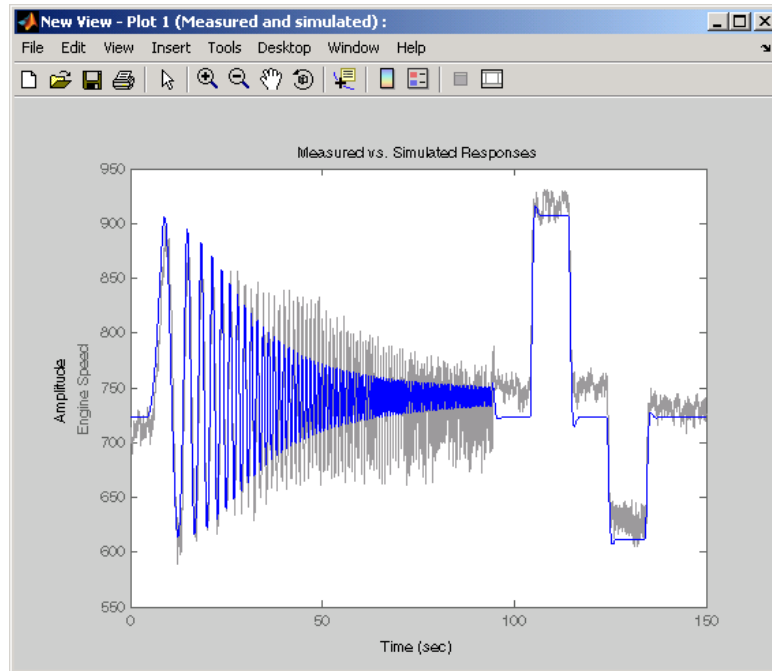


- 3** In the **Select plot types** table, select the **Plot Type** from the drop-down list. In this example, select **Cost function**.



- 4** Select **Measured and simulated** as the **Plot Type** for **Plot 2**. This plot will be used in “Performing Validation” on page 1-39.
- 5** In the **Options** area, select the check-box for both **Plot 1** and **Plot 2**.
- 6** Click **Show Plots**. This displays an empty cost function plot and a plot of the measured data. When you run the estimation, the plot updates automatically.

The next figure shows the plot generated by running the estimation, as described in “Running the Estimation” on page 1-33.



Types of Plots

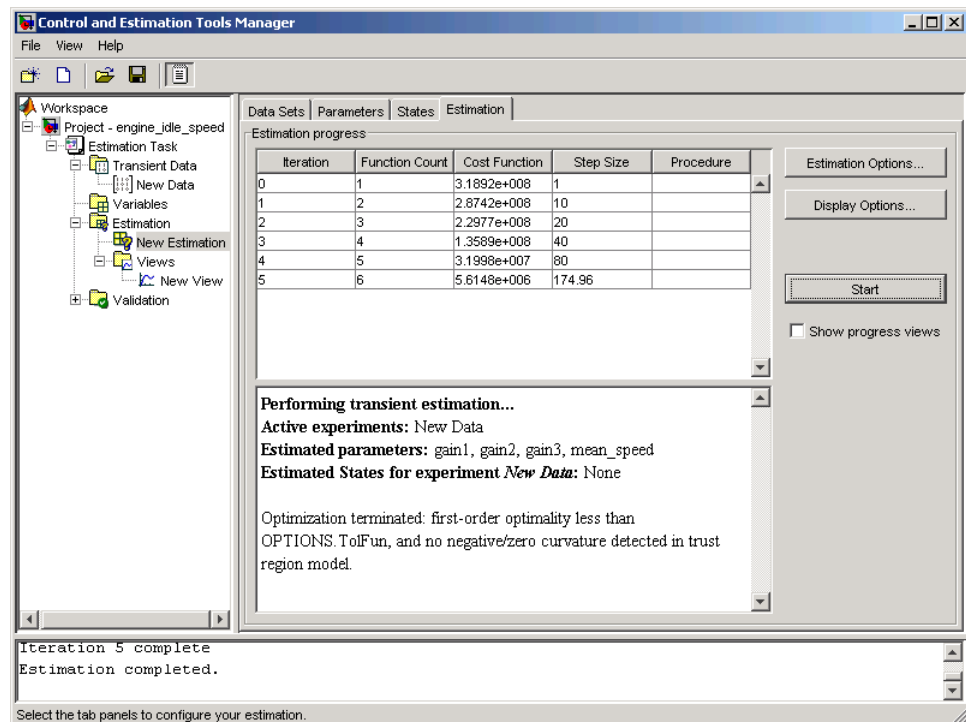
You can choose the plot type from the drop-down list under **Plot Type**. Various types of plots are available, including

- **Cost function** — Plot the cost function values.
- **Measured and simulated** — Plot empirical data against simulated data.
- **Parameter sensitivity** — Plot the rate of change of the cost function as a function of the change in the parameter. That is, plot the derivative of the cost function with respect to the parameter being varied.
- **Parameter trajectory** — Plot the parameter values as they change.
- **Residuals** — Plot the error between the experimental data and the simulated output.

Running the Estimation

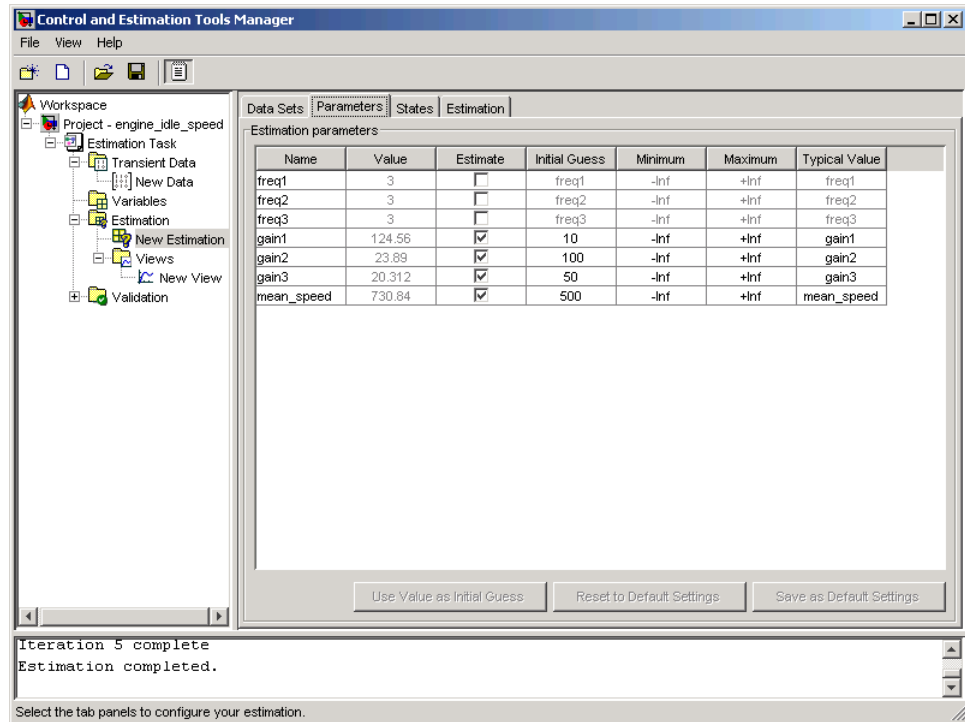
In the Control and Estimation Tools Manager, select the **New Estimation** node and click the **Estimation** tab.

Click **Start** to begin the estimation process. At the end of the iterations, the window should resemble the following:



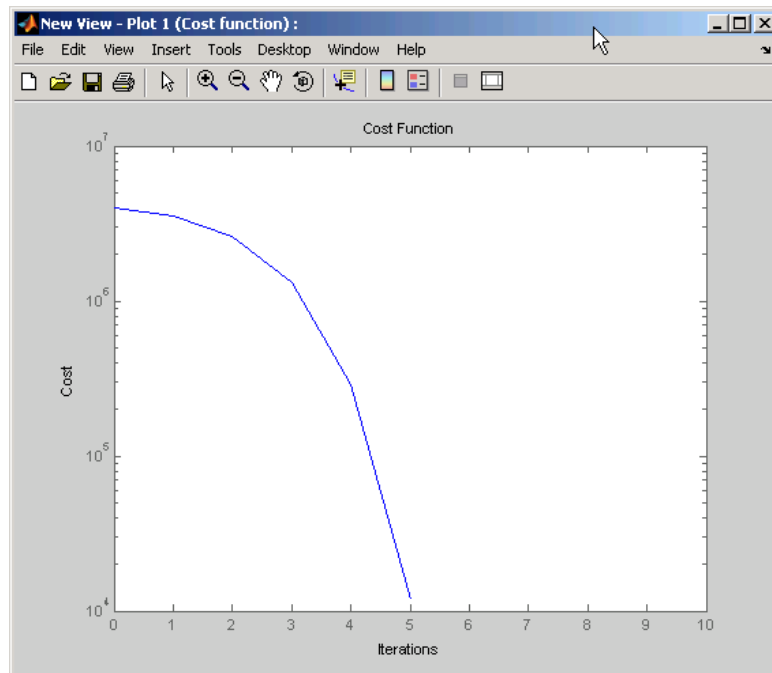
Usually, a lower cost function value indicates a successful estimation, meaning that the experimental data matches the model simulation with the estimated parameters.

The **Estimation** pane displays each iteration of the optimization algorithm. To see the final values for the parameters, click the **Parameters** tab.



The values of these parameters are also updated in the MATLAB workspace. If you specify the variable name in the **Initial Guess** column, you can restart the estimation from where you left off at the end of a previous estimation.

After the estimation process completes, the cost function minimization plot appears as shown in the following figure.



If the optimization went well, you should see your cost function converge on a minimum value. The lower the cost, the more successful is the estimation.

Model Validation

In this section...

“Basic Steps for Model Validation” on page 1-36

“Example: Validating the Engine Idle Speed Model” on page 1-37

“Loading and Importing the Validation Data” on page 1-38

“Performing Validation” on page 1-39

“Comparing Residuals” on page 1-43

Basic Steps for Model Validation

After you complete estimating the parameters, you can validate your results against another set of data.

These are the basic steps needed to validate a model using the Control and Estimation Tools Manager:

- 1 Add the validation data set to the **Transient Data** node.
- 2 Add a new validation task in the **Validation** node in the workspace directory tree.
- 3 Configure the validation settings by selecting the plot types and the validation data set from the **Validation Setup** pane.
- 4 Click **Show Plots** in the **Validation Setup** pane and view the results in the plot window.
- 5 Compare the validation plots to the corresponding view plots to see if they match.

The basic difference between the validation and views features is that you can run validation after the estimation is complete. All views should be set up before an estimation, and you can watch the views update in real time. Validations can use other validation data sets for comparison with the model response. Also, validations appear after you have completed an estimation and do not update.

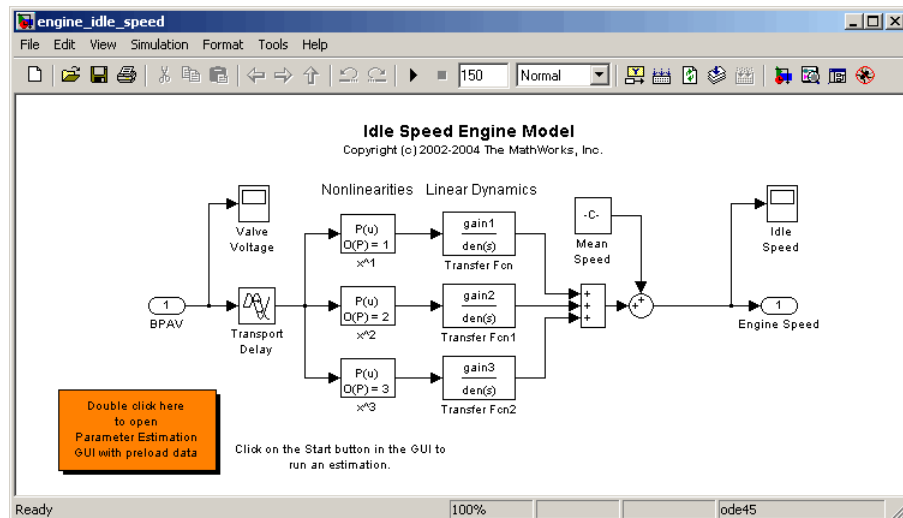
You can validate your data by comparing measured vs. simulated data for your estimation data and validation data sets. Also, it is often useful to compare residuals in the same way.

Example: Validating the Engine Idle Speed Model

If you have not run the engine idle speed demo, type:

```
engine_idle_speed
```

at the MATLAB prompt and run the estimation. For more information on how to run the estimation, see “How Simulink® Parameter Estimation Software Works” on page 1-5. You can double-click the box in the upper-left corner of the model to import data and populate the required fields in the Control and Estimation Tools Manager.



engine_idle_speed Simulink® Model

Now that the estimation data is loaded, and the estimation task has been created, the next step is to import validation data into the Control and Estimation Tools Manager.

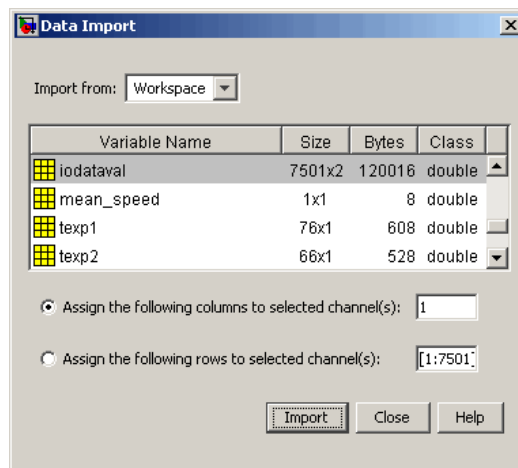
Loading and Importing the Validation Data

To load the validation data, type

```
load iodataval
```

at the MATLAB prompt. This loads the data into the MATLAB workspace. The next step is to import this data into the Control and Estimation Tools Manager. See “Importing Transient Data” on page 1-11 for information on importing data, but the quickest way is to follow these steps:

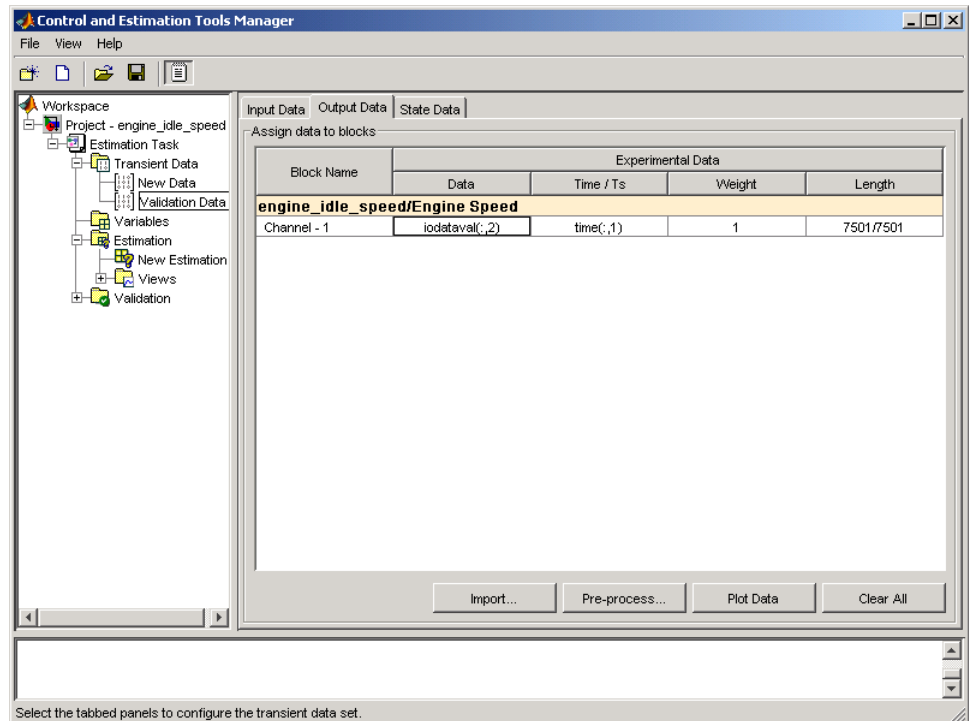
- 1 Right-click the **Transient Data** node in the workspace directory tree in the Control and Estimation Tools Manager and select **New**.
- 2 Select **New Data (2)** from the **Transient Data** pane.
- 3 Right-click the **New Data (2)** node in the workspace directory tree and select **Rename**. Change the name of the data to **Validation Data**.
- 4 In the **Input Data** pane, select the **Data** cell associated with Channel 1 and click **Import**. In the Data Import dialog box, select `iodataval` and assign column 1 to the selected channel by entering 1 in the **Assign columns** field. Click **Import** to import the input data.



- 5 Select the **Time/Ts** cell and import time using the Data Import dialog box.

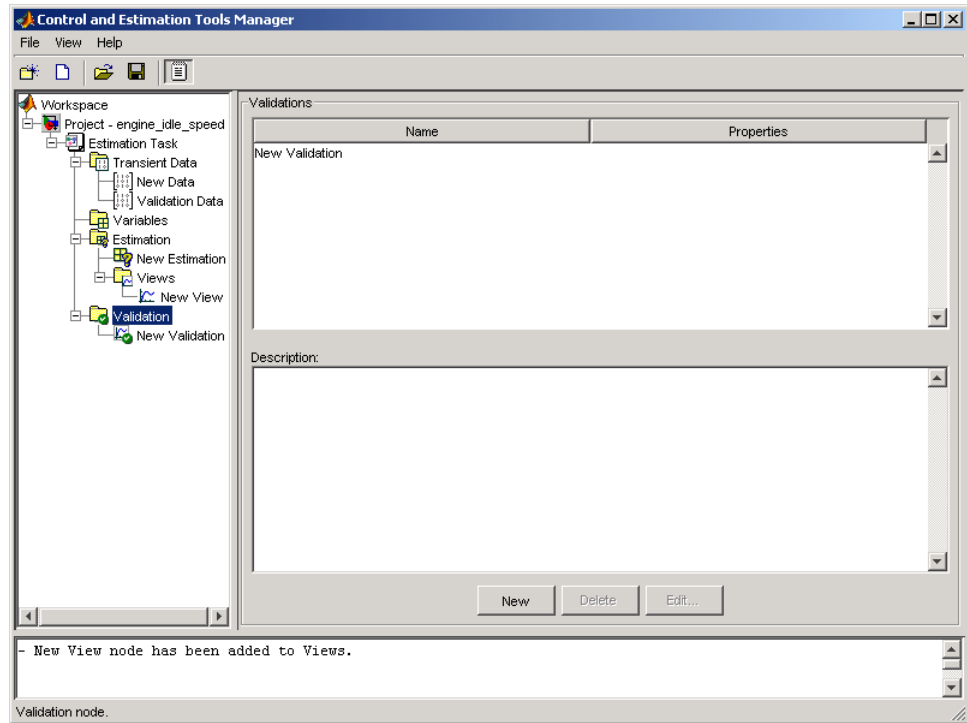
- 6 Similarly, in the **Output Data** pane, select **Time/Ts** and import time.
- 7 In the **Output Data** pane, select the **Data** cell associated with Channel - 1 and click **Import**. Import the second column of data in `iodataval` by selecting it from the list in the Import Data dialog box and entering 2 in the **Assign columns** field. Click **Import** to import the output data.

Your Control and Estimation Tools Manager should resemble this figure.



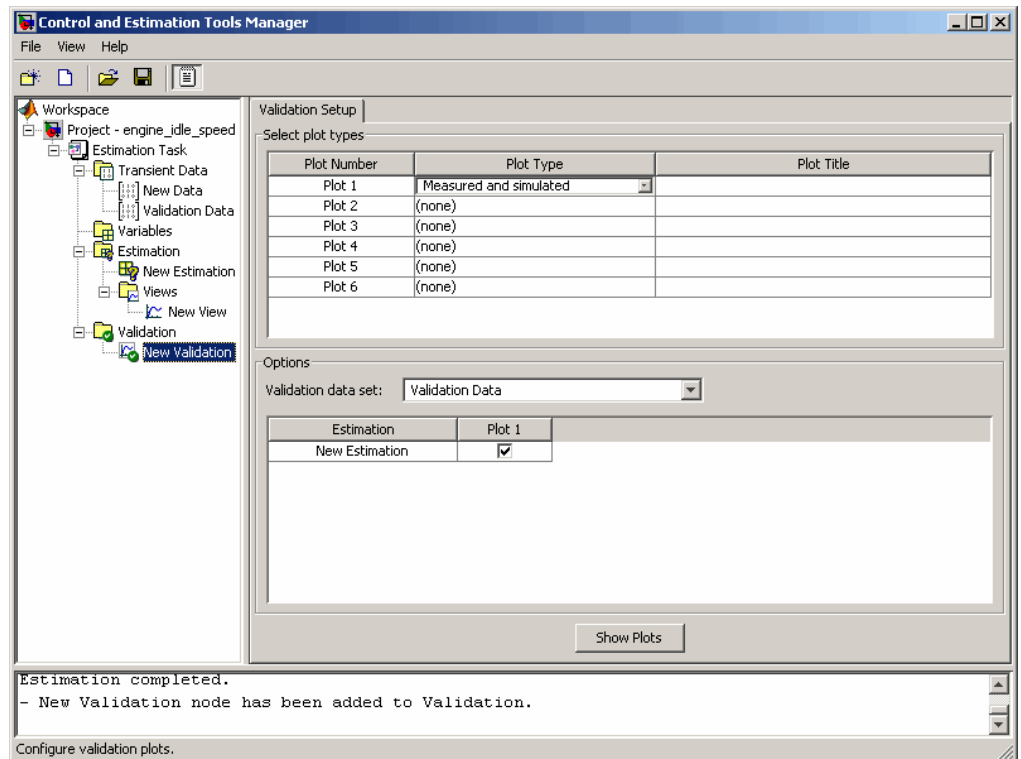
Performing Validation

After you import the data, right-click the **Validation** node and select **New**. This creates a **New Validation** node in the Control and Estimation Tools Manager.

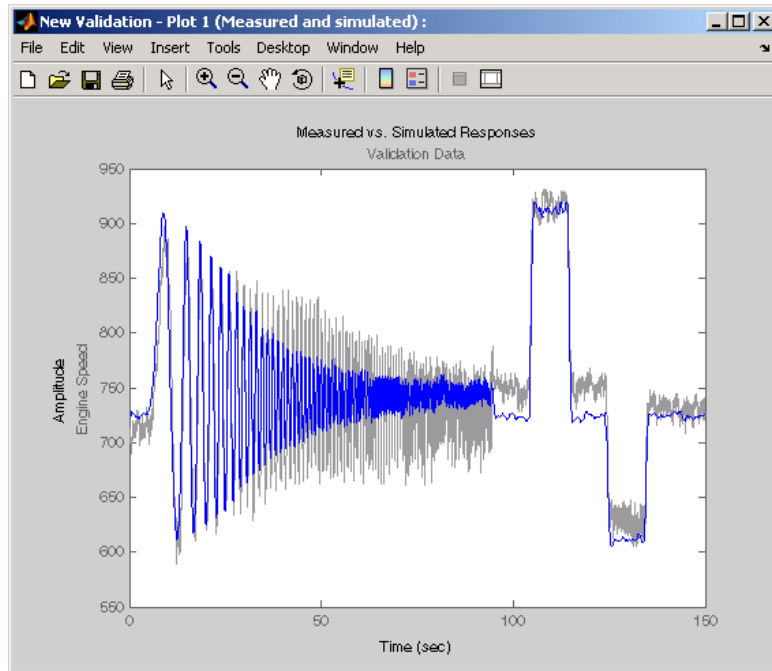


To perform the validation:

- 1 Select **New Validation** in the workspace directory tree to open the **Validation Setup** pane.

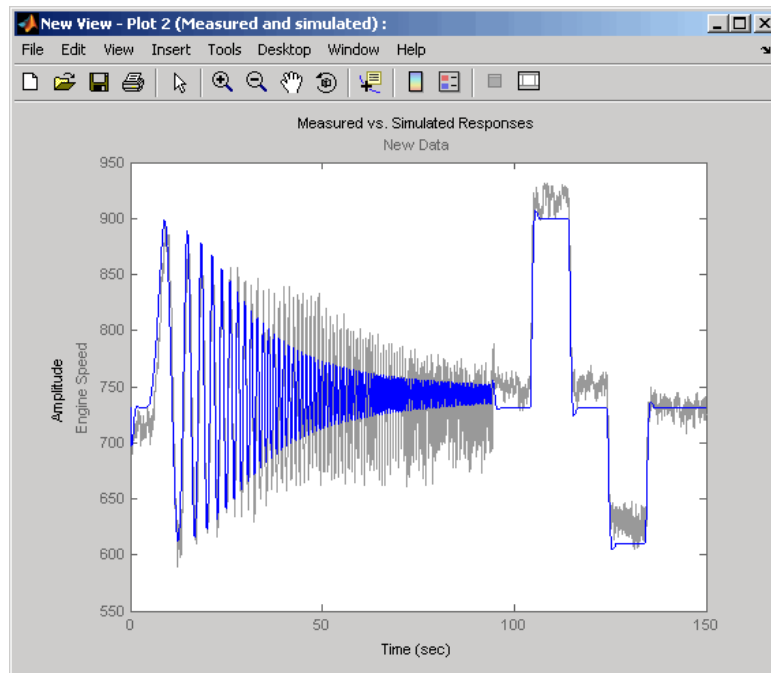


- 2** Click the **Plot Type** cell for Plot 1 and select Measured and simulated from the drop-down menu.
- 3** In the **Options** area, select Validation Data in the **Validation data set** drop-down list.
- 4** Click **Show Plots** to open a plot figure window as shown next.



Measured Versus Simulated Data Plot for Validation Data

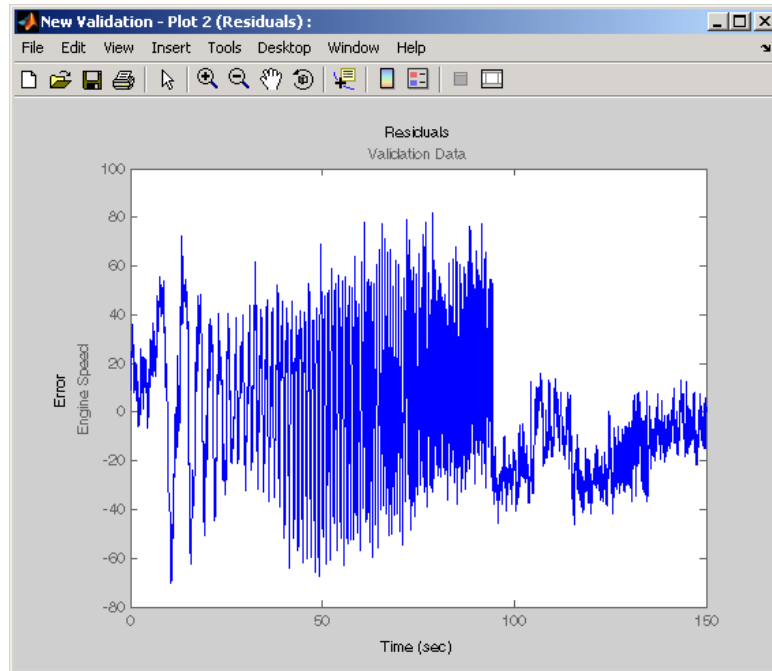
- 5 Compare this plot with the plot of Measured and simulated data for the validation data. For more information on how to create this plot, see “Selecting Views for Plotting” on page 1-29.



Measured and Simulated Data Views Plot

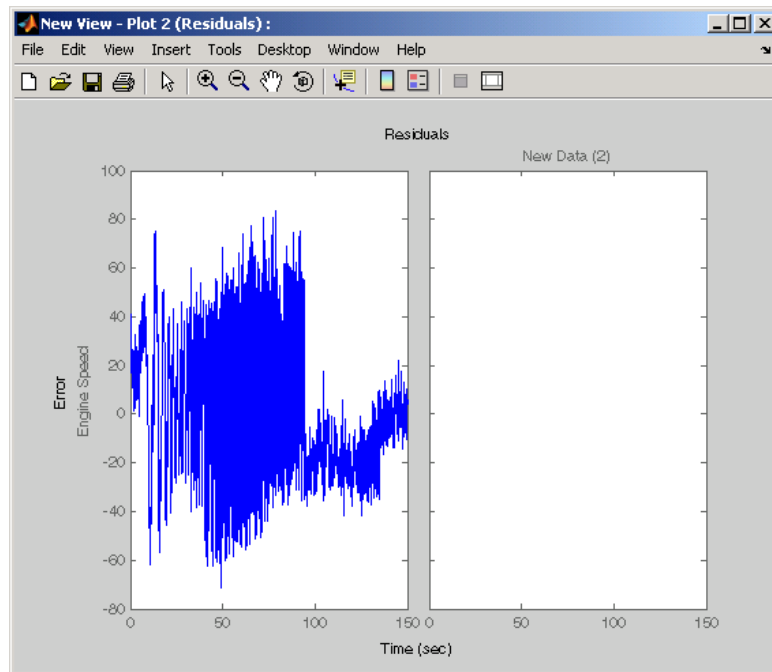
Comparing Residuals

To look at the residuals, select **Residuals** as the **Plot Type** for **Plot 2** in the **New Validation** pane. In the **Options** area, select the **Plot 2** check box and click **Show Plots**. The following figure shows the resulting residuals plot.



Plot of Residuals Using the Validation Data

Compare the validation data residuals with the original data set residuals from the **Views** node in the workspace directory tree. To create the plot of residuals for the original data set, select the **New View** node and choose **Residuals** as the **Plot Type**.



Plot of Residuals Using the Test Data

The plot on the left agrees with the plot of the residuals for the validation data. The right side has no plot because residuals were not calculated for the validation data during the original estimation process.

Setting Options for Optimization

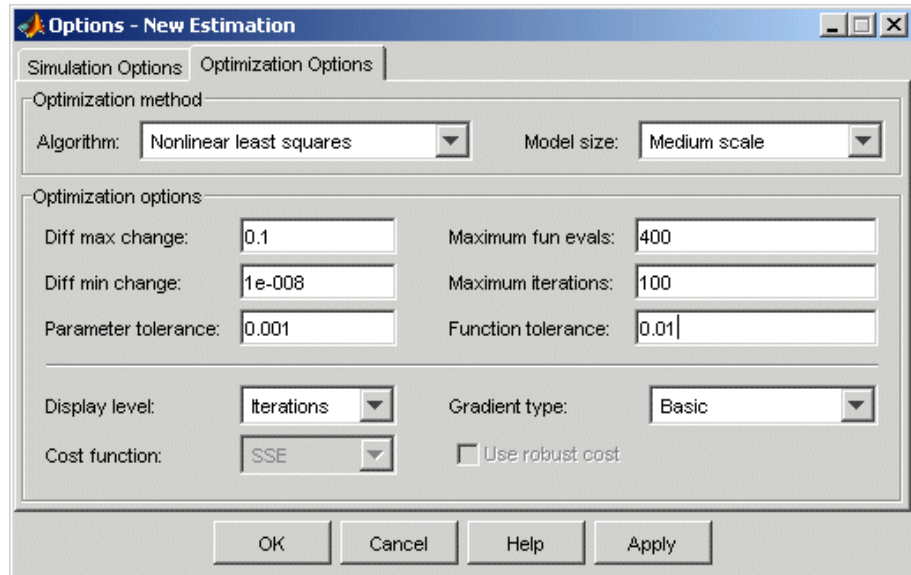
In this section...
“Tuning the Results of Optimization” on page 1-46
“Selecting Optimization Methods” on page 1-47
“Selecting Optimization Termination Options” on page 1-48
“Selecting Additional Optimization Options” on page 1-49
“Specifying the Cost Function” on page 1-49

Tuning the Results of Optimization

You can set several options to tune the results of the optimization. These options include the optimization algorithms and their tolerances.

To set options for optimization:

- 1 Select the **New Estimation** node in the workspace directory tree.
- 2 Click the **Estimation** tab.
- 3 Click **Estimation Options** to open the Options dialog box.



- 4** Click the **Optimization Options** tab and specify the options, as described in the following sections:

Selecting Optimization Methods

Both the algorithm and model size define the optimization method. Use the **Optimization method** area in the Options dialog box to set algorithm and the model size.



For the **Algorithm** parameter, the four options are

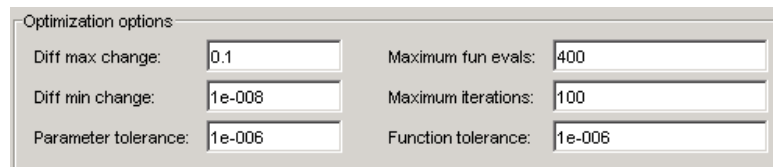
- **Gradient descent** — Uses the Optimization Toolbox function `fmincon` to optimize the response signal subject to the constraints
- **Nonlinear least squares** — Uses a nonlinear least squares, `lsqnonlin`, optimization algorithm.
- **Pattern search** — Uses an advanced pattern search algorithm. This option requires Genetic Algorithm and Direct Search Toolbox™ software.

- **Simplex search** — Uses the Optimization Toolbox function `fminsearch`, which is a direct search method to optimize the response. **Simplex search** is most useful for simple problems and is sometimes faster than **Function minimization** for models that contain discontinuities.

For **Model size**, the two options are **Large scale** and **Medium scale**. By default, **Model size** is set to **Large scale**. The **Large scale** methods will always run faster, if there is some sparsity structure in the Jacobian (or Hessian) that can be taken advantage of. To learn more, see “Large-Scale vs. Medium-Scale Algorithms” in the Optimization Toolbox documentation.

Selecting Optimization Termination Options

Specify termination options in the **Optimization options** area.



The screenshot shows a dialog box titled "Optimization options" with six input fields arranged in two columns. The left column contains "Diff max change" (0.1), "Diff min change" (1e-008), and "Parameter tolerance" (1e-006). The right column contains "Maximum fun evals" (400), "Maximum iterations" (100), and "Function tolerance" (1e-006).

Option	Value
Diff max change	0.1
Diff min change	1e-008
Parameter tolerance	1e-006
Maximum fun evals	400
Maximum iterations	100
Function tolerance	1e-006

Several options define when the optimization terminates:

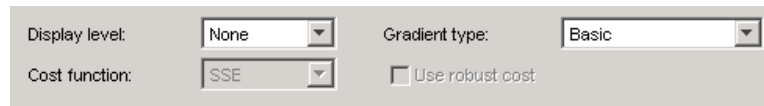
- **Diff max change** — The maximum allowable change in variables for finite-difference derivatives. See `fmincon` in the Optimization Toolbox documentation for details.
- **Diff min change** — The minimum allowable change in variables for finite-difference derivatives. See `fmincon` in the Optimization Toolbox documentation for details.
- **Parameter tolerance** — Optimization terminates when successive parameter values change by less than this number.
- **Maximum fun evals** — The maximum number of cost function evaluations allowed. The optimization terminates when the number of function evaluations exceeds this value.
- **Maximum iterations** — The maximum number of iterations allowed. The optimization terminates when the number of iterations exceeds this value.

- **Function tolerance** — The optimization terminates when successive function values are less than this value.

By varying these parameters, you can force the optimization to continue searching for a solution or to continue searching for a more accurate solution.

Selecting Additional Optimization Options

At the bottom of the **Optimization options** pane is a group of additional optimization options.



The screenshot shows a control panel with the following settings:

- Display level: None
- Gradient type: Basic
- Cost function: SSE
- Use robust cost:

Additional options for optimization include

- **Display level** — Specifies the form of the output that appears in the MATLAB command window. The options are **Iteration**, which displays information after each iteration, **None**, which turns off all output, **Notify**, which displays output only if the function does not converge, and **Final**, which only displays the final output. Refer to the Optimization Toolbox documentation for more information on what type of iterative output each algorithm displays.
- **Gradient type** — When using Gradient Descent or Nonlinear least squares as the **Algorithm**, the gradients are calculated based on finite difference methods. The **Refined** method offers a more robust and less noisy gradient calculation method than **Basic**, although it does take longer to run optimizations using the **Refined** method.

Specifying the Cost Function

The *cost function* is a function that optimization algorithms attempt to minimize. You have the following options when selecting a cost function:

- **Cost function** — The default is SSE (sum of squared errors), which uses a least-squares approach. You can also use SAE, the sum of absolute errors.

- **Use robust cost** — Makes the optimizer use a robust cost function instead of the default least-squares cost. This is useful if the experimental data has many outliers, or if your data is noisy.

Setting Options for the Simulation

In this section...

“Specifying Simulation Options” on page 1-51

“Selecting Simulation Time” on page 1-52

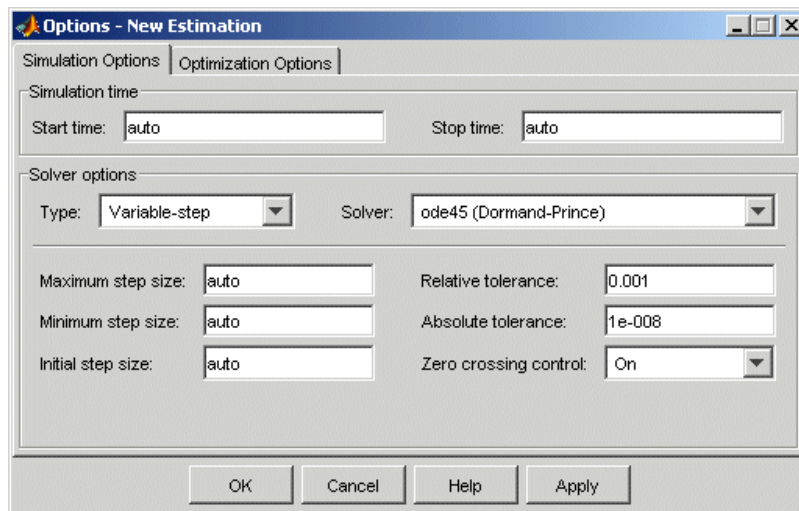
“Selecting Solvers” on page 1-53

Specifying Simulation Options

To optimize the response signals of a model, Simulink Parameter Estimation software runs simulations of the model.

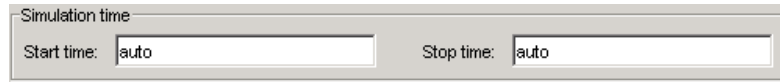
To set options for simulation:

- 1** Select the **New Estimation** node in the workspace directory tree.
- 2** Click the **Estimation** tab.
- 3** Click **Estimation Options** to open the Options dialog box.



- 4 Click the **Simulation Options** tab and specify the options, as described in the following sections.

Selecting Simulation Time



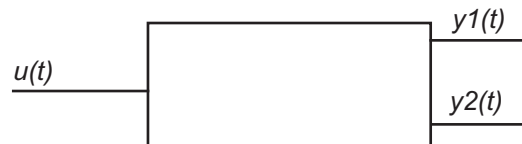
By default, **Start time** and **Stop time** are automatically computed based on the start and stop times specified in the Simulink model.

To set alternative start and stop times for the optimization, enter them under **Simulation time**. This action overwrites the simulation start and stop times specified in the Simulink model.

Simulation Time for Data Sets with Different Time Lengths

Simulink Parameter Estimation software can simulate models containing empirical data sets of different time lengths. You can use experimental data sets for estimation that contain I/O samples collected at different time points.

The following example shows a single-input, two-output model for which you want to estimate the parameters.



The model uses two output data sets containing transient data samples for parameter estimation:

- Output $y1(t)$ at time points $t1 = \{t_1^1, t_2^1, \dots, t_n^1\}$.
- Output $y2(t)$ at time points $t2 = \{t_1^2, t_2^2, \dots, t_m^2\}$.

The simulation time t is computed as:

$$t = t1 \cup t2 = \{t_1^1, t_1^2, t_2^1, t_2^2, \dots, t_n^1, t_m^2\}$$

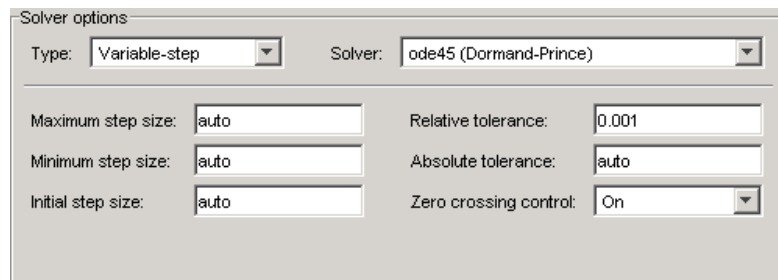
This new set ranges from t_{min} to t_{max} . The values t_{min} and t_{max} represent the minimum and maximum time points in t respectively.

When you run the estimation, the model is simulated over the time range t . Simulink extracts the simulated data for each output based on the following criteria:

- **Start time** — Typically, the start time in the Simulink model is set to 0. For a nonzero start time, the simulated data corresponding to time points before t_1^1 for $y1(t)$ and t_1^2 for $y2(t)$ are discarded.
- **Stop time** — If the stop time $t_{stop} \geq t_{max}$, the simulated data corresponding to time points in $t1$ are extracted for $y1(t)$. Similarly, the simulated data for time points in $t2$ are extracted for $y2(t)$.

If the stop time $t_{stop} < t_{max}$, the data spanning time points $> t_{stop}$ are discarded for both $y1(t)$ and $y2(t)$.

Selecting Solvers



The screenshot shows the 'Solver options' dialog box. At the top, 'Type' is set to 'Variable-step' and 'Solver' is set to 'ode45 (Dormand-Prince)'. Below this, there are two columns of settings. The left column contains 'Maximum step size' (set to 'auto'), 'Minimum step size' (set to 'auto'), and 'Initial step size' (set to 'auto'). The right column contains 'Relative tolerance' (set to '0.001'), 'Absolute tolerance' (set to 'auto'), and 'Zero crossing control' (set to 'On').

When running the simulation, the dynamic system is solved using one of several Simulink solvers. You can specify several solver options using the **Solver options** area in the Options dialog box. The **Type** of solver can be **variable-step** or **fixed-step**. Variable-step solvers keep the error within specified tolerances by adjusting the step-size the solver uses. Fixed-step solvers use a constant step-size. When your model's states are likely to vary

rapidly, a variable-step solver is often faster. See the Simulink documentation for information about solvers.

Variable-Step Solvers

When you select `Variable-step` as the solver **Type**, you can choose any of the following as the **Solver**:

- `discrete` (no continuous states)
- `ode45` (Dormand-Prince)
- `ode23` (Bogacki-Shampine)
- `ode113` (Adams)
- `ode15s` (stiff/NDF)
- `ode23s` (stiff/Mod. Rosenbrock)
- `ode23t` (Mod. stiff/Trapezoidal)
- `ode23tb` (stiff/TR-BDF2)

Variable-Step Solver Options

When you select `Variable-step` as the Simulink solver **Type**, you can also set several other parameters that affect the step-size of the simulation:

- **Maximum step size** — The largest step-size the solver can use during a simulation.
- **Minimum step size** — The smallest step-size the solver can use during a simulation.
- **Initial step size** — The step-size the solver uses to begin the simulation.
- **Relative tolerance** — The largest allowable relative error at any step in the simulation.
- **Absolute tolerance** — The largest allowable absolute error at any step in the simulation.
- **Zero crossing control** — Set to on for the solver to compute exactly where the signal crosses the x -axis. This is useful when using functions

that are nonsmooth and the output depends on when a signal crosses the x -axis, such as absolute values.

By default, the values for these options are automatically chosen. To choose your own values, enter them in the appropriate fields.

Fixed-Step Solvers

When you select **Fixed-step** as the solver **Type**, you can choose any of the following as the **Solver**:

- discrete (no continuous states)
- ode5 (Dormand-Prince)
- ode4 (Runge-Kutta)
- ode3 (Bogacki-Shampine)
- ode2 (Heun)
- ode1 (Euler)

When you select **Fixed-step** as the solver **Type**, you can also set **Fixed step size**, which determines the step-size the solver uses during the simulation. By default, a value for this option is automatically chosen.

Accelerating Model Simulations During Estimation

In this section...

“About Accelerating Model Simulations During Estimation” on page 1-56

“Limitations” on page 1-56

“Setting the Accelerator Mode for Parameter Estimation” on page 1-56

About Accelerating Model Simulations During Estimation

You can accelerate the parameter estimation computations by changing the simulation mode of your Simulink model. Simulink Parameter Estimation software supports `Normal` and `Accelerator` simulation modes. For more information about these modes, see “Accelerating Models” in the Simulink documentation.

The default simulation mode is `Normal`. In this mode, Simulink software uses interpreted code, rather than compiled C code during simulations.

In the `Accelerator` mode, Simulink Parameter Estimation software runs simulations during estimation with compiled C code. Using compiled C code speeds up the simulations and reduces the time to estimate parameters.

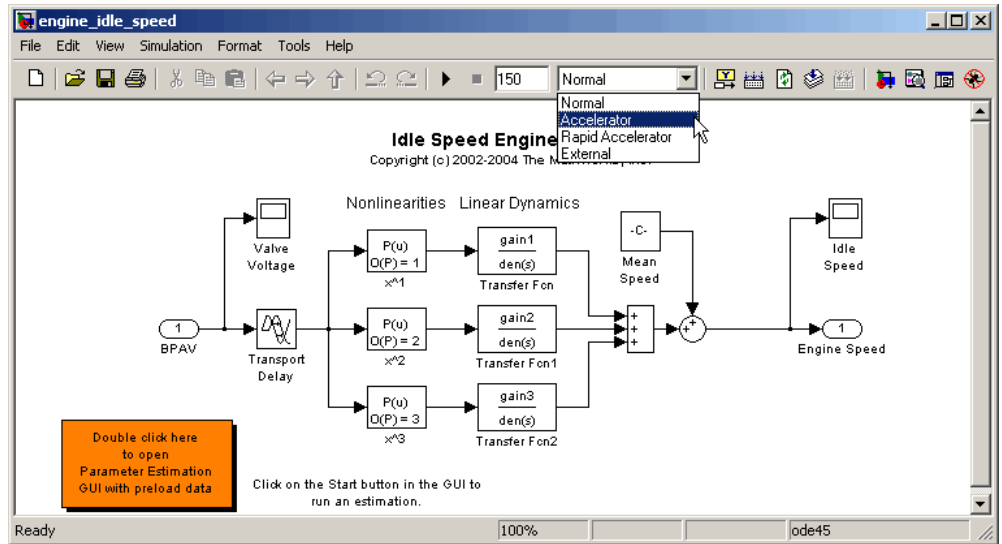
Limitations

You cannot use the `Accelerator` mode if your model contains algebraic loops. If the model contains MATLAB function blocks, you must either remove them or replace them with Fcn blocks.

Setting the Accelerator Mode for Parameter Estimation

To set the simulation mode to `Accelerator`, open the Simulink model window and perform one of the following actions:

- Select **Simulation > Accelerator**.
- Choose `Accelerator` from the drop-down list as shown in the next figure.



Tip To obtain the maximum performance from the Accelerator mode, close all Scope blocks in your model.

Estimating Independent Parameters

In this section...
“Basic Steps for Estimating Independent Parameters” on page 1-58
“Example: Estimating Independent Parameters” on page 1-58

Basic Steps for Estimating Independent Parameters

Sometimes parameters in your model depend on independent parameters that do not appear in the model. The following steps give an overview of how to estimate independent parameters:

- 1** Add the independent parameters to the model workspace (along with initial values).
- 2** Define a Simulation Start function that runs before each simulation of the model. This Simulation Start function defines the relationship between the dependent parameters in the model and the independent parameters in the model workspace.
- 3** The independent parameters now appear in the Select Parameters dialog box. Add these parameters to the list of parameters to be estimated.

Caution Avoid adding independent parameters together with their corresponding dependent parameters to the lists of parameters to be estimated. Otherwise the estimation could give incorrect results. For example, when a parameter x depends on the parameters a and b , avoid adding all three parameters to the list.

Example: Estimating Independent Parameters

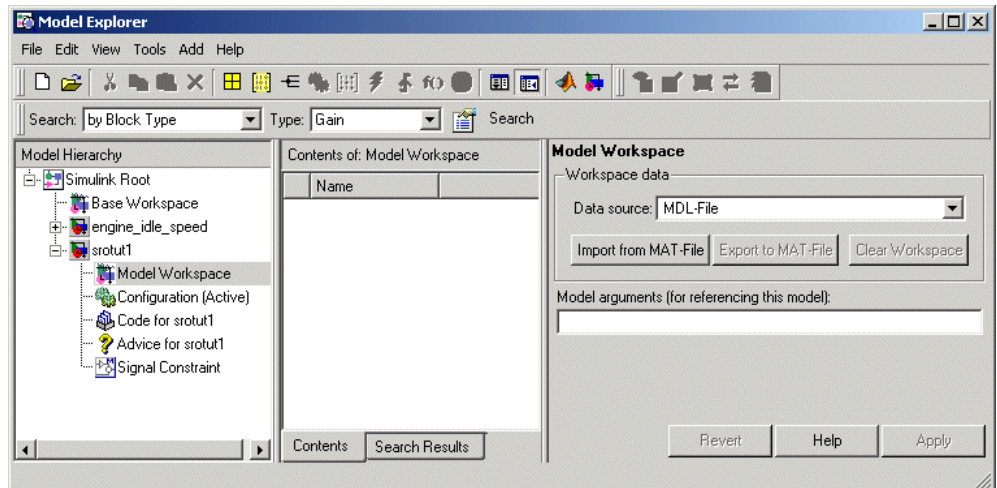
Assume that the parameter K_{int} in the model `srotut1` is related to the parameters x and y according to the relationship $K_{int}=x+y$. Also assume that the initial values of x and y are 1 and -0.7 respectively. To estimate x and y instead of K_{int} , first define these parameters in the model workspace. To do this:

- 1 At the MATLAB prompt, type

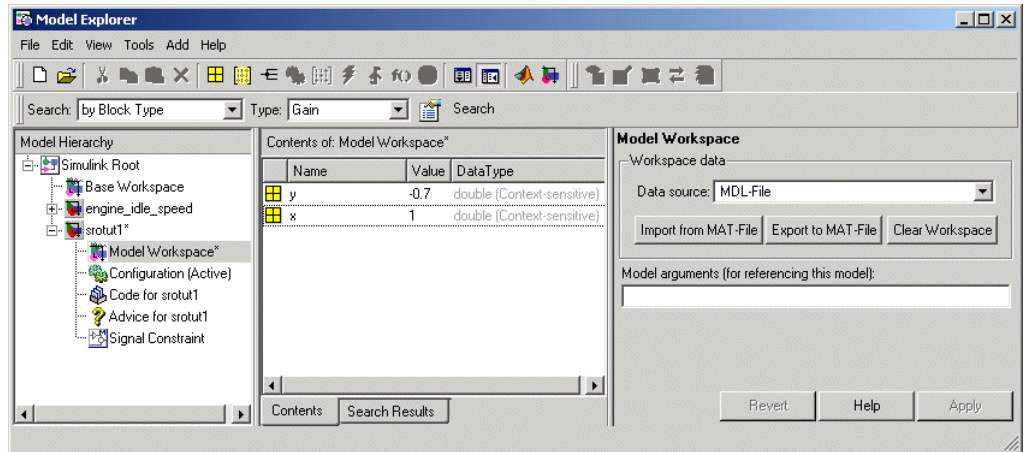
```
srotut1
```

This opens the `srotut1` model window.

- 2 Select **View > Model Explorer** from the `srotut1` window to open the Model Explorer window.
- 3 In the Model Hierarchy tree, select the `srotut1 > Model Workspace` node.



- 4 Select **Add > MATLAB Variable** to add a new variable to the model workspace. A new variable with a default name `Var` appears in the **Contents of: Model Workspace** pane.
- 5 Double-click `Var` to make it editable and change the variable name to `x`. Edit the initial Value to 1.
- 6 Repeat steps 4 and 5 to add a variable `y` with an initial value of `-0.7`. The Model Explorer window should resemble the following figure.

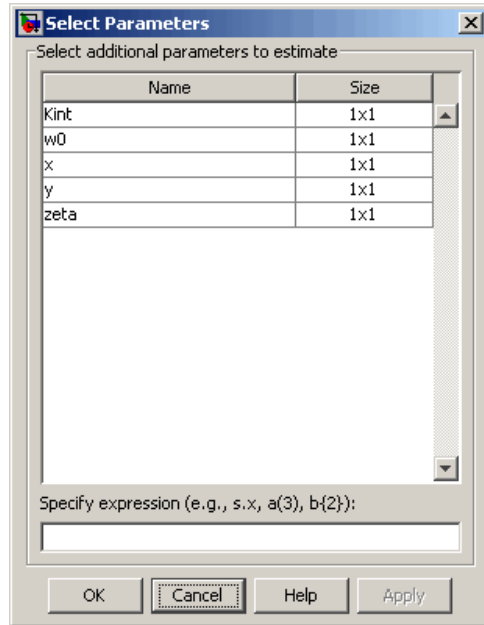


- 7 To add the Simulation Start function defining the relationship between Kint and the independent parameters x and y, select **File > Model Properties** in the Simulink model for srotut1.
- 8 In the Model Properties window, click the **Callbacks** tab.
- 9 To enter a Simulation start function in **StartFcn***, type the name of a new M-file, for example, srotut1_start.
- 10 Create a new M-file with this name. The contents of the M-file should define the relationship between the parameters in the model and the parameters in the workspace. For this example, the M-file should resemble the following:

```
wks = get_param(gcs, 'ModelWorkspace')
x = wks.evalin('x')
y = wks.evalin('y')
Kint = x+y;
```

Note You must first use the `get_param` function to get the variables x and y from the model workspace before you can use them to define Kint.

- 11** When you select parameters for estimation in the **Variables** node, x and y appear in the Select Parameters dialog box.



- 12** Follow the workflow described in “Setting Up an Estimation Project” on page 1-23 to estimate the parameters.

Tutorial — Preparing Data for Parameter Estimation Using the GUI

- “About This Tutorial” on page 2-2
- “Configuring a Project for Parameter Estimation” on page 2-4
- “Importing Data into the GUI” on page 2-6
- “Analyzing Data” on page 2-14
- “Selecting Data for Estimation” on page 2-16
- “Removing Outliers” on page 2-25
- “Filtering Data” on page 2-29
- “Interpolating Missing Data” on page 2-34
- “Saving the Project” on page 2-37

About This Tutorial

In this section...
“Objectives” on page 2-2
“About the Sample Data” on page 2-2

Objectives

In this tutorial, you learn how to import, analyze, and prepare measured input and output (I/O) data for estimating parameters of a Simulink model.

Note Simulink Parameter Estimation software estimates parameters from real, time-domain data only.

You learn to perform the following tasks using the GUI:

- Import data from the MATLAB workspace.
- Analyze data quality using a time plot.
- Select a subset of data for estimation.
- Remove outliers.
- Filter high-frequency noise.
- Compute missing data using interpolation.

About the Sample Data

In this tutorial, you use `spe_engine_throttle1.mat`, which contains I/O data measured from an engine throttle system. The MAT-file includes the following variables:

- `input1` — Input data samples
- `position1` — Output data samples
- `time1` — Time vector

Note The number of input and output data samples must be equal to the length of the corresponding time vector.

The engine throttle system controls the flow of air and fuel mixture to the engine cylinders. The throttle body contains a butterfly valve which opens when a driver presses the accelerator pedal. Opening this valve increases the amount of fuel mixture entering the cylinders, which increases the engine speed. A DC motor controls the opening angle of the butterfly valve in the throttle system. The input to the throttle system is the motor current (in amperes), and the output is the angular position of the butterfly valve (in degrees).

`spe_engine_throttle1.mdl` contains the Simulink model of the engine throttle system. For more information on building models, see “Creating a Simulink Model” in the Simulink documentation.

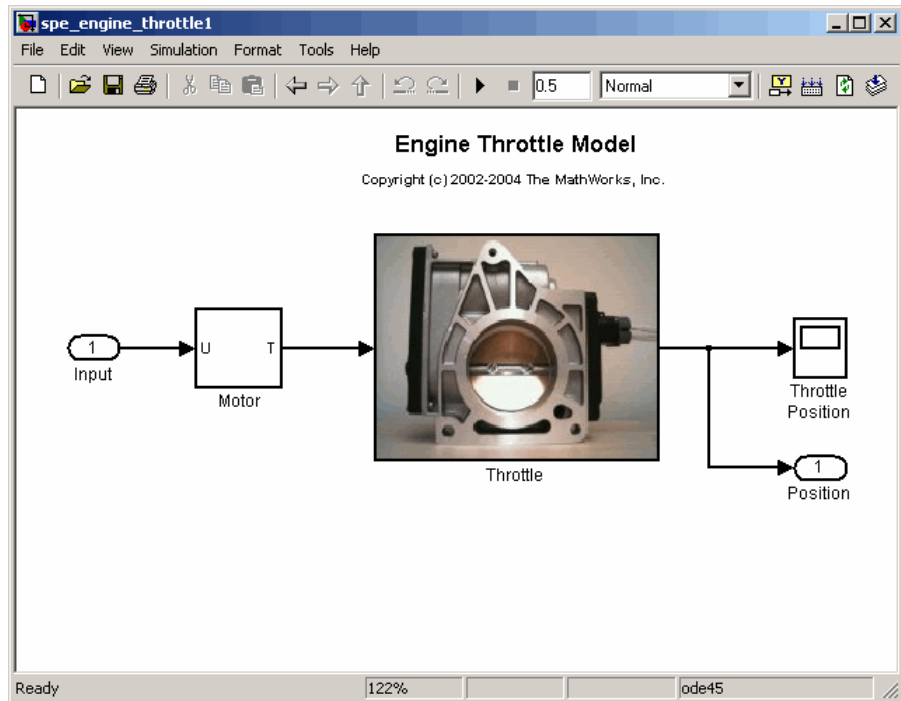
Configuring a Project for Parameter Estimation

To perform parameter estimation, you must first configure a Control and Estimation Tools Manager project.

- 1 Open the engine throttle system model by typing the following at the MATLAB prompt:

```
spe_engine_throttle1
```

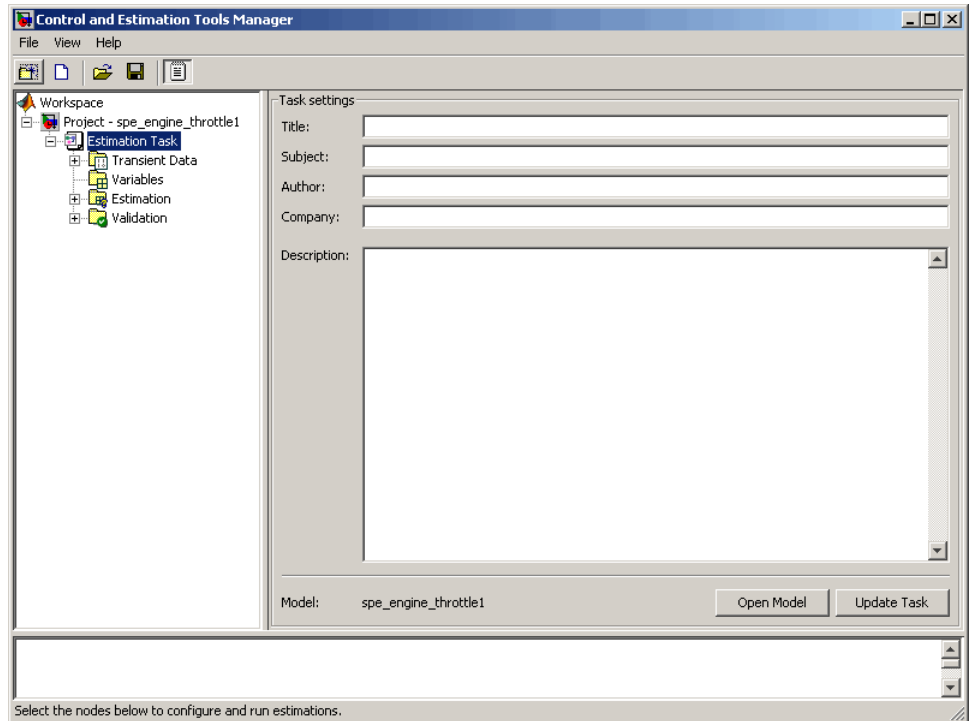
This command opens the Simulink model, and loads the data into the MATLAB workspace.



Simulink® Model of Engine Throttle System

- 2 In the Simulink model window, select **Tools > Parameter Estimation**.

This action opens a new project named **Project - spe_engine_throttle1** in the Control and Estimation Tools Manager GUI. This project contains the **Estimation Task**, as shown in the next figure.



Importing Data into the GUI

In this section...
“Importing Input Data and Time Vector” on page 2-6
“Importing Output Data and Time Vector” on page 2-11

Importing Input Data and Time Vector

In this portion of the tutorial, you import measured I/O data into the Control and Estimation Tools Manager GUI. Importing data assigns the data to the corresponding model input and output signals.

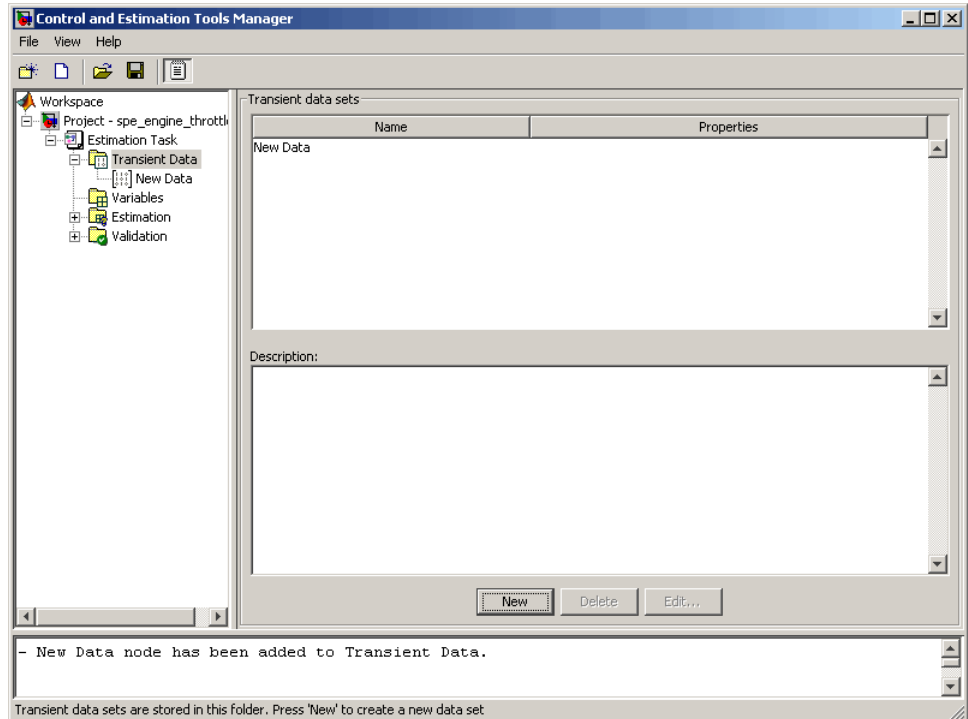
The model input and output signals are designated with the **Inport Input** and **Outport Position** blocks respectively, as shown in the figure **Simulink® Model of Engine Throttle System** on page 2-4. These blocks let you import I/O data into the GUI. To learn more about the blocks, see the **Inport** and **Outport** block reference pages in the Simulink documentation.

You must have already configured the parameter estimation project, as described in “Configuring a Project for Parameter Estimation” on page 2-4.

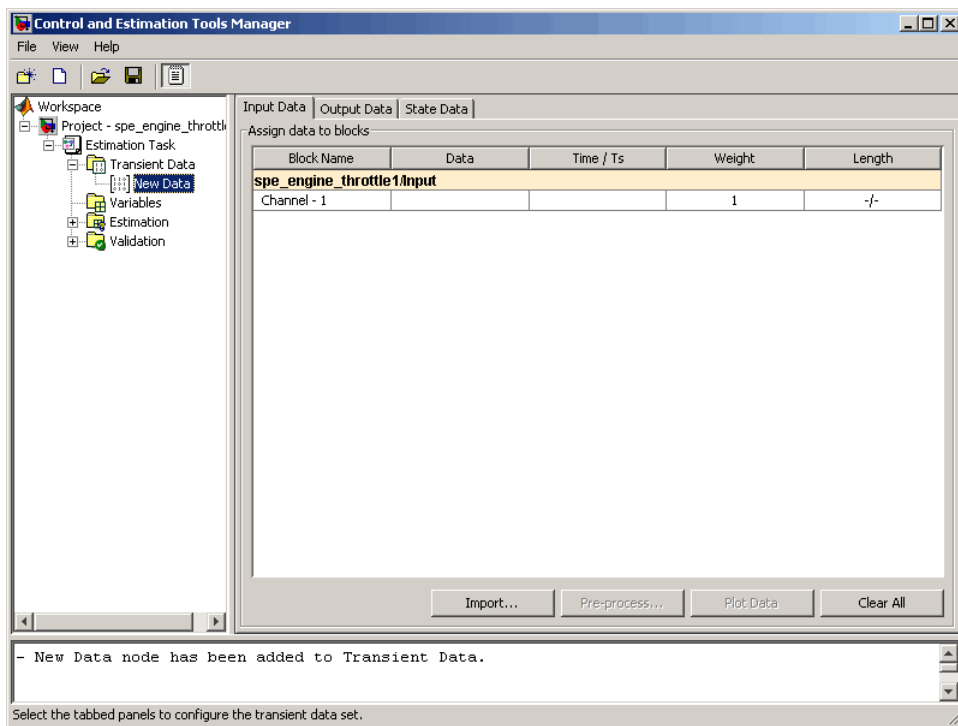
To import input data and time vector into the Control and Estimation Tools Manager GUI:

- 1 In the Control and Estimation Tools Manager GUI, select **Transient Data** under the **Estimation Task** node, and click **New**.

This action creates a **New Data** node under the **Transient Data** node.



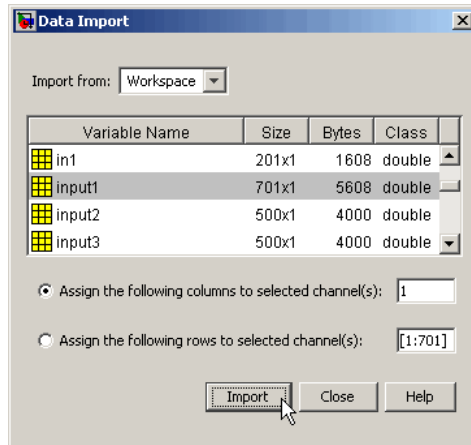
2 Select the **New Data** node.



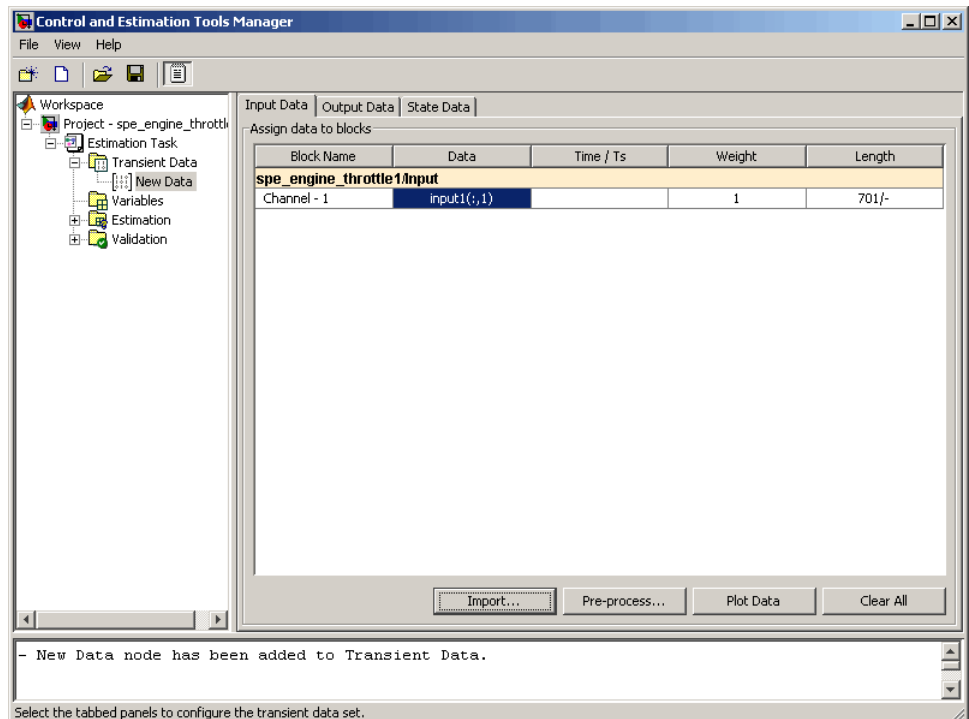
3 In the **Input Data** tab, select the **Data** cell for **Channel - 1**, and click **Import**.

This action opens the Data Import dialog box.

- 4 In the Data Import dialog box, select input1, and click **Import**.



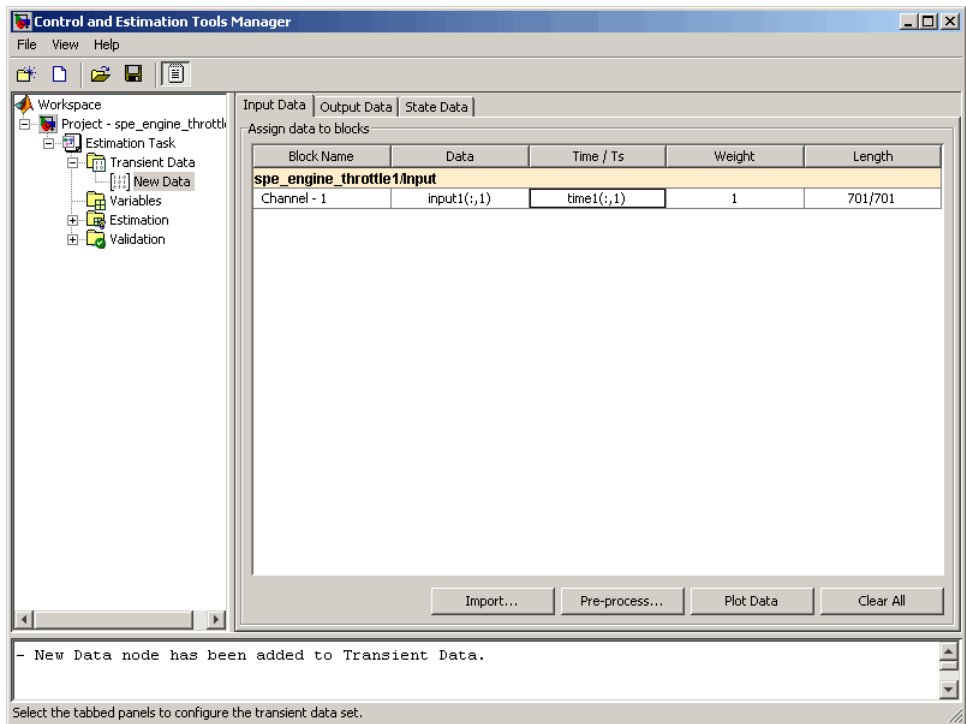
This action assigns the input data input1 to the model input signal `spe_engine_throttle1/Input`.



5 Select the **Time / Ts** cell for **Channel - 1**.

6 In the Data Import dialog box, select `time1`, and click **Import**.

This action assigns the time vector to the model input signal `spe_engine_throttle1/Input`.

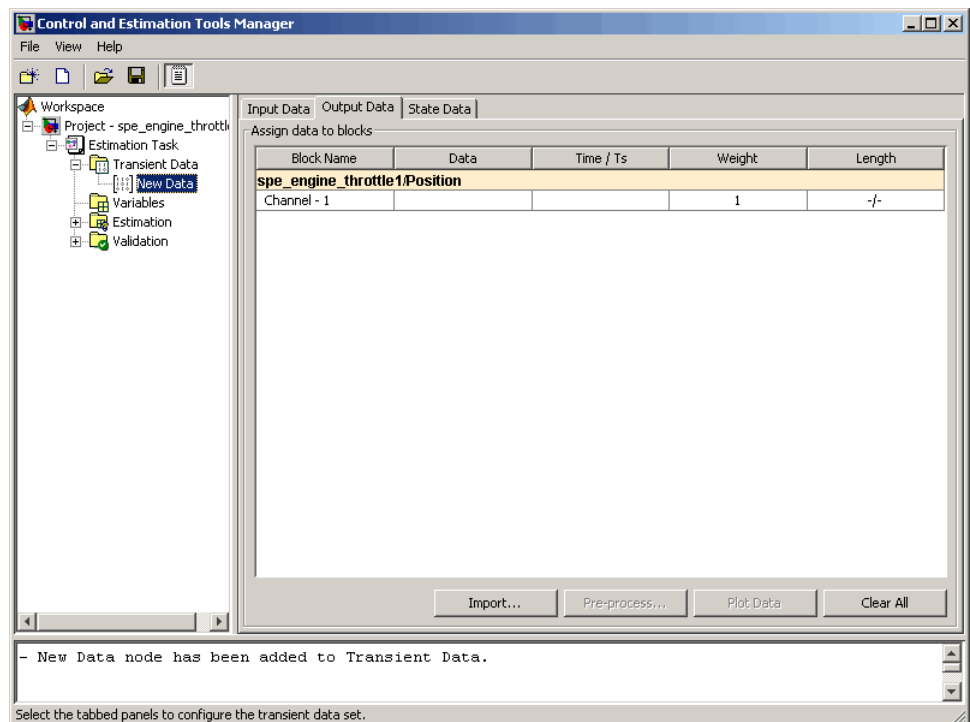


7 In the Data Import dialog box, click **Close**.

Importing Output Data and Time Vector

To import output data and time vector into the Control and Estimation Tools Manager GUI:

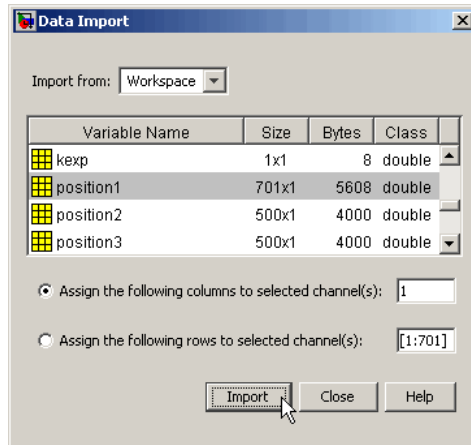
- 1 In the Control and Estimation Tools Manger GUI, select the **Output Data** tab of the **New Data** node.



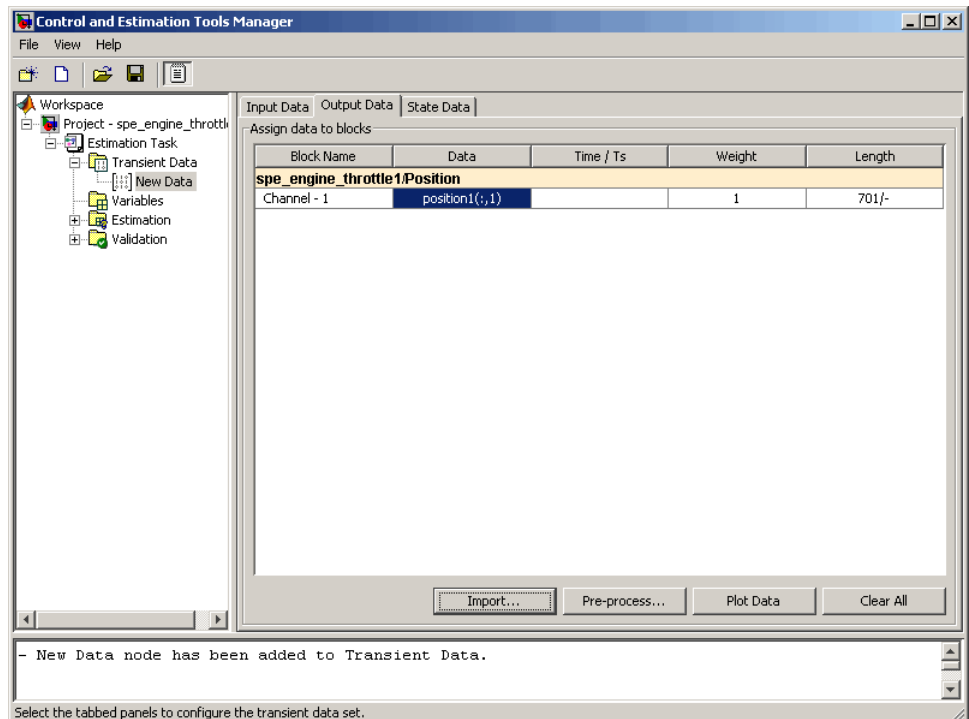
- 2 Select the **Data** cell for **Channel - 1**, and click **Import**.

This action opens the Data Import dialog box.

3 In the Data Import dialog box, select `position1`, and click **Import**.



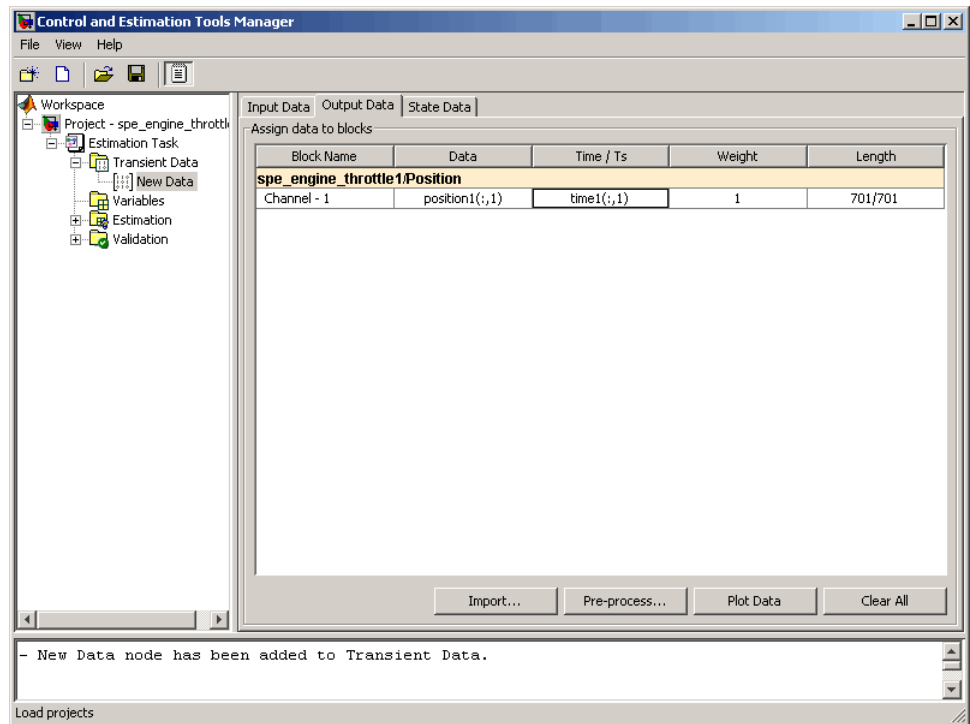
This action assigns the output data `position1` to the model output signal `spe_engine_throttle1/Position`.



4 Select the **Time / Ts** cell for **Channel - 1**.

5 In the Data Import dialog box, select **time1**, and click **Import**.

This action assigns the time vector to the model output signal **spe_engine_throttle1/Position**.



6 In Data Import dialog box, click **Close**.

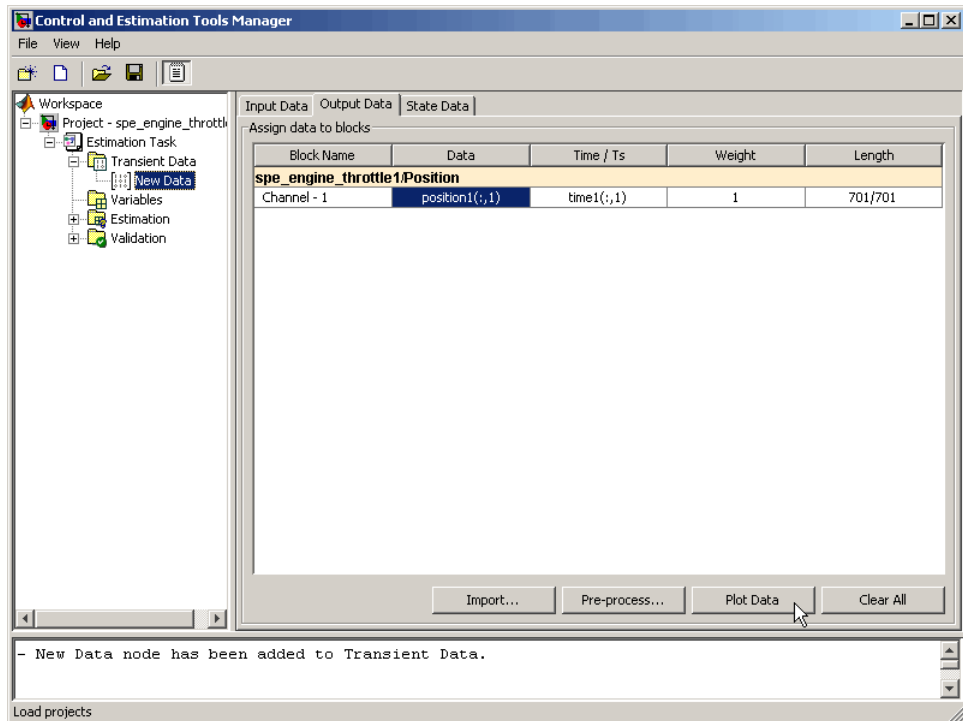
You have now imported the I/O data into the Control and Estimation Tools Manager GUI, and assigned the data to the corresponding model signals.

Analyzing Data

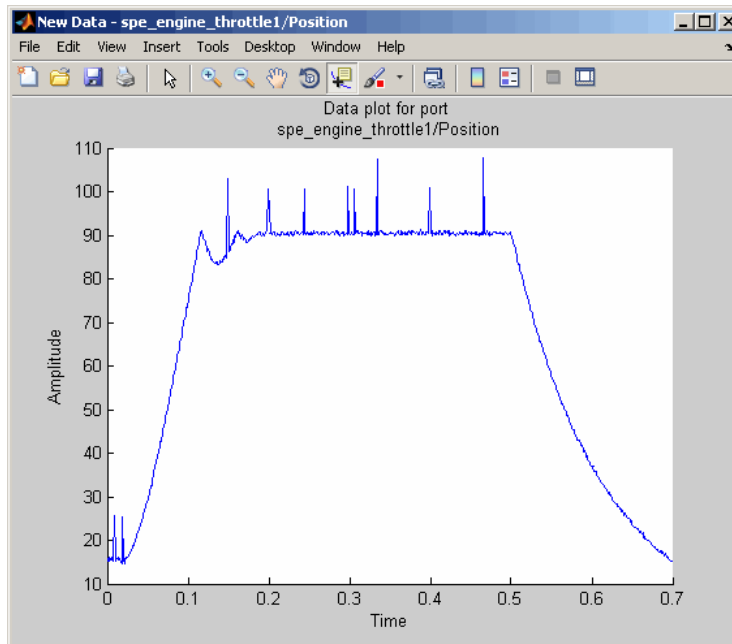
In this portion of the tutorial, you analyze the output data quality by viewing the data characteristics on a time plot. Based on the analysis, you decide whether to preprocess the data before estimating parameters. For example, if the data contains noise, you might want to filter the noise from the system dynamics before estimating parameters.

You must have already imported the data into the Control and Estimation Tools Manager GUI, as described in “Importing Data into the GUI” on page 2-6. If you have not imported the data, you can instead load the project `spe_engine_throttle1_import.mat` from the `matlabroot\toolbox\slestim\slestdemos` directory by selecting **File > Load** in the Control and Estimation Tools Manager GUI.

To plot the output data on a time plot, select the `position1(:,1)` cell in the **Output Data** tab, and click **Plot Data**.



This action plots the measured output data `position1(:,1)`, as shown in the next figure.



The time plot shows the output data in response to a step input, as described in “About the Sample Data” on page 2-2. The plot shows a rapid decrease in the response after $t = 0.5$ s because the system is shut down. To focus parameter estimation on the time period when the system is active, you select the data samples between $t = 0$ s and $t = 0.5$ s, as described in “Selecting Data for Estimation” on page 2-16 section of this tutorial.

The spikes in the data indicate *outliers*, defined as data values that deviate from the mean by more than three standard deviations. They may be caused by measurement errors or sensor problems. The response also contains noise. Before estimating model parameters from this data, you remove the outliers and filter the noise, as described in “Removing Outliers” on page 2-25, and “Filtering Data” on page 2-29 sections of this tutorial.

Tip You can also plot the input data on a time plot by selecting the `input1(:,1)` cell in the **Input Data** tab, and clicking **Plot Data**.

Selecting Data for Estimation

In this section...
“Selecting Output Data” on page 2-16
“Selecting Input Data” on page 2-22

Selecting Output Data

In this portion of the tutorial, you select a subset of I/O data for estimation. As described in “Analyzing Data” on page 2-14, the system is shut down at $t = 0.5$ s. To focus the estimation on the time period before $t = 0.5$ s, you exclude the data samples beyond $t = 0.5$ s. This operation selects the data between $t = 0$ s and $t = 0.5$ s for estimation.

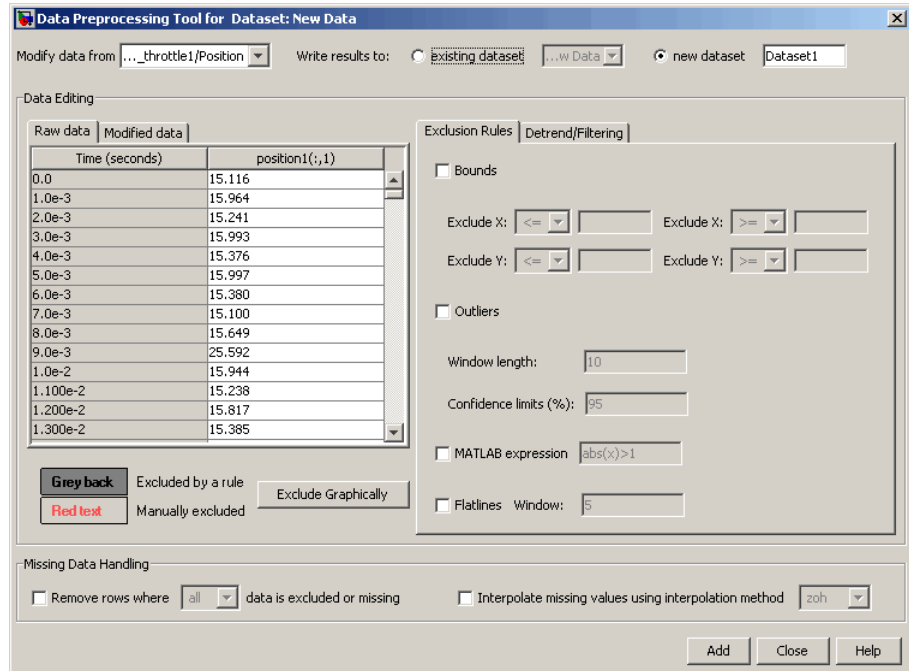
You must have already imported the data into the Control and Estimation Tools Manager GUI, as described in “Importing Data into the GUI” on page 2-6. If you have not imported the data, you can instead load the project `spe_engine_throttle1_import.mat` from the `matlabroot\toolbox\slestim\slestdemos` directory by selecting **File > Load** in the Control and Estimation Tools Manager GUI.

To select the portion of data between $t = 0$ s and $t = 0.5$ s:

- 1 In the Control and Estimation Tools Manager window, select the **New Data** node under the **Transient Data** node.

- 2 Select the `position1(:,1)` cell in the **Output Data** tab, and click **Pre-process**.

This action opens the Data Preprocessing Tool GUI.

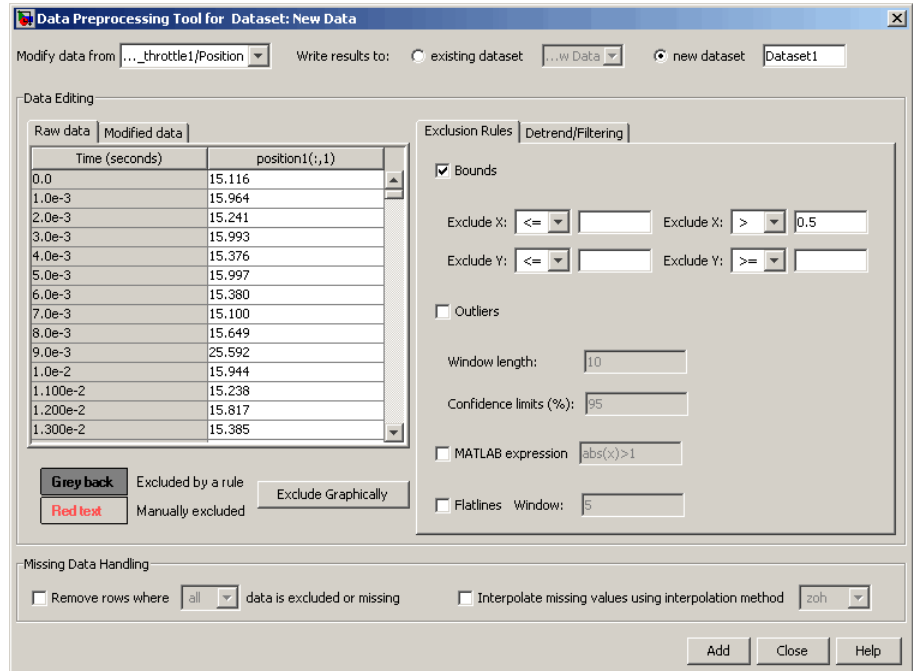


The **Data Editing** area of the Data Preprocessing Tool GUI shows the output data and time vector in the **position1(:,1)** and **Time (seconds)** columns, respectively. The Data Preprocessing Tool GUI lets you perform the following types of preprocessing operations:

- Excluding data
 - Detrending and filtering data
 - Handling missing data
- 3 To exclude the output data beyond $t = 0.5$ s:
- a In the **Exclusion Rules** tab, select the **Bounds** check box.

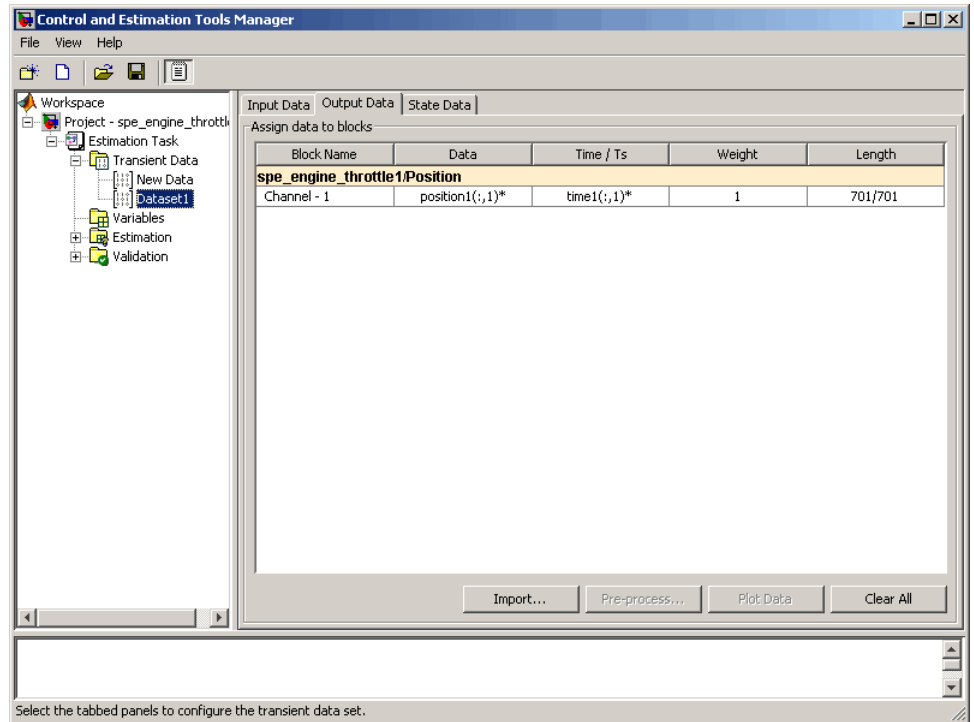
- b In the **Exclude X** \geq field, where **X** corresponds to the time vector, select $>$ from the drop-down list. Enter 0.5 in the adjacent field to specify the upper limit of the data to select for estimation.

The Data Preprocessing Tool GUI resembles the next figure.

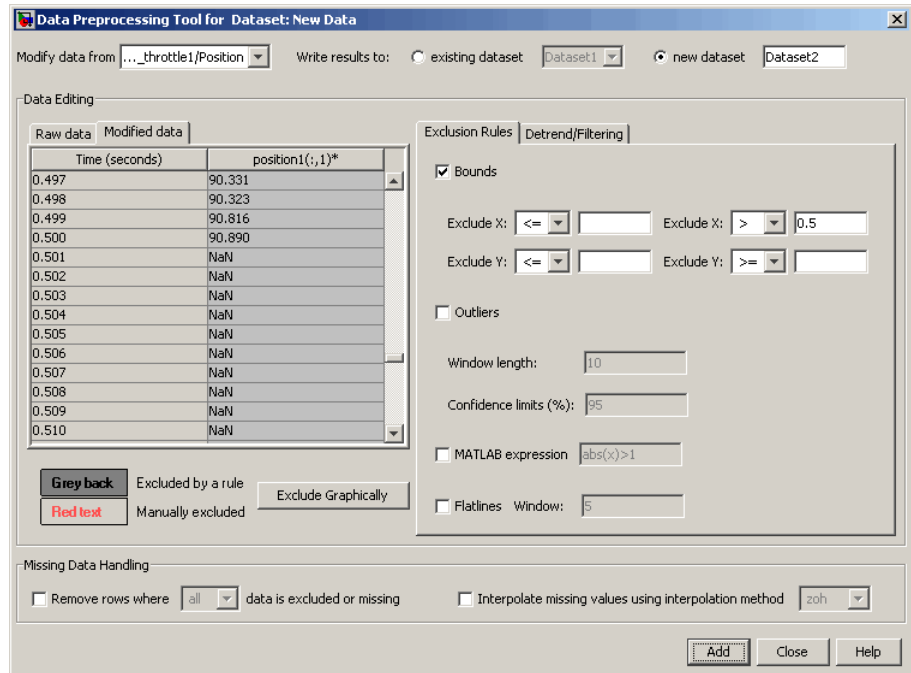


c Click **Add**.

This action adds a new **Dataset1** node under the **Transient Data** node in the Control and Estimation Tools Manager GUI. The **Dataset1** node contains the modified output data $\text{position1}(:,1)^*$ in the **Output Data** tab.



This operation also replaces the output data samples beyond $t = 0.5$ s in `position1(:,1)*` with NaNs. You can view the NaNs by selecting the **Modified data** tab in the Data Preprocessing Tool GUI, as shown in the next figure.

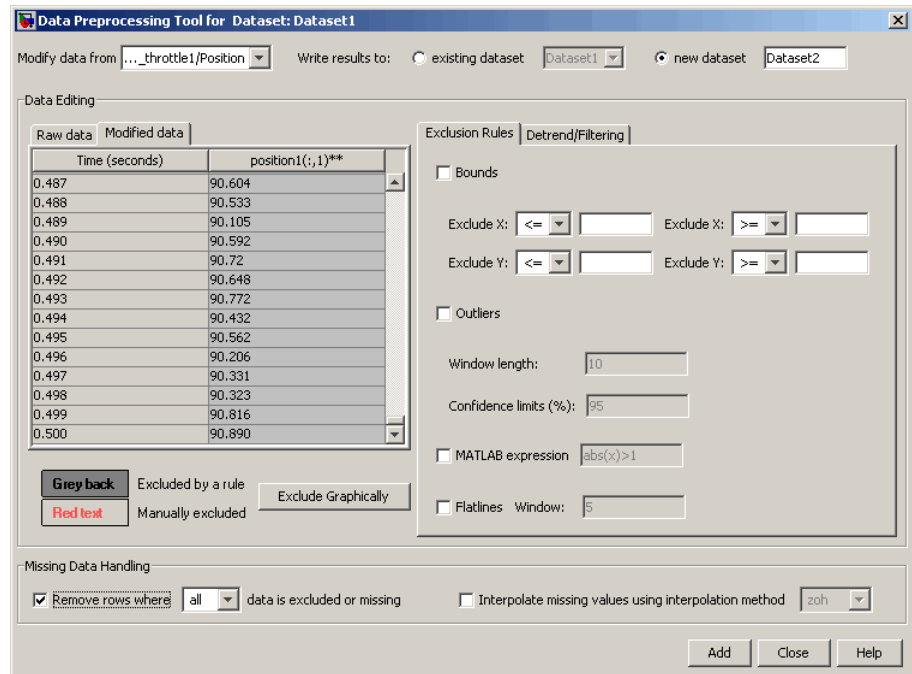


4 To remove the NaNs:

- a In the Control and Estimation Tools Manager GUI, select the **Output Data** tab of the **Dataset1** node.
- b Select the `position1(:,1)*` cell, and click **Pre-process**.

This action updates the Data Preprocessing Tool GUI with the selected data `position1(:,1)*`.

- c In the **Missing Data Handling** area, select the **Remove rows where** **data is excluded or missing** check box.



Tip You can view the results of this operation in the **Modified data** tab.

- d In the **Write results to** area, select the **existing dataset** option.

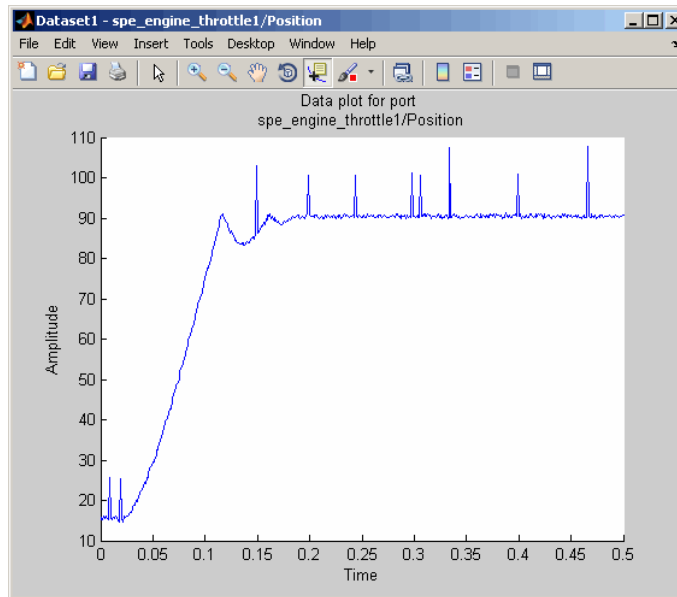


- e Click **Add**.

The Update table data dialog box appears. Click **Yes** to overwrite the `position1(:,1)*` data set with the modified data.

- 5 To plot the data, select the `position1(:,1)*` cell in the **Output Data** tab of the **Dataset1** node, and click **Plot Data**.

The selected output data from $t = 0$ s to $t = 0.5$ s is shown in the next figure.



Selecting Input Data

After you select the output data between $t = 0$ s and $t = 0.5$ s, as described in “Selecting Output Data” on page 2-16, you must also select the corresponding input data samples. This operation makes the number of I/O data samples equal.

- 1 In the Control and Estimation Tools Manager GUI, select the **New Data** node under the **Transient Data** node.
- 2 In the **Input Data** tab, select the `input1(:,1)` cell, and click **Pre-process**.

This action updates the Data Preprocessing Tool GUI with the selected data `input1(:,1)`.

3 To exclude the data beyond $t = 0.5$ s:

- a** In the **Exclusion Rules** tab, select the **Bounds** check box.
- b** In the **Exclude X** field, where **X** corresponds to the time vector, select **>** from the drop-down list. Enter **0.5** in the adjacent field to specify the upper limit of the input data to select for estimation.
- c** In the **Write results to** area, verify that the **existing dataset** option remains selected.
- d** Click **Add**.

This action adds the modified data $\text{input1}(:,1)^*$ to the **Input Data** tab of the **Dataset1** node. This operation also replaces the input data samples beyond $t = 0.5$ s in $\text{input1}(:,1)^*$ with NaNs.

4 To remove the NaNs:

- a** In the Control and Estimation Tools Manager GUI, select the $\text{input1}(:,1)^*$ cell in the **Input Data** tab of the **Dataset1** node, and click **Pre-process**.

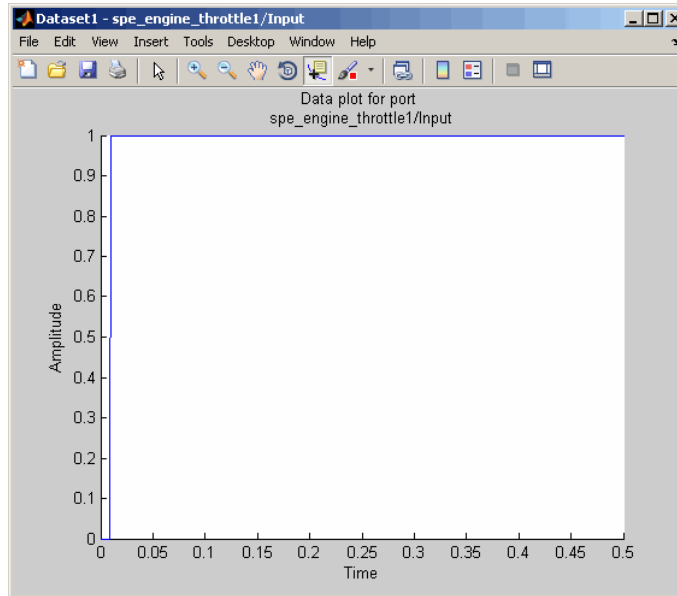
This action updates the Data Preprocessing Tool GUI with the selected data $\text{input1}(:,1)^*$.

- b** In the **Missing Data Handling** area, select the **Remove rows where** **data is excluded or missing** check box.
- c** In the **Write results to** area, verify that the **existing dataset** option remains selected.
- d** Click **Add**.

The Update table data dialog box appears. Click **Yes** to overwrite the $\text{input1}(:,1)^*$ data set with the modified data.

- 5 To plot the data, select the `input1(:,1)*` cell in the **Input Data** tab of the **Dataset1** node, and click **Plot Data**.

The selected input data from $t = 0$ s to $t = 0.5$ s is shown in the next figure.



Removing Outliers

In this section...

“Why Remove Outliers” on page 2-25

“How to Remove Outliers” on page 2-25

Why Remove Outliers

Outliers are data values that deviate from the mean by more than three standard deviations. When estimating parameters from data containing outliers, the results may not be accurate.

Removing outliers replaces the data samples containing outliers with NaNs, which represent missing data. You interpolate the missing data values in a subsequent operation, as described in “Interpolating Missing Data” on page 2-34.

How to Remove Outliers

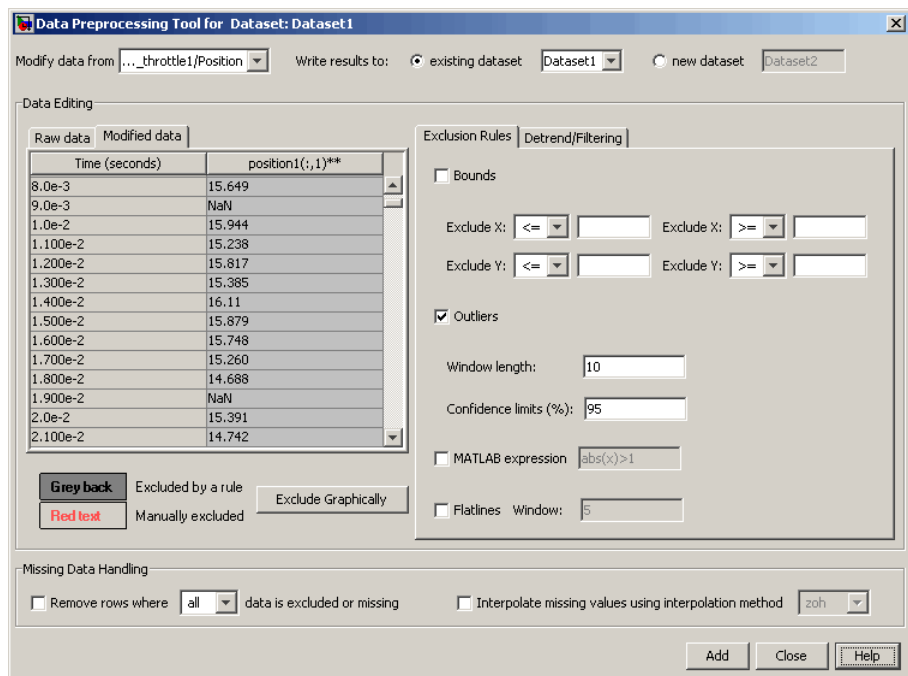
In this portion of the tutorial, you remove outliers from the output data. You must have already selected a subset of the data, as described in “Selecting Data for Estimation” on page 2-16. If you have not done this preparation, you can instead load the project `spe_engine_throttle1_exclude.mat` from the `matlabroot\toolbox\slestim\slestdemos` directory by selecting **File > Load** in the Control and Estimation Tools Manager GUI.

- 1 In the Control and Estimation Tools Manger, select the `position1(:,1)*` cell in the **Output Data** tab of the **Dataset1** node, and click **Pre-process**.

This action updates the Data Preprocessing Tool GUI with the selected data `position1(:,1)*`.

2 In the **Exclusion Rules** tab, select the **Outliers** check box.

By default, the **Window length** and **Confidence limits** fields are set to 10 and 95 respectively. The **Window length** field specifies the number of successive data samples the software uses to compute the mean and standard deviation. The **Confidence limits** field specifies the threshold number for identifying outliers. In this example, the mean and standard deviation of 10 successive data samples are computed, and data values that exceed 95% of standard deviation are identified as outliers.



Note The data samples containing outliers are replaced with NaNs.

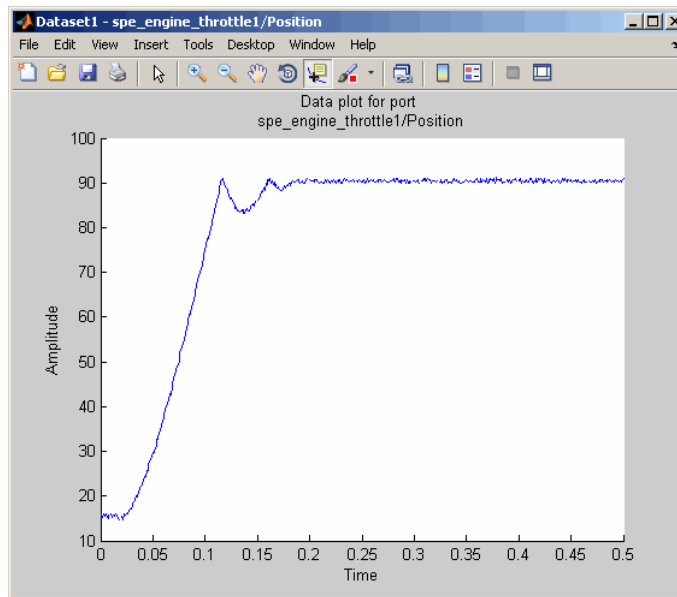
3 In the **Write results to** area, verify that the **existing dataset** **Dataset1** option remains selected.

4 Click **Add**.

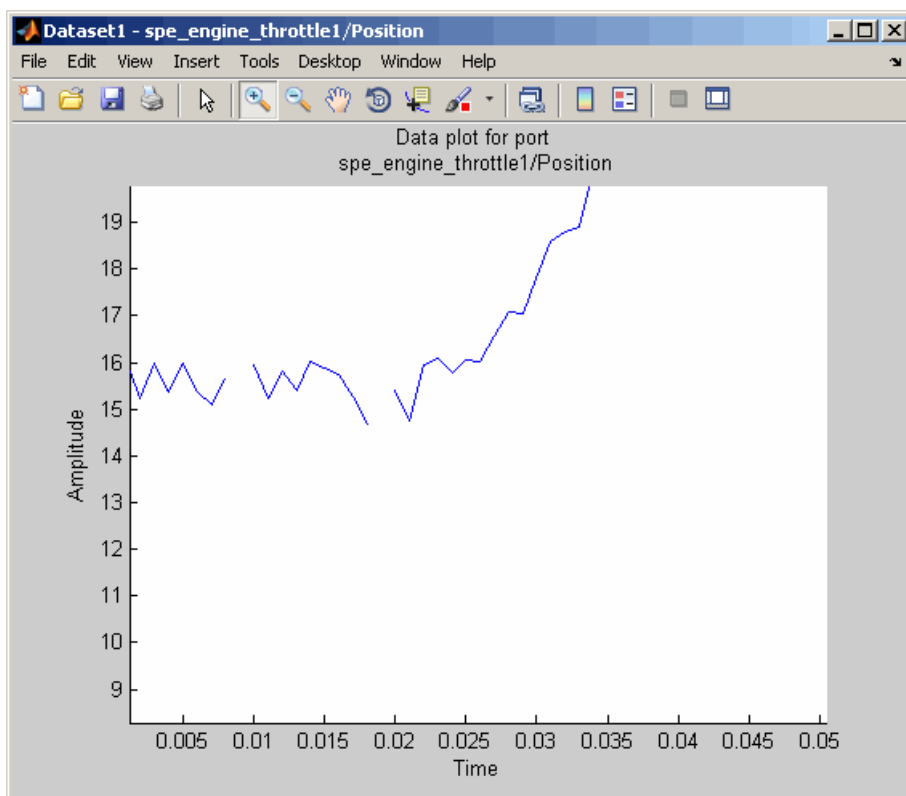
The Update table data dialog box appears. Click **Yes** to overwrite the `position1(:,1)*` data set with the modified data.

5 To plot the data, select the `position1(:,1)*` cell in the **Output Data** tab of the **Dataset1** node, and click **Plot Data**.

The spikes, which indicate outliers, no longer appear on the time plot, as shown in the next figure.



The missing data samples, represented by NaNs, appear as gaps on the time plot. To see an example, zoom in to the bottom-left corner of the plot. As shown in the next figure, the data values corresponding to $t = 0.009$ s and $t = 0.019$ s are missing.



Filtering Data

In this section...
“Filtering Output Data” on page 2-29
“Filtering Input Data” on page 2-32

Filtering Output Data

In this portion of the tutorial, you filter the noise, and remove any periodic trends from the output data. To avoid relative phase shift between the I/O data, you must also apply the same filter to the input data.

You must have already removed outliers from the output data, as described in “Removing Outliers” on page 2-25. If you have not done this preparation, you can instead load the project `spe_engine_throttle1_removeoutlier.mat` from the `matlabroot\toolbox\slestim\slestdemos` directory by selecting **File > Load** in the Control and Estimation Tools Manager GUI.

- 1 In the Control and Estimation Tools Manager window, select the `position1(:,1)*` cell in the **Output Data** tab of the **Dataset1** node, and click **Pre-process**.

This action updates the Data Preprocessing Tool GUI with the selected data `position1(:,1)*`.

2 In the **Detrend/Filtering** tab:

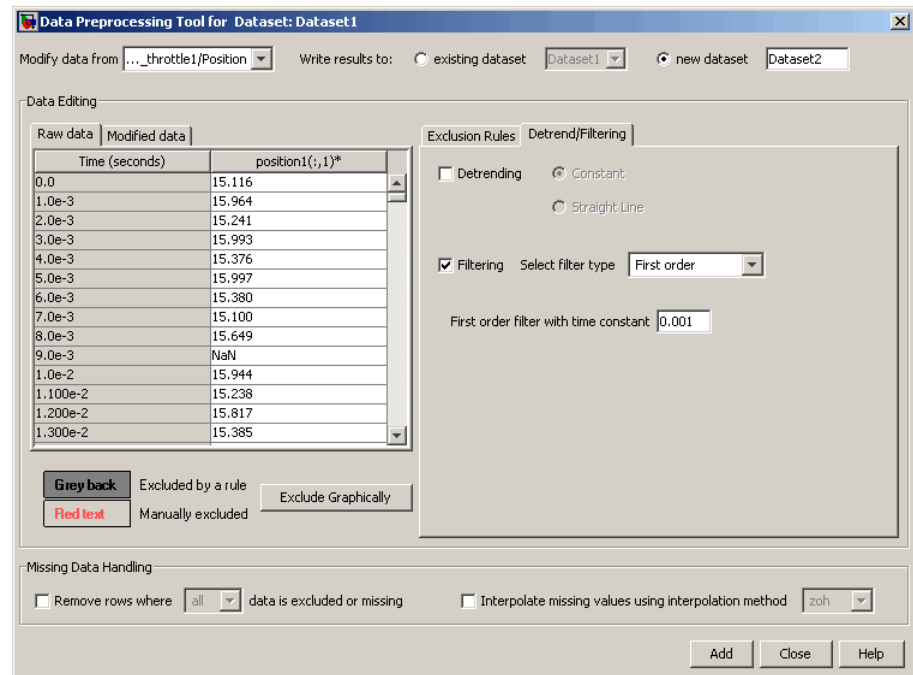
- a Select the **Filtering** check box.

By default, **First order** is selected as the filter type. To learn more about the filters, see “Filtering” on page 6-16.

- b Specify **0.001** in the **First order filter with time constant** field.

This field specifies the time constant for the first-order filter.

Tip For calculating the time constant, you can visually inspect the time plot to determine the frequency components.



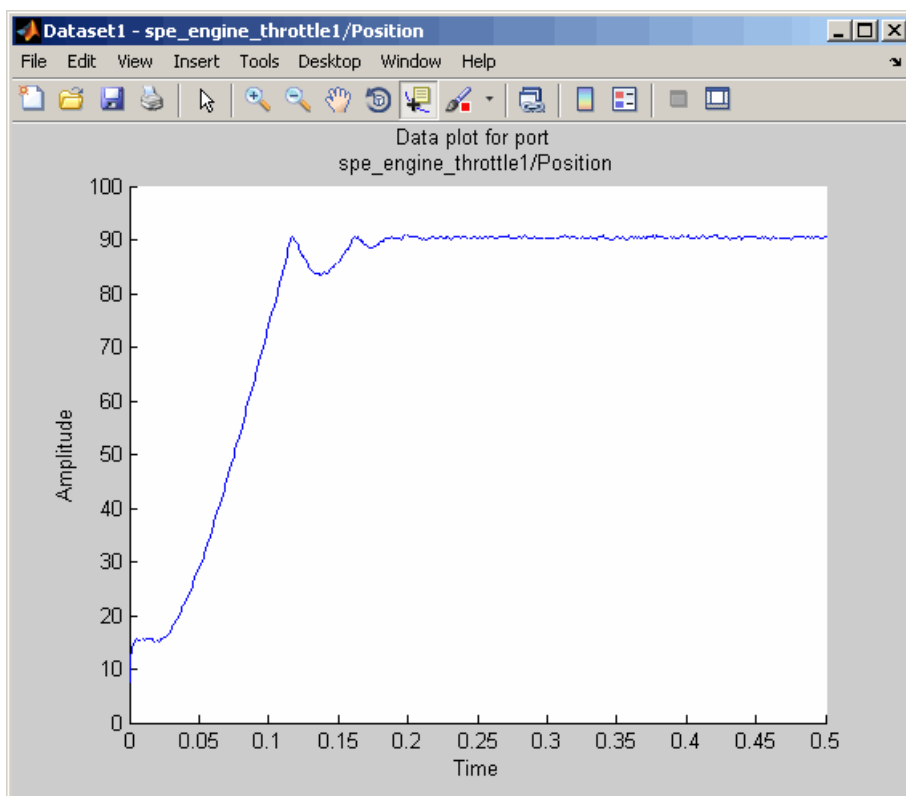
- 3 In the **Write results to** area, verify that the **existing dataset** **Dataset1** option remains selected.

4 Click **Add**.

The Update table data dialog box appears. Click **Yes** to overwrite the `position1(:,1)*` data set with the modified data.

5 To plot the data, select the `position1(:,1)*` cell in the **Output Data** tab of the **Dataset1** node, and click **Plot Data**.

The noise is filtered and the output data appears smooth, as shown in the next figure.



Filtering Input Data

After you filter the output data, as described in “Filtering Output Data” on page 2-29, you must also filter the input data with the same filter.

- 1 In the Control and Estimation Tools Manager window, select the `input1(:,1)*` cell in the **Input Data** tab of the **Dataset1** node, and click **Pre-process**.


This action updates the Data Preprocessing Tool GUI with the selected data `input1(:,1)*`.

- 2 In the **Detrend/Filtering** tab:

- a Select the **Filtering** check box.

By default, `First order` is selected as the filter type.

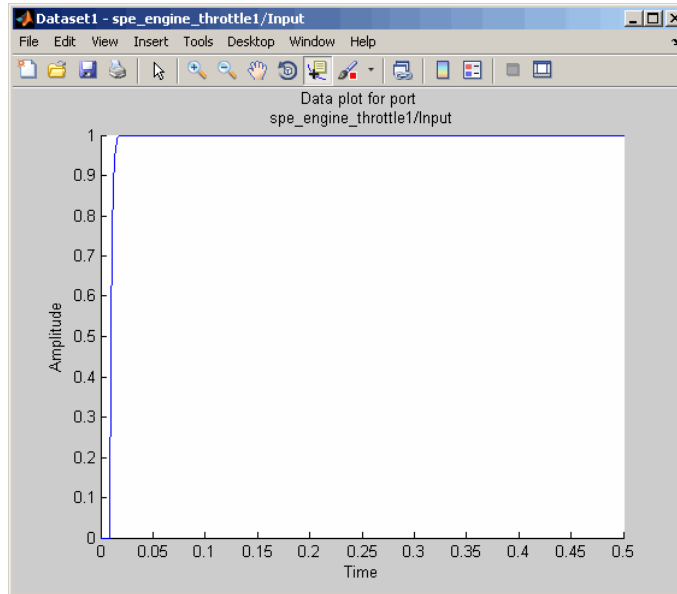
- b Specify `0.001` in the **First order filter with time constant** field.

- 3 In the **Write results to** area, verify that the **existing dataset**  option remains selected.

- 4 Click **Add**.

The Update table data dialog box appears. Click **Yes** to overwrite the `input1(:,1)*` data set with the modified data.

- 5 To plot the data, select the `input1(:,1)*` cell in the **Input Data** tab of the **Dataset1** node, and click **Plot Data**.



Interpolating Missing Data

In this portion of the tutorial, you interpolate the output data to compute the missing values created when removing outliers, as described in “Removing Outliers” on page 2-25. The interpolation operation uses the known data values to compute the missing data values.

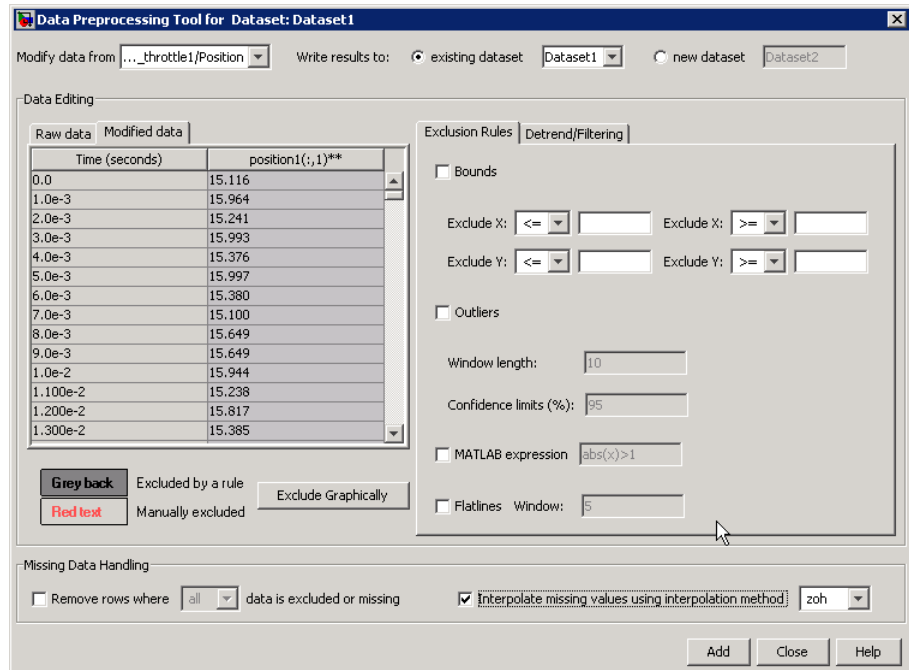
You must have already filtered the noise, as described in “Filtering Data” on page 2-29. If you have not already done this preparation, you can instead load the saved project `spe_engine_throttle1_filter.mat` from the `matlabroot\toolbox\slestim\slestdemos` directory by selecting **File > Load** in the Control and Estimation Tools Manager GUI.

- 1 In the Control and Estimation Tools Manager GUI, select the `position1(:,1)*` cell in the **Output Data** tab of the **Dataset1** node, and click **Pre-process**.

This action updates the Data Preprocessing Tool GUI with the selected data `position1(:,1)*`.

- 2** In the **Missing Data Handling** area, select the **Interpolate missing values using interpolation method** check box.

By default, `zoh` is selected as the interpolation method. This method fills the missing data sample with the data value immediately preceding it.



Tip You can view the interpolated data samples in the **Modified data** tab.

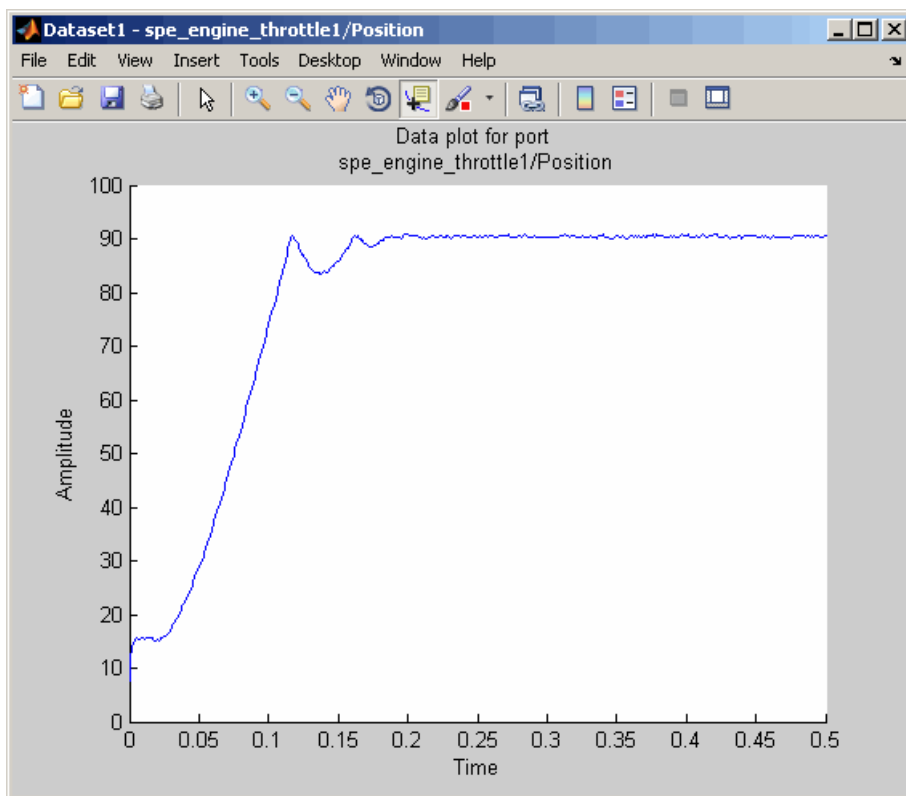
- 3** In the **Write results to** area, verify that the **existing dataset** `Dataset1` option remains selected.

- 4** Click **Add**.

The Update table data dialog box appears. Click **Yes** to overwrite the `position1(:,1)*` data set with the modified data.

- 5 To plot the data, select the `position1(:,1)*` cell in the **Output Data** tab of the **Dataset1** node, and click **Plot Data**.

The new estimation data, prepared by removing outliers, filtering noise, and interpolating missing data, is shown the next figure.



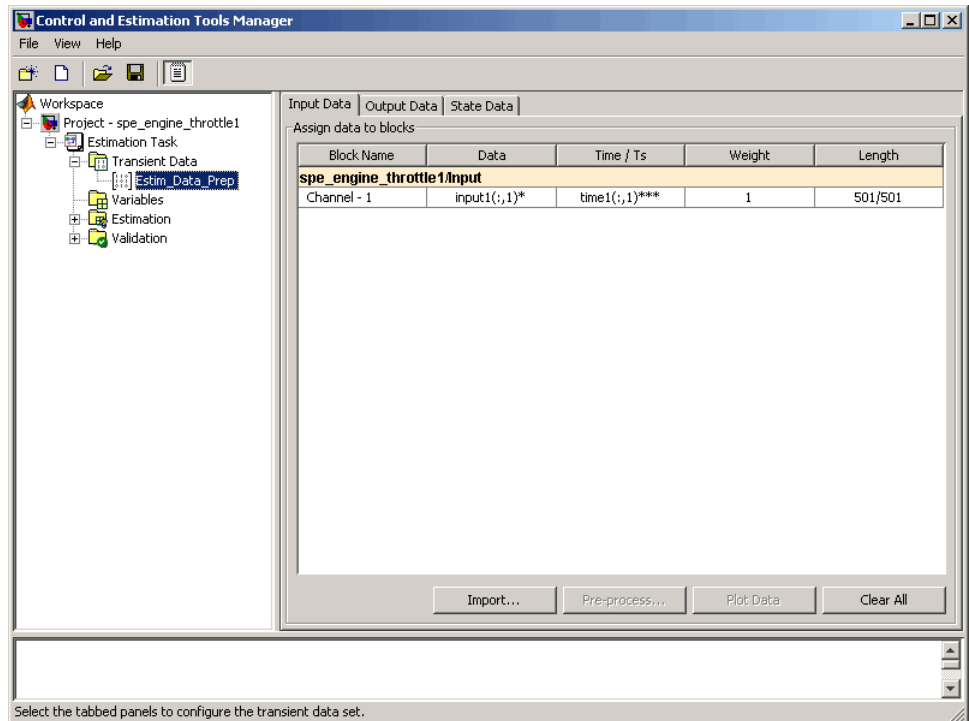
Saving the Project

After you prepare the data, you can delete the data in the **New Data** node, rename the prepared data, and save the session. To skip the data preparation steps, load the project `spe_engine_throttle1_interpolate.mat` from the `matlabroot\toolbox\slestim\slestdemos` directory by selecting **File > Load** in the Controls and Estimation Tools Manager GUI.

- 1** In the Control and Estimation Tools Manger GUI, select the **New Data** node under the **Transient Data** node.
- 2** Right-click the **New Data** node, and select **Delete**.
- 3** Select the **Dataset1** node under the **Transient Data** node.

- 4 Right-click the **Dataset1** node, and select **Rename**. Specify **Estim_Data_Prep** as the name of the new estimation data set.

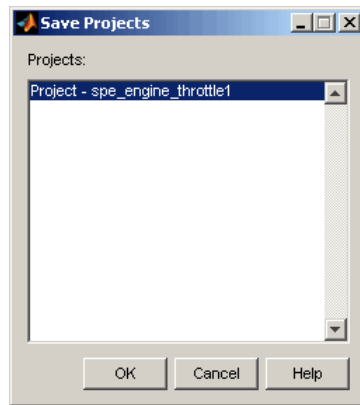
The Control and Estimation Tools Manager GUI resembles the next figure.



- 5 Save the Control and Estimation Tools Manager project:
 - a In the Control and Estimation Tools Manager GUI, select **File > Save**.

This action opens the Save Projects dialog box.

- b In the Save Projects dialog box, click **OK**.



- c In the Save Projects window, specify `spe_engine_throttle1p.mat` in the **File name** field, and click **Save**.

The action saves the project as a MAT-file.

To learn how to estimate parameters from this data, see Chapter 3, “Tutorial — Estimating Parameters from Measured Data Using the GUI”.

Tutorial — Estimating Parameters from Measured Data Using the GUI

- “About This Tutorial” on page 3-2
- “Estimating Model Parameters Using Default Estimation Settings” on page 3-7
- “Improving Estimation Results Using Parameter Bounds” on page 3-20
- “Validating Estimated Model Parameters” on page 3-26

About This Tutorial

In this section...
“Objectives” on page 3-2
“About the Model” on page 3-3

Objectives

In this tutorial, you learn how to estimate parameters of a single-input single-output (SISO) Simulink model from measured input and output (I/O) data.

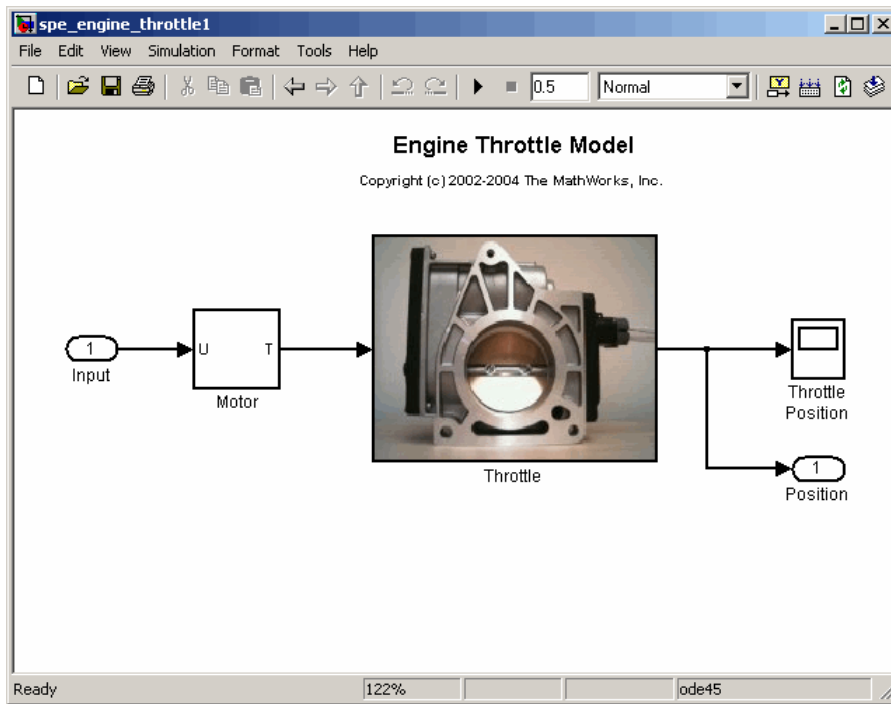
Note Simulink Parameter Estimation software estimates parameters from real, time-domain data only.

You learn to perform the following tasks using the GUI:

- Load a saved project containing data.
- Estimate model parameters using default settings.
- Validate the model, and refine the estimation results.

About the Model

In this tutorial, you use the `spe_engine_throttle1` Simulink model. This model represents an engine throttle system, as shown in the next figure.

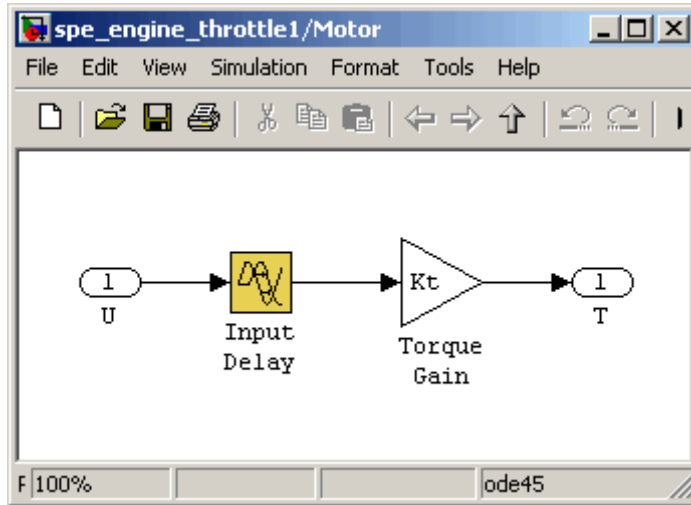


The throttle system controls the flow of air and fuel mixture to the engine cylinders. The throttle body contains a butterfly valve which opens when a driver presses the accelerator pedal. Opening this valve increases the amount of fuel mixture entering the cylinders, which increases the engine speed. A DC motor controls the opening angle of the butterfly valve in the throttle system. The models for these components are described in “Motor Subsystem” on page 3-4, and “Throttle Subsystem” on page 3-5.

The input to the throttle system is the motor current (in amperes), and the output is the angular position of the butterfly valve (in degrees).

Motor Subsystem

The Motor subsystem contains the DC motor model. To open the model, double-click the corresponding block.

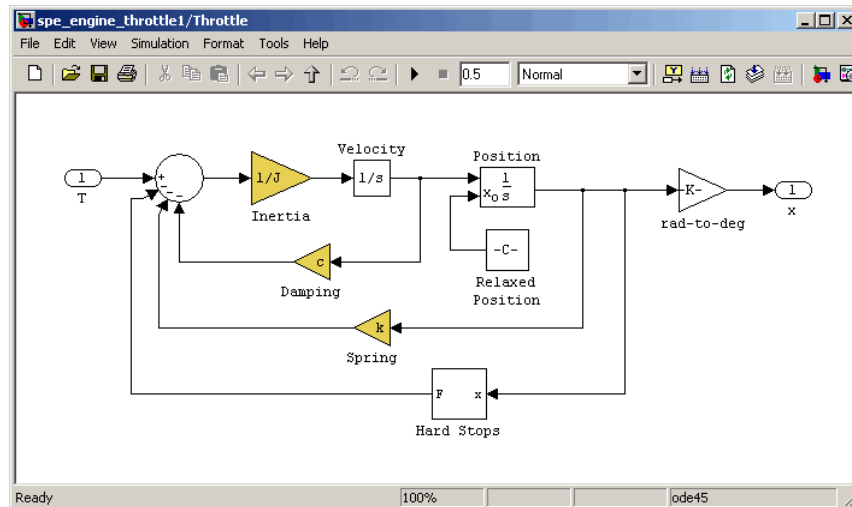


The following table describes the variables, parameters, equation, input, and output of the Motor subsystem.

Variables	<p>U is the input current to the motor.</p> <p>T is the torque applied by the motor.</p>
Parameters	<p>K_t is the torque gain of the motor, represented by Kt in the model.</p> <p>t_d is the input time delay of the motor, represented by <code>input_delay</code> in the model.</p>
Equation	<p>The torque applied by the motor is described in the following equation:</p> $T(t) = K_t U(t - t_d)$ <p>where t is time.</p>
Input	U
Output	T

Throttle Subsystem

The Throttle subsystem contains the butterfly valve model. To open the model, right-click the corresponding block, and select **Look Under Mask**.



The Hard Stops block models the valve angular position limit of 15° to 90° .

The following table describes the variables, parameters, states, differential equations, inputs, and outputs of the Throttle subsystem.

Variables	<p>T is the torque applied by the DC motor.</p> <p>θ is the angular position of the valve, represented by x in the model.</p> <p>$T_{hardstop}$ is the torque applied by the hard stop.</p>
Parameters	<p>J is the inertia.</p> <p>c is the viscous friction.</p> <p>k is the spring constant.</p>
States	<p>θ is the angular position.</p> <p>$\dot{\theta}$ is the angular velocity.</p>

Equations	<p>The mathematical system for the butterfly valve is described in the following equation:</p> $J\ddot{\theta} + c\dot{\theta} + k\theta = T + T_{hardstop}$ <p>where $15^\circ \leq \theta \leq 90^\circ$, with initial conditions $\theta_0 = 15^\circ$, and $\dot{\theta}_0 = 0$. The torque applied by the Hard Stops block is described in the following equation:</p> $T_{hardstop} = \begin{cases} 0, & 15^\circ \leq \theta \leq 90^\circ \\ K(90^\circ - \theta), & \theta > 90^\circ \\ K(15^\circ - \theta), & \theta < 15^\circ \end{cases}$ <p>where K is the gain of the Hard Stops block.</p>
Input	T
Output	θ

Estimating Model Parameters Using Default Estimation Settings

In this section...

“Overview of the Estimation Process” on page 3-7

“Specifying Parameters and Estimation Data” on page 3-8

“Validating Model Parameters” on page 3-13

Overview of the Estimation Process

Simulink Parameter Estimation software uses optimization techniques to estimate model parameters. In each optimization iteration, the model is simulated with the current parameter values. The error between the simulated and measured output is computed and minimized. The estimation is complete when the optimization algorithm finds a local minimum.

You perform the following tasks to estimate the model parameters:

- “Specifying Parameters and Estimation Data” on page 3-8
- “Validating Model Parameters” on page 3-13

Specifying Parameters and Estimation Data

To specify parameters and estimation data:

- 1 Load the saved project `spe_engine_throttle1p.mat`.

This MAT-file contains the estimation data. To learn more about the data and how to prepare it, see Chapter 2, “Tutorial — Preparing Data for Parameter Estimation Using the GUI”.

- a Open the engine throttle system Simulink model by typing the following at the MATLAB prompt:

```
spe_engine_throttle1
```

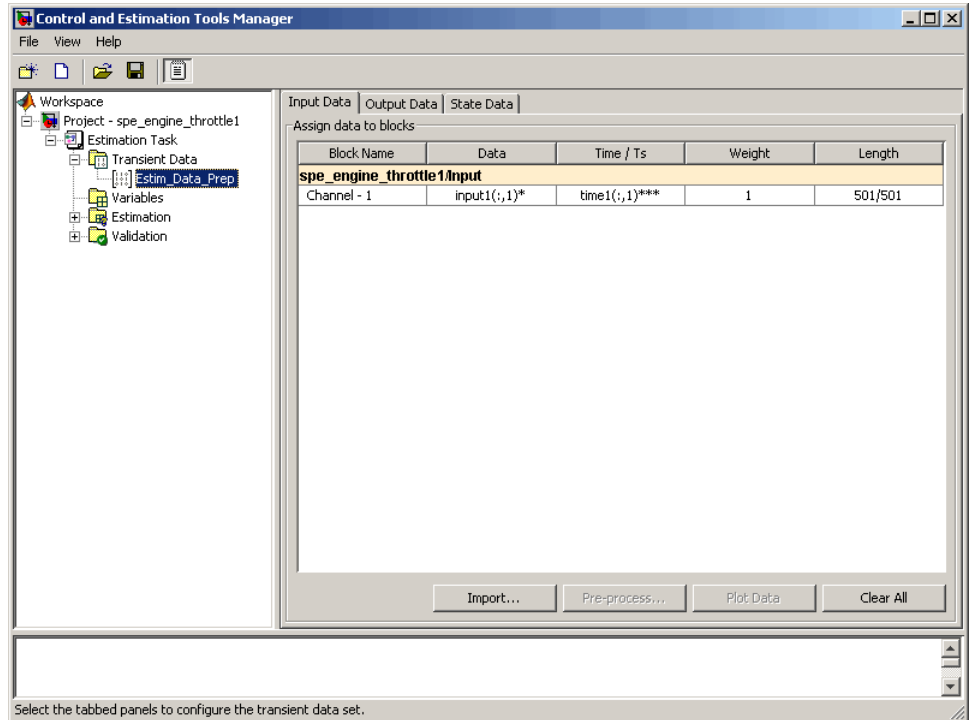
- b In the Simulink model window, select **Tools > Parameter Estimation**.

This action opens a new project named **Project - spe_engine_throttle1** in the Control and Estimation Tools Manager GUI.

- c In the Control and Estimation Tools Manager GUI, select **File > Load**. Browse to the `matlabroot\toolbox\slestim\slestdemos` directory, and select `spe_engine_throttle1p.mat`.

The Confirm Node Replace dialog box appears. Click **Yes** to load the project into the Control and Estimation Tools Manager GUI.

When you expand the **Estimation Task** node, the Controls and Estimation Tools Manager GUI resembles the next figure.



Input Data | Output Data | State Data |

-Assign data to blocks-

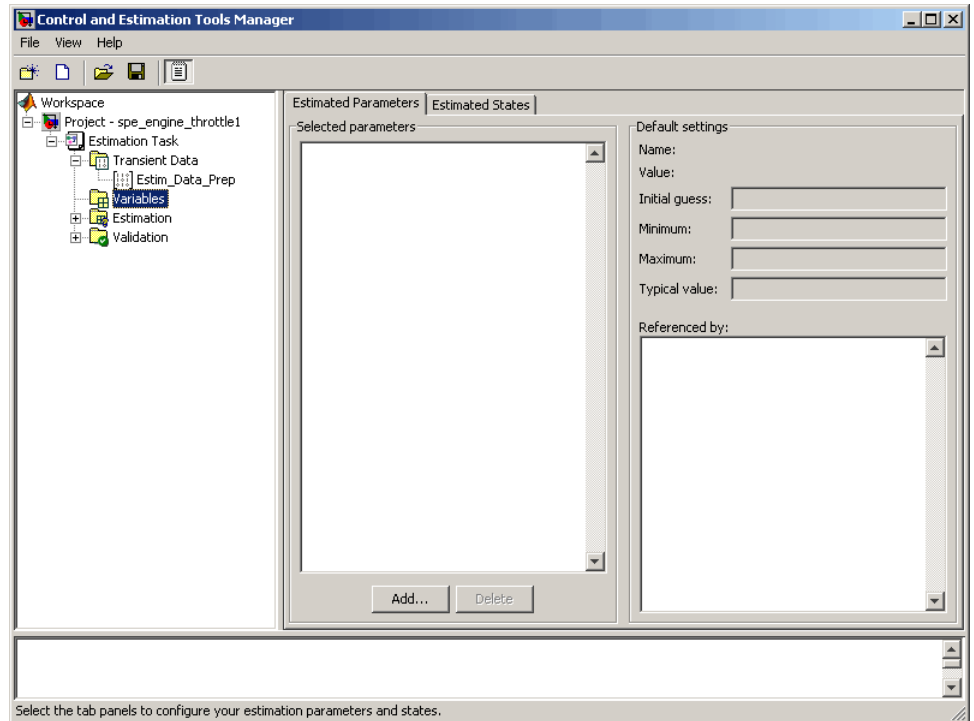
Block Name	Data	Time / Ts	Weight	Length
spe_engine_throttle1/input				
Channel - 1	input1(:,1)*	time1(:,1)**3	1	501/501

Buttons: Import... Pre-process... Plot Data Clear All

Select the tabbed panels to configure the transient data set.

2 Specify parameters for estimation.

- a Select the **Variables** node under the **Estimation Task** node.

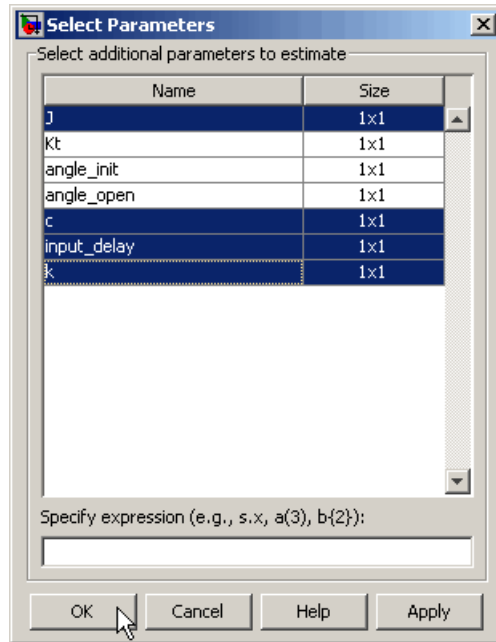


- b Click **Add**.

This action opens the Select Parameters dialog box, which shows the model parameters for the Simulink model.

- c Select the parameters **J**, **c**, **input_delay**, and **k** by pressing the **Ctrl** key while clicking each name, and then click **OK**.

This action adds the selected parameters to the **Estimated Parameters** tab.



Tip When estimating a large number of parameters, you can select a subset of parameters to estimate. To learn more, see the Inverted Pendulum Parameter Estimation demo in the **Demos** pane in the MATLAB Help browser.

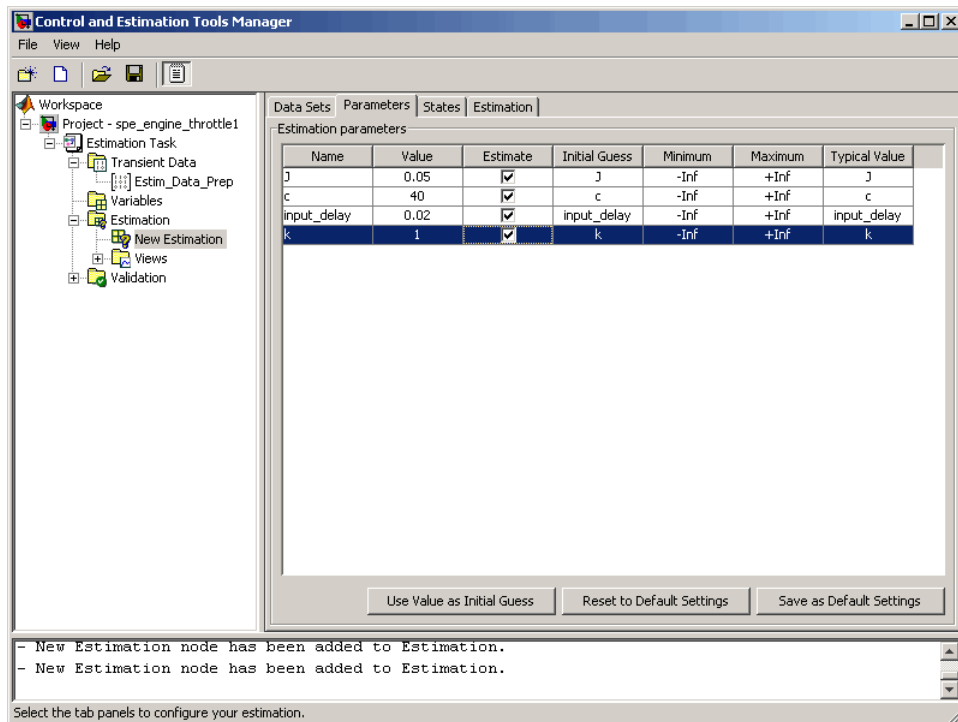
- d** Select the **Estimation** node, and click **New**.

This action adds a **New Estimation** node under the **Estimation** node.

- e** Select the **New Estimation** node.

- f In the **Parameters** tab, select the **Estimate** check box for all the parameters.

This action specifies the selected parameters for estimation.



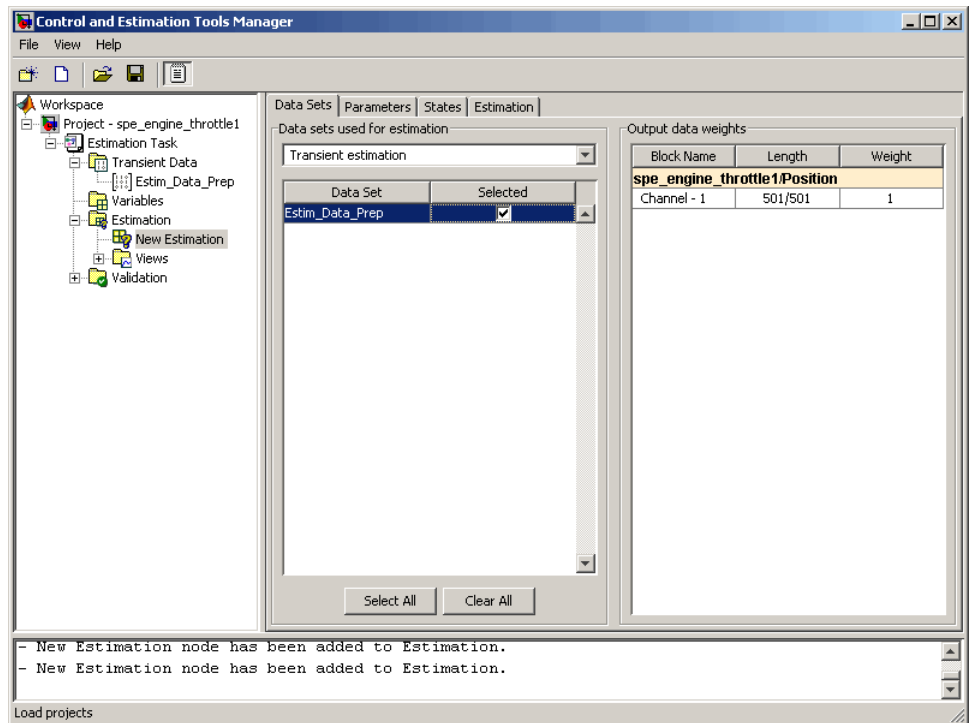
The **Parameters** tab, as shown in the previous figure, displays the following information for each parameter:

- **Value:** Current parameter value. By default, it is equal to the value specified in the Simulink model.

During estimation, the optimization algorithm might change a parameter value to minimize the error between the measured and simulated output.

- **Initial Guess:** Initial parameter value. By default, it is equal to the value in the **Value** field and is used at the start of estimation.

- **Minimum** and **Maximum**: Bounds on the parameter value. By default, they are set to $-\text{Inf}$, and $+\text{Inf}$, respectively. This means that the optimization algorithm searches for a solution in the range $[-\text{Inf}, +\text{Inf}]$.
 - **Typical Value**: Order of magnitude of the parameter value. By default, it is equal to the value in the **Initial Guess** field.
- 3 Specify data for estimation by checking the **Selected** check box for **Estim_Data_Prep** in the **Data Sets** tab.

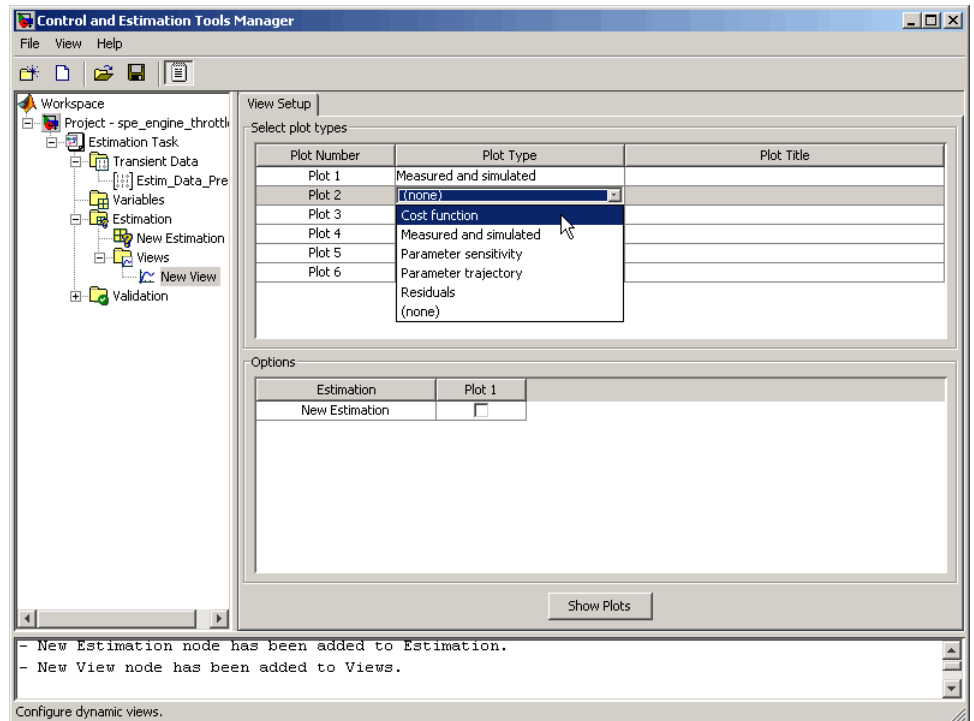


Validating Model Parameters

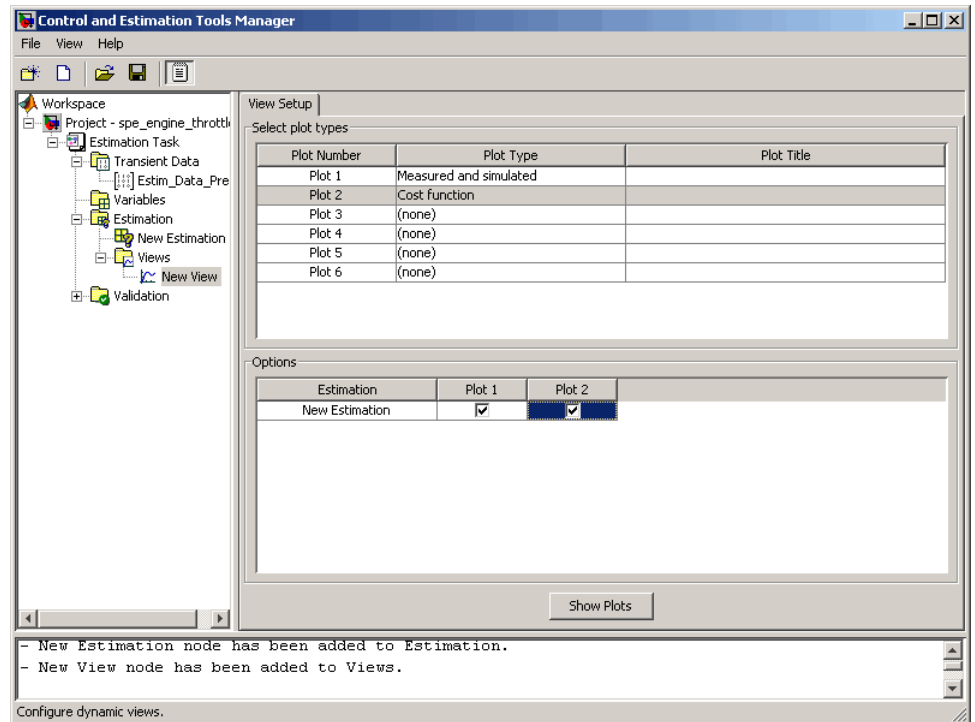
After you specify parameters and data for estimation, as described in “Specifying Parameters and Estimation Data” on page 3-8, estimate the parameters and analyze the results to determine if the model needs to be refined.

- 1 Create plots to view the estimation results.
 - a Select the **Views** node under the **Estimation** node, and click **New**.

This action creates a **New View** node under the **Views** node.
 - b Select the **New View** node.
 - c In the **View Setup** tab, select the following plots from the drop-down list in the **Plot Type** column:
 - For **Plot 1**, select **Measured and simulated**.
 - For **Plot 2**, select **Cost function**.



- d In the **Options** area, select the **Plot 1** and **Plot 2** check boxes, and click **Show Plots**.

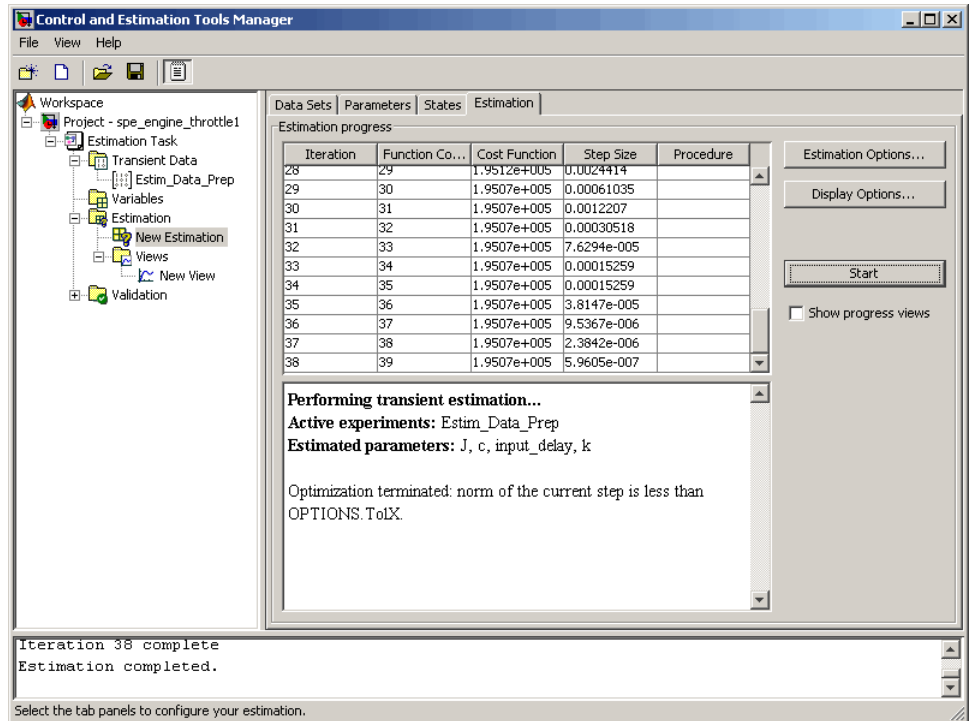


Clicking **Show Plots** opens the following figures:

- **New View - Plot 1 (Measured and simulated):** Displays the measured output data `Estim_Prep_Data`. During estimation, this plot updates to display the simulated response at each iteration.
- **New View - Plot 2 (Cost function):** Displays the error between the measured and simulated output, computed at each iteration. By default, the error is computed using *sum-of-squared-errors*, which is a least-squares method.

- 2 Estimate the parameters by selecting the **Estimation** tab of the **New Estimation** node, and clicking **Start**.

The **Estimation** tab updates at each iteration, and provides information about the estimation progress. When the estimation completes, the **Estimation** tab resembles the next figure.



Note The results of the optimization may differ slightly because of different numerical precision across platforms.

The table displays the following information for each iteration:

- **Cost Function:** Error between the simulated and measured output.

During estimation, the default optimization algorithm `Nonlinear least squares, lsqnonlin`, minimizes the cost function by changing the parameter values. As shown in the previous figure, the cost function decreases by only 27% from 2.69e5 to 1.95e5.

- **Step Size:** Displacement of the optimization algorithm in the current search direction when minimizing the cost function. A final value close to zero indicates that the algorithm has found a local minimum. As shown in the previous figure, the final step size in the last iteration is $5.96e-7$.
- 3 View the estimated parameter values in the **Value** column of the **Parameters** tab.

Control and Estimation Tools Manager

File View Help

Workspace

- Project - spe_engine_throttl
 - Estimation Task
 - Transient Data
 - Estim_Data_Pre
 - Variables
 - Estimation
 - New Estimation
 - Views
 - New View
 - Validation

Data Sets Parameters States Estimation

Estimation parameters

Name	Value	Estimate	Initial Guess	Minimum	Maximum	Typical Value
J	0.0017013	<input checked="" type="checkbox"/>	J	-Inf	+Inf	J
c	39.994	<input checked="" type="checkbox"/>	c	-Inf	+Inf	c
input_delay	0.00016981	<input checked="" type="checkbox"/>	input_delay	-Inf	+Inf	input_delay
k	0.99889	<input checked="" type="checkbox"/>	k	-Inf	+Inf	k

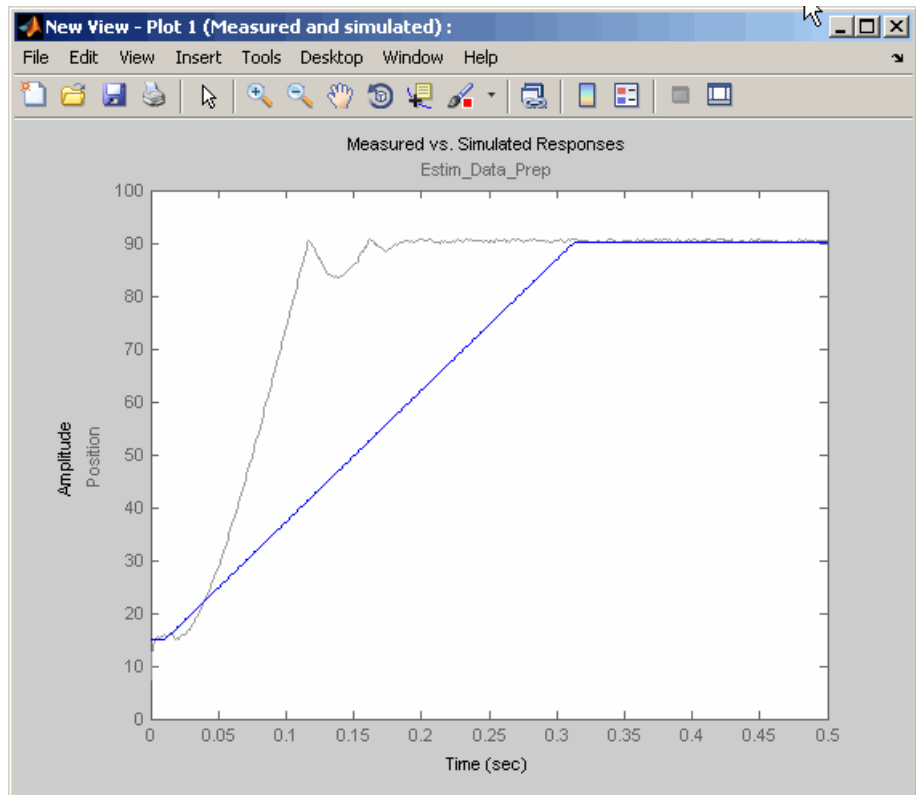
Use Value as Initial Guess Reset to Default Settings Save as Default Settings

Iteration 37 complete
Estimation completed.

Select the tab panels to configure your estimation.

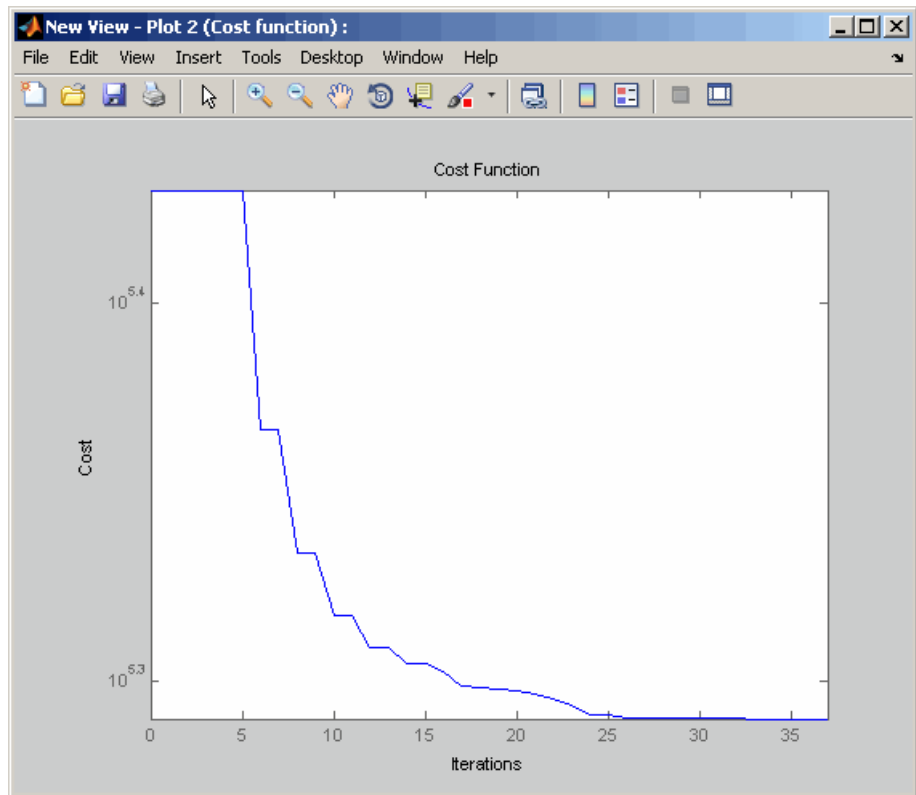
- 4 Examine the simulated response plot to see how well the simulated output matches the measured output.

The simulated response with the estimated parameters is shown in blue on the New View - Plot 1 (Measured and simulated) plot, and significantly differs from the measured data. This difference indicates that the estimated parameters are not accurate.



- 5 Examine the cost function plot to see how the cost function changes during the estimation.

The cost function values shown in step 2 are plotted on the New View - Plot 2 (Cost function) plot. The cost function decreases by only 27% from its initial value, and in conjunction with the measured and simulated output plot, indicates that the estimated parameters are not accurate.



To improve the estimation results, you apply bounds on the parameter values, as described in “Improving Estimation Results Using Parameter Bounds” on page 3-20.

Improving Estimation Results Using Parameter Bounds

In this section...

“Strategy for Improving the Estimation Results” on page 3-20

“How to Specify Parameter Bounds” on page 3-20

Strategy for Improving the Estimation Results

Analyzing the Measured and Simulated and Cost function plots, as described in “Validating Model Parameters” on page 3-13, you see that the model parameters estimated using the default estimation settings are not accurate. You must run additional estimations to improve the accuracy of the model. There are several techniques to improve the accuracy of the estimated parameters. For more information, see “Tuning the Results of Optimization” on page 1-46.

In this portion of the tutorial, you improve the results by specifying bounds on parameter values. This technique restricts the region in which the optimization algorithm searches for a local minima.

Based on physical insight, you know the following characteristics of the engine throttle system:

- All the parameter values are positive.
- Maximum time delay of the system, represented by `input_delay`, is 0.1 s.

Therefore, you specify 0 as the minimum value for all parameters, and 0.1 as the maximum value of `input_delay`.

After you estimate the parameters, analyze the results using the Measured and Simulated and Cost function plots.

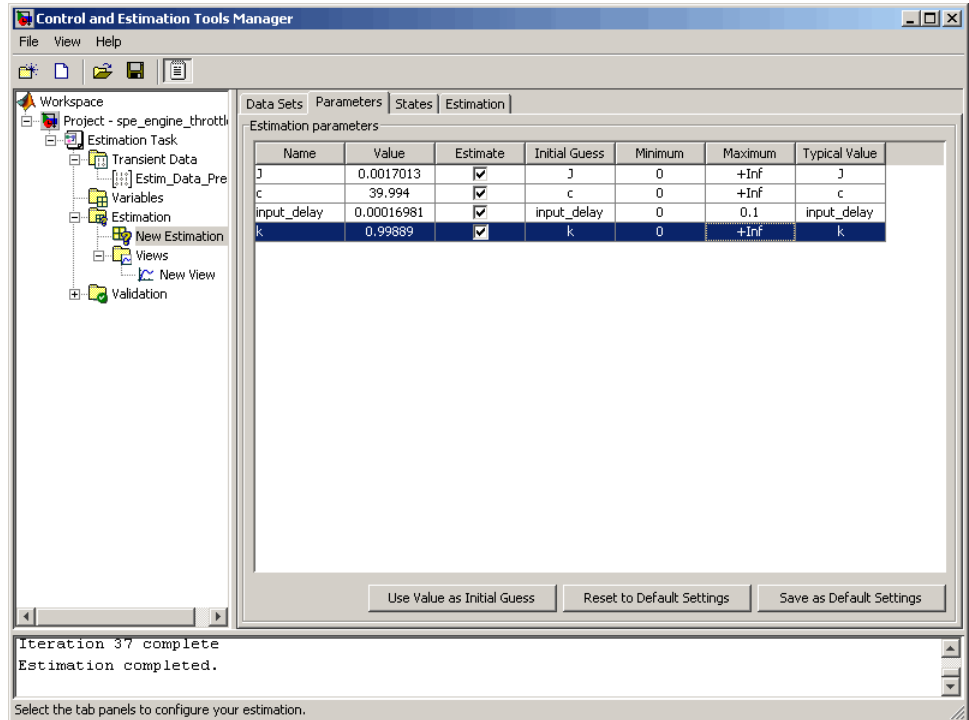
How to Specify Parameter Bounds

You must have already estimated the parameters using default settings, as described in “Estimating Model Parameters Using Default Estimation Settings” on page 3-7.

To improve the estimation results by specifying parameter bounds:

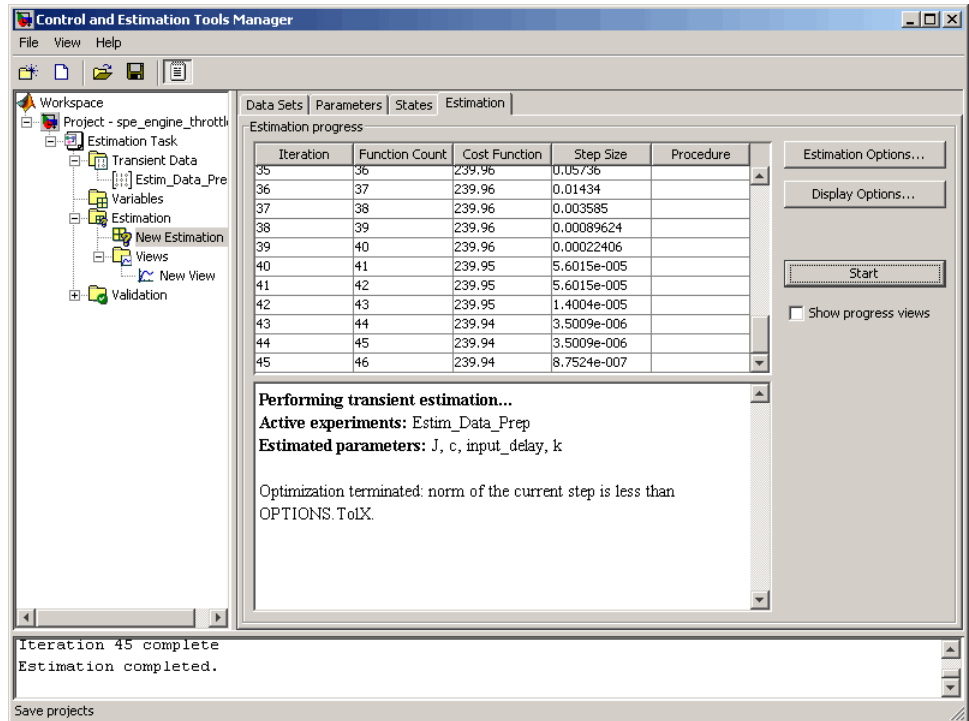
- 1 Select the **Parameters** tab of the **New Estimation** node.
- 2 Specify the minimum value for each parameter by double-clicking the corresponding **Minimum** cell, and replacing $-\text{Inf}$ with 0.
- 3 Specify the maximum value for `input_delay` by double-clicking the corresponding **Maximum** cell and replacing $+\text{Inf}$ with 0.1.

The **Parameters** tab resembles the next figure.



4 Select the **Estimation** tab, and click **Start**.

The **Estimation** tab updates at each iteration, and provides information about the estimation progress. When the estimation completes, the **Estimation** tab resembles the next figure.



Note The results of the optimization may differ slightly because of different numerical precision across platforms.

The cost function decreases by 99% from $1.95e5$ to 239.94. The final step size $8.75e-7$ is close to 0, which indicates that the optimization algorithm has found a local minimum.

- 5 View the estimated parameter values in the **Value** column of the **Parameters** tab.

The screenshot shows the 'Control and Estimation Tools Manager' window. The 'Parameters' tab is active, displaying a table of 'Estimation parameters'. The table has columns for Name, Value, Estimate, Initial Guess, Minimum, Maximum, and Typical Value. The parameters listed are J, c, input_delay, and k. The 'Estimate' column for all parameters has a checkmark, indicating they have been estimated. The 'Value' column shows the estimated values: 0.22295 for J, 9.0178 for c, 0.0090554 for input_delay, and 20.216 for k. The 'Initial Guess' column shows the initial values: J, c, input_delay, and k. The 'Minimum' and 'Maximum' columns show the bounds: 0 and +Inf for J, c, and k, and 0 and 0.1 for input_delay. The 'Typical Value' column shows the typical values: J, c, input_delay, and k.

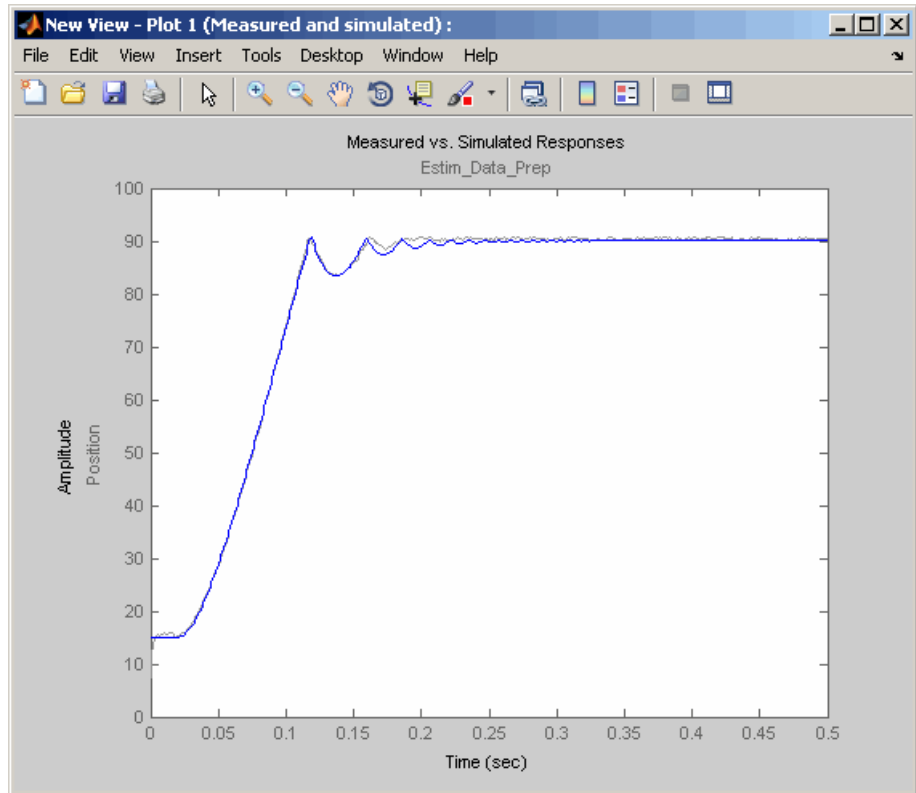
Name	Value	Estimate	Initial Guess	Minimum	Maximum	Typical Value
J	0.22295	<input checked="" type="checkbox"/>	J	0	+Inf	J
c	9.0178	<input checked="" type="checkbox"/>	c	0	+Inf	c
input_delay	0.0090554	<input checked="" type="checkbox"/>	input_delay	0	0.1	input_delay
k	20.216	<input checked="" type="checkbox"/>	k	0	+Inf	k

Iteration 45 complete
Estimation completed.

Select the tab panels to configure your estimation.

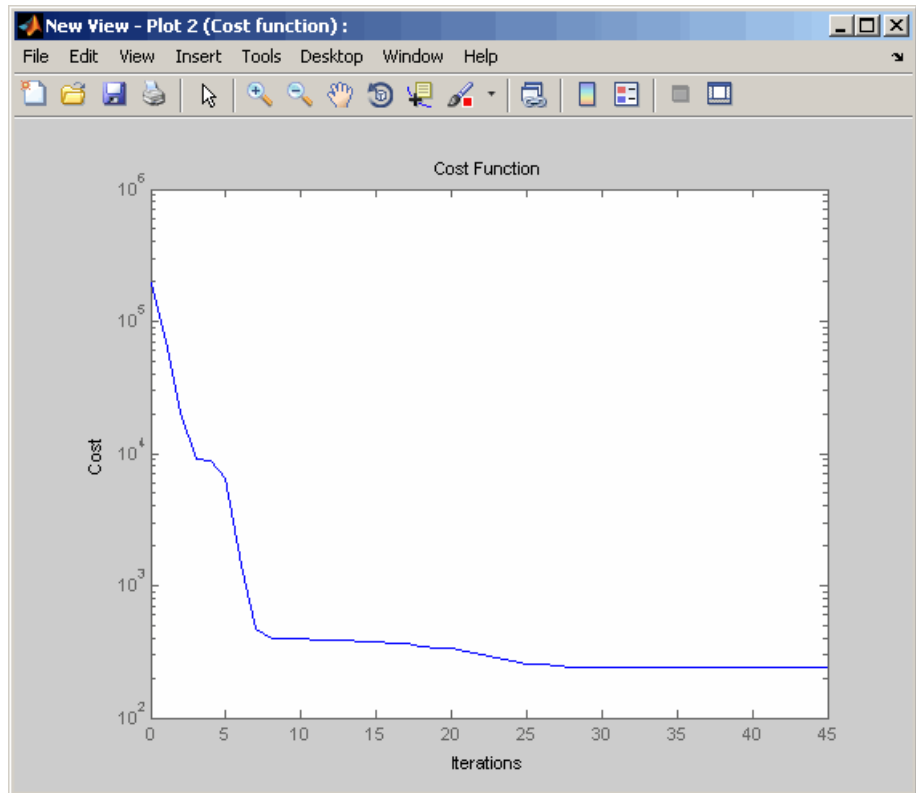
- 6 Examine the simulated response plot to see how well the simulated output matches the measured output.

The simulated response plot, shown in blue on the New View - Plot 1 (Measured and simulated) plot, is overlaid on the measured output data. The simulated output closely matches the measured data.



- 7 Examine the cost function plot to see how the cost function changes during the estimation.

The cost function values shown in step 4 are plotted on the New View - Plot 2 (Cost function) plot. The cost function also decreases by 99% from its initial value, and in conjunction with the measured and simulated response plot indicates a good fit of the simulated response with the measured data.



Validating Estimated Model Parameters

After you estimate the model parameters, as described in “Improving Estimation Results Using Parameter Bounds” on page 3-20, validate the model using another data set (*validation data*). A good match between the simulated response and the validation data indicates that you have not overfitted the model.

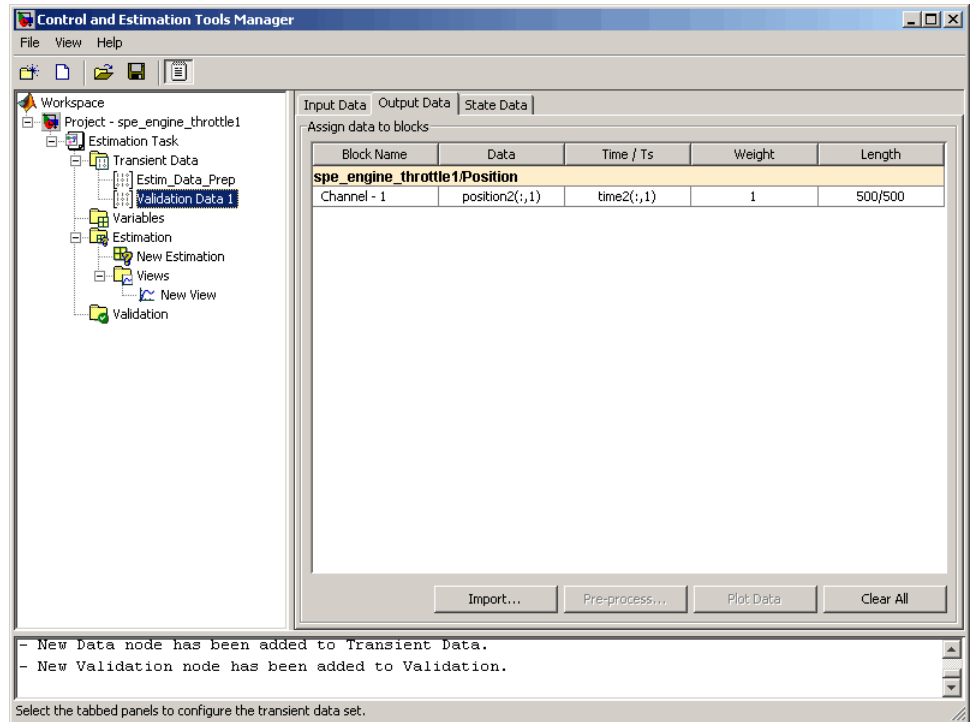
To validate the estimated parameters using a validation data set:

- 1 Load the project `spe_engine_throttle1_importValData1.mat` from the `matlabroot\toolbox\slestim\slestdemos` directory by selecting **File > Load** in the Control and Estimation Tools Manager GUI.

The MAT-file contains a validation data set already imported into the GUI, and the estimated parameters as described in “Improving Estimation Results Using Parameter Bounds” on page 3-20. The validation data contains the input data, output data and time vector in the MATLAB variables `input2`, `position2` and `time2` respectively.

Tip To learn how to import data and time vector into the Control and Estimation Tools Manager GUI, see “Importing Data into the GUI” on page 2-6.

When you expand the **Estimation Task** node, the Control and Estimation Tools Manager GUI resembles the next figure.



2 Plot the measured and simulated response, and residuals.

- a Select the **Validation** node, and click **New**.

This action creates a **New Validation** node.

- b Select the **New Validation** node.

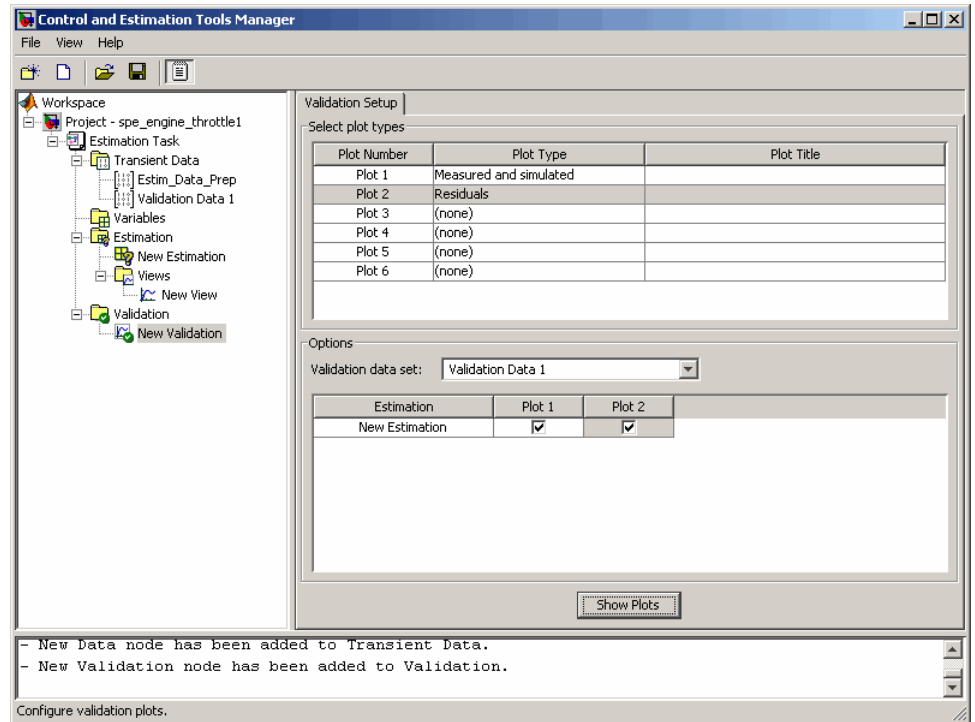
- c In the **Validation Setup** tab, select the following plots in the **Plot Type** column.

- For **Plot 1**, select Measured and simulated.
- For **Plot 2**, select Residuals.

- d In the **Options** area, select the **Plot 1**, and **Plot 2** check boxes.

- e Select Validation Data 1 from the **Validation data set** drop-down list.

The **Validation Setup** tab resembles the next figure.



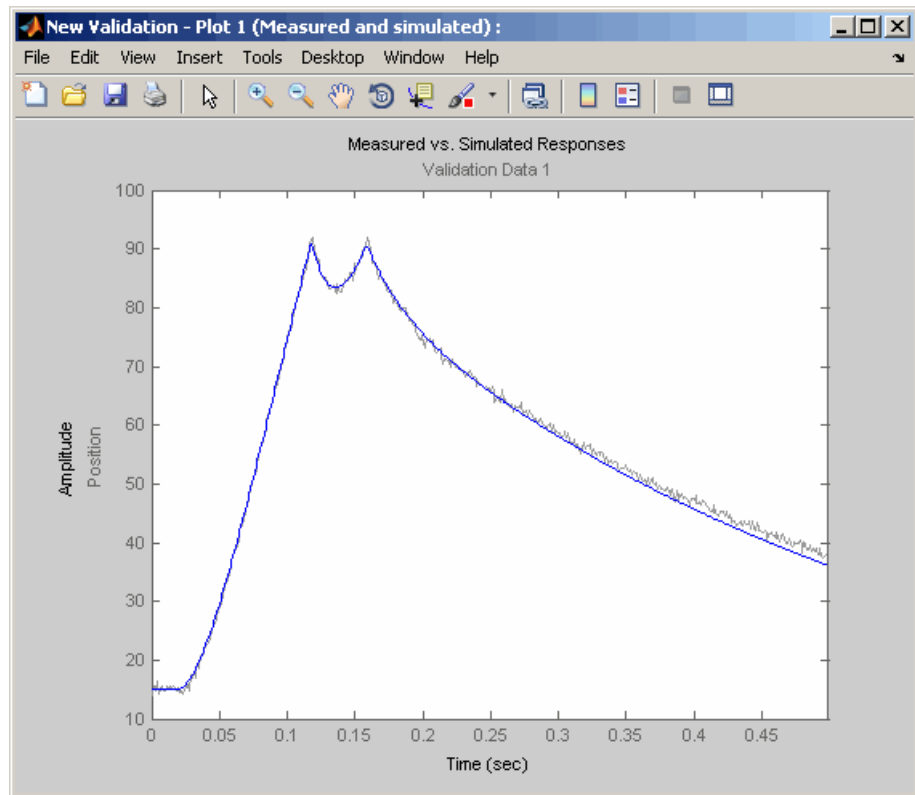
f Click **Show Plots**.

This action opens the following figures:

- New Validation - Plot 1 (Measured and simulated): Displays the validation data and simulated response.
- New Validation - Plot 2(Residuals): Displays the difference between the measured data and simulated response.

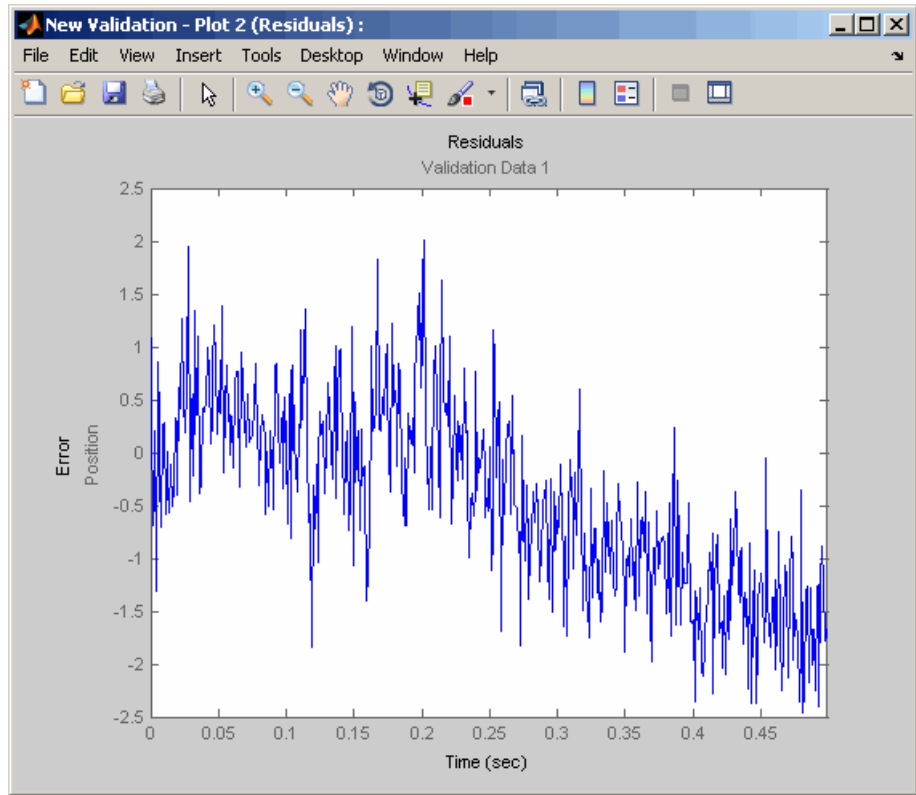
3 Examine the simulated response plot to see how well the simulated output matches the validation data.

The simulated response, as shown in blue on the New Validation - Plot 1 (Measured and simulated) plot, is overlaid on the measured output data, and closely matches the measured data Validation Data 1.



- 4 Examine the residuals plot to compare the difference between the simulated response and measured data.

The difference between the simulated and measured data, as shown in the New Validation - Plot 2(Residuals) plot, varies between 2 and -2.5. The residuals lie within 6% of the maximum output variation, and do not display any systematic patterns. This indicates a good fit between the simulated output and measured data.



5 Save the Control and Estimation Tools Manager project.

a In the Control and Estimation Tools Manager GUI, select **File > Save**.

This action opens the Save Projects dialog box.

b In the Save Projects dialog box, click **OK**.

c In the Save Projects window, specify the name of the project in the **File name** field, and click **Save**.

This action saves the project as a MAT-file.

Tip You can load a project by selecting **File > Load** in the Control and Estimation Tools Manager GUI. To view the estimated parameter values, select the **Parameters** tab of **New Estimation** node.

Tutorial — Modeling a System Using Adaptive Lookup Table

- “About This Tutorial” on page 4-2
- “Building a Model Using Adaptive Lookup Table Blocks” on page 4-4
- “Adapting the Lookup Table Values Using Time-Varying I/O Data” on page 4-16

About This Tutorial

In this section...
“Objectives” on page 4-2
“About the Data” on page 4-2
“Overview of Modeling a System Using Adaptive Lookup Table” on page 4-3

Objectives

In this tutorial, you learn how to capture the time-varying behavior of an engine using an n-D adaptive lookup table. You accomplish the following tasks using the Simulink software:

- Configure an adaptive lookup table block to model your system.
- Simulate the model to update the lookup table values dynamically.
- Export the adapted lookup table values to the MATLAB workspace.
- Disable the adaptation process and use the adaptive lookup table as a static lookup table.

About the Data

In this tutorial, you use the data in `vedata.mat` which contains the following variables measured from the engine:

- `X` — 10 input breakpoints for intake manifold pressure in the range [10,100]
- `Y` — 36 input breakpoints for engine speed in the range [0,7000]
- `Z` — 10x36 matrix of table data for engine volumetric efficiency

To learn more about breakpoints and table data, see “Anatomy of a Lookup Table” in the Simulink documentation.

The output volumetric efficiency of the engine is time varying, and a function of two inputs—intake manifold pressure and engine speed. The data in the MAT-file is used to generate the time-varying input and output (I/O) data for the engine.

Overview of Modeling a System Using Adaptive Lookup Table

The process for modeling a system using adaptive lookup table consists of the following tasks:

- “Building a Model Using Adaptive Lookup Table Blocks” on page 4-4.
- “Adapting the Lookup Table Values Using Time-Varying I/O Data” on page 4-16.

Simulink Parameter Estimation software provides blocks for modeling systems as adaptive lookup tables. You build a model using the adaptive lookup table blocks, and then simulate the model to adapt the lookup table values to the time-varying I/O data. During simulation, the software uses the input data to locate the table values, and then uses the output data to recalculate the table values. The updated table values are stored in the adaptive lookup table block. To learn more about adaptive lookup tables, see “Adaptive Lookup Tables” on page 8-3.

Building a Model Using Adaptive Lookup Table Blocks

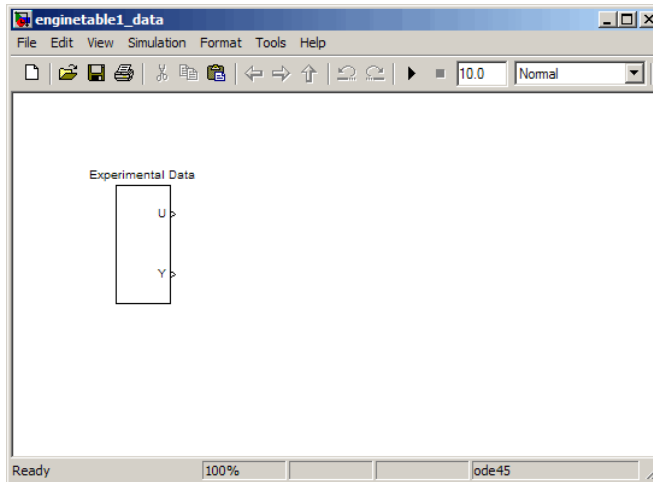
In this portion of the tutorial, you learn how to build a model of an engine using an Adaptive Lookup Table block.

- 1 Open a preconfigured Simulink model by typing the model name at the MATLAB prompt:

```
enginetable1_data
```

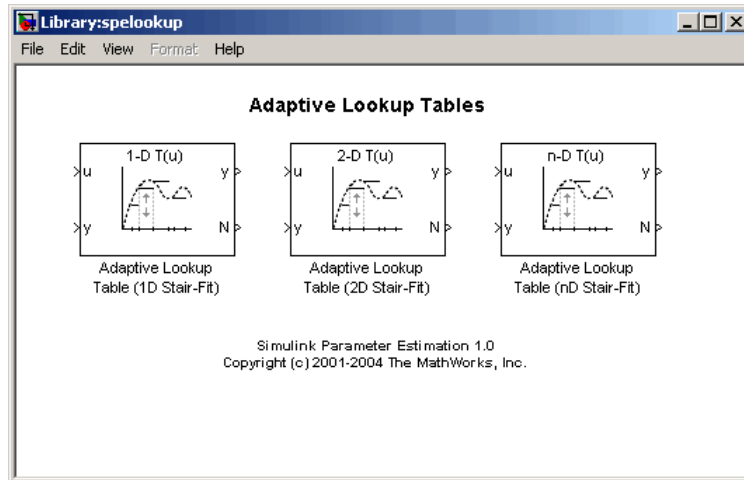
This command also loads the variables X, Y and Z into the MATLAB workspace. To learn more about this data, see “About the Data” on page 4-2.

The Experimental Data subsystem in the Simulink model generates the time-varying I/O data during simulation.



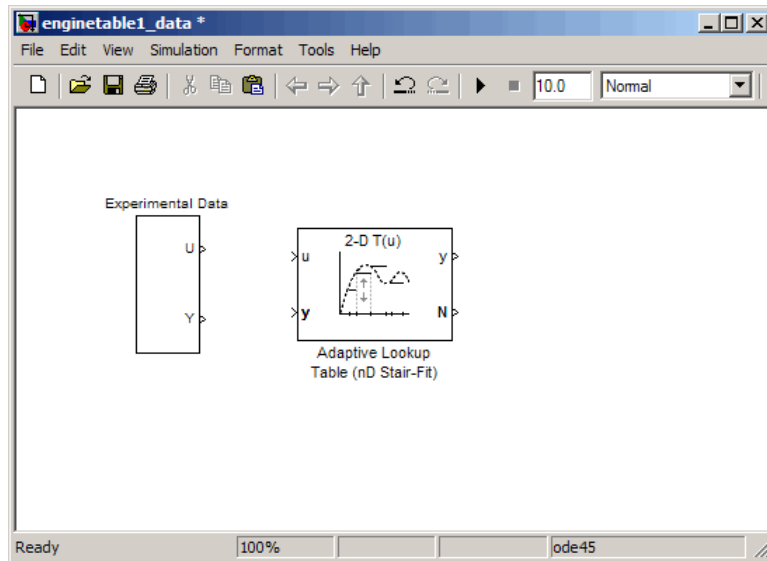
- 2 Open the Adaptive Lookup Table library by typing the following command at the MATLAB prompt:

```
spelookup
```

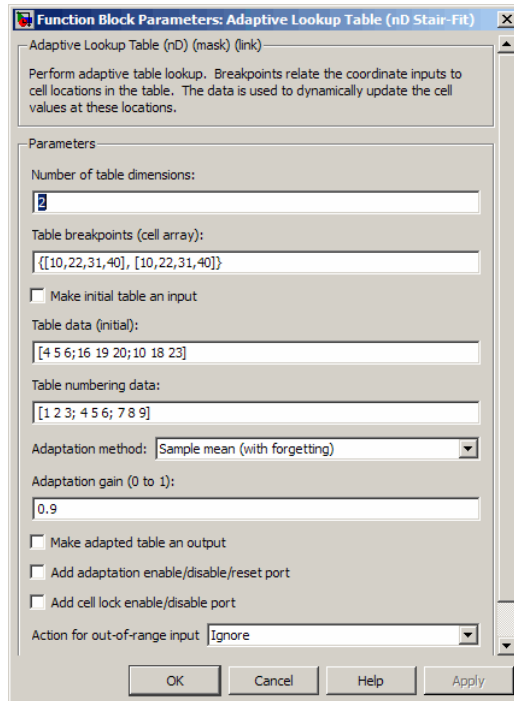


Simulink Parameter Estimation software provides three Adaptive Lookup Table blocks. In this tutorial, you use the Adaptive Lookup Table (nD Stair-Fit) block to model your system. To learn more about the blocks, see Chapter 10, “Block Reference”.

- 3 Drag and drop the Adaptive Lookup Table (nD Stair-Fit) block from the Simulink Parameter Estimation library to the Simulink model window.



- 4 Double-click the Adaptive Lookup Table (nD Stair-Fit) block to open the Function Block Parameters: Adaptive Lookup Table (D Stair-Fit) dialog box.



- 5 In the Function Block Parameters dialog box:

- a Specify the following block parameters:
 - **Table breakpoints (cell array)** — Enter `{[X; 110], [Y; 7200]}` to specify the range of input breakpoints.
 - **Table data (initial)** — Enter `rand(10,36)` to specify random numbers as the initial table values for the volumetric efficiency.
 - **Table numbering data** — Enter `reshape(1:360,10,36)` to specify a numbering scheme for the table cells.
- b Verify that **Sample mean (with forgetting)** is selected in the **Adaptation method** drop-down list.

- c** Enter 0.98 in the **Adaptation gain (0 to 1)** field to specify the *forgetting factor* for the Sample mean (with forgetting) adaptation algorithm.

An adaptation gain close to 1 indicates high robustness of the lookup table values to input noise. To learn more about the adaptation gain, see “Sample Mean with Forgetting” on page 8-42 in “Selecting an Adaptation Method” on page 8-41.

- d** Select the **Make adapted table an output** check box.

This action adds a new port named `Tout` to the Adaptive Lookup Table block. You use this port to plot the table values as they are being adapted.

- e** Select the **Add adaptation enable/disable/reset port** check box.

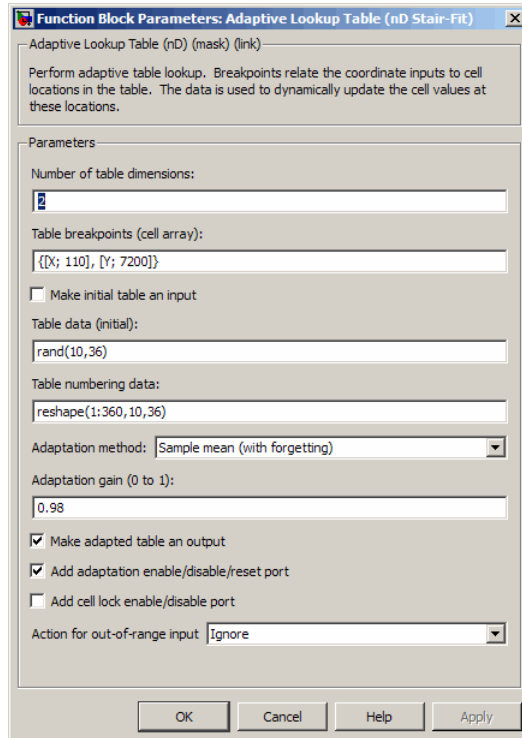
This action adds a new port named `Enable` to the Adaptive Lookup Table block. You use this port to enable or disable the adaptation process.

- f** Verify that `Ignore` is selected in the **Action for out-of-range** drop-down list.

This selection specifies that the software ignores any time-varying inputs outside the range of input breakpoints during adaptation.

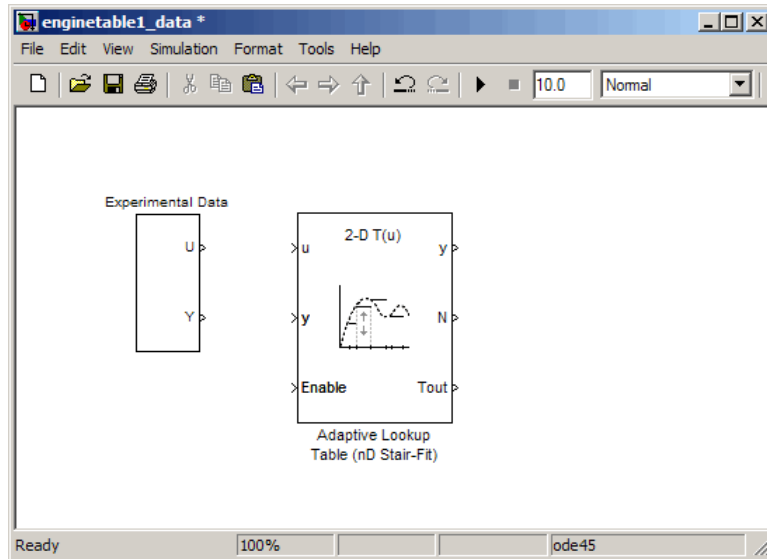
Tip To learn more about the Adaptive Lookup Table (nD Stair-Fit) block parameters, see the Adaptive Lookup Table (nD Stair-Fit) block reference page.

After you configure the parameters, the block parameters dialog box looks like the following figure.

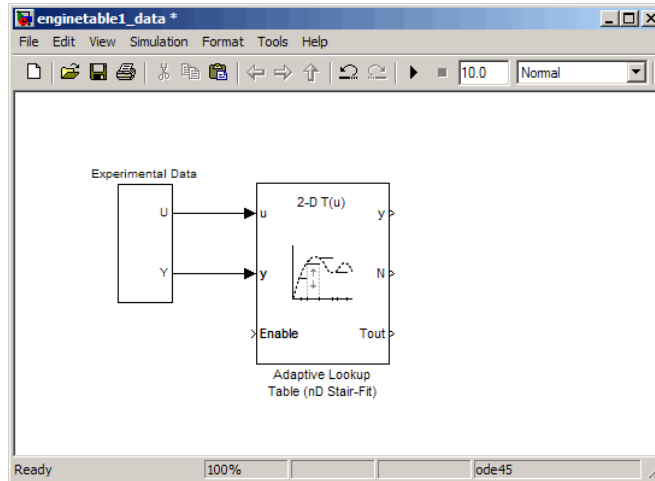


- 6 Click **OK** to close the Function Block Parameters dialog box.

The Simulink model now looks similar to the following figure.



- 7 Assign the input and output data to the engine model by connecting the U and Y ports of the Experimental Data block to the u and y ports of the Adaptive Lookup Table block, respectively.



Tip To learn how to connect blocks in the Simulink model window, see “Connecting Blocks in the Model Window” in the Simulink documentation.

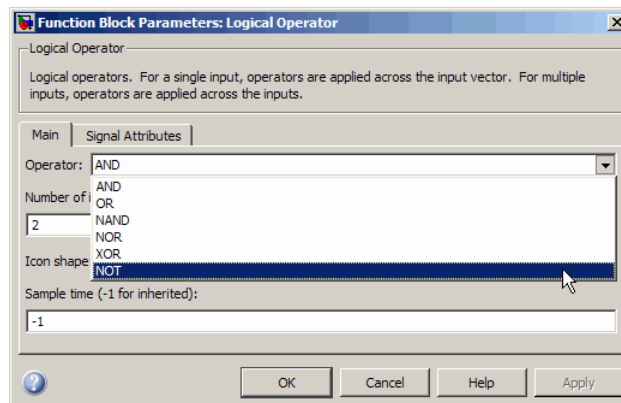
- 8 Design a logic that enables or disables the adaptation process:
- Open the Simulink Library Browser by typing `simulink` at the MATLAB prompt.
 - In the **Libraries** area of the Simulink Library Browser window, select **Commonly Used Blocks**.
 - Drag the following blocks to the Simulink model window:
 - Constant block
 - Logical Operator block

To learn more about the blocks, see the Constant, and Logical Operator block reference pages in the Simulink documentation.

- d** In the **Libraries** area of the Simulink Library Browser window, select **Signal Routing**, and drag the Manual Switch block to the Simulink model window.

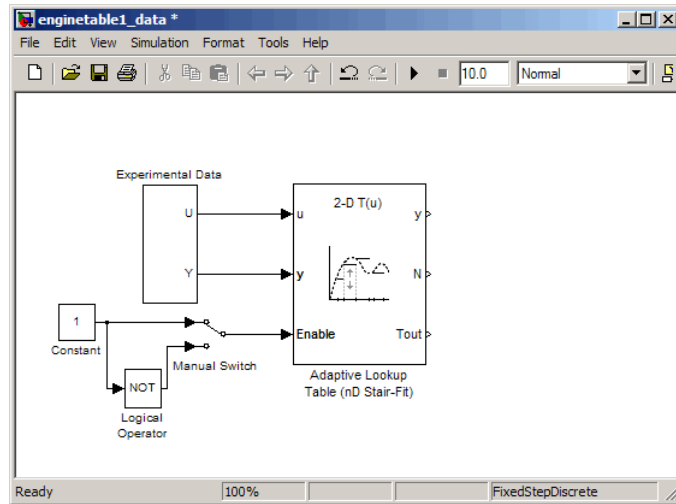
To learn more about the block, see the Manual Switch block reference page in the Simulink documentation.

- e** Double-click the Logical Operator block to open the Function Block Parameters: Logical Operator dialog box.
- f** Select **NOT** from the **Operator** drop-down list.



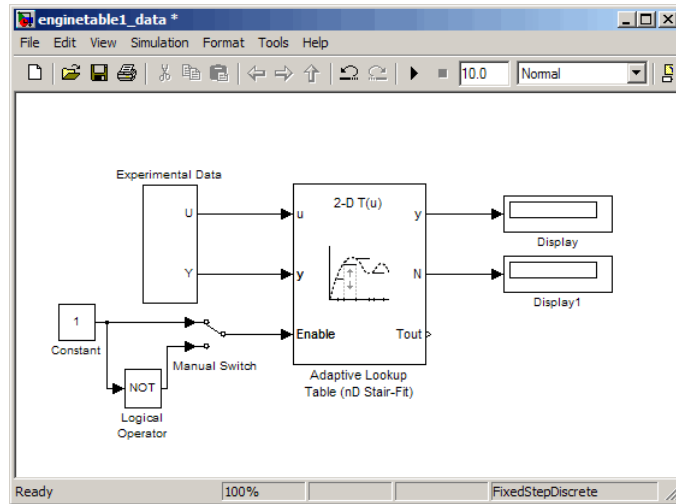
- g** Click **OK**.

- h** Connect the blocks to create a logic, as shown in the next figure.



This logic outputs an initial value of 1 which enables the adaptation process.

- 9** Add Display blocks to view the output of the adaptation process:
- a** In the **Libraries** area of the Simulink Library Browser window, select **Sinks**, and drag the Display block to the Simulink model window.
 - b** Drag another Display block from the Simulink Library Browser window to the model window.
 - c** Connect the Display blocks to the y and N ports of the Adaptive Lookup Table block, as shown in the next figure.



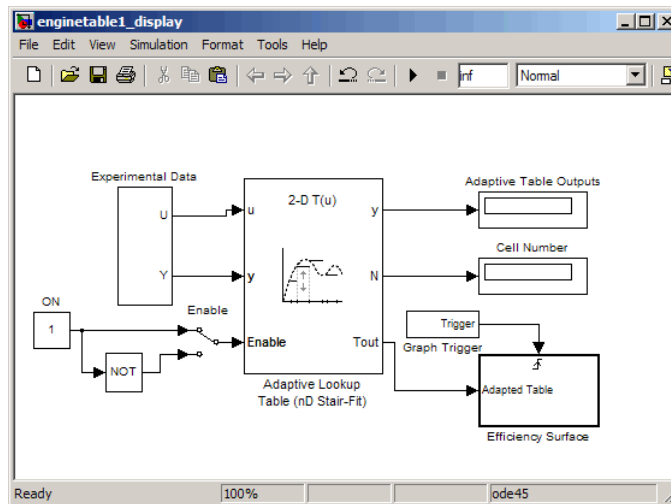
During simulation, the Display blocks show the following:

- Display block — Shows the value of the current cell being adapted.
- Display1 block — Shows the number of the current cell being adapted.

- 10** Open a preconfigured Simulink model that already contains a plot to display the lookup table values as they adapt during simulation. To do so, type the following model name at the MATLAB prompt:

```
enginetable1
```

The model opens, as shown in the next figure.



This model already contains a subsystem named Efficiency Surface that generates a plot of the lookup table values as they adapt during simulation.

Tip You can also write your own MATLAB function to plot the adapted table values during simulation.

You have now configured the adaptive lookup table parameters and built the model for updating and viewing the adaptive lookup table values. You must now simulate the model to start the adaptation, as described in “Adapting the Lookup Table Values Using Time-Varying I/O Data” on page 4-16.

Adapting the Lookup Table Values Using Time-Varying I/O Data

In this portion of the tutorial, you learn how to update the lookup table values to adapt to the time-varying input and output values.

You must have already built the Simulink model, as described “Building a Model Using Adaptive Lookup Table Blocks” on page 4-4. To open a saved model that already contains the block parameters and display blocks, type the following model name at the MATLAB prompt:

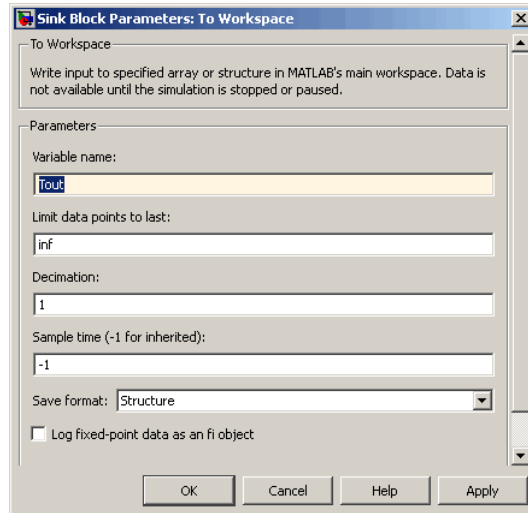
```
enginetable1
```

To perform the adaptation:

- 1 Connect a To Workspace Simulink block to export the adapted table values:
 - a Open the Simulink Library Browser, if it is not already open, by typing `simulink` at the MATLAB prompt.
 - b In the **Libraries** area of the Simulink Library Browser window, select **Sinks**, and drag the To Workspace block to the `enginetable1` Simulink model window.

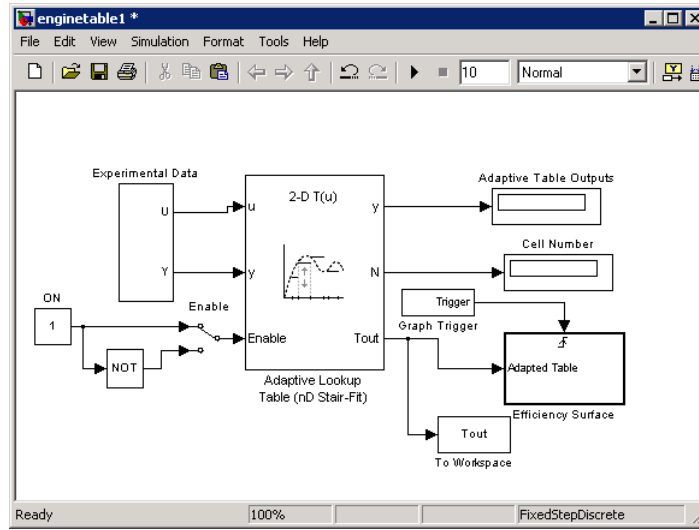
To learn more about this block, see the To Workspace block reference page in the Simulink documentation.

- c Double-click the To Workspace block to open the Sink Block Parameters dialog box, and type **Tout** in the **Variable name** field.

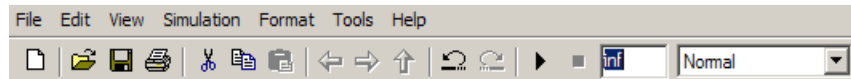


- d Click **OK**.

- e Connect the To Workspace block to the adaptive lookup table output signal Tout, as shown in the next figure.



- 2 In the Simulink model window, enter `inf` as the simulation time.

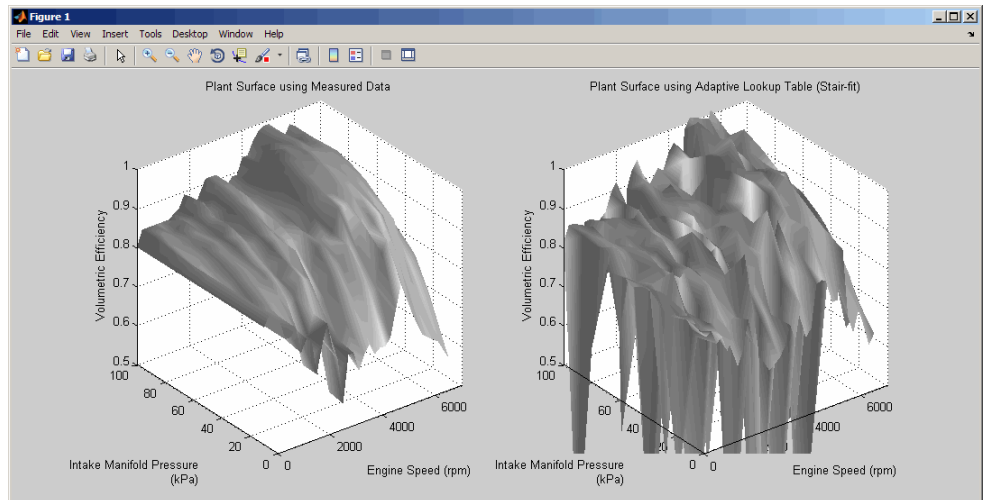


The simulation time of infinity specifies that the adaptation process continues as long as the input and output values of the engine change.

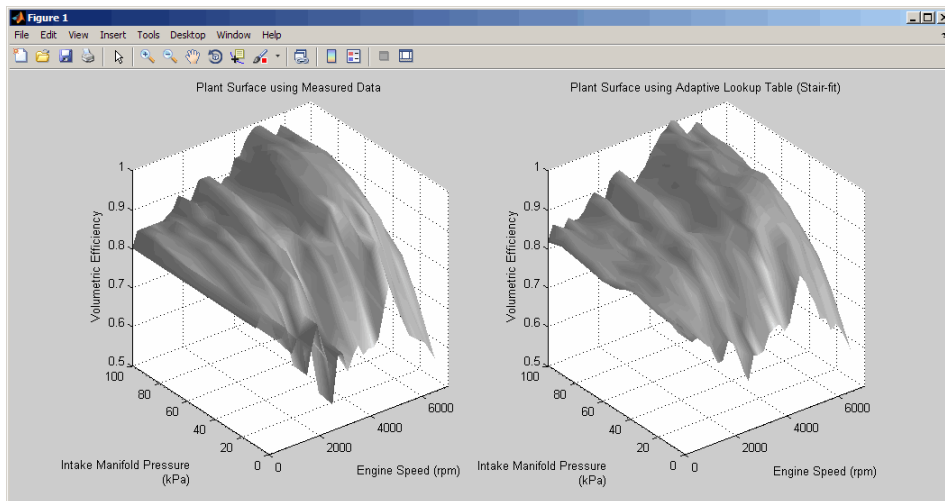
- 3** In the Simulink model window, select **Simulation > Start** to start the adaptation process.

A figure window opens that shows the volumetric efficiency of the engine as a function of the intake manifold pressure and engine speed:

- The left plot shows the measured volumetric efficiency as a function of intake manifold pressure and engine speed.
- The right plot shows the volumetric efficiency as it adapts with the time-varying intake manifold pressure and engine speed.



During simulation, the lookup table values displayed on the right plot adapt to the variations in the I/O data. The left and the right plots resemble each other after a few seconds, as shown in the next figure.




Tip During simulation, the **Cell Number** and **Adaptive Table Outputs** blocks in the Simulink model display the cell number, and the adapted lookup table value in the cell, respectively.

- 4 Examine that the left and the right plots match. This resemblance indicates that the table values have adapted to the time-varying I/O data.
- 5 Pause the simulation by selecting **Simulation > Pause**.

This action also exports the adapted table values T_{out} to the MATLAB workspace.

Note After you pause the simulation, the adapted table values are stored in the Adaptive Lookup Table block.

- 6 After the table values adapt to the time-varying I/O data, you can continue to use the Adaptive Lookup Table block as a static lookup table:
 - a In the Simulink model window, double-click the Manual Switch block.

This action toggles the switch to , and disables the adaptation.

- b** Select **Simulation > Start** to restart the simulation.

During simulation, the Adaptive Lookup Table block works like a static lookup table, and continues to estimate the output values as the input values change. You can see the current lookup table value in the Adaptive Table Outputs block in the Simulink model window.

Note After you disable the adaptation, the Adaptive Lookup Table block does not update the stored table values, and the figure that displays the table values does not update.

Estimating Initial Conditions

- “Why Estimate Initial Conditions?” on page 5-2
- “Estimating Initial Conditions for Blocks with External Initial Conditions” on page 5-3
- “Example: Estimating Initial Conditions of a Mass-Spring-Damper System” on page 5-4

Why Estimate Initial Conditions?

Often, sets of measured data are collected at various times and under different initial conditions. When you estimate model parameters using one data set and subsequently run another estimation with a second data set, your parameter values may not match. Given that the Simulink Parameter Estimation software attempts to find constant values for parameters, this is clearly a problem.

You can estimate the initial conditions using procedures that are similar to those you use to estimate parameters. You can then use these initial condition estimates as a basis for estimating parameters for your Simulink model. The Control and Estimation Tools Manager has an **Estimated States** pane that lists the states available for initial condition estimation.

This chapter focuses on the steps required to estimate initial conditions, and then estimate the parameters from these initial conditions.

Estimating Initial Conditions for Blocks with External Initial Conditions

When an integrator block uses an initial-condition port, which you specify by an IC block feeding into the integrator block, you cannot estimate the initial conditions (ICs) of the integrator using Simulink Parameter Estimation software. This is because external ICs have priority over the ICs of a specific block to maintain the integrity of the model.

To tune the ICs of an integrator block with external ICs, you must modify the model to make the external signal into a tunable parameter. For example, you can set the IC block that feeds into the integrator to be a tunable variable and estimate it.

Example: Estimating Initial Conditions of a Mass-Spring-Damper System

In this section...
“Loading the Example” on page 5-4
“Model Parameters” on page 5-5
“Setting Up the Estimation Project” on page 5-6
“Importing Transient Data and Selecting Parameters for Estimation” on page 5-7
“Selecting Parameters and Initial Conditions for Estimation” on page 5-8
“Creating the Estimation Task” on page 5-10
“Running the Estimation and Viewing Results” on page 5-11

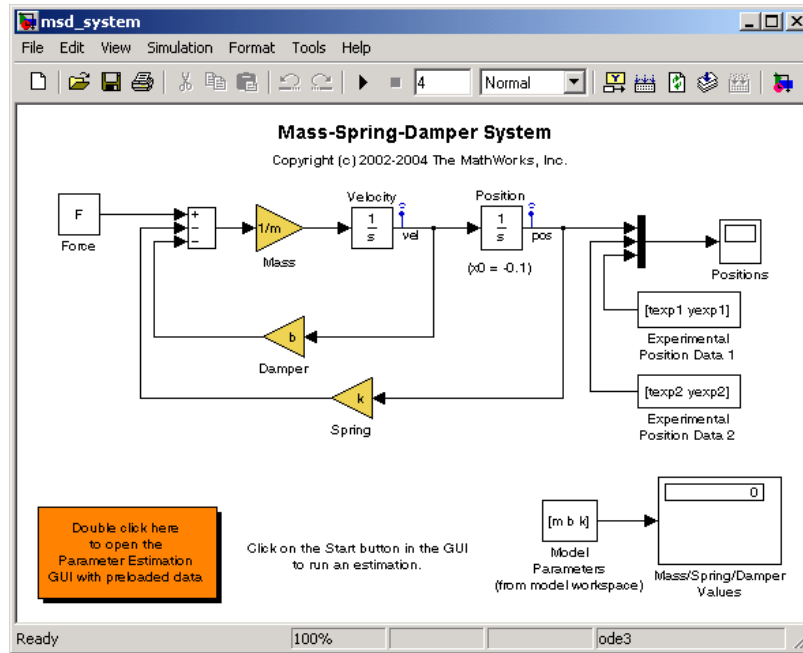
Loading the Example

To open the Simulink model of a mass-spring-damper system and two sets of model data with differing initial conditions, type:

```
msd_system
```

at the MATLAB prompt.


The figure shown next is a model of a mass-spring-damper system.

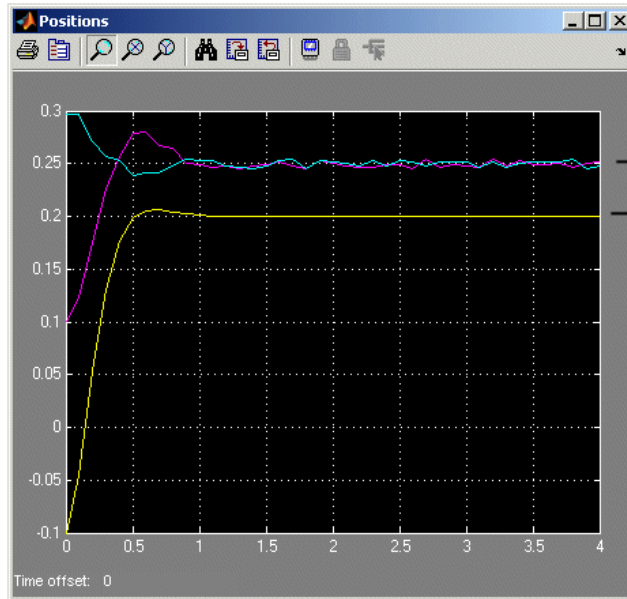


You can run the demo from **Simulink > Simulink Parameter Estimation** on the **Demo** pane of the Help browser.

This example goes beyond what is included in the Simulink Parameter Estimation demo that uses this model by providing in-depth discussion of each task.

Model Parameters

The Simulink `msd_system` model's output is the displacement (or position) of the mass in a mass-spring-damper system, subject to a constant force F , and an initial condition, x_0 , for the mass displacement. x_0 is indicated by the initial condition of the Position integrator block. Click the **Start Simulation** button  to run the simulation once and observe the response of the model to two sets of parameter values.



Magenta and cyan lines are empirical responses for data with different initial conditions.

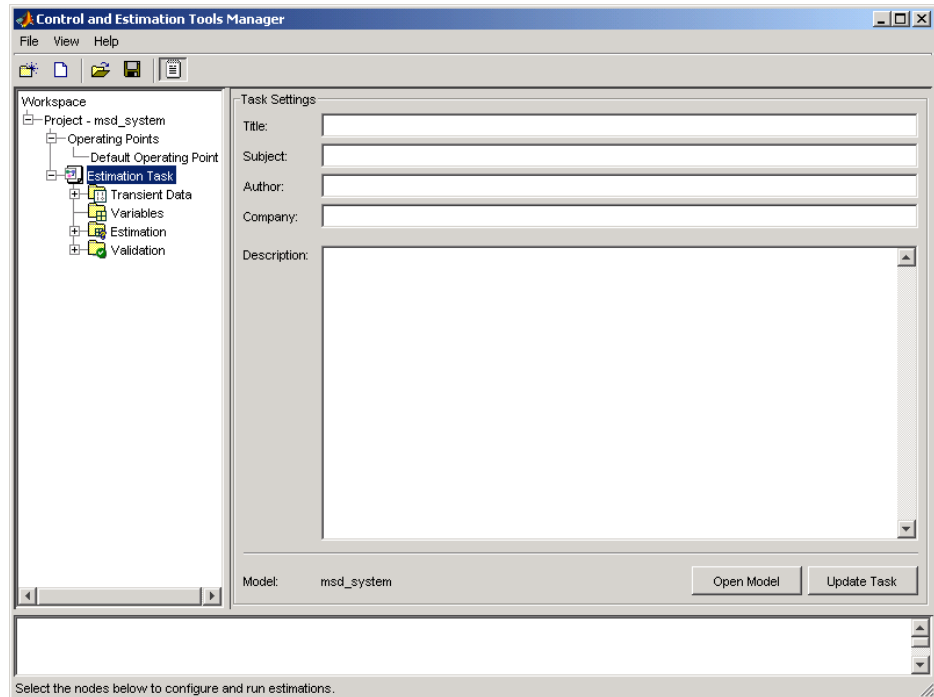
Yellow line is the response of the model to a constant force.

The model parameters of interest are the mass, m , the viscous damping, b , and the spring constant, k . For more information about physical modeling of mass-spring-damper systems, see any elementary book on mathematical modeling or on automatic control systems.

For the estimation of the model parameters m , b , and k , this model uses two sets of experimental data. These data sets were obtained using two different initial positions, $x_0=0.1$ and $x_0=0.3$, and also contain additive noise. A plot of these data sets is shown in the figure above (top curves), along with the simulated response (bottom curve) of the Simulink model `msd_system` for $x_0=-0.1$ and a nominal set of parameter values, $m=8$, $k=500$, and $b=100$.

Setting Up the Estimation Project

To set up the estimation of initial conditions and then transient state space data, select **Tools > Parameter Estimation** in the `msd_system` model window.



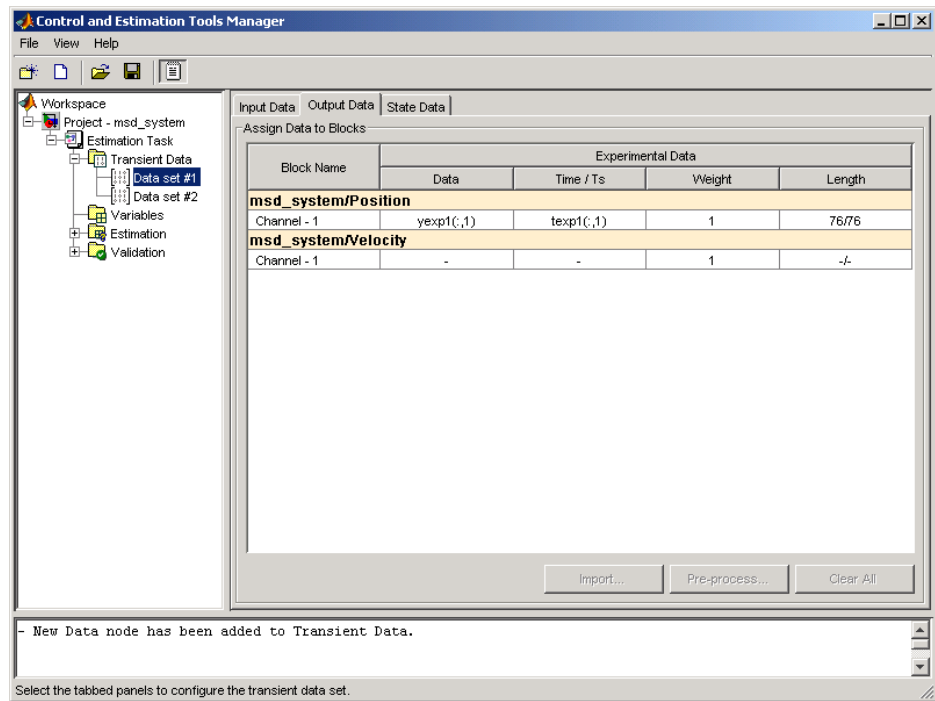
Importing Transient Data and Selecting Parameters for Estimation

The process for importing transient data and selecting parameters for estimation is discussed in “Importing Transient Data” on page 1-11 and “Selecting Parameters for Estimation” on page 1-17.

- 1** In the Control and Estimation Tools Manager, select **Estimation Task** > **Transient Data** in the workspace directory tree.
- 2** Right-click **Transient Data** and select **New** to add a new data set.
- 3** Right-click the **New Data** node in the workspace directory tree and select **Edit** to open the **Input Data**, **Output Data**, and **State Data** panes.
- 4** In the **Output Data** pane, click **Import** and add **yexp1** to the **Data** column and **texp1** to the **Time/Ts** column of the **msd_system/Position** state.

- 5 If you like, right-click **New Data** in the workspace directory tree and rename it to **Data set #1**.
- 6 Repeat steps 1 to 5 to add a second data set, **yexp2** and **texp2**, and rename it to **Data set #2**.

Your Control and Estimation Tools Manager should resemble this figure:

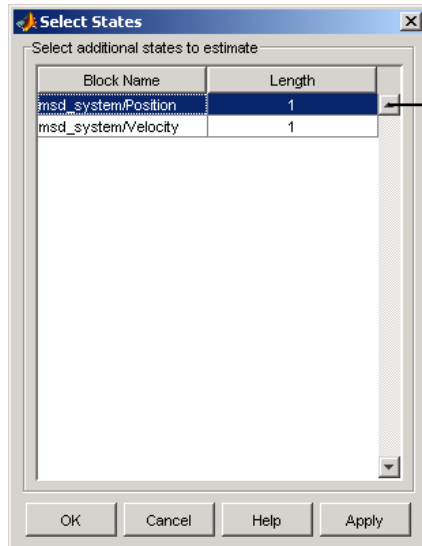


Selecting Parameters and Initial Conditions for Estimation

First, select the parameters you want to estimate for the Simulink `msd_system` model. In this case, select b , k , and m . To do this:

- 1 Select the **Variables** node in the workspace directory tree of the Control and Estimation Tools Manager.
- 2 Click the **Estimation Parameters** tab.

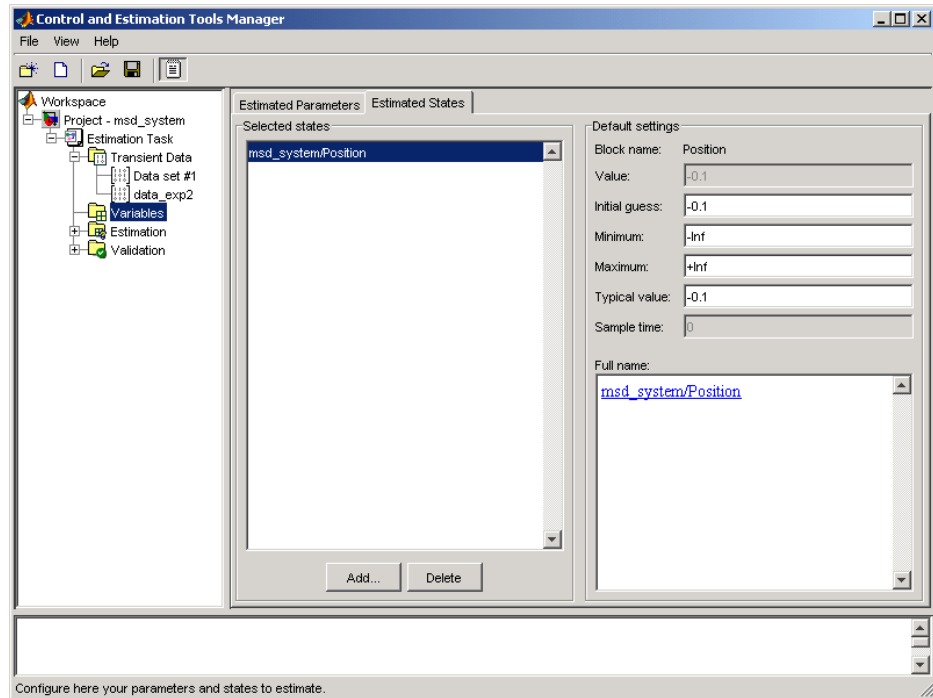
- 3 Click **Add** to open the Select Parameters dialog box.
- 4 Select the parameters b , k , and m , and then click **OK**.
- 5 Do the same with the **Estimation States** pane, and select `msd_system/Position` from the Select States dialog box.



Select states with initial conditions that you want to estimate.

Hold down **Shift** and use your mouse to select groups of adjacent states. Hold down **Ctrl** and use your mouse to select nonadjacent states.

Your Control and Estimation Tools Manager should look like this.

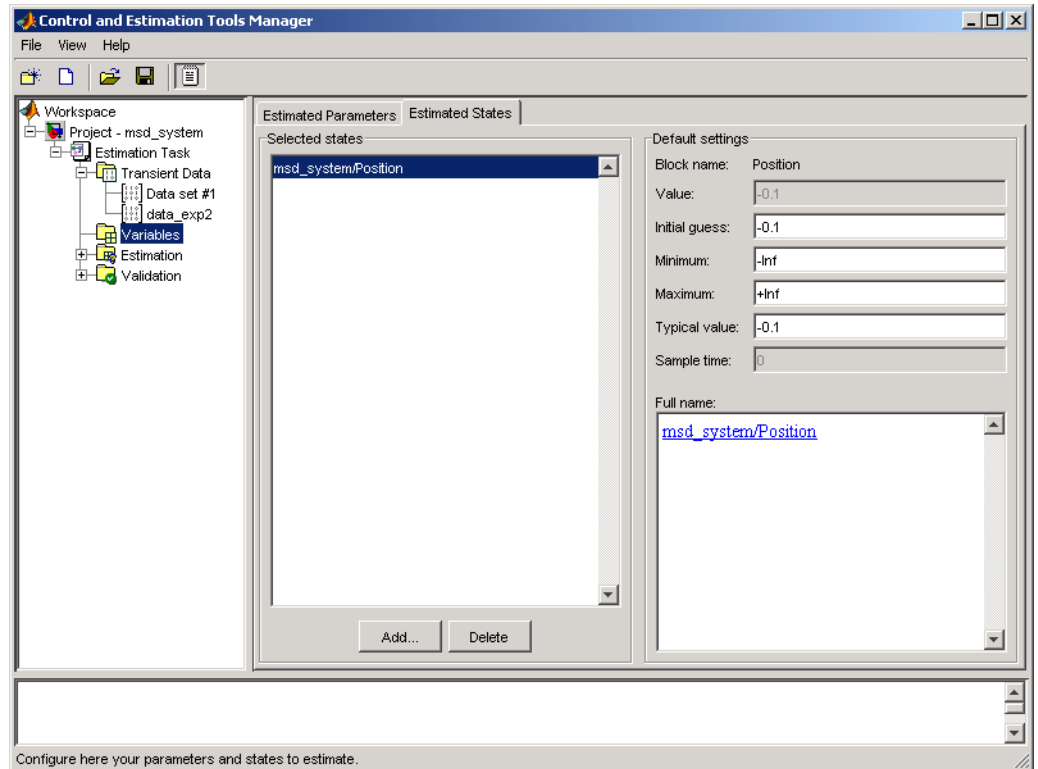


Creating the Estimation Task

To create the **New Estimation** task in the Control and Estimation Tools Manager, right-click the **Estimation** node in the workspace directory tree and select **Add**. While the initial velocity is also a state of the model, assume (for simplicity) that it is known to be 0. The estimation task for this case is **Estim** (with **IC**).

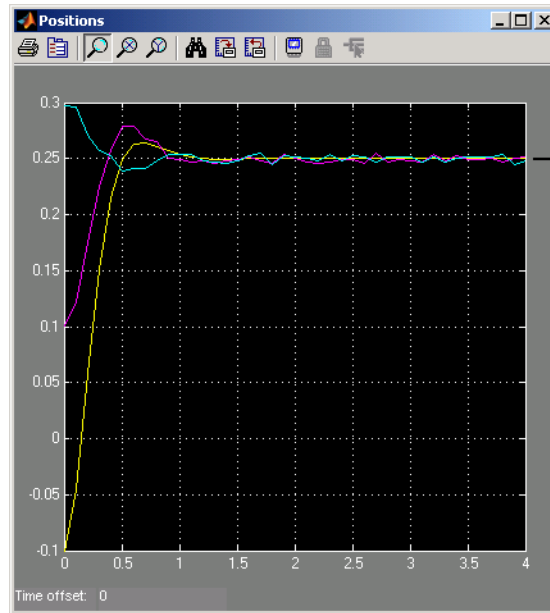
In the **Data Sets**, **Parameters**, and **States** panes for the **New Estimation** task, select all the check boxes in each table. Be sure to select **Position** for both data sets in the **States** pane to estimate the initial condition for the spring's position.

The initial position estimates for the two data sets are known to differ, but set the initial state guesses for both data sets to **-0.1**.



Running the Estimation and Viewing Results

Click **Start** in the **Estimation** pane to run the estimation. As the estimation proceeds, the most current estimation of position response (yellow curve) updates itself in the Scope. The curve appears to toggle between the two experimental data sets, since the estimator uses the two sets successively to update the estimates of the parameter values. The estimator converges to the correct parameter values, within the scope of experimental noise and optimization options settings, as indicated by the closeness of the estimated response (yellow) to the experimental data (magenta). Good state estimates for the initial position are also obtained, as can be observed from the **States** tab of **Estim**(with **IC**) estimation task.



The estimation of initial states is important for obtaining the correct estimates of the model parameters. Why not set the initial states (x_0 in this case) as parameters as well? The reason is that the initial states are not fixed physical properties of the system. For different experimental data or operating conditions, these states need not be unique. In this example, two data sets, with distinct initial positions, were used together for a single estimation of model parameters. While the estimates of the model parameters are unique, the initial state (position) is different, and is estimated individually for each data set.

Preprocessing Data

- “Why Preprocess Data?” on page 6-2
- “Data Preprocessing Tool” on page 6-3
- “Excluding Data” on page 6-6
- “Detrending and Filtering” on page 6-15
- “Handling Missing Data” on page 6-17
- “Adding Preprocessed Data Sets to an Estimation Project” on page 6-19
- “Exporting Preprocessed Data to the MATLAB Workspace” on page 6-22

Why Preprocess Data?

When dealing with empirical data, it is often useful to remove outliers, smooth, detrend, or otherwise treat the data to make it more tractable for analysis and estimation purposes. You can perform the following tasks using Simulink Parameter Estimation software:

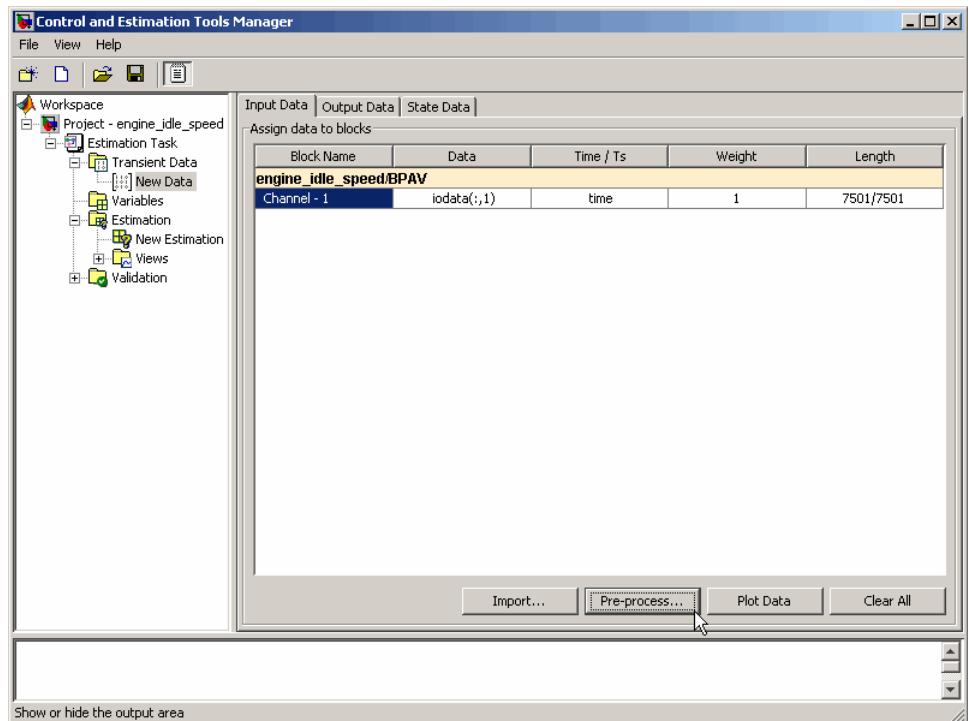
- Exclusion — Eliminate outliers, represent them as NaNs, or use interpolation.
- Detrending — Remove mean values or a straight line trend.
- Filter — Smooth data using a first-order filter, an arbitrary transfer function, or an ideal filter.

You can overwrite the existing data with the preprocessed data, or store it in a new file.

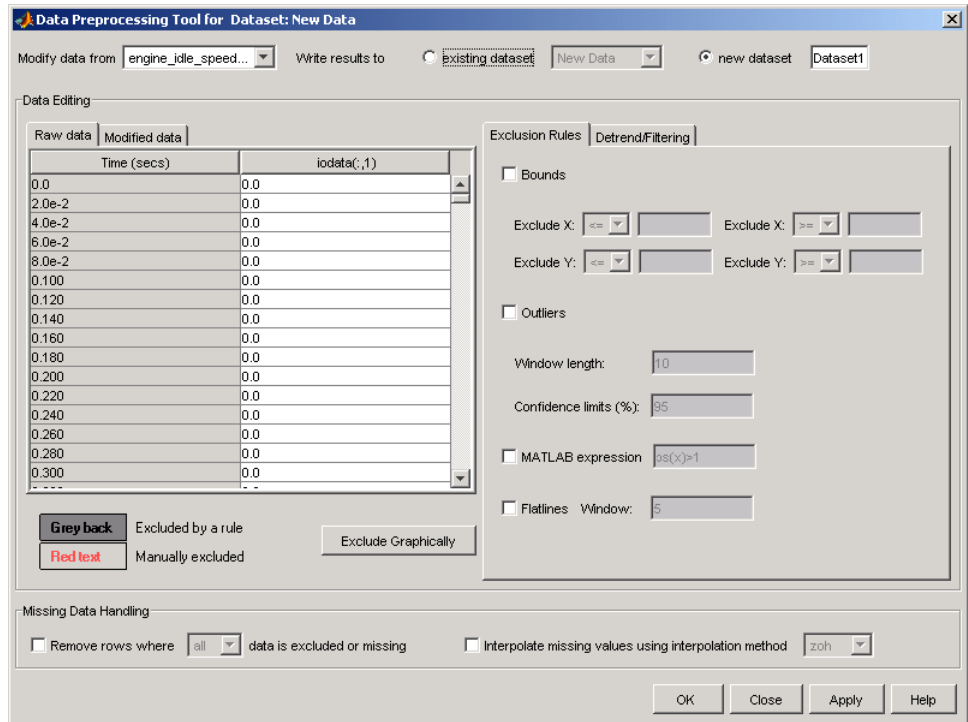
Data Preprocessing Tool

You can preprocess your data using the Data Preprocessing Tool. To open it:

- 1 Open the Control and Estimation Tools Manager.
- 2 In the workspace directory tree, expand the **Estimation Task** node.
- 3 Select the **Transient Data** node in the workspace directory tree, and then choose the data you want to modify in the **Input Data**, or **Output Data** pane. This enables the **Pre-process** button.



- 4 Click the **Pre-process** button to open the Data Preprocessing Tool.



Tip When you have multiple data sets, select the data set that you want to prepare from the **Modify data from** drop-down list in the Data Preprocessing Tool.

In this chapter, the sample data set is the same as used in the engine_idle_speed Simulink model. See “How Simulink® Parameter Estimation Software Works” on page 1-5 for an overview of creating estimation projects and adding data sets.

With the Data Preprocessing Tool, you can

- Exclude data by selecting it with your mouse.
- Exclude data graphically by selecting regions on a plot.
- Exclude data by rules, such as upper or lower bounds.
- Detrend data.
- Filter data.

Excluding Data

In this section...
“Techniques for Excluding Data in the Data Preprocessing Tool” on page 6-6
“Selecting Data for Exclusion from the Data Editing Table” on page 6-6
“Selecting Data for Exclusion from a Plot of the Data” on page 6-9
“Selecting Data for Exclusion by a Rule” on page 6-12

Techniques for Excluding Data in the Data Preprocessing Tool

You can use the Data Preprocessing Tool to select a portion of the data to be excluded from the estimation process. You can choose one of the following techniques:

- Selecting data from the Data Editing Table.
- Selecting data from a plot of the Data.
- Specifying a Rule.

You accomplish the first two manually, and for the last you specify a rule. When you exclude data using manual selection, the excluded data is shown as red. When you exclude data using a rule, the background color of the cell becomes gray. When a portion of the data is excluded both manually and by a rule, the data is red, and the background is gray.

Note Changes in data are visible everywhere. When you use the **Data Editing** table, you can view the results in the data plot.

Selecting Data for Exclusion from the Data Editing Table

The **Data Editing** table lists both the raw data set and the modified data that you create.

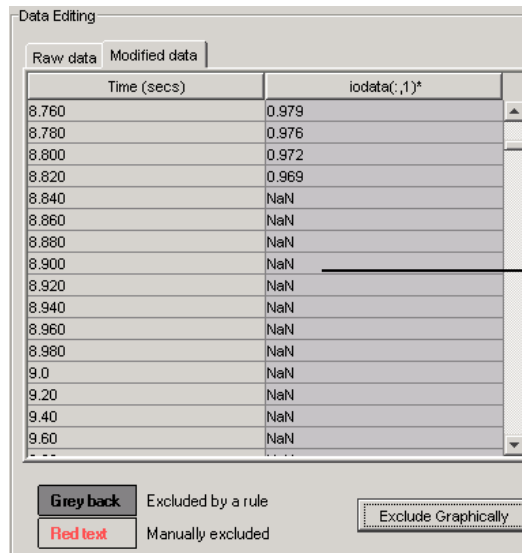
Use your mouse to select groups of cells for exclusion. Selected cells become blue. Right-click and select **Exclude**. The background becomes white, but the numbers are now red.

Click this button to view the data graphically.

Time (secs)	iodata(:,1)
21.680	-0.472
21.700	-0.515
21.720	-0.557
21.740	-0.598
21.760	-0.638
21.780	-0.676
21.800	-0.712
21.820	-0.746
21.840	-0.779
21.860	-0.809
21.880	-0.838
21.900	-0.865
21.920	-0.889
21.940	-0.911
21.960	-0.931
21.980	-0.948

Grey back Excluded by a rule
Red text Manually excluded
Exclude Graphically

There are two tabs in the **Data Editing** pane: **Raw data** and **Modified data**. The **Raw Data** pane shows the working copy of the data. For example, if you exclude rows of data in the **Raw data** pane, the corresponding rows of numbers become red in this table. By default the **Modified data** pane represents the rows you removed by inserting NaNs.



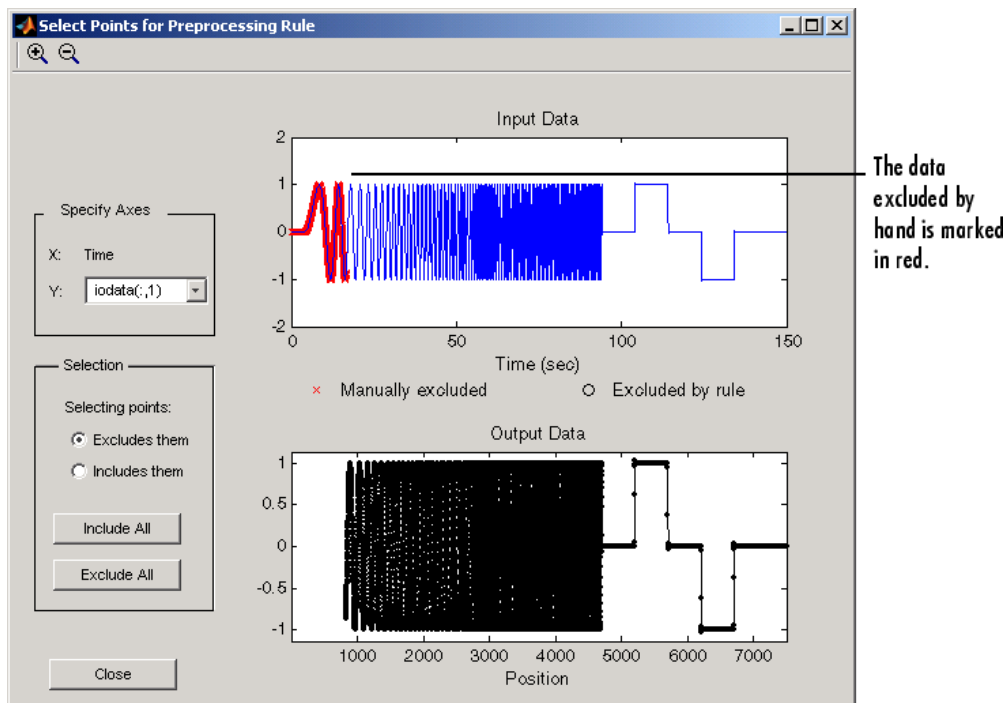
The screenshot shows the 'Data Editing' dialog box with two tabs: 'Raw data' and 'Modified data'. The 'Modified data' tab is selected, displaying a table with two columns: 'Time (secs)' and 'iodata(:,1)*'. The table contains 15 rows of data. The first three rows have numerical values, while the remaining rows have NaN values. Below the table, there are three buttons: 'Grey back' (labeled 'Excluded by a rule'), 'Red text' (labeled 'Manually excluded'), and 'Exclude Graphically'.

Time (secs)	iodata(:,1)*
8.760	0.979
8.780	0.976
8.800	0.972
8.820	0.969
8.840	NaN
8.860	NaN
8.880	NaN
8.900	NaN
8.920	NaN
8.940	NaN
8.960	NaN
8.980	NaN
9.0	NaN
9.20	NaN
9.40	NaN
9.60	NaN

By default, data that you excluded from the **Raw Data** table is represented by NaNs in the **Modified Data** table. If you choose to interpolate or remove missing data, the results of that action are shown in the **Modified Data** table.

In the **Modified data** pane, you can choose to remove the excluded data completely or interpolate it. See “Handling Missing Data” on page 6-17 for more information.

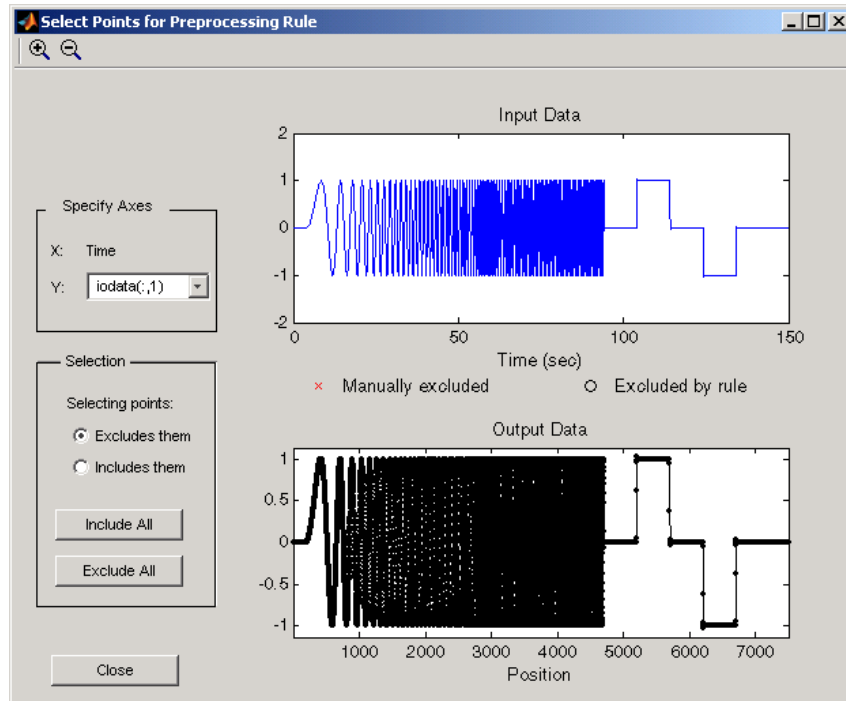
After you select data for exclusion, you can view it graphically by clicking **Exclude Graphically**.



As you make changes in the **Data Editing** pane, they immediately appear in the Select Points for Preprocessing Rule window, and vice versa.

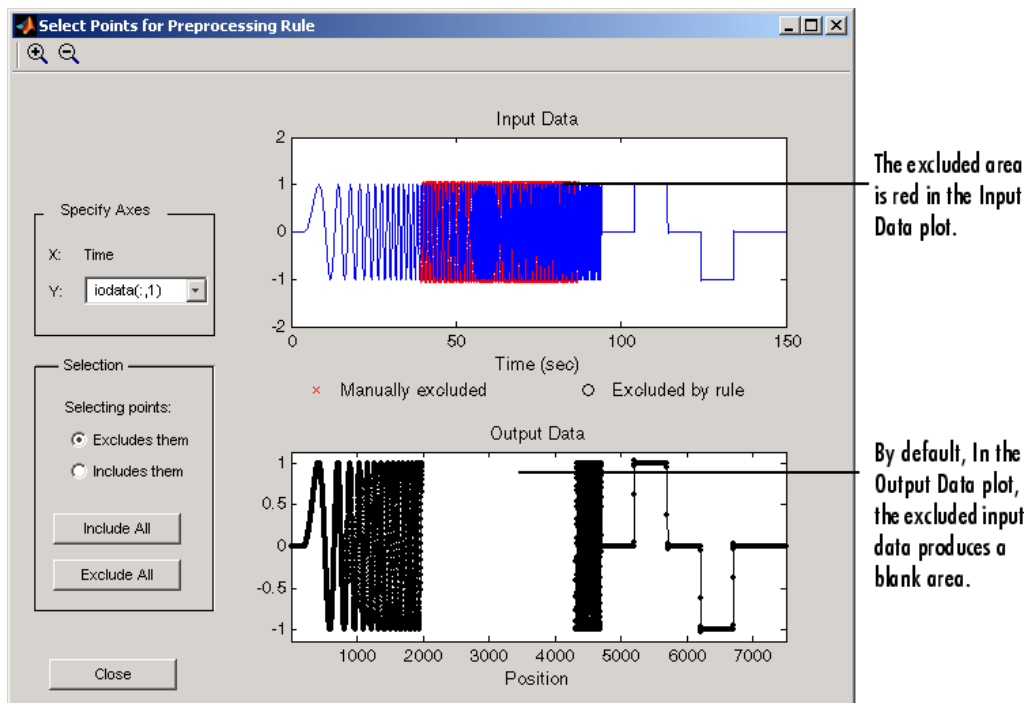
Selecting Data for Exclusion from a Plot of the Data

You can exclude data graphically. Click **Exclude Graphically** to open the Select Points for Preprocessing Rule window.



The way you exclude data is similar to the way you select a region for zooming: place your cursor in the **Input Data** plot and drag the mouse to draw a region of exclusion.

This figure shows an example of resulting data exclusion in the input data.

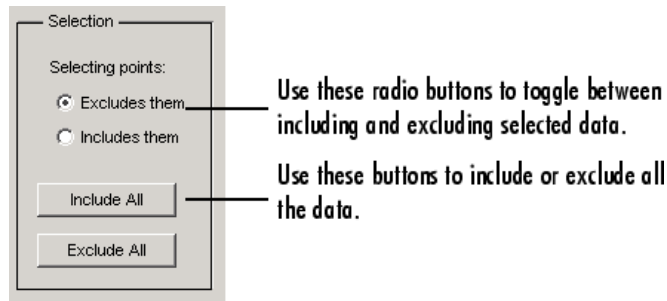


In the **Output Data** plot, the excluded input data produces a blank area by default. This corresponds to the NaNs that now represent excluded data. If you choose to interpolate or remove the excluded data, the output data shows the interpolated points.

When you make changes in the Select Points for Preprocessing Rule window, they immediately appear in the **Data Editing** pane, and vice versa.

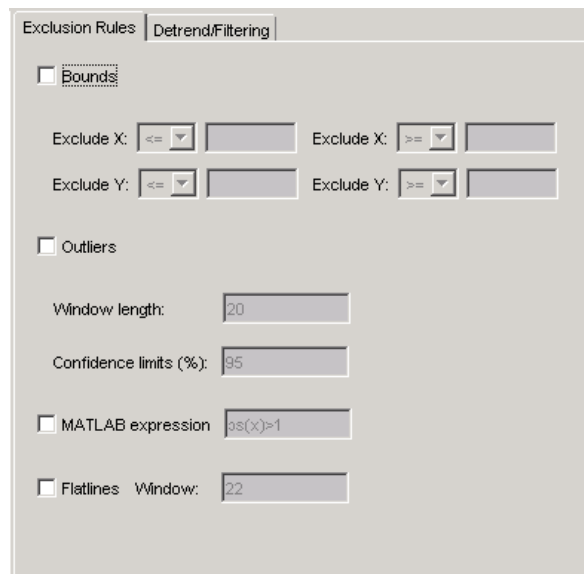
Selection Pane

By default, any box that you draw with your mouse selects data for exclusion, but you can toggle between exclusion and inclusion using the **Selection** pane on the left side of the Select Points for Preprocessing Rule window.



Selecting Data for Exclusion by a Rule

A more precise way to exclude data is to use mathematical rules. The **Exclusion Rules** pane in the Data Preprocessing Tool allows you to enter customized rules for excluding data.



These are the rules you can use to exclude data:

- “Upper and Lower Bounds” on page 6-13
- “Outliers” on page 6-13

- “MATLAB Expressions” on page 6-13
- “Flatlines” on page 6-13

Upper and Lower Bounds

Select the **Bounds** check box to activate upper and lower bound exclusion. Enter numbers in the **Exclude X** and **Exclude Y** fields for upper and lower bound exclusion. By default, the exclusion rule is to include the boundary values, but you can use the menu to exclude the boundaries as well.

Outliers

Select the **Outliers** check box to activate outlier exclusion. You can set the **Window length** to any positive integer, and use confidence limits from 0 to 100%. The window length specifies the number of data points used when calculating outliers.

MATLAB Expressions

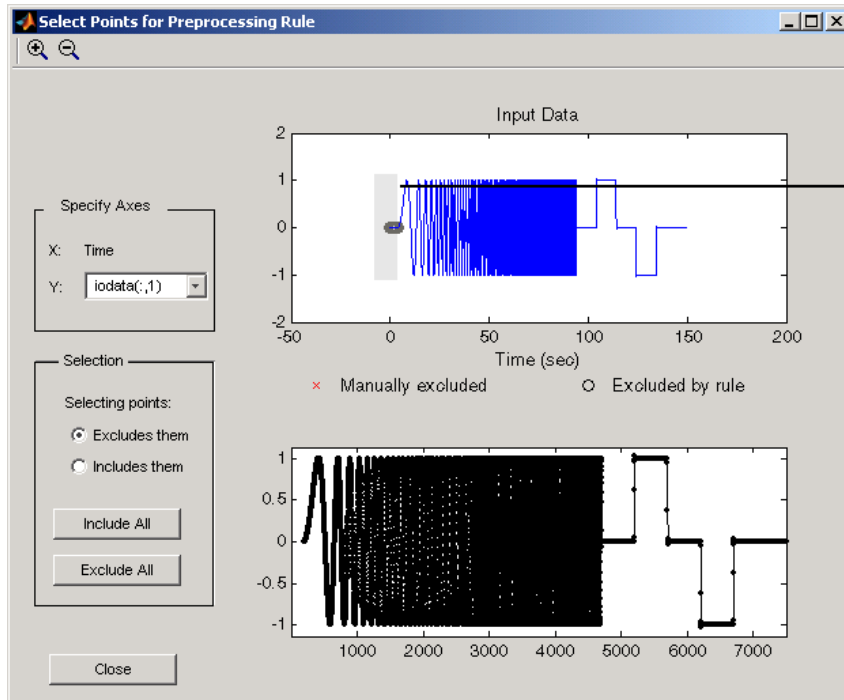
Use the **MATLAB expression** field to enter any mathematical expression using MATLAB code. Use x as the variable name in your expression for the data being tested.

Flatlines

If you have areas of your data set where the data is constant, providing no new information, then you can choose to exclude those data points as flatlines. The **Window length** field sets the minimum number of constant data points required to define the area as a flatline.

Example of Rule Exclusion

This figure shows data with a region of the x-axis excluded.



The region of data excluded by rule is shaded gray.

Detrending and Filtering

In this section...

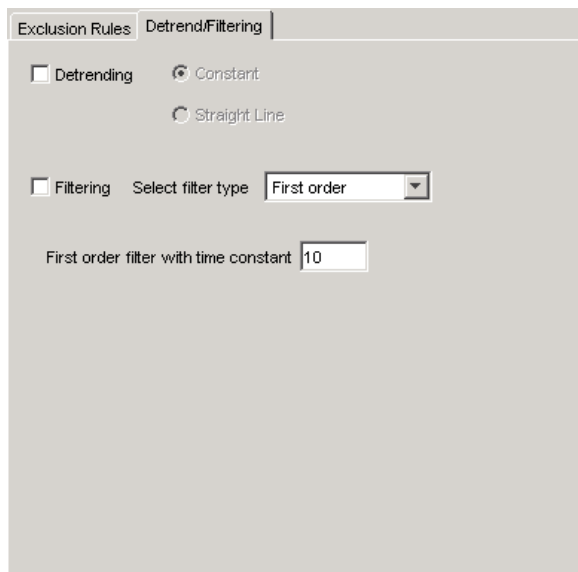
“Detrending and Filtering Data in the Data Preprocessing Tool” on page 6-15

“Detrending” on page 6-15

“Filtering” on page 6-16

Detrending and Filtering Data in the Data Preprocessing Tool

You can both detrend and filter data using the **Detrend/Filtering** pane in the Data Preprocessing Tool.



The screenshot shows the 'Detrend/Filtering' pane of the Data Preprocessing Tool. It has two tabs: 'Exclusion Rules' and 'Detrend/Filtering'. The 'Detrend/Filtering' tab is active. It contains the following controls:

- Detrending
- Constant
- Straight Line
- Filtering
- Select filter type:
- First order filter with time constant:

Detrending

To detrend, select the **Detrending** check box. You can choose constant or straight line detrending. Constant detrending removes the mean of the data

to create zero-mean data. Straight line detrending finds linear trends (in the least-squares sense) and then removes them.

Filtering

You have these choices for filtering your data:

- **First order** — A filter of the type $\frac{1}{\tau s + 1}$ where τ is the time constant that you specify in the associated field.

- **Transfer function** — A filter of the type

$$\frac{a_n s^n + a_{n-1} s^{n-1} + \dots + a_0}{b_m s^m + b_{m-1} s^{m-1} + \dots + b_0}$$

where you specify the coefficients as vectors in the associated **A coefficients** and **B coefficients** fields.

- **Ideal** — An idealized (noncausal) filter, either stop or pass band. Specify either filter as a two-element vector in the **Range (Hz)** field. These filters are ideal in the sense that there is no finite rolloff or ripple; the ends of the ranges are perfectly horizontal in the frequency domain.

Handling Missing Data

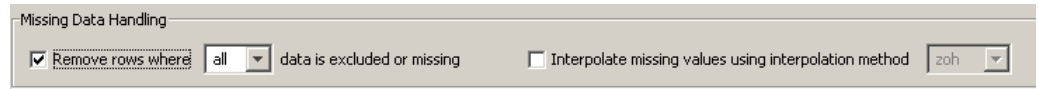
In this section...

“Removing Missing Data” on page 6-17

“Interpolating Missing Data” on page 6-17

Removing Missing Data

To remove the rows containing missing or excluded data, represented by NaNs, select the **Remove rows where** check box in the **Missing Data Handling** area of the Data Preprocessing Tool GUI.



When the data set contains multiple columns of data, select **all** to remove rows in which all the data is excluded. Select **any** to remove any excluded cell. In the case of one-column data, **any** and **all** are equivalent.

Tip You can view the modified data in the **Modified data** tab of the Data Preprocessing Tool GUI.

Interpolating Missing Data

The interpolation operation computes the missing data values using known data values. When you select the **Interpolate missing values using interpolation method** check box in the **Missing Data Handling** area of the Data Preprocessing Tool GUI, the software interpolates the missing data values.



You can compute the missing data values using one of the following interpolation methods:

- Zero-order hold (zoh) — Fills the missing data sample with the data value immediately preceding it.
- Linear interpolation (Linear) — Fills the missing data sample with the average of the data values immediately preceding and following it.

By default, the interpolation method is set to zoh. You can select the Linear interpolation method from the **Interpolate missing values using interpolation method** drop-down list.

Tip You can view the results of interpolation in the **Modified data** tab of the Data Preprocessing Tool GUI.

Adding Preprocessed Data Sets to an Estimation Project

In this section...

“Overwriting an Existing Data Set” on page 6-19

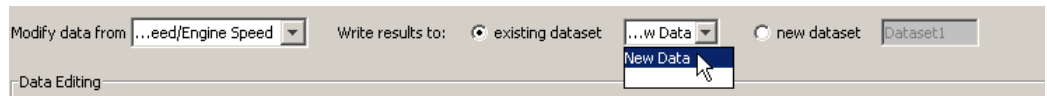
“Creating a New Data Set” on page 6-20

Overwriting an Existing Data Set

After you preprocess the data using the techniques in the Data Preprocessing Tool, you can add the data set to an estimation project either by overwriting or creating a new data set. For more information on how to create a new data set, see “Creating a New Data Set” on page 6-20.

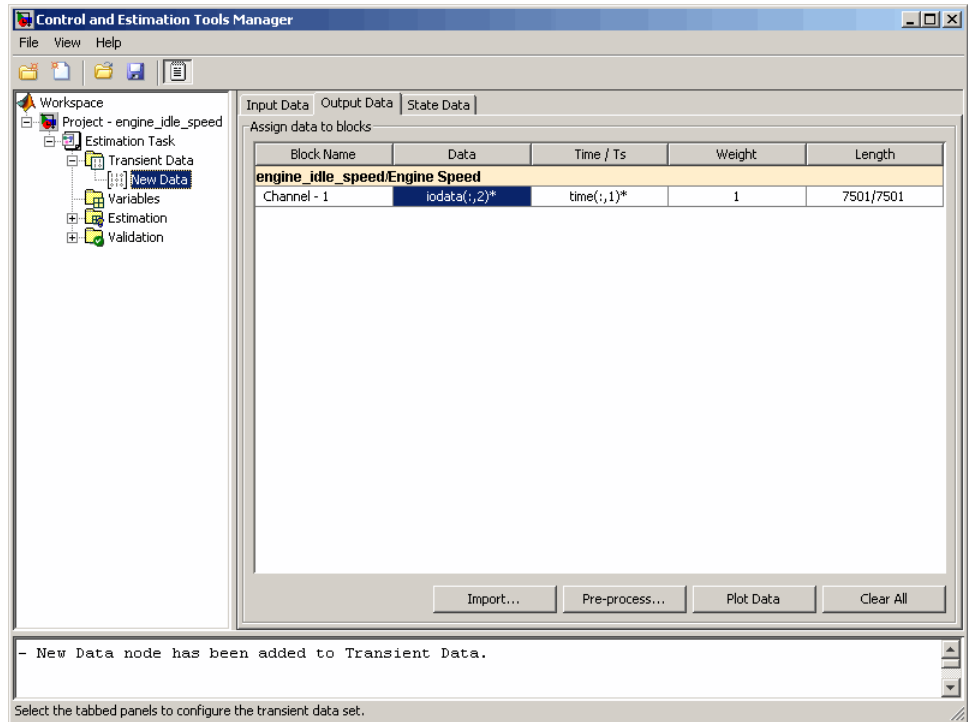
To overwrite an existing data set with the preprocessed data:

- 1 In the **Write results to** area of the Data Preprocessing Tool GUI, select the **existing dataset** option.
- 2 Choose the data set you want to overwrite from the drop-down list.



- 3 Click **Add**.

This action overwrites the selected data set with the modified data in the Control and Estimation Tools Manager GUI.



Tip You can export the preprocessed data to the MATLAB Workspace, as described in “Exporting Preprocessed Data to the MATLAB Workspace” on page 6-22.

Creating a New Data Set

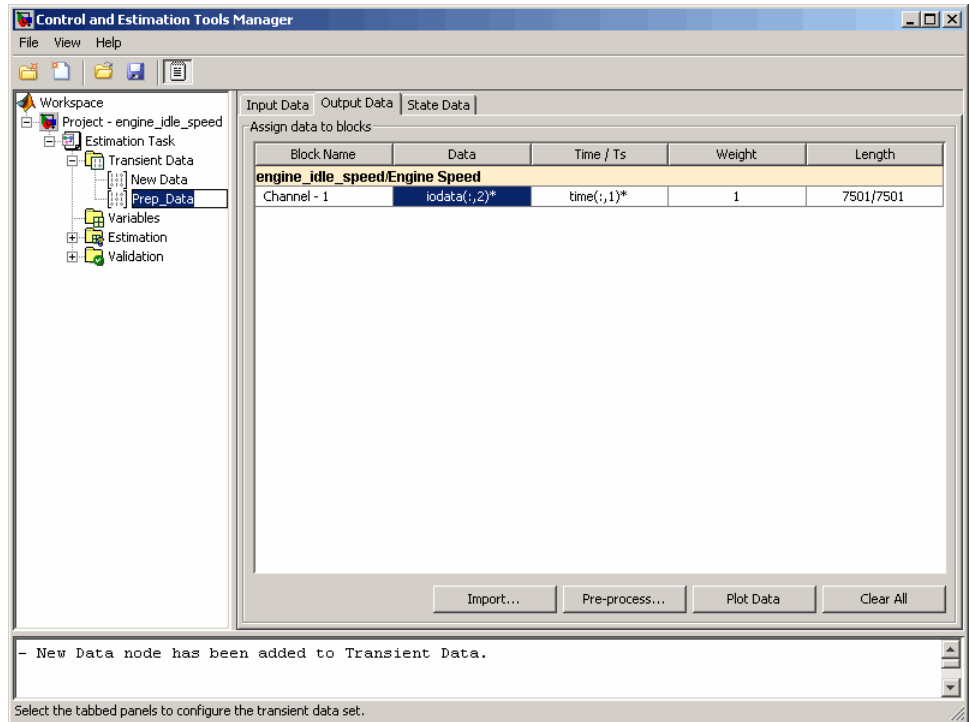
If you do not want to overwrite an existing data set with the preprocessed data, as described in “Overwriting an Existing Data Set” on page 6-19, you can create a new data set for the preprocessed data:

- 1 In the **Write results to** area of the Data Preprocessing Tool GUI, select the **new dataset** option.
- 2 Specify the name of the data set in the adjacent field.



3 Click Add.

This action adds a new data node in the Control and Estimation Tools Manager GUI containing the modified data.



Tip You can export the preprocessed data to the MATLAB Workspace, as described in “Exporting Preprocessed Data to the MATLAB Workspace” on page 6-22.

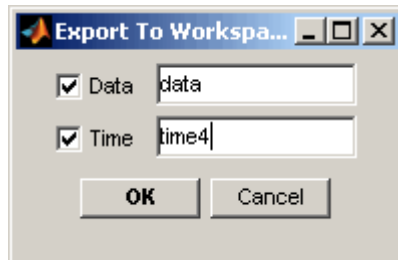
Exporting Preprocessed Data to the MATLAB Workspace

After you add the preprocessed data to an estimation project, as described in “Adding Preprocessed Data Sets to an Estimation Project” on page 6-19, you can export the data set to the MATLAB Workspace. You can use the data to further prepare it or estimate parameters using the data.

- 1 In the **Transient Data** node of the Control and Estimation Tools Manager GUI, select the node containing the prepared data set.
- 2 Right-click the table **Data** cell containing the data that you want to export, and select **Export**.

The Export to Workspace dialog box opens.

- 3 Specify the MATLAB variable names for the prepared data and the corresponding time vector in the **Data** and **Time** fields, respectively.



- 4 Click **OK**.

The resulting MATLAB variables `data` and `time4` appear in the MATLAB Workspace browser.

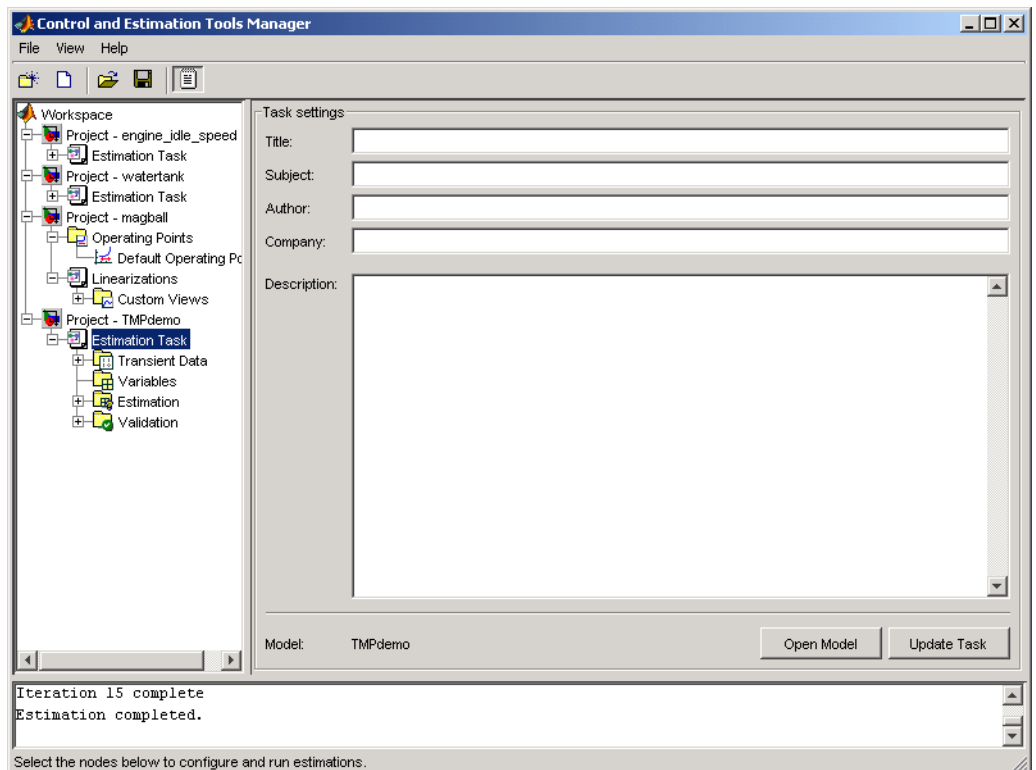
Managing Multiple Projects

- “Multiple Projects and Tasks” on page 7-2
- “Saving Control and Estimation Tools Manager Projects” on page 7-3
- “Opening Control and Estimation Tools Manager Projects” on page 7-4

Multiple Projects and Tasks

The Control and Estimation Tools Manager works seamlessly with products in the Controls and Estimation family. In particular, if you have licenses for Simulink Control Design or Model Predictive Control Toolbox software, you can use these products to perform tasks on projects that you have created in Simulink Parameter Estimation software, and vice versa.

This figure shows a tools manager with multiple projects and multiple tasks.

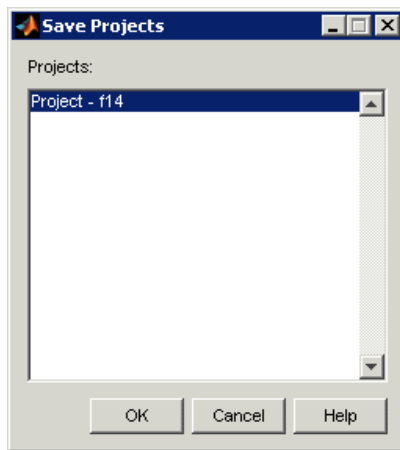


You can save projects individually, or group multiple projects together in one saved file. This chapter describes how to do this.

Saving Control and Estimation Tools Manager Projects

A Control and Estimation Tools Manager project can consist of tasks from products such as Simulink Control Design, Simulink Parameter Estimation, and Model Predictive Control Toolbox software. Each task contains data, objects, and results for the analysis of a particular model.

To save your project as a MAT-file, select **File > Save** in the Control and Estimation Tools Manager window.

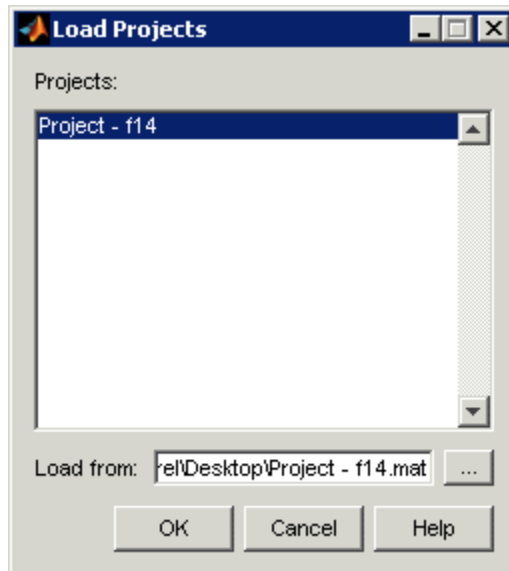


To save multiple projects within one file:

- 1** In the Save Projects dialog box, select the projects that you want to save.
- 2** Click **OK**.
- 3** Choose a directory and name for your project file by either browsing for a file or typing the full path and filename in the **Save as** field. Click **Save**.

Opening Control and Estimation Tools Manager Projects

To open previously saved projects, select **File > Load** in the Control and Estimation Tools Manager window.



In the Load Projects dialog box, choose a project file by either browsing for the directory and file, or by typing the full path and filename in the **Load from** field. Project files are always MAT-files. The projects within this file appear in the **Projects** list.

Select the projects that you want to load, then click **OK**. When a file contains multiple projects, you can choose to load them all or just a few.

Approximating System Models Using Lookup Tables

- “What Are Lookup Tables?” on page 8-2
- “Estimating Values of Lookup Tables” on page 8-5
- “Capturing Time-Varying System Behavior Using Adaptive Lookup Tables” on page 8-37

What Are Lookup Tables?

In this section...
“Static Lookup Tables” on page 8-2
“Adaptive Lookup Tables” on page 8-3

Static Lookup Tables

Lookup tables are tables that store numeric data in a multidimensional array format. In the simpler two-dimensional case, lookup tables can be represented by matrices. Each element of a matrix is a numerical quantity, which can be precisely located in terms of two indexing variables. At higher dimensions, lookup tables can be represented by multidimensional matrices, whose elements are described in terms of a corresponding number of *indexing variables*.

Lookup tables provide a means to capture the dynamic behavior of a physical (mechanical, electronic, software) system. The behavior of a system with M inputs and N outputs can be approximately described by using N lookup tables, each consisting of an array with M dimensions.

You usually generate lookup tables by experimentally collecting or artificially creating the input and output data of a system. In general, you need as many indexing parameters as the number of input variables. Each indexing parameter may take a value within a predetermined set of data points, which are called the *breakpoints*. The set of all breakpoints corresponding to an indexing variable is called a *grid*. Thus, a system with M inputs is gridded by M sets of breakpoints. The software uses the breakpoints to locate the array elements, where the output data of the system are stored. For a system with N outputs, the software locates the N array elements and then stores the corresponding data at these locations.

After you create a lookup table using the input and output measurements as described previously, you can use the corresponding multidimensional array of values in applications without having to remeasure the system outputs. In fact, you need only the input data to locate the appropriate array elements in the lookup table because the software reads the approximate system output from the data stored at these locations. Therefore, a lookup table provides a

suitable means of capturing the input-output mapping of a *static* system in the form of numeric data stored at predetermined array locations. For more information, see “About Lookup Table Blocks” in the Simulink documentation.

You can use Simulink Parameter Estimation software to estimate lookup table values, as described in “Estimating Values of Lookup Tables” on page 8-5.

Adaptive Lookup Tables

Statically defined lookup tables, as described in “Static Lookup Tables” on page 8-2, cannot accommodate the *time-varying* behavior (characteristics) of a physical plant. Static lookup tables establish a permanent and static mapping of input-output behavior of a physical system. Conversely, the behavior of actual physical systems often varies with time due to wear, environmental conditions, and manufacturing tolerances. With such variations, the static mapping of input-output behavior of a plant described by the lookup table may no longer provide a valid representation of the plant characteristics.

Adaptive lookup tables incorporate the time-varying behavior of physical plants into the lookup table generation and maintenance process while providing all of the functionality of a regular lookup table.

The adaptive lookup table receives the input and output measurements of a plant’s behavior, which are then used to dynamically create and update the content of the underlying lookup table. In addition to requiring the input data to create the lookup table, the adaptive lookup table also uses the output data of the plant to recalculate the table values. For example, you can collect the output data of the plant by placing sensors at appropriate locations in a physical system.

The software uses the input measurements to locate the array elements by comparing these input values with the breakpoints defined for each indexing variable. Next, it uses the output measurements to recalculate the numeric value stored at these array locations. However, unlike a regular table, which only stores the array data before the actual use of the lookup table, the adaptive table continuously improves the content of the lookup table. This continuous improvement of the table data is referred to as the *adaptation process* or *learning process*.

The adaptation process involves statistical and signal processing algorithms to recapture the input-output behavior of the plant. The adaptive lookup table always tries to provide a valid representation of the plant dynamics even though the plant behavior may be time varying. The underlying signal processing algorithms are also robust against reasonable measurement noise and they provide appropriate filtering of noisy output measurements. To learn more about how to model systems using adaptive lookup tables, see “Capturing Time-Varying System Behavior Using Adaptive Lookup Tables” on page 8-37.

Estimating Values of Lookup Tables

In this section...

“How to Estimate Values of a Lookup Table” on page 8-5

“Example — Estimating Lookup Table Values from Data” on page 8-6

“Example — Estimating Constrained Values of a Lookup Table” on page 8-20

How to Estimate Values of a Lookup Table

You can use lookup table Simulink blocks to approximate a system’s behavior, as described in “Working with Lookup Tables” in the Simulink documentation. After you build your system using lookup tables, you can use Simulink Parameter Estimation software to estimate the table values from measured I/O data. The workflow for estimating parameters of a lookup table consist of the following tasks:

- 1** Creating a Simulink model using lookup table blocks.
- 2** Importing the measured input and output (I/O) data from which you want to estimate the table values.
- 3** Analyzing and preparing the I/O data for estimation.
- 4** Estimating the lookup table values.
- 5** Validating the estimated table values using a validation data set.

The following examples illustrate how to estimate the lookup table values:

- “Example — Estimating Lookup Table Values from Data” on page 8-6
- “Example — Estimating Constrained Values of a Lookup Table” on page 8-20

Example — Estimating Lookup Table Values from Data

- “Objectives” on page 8-6
- “About the Data” on page 8-6
- “Configuring a Project for Parameter Estimation” on page 8-6
- “Estimating the Table Values Using Default Settings” on page 8-8
- “Validating the Estimation Results” on page 8-15

Objectives

This example shows how to estimate lookup table values from time-domain input-output (I/O) data.

About the Data

In this example, you use the I/O data in `lookup_regular.mat` to estimate the values of a lookup table. The MAT-file includes the following variables:

- `xdata1` — Consists of 63 uniformly-sampled input data points in the range `[0,6.5]`.
- `ydata1` — Consists of output data corresponding to the input data samples.
- `time1` — Time vector.

You use the I/O data to estimate the lookup table values in the `lookup_regular` Simulink model. The lookup table in the model contains ten values, which are stored in the MATLAB variable `table`. The initial table values comprise a vector of 0s. To learn more about how to model a system using lookup tables, see “Working with Lookup Tables” in the Simulink documentation.

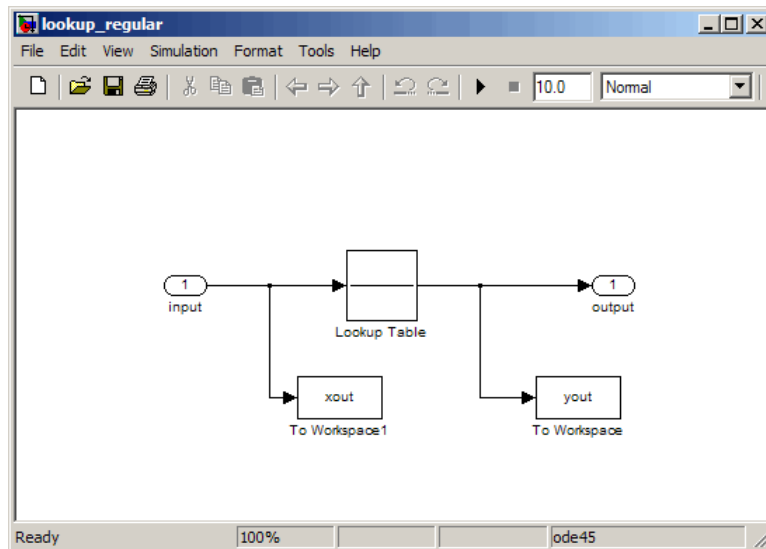
Configuring a Project for Parameter Estimation

To estimate the lookup table values, you must first configure a Control and Estimation Tools Manager project.

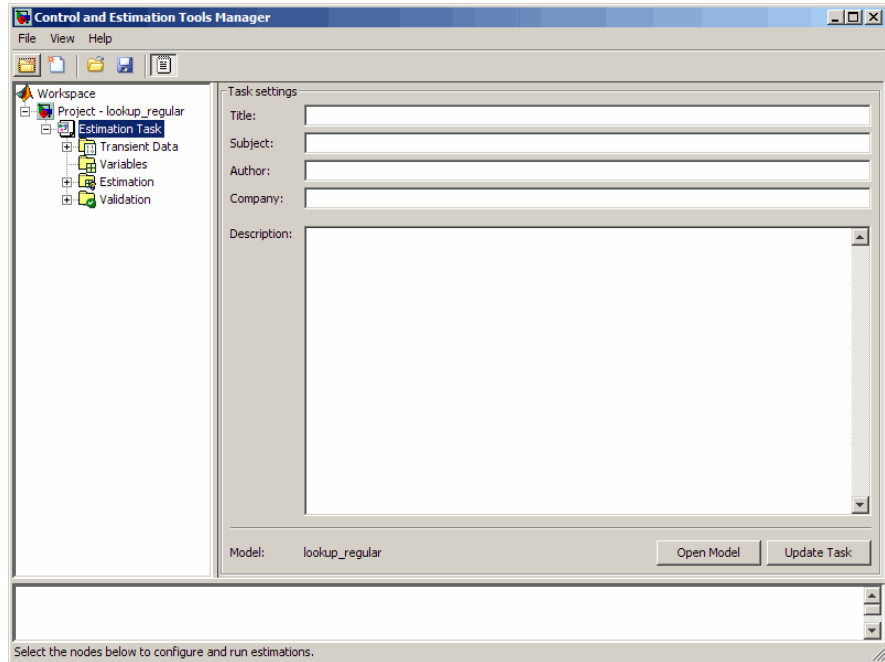
- 1 Open the lookup table model by typing the following command at the MATLAB prompt:

```
lookup_regular
```

This command opens the Simulink model, and loads the estimation data into the MATLAB workspace.



- 2 In the Simulink model, select **Tools > Parameter Estimation** to open a new project named **lookup_regular** in the Control and Estimation Tools Manager GUI.



Estimating the Table Values Using Default Settings

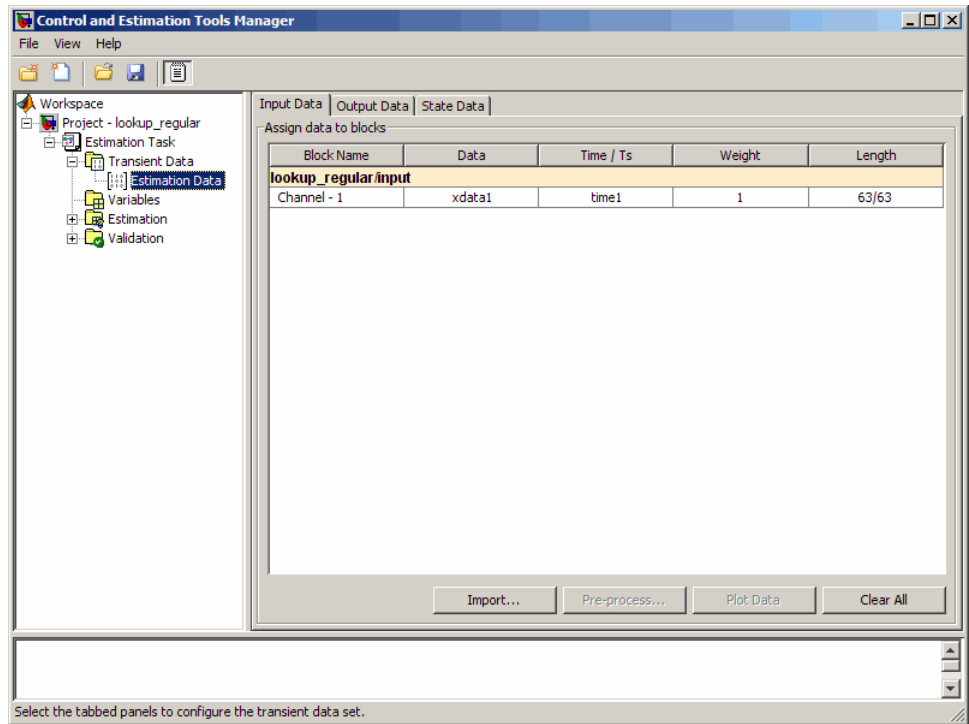
After you configure a project for parameter estimation, as described in “Configuring a Project for Parameter Estimation” on page 8-6, use the following steps to estimate the lookup table values.

- 1 Import the estimation data, as described in the “Importing Data into the GUI” on page 2-6 section of Chapter 2, “Tutorial — Preparing Data for Parameter Estimation Using the GUI”.

You can also load a preconfigured project that already contains the imported data. To do so, type the following command at the MATLAB prompt:

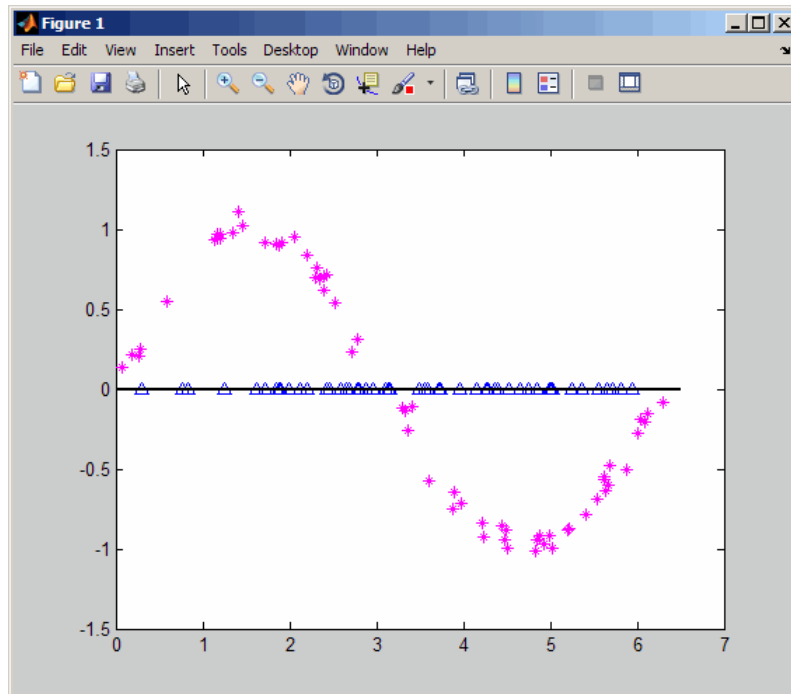
```
lookup_regular;  
spload('lookup_regular_import', 'Project - lookup_regular',...
```


'Estimation Data');



- 2 Run an initial simulation to view the I/O data, simulated output, and the initial table values. To do so, type the following commands at the MATLAB prompt:

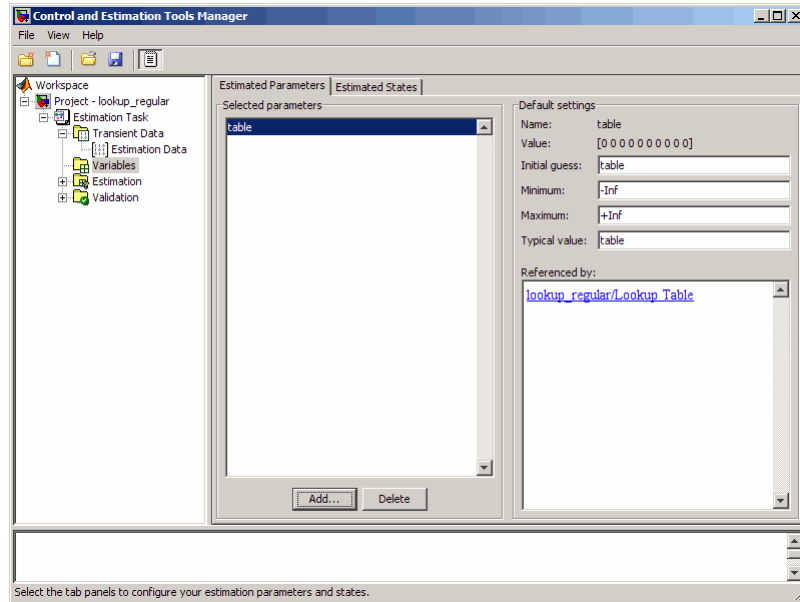
```
sim('lookup_regular')
figure(1); plot(xdata1,ydata1, 'm*', xout, yout, 'b^')
hold on; plot(linspace(0,6.5,10), table, 'k', 'LineWidth', 2)
```



The x- and y-axes of the figure represent the input and output data, respectively. The figure shows the following plots:

- Measured data — Represented by the magenta stars (*).
 - Initial table values — Represented by the black line.
 - Initial simulation data — Represented by the blue deltas (Δ).
- 3 Select the table values to estimate.
 - a In the Control and Estimation Tools Manager GUI, select the **Variables** node under the **Estimation Task** node.

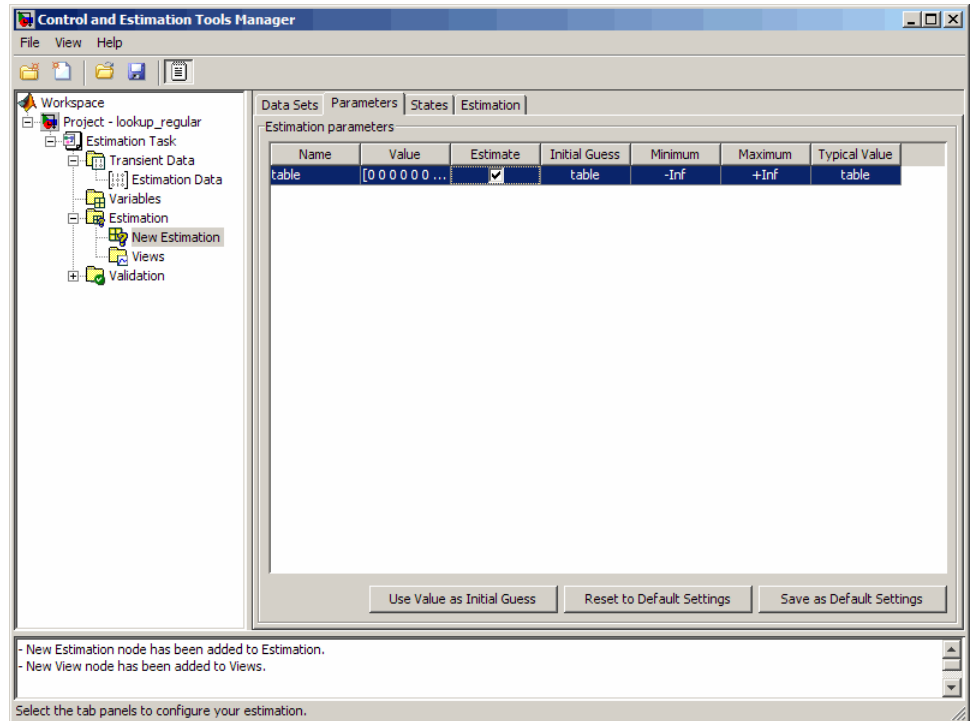
- b Click **Add** to open the Select Parameters dialog box, which shows the Simulink model parameters.
- c Select **table**, and click **OK** to add the table values to the **Estimated Parameters** tab.



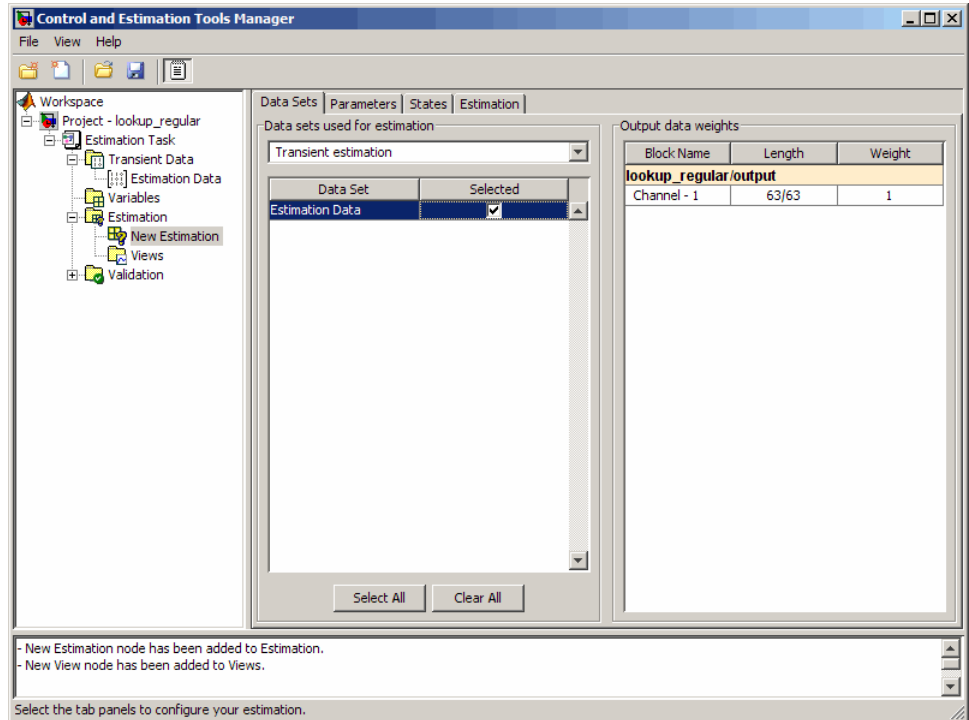
The **Default settings** area of the GUI displays the default settings for the table values. The **Value** field displays the initial table values, which comprise a vector of ten 0s.

- d Select the **Estimation** node, and click **New** to add a **New Estimation** node.

- e In the **Parameters** tab of the **New Estimation** node, select the **Estimate** check box to specify the lookup table values, **table**, for estimation.



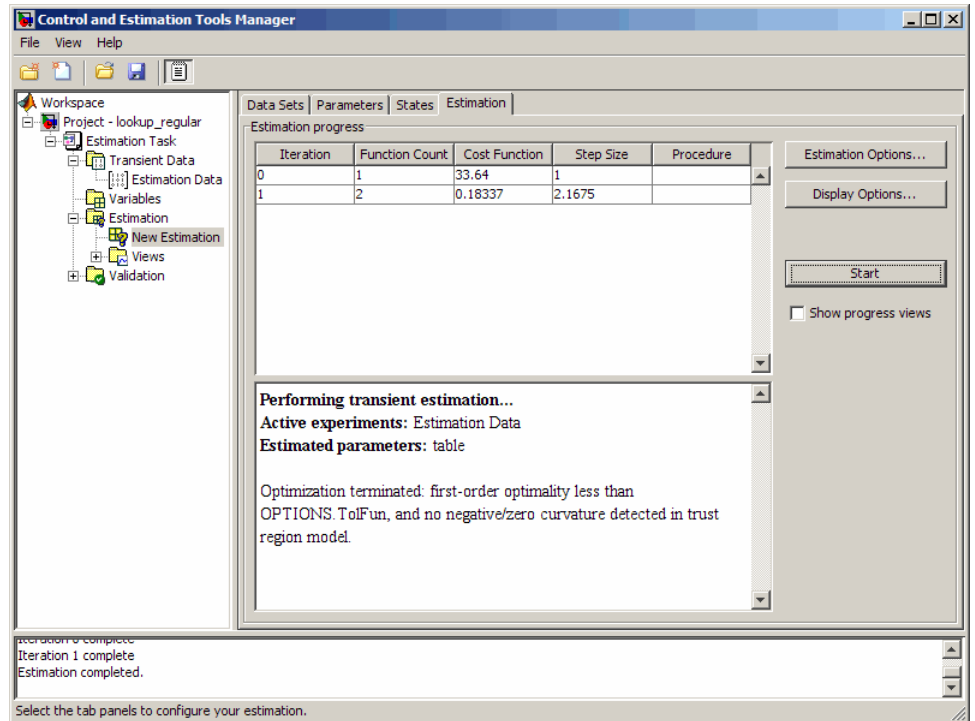
- 4 In the **Data Sets** tab of the **New Estimation** node, select the **Selected** check box to specify the estimation data set.



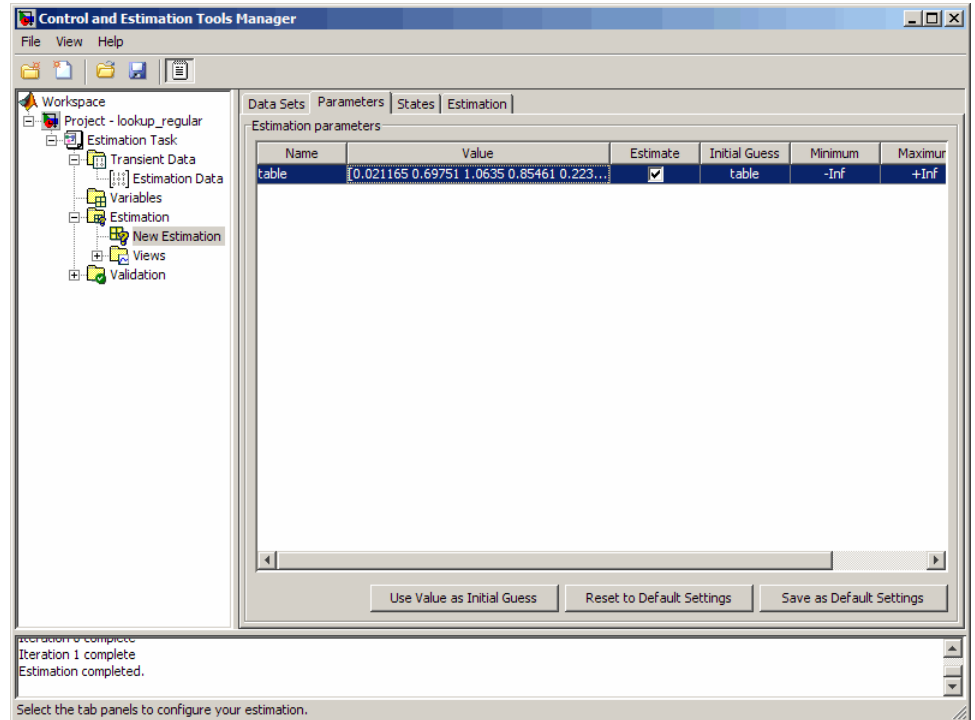
- 5 Estimate the table values using the default settings.

- In the **Estimation** tab of the **New Estimation** node, click **Start** to start the estimation.

The Control and Tools Manager GUI updates at each iteration, and provides information about the estimation progress. After the estimation completes, the Control and Estimation Tools Manager GUI looks similar to the following figure.



- b** Select the **Parameters** tab in the **New Estimation** node to view the estimated table values, which appear in the **Value** field.



Validating the Estimation Results

After you estimate the table values, as described in “Estimating the Table Values Using Default Settings” on page 8-8, you must use another data set to validate that you have not overfitted the model. You plot and examine the following plots to validate the estimation results:

- Residuals plot
- Measured and simulated data plots

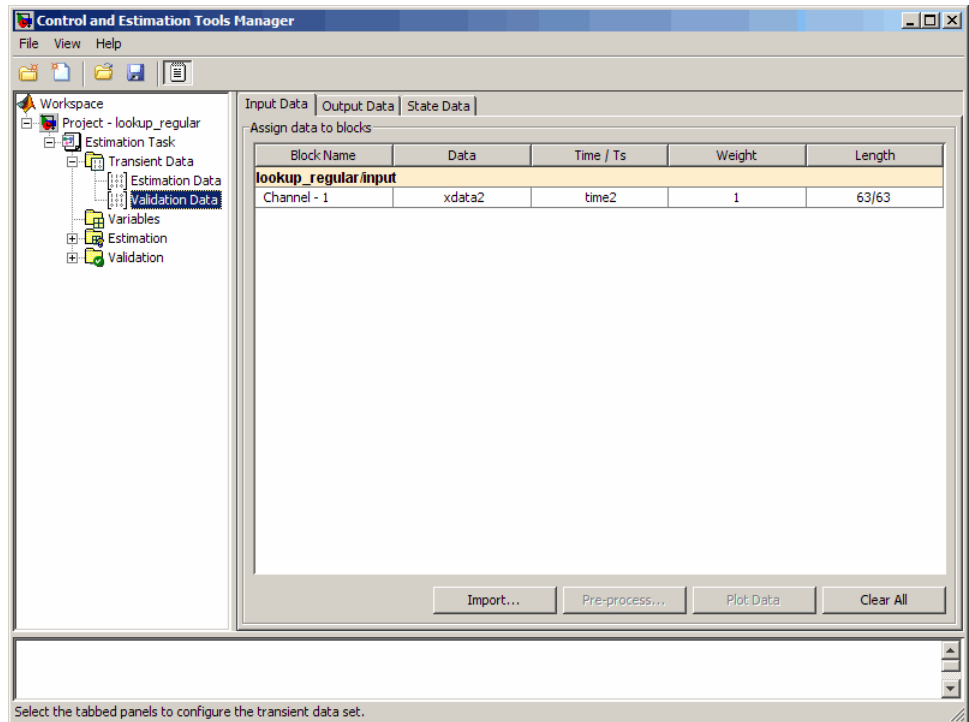
To validate the estimation results:

- 1 Import the validation data set in the Control and Estimation Tools Manager GUI, as described in “Importing Data into the GUI” on page 2-6 section of Chapter 2, “Tutorial — Preparing Data for Parameter Estimation Using the GUI”.

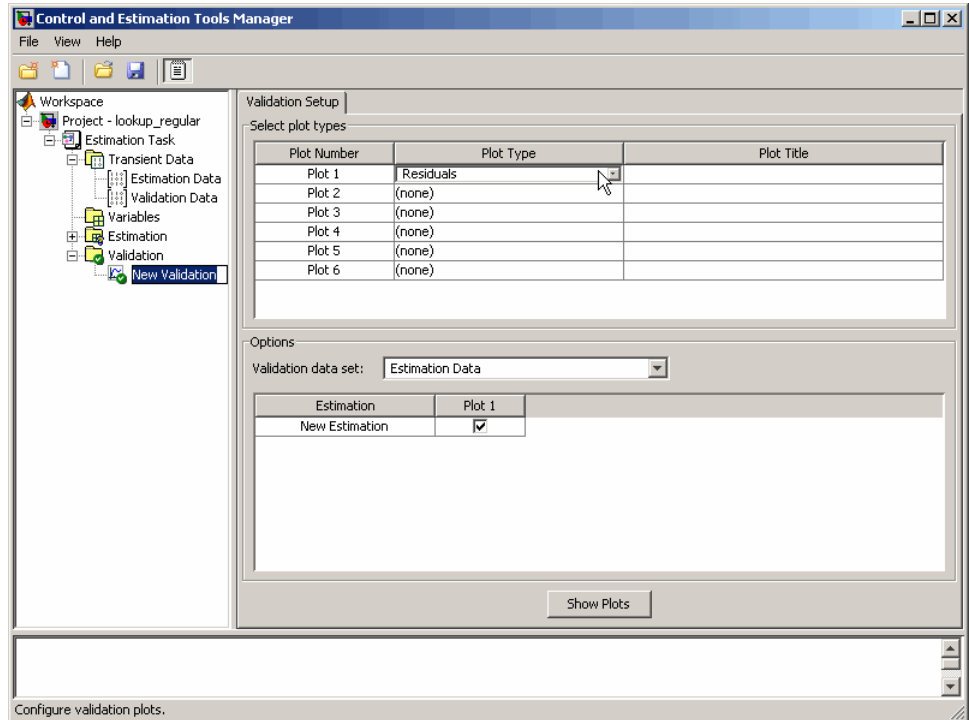
The validation data contains the input data, output data and time vector in the MATLAB variables `xdata2`, `ydata2` and `time2` respectively.

You can also load a project that already contains the estimated parameters, and the validation data set. To do so, type the following commands at the MATLAB prompt:

```
lookup_regular;
spload('lookup_regular_val', 'Project - lookup_regular',...
'Validation Data')
```

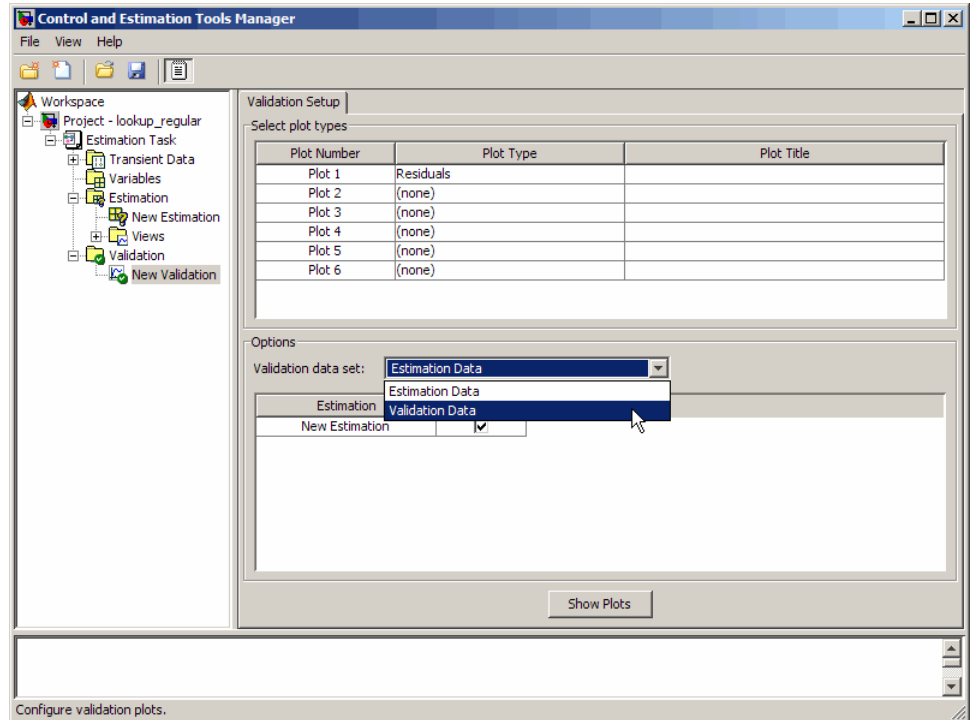


This project also contains the **Residuals** plot already configured in the **Select plot types** area of the GUI, as shown in the next figure.

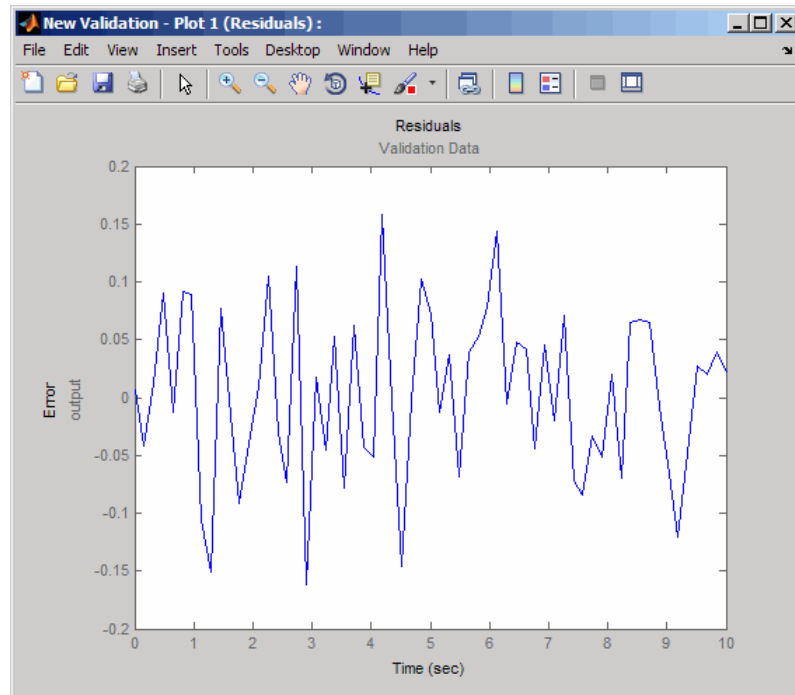


- 2 Plot and examine the residuals:
 - a Select the **New Validation** node under the **Validation** node.

- b** In the **Options** area, select **Validation Data** from the **Validation data set** drop-down list.



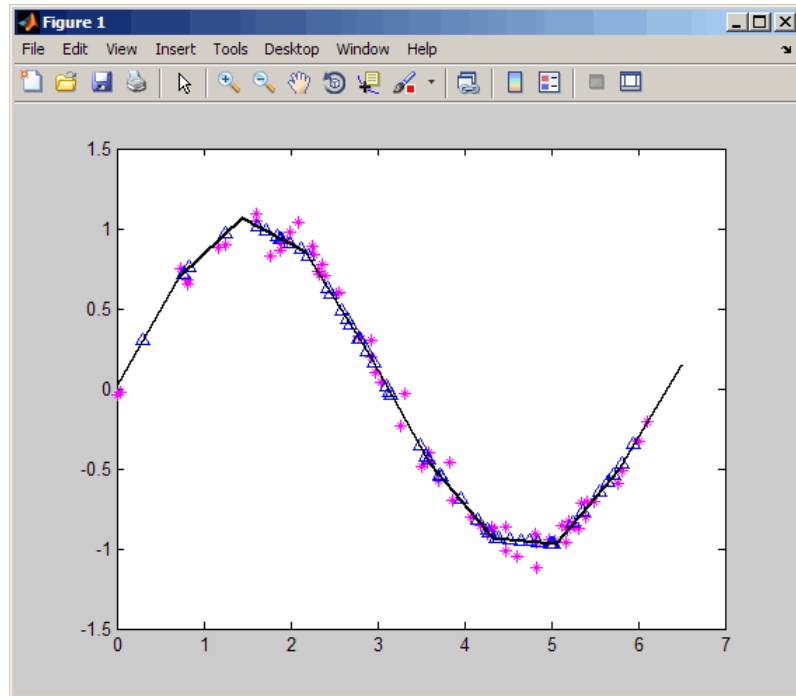
- c Click **Show Plots** to open the residuals plot.



The residuals, which show the difference between the simulated and measured data, lie in the range $[-0.15, 0.15]$ — within 15% of the maximum output variation. This indicates a good match between the measured and the simulated table data values.

- d Plot and examine the estimated table values against the validation data set and the simulated table values by typing the following commands at the MATLAB prompt.

```
sim('lookup_regular')
figure(2); plot(xdata2,ydata2, 'm*', xout, yout,'b^')
hold on; plot(linspace(0,6.5,10), table, 'k', 'LineWidth', 2)
```



The plot shows that the table values, displayed as the black line, match both the validation data and the simulated table values. The table data values cover the entire range of input values, which indicates that all the lookup table values have been estimated.

Example – Estimating Constrained Values of a Lookup Table

- “Objectives” on page 8-21

- “About the Data” on page 8-21
- “Configuring a Project for Parameter Estimation” on page 8-21
- “Estimating the Monotonically Increasing Table Values Using Default Settings” on page 8-24
- “Validating the Estimation Results” on page 8-31

Objectives

This example shows how to estimate constrained values of a lookup table. You apply monotonically increasing constraints to the lookup table values, and use the Simulink Parameter Estimation GUI to estimate the table values.

About the Data

In this example, you use `lookup_increasing.mat`, which contains the measured I/O data for estimating the lookup table values. The MAT-file includes the following variables:

- `xdata1` — Consists of 602 uniformly-sampled input data points in the range $[-5,5]$.
- `ydata1` — Output data corresponding to the input data samples.

Note The output data is a monotonically increasing function of the input data.

- `time1` — Time vector.

You use the I/O data to estimate the values of the lookup table in the `lookup_increasing` Simulink model. The table contains eleven values, which are stored in the MATLAB variable `table`. To learn more about how to specify the table’s values, see “Entering Breakpoints and Table Data” in the Simulink documentation.

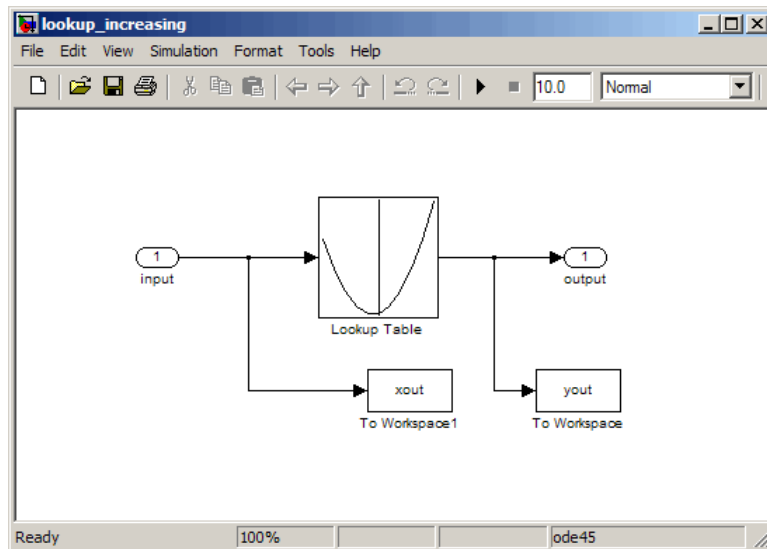
Configuring a Project for Parameter Estimation

To estimate the monotonically increasing lookup table values, you must first configure a Control and Estimation Tools Manager project.

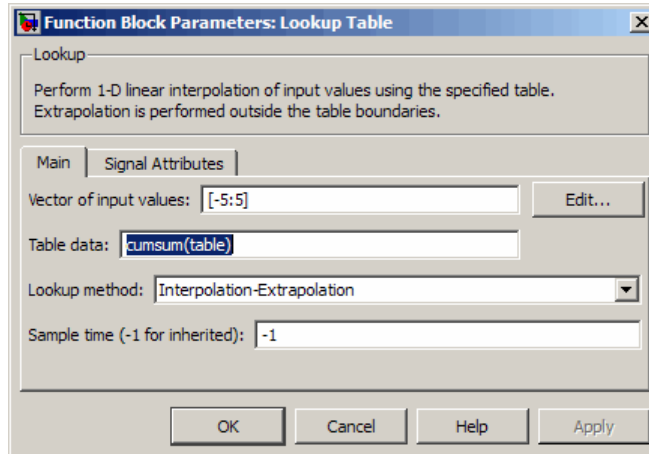
- 1 Open the lookup table model by typing the following command at the MATLAB prompt:

```
lookup_increasing
```

This command opens the Simulink model, and loads the estimation data in the MATLAB workspace.

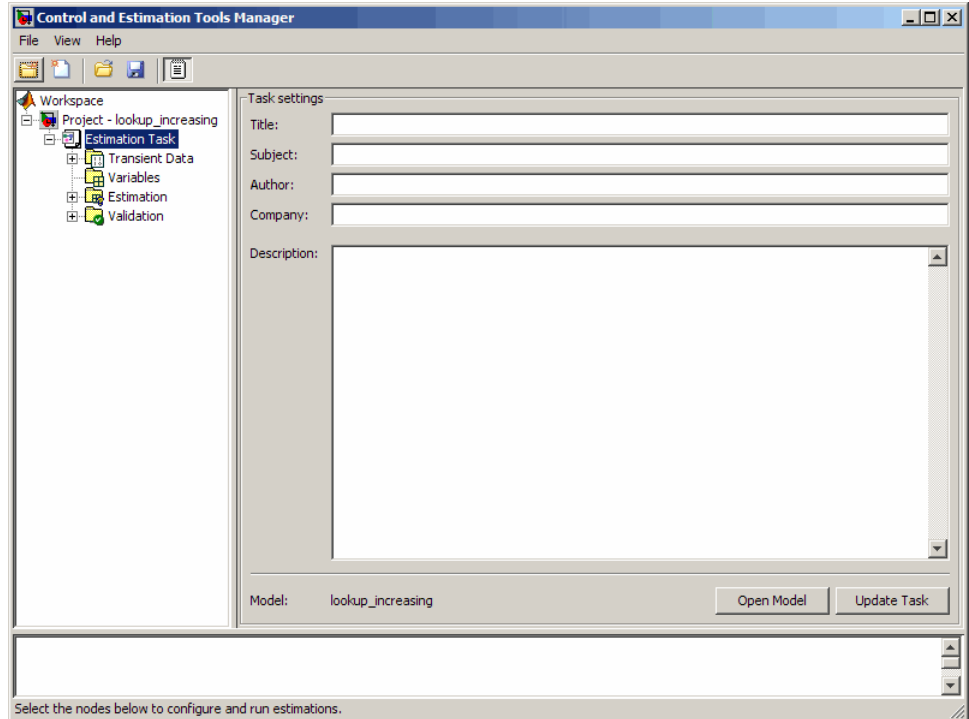


- 2 Double-click the Lookup Table block to view the monotonically increasing constraint applied to the table output values.



The **Table data** field of the Function Block Parameters dialog box shows the constraint. The cumulative sum function, `cumsum`, applies a monotonically increasing constraint on the table output values. This function computes the cumulative sum of the table values based on estimation of the individual table elements from the I/O data.

- 3 In the Simulink model, select **Tools > Parameter Estimation** to open a new project named **lookup_increasing** in the Control and Estimation Tools Manager GUI.



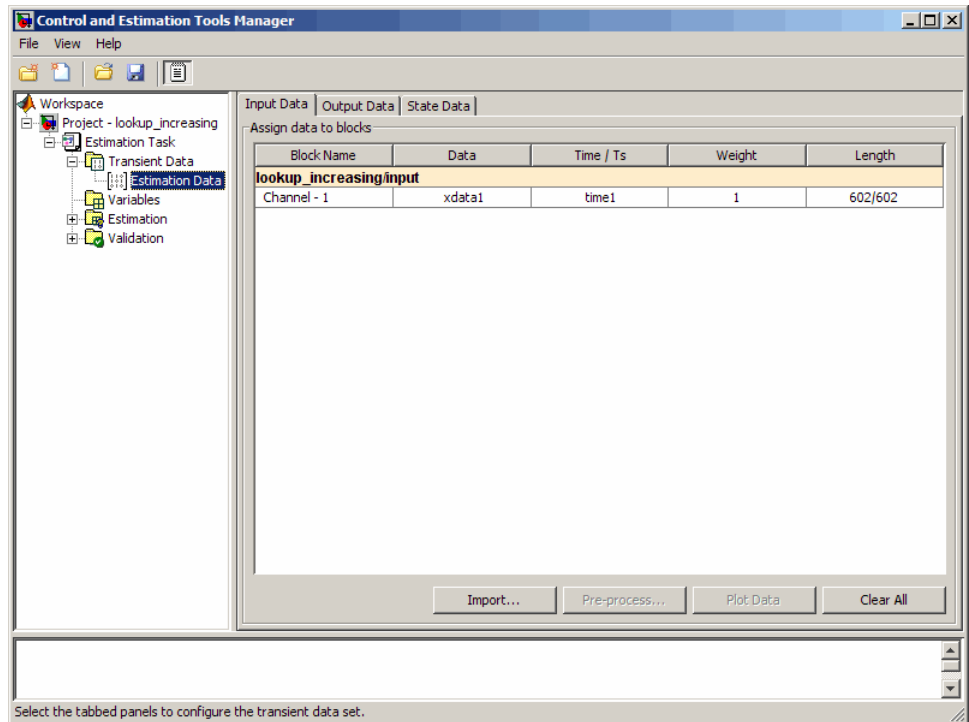
Estimating the Monotonically Increasing Table Values Using Default Settings

After you configure a project for parameter estimation, as described in “Configuring a Project for Parameter Estimation” on page 8-21, use the following steps to estimate the constrained lookup table values:

- 1 Import the estimation data, as described in the “Importing Data into the GUI” on page 2-6 section of Chapter 2, “Tutorial — Preparing Data for Parameter Estimation Using the GUI”.

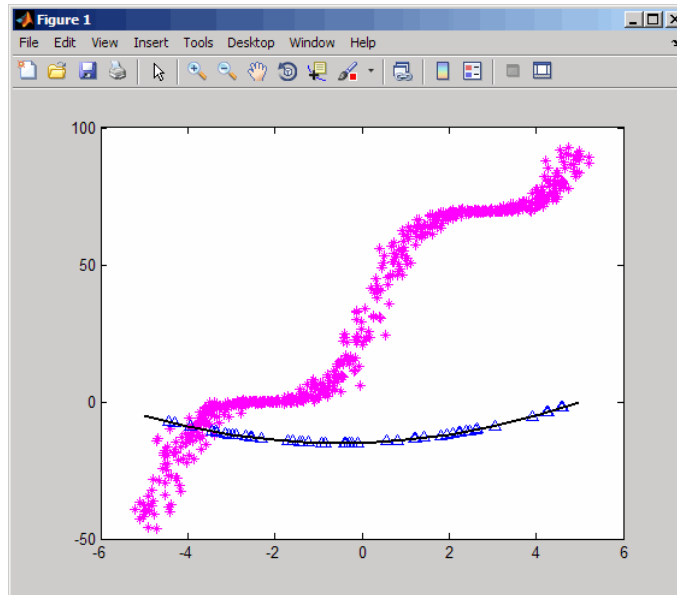
You can also load a preconfigured project that already contains the imported data. To do so, type the following commands at the MATLAB prompt:

```
lookup_increasing;  
speoad('lookup_increasing_import', 'Project - lookup_increasing',...  
      'Estimation Data')
```



- 2 Run an initial simulation to view the measured data, simulated table values and the initial table values by typing the following commands at the MATLAB prompt:

```
sim('lookup_increasing')  
figure(1); plot(xdata1,ydata1, 'm*', xout, yout, 'b^')  
hold on; plot(-5:5, cumsum(table), 'k', 'LineWidth', 2)
```



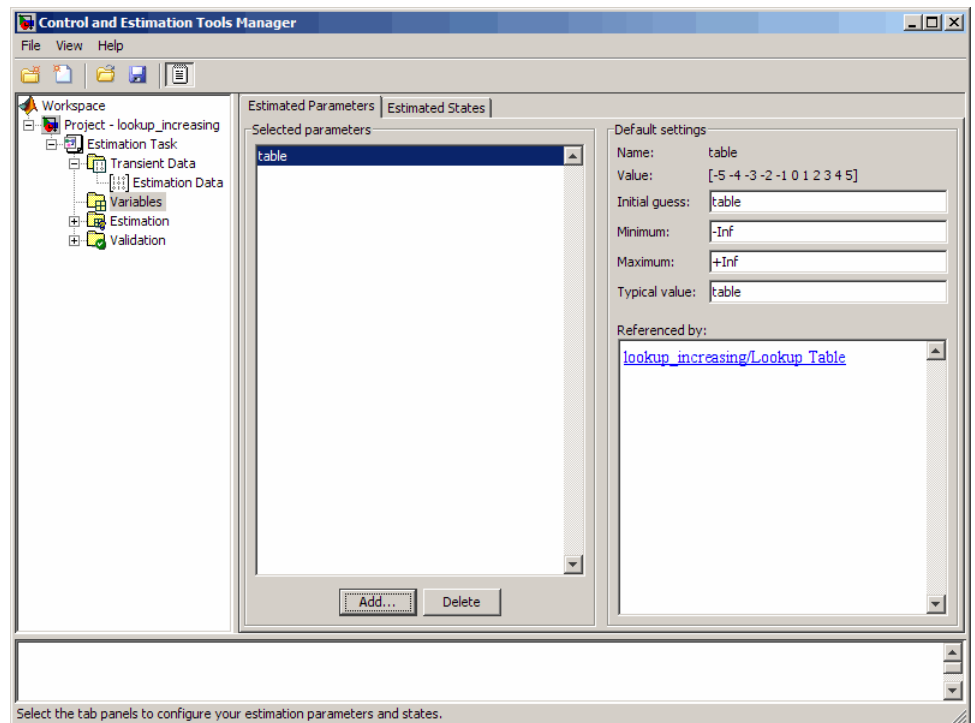
The x- and y-axes represents the input and output data, respectively. The figure shows the following plots:

- Measured data — Represented by the magenta stars (*).

Note As described in “About the Data” on page 8-21, the output data is a monotonically increasing function of the input data.

- Initial table values — Represented by the black line.
- Initial simulation data — Represented by the blue deltas (Δ).

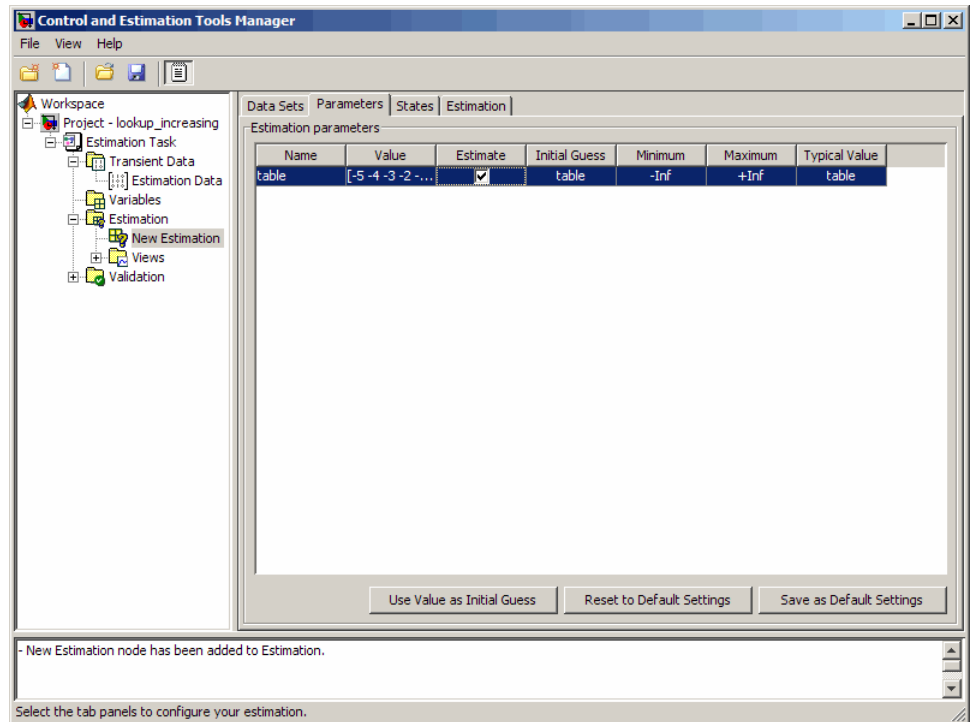
- 3 Select the table output values to estimate.
 - a In the Control and Estimation Tools Manager GUI, select the **Variables** node under the **Estimation Task** node.
 - b Click **Add** to open the Select Parameters dialog box, where you see the Simulink model parameters.
 - c Select **table**, and click **OK** to add the table values to the **Estimated Parameters** tab.



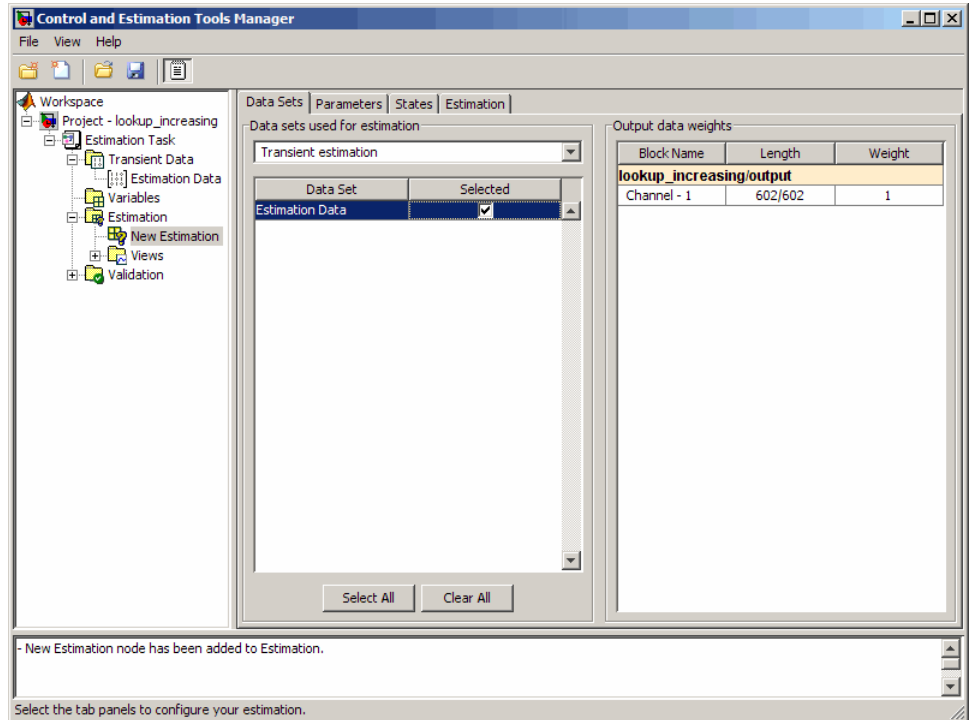
The **Default settings** area of the GUI displays the default settings for the table values. The **Value** field displays the initial table values.

- d Select the **Estimation** node, and click **New** to add a **New Estimation** node.

- e In the **Parameters** tab of the **New Estimation** node, select the lookup table values **table** for estimation.



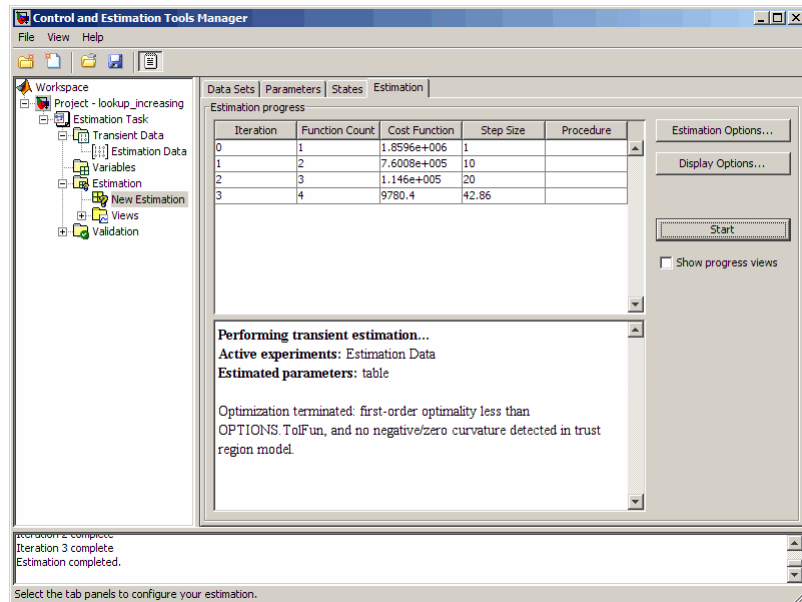
- 4 In the **Data Sets** tab of the **New Estimation** node, select the **Selected** check-box to specify the estimation data set.



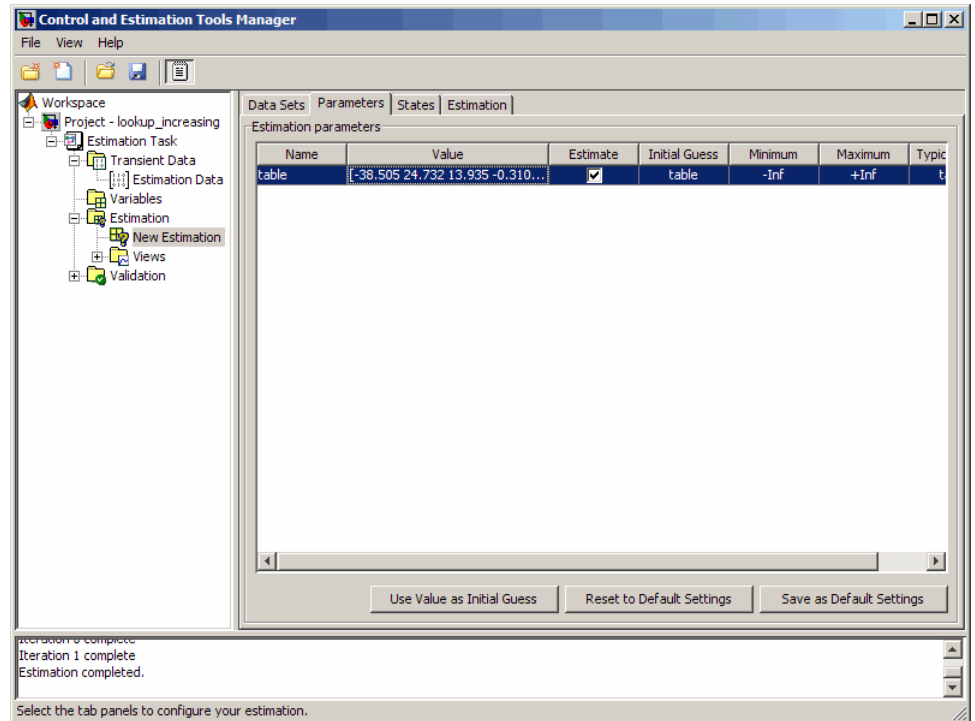
- 5 Estimate the parameters using the default settings.

- In the **Estimation** tab of the **New Estimation** node, click **Start** to start the estimation.

The Control and Tools Manager GUI updates at each iteration, and provides information about the estimation progress. After the estimation completes, the Control and Estimation Tools Manager GUI looks similar to the following figure.



- b Select the **Parameters** tab of the **New Estimation** node to view the estimated table values. The **Value** field displays the estimated table values.



Validating the Estimation Results

After you estimate the table values, as described in “Estimating the Monotonically Increasing Table Values Using Default Settings” on page 8-24, you must use another data set to validate that you have not overfitted the model. You plot and examine the following plots to validate the estimation results:

- Residuals plot
- Measured and simulated data plots

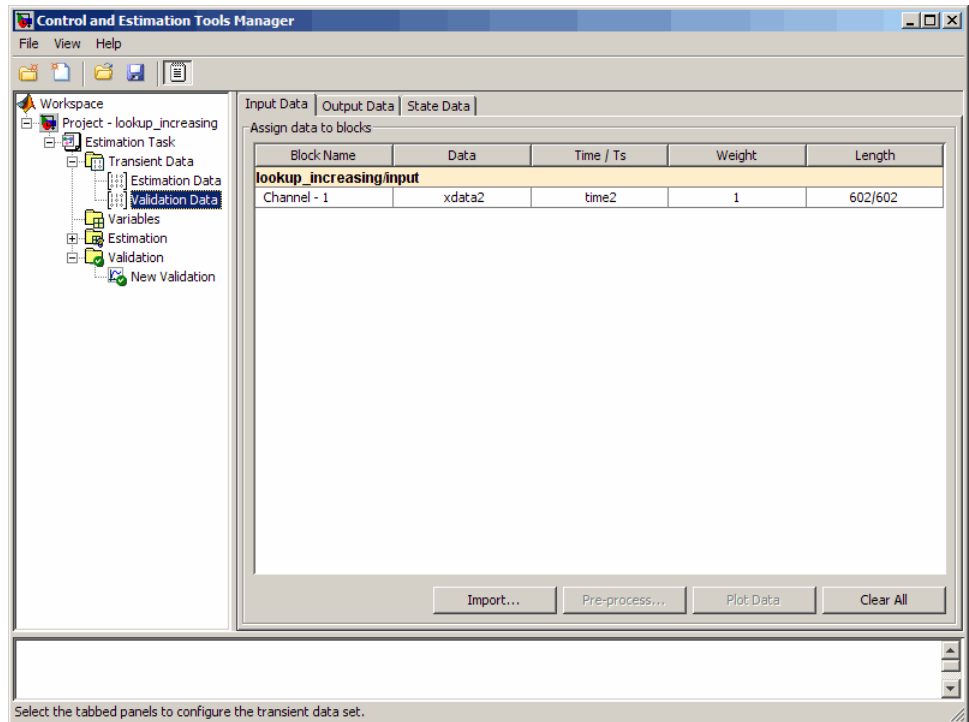
To validate the estimation results:

- 1 Import the validation data set in the Control and Estimation Tools Manager GUI, as described in “Importing Data into the GUI” on page 2-6 section of Chapter 2, “Tutorial — Preparing Data for Parameter Estimation Using the GUI”.

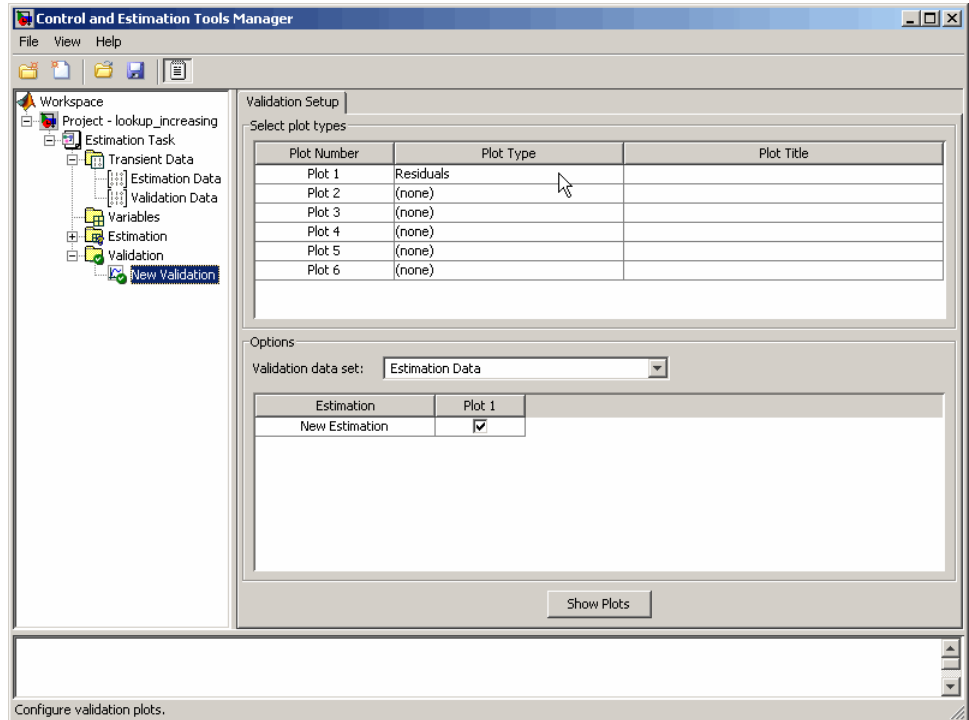
The validation data contains the input data, output data and time vector in the MATLAB variables `xdata2`, `ydata2` and `time2` respectively.

You can load a project that already contains the estimated parameters, validation data set, and residuals plot. To do so, type the following commands at the MATLAB prompt:

```
lookup_increasing;
spload('lookup_increasing_val', 'Project - lookup_increasing', ...
'Validation Data')
```

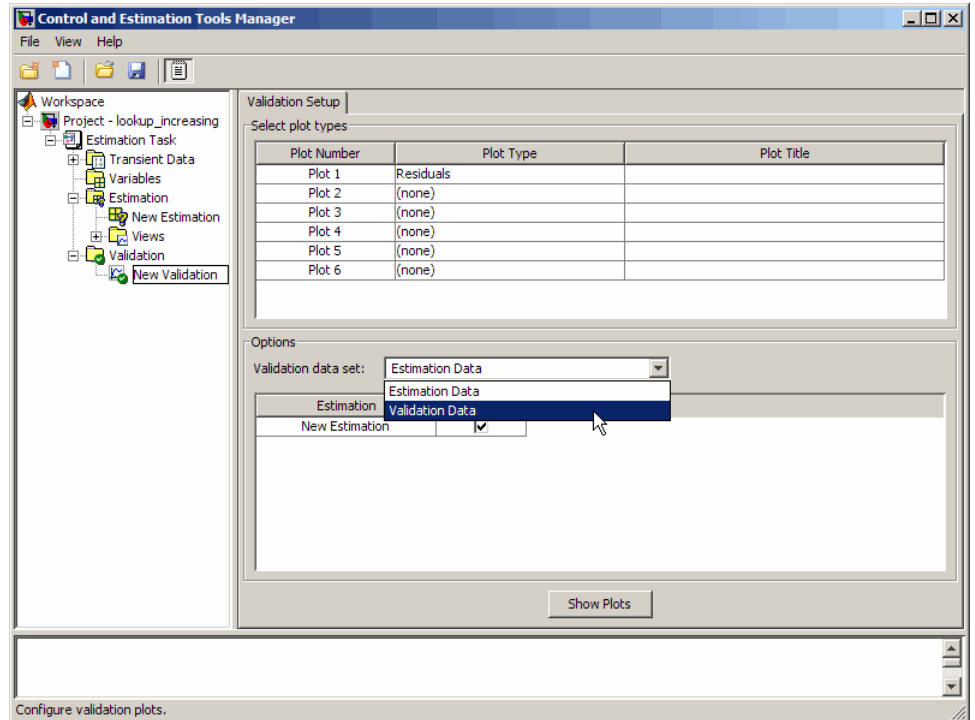


This project also contains the **Residuals** plot already configured in the **Select plot types** area of the GUI, as shown in the next figure.

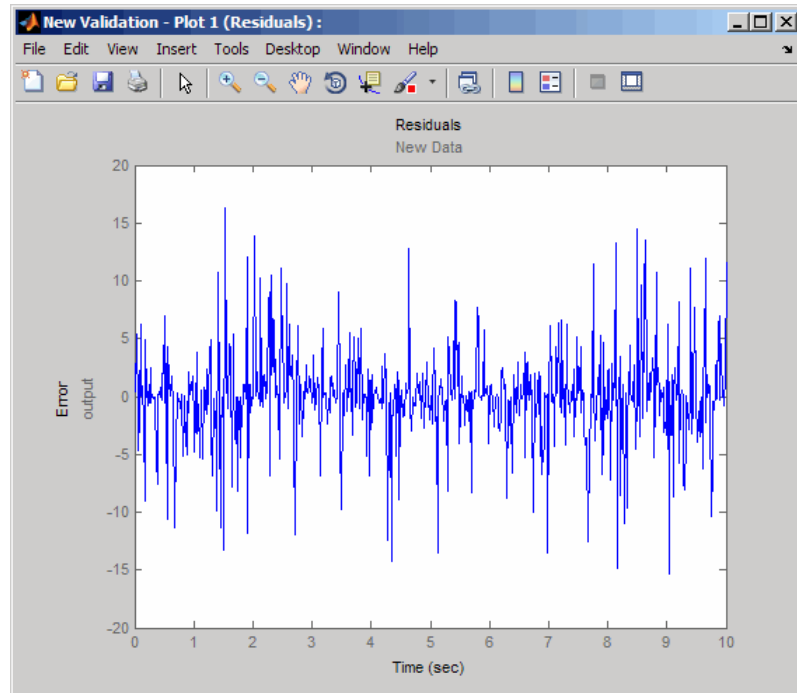


- 2 Plot and examine the residuals.
 - a Select the **New Validation** node under the **Validation** node.

- b In the **Options** area, select **Validation Data** from the **Validation data set** drop-down list.



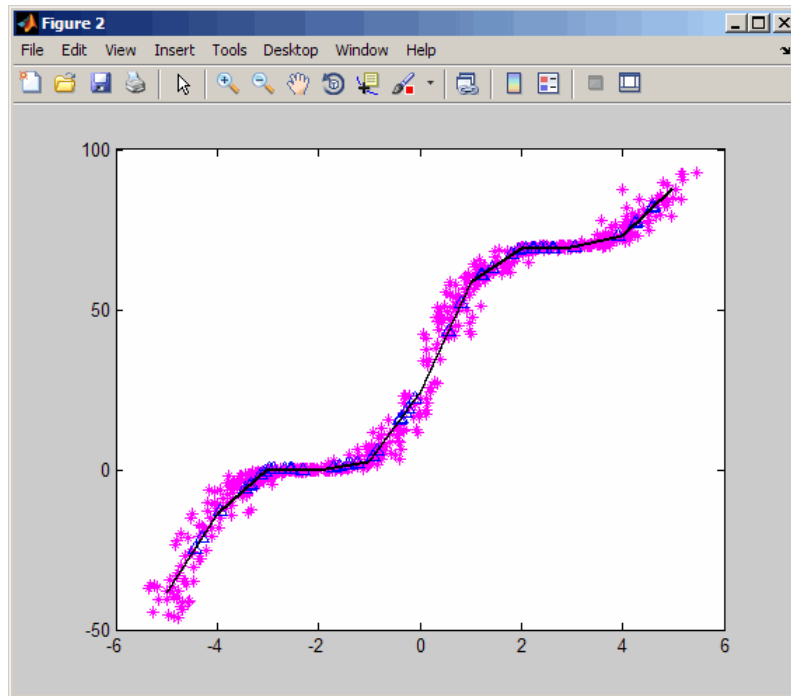
- c Click **Show Plots** to open the residuals plot.



The residuals, which show the difference between the simulated and measured data, lie the range $[-15,15]$ — within 20% of the maximum output variation. This indicates a good match between the measured and the simulated table data values.

- 3 Plot and examine the validation data, simulated data and estimated table values.

```
sim('lookup_increasing')  
figure(2); plot(xdata2,ydata2, 'm*', xout, yout, 'b^')  
hold on; plot(-5:5, cumsum(table), 'k', 'LineWidth', 2)
```



The plot shows that the table values, shown as the black line, match both the measured data and the simulated table values. The table data values cover the entire range of input values, which indicates that all the lookup table values have been estimated.

Capturing Time-Varying System Behavior Using Adaptive Lookup Tables

In this section...

“Building Models Using Adaptive Lookup Table Blocks” on page 8-37

“Setting Adaptive Lookup Table Parameters” on page 8-40

“Selecting an Adaptation Method” on page 8-41

“Example: n-D Adaptive Lookup Table” on page 8-43

“Using Adaptive Lookup Tables in Real-Time Environment” on page 8-46

Building Models Using Adaptive Lookup Table Blocks

You can use the adaptive lookup table blocks in Simulink Parameter Estimation software to create lookup tables from measured or simulated data. For more information, see “Adaptive Lookup Tables” on page 8-3.

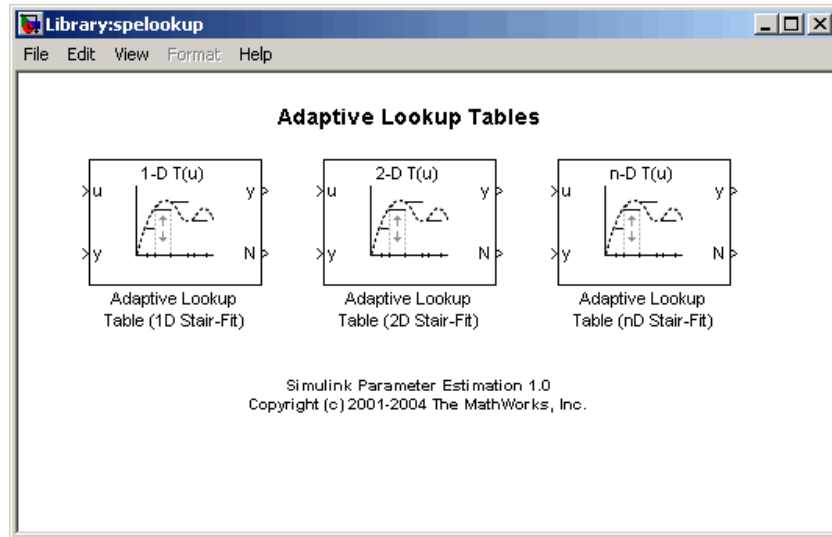
The Adaptive Lookup Table library has the following three blocks:

- Adaptive Lookup Table (1D Stair-Fit) — One-dimensional adaptive lookup table
- Adaptive Lookup Table (2D Stair-Fit) — Two-dimensional adaptive lookup table
- Adaptive Lookup Table (nD Stair-Fit) — Multidimensional adaptive lookup table

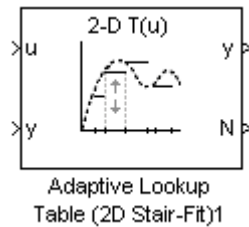
Note Use the n-D Adaptive Lookup Table block to create lookup tables of three or more dimensions.

To access the Adaptive Lookup Tables library, type the following command at the MATLAB prompt:

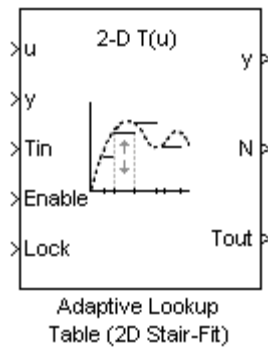
```
spellookup
```



By default, the Adaptive Lookup Table blocks have two inputs and outputs as shown in the next figure.



You can display additional inputs and outputs in a block by selecting the corresponding options in the Function Block Parameters dialog box. To learn more about the options, see Chapter 10, “Block Reference”.



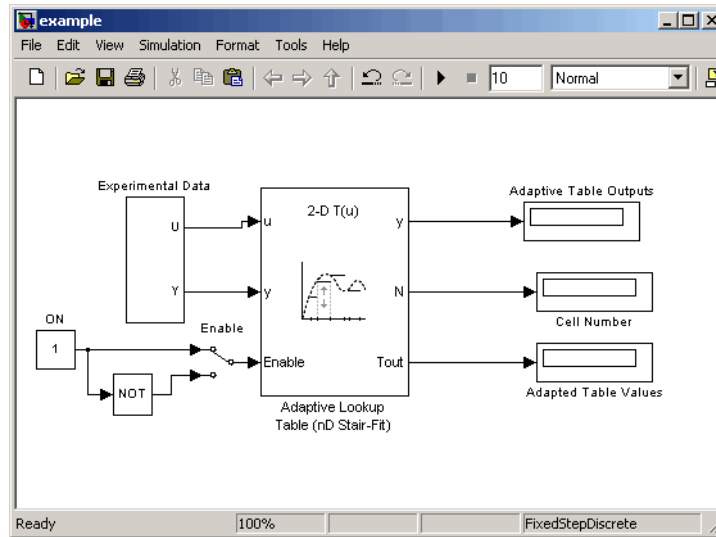
Adaptive Lookup Table Block Showing Inputs and Outputs

The 2-D Adaptive Lookup Table block has the following inputs and outputs:

- u and y — Input and output data of the system being modeled, respectively
 For example, to model an engine's efficiency as a function of engine rpm and manifold pressure, specify u as the rpm, y as the pressure, and y as the efficiency signals.
- T_{in} — The initial table data
- $Enable$ — Signal to enable, disable, or reset the adaptation process
- $Lock$ — Signal to update only specified cells in the table
- y — Value of the cell currently being adapted
- N — Number of the cell currently being adapted
- T_{out} — Values of the adapted table data

For more information on how to use adaptive lookup tables, see Chapter 4, "Tutorial — Modeling a System Using Adaptive Lookup Table".

A typical Simulink diagram using an adaptive lookup table block is shown in the next figure.

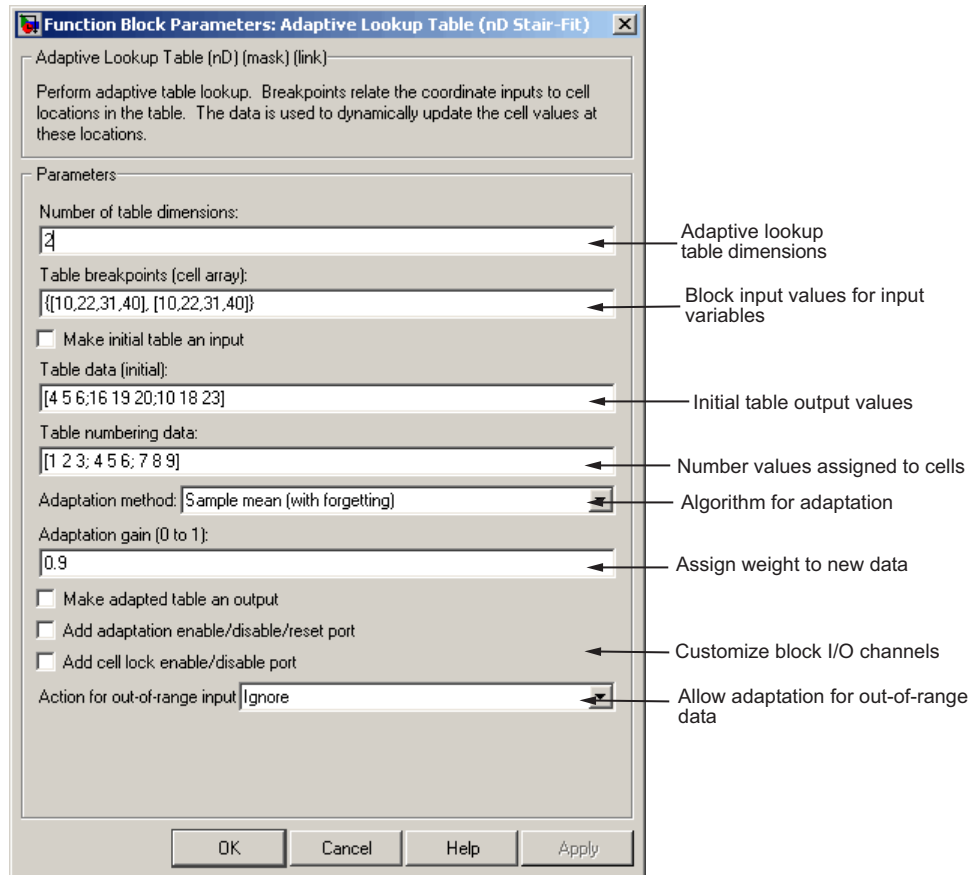


Simulink® Diagram Using an Adaptive Lookup Table

In this figure, the Experiment Data block imports a set of experimental data into Simulink through MATLAB workspace variables. The initial table is specified in the block mask parameters. When the simulation runs, the initial table begins to adapt to new data inputs and the resulting table is copied to the block's output.

Setting Adaptive Lookup Table Parameters

You can configure the Adaptive Lookup Table parameters in the Function Block Parameters dialog box. Double-click the block to open the dialog box shown in the next figure.



n-D Adaptive Lookup Table Dialog Box

For details on how to set these parameters, see the individual Chapter 10, “Block Reference” reference pages.

Selecting an Adaptation Method

You can select an adaptation algorithm from the **Adaptation Method** drop-down list in the Function Block Parameters dialog box. This section discusses the details of these algorithms.

Sample Mean

Sample mean provides the average value of n output data samples and is defined as:

$$\hat{y}(n) = \frac{1}{n} \sum_{i=1}^n y(i)$$

where $y(i)$ is the i^{th} measurement collected within a particular *cell*. For each input data u , the sample mean at the corresponding cell is updated using the output data measurement, y . Instead of accumulating n samples of data for each cell, a recursive relation is used to calculate the sample mean. The recursive expression is obtained by the following equation:

$$\hat{y}(n) = \frac{1}{n} \left[\sum_{i=1}^{n-1} y(i) + y(n) \right] = \frac{n-1}{n} \left[\frac{1}{n-1} \sum_{i=1}^{n-1} y(i) \right] + \frac{1}{n} y(n) = \frac{n-1}{n} \hat{y}(n-1) + \frac{1}{n} y(n)$$

where $y(n)$ is the n^{th} data sample.

Defining *a priori estimation error* as $e(n) = y(n) - \hat{y}(n-1)$, the recursive relation can be written as:

$$\hat{y}(n) = \hat{y}(n-1) + \frac{1}{n} e(n)$$

where $n \geq 1$ and the initial estimate $\hat{y}(0)$ is arbitrary.

In this expression, only the number of samples, n , for each cell—rather than n data samples—is stored in memory.

Sample Mean with Forgetting

The adaptation method “Sample Mean” on page 8-42 has an *infinite memory*. The past data samples have the same weight as the final sample in calculating the sample mean. Sample mean (with forgetting) uses an algorithm with a *forgetting factor* or **Adaptation gain** that puts more weight on the more recent samples. This algorithm provides robustness against initial response transients of the plant and an adjustable speed of adaptation. Sample mean (with forgetting) is defined as:

$$\begin{aligned}\hat{y}(n) &= \frac{1}{\sum_{i=1}^n \lambda^{n-i}} \sum_{i=1}^n \lambda^{n-i} y(i) \\ &= \frac{1}{\sum_{i=1}^n \lambda^{n-i}} \left[\sum_{i=1}^{n-1} \lambda^{n-i} y(i) + y(n) \right] = \frac{s(n-1)}{s(n)} \hat{y}(n-1) + \frac{1}{s(n)} y(n)\end{aligned}$$

where $\lambda \in [0,1]$ is the **Adaptation gain** and $s(k) = \sum_{i=1}^k \lambda^{n-i}$.

Defining *a priori estimation error* as $e(n) = y(n) - \hat{y}(n-1)$, where $n \geq 1$ and the initial estimate $\hat{y}(0)$ is arbitrary, the recursive relation can be written as:

$$\hat{y}(n) = \hat{y}(n-1) + \frac{1}{s(n)} e(n) = \hat{y}(n-1) + \frac{1-\lambda}{1-\lambda^n} e(n)$$

A small value of λ results in faster adaptation. A value of 0 indicates short memory (last data becomes the table value), and a value of 1 indicates long memory (average all data received in a cell).

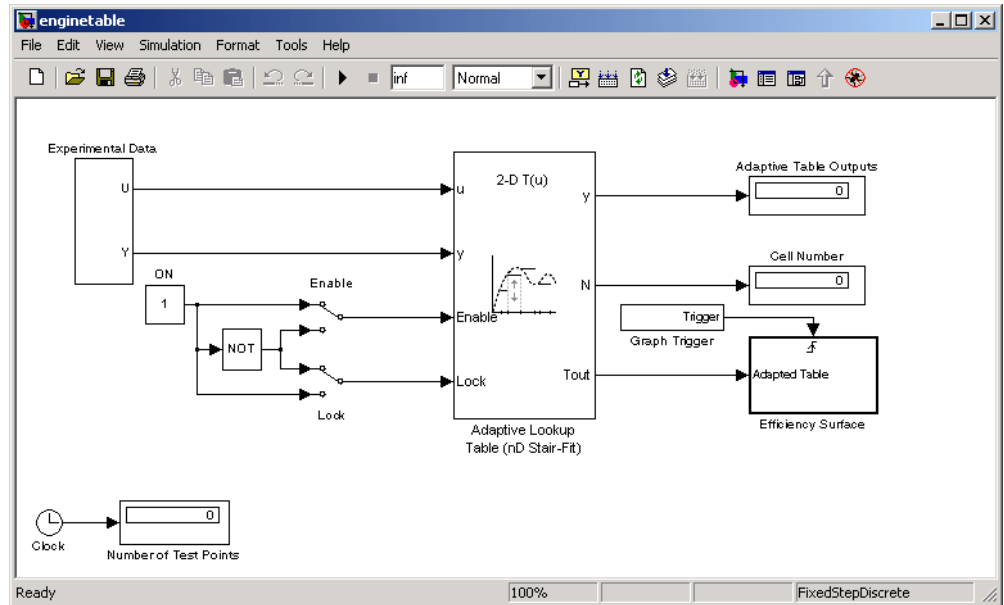
Example: n-D Adaptive Lookup Table

- “Loading the Example” on page 8-43
- “Running the Example” on page 8-44

Loading the Example

This example shows an n-D adaptive lookup table at work and includes many of the key features associated with adaptive lookup tables. Type the following command at the MATLAB prompt to open this model:


```
enginetable
```



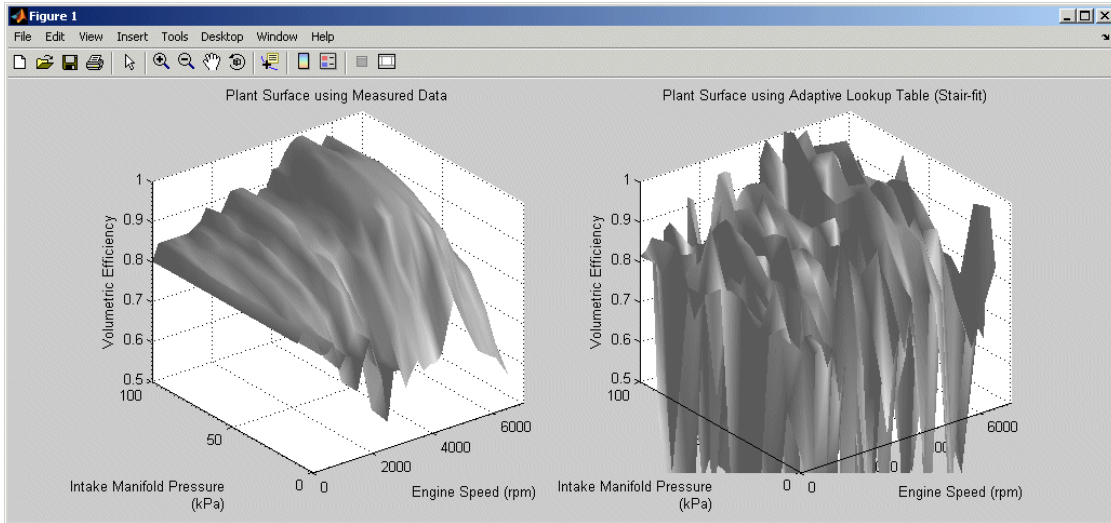
This model has several key features:

- Input — The adaptive lookup table input is the experimental data. It is also possible to make the original table itself an input.
- An enable feature — You can turn the adaptation on and off during the estimation to see how the basic features work.
- A lock feature — You can lock the table so that only one cell is adapting. You may find this feature useful if you have one section in your data that is highly erratic or otherwise difficult for the algorithm to handle.
- Output — Adaptive lookup table values.

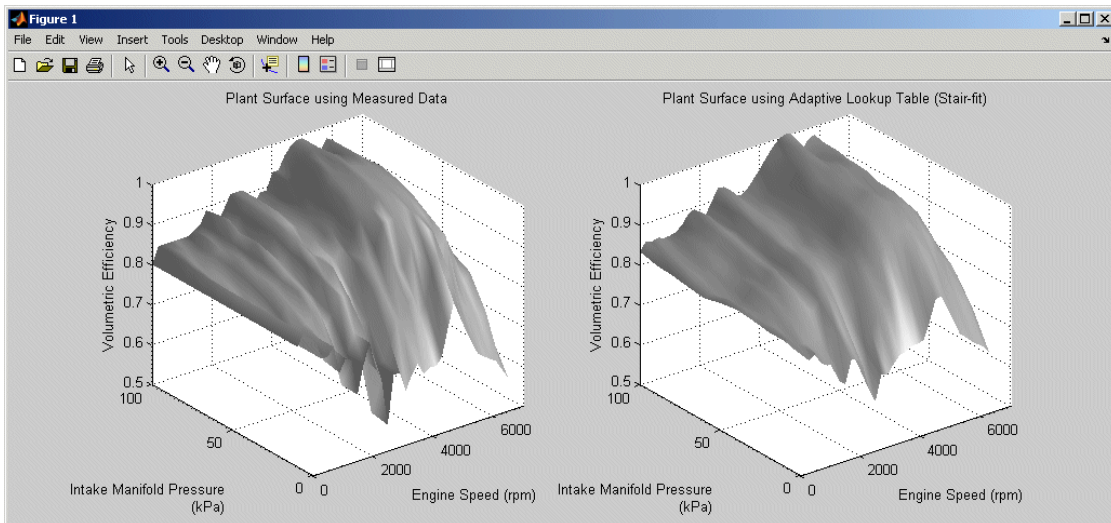
Running the Example

To start the `enginetable` simulation, select **Simulation > Start**. Alternatively, you can click the **Start Simulation** button  in the Simulink toolbar.

The simulation begins by populating the adaptive lookup table with random data. This figure shows the input and adapted data side by side.



As the simulation progresses, the surface on the right adapts to match the measured input data. The next figure shows the final adaptation.



The simulation indicates a very good fit. Next, try using the enable and lock features to see how they change the adaptation.

Using Adaptive Lookup Tables in Real-Time Environment

You can use experimental data from sensor measurements collected by running various tests on a system in real time. The measured data is then sent to the adaptive table block to generate a lookup table describing the relation between the system inputs and output.

You can also use the Adaptive Lookup Table block in a real-time environment, where some time-varying properties of a system need to be captured. To do so, generate C code using Real-Time Workshop® code generation software that can then be run in an xPC Target™ or dSPACE® software. Because you can start, stop, or reset the adaptation if you want, use logic to enable the adaptation of the table data only when it is desired. The cell number output *N*, and the *Enable* and *Lock* inputs facilitate this process. Use the *Enable* input to start and stop the adaptation and the *Lock* input to update only one of the table cells. The *Lock* input combined with some logic using the cell number output *N* provide the means for updating only the desired table cells during a simulation run.

Estimating from the Command Line

- “Introduction” on page 9-2
- “Example: Estimating Parameters and Initial Conditions of the F14 Model” on page 9-4
- “Creating and Customizing Estimation Projects” on page 9-14
- “Creating Transient Data Objects” on page 9-15
- “Creating State Data Objects” on page 9-20
- “Creating Transient Experiment Objects” on page 9-23
- “Creating Parameter Objects” on page 9-26
- “Creating State Objects” on page 9-30
- “Creating Estimation Objects” on page 9-34

Introduction

In addition to the Control and Estimation Tools Manager GUI, you can also use Simulink Parameter Estimation functions to perform parameter and state estimation. These functions perform the same tasks as the tools manager, but have the advantages of command-line execution. When you perform a state or parameter estimation using Simulink Parameter Estimation GUI, you create MATLAB objects for all the states and parameters of your model. If you have a large number of states or parameters, this can use up large amounts of memory and cause computational delays. With the command-line approach, only those states and parameters that you select are assigned MATLAB objects, which is more efficient.

In addition, the command-line approach is useful for batch jobs where you can estimate parameters for a large numbers of models.

Simulink Parameter Estimation software uses MATLAB objects to perform estimation tasks. This chapter discusses what you need to know about object-oriented programming for using this product. See the *Object-Oriented Programming* documentation for a description of object-oriented programming in MATLAB.

Simulink Parameter Estimation command-line interface requires a Simulink model as a starting point for analysis and estimation.

Note The Simulink model must contain an Inport or Outport block or logged signals to enable assigning data to the signals. For more information on how to prepare a Simulink model for parameter estimation, see “Preparing a Model for Parameter Estimation” on page 1-8.

After you prepare your model for parameter estimation, as described in “Preparing a Model for Parameter Estimation” on page 1-8, the estimation process consists of the following steps:

- 1 Defining experiments consisting of empirical data sets, and the operating conditions and/or initial conditions of your model.
- 2 Selecting the variables and states to be estimated.

- 3** Performing the estimation.
- 4** Reviewing the results and iterating as necessary.
- 5** Validating estimation results.

The following sections discuss these topics:

- “Example: Estimating Parameters and Initial Conditions of the F14 Model” on page 9-4 — How to perform the estimation process using command-line functions
- “Creating and Customizing Estimation Projects” on page 9-14 — How to use methods and properties to customize your estimation project’s features

Example: Estimating Parameters and Initial Conditions of the F14 Model

In this section...
“Loading the F14 Jet Model” on page 9-4
“Baseline Simulation” on page 9-5
“Creating a Transient Experiment Object” on page 9-7
“Assigning Experimental Data to Inputs and Outputs of the Model” on page 9-8
“Creating Parameter Objects for Estimation” on page 9-9
“Creating an Estimation Object and Running the Estimation” on page 9-10

Loading the F14 Jet Model

To define an experiment, you must start with a Simulink model. For this example, type

```
f14
```

to load the F14 fighter jet model into the MATLAB workspace. The following figure shows the f14 model.

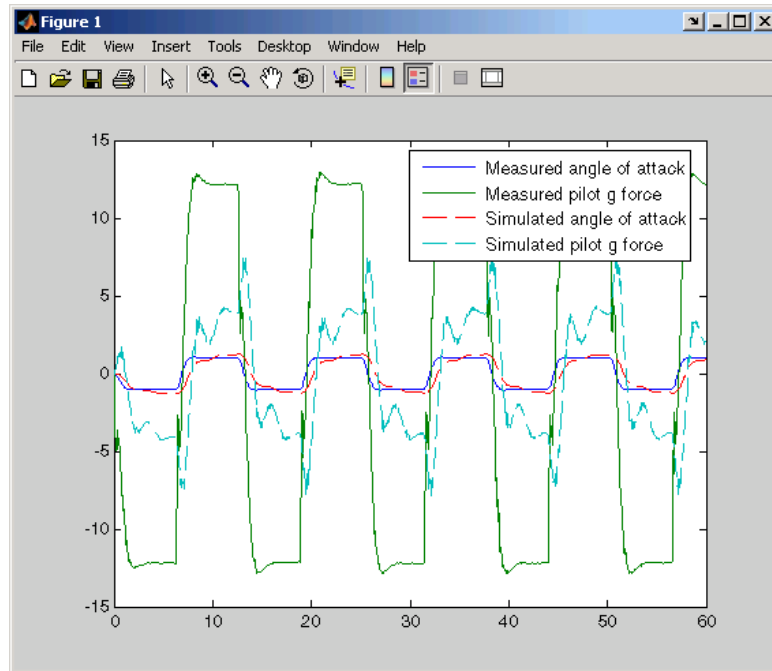

```
%% Open the model and load experimental data.
open_system('f14')
load f14_estim % Load empirical I/O data.

%% Set initialize unknown parameters
% Actuator time constant (ideal: Ta = 0.05)
Ta = 0.5;

% Aircraft dynamic model parameters (ideal: Md = -6.8847,
% Zd = -63.998)
Md = -1; Zd = -80;

%% Plot measured data and simulation results
[T,X,Y] = sim('f14', time, [], [time iodata(:,1)]);
plot(time, iodata(:,2:3), T, Y, '-');
legend( 'Measured angle of attack', 'Measured pilot g force', ...
        'Simulated angle of attack', 'Simulated pilot g force');
```

The following figure appears.



As you can see, the measured and simulated data are a poor match. The rest of this section describes how to estimate values for T_a , Z_d , and M_d that result in a better match of data sets.

Creating a Transient Experiment Object

After you have a model and identify the parameters you want to estimate, the next step is to create the objects required for an estimation. `ParameterEstimator` is both the name of the *class* and the *object* instantiated by that class. Classes are created by a *constructor*; objects are created by invoking the class name with parameters.

First, create an estimation project object. This is the constructor syntax:

```
hExp = ParameterEstimator.TransientExperiment('f14')
```

This command returns the following information about the f14 model.

Experimental transient data set for the model 'f14':

Output Data

- (1) f14/alpha (rad)
- (2) f14/Nz Pilot (g)

Input Data

- (1) f14/u

Initial States

- (1) f14/Actuator Model
- (2) f14/Aircraft Dynamics Model/Transfer Fcn.1
- (3) f14/Aircraft Dynamics Model/Transfer Fcn.2
- (4) f14/Controller/Alpha-sensor Low-pass Filter
- (5) f14/Controller/Pitch Rate Lead Filter
- (6) f14/Controller/Proportional plus integral compensator
- (7) f14/Controller/Stick Prefilter
- (8) f14/Dryden Wind Gust Models/Q-gust model
- (9) f14/Dryden Wind Gust Models/W-gust model

Assigning Experimental Data to Inputs and Outputs of the Model

After you create a ParameterEstimator object, assign input and output experimental (i.e., empirical) data.

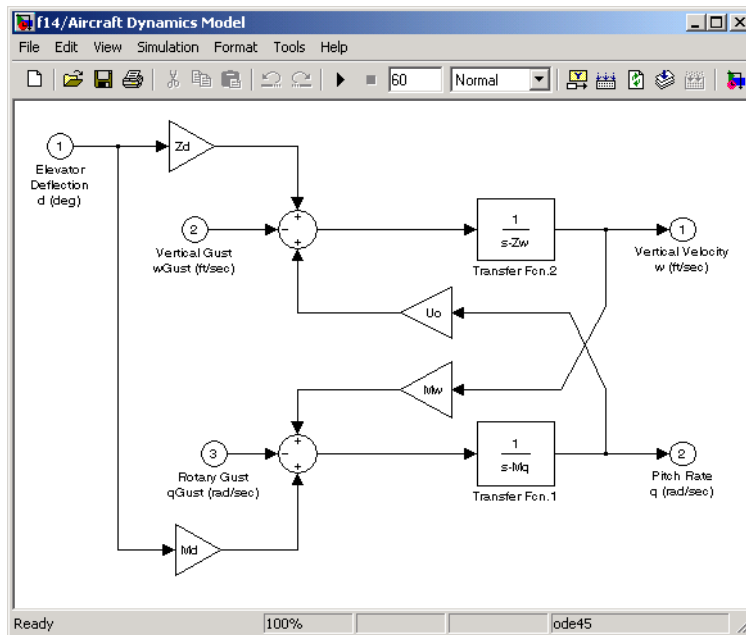
```
%% Create objects to represent the experimental data sets.
set(hExp.InputData(1), 'Data', iodata(:,1), 'Time', time);

set(hExp.OutputData(1), 'Data', iodata(:,2), 'Time', ...
    time, 'Weight', 5);
set(hExp.OutputData(2), 'Data', iodata(:,3), 'Time', time);
```

Note In general, for models with multiple inputs and outputs, you must independently assign one data object to each input and output port. The data object you assign to a specific port can be a vector or a matrix that corresponds to that channel. You cannot use a single I/O port to represent multiple channels.

Creating Parameter Objects for Estimation

To activate parameters for estimation, you must create parameter objects for the parameters you want to estimate. For this example, use T_a , the actuator time constant, and Z_d and M_d , the vertical velocity and pitch rate gains, respectively. The Z_d and M_d gains are located in the F14 aircraft dynamics subsystem.



First, create `ParameterEstimator` objects for the parameters you want to estimate.

```

%% Create objects to represent parameters.
hPar(1) = ParameterEstimator.Parameter('Ta');
set(hPar(1), 'Minimum', 0.01, 'Maximum', 1, 'Estimated', true)

hPar(2) = ParameterEstimator.Parameter('Md');
set(hPar(2), 'Minimum', -10, 'Maximum', 0, 'Estimated', true)

hPar(3) = ParameterEstimator.Parameter('Zd');
set(hPar(3), 'Minimum', -100, 'Maximum', 0, 'Estimated', true)

```

```
% Create objects to represent initial states.  
hIc(1) = ParameterEstimator.State('f14/Actuator Model');  
set(hIc(1), 'Minimum', 0, 'Estimated', false);
```

You can also use dot notation here. For example, instead of

```
set(hPar(2), 'Minimum', -10, 'Maximum', 0, 'Estimated', true)
```

you can write

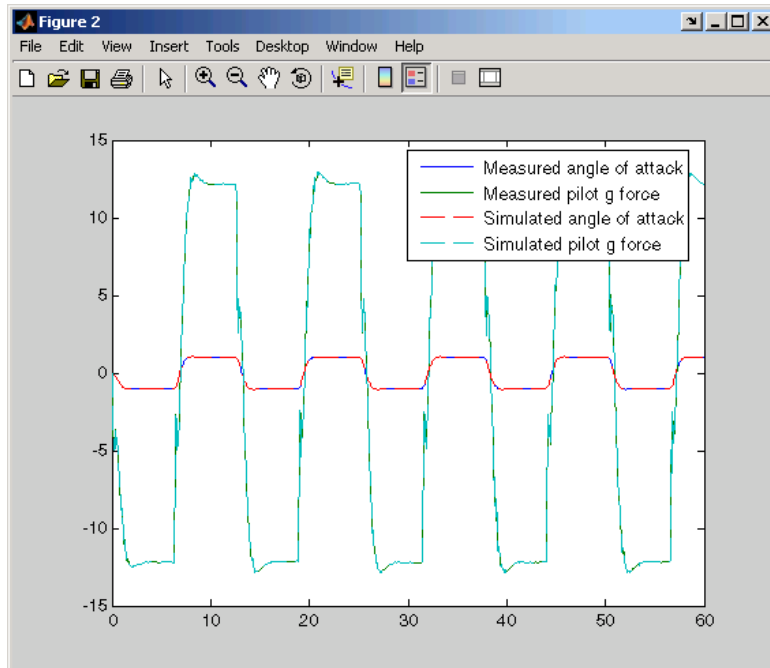
```
hPar(2).Estimated=true;  
hPar(2).Minimum=-10;  
hPar(2).Maximum=0;
```

Creating an Estimation Object and Running the Estimation

Finally, create an estimation object and run the estimation, using `gcs` to get the full pathname to the Simulink model.

```
hEst = ParameterEstimator.Estimation(gcs, hPar, hExp);  
hEst.States = hIc;  
  
%% Setup estimation options  
hEst.OptimOptions.Algorithm = 'lsqnonlin';  
hEst.OptimOptions.GradientType = 'refined';  
hEst.OptimOptions.Display = 'iter';  
  
%% Run the estimation  
estimate(hEst);  
  
%% Plot measured data and final simulation results  
[T,X,Y] = sim('f14', time, [], [time iodata(:,1)]);  
figure  
plot(time, iodata(:,2:3), T, Y, '-');  
legend( 'Measured angle of attack', 'Measured pilot g force', ...  
        'Simulated angle of attack', 'Simulated pilot g force');
```


This figure shows the results of the estimation.



The measured and simulated outputs now appear to be a close match. Next, look at the estimated values to see how they compare with the default values of the f14 model.

```
%% Look at the estimated values  
find(hEst.Parameters, 'Estimated', true)
```

This command returns the following result:

(1) Parameter data for 'Ta':

Parameter value : 0.05

Initial guess : 0.5

Estimated : true

Referenced by:

(2) Parameter data for 'Md':

Parameter value : -6.884

Initial guess : -1

Estimated : true

Referenced by:

(3) Parameter data for 'Zd':

Parameter value : -63.99

Initial guess : -80

Estimated : true

Referenced by:

Note You can use the `find` command to identify scalar, vector, or matrix parameters. The dimensions of the `Estimated` value you specify as the `find` argument must match the dimensions of the parameters you are trying to find. For example,

```
find(hEst.Parameters, 'Estimated', true)
```

finds only scalar estimated parameters. However,

```
find(hEst.Parameters, 'Estimated', [true;true])
```

finds only vector estimated parameters with dimensions 1-by-2 and excludes all scalar parameters.

You can verify that these values match the default values of the `f14` model by clearing your workspace, loading the model, and checking the values.

```
clear all
f14
whos
```

Creating and Customizing Estimation Projects

The following sections describe in more detail how to create and modify transient data and estimation objects:

- “Creating Transient Data Objects” on page 9-15
- “Creating State Data Objects” on page 9-20
- “Creating Transient Experiment Objects” on page 9-23
- “Creating Parameter Objects” on page 9-26
- “Creating State Objects” on page 9-30
- “Creating Estimation Objects” on page 9-34

First, a quick look at terminology:

- *Objects* are instantiations of *classes*.
- *Classes* contain, or rather, define, *properties* and *methods*.
- You use a *constructor* to create an instance of an object, and use the `set` method or dot notation to modify the properties of your objects.

Creating Transient Data Objects

- “What is a Transient Data Object” on page 9-15
- “Constructor” on page 9-15
- “Properties of Transient Data Objects” on page 9-16
- “Modifying Properties of Transient Data Objects” on page 9-18
- “Using Class Methods” on page 9-19

What is a Transient Data Object

The `@TransientData` object encapsulates the data measured at a single input or output of a physical system during an experiment. Transient data objects are associated with three types of Simulink blocks:

- Inport blocks
- Outport blocks
- Internal blocks used in conjunction with signal logging.

Each `@TransientData` object describes the time history of a signal at a Simulink port. A data set is identified by the `Block` property of this object corresponding to a block name in the Simulink model. A `PortNumber` value is also necessary for internal blocks to uniquely identify signals within the block diagram.

Constructor

Estimating parameters requires a transient data object, which you create using a constructor. The syntax to create a transient data object is

```
% I/O port block
h = ParameterEstimator.TransientData('block');
% Internal block
h = ParameterEstimator.TransientData('block',portnumber);

h = ParameterEstimator.TransientData('block',data,time);
h = ParameterEstimator.TransientData('block',data,Ts);
h = ParameterEstimator.TransientData('block',portnumber,data,time);
```

```
h = ParameterEstimator.TransientData('block', portnumber, data, Ts);
```

Properties of Transient Data Objects

Descriptions of properties of the transient data object and the associated input parameters are as follows.

Transient Data Object Properties

Property	Description
Block	Name of the Simulink block with which the data is associated. Must be a string.
PortType	The type of signal that this object represents is determined in the constructor from the Block property, which may be Inport, Outport, or Signal.
PortNumber	For data associated with the outputs of regular blocks or subsystems, this property specifies the output port number of interest. The default value is 1.
Dimensions	Dimensions of the data required for this data set. It is computed from the CompiledPortDimensions property of the appropriate port of the block, and it defines the size of other properties. Currently, Simulink supports scalar, vector, or matrix signals, so Dimensions is either a scalar or a 1-by-2 array.
Data	<p>Actual experimental data. Its size must be consistent with the Dimensions property. To conform with Simulink conventions, the data is stored in the following formats:</p> <ul style="list-style-type: none"> • Scalar or vector-valued data. The data is of the form Ns m, where Ns is the number of data samples, and m is the number of channels in the signal. • Multidimensional data (matrix and higher dimensions). The data is of the form $m1 \dots mn$ Ns, where Ns is the number of data samples, and mi is the number of channels in the ith dimension of the signal. • For missing or unspecified data, NaNs are used.

Transient Data Object Properties (Continued)

Property	Description
Ts, Tstart, Tstop	<p>For uniformly sampled data, Ts is the sample time and Tstart is the start time of the signal. The stop time Tstop and the time vector Time are given by</p> $Tstop = Tstart + Ts * (Ns - 1)$ $Time = Tstart : Ts : Tstop$ <p>For nonuniform time data, Ts is set to NaN, and the start and stop times are calculated from the time vector.</p>
Time	<p>The time data in column vector format. The length of Time must be consistent with the number of samples in Data.</p> <p>For a nonuniformly spaced Time vector, its length should match the length of Data.</p> <p>Otherwise, Time is automatically adjusted based on the length of Data.</p> <p>Modifying Ts resets Time internally. In this case, Time is a virtual property whose value is computed from Ts and Tstart when you request it. The rules for setting time related properties are</p> <ul style="list-style-type: none"> • Modifying Time sets <ul style="list-style-type: none"> Ts = NaN Tstart = Time(1) • If the time vector is uniformly spaced, a sample time Ts is calculated. • Modifying Tstart translates time forward or backward. • Modifying Ts sets Time = [] internally and generates it when required by the simulation.

Transient Data Object Properties (Continued)

Property	Description
Weight	The weight associated with each channel of this data set. It is used to specify the relative importance of signals. The default value is 1.
InterSample	Interpolation method between samples can be zero-order hold (zoh) or first-order hold (foh). This property is used for data preprocessing.

Modifying Properties of Transient Data Objects

After a transient data object is created, you can modify its properties using this syntax:

```
in1.Data = rand(2,1,10); % 10 data values each of size [2 1]
in1.Time = 1:10; % Automatically converted to column vector
```

Some properties (e.g., `Weight`) support scalar expansion with respect to the value of the `Dimensions` property.

Example: Assigning Input Port Data

To assign data to an input port with 2-by-3 port dimensions, use

```
in1 = ParameterEstimator.TransientData(gcb, rand(2,3,100), 0.05)
```

This command returns the following result:

```
(1) Transient data for Inport block 'portdata_test_noSim/By//Pass
Air Valve Voltage':
Sampling interval: 0.05 sec.
Data set has 100 samples and 6 channels.
```


Using Class Methods

Descriptions of two important methods are given next:

- `select` — Extracts a portion of data. The result is returned in a new transient data object.

```
in2 = select(in1, 'Sample', 10:100); % 91 samples
in3 = select(in1, 'Range', [1 4]); % Samples for 1<t<4
% ... or an alternative
in3 = select(in1, 'Sample', find(in1.Time > 1 & in1.Time < 4));
```

To extract data from a subset of available channels, use

```
in4 = select(in1, 'Channel', [1 3 2]);
% channels 1,3,and 2 in this order
```

- `hiliteBlock` — Highlights the block associated with this object in the Simulink diagram.

Creating State Data Objects

In this section...

“What is a State Data Object” on page 9-20

“Constructor” on page 9-20

“Properties of the State Data Object” on page 9-21

“Example: Initial Condition Data” on page 9-22

“Modifying Properties” on page 9-22

“Using Class Methods” on page 9-22

What is a State Data Object

The `ParameterEstimator.StateData` object defines the states of a dynamic Simulink block. It is used in a transient estimation context to define known initial conditions of a block diagram model, and in a steady-state estimation context to define the known states of the model.

For example, the Simulink model of a simple mass-spring-damper system has two integrator blocks to generate velocity and position signals from acceleration and velocity values, respectively, during simulation. If the corresponding physical system is known to be at rest at the beginning of an experiment, the initial states (velocity and position) of these integrators are zero. So, two `@StateData` objects can be created to describe these known initial conditions.

Constructor

The syntax for creating this object is

```
h = ParameterEstimator.StateData('block');  
h = ParameterEstimator.StateData('block', data);
```

In the first constructor, the state vector is initialized from the model containing the block.

Properties of the State Data Object

Descriptions of some important properties are given in the following table .

State Data Object Properties

Property	Description
Block	Name of the Simulink block whose states are defined by this object.
Dimensions	Scalar value to store the number of states of the relevant block.
Data	<p>Column vector to store the initial value of the state for the block specified by this object. The length of this vector should be consistent with the <code>Dimensions</code> property. Since the underlying Simulink model also stores an initial state vector for all dynamic blocks, the following conventions are used to resolve the initial state values during estimations:</p> <ul style="list-style-type: none"> • If <code>Data</code> is not empty, use it when forming the state vector. • If <code>Data</code> is empty, get the state vector for this block from the model. This behavior is useful when using helper methods to create an experiment object that instantiates empty state data objects for all dynamic blocks in the Simulink model. • If there is no state data object for a dynamic block in the model, get the state vector of that block from the model. This behavior is useful for command-line users, when there are too many states in the model and only a few of them have to be set to different initial values.
<code>Ts</code>	Sampling time of discrete blocks. Set to 0 for continuous blocks. This property is read only and is currently used for information only.
Domain	String to hold the physical domain of the block. Used for <code>SimMechanics</code> or <code>SimPowerSystems</code> blocks with states.

Example: Initial Condition Data

To create an empty initial condition object for the `engine_idle_speed/TransferFcn2`, use

```
st1 = ParameterEstimator.StateData ...  
('engine_idle_speed/Transfer Fcn2', [1 2])  
  
(1) State data for 'f14/Dryden Wind Gust Models/W-gust model'  
block:  
The block has 2 continuous state(s).  
State value : [1;2]
```

Modifying Properties

After a state data object is created, you can modify its properties using this syntax:

```
st1.Data = [2 3]; % State vector of size 2
```

Some properties (e.g., `Data`) support scalar expansion with respect to the value of the `Dimensions` property.

Using Class Methods

Description of two important methods are given next:

- `hiliteBlock` — Highlights the block associated with this object in the Simulink diagram.
- `update` — Updates the object after the Simulink model has been modified. If the `Dimensions` property value changes, the other properties are reset to their default values.

Creating Transient Experiment Objects

In this section...

“What is a Transient Experiment Object” on page 9-23

“Constructor” on page 9-23

“Properties of Transient Experiment Objects” on page 9-23

“Example: Creating an F14 Experiment” on page 9-24

“Example: Creating a Van der Pol Experiment from User Objects” on page 9-25

“Modifying Properties” on page 9-25

“Using Class Methods” on page 9-25

What is a Transient Experiment Object

The @TransientExperiment object encapsulates the data measured at the input and output ports of a system during a single experiment, as well as the system’s known initial states.

Constructor

The syntax to create a transient experiment object is

```
h = ParameterEstimator.TransientExperiment('model');
```

where `model` specifies the name of the Simulink model.

Properties of Transient Experiment Objects

Descriptions of some important properties are given in the following table.

Transient Experiment Object Properties

Property	Description
Model	Simulink model with which this experiment is associated.

Transient Experiment Object Properties (Continued)

Property	Description
InputData, OutputData	<p>Transient data objects associated with appropriate I/O blocks in the model. Blocks with unassigned objects or objects with no data are not used in estimations, meaning:</p> <ul style="list-style-type: none"> • For input ports, assign zeros to these ports/channels during simulation. • For output ports, don't use these ports/channels in the cost function.
InitialStates	State data objects associated with appropriate dynamic blocks in the model.
InitFcn	Function to be executed to configure the model for this particular experiment.

Example: Creating an F14 Experiment

To create an empty transient experiment for the f14 model, use

```
exp1 = ParameterEstimator.TransientExperiment('f14')
Experimental (Transient) data set for the model 'f14':
Outputs
(1) f14/alpha (rad)
(2) f14/Nz Pilot (g)
Inputs
(1) f14/u
Initial States
(1) f14/Actuator Model
(2) f14/Aircraft Dynamics Model/Transfer Fcn.1
(3) f14/Aircraft Dynamics Model/Transfer Fcn.2
(4) f14/Controller/Alpha-sensor Low-pass Filter
(5) f14/Controller/Pitch Rate Lead Filter
(6) f14/Controller/Proportional plus integral compensator
(7) f14/Controller/Stick Prefilter
(8) f14/Dryden Wind Gust Models/Q-gust model
```

(9) f14/Dryden Wind Gust Models/W-gust model

Example: Creating a Van der Pol Experiment from User Objects

To create a transient experiment from user objects for I/Os and states, use

```
out1 = ParameterEstimator.TransientData('vdp/Out1');
ic1 = ParameterEstimator.StateData('vdp/x1');
exp1 = ParameterEstimator.TransientExperiment...
(gcs, [], out1, ic1);
Experimental (Transient) data set for the model 'vdp':
Outputs
(1) vdp/Out1
Inputs
(none)
Initial States
(1) vdp/x1
```

Modifying Properties

The objects referred in `InputData`, `OutputData`, and `InitialStates` properties can be modified or removed as necessary.

Using Class Methods

The description of one important method is given next:

`update` — Updates the object after the Simulink model has been modified. The object listed in the `InputData`, `OutputData`, and `InitialStates` properties are updated in turn.

Creating Parameter Objects

In this section...

“What is a Parameter Object” on page 9-26

“Constructor” on page 9-26

“Properties of Parameter Objects” on page 9-26

“Example: F14 Model” on page 9-28

“Example: Gain Matrix” on page 9-29

“Modifying Properties” on page 9-29

“Using Class Methods” on page 9-29

What is a Parameter Object

The `@Parameter` object refers to the parameters of the Simulink model marked for estimation. Some of the Simulink model parameters are to be estimated and storage is required for the initial values, current values, ranges, etc. One `@Parameter` object corresponds to each parameter in the Simulink model to be potentially estimated. These objects represent estimation parameters of any type such as scalars, vectors, and multidimensional arrays.

Constructor

The syntax to create a parameter object is

```
h = ParameterEstimator.Parameter('Name');  
h = ParameterEstimator.Parameter('Name', Value);  
h = ParameterEstimator.Parameter('Name', Value, Minimum,  
    Maximum);
```

In the first case, `Name` is a workspace variable. In the other cases, `Name` does not need to exist in the workspace at the time of object creation. However, it is required at estimation time.

Properties of Parameter Objects

Descriptions of some important properties are given in the following table.

Parameter Object Properties

Property	Description
Name	Parameter name. The parameter can be a multidimensional array of any size.
Dimensions	Dimensions of the value of the parameter. This is the defining property for the size of other properties.
Value	The current or estimated value of the parameter. This is the defining property for size checking and scalar expansions.
Estimated	<p>A Boolean array of the same size as that of <code>Value</code>. Depending on the value of the elements of the <code>Estimated</code> property, the behavior of the corresponding elements of <code>Value</code> is as follows:</p> <ul style="list-style-type: none"> • The elements of <code>Value</code> is estimated if the corresponding elements in <code>Estimated</code> are set to true. The result is stored in the <code>Value</code> property. • The elements of <code>Value</code> are not estimated if the corresponding elements in <code>Estimated</code> are set to false. However, these elements are used to reset the corresponding workspace parameter during estimations. <p>This property is set to false by default, meaning that the parameter value is not estimated.</p>

Parameter Object Properties (Continued)

Property	Description
InitialGuess	<p>Separate properties are required to hold the initial and current values of the parameters. So, when the InitialGuess property is initialized with a value, both it and the Value property are assigned the same value. Depending on the value of the elements of the Estimated property, the behavior of the corresponding elements of InitialGuess is as follows:</p> <ul style="list-style-type: none"> • If any element in Estimated is set to true, then the corresponding element of InitialGuess is used to initialize the workspace parameter during estimations. • If any element in Estimated is set to false, then the corresponding element of InitialGuess is not used in any way.
Minimum, Maximum	Parameter range.
TypicalValue	The typical values of the parameters. This property is used in estimations for scaling purposes. The default value is 1.

Example: F14 Model

To create a parameter object for the parameter Ta in the f14 model, use

```
par1 = ParameterEstimator.Parameter('Ta')
(1) Parameter data for 'Ta':
Parameter value : 0.05
Initial value : 0.05
Estimated : false
Referenced by the blocks:
f14/Actuator Model
```

Example: Gain Matrix

To create a parameter object for a matrix parameter K of size 4-by-1, use

```
par1 = ParameterEstimator.Parameter('K', [1 2 3 4]')
(1) Parameter data for 'K':
Parameter value : [1;2;3;4]
Initial value   : [1;2;3;4]
Estimated elements : [false;false;false;false]
Referenced by the blocks:
```

Modifying Properties

After a parameter object is created, you can modify its properties using this syntax:

```
par1.Estimated = true; % Estimate this parameter
```

Most of the properties, for example, `Estimated` and `TypicalValue` support scalar expansion with respect to the size of `Value`.

Using Class Methods

Descriptions of two important methods are given next:

- `hiliteBlock` — Highlights the referenced blocks associated with parameter objects in the Simulink diagram.
- `update` — Updates the parameter object after the Simulink model has been modified. If the size of the `Value` property changes, then the other properties are reset to their default values.

Creating State Objects

In this section...

“What is a State Object” on page 9-30

“Constructor” on page 9-30

“Properties of State Objects” on page 9-30

“Example: F14 Model” on page 9-32

“Modifying Properties” on page 9-33

“Using Class Methods” on page 9-33

What is a State Object

The @State object is similar to the @Parameter object. One @State object corresponds to each Simulink block with states in the model.

Constructor

The syntax to create a state object is

```
h = ParameterEstimator.State('block');  
h = ParameterEstimator.State('block', Value);  
h = ParameterEstimator.State('block', Value, Minimum,  
    Maximum);
```

In the first case, the state vector is initialized from the model containing the block. In the other cases, block does not need to exist in the workspace at the time of object creation. However, it is required at estimation time.

Properties of State Objects

Descriptions of some important properties of state objects are given in the following table.

State Object Properties

Property	Description
Block	Name of the Simulink block whose states are defined by this object.
Dimensions	Scalar value to store the number of states of the relevant block.
Value	Column vector to store the value of the state for the block specified by this object. The length of this vector should be consistent with the <code>Dimensions</code> property.
Estimated	<p>A Boolean array of the same size as that of <code>Value</code>. Depending on the value of the elements of the <code>Estimated</code> property, the behavior of the corresponding elements of <code>Value</code> is as follows:</p> <ul style="list-style-type: none">• The elements of <code>Value</code> are estimated if the corresponding elements in <code>Estimated</code> are set to true. The result is stored in the <code>Value</code> property.• The elements of <code>Value</code> are not estimated if the corresponding elements in <code>Estimated</code> are set to false. However, these elements are used to reset the corresponding states during estimations. <p>This property is set to false by default, meaning that the state value is not estimated.</p>

State Object Properties (Continued)

Property	Description
InitialGuess	<p>Separate properties are required to hold the initial and current values of the states. So, when the InitialGuess property is initialized with a value, both it and the Value property are assigned the same value. Depending on the value of the elements of the Estimated property, the behavior of the corresponding elements of InitialGuess is as follows:</p> <ul style="list-style-type: none"> • If any element in Estimated is set to true, then the corresponding element of InitialGuess is used to initialize the state during estimations. • If any element in Estimated is set to false, then the corresponding element of InitialGuess is not used in any way.
Minimum, Maximum	State vector range.
TypicalValue	The typical values of the states. This property is used in estimations for scaling purposes. The default value is 1.
Ts	Sampling time of discrete blocks. Set to zero for continuous blocks. This property is read-only and is currently used for information only.
Domain	String to hold the physical domain of the block. Used for SimMechanics™ or SimPowerSystems™ blocks with states.

Example: F14 Model

To create a state object for the f14/Actuator Model block in the f14 model, use

```
st1 = ParameterEstimator.State(gcb)
```

This command returns the following result:

(1) State data for f14/Actuator Model block:

The block has 1 continuous state(s).

```
State value : 0
Initial guess : 0
Estimated : false
```

Modifying Properties

After a state object is created, you can modify its properties using this syntax:

```
ic1.Estimated = true; % Estimate this state
```

Most of the properties, for example, `Estimated` and `TypicalValue`, support scalar expansion with respect to the size of `Value`.

Using Class Methods

Description of two important methods are given next:

- `hiliteBlock` — Highlights the referenced blocks associated with state objects in the Simulink diagram.
- `update` — Updates the state object after the Simulink model has been modified. If the size of `Value` property changes, then the other properties are reset to their default values.

Creating Estimation Objects

In this section...

“What is an Estimation Object” on page 9-34

“Constructor” on page 9-34

“Properties of Estimation Objects” on page 9-34

“Example: F14 Model” on page 9-36

“Modifying Properties” on page 9-36

“Using Class Methods” on page 9-36

What is an Estimation Object

The @Estimation object is the coordinator between the model, experiments, and parameter objects.

Constructor

The @Estimation object is the coordinator of the model, experiment, and parameter objects. The syntax to create an estimation object is

```
h = ParameterEstimator.Estimation('model');
h = ParameterEstimator.Estimation('model', hParam);
h = ParameterEstimator.Estimation('model', hParam, hExps);
```

Properties of Estimation Objects

Descriptions of some important properties of estimation objects are given in the following table.

Estimation Object Properties

Property	Description
Model	Name of the Simulink model with which this estimation is associated.

Estimation Object Properties (Continued)

Property	Description
Experiments	Experiments to be used in estimations. For multiple experiments, the cost function uses a concatenation of the output error vectors obtained using each experimental data set.
Parameters	Parameter objects to be used in estimations.
States	State objects to be used in estimations. This is a handle matrix with as many columns as there are experiments, and as many rows as there are states in <code>Model</code> . The handle matrix is created automatically in the constructor. You can reorganize its rows to specify shared states between experiments, and set the <code>Estimated</code> flag of desired states. If state data is provided in an experiment, the state objects stored in the columns of this matrix are initialized from the experiments.
SimOptions	Same as <code>simset</code> structure. This property is initialized to <code>simget(this.Model)</code> .
OptimOptions	Same as <code>optimset</code> structure.
EstimInfo	This property is used to store estimation-related information at each iteration of the optimizer, and is initialized as <pre>this.EstimInfo = struct('Cost', [],... 'Covariance', [],... 'FCount', [],... 'FirstOrd', [],... 'Gradient', [],... 'Iteration', [],... 'Procedure', [],... 'StepSize', [],... 'Values', []);</pre>

Example: F14 Model

To create an estimation object for the f14 model to estimate the parameters Ta and Kf and two states, use

```
exp1 = ParameterEstimator.TransientExperiment(gcs);
par1 = ParameterEstimator.Parameter('Ta', 'Estimated', true);
par2 = ParameterEstimator.Parameter('Kf', 'Estimated', true);
est1 = ParameterEstimator.Estimation(gcs, [par1, par2], exp1);
est1.States(1,1).Estimated = true;
est1.States(6,1).Estimated = true;
est1
```

This command returns the following result:

```
Estimated variables for the model 'f14':
```

```
Estimated Parameters
```

```
Using Experiments
```

```
(1) f14 experiment
```

```
Estimated States for Experiment 'f14 experiment'
```

```
(1) f14/Actuator Model
```

```
(6) f14/Controller/Proportional plus integral compensator
```

Modifying Properties

After an estimation object is created, you can modify its properties using this syntax:

```
est.OptimOptions.Algorithm = 'fmincon'; % Estimation method
est.OptimOptions.Display = 'iter'; % Show estimation information
...in workspace
est.Parameters(1).Estimated = false; % Do not estimate first
...parameter
est.States(2,3).Estimated = false; % Do not estimate second state
...of third expression
```

Using Class Methods

Descriptions of some of the important methods are given next:

- `compare` — Compares an experiment and a simulation.
- `simulate` — Simulates the model with current parameters and states.
- `estimate` — Runs an estimation.
- `restart` — Restarts an estimation after it has finished running.
- `update` — Updates the estimation object after the Simulink model has been modified.

Block Reference

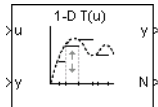
Adaptive Lookup Table (1D Stair-Fit)

Adaptive Lookup Table (2D Stair-Fit)

Adaptive Lookup Table (nD Stair-Fit)

Adaptive Lookup Table (1D Stair-Fit)

Purpose Perform one-dimensional adaptive table lookup



Description

The Adaptive Lookup Table (1D Stair-Fit) block creates a one-dimensional adaptive lookup table by dynamically updating the underlying lookup table. The block uses the outputs, *ydata*, of your system to do the adaptations.

Each indexing parameter *U* may take a value within a set of adapting data points, which are called *breakpoints*. Two breakpoints in each dimension define a *cell*. The set of all breakpoints in one of the dimensions defines a *grid*. In the one-dimensional case, each cell has two breakpoints, and the cell is a line segment.

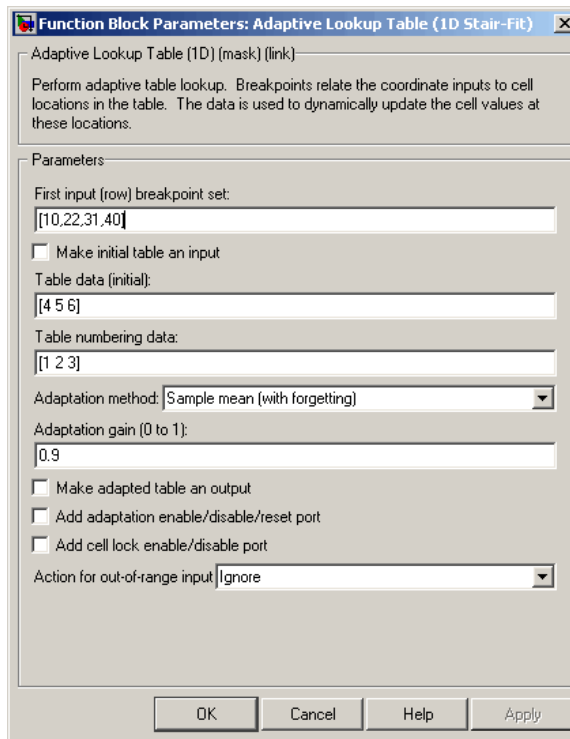
You can use the Adaptive Lookup Table (1D Stair Fit) block to model time-varying systems.

Data Type Support

Doubles only

Adaptive Lookup Table (1D Stair-Fit)

Dialog Box



First input (row) breakpoint set

The vector of values containing possible block input values. The input vector must be monotonically increasing.

Make initial table an input

Selecting this check box forces the Adaptive Lookup Table (1D Stair-Fit) block to ignore the **Table data (initial)** parameter, and creates a new input port **Tin**. Use this port to input the table data.

Table data (initial)

The initial table output values. This vector must be of size $N-1$, where N is the number of breakpoints.

Adaptive Lookup Table (1D Stair-Fit)

Table numbering data

Number values assigned to cells. This vector must be the same size as the table data vector, and each value must be unique.

Adaptation method

Choose `Sample mean` or `Sample mean (with forgetting)`. `Sample mean` averages all the values received within a cell. `Sample mean with forgetting` gives more weight to the new data. How much weight is determined by the **Adaptation gain** parameter. For more information, see “Selecting an Adaptation Method” on page 8-41.

Adaptation gain (0 to 1)

A number between 0 and 1 that regulates the weight given to new data during the adaptation. A 0 means short memory (last data becomes the table value), and 1 means long memory (average all data received in a cell).

Make adapted table an output

Selecting this check box creates an additional output port `Tout` for the adapted table.

Add adaptation enable/disable/reset port

Selecting this check box creates an additional input port `Enable` that enables, disables, or resets the adaptive lookup table. 0 = disable; 1 = enable; 2 = reset to initial table data.

Add cell lock enable/disable port

Selecting this check box creates an additional input port `Lock` that provides the means for updating only specified cells during a simulation run. 0 = unlock; 1 = lock current cell.

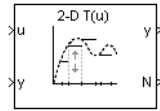
Action for out-of-range input

Ignore or Adapt by extrapolating beyond the extreme breakpoints.

Adaptive Lookup Table (2D Stair-Fit)

Purpose

Perform two-dimensional adaptive table lookup



Description

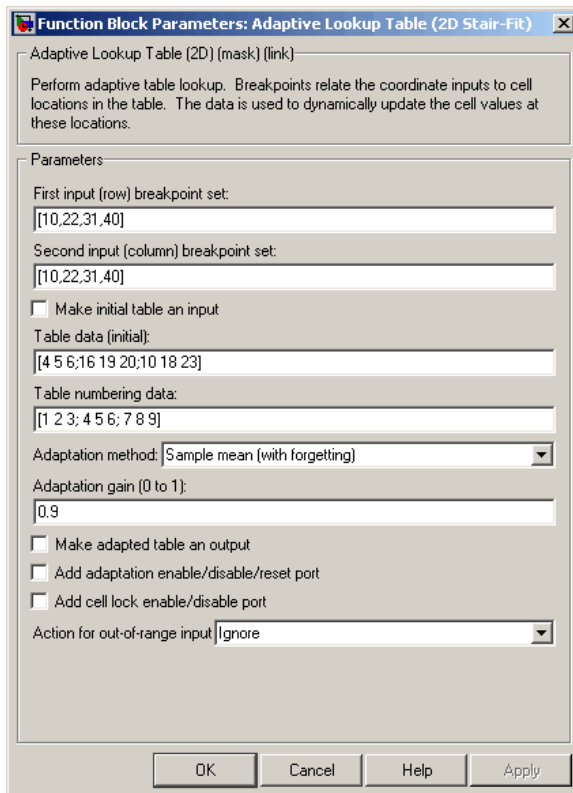
The Adaptive Lookup Table (2D Stair-Fit) block creates a two-dimensional adaptive lookup table by dynamically updating the underlying lookup table. The block uses the outputs (*ydata*) of your system to do the adaptations.

Each indexing parameter *U* may take a value within a set of adapting data points, which are called *breakpoints*. Two breakpoints in each dimension define a *cell*. The set of all breakpoints in one of the dimensions defines a *grid*. In the two-dimensional case, each cell has four breakpoints and is a flat surface.

You can use the Adaptive Lookup Table (2D Stair-Fit) block to model time-varying systems.

Adaptive Lookup Table (2D Stair-Fit)

Dialog Box



First input (row) breakpoint set

The vector of values containing possible block input values for the first input variable. The first input vector must be monotonically increasing.

Second input (column) breakpoint set

The vector of values containing possible block input values for the second input variable. The second input vector must be monotonically increasing.

Adaptive Lookup Table (2D Stair-Fit)

Make initial table an input

Selecting this check box forces the Adaptive Lookup Table (2D Stair-Fit) block to ignore the **Table data (initial)** parameter, and creates a new input port T_{in} . Use this port to input the table data.

Table data (initial)

The initial table output values. This 2-by-2 matrix must be of size $(n-1)$ -by- $(m-1)$, where n is the number of first input breakpoints and m is the number of second input breakpoints.

Table numbering data

Number values assigned to cells. This matrix must be the same size as the table data matrix, and each value must be unique.

Adaptation method

Choose **Sample mean** or **Sample mean with forgetting**. **Sample mean** averages all the values received within a cell. **Sample mean with forgetting** gives more weight to the new data. How much weight is determined by the **Adaptation gain** parameter. For more information, see “Selecting an Adaptation Method” on page 8-41.

Adaptation gain (0 to 1)

A number from 0 to 1 that regulates the weight given to new data during the adaptation. A 0 means short memory (last data becomes the table value), and 1 means long memory (average all data received in a cell).

Make adapted table an output

Selecting this check box creates an additional output port T_{out} for the adapted table.

Add adaptation enable/disable/reset port

Selecting this check box creates an additional input port **Enable** that enables, disables, or resets the adaptive lookup table.

Add cell lock enable/disable port

Selecting this check box creates an additional input port **Lock** that provides the means for updating only specified cells during a simulation run.

Adaptive Lookup Table (2D Stair-Fit)

Action for out-of-range input

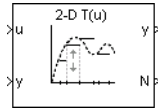
Ignore or Adapt by extrapolating beyond the extreme breakpoints.

Adaptive Lookup Table (nD Stair-Fit)

Purpose

Create adaptive lookup table of arbitrary dimension

Description



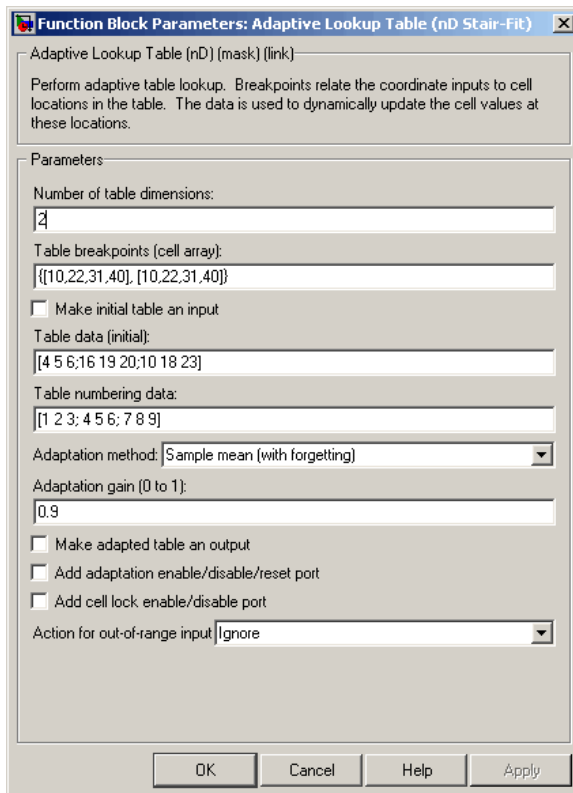
The Adaptive Lookup Table (nD Stair-Fit) block creates an adaptive lookup table of arbitrary dimension by dynamically updating the underlying lookup table. The block uses the outputs of your system to do the adaptations.

Each indexing parameter may take a value within a set of adapting data points, which are called *breakpoints*. Breakpoints in each dimension define a *cell*. The set of all breakpoints in one of the dimensions defines a *grid*. In the n-dimensional case, each cell has two n breakpoints and is an (n-1) hypersurface.

You can use the Adaptive Lookup Table (nD Stair-Fit) block to model time-varying systems.

Adaptive Lookup Table (nD Stair-Fit)

Dialog Box



Number of table dimensions

The number of dimensions for the adaptive lookup table.

Table breakpoints (cell array)

A set of one-dimensional vectors that contains possible block input values for the input variables. Each input row must be monotonically increasing, but the rows do not have to be the same length. For example, if the **Number of table dimensions** is 3, you can set the table breakpoints as follows:

$$\{[1 \ 2 \ 3], [5 \ 7], [1 \ 3 \ 5 \ 7]\}$$

Adaptive Lookup Table (nD Stair-Fit)

Make initial table an input

Selecting this check box forces the Adaptive Lookup Table (nD Stair-Fit) block to ignore the **Table data (initial)** parameter, and creates a new input port T_{in} . Use this port to input the table data.

Table data (initial)

The initial table output values. This (n-D) array must be of size (n-1)-by-(n-1) ... -by- (n-1), (D times), where D is the number of dimensions and n is the number of input breakpoints.

Table numbering data

Number values assigned to cells. This vector must be the same size as the table data array, and each value must be unique.

Adaptation method

Choose **Sample mean** or **Sample mean with forgetting**. **Sample mean** averages all the values received within a cell. **Sample mean with forgetting** gives more weight to the new data. How much weight is determined by the **Adaptation gain** parameter. For more information, see “Selecting an Adaptation Method” on page 8-41.

Adaptation gain (0 to 1)

A number from 0 to 1 that regulates the weight given to new data during the adaptation. A 0 means short memory (last data becomes the table value), and 1 means long memory (average all data received in a cell).

Make adapted table an output

Selecting this check box creates an additional output port T_{out} for the adapted table.

Note The Adaptive Lookup Table (n-D Stair Fit) block cannot output a table of 3 or more dimensions.

Adaptive Lookup Table (nD Stair-Fit)

Add adaptation enable/disable/reset port

Selecting this check box creates an additional input port **Enable** that enables, disables, or resets the adaptive lookup table.

Add cell lock enable/disable port

Selecting this check box creates an additional input port **Lock** that provides the means for updating only specified cells during a simulation run.

Action for out-of-range input

Ignore or Adapt by extrapolating beyond the extreme breakpoints.

Function Reference

spetool

Purpose Open Simulink Parameter Estimation GUI

Syntax `spetool('modelname')`

Description `spetool('modelname')` opens Simulink Parameter Estimation GUI for the Simulink model with the name `modelname`.

See Also For more information on estimating parameters using Simulink Parameter Estimation GUI, see Chapter 1, “Getting Started”.

Examples

Use this list to find examples in the documentation.

Getting Started

Chapter 2, “Tutorial — Preparing Data for Parameter Estimation Using the GUI”

Chapter 3, “Tutorial — Estimating Parameters from Measured Data Using the GUI”

Chapter 4, “Tutorial — Modeling a System Using Adaptive Lookup Table”

Estimating Parameters and Initial Conditions

“Example: Estimating Initial Conditions of a Mass-Spring-Damper System” on page 5-4

“Example: Estimating Parameters and Initial Conditions of the F14 Model” on page 9-4

Estimating Lookup Table Values

“Example — Estimating Lookup Table Values from Data” on page 8-6

“Example — Estimating Constrained Values of a Lookup Table” on page 8-20

A

- acceleration 1-56
- accelerator mode 1-56
- adaptation gain 8-42
- adaptive lookup tables 8-2
- adding data sets 1-23

C

- command-line estimation 9-2
- Controls and Estimation Tools Manager 1-9

D

- data
 - detrending 6-15
 - exclusion 6-6
 - filtering 6-15
 - preprocessing 6-3
- Data Import dialog box 1-15
- data sets
 - adding 1-23
- detrending data 6-15
- different time lengths 1-52
- display options for estimation 1-27

E

- estimation
 - display options 1-27
 - example of command-line estimation 9-4
 - from the command line 9-2
 - running 1-33
 - selecting parameters 1-17
 - selecting states 1-19
 - setting up a project 1-23
- excluding data 6-6

F

- filtering data 6-15

I

- importing
 - initial conditions 1-16
 - transient data 1-11
- initial conditions
 - example of estimating 5-4
 - importing 1-16
- initial guesses 1-20

L

- lookup tables
 - adaptive 8-2

M

- multiple projects and tasks 7-2

O

- optimization
 - setting options for 1-46

P

- parameter estimation 1-56
- parameters
 - selecting for estimation 1-17
 - specification of 1-25
- preprocessing data 6-3
- projects
 - definition of 1-5
 - saving 7-3

R

- running an estimation 1-33

S

- Sample mean 8-42

- Sample mean with forgetting 8-42
- saving projects 7-3
- selecting views 1-29
- setting options
 - for optimization 1-46
 - for simulation 1-46
 - upper/lower bounds 1-20
- simulation
 - setting options for 1-46
- simulation with 1-51
- specifying parameters 1-25
- states
 - selecting for estimation 1-19

T

- transient data
 - importing 1-11

U

- upper/lower bounds
 - setting 1-20

V

- views
 - selecting 1-29