

Simulink[®] Response Optimization

For Use with Simulink[®]

- Modeling
- Simulation
- Implementation

User's Guide

Version 2



How to Contact The MathWorks:



www.mathworks.com Web
comp.soft-sys.matlab Newsgroup



support@mathworks.com Technical Support
suggest@mathworks.com Product enhancement suggestions
bugs@mathworks.com Bug reports
doc@mathworks.com Documentation error reports
service@mathworks.com Order status, license renewals, passcodes
info@mathworks.com Sales, pricing, and general information



508-647-7000 Phone



508-647-7001 Fax



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

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

Simulink Response Optimization User's Guide

© COPYRIGHT 2004–2005 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, Simulink, Stateflow, Handle Graphics, Real-Time Workshop, and xPC TargetBox are registered trademarks of The MathWorks, Inc.

Other product or brand names are 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	Online only	New for Version 2.0 (Release 14) (Renamed from <i>Nonlinear Control Design Blockset User's Guide</i>)
October 2004	Online only	Revised for Version 2.1 (Release 14SP1)
March 2005	Online only	Revised for Version 2.2 (Release 14SP2)
September 2005	Online only	Revised for Version 2.3 (Release 14SP3)

Approaching Response Optimization

1

Choosing Signals to Constrain	1-2
Attaching Signal Constraint Blocks	1-2
Creating a Response Optimization Project	1-3
Saving and Reloading Response Optimization Projects	1-4
Saving Response Optimization Projects	1-4
Saving Additional Settings	1-6
Reloading Response Optimization Projects	1-6

Specifying the Desired Response

2

Specifying Signal Bounds	2-2
Moving Constraints	2-2
Including Gridlines on the Axes	2-3
Positioning Constraints Exactly	2-3
Adjusting Constraint Weightings	2-4
Edit Constraint Dialog	2-5
Scaling Constraints	2-8
Splitting and Joining Constraints	2-9
Choosing Step Response Specifications	2-10
Tracking Reference Signals	2-13
Specifying the Reference Signal	2-13
Plotting Responses in the Signal Constraint Window ..	2-14
Reference Signals	2-14
Current Response	2-14

Initial Response	2-14
Intermediate Steps	2-14
Response Plots Property Editor	2-15
Labels Pane	2-15
Limits Pane	2-16

Choosing Optimized Parameters

3

Specifying Tuned Parameters in the Model	3-2
Adding Tuned Parameters	3-2
Changing Tuned Parameter Specifications	3-3
Including Uncertainty in Parameter Values	3-5
Adding Uncertain Parameters	3-6
Changing Uncertain Parameter Specifications	3-7
Including Independent Parameters	3-9
Example:	3-9

Running the Optimization

4

Running the Optimization	4-2
Tuning the Optimization Results	4-4
Selecting Optimization Methods	4-4
Selecting Optimization Termination Options	4-5
Selecting Additional Optimization Options	4-6
Setting Options for the Simulation	4-8
Selecting Simulation Time	4-8
Selecting Solvers	4-9

Accelerating the Optimization 4-12

Functions — Categorical List

5

Response Optimization Projects 5-2

Constraints and Parameters 5-2

Optimization and Simulation Settings 5-2

Functions — Alphabetical List

6

Blocks — Alphabetical List

7

Index

Approaching Response Optimization

This chapter outlines preliminary steps for approaching and setting up a Simulink® Response Optimization project with the graphical user interface (GUI).

Choosing Signals to Constrain
(p. 1-2)

Inserting Signal Constraint blocks

Creating a Response Optimization Project (p. 1-3)

Components of a response optimization project and how to create one

Saving and Reloading Response Optimization Projects (p. 1-4)

Using the save and load commands

Choosing Signals to Constrain

Simulink® Response Optimization works by adjusting parameters in a Simulink model so that chosen response signals within the system behave in a specified way. You choose the signals that you want to shape or constrain by attaching Signal Constraint blocks to them. The constraints on the behavior of the response signals and the tuned parameters are set within the Signal Constraint blocks.

The first step in the response optimization process is to choose which signals in your Simulink model you would like to constrain and to attach Signal Constraint blocks to these signals.

Attaching Signal Constraint Blocks

Once you have selected signals to constrain, you need to attach a Signal Constraint block to each of these signals. You can find the Signal Constraint block under Simulink Response Optimization within the Simulink Library Browser. Alternatively, you can open the Simulink Response Optimization library by typing `srolib` at the MATLAB® prompt.

To attach a Signal Constraint block to a signal in your model, drag the block from the block library into the model and join the signal line to the input of the Signal Constraint block. A model can include multiple Signal Constraint blocks and you can attach the Signal Constraint block to any signal, including signals within subsystems of your model.

Note The Signal Constraint block is not an output block of the system and will not interfere with a linearization of your model (as opposed to blocks in the Nonlinear Control Design Blockset, the previous name for this product, which were output blocks).

Creating a Response Optimization Project

Double-click on a Signal Constraint block to open the Signal Constraint window associated with it. Within this window you can specify the constraints imposed on the signal, along with other information needed for the response optimization project.

Although you must specify the constraints for each signal individually within each Signal Constraint block, you only need to set the remaining settings such as tuned parameters and optimization settings within one Signal Constraint window as they apply to the whole project.

Opening a Signal Constraint window, automatically creates a response optimization project. The project consists of the following information:

- Constraints on all signals that have Signal Constraint blocks attached
- Tuned parameters in the system and specifications for these parameters such as initial guesses and maximum and minimum values
- Uncertain parameters in the system and specifications for these parameters
- Optimization and simulation setup options

A response optimization project exists within a single model; there are no cross-model projects. Additionally, although you can create different sets of constraints and tuned parameters and save these as different response optimization projects, you can only associate one project with the model at any time.

The remaining steps involved in specifying the settings of a response optimization project are discussed in the following sections:

- Chapter 2, “Specifying the Desired Response”
- Chapter 3, “Choosing Optimized Parameters”

To save the project for use in a later session, see “Saving and Reloading Response Optimization Projects” on page 1-4.

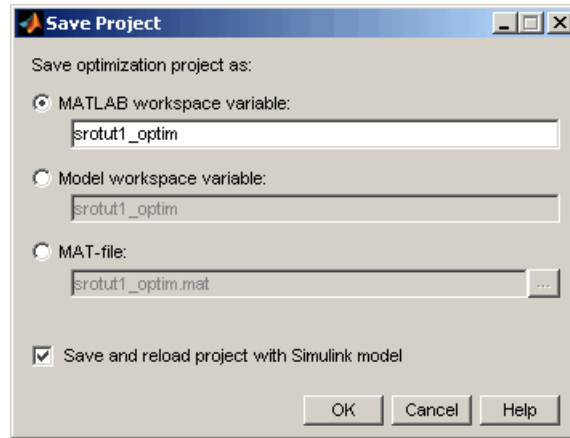
Saving and Reloading Response Optimization Projects

Saving a response optimization project allows you to reuse your settings during a later session. These settings include constraint bounds, tuned and uncertain parameters, and settings for optimization and simulation. Additional settings such as the position of the Signal Constraint window, axis limit settings, and the name and location of the project are saved with the Simulink *model*.

Saving Response Optimization Projects

To save the constraints and data of the response optimization project to a workspace variable or a file, select **File > Save** from a Signal Constraint window in the model. Within the Save Project dialog (shown below), you can save the project as a

- **MATLAB workspace variable:** Enter the name of a MATLAB workspace variable, and then click **OK** to save the project. This is obviously a temporary solution as the project will no longer exist once you terminate your MATLAB session.
- **Model workspace variable:** Enter the name of a model workspace variable, and then click **OK** to save the project. This method is convenient as the project is stored with the model and you do not need to worry about keeping a separate file or variable available.
- **MAT-file:** Enter a filename, and then click **OK** to save the project. Alternatively, you can save the project to an existing file by clicking the button to the right of **MAT-file** and selecting a file from the directory. Saving the project as a MAT-file is convenient when you want to save multiple projects for a single model.



To automatically reload the project when reopening the Simulink model, select the **Save and reload project with Simulink model** check box at the bottom of the window. The Simulink model stores the location and name of the project. Hence, when you change the location or filename for the project, you must also resave the Simulink model.

When reopening a model in which a response optimization project has been previously saved and the **Save and reload project with Simulink model** check box was selected, the model will search for the variable or file containing the project and automatically load it into the model. If the file cannot be found, a warning dialog will appear.

Although the save command is issued from a single Signal Constraint window, the constraints and data from *all* Signal Constraint windows are saved as a single project that includes signal constraints, tuned parameters, uncertain parameters, and setup options.

To save the constraints and data of the response optimization project under a new name, select **File > Save As** from a Signal Constraint window in the model, and then follow the instructions above.

Saving Additional Settings

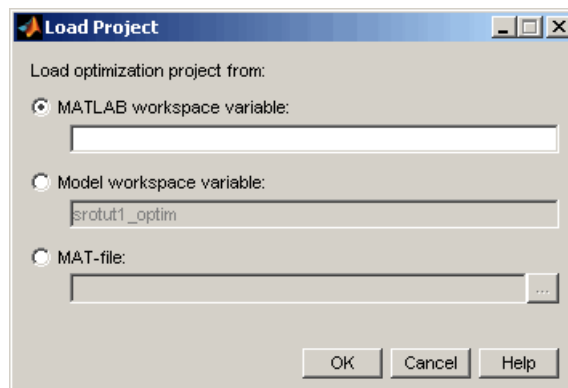
In addition to settings that you save with the response optimization project, there are several other settings that you save with the *model*. These settings include the following:

- The position of the Signal Constraint window on the screen
- The axis limit settings within the Signal Constraint window
- The location where the response optimization project, either a workspace variable or a MAT-file, is saved
- The name of the response optimization project

When you modify one or more of these attributes, you must resave the Simulink model to retain the settings when reloading the model in a subsequent session. To save the Simulink model, select **File > Save** within the model window.

Reloading Response Optimization Projects

To reload a response optimization project from the MATLAB workspace, model workspace, or a file, select **File > Load** from a Signal Constraint window in the model. In the Load Project dialog (shown below), enter the name of the MATLAB workspace variable, model workspace variable, or MAT-file that contains the project, and then click **OK**. Alternatively, you can load the project from an existing file by clicking the button to the right of **MAT-file** and selecting a file from the directory.



Although the load command is issued from a single Signal Constraint window, the constraints are loaded into *all* Signal Constraint blocks in the model. Additionally, tuned parameters, uncertain parameters, and optimization and simulation setup options are loaded into the model.

Note Loading a project cannot be undone.

Specifying the Desired Response

With Simulink Response Optimization you can specify the desired response of a signal by enforcing signal bounds or by tracking a reference signal. To enforce signal bounds, select this option at the bottom of the Signal Constraint window, and then position time-domain-based constraint bound segments in the Signal Constraint window. To track a reference signal, select this option at the bottom of the Signal Constraint window, and then plot the signal in the Signal Constraint window. This chapter provides further details on both methods as well as instructions for editing the figure axes and plotting additional responses.

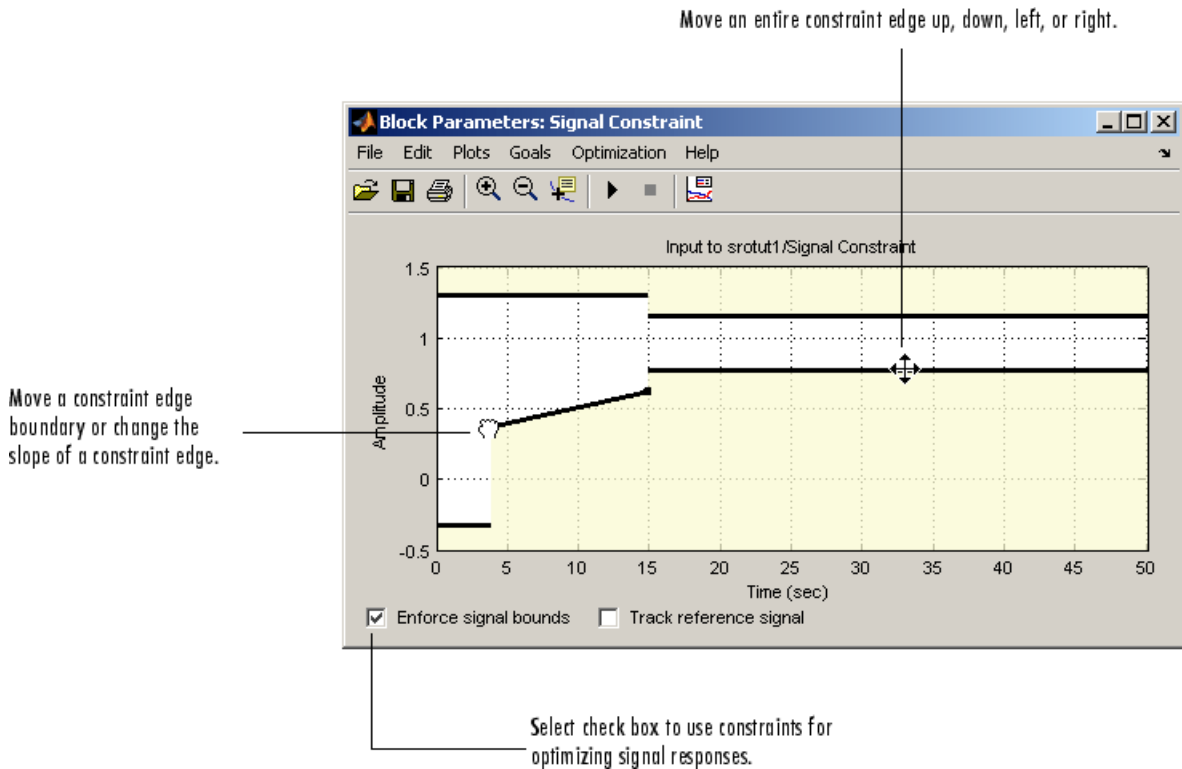
Specifying Signal Bounds (p. 2-2)	Enforce signal bounds by positioning and editing signal constraints
Tracking Reference Signals (p. 2-13)	Specify the desired signal response by plotting and tracking a reference signal
Plotting Responses in the Signal Constraint Window (p. 2-14)	Choose which types of response signals to plot
Response Plots Property Editor (p. 2-15)	Change axes limits and labels

Specifying Signal Bounds

To specify the desired response signal using time-domain-based constraints, you must first select the **Enforce signal bounds** option at the bottom of the Signal Constraint window. Then, constrain the response signal by positioning the constraint bound segments within the figure axes using the following techniques.

Moving Constraints

Constraint-bound segments define the time-domain constraints you would like to place on a particular signal in your model. To position these segments, which appear as a yellow shaded region bordered by a black line, use the mouse to click and drag segments within the Signal Constraint window as shown in the following figure.



- To move a constraint edge boundary or to change the slope of a constraint edge, position the pointer over a constraint edge endpoint, and press and hold down the left mouse button. The pointer should change to a hand symbol. While still holding the button down, drag the pointer to the target location, and release the mouse button. Note that the segments on either side of the boundary might not maintain their slopes.
- To move an entire constraint edge up, down, left, or right, position the mouse pointer over the segment and press and hold down the left mouse button. The pointer should change to a four-way arrow. While still holding the button down, drag the pointer to the target location, and release the mouse button. Note that the segments on either side of the boundary might not maintain their slopes.

Tip To move a constraint edge to a perfectly horizontal or vertical position, hold down the **Shift** key while clicking and dragging the constraint edge. This will cause the constraint edge to *snap* to a horizontal or vertical position.

To use these constraints to optimize signal responses, make sure that the **Enforce signal** bounds check box is selected at the bottom of the window.

Note It is possible to move a lower bound constraint edge above an upper bound constraint edge, or vice versa, but this will produce an error when you attempt to run the optimization.

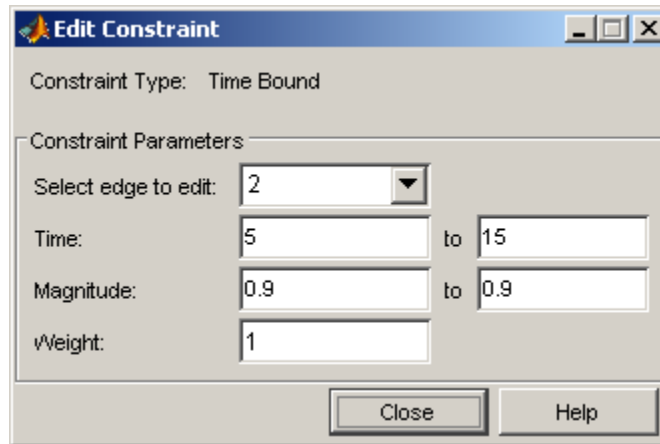
Including Gridlines on the Axes

When moving constraint bound segments in the Signal Constraint window, it is sometimes helpful to display gridlines on the axes for careful alignment of the constraint bound segments. To turn the gridlines on or off, right-click within the axes of the Signal Constraint window and select **Grid**.

Positioning Constraints Exactly

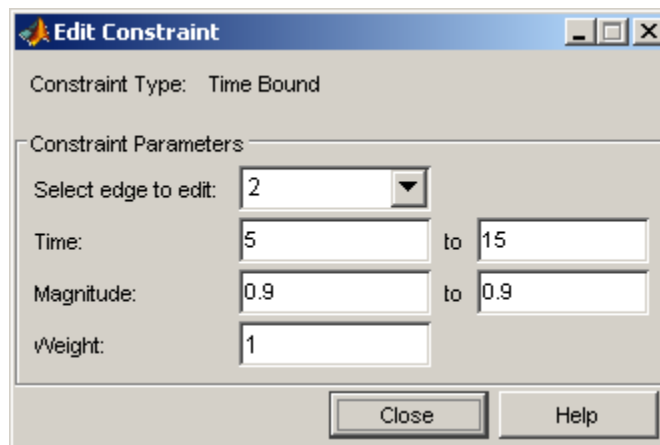
To position a constraint edge exactly, position the pointer over the segment you want to move and press the right mouse button. Select **Edit** from the

menu to open the Edit Constraint dialog, shown below. For information on using the Edit Constraint dialog, see “Edit Constraint Dialog” on page 2-5.



Adjusting Constraint Weightings

To change the weight of a constraint edge, position the pointer over the segment you want to weight and click the right mouse button. Select **Edit** from the menu to open the Edit Constraint dialog, shown below. For information on using the Edit Constraint dialog, see “Edit Constraint Dialog” on page 2-5.

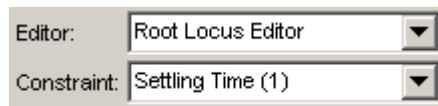


Edit Constraint Dialog

The Edit Constraint dialog allows you to exactly position constraint segments and to edit other properties of these constraints. The dialog has two main components:

- An upper panel to specify the constraint you are editing
- A lower panel to edit the constraint parameters

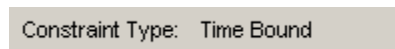
When used with the SISO Tool in the Control System Toolbox, the upper panel of the Edit Constraint dialog resembles the image below. **Editor** refers to the particular editor within the SISO Tool that contains the constraint and **Constraint** refers to a particular constraint within that editor. To edit other constraints within the SISO Tool, change either the **Editor** or **Constraint** field within this dialog.



Editor: Root Locus Editor

Constraint: Settling Time (1)

When used with Simulink Response Optimization, the upper panel of the Edit Constraint dialog resembles the image below. Simulink Response Optimization constraints are always **Time Bound** constraints. The **Constraint Type** field of this dialog is not editable.



Constraint Type: Time Bound

Constraint Parameters

The particular constraint parameters shown within the lower panel of the Edit Constraint(s) dialog depend on the type of constraint. The table below summarizes the various constraint parameters.

Edit Constraint Dialog Constraint Parameters

Constraint Parameter	Found in	Description
Select edge to edit	Simulink Response Optimization, SISO Tool Open-Loop Bode Editor, Prefilter Bode Editor, SISO Tool Open-Loop Nichols Editor, SISO Tool Root Locus Editor	When a constraint contains more than one segment, also known as an edge, you can use this menu to choose the edge you want to edit. Edges are numbered from left to right starting at 1.
Time	Simulink Response Optimization	Defines the time range of an edge within a constraint.
Magnitude	Simulink Response Optimization, SISO Tool Open-Loop Bode Editor, Prefilter Bode Editor	Defines beginning and ending amplitude of a constraint edge.
Weight	Simulink Response Optimization	Defines the weight of an edge within a constraint. The weight is a measure of the relative importance of this constraint edge when used in a response optimization project. Weights can vary between 0 and 1 where 0 implies that the constraint edge is disabled and does not have to be satisfied and 1 implies that the constraint edge must be satisfied. The weight of a constraint edge is graphically represented by the thickness of the black constraint line. An invisible constraint edge represents a weight of 0 and a thick constraint edge represents a weight of 1.
Frequency	SISO Tool Open-Loop Bode Editor, Prefilter Bode Editor	Defines the frequency range of an edge within a constraint.

Edit Constraint Dialog Constraint Parameters (Continued)

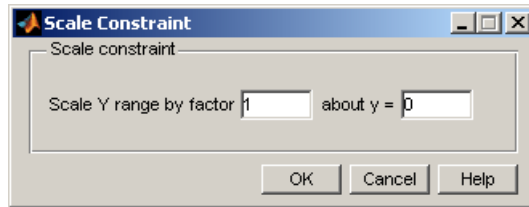
Constraint Parameter	Found in	Description
Slope (dB/decade)	SISO Tool Open-Loop Bode Editor, Prefilter Bode Editor	Defines the slope, in dB/decade, of a constraint edge. It is an alternative method of specifying the magnitude values. Entering a new Slope value changes any previously defined magnitude values.
Settling Time <	SISO Tool Root Locus Editor	Defines a constraint edge for a particular settling time.
Percent overshoot <	SISO Tool Root Locus Editor	Defines constraint edges for a particular percent overshoot.
Damping Ratio >	SISO Tool Root Locus Editor	Defines constraint edges for a particular damping ratio.
Natural Frequency	SISO Tool Root Locus Editor	Defines a constraint edge for a particular natural frequency. To specify the constraint, choose at least or at most from the menu, and then specify the natural frequency of interest.
Real	SISO Tool Root Locus Editor	Defines the beginning and end of the real component of a pole-zero region constraint.
Imaginary	SISO Tool Root Locus Editor	Defines beginning and end of the imaginary component of a pole-zero region constraint.
Phase Margin >	SISO Tool Open-Loop Nichols Editor	Defines a constraint edge for a minimum phase margin. The phase margin specified should be a number greater than 0.

Edit Constraint Dialog Constraint Parameters (Continued)

Constraint Parameter	Found in	Description
Located at	SISO Tool Open-Loop Nichols Editor	Defines the center, in degrees, of the constraint edge defining the phase margin, gain margin, or closed-loop peak gain. The location must be -180 plus a multiple of 360 degrees. If you enter an invalid location point, the closest valid location is selected.
Gain Margin >	SISO Tool Open-Loop Nichols Editor	Defines a constraint edge for a particular gain margin.
Closed-Loop Peak Gain <	SISO Tool Open-Loop Nichols Editor	Defines a constraint edge for a particular closed-loop peak gain. The specified value can be positive or negative in dB. The constraint follows the curves of the Nichols plot grid, so we recommend that you have the grid on when using this feature.
Open loop phase	SISO Tool Open-Loop Nichols Editor	Defines the beginning and end of the open loop phase component of a gain-phase constraint edge.
Open loop gain	SISO Tool Open-Loop Nichols Editor	Defines the beginning and end of the open loop gain component of a gain-phase constraint edge.

Scaling Constraints

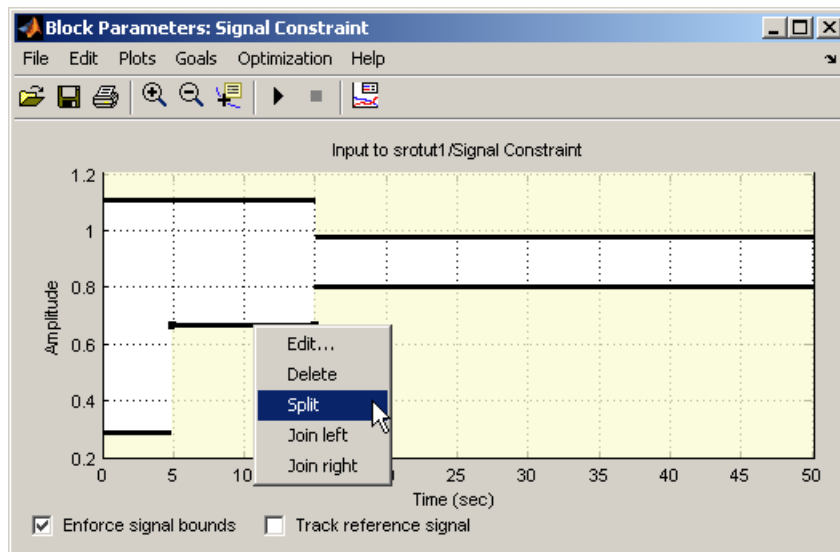
Instead of clicking and dragging the constraints to their new positions, you can scale the constraints. To scale the constraints, select **Edit > Scale Constraint** in the Signal Constraint window. This displays the Scale Constraint dialog.



Enter the amount by which you want the constraints to scale, and the point about which you want to scale them, and then click **OK**.

Splitting and Joining Constraints

To split a constraint edge, position the pointer over the segment to be split, and press the right mouse button. Select **Split** from the context menu. The segment splits in half. You can now manipulate each segment individually.

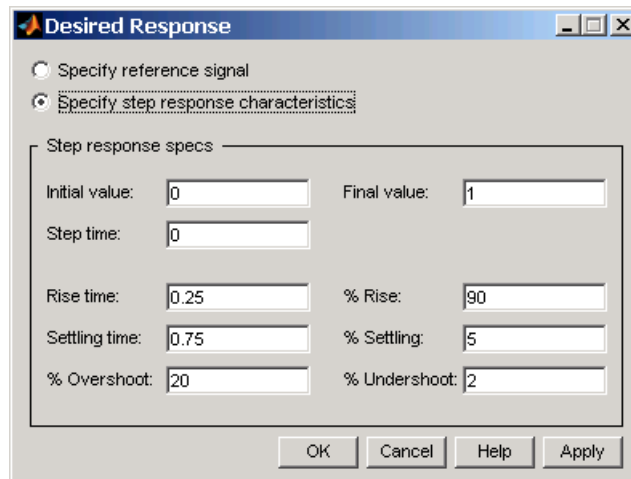


To join two neighboring constraint edges, position the pointer over one constraint segment, and press the right mouse button. Select **Join left** or **Join right** from the menu to join the segment to the left or right respectively.

Choosing Step Response Specifications

When you are optimizing the step response of your system, an alternative method of positioning the constraint bound segments is to specify the desired step response characteristics such as rise time, settling time, and overshoot.

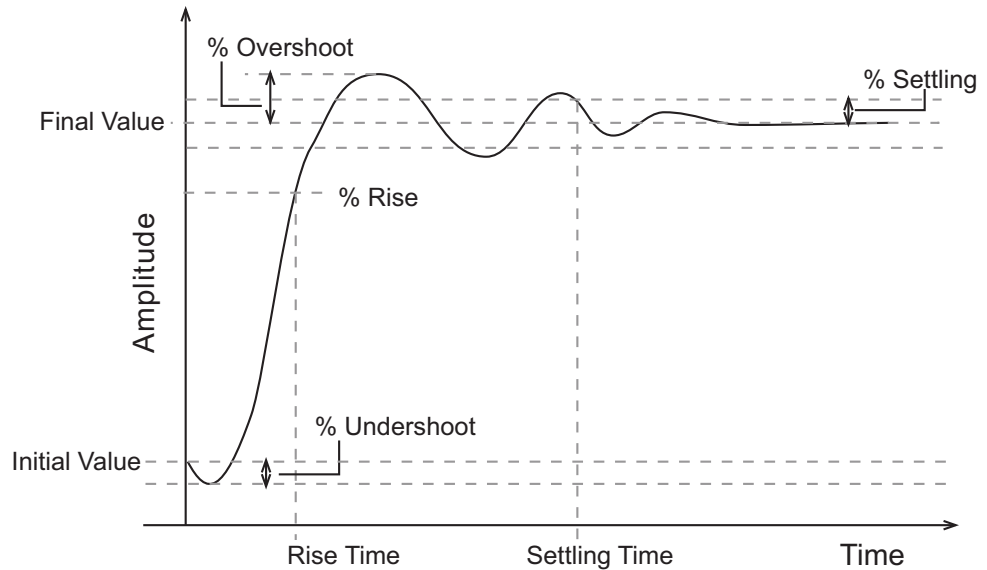
To specify step response characteristics select **Goals > Desired Response** in the Signal Constraint window or right-click in the white space of the figure window and select **Desired Response** from the context menu. This will display the Desired Response dialog. Select **Specify step response characteristics** to display the step response specifications as shown below.



The top three options specify the details of the step input:

- **Initial value:** input level before the step occurs
- **Step time:** time at which the step takes place
- **Final value:** input level after the step occurs

The remaining options specify the characteristics of the response signal. Each of the step response characteristics is described in the figure below.



- **Rise time:** The time taken for the response signal to reach a specified percentage of the step's range. The step's range is the difference between the final and initial values.
- **% Rise:** The percentage used in the rise time.
- **Settling time:** The time taken until the response signal settles within a specified region around the final value. This settling region is defined as the final step value plus or minus the specified percentage of the final value.
- **% Settling:** The percentage used in the settling time.
- **% Overshoot:** The amount by which the response signal can exceed the final value. This amount is specified as a percentage of the step's range. The step's range is the difference between the final and initial values.
- **% Undershoot:** The amount by which the response signal can undershoot the initial value. This amount is specified as a percentage of the step's range. The step's range is the difference between the final and initial values.

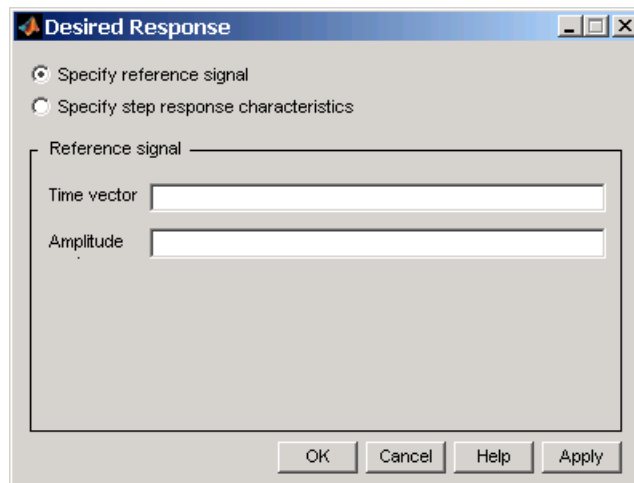
Enter values for the response specifications in the Response Specifications dialog, based on the requirements of your model, and then click **OK**. The constraint edges will now reflect the constraints specified.

Tracking Reference Signals

You can specifying the desired response as an ideal or *reference* trajectory. First select the **Track reference signal** option at the bottom of the Signal Constraint window. Then, plot the reference signal within the figure axes using the following techniques. You can use this reference signal in addition to, or instead of, enforcing signal bounds.

Specifying the Reference Signal

Plot a reference signal by selecting **Goals > Desired Response** in the Signal Constraint window or by right-clicking in the white space of the figure window and selecting **Desired Response** from the context menu. This will display the Desired Response dialog. Select the radio button labeled **Specify reference signal** to display the reference signal setup as shown below.



Define the reference signal by entering vectors, or variables from the workspace, for the time and amplitude of the signal, and then clicking **OK**. To turn the reference signal on or off, right-click in the white space of the figure window and select **Show > Reference Signal**.

Plotting Responses in the Signal Constraint Window

You can choose to plot several different signals in the Signal Constraint window including reference signals, initial response signals, and response signals generated during the optimization.

Reference Signals

To plot a reference signal, use the methods in “Specifying the Reference Signal” on page 2-13.

Current Response

To display the current response signal, based on the current parameter values, right-click within the white space of the Signal Constraint window and select **Plot Current Response**. The current response appears as a thick white line.

Initial Response

To turn the display of the initial response signal on or off, right-click within the white space of the Signal Constraint window and select **Show > Initial Response**. The initial response is the response of the signal based on parameter values in place before the optimization is run. The initial response appears as a blue line.

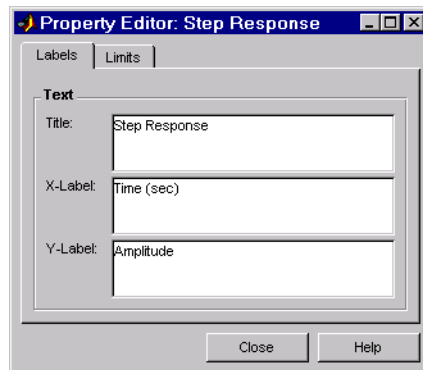
Intermediate Steps

To turn on, or off, the display of the response signal at intermediate steps during the optimization, right-click within the white space of the Signal Constraint window and select **Show > Intermediate Steps**. The response signal at an intermediate step is based on parameter values at an intermediate point in the optimization.

Response Plots Property Editor

Note Click on the tabs to get help on panes in the Property Editor.

This figure shows the Property Editor dialog for a step response.



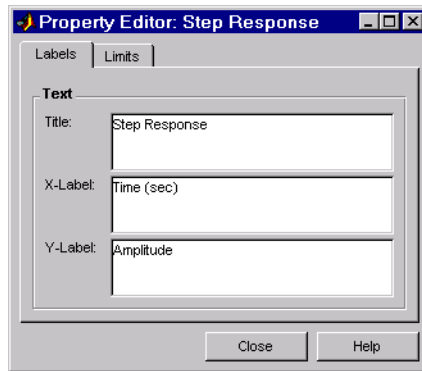
In general, you can change the following properties of response plots.

- **Labels** – Titles and X- and Y-labels
- **Limits** – Numerical ranges of the X and Y axes

As you make changes in the Property Editor, they display immediately in the response plot. Conversely, if you make changes in a plot using right-click menus, the Property Editor for that plot automatically updates. The Property Editor and its associated plot are dynamically linked.

Labels Pane

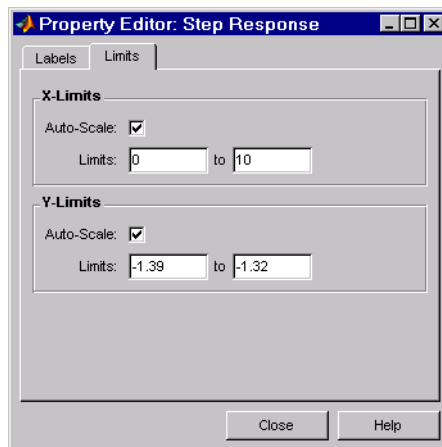
Note Click on the tabs below to get help on the Property Editor.



To specify new text for plot titles and axis labels, type the new string in the field next to the label you want to change. Note that the label changes immediately as you type, so you can see how the new text looks as you are typing.

Limits Pane

Note Click on the tabs to get help on the Property Editor.



Default values for the axes limits make sure that the maximum and minimum x and y values are displayed. If you want to override the default settings,

change the values in the **Limits** pane fields. The **Auto-Scale** check box automatically clears if you click on a different field. The new limits appear immediately in the response plot.

To reestablish the default values, select the **Auto-Scale** check box again.

Choosing Optimized Parameters

Before running the optimization, you need to define which system parameters are tunable. By tuning these parameters, Simulink Response Optimization makes the response signal meet the imposed constraints. In addition, you can define other parameters to account for plant uncertainty in your response optimization project.

Specifying Tuned Parameters in the Model (p. 3-2)

Setting the tuned parameters and their characteristics

Including Uncertainty in Parameter Values (p. 3-5)

Setting the uncertain parameters and the methods for accounting for uncertainty in the response optimization project

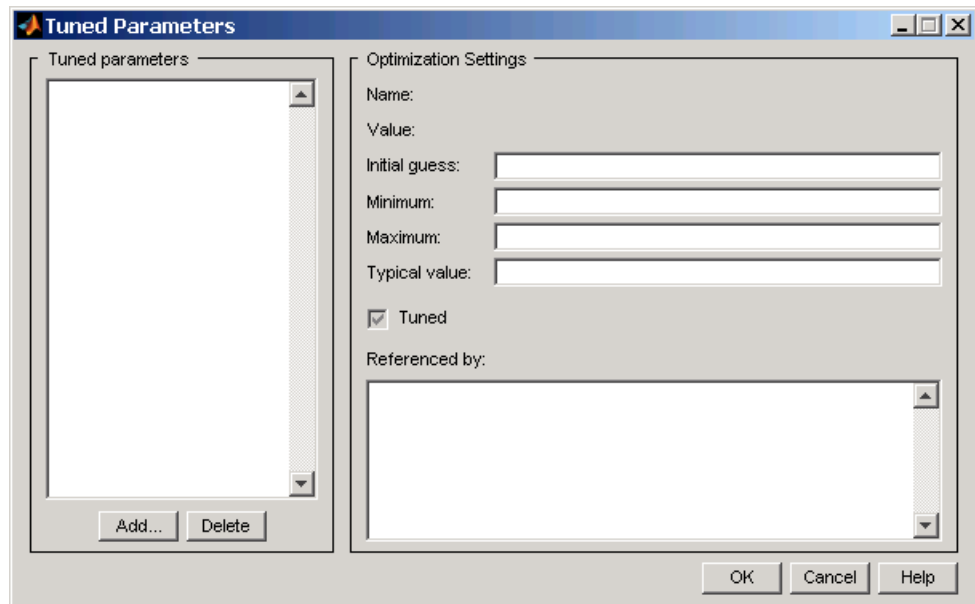
Including Independent Parameters (p. 3-9)

Tuning and adding uncertainty to independent parameters

Specifying Tuned Parameters in the Model

Simulink Response Optimization optimizes the response signals of the model by varying the model's tuned parameters so that the response signals lie within the constraint bound segments or closely match a specified reference signal. You can specify these tuned parameters by selecting **Optimization > Tuned Parameters** in a Signal Constraint window.

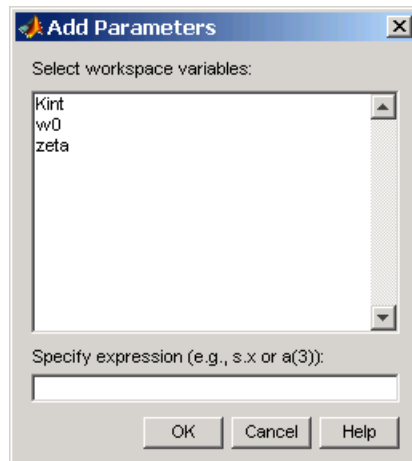
Note When you have more than one Signal Constraint block in your model, you need to specify the tuned parameters in only one window as these settings apply to all constrained signals within the model.



Adding Tuned Parameters

Within the Tuned Parameters dialog, the tuned parameters are shown in a list on the left. To add a tuned parameter to your response optimization project, click the **Add** button. This displays the Select Parameters dialog containing a list of all model parameters of the model currently available in the MATLAB

workspace (if a parameter is already listed in the tuned parameters list, it does not appear in the Select Parameters dialog).



Select the parameters that you want to tune, then click **OK** to add them to the list of tuned parameters. To delete a parameter from the tuned parameters list, select the parameter you want to delete and click **Delete**.

Changing Tuned Parameter Specifications

To display the settings for a particular tuned parameter, select it within the **Tuned Parameters** list. Its settings appear on the right under **Optimization Settings**. These settings include

Setting	Description	Default
Name	The name of the parameter.	Not an editable field
Value	The current value of the parameter.	Not an editable field
Initial guess	The initial value used by the optimization algorithm. A well-chosen initial guess can speed up the optimization and help keep the solution away from undesirable local minima. You can edit this field with numbers, variables, or expressions to provide an alternate initial guess.	The current value of the parameter

Setting	Description	Default
Minimum	The minimum value, or lower bound, that you would like the parameter to take. You can edit this field to provide an alternate minimum value.	- Inf
Maximum	The maximum value, or upper bound, that you would like the parameter to take. You can edit this field to provide an alternate maximum value.	Inf
Typical value	The tuned parameters are scaled, or normalized, by dividing their current value by a typical value. You can edit this field to provide an alternate scaling factor.	The initial value of the parameter
Tuned	This check box indicates whether this parameter is tunable. Select it if you want this parameter to be tuned during the optimization. Unselect if you do not want this parameter to be tuned during the optimization but you would like to keep it on the list of tuned parameters (for a subsequent optimization).	Selected
Referenced by	A list of all blocks this parameter appears in.	Not an editable field

After selecting the tuned parameters for the project and editing their optimization settings, click **OK** to save your changes and exit the Tuned Parameters dialog.

Including Uncertainty in Parameter Values

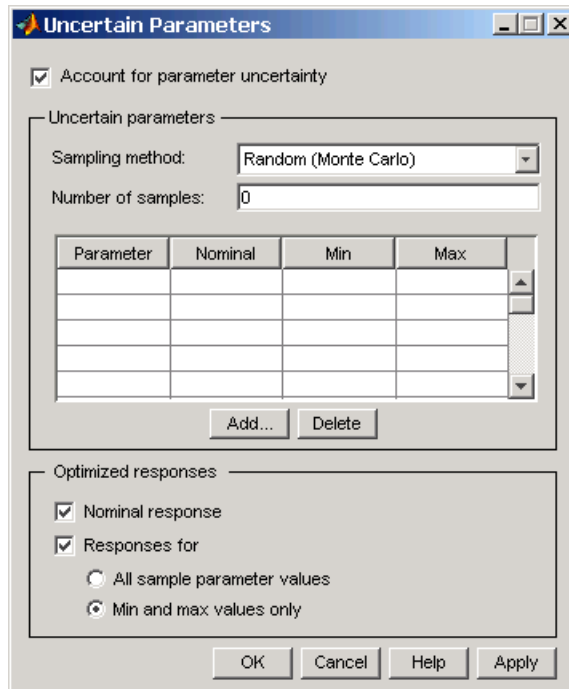
As discussed in “Simple Control Design Example” in the Getting Started Guide, a precise plant model might not be known for your particular problem. Instead you might know what the nominal plant should be and have some idea of the uncertainty inherent in various components of the plant. For example, in “Adding Uncertainty”, the plant parameter zeta varies up to 5% about its nominal value and w_0 varies between 0.7 and 1.45.

Simulink Response Optimization allows you to incorporate uncertainty into your design in two different ways:

- **Passive mode:** Optimize the signals based on the nominal parameter values only. Use the uncertain parameter values to validate the results by plotting responses based on these perturbed values.
- **Active mode:** Optimize signals based on both nominal parameter values as well as perturbed (uncertain) parameter values. This mode is more time consuming.

To specify uncertainty in parameters, select **Optimization > Uncertain Parameters** from the Signal Constraint window.

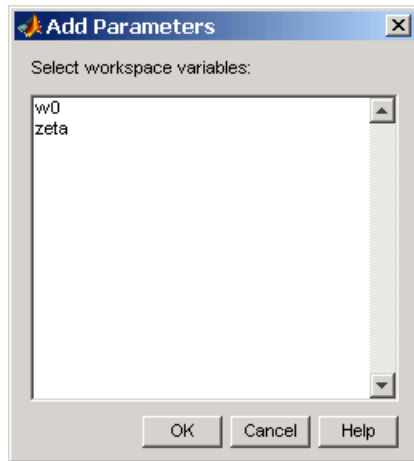
Note When you have more than one Signal Constraint block in your model, you need to specify the uncertain parameters in only one window as these settings apply to all constrained signals within the model.



By default, the **Account for parameter uncertainty** check box is selected when you open the Uncertain Parameters dialog. This indicates that you want to include parameter uncertainty in your optimization. By clearing this option, you can turn off parameter uncertainty without deleting information you have already entered in the **Uncertain parameters** list.

Adding Uncertain Parameters

To add a new uncertain parameter to the **Uncertain parameters** list, click the **Add** button. This displays the Select Parameters dialog containing a list of all model parameters currently available in the MATLAB workspace (if a parameter is already listed in either the tuned parameters or uncertain parameters list, it will not appear in the Select Parameters dialog).



Select the parameters that you want to add uncertainty to, and then click **OK** to add them to the list of uncertain parameters. To delete a parameter from the **Uncertain parameters** list, select the parameter you want to delete and click **Delete**.

Changing Uncertain Parameter Specifications

There are two sampling methods that you can use to investigate the uncertain parameters. Both involve using several sample parameter values within the range of uncertainty.

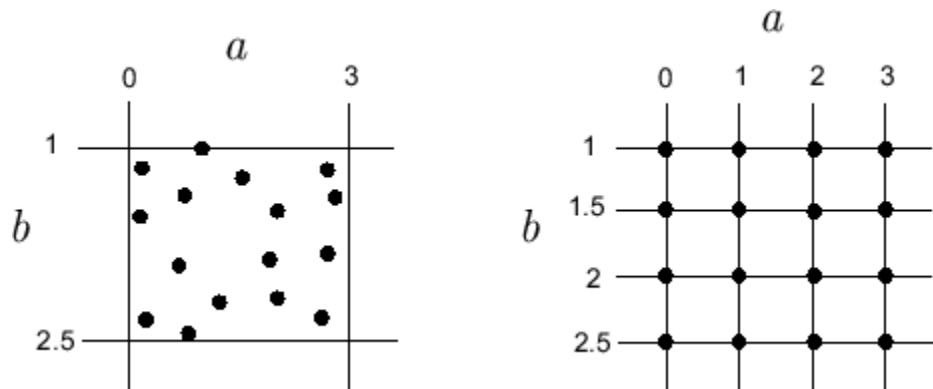
- **Random (Monte Carlo)** computes the optimization at several random parameter values within the range of uncertainty. When this method is selected, you must also enter a value for **Number of samples**, which indicates the number of random parameters that Simulink Response Optimization uses. For each parameter in the uncertain parameters list, you can change the nominal value as well as the range of uncertainty indicated by the maximum and minimum values the parameter can take.

When more than one parameter contains uncertainty, random parameter combinations are chosen within the hyper-rectangle defined by the minimum and maximum values of all parameters. For example, in the case of two uncertain parameters a and b , with values ranging from 0 to 3 and from 1 to 2.5 respectively, the sample values, represented by black

dots, are scattered randomly within the rectangle shown on the left of the following figure.

- **Grid** computes the optimization at several specified parameter values within the range of uncertainty. For each parameter in the uncertain parameters list, you can change the nominal value as well as specify a vector of sample parameter values. **Number of samples** is computed from the sample values specified in the list.

When more than one parameter contains uncertainty, the sample values form a grid of parameter combinations. For example, in the case of two uncertain parameters a and b , with sample values [0 1 2 3] and [1 1.5 2 2.5], the sample values, represented by black dots, form the grid of parameter combinations shown on the left of the following figure.



To increase the speed of the computation, you can choose not to use all sample parameter values in the optimization. To include the nominal parameter values in the optimization, select the **Nominal response** check box. To include parameter values other than the nominal value in the optimization, select the **Response for** check box and then select either **All sample parameter values** or **Min and max values only**. These options include either all sample parameter value combinations or all combinations of minimum and maximum parameter values, respectively. Only the optimized responses are used to adjust the tuned parameters. Responses based on other sample parameter values may still be plotted in the Signal Constraint window.

Including 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 use Simulink Response Optimization to tune and include uncertainty in these independent parameters and an example follows in the next section:

- 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 Add Parameters dialog when you select **Tuned parameters** or **Uncertain parameters**. Add these parameters to the list of tuned parameters to tune them during the response optimization.

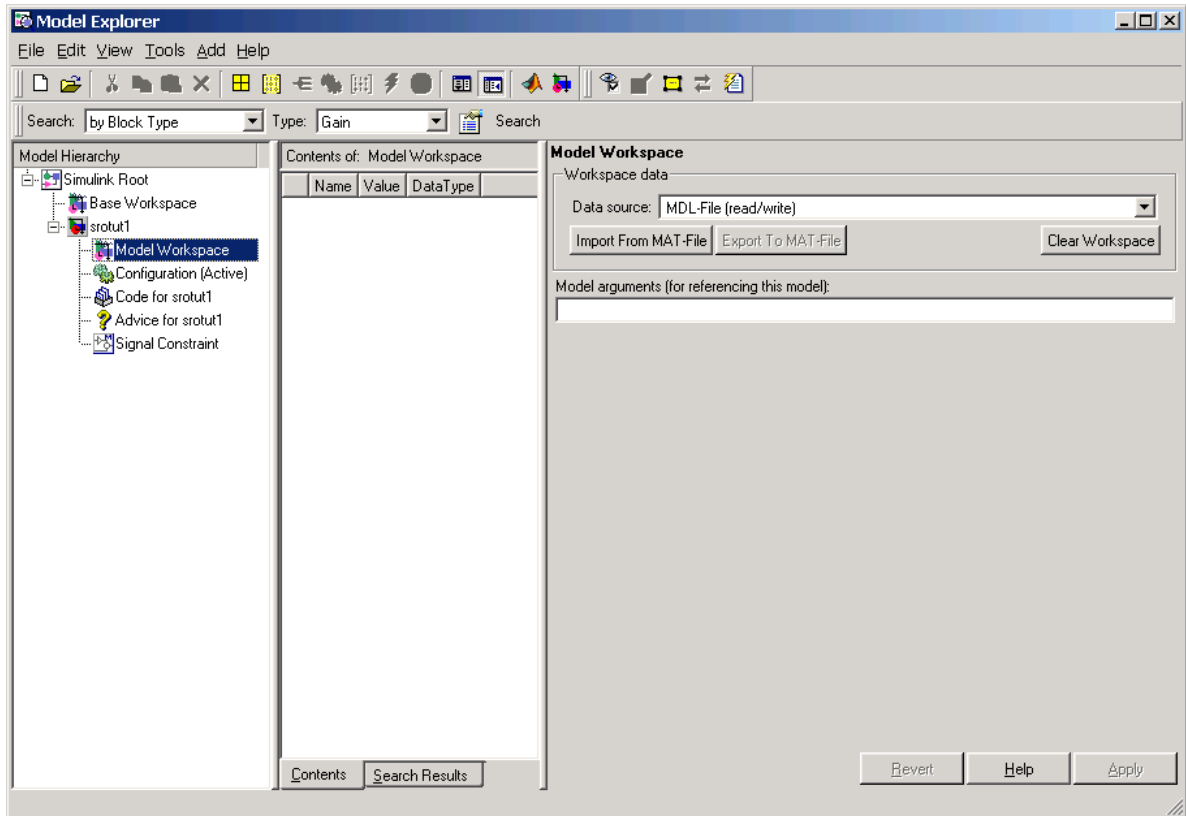
Caution Avoid adding independent parameters together with their corresponding dependent parameters to the lists of tuned and uncertain parameters. Otherwise the optimization could give incorrect results. For example, when a parameter x depends on the parameters a and b , avoid adding all three parameters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

Example:

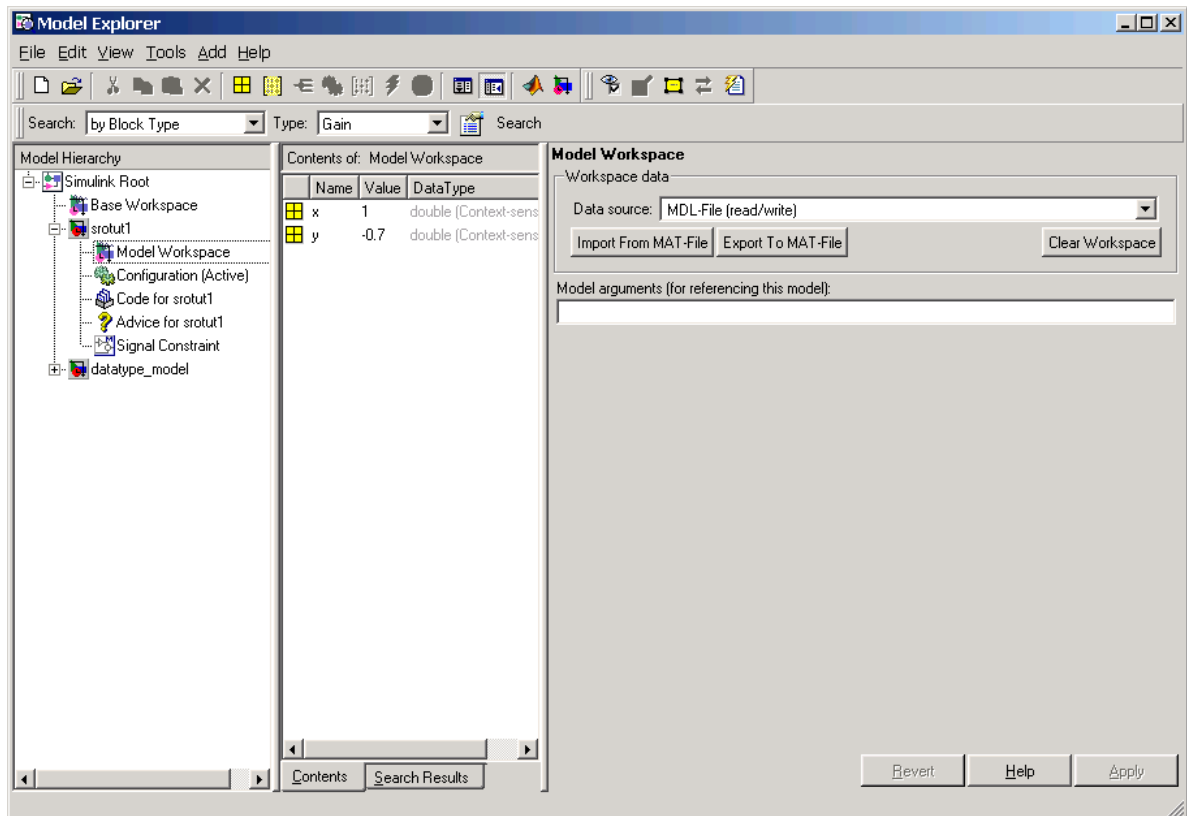
Assume that the parameter K_{int} in the model `srotut1` is related to the parameters x and y according to the relationship $K_{int}=x+y$. Also assume that the initial values of x and y are 1 and -0.7 respectively. To tune x and y instead of K_{int} , first define these parameters in the model workspace.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

- 1 Select **View > Model Explorer** from the `srotut1` window.
- 2 Select **Model Workspace** under the `srotut1` node in the tree browser within the Model Explorer window.

3 Choosing Optimized Parameters



- 3 Select **Add > MATLAB Variable** within the Model Explorer to add a new variable to the model workspace. A new variable appears within the pane labeled **Contents of: Model Workspace**. Change the variable name to **x** and the initial value to **1**.
- 4 Repeat step 3 to add a variable **y** with an initial value of **-0.7**. The Model Explorer window should now look like the following figure.



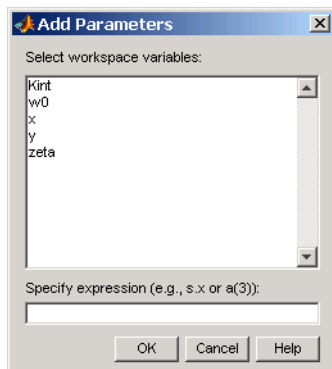
- 5 To add the Simulation Start function defining the relationship between Kint and the independent parameters x and y, select **File > Model Properties** in the srotut1 window, and then select **Callbacks** in the Model Properties dialog.
- 6 Under **Simulation start function**, enter the name of a new M-file, for example, srotut1_start.
- 7 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 look something like 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`.

- 8** When you add a new tuned or uncertain parameter, `x` and `y` should now appear in the Add Parameters dialog.



Running the Optimization

Once you have specified constraints and set the tuned and uncertain parameters, you can run the optimization. If the optimization does not converge the first time, it will often converge after adjusting the constraints or tuned parameter characteristics, or choosing different options. The latter sections of this chapter give advice on how set optimization options and options for the simulations used in the optimization.

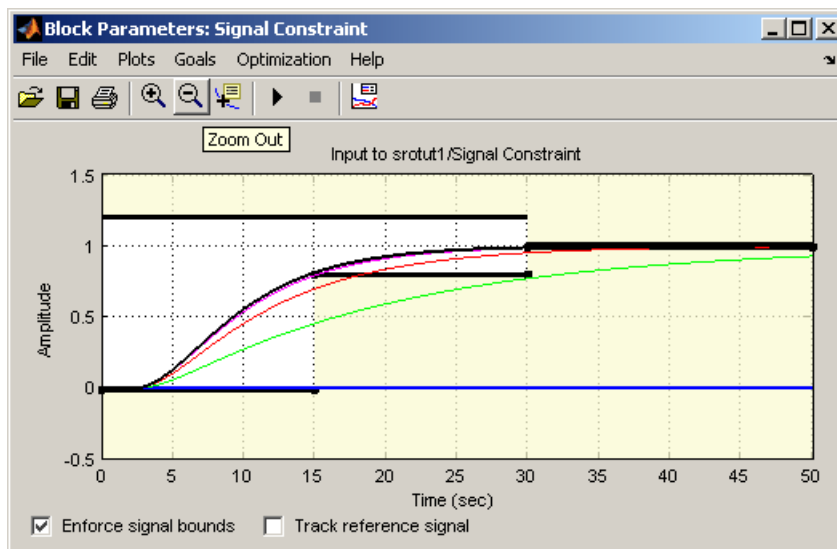
Running the Optimization (p. 4-2)	How to run a response optimization project
Tuning the Optimization Results (p. 4-4)	Description of the various optimization options and suggestions for their use
Setting Options for the Simulation (p. 4-8)	Description of the various simulation options and suggestions for their use
Accelerating the Optimization (p. 4-12)	Using the Simulink Accelerator to increase optimization speed

Running the Optimization

Simulink Response Optimization uses optimization algorithms to find parameter values that allow a feasible solution, or best fit in the case of reference tracking, to the given constraints. Once the appropriate signals have been constrained with signal bounds or by tracking a reference signal, the tuned parameters set, and (optionally) any uncertain parameters and optimization settings specified, you are ready to run the optimization.

Run the optimization by selecting **Optimization > Start** in the Signal Constraint window, or click the Start button, which is the small triangle located on the control panel below the menus.

Simulink Response Optimization begins by plotting the initial response in blue in the Signal Constraint window. During the optimization, intermediate responses are also plotted in various colors. The final response is plotted in black. If uncertainty is included in the optimization, the uncertain response signals are plotted as dashed lines, along with the nominal response as a solid line.

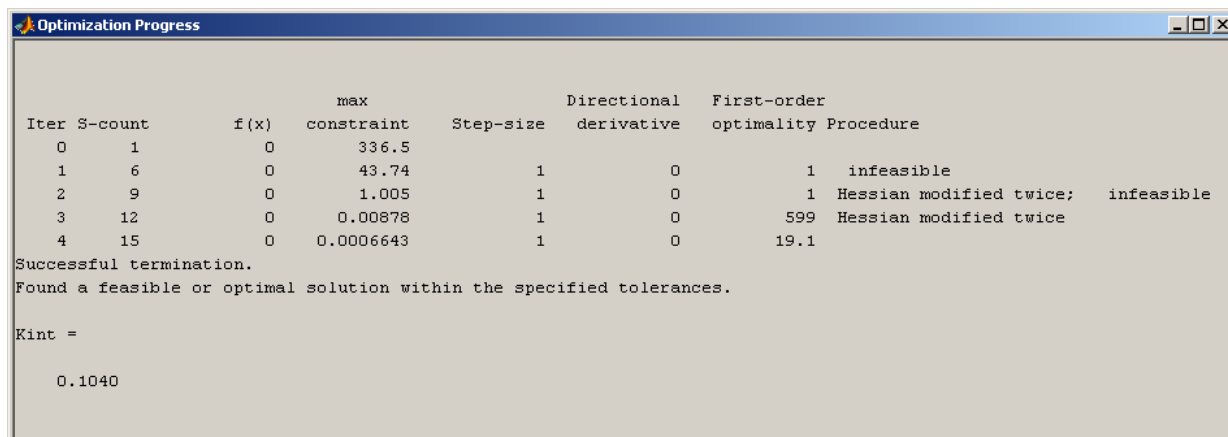


Simulink Response Optimization changes the values of the tuned parameters within the MATLAB workspace and displays the final value in the

Optimization Progress window. Alternatively, you can enter a parameter name at the MATLAB prompt to see its final value.

Note After the optimization, the values of the tuned parameters are changed to the new optimized values. This means that if you want to run another optimization, it will use these tuned values of the parameters as initial values unless you specify alternative initial values in the Tuned Parameters dialog. To revert to the unoptimized parameter values select **Edit > Undo Optimize Parameters** from the Signal Constraint window.

In addition to plotting the response signals and changing the tuned parameter values, numerical output is displayed in the Optimization Progress window. The form of this output depends on the optimization algorithm being used. Refer to the Optimization Toolbox documentation or Genetic Algorithm and Direct Search Toolbox documentation for more information on what type of iterative output is displayed for each algorithm.



```

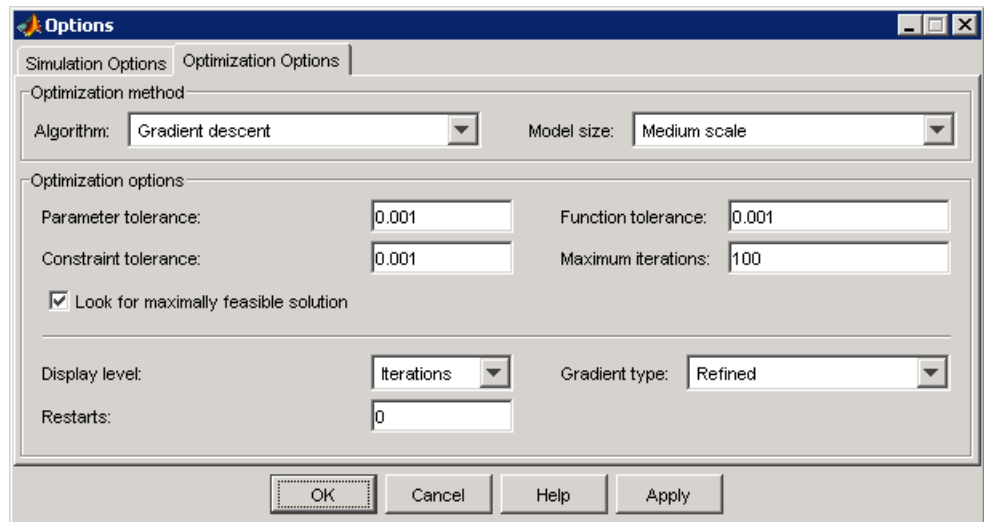
Optimization Progress
-----
Iter S-count      f(x)      max      Directional      First-order
          constraint  Step-size  derivative  optimality Procedure
0         1         0         336.5
1         6         0         43.74          1          0          1  infeasible
2         9         0         1.005          1          0          1  Hessian modified twice;  infeasible
3        12         0         0.00878        1          0         599  Hessian modified twice
4        15         0         0.0006643      1          0         19.1
Successful termination.
Found a feasible or optimal solution within the specified tolerances.

Kint =

    0.1040
  
```

Tuning the Optimization Results

Several options can be set to tune the results of the optimization. These options include the optimization algorithm and the tolerances the algorithms use. To set options for optimization select **Optimization > Optimization Options** in the Signal Constraint window. This opens the Options dialog.



Note If the optimization fails, a good first work-around is to change the **Gradient-type** to **Refined**. For more information on this option, refer to “Selecting Additional Optimization Options” on page 4-6.

Selecting Optimization Methods

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



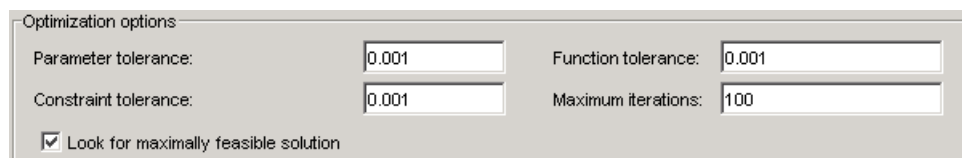
For the **Algorithm** parameter, the three options are

- **Gradient descent** — uses the Optimization Toolbox function `fmincon` to optimize the response signal subject to the constraints.
- **Pattern search** — uses the Genetic Algorithm and Direct Search Toolbox function `patternsearch`, an advanced direct search method, to optimize the response. This option requires the Genetic Algorithm and Direct Search Toolbox.
- **Simplex search** — uses the Optimization Toolbox function `fminsearch`, a direct search method, to optimize the response. **Simplex search** is most useful for simple problems and is sometimes faster than **Gradient descent** for models that contain discontinuities.

By default, the **Model Size** parameter is set to `Medium` scale. When the model is very large and **Gradient descent** is selected as the optimization algorithm, you can change **Model Size** to `Large` scale to increase computation speed. See the Optimization Toolbox documentation or the Genetic Algorithm and Direct Search Toolbox documentation for more information about the optimization methods.

Selecting Optimization Termination Options

Use the **Optimization options** panel to specify when the optimization will terminate.



The screenshot shows the 'Optimization options' dialog box. It contains four input fields: 'Parameter tolerance' (0.001), 'Function tolerance' (0.001), 'Constraint tolerance' (0.001), and 'Maximum iterations' (100). There is also a checked checkbox labeled 'Look for maximally feasible solution'.

- **Parameter tolerance:** When using the **Simplex search** algorithm, the optimization will terminate when successive parameter values change by less than this number. For more details, refer to the discussion of the parameter `TolX` in the reference page for the Optimization Toolbox function `fmincon`.

- **Constraint tolerance:** This number represents the maximum relative amount by which the constraints can be violated and still allow a successful convergence.
- **Function tolerance:** The optimization will terminate when successive function values are less than this value. Changing the default **Function tolerance** value is only useful when you are tracking a reference signal or using the Simplex search algorithm. For more details, refer to the discussion of the parameter TolFun in the reference page for the Optimization Toolbox function fmincon.
- **Maximum iterations:** The maximum number of iterations allowed. The optimization will terminate when the number of iterations exceeds this number.
- **Look for maximally feasible solution:** When selected, the optimization will continue after it has found an initial solution, until it finds a maximally feasible, optimal solution. When this option is unselected, the optimization terminates as soon as it finds a solution that satisfies the constraints and the resulting response signal sometimes lies very close to the constraint edge. In contrast, a maximally feasible solution will typically be located further inside the constraint region.

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** panel is a group of additional optimization options.



The image shows a portion of the Optimization Options panel. It contains three controls: a dropdown menu for 'Display level' with 'Iterations' selected, a dropdown menu for 'Gradient type' with 'Basic' selected, and a text input field for 'Restarts' containing the number '0'.

Additional options for optimization include

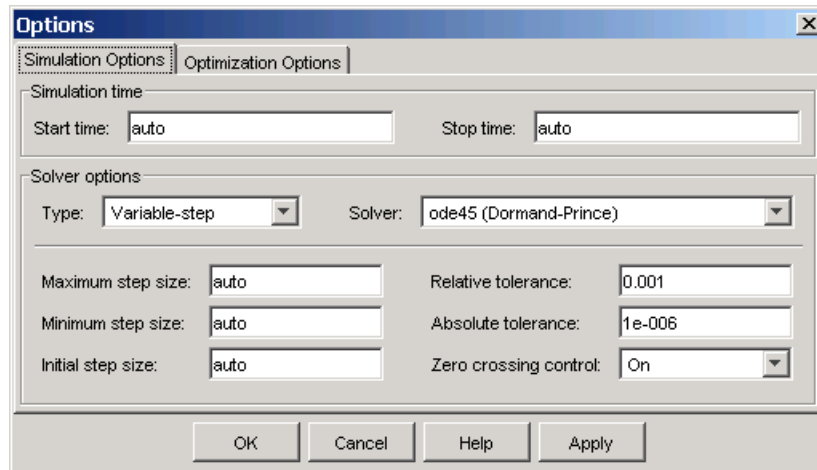
- **Display level:** This option specifies the form of the output that appears in the Optimization Progress window. The options are *Iterations*, which displays information after each iteration, *None*, which turns off all output, *Notify*, which displays output only if the function does not

converge, and Termination, which only displays the final output. Refer to the Optimization Toolbox documentation or Genetic Algorithm and Direct Search Toolbox documentation for more information on what type of iterative output each algorithm displays.

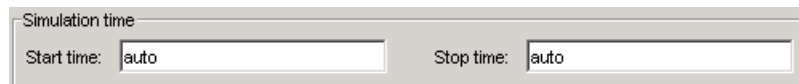
- **Restarts:** In some optimizations the Hessian may become ill conditioned and the optimization does not converge. In these cases it is sometimes useful to restart the optimization after it stops, using the endpoint of the previous optimization as the starting point for the next one. To automatically restart the optimization, indicate the number of times you want to restart in this field.
- **Gradient type:** When using Gradient descent as the optimization algorithm, Simulink Response Optimization calculates gradients based on finite difference methods. The default method for computing the gradients is Basic. The Refined method offers a more robust and less noisy gradient calculation method than Basic, although it is sometimes more expensive and does not work with certain models such as SimPowerSystems models. If the optimization fails, a good first work-around, before changing solvers or adding parameter bounds, is to change **Gradient type** to Refined.

Setting Options for the Simulation

To optimize the response signals of a model, Simulink Response Optimization runs simulations of the model. You can set options for these simulations by selecting **Optimization > Simulation Options** in the Signal Constraint window. This opens the Options dialog.



Selecting Simulation Time



By default, the **Start time** and **Stop time** are automatically set to the model's start and stop times. To specify alternative start and stop times for the response optimization project, enter them in the **Simulation time** pane.

Note Because a stop time of Inf causes Simulink Response Optimization to enter an infinite loop, Simulink Response Optimization automatically replaces this value with the largest time value in the constraints.

Selecting Solvers

Solver options

Type: Variable-step Solver: ode45 (Dormand-Prince)

Maximum step size: auto Relative tolerance: 0.001

Minimum step size: auto Absolute tolerance: 1e-006

Initial step size: auto Zero crossing control: On

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

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)

See the Simulink documentation for information on these solvers.

Variable-Step Solver Options

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

- **Maximum step size:** The largest step-size Simulink can use during a simulation
- **Minimum step size:** The smallest step-size Simulink can use during a simulation
- **Initial step size:** The step-size Simulink 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, Simulink automatically chooses values for these options. To choose your own values, enter them in the appropriate fields. For more information on these options, and the circumstances in which to use them, see the Simulink documentation.

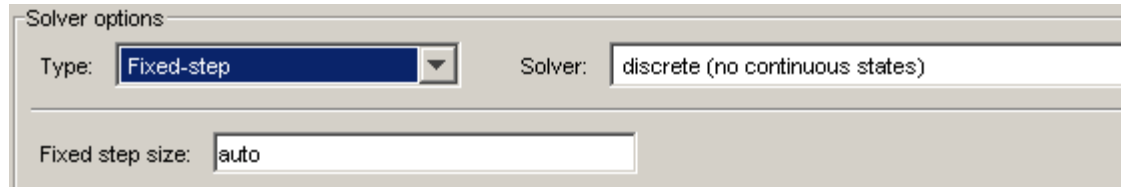
Fixed-Step Solvers

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)

See the Simulink documentation for information on these solvers.

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, Simulink automatically chooses a value for this option.



The image shows a screenshot of the 'Solver options' dialog box in Simulink. The dialog has a title bar that says 'Solver options'. Inside, there are three main settings:

- Type:** A dropdown menu with 'Fixed-step' selected.
- Solver:** A text field containing 'discrete (no continuous states)'.
- Fixed step size:** A text field containing 'auto'.

Accelerating the Optimization

Simulink Response Optimization works automatically and seamlessly with the Simulink Accelerator. Since the majority of optimization time with Simulink Response Optimization is spent conducting simulations, using the Simulink Accelerator could greatly decrease the amount of time it takes to perform an optimization.

If you have the Simulink Accelerator installed on your system, you can use it with Simulink Response Optimization by selecting **Simulation > Accelerator** in the model window. Simulink Response Optimization then runs with the C code generated by the Simulink Accelerator (during the first simulation, the C code will be generated).

To get the most speed from the Simulink Accelerator, you should close all Scope blocks and either remove MATLAB function blocks or replace them with Fcn blocks. The Accelerator does not work with algebraic loops.

Functions — Categorical List

Response Optimization Projects

<code>getsro</code>	Get response optimization project for given Simulink model
<code>ncdupdate</code>	Upgrade models with old NCD blocks
<code>newsro</code>	Create a default Simulink Response Optimization project
<code>optimize</code>	Run response optimization project

Constraints and Parameters

<code>findconstr</code>	Find constraints on a given response
<code>findpar</code>	Find specifications for a given tuned parameter
<code>gridunc</code>	Construct N-D grid of uncertain parameter values
<code>initpar</code>	Initialize tuned parameters
<code>randunc</code>	Randomly sample uncertain parameters
<code>setunc</code>	Specify parameter uncertainty in optimization project

Optimization and Simulation Settings

<code>optimget</code>	Retrieve current optimizer settings
<code>optimset</code>	Modify optimizer settings
<code>simget</code>	Retrieve current simulation settings
<code>simset</code>	Modify simulation settings

Functions — Alphabetical List

findconstr

Purpose Find constraints on a given response

Syntax `constraints=findconstr(proj,'blockname')`

Description `constraints=findconstr(proj,'blockname')` returns a constraint object for the Signal Constraint block, `blockname`, within the response optimization project, `proj`. This object contains all the data defining the desired response including the positions of the constraint bound segments as well as the reference signals. The constraints are used in a response optimization project to define the region in which the response signal must lie.

Modify the constraint object properties `UpperBoundX`, `UpperBoundY`, `LowerBoundX`, and `LowerBoundY` to specify new constraint bound segments on the signals. These properties define the x and y values for the beginning and ending points of each constraint edge.

Modify the constraint object properties `ReferenceX` and `ReferenceY` to specify a new reference signal to track. These properties contain the vectors of x and y data defining the reference signal.

Example Open the model `srotut1` by typing

```
srotut1
```

Create a response optimization project.

```
proj=newsro('srotut1','Kint');
```

Find the constraints object for this project.

```
constraint=findconstr(proj,'srotut1/Signal Constraint')
```

This returns

```
ConstrEnable: 'on'  
isFeasible: 1  
CostEnable: 'off'
```

```

        Enable: 'on'
        Name: 'Signal Constraint'
        SignalSize: [1 1]
        LowerBoundX: [3x2 double]
        LowerBoundY: [3x2 double]
        LowerBoundWeight: [3x1 double]
        UpperBoundX: [2x2 double]
        UpperBoundY: [2x2 double]
        UpperBoundWeight: [2x1 double]
        ReferenceX: []
        ReferenceY: []
        ReferenceWeight: []
    
```

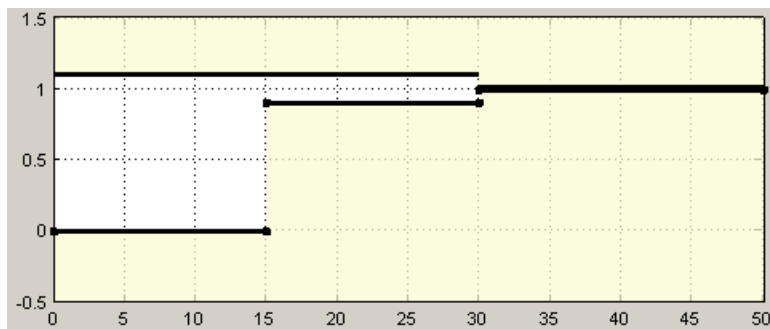
Signal Constraint.

Change the positioning of the constraint bounds by editing the upper and lower bound matrices.

```

constraint.UpperBoundY=[1.1 1.1;1.01 1.01]
constraint.LowerBoundY=[0 0;0.9 0.9;0.99 0.99]
constraint.UpperBoundX=[0 30;30 50]
constraint.LowerBoundX=[0 15;15 30;30 50]
    
```

These bounds define the constraints given in the following figure.

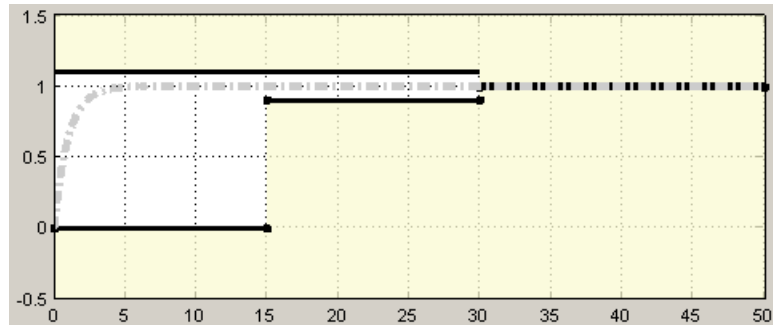


Include a reference signal with the following commands:

findconstr

```
constraint.ReferenceX=linspace(0,50,1000);  
constraint.ReferenceY=1-exp(-linspace(0,50,1000));
```

This defines the reference signal shown in the following figure.



See Also

getsro, newsro, optimize

Purpose Find specifications for a given tuned parameter

Syntax `p=findpar(proj,'param')`

Description `p=findpar(proj,'param')` returns a tuned parameters object for the parameter with the name `param` within the response optimization project, `proj`. The tuned parameters object defines specifications for each tuned parameter that the response optimization algorithm uses, such as initial guesses, lower bounds, etc.

The properties of each tuned parameter object are

Name	A string giving the parameter's name.
Value	The current value of the parameter. This will change during the optimization.
InitialGuess	The initial guess for the parameter value for the optimization
Minimum	The minimum value this parameter can take. By default it is set to <code>-Inf</code> .
Maximum	The maximum value this parameter can take. By default it is set to <code>Inf</code> .
TypicalValue	A value that the tuned parameter is scaled by during the optimization.
ReferencedBy	The block, or blocks, in which the parameter appears.
Description	An optional string giving a description of the parameter
Tuned	Set to 1 or 0 to indicate if this parameter is to be tuned or not.

Edit these properties to specify additional information about your parameters.

findpar

Example

Create a response optimization project for srotut1.

```
proj=newsro('srotut1','Kint');
```

Find the tuned parameters object for the parameter Kint.

```
p=findpar(proj,'Kint')
```

This returns

```
      Name: 'Kint'  
      Value: 0  
InitialGuess: 0  
      Minimum: -Inf  
      Maximum: Inf  
TypicalValue: 0  
ReferencedBy: {0x1 cell}  
Description: ''  
      Tuned: 1
```

Tuned parameter.

Change the initial guess to 0.5, and the minimum value to 0 with the set function.

```
set(p,'InitialGuess',0.5,'Minimum',0)
```

See Also

getsro, newsro, optimize

Purpose Get response optimization project for given Simulink model

Syntax `proj=getsro('modelName')`

Description `proj=getsro('modelName')` returns the response optimization project, `proj`, currently associated with the Simulink model with name, `modelName`. The model should be open and contain Simulink Response Optimization blocks. Use the project with the `optimize` function to optimize response signals in the model by tuning specified parameters.

Example Open the model `pidtune_demo` by typing

```
pidtune_demo
```

Extract the response optimization project from this model

```
proj=getsro('pidtune_demo')
```

This returns

```

Name: 'pidtune_demo'
Parameters: [3x1 ResponseOptimizer.Parameter]
OptimOptions: [1x1 ResponseOptimizer.OptimOptions]
Tests: [1x1 ResponseOptimizer.SimTest]
Model: 'pidtune_demo'
```

Simulink Response Optimization Project.

Use the `findpar` and `findconstr` functions to specify signal constraints and tuned parameters.

See Also `findconstr`, `findpar`, `newsro`, `optimize`

gridunc

Purpose Construct N-D grid of uncertain parameter values

Syntax `uset=gridunc('P1',Values1,'P2',Values2,...)`

Description `uset=gridunc('P1',Values1,'P2',Values2,...)` takes vectors (for scalar-valued parameters) or cell arrays of values `Values1`, `Values2`,... for the uncertain parameters `P1`, `P2`,... and constructs `uset`, an object containing a multidimensional grid of all parameter value combinations.

Optimize the responses based on uncertain parameter values by setting the `Optimized` property of the uncertain parameter object, `uset`, to `true` (by default this value is set to `false`).

Use the `setunc` function to set the uncertain parameter values within the response optimization project.

Example Create a grid of uncertain parameter values for the parameters `P`, `I`, and `D`.

```
uset=gridunc('P',[1,2,3,4],'I',[0.1,0.2,0.3],'D',[30,35,40])
```

This returns

```
Optimized: [4x3x3 logical]
P: [4x3x3 double]
I: [4x3x3 double]
D: [4x3x3 double]
```

4x3x3 grid of parameter vectors.

View the data in detail using dot notation. For example:

```
uset.P
ans(:,:,1) =
```

```
    1    1    1
    2    2    2
    3    3    3
```

```
4 4 4
```

```
ans(:,:,2) =
```

```
1 1 1
2 2 2
3 3 3
4 4 4
```

```
ans(:,:,3) =
```

```
1 1 1
2 2 2
3 3 3
4 4 4
```

To optimize responses based on all the parameter combinations within `uset`, enter the following command.

```
uset.Optimized(1:end)=true
```

See Also

`randunc`, `setunc`

initpar

Purpose	Initialize tuned parameters
Syntax	<code>initpar(proj)</code>
Description	<code>initpar(proj)</code> sets the <code>InitialGuess</code> value of tuned parameters in the response optimization project, <code>proj</code> , with the values of the parameters that are currently in the model or base workspace.
See Also	<code>findpar</code>

Purpose	Upgrade models with old NCD blocks
Syntax	<code>ncdupdate('modelName')</code>
Description	<p><code>ncdupdate('modelName')</code> searches the Simulink model specified by the string 'modelName' for Nonlinear Control Design Blockset (NCD) Outport blocks and replaces them by the equivalent Signal Constraint block from the Simulink Response Optimization library. The model must be open prior to calling <code>ncdupdate</code>. NCD is the version of Simulink Response Optimization that existed before Release 14.</p> <p>When your model automatically loads its NCD settings from an <code>ncdStruct</code> variable, this variable will change in the workspace during the update so that it is compatible with Simulink Response Optimization. Make sure to resave this variable after the update so that the correct settings will load with your model.</p> <p>When your NCD settings are stored in an <code>ncdStruct</code> variable, but do not automatically load with the model, first load the <code>ncdStruct</code> variable into the workspace before calling <code>ncdupdate</code>, and then resave the variable afterwards.</p> <p>To retain the upgraded blocks, make sure you also save the model after running <code>ncdupdate</code>.</p>
See Also	<code>slupdate</code>

Purpose Create a default Simulink Response Optimization project

Syntax `proj=newsro('modelName',params)`

Description `proj=newsro('modelName',params)` creates a new response optimization project, `proj`, for the Simulink model with name `modelName`. The tuned parameters are specified by the cell array of strings, `parameters`. The specified model should contain at least one block from the Simulink Response Optimization library. Type `srolib` to open the library. Use the project with the `optimize` function to optimize response signals in the model by tuning specified parameters.

Example Create a project, `proj`, for the model `pidtune_demo` with the tuned parameters `Kp`, `Ki`, and `Kd`.

```
proj = newsro('pidtune_demo',{'Kp' 'Ki' 'Kd'})
```

This returns

```
      Name: 'pidtune_demo'  
      Parameters: [3x1 ResponseOptimizer.Parameter]  
      OptimOptions: [1x1 ResponseOptimizer.OptimOptions]  
      Tests: [1x1 ResponseOptimizer.SimTest]  
      Model: 'pidtune_demo'
```

Simulink Response Optimization Project.

Use the `findpar` and `findconstr` functions to specify signal constraints and tuned parameters.

See Also `findconstr`, `findpar`, `getsro`, `optimize`

Purpose Retrieve current optimizer settings

Syntax `opt_settings=optimget(proj)`

Description `opt_settings=optimget(proj)` returns the current optimization settings object, `opt_settings`, for the response optimization project `proj`. Use `optimset` to modify the optimization options.

Option	Description	Possible Settings
Algorithm	The optimization algorithm used.	'fmincon' uses the Optimization Toolbox function <code>fmincon</code> ; 'patternsearch' uses the Genetic Algorithm and Direct Search Toolbox function <code>patternsearch</code> ; 'fminsearch' uses the Optimization Toolbox function <code>fminsearch</code> .
Display	The level of information that the optimization displays.	'off' displays no output; 'iter' displays output at each iteration; 'final' displays just the final output; 'notify' displays output only if the function does not converge.

Option	Description	Possible Settings
GradientType	When using 'fmincon' as the Algorithm, Simulink Response Optimization calculates gradients based on finite difference methods. The default method for computing the gradients is 'basic'. The 'refined' method offers a more robust and less noisy gradient calculation method than 'basic', although it is sometimes more expensive and does not work with certain models such as SimPowerSystems models.	'basic' or 'refined'
MaximallyFeasible	By default, the optimization terminates as soon as it finds a solution that satisfies the constraints and the resulting response signal sometimes lies very close to the constraint edge. However, the optimization can continue to search for a maximally feasible solution that will typically be located further inside the constraint region.	0 to terminate the optimization after an initial solution is found; 1 to continue the optimization after an initial solution, in search of a maximally feasible solution.
MaxIter	Maximum number of iterations allowed.	Positive integer
TolCon	Termination tolerance on the constraints.	Positive scalar

Option	Description	Possible Settings
ToIFun	Termination tolerance on the function value.	Positive scalar
ToIX	Termination tolerance on the parameter values.	Positive scalar
Restarts	In some optimizations the Hessian may become ill-conditioned and the optimization does not converge. In these cases it is sometimes useful to restart the optimization after it stops, using the end-point of the previous optimization as the starting point for the next one. To automatically restart the optimization, use this option to indicate the number of times you want to restart.	Nonnegative integer
SearchMethod	Search options for use with the patternsearch algorithm.	See “Search Options” in the Genetic Algorithm and Direct Search Toolbox documentation.

For more information on the possible settings and the values they can take, see the reference page for the MATLAB function `optimset`.

Example

Create a new default response optimization project for the model `srotut1`.

```
proj=newsro('srotut1','Kint');
```

Get the optimization settings for this project.

optimget

```
opt_settings=optimget(proj)
```

This returns the following list of optimization settings and their current values.

```
          Algorithm: 'fmincon'  
          Display: 'iter'  
          GradientType: 'basic'  
MaximallyFeasible: 0  
          MaxIter: 100  
          TolCon: 1.0000e-003  
          TolFun: 1.0000e-003  
          TolX: 1.0000e-003  
          Restarts: 0  
          SearchMethod: []
```

See Also

optimset, simget, simset

Purpose	Run response optimization project
Syntax	<code>result=optimize(proj)</code>
Description	<p><code>result=optimize(proj)</code> optimizes the responses specified in the response optimization project, <code>proj</code>, with the constraints, parameters and settings. The response optimization results are displayed after each iteration. The tuned parameters are changed in the workspace. Enter the parameter name at the MATLAB prompt to see its new value.</p> <p>A results object, <code>result</code>, is also returned. The properties of this object are</p> <ul style="list-style-type: none">• Cost: The final value of the cost function.• ExitFlag: 1 if the optimization terminated successfully, 0 if it did not.• Iteration: The number of iterations. <p>For more information on the results properties, see the reference pages for the Optimization Toolbox functions <code>fmincon</code> and <code>fminsearch</code> and the Genetic Algorithm and Direct Search Toolbox function <code>patternsearch</code>.</p>
Example	<p>Open the <code>pitchrate_demo</code> model.</p> <pre>pitchrate_demo</pre> <p>Create a response optimization project based on the current settings in the model.</p> <pre>proj=getsro('pitchrate_demo');</pre> <p>Run the optimization with the following command.</p> <pre>results=optimize(proj)</pre> <p>The results are displayed as follows.</p>

optimize

Iter	S-count	f(x)	max constraint	Step-size	Directional derivative	First-order optimality	Procedure
0	1	0	1803				
1	14	0	160	1	0	0.0152	
2	21	0	0.2607	1	0	0.00598	Hessian modified
3	28	0	0.04203	1	0	0.0122	Hessian modified
4	35	0	0.001894	1	0	0.00112	Hessian modified
5	42	0	7.631e-006	1	0	5.01e-006	Hessian modified

Successful termination.

Found a feasible or optimal solution within the specified tolerances.

k1 =

0.8674

k2 =

-0.1513

k3 =

-0.5003

results =

Cost: 0
X: [4x1 double]
ExitFlag: 1
Iteration: 5

See Also

findconstr, findpar, getsro, newsro, optimget, optimset

Purpose Modify optimizer settings

Syntax `optimset(proj,'setting1',value1,'setting2',value2,...)`

Description `optimset(proj,'setting1',value1,'setting2',value2,...)` modifies the optimization settings within the response optimization project, `proj`. The value of the optimization setting, `setting1`, is set to `value1`, `setting2` is set to `value2`, etc.

Option	Description	Possible Settings
Algorithm	The optimization algorithm used.	'fmincon' uses the Optimization Toolbox function <code>fmincon</code> ; 'patternsearch' uses the Genetic Algorithm and Direct Search Toolbox function <code>patternsearch</code> ; 'fminsearch' uses the Optimization Toolbox function <code>fminsearch</code> .
Display	The level of information that the optimization displays.	'off' displays no output; 'iter' displays output at each iteration; 'final' displays just the final output; 'notify' displays output only if the function does not converge.

optimset

Option	Description	Possible Settings
GradientType	When using 'fmincon' as the Algorithm, Simulink Response Optimization calculates gradients based on finite difference methods. The default method for computing the gradients is 'basic'. The 'refined' method offers a more robust and less noisy gradient calculation method than 'basic', although it is sometimes more expensive and does not work with certain models such as SimPowerSystems models.	'basic' or 'refined'
MaximallyFeasible	By default, the optimization terminates as soon as it finds a solution that satisfies the constraints and the resulting response signal sometimes lies very close to the constraint edge. However, the optimization can continue to search for a maximally feasible solution that will typically be located further inside the constraint region.	0 to terminate the optimization after an initial solution is found; 1 to continue the optimization after an initial solution, in search of a maximally feasible solution.
MaxIter	Maximum number of iterations allowed.	Positive integer
TolCon	Termination tolerance on the constraints.	Positive scalar

Option	Description	Possible Settings
ToIFun	Termination tolerance on the function value.	Positive scalar
ToIX	Termination tolerance on the parameter values.	Positive scalar
Restarts	In some optimizations the Hessian may become ill-conditioned and the optimization does not converge. In these cases it is sometimes useful to restart the optimization after it stops, using the end-point of the previous optimization as the starting point for the next one. To automatically restart the optimization, use this option to indicate the number of times you want to restart.	Nonnegative integer
SearchMethod	Search options for use with the patternsearch algorithm.	See “Search Options” in the Genetic Algorithm and Direct Search Toolbox documentation.

For more information on the possible settings and the values they can take, see the reference page for the MATLAB function `optimset`.

Example

Create a default response optimization project for the model `srotut1`.

```
proj=newsro('srotut1','Kint');
```

Get the optimization settings for this project.

```
opt_settings=optimget(proj)
```

optimset

This returns the following list of optimization settings and their current values.

```
Algorithm: 'fmincon'  
Display: 'iter'  
GradientType: 'basic'  
MaxIter: 100  
TolCon: 1.0000e-003  
TolFun: 1.0000e-003  
TolX: 1.0000e-003  
Restarts: 0  
SearchMethod: []
```

Use `optimset` to change the maximum number of iterations to 150.

```
optimset(proj, 'MaxIter', 150)
```

To view the changes to `opt_settings`, enter the variable name at the MATLAB prompt.

```
opt_settings
```

This returns

```
Algorithm: 'fmincon'  
Display: 'iter'  
GradientType: 'basic'  
MaxIter: 150  
TolCon: 1.0000e-003  
TolFun: 1.0000e-003  
TolX: 1.0000e-003  
Restarts: 0  
SearchMethod: []
```

See Also

`optimget`, `simset`, `simset`

Purpose

Randomly sample uncertain parameters

Syntax

```
uset=randunc(N, 'P1', Range1, 'P2', Range2, ...)
```

Description

`uset=randunc(N, 'P1', Range1, 'P2', Range2, ...)` generates random values for the parameters P1, P2, ... subject to the range constraints Range1, Range2, ...

The parameters P1, P2, ... are uncertain parameters in a response optimization project. Each range constraint specifies lower and upper bounds for the uncertain parameter value.

For a scalar-valued parameter, p , specify the range as $[Min, Max]$ or $\{Min, Max\}$. The interpretation is then

$$Min \leq p \leq Max$$

For vector- or matrix-valued parameters, specify the range as $\{Min, Max\}$ where Min and Max are commensurate vectors or matrices. The interpretation is then

$$Min(i, j) \leq p(i, j) \leq Max(i, j)$$

The set of uncertain parameter values consists of

- All vertices of the parameter box specified by the upper and lower bounds (2^S values if there are S parameters)
- N randomly picked points inside the parameter box, where N is the first input argument to `randunc`.

To optimize the responses based on uncertain parameter values, set the `Optimized` property of the uncertain parameter object, `uset`, to `true`. By default this value is set to `false`.

Use the `setunc` function to set the uncertain parameter values within the response optimization project.

randunc

Example

Create a set of 12 randomly generated uncertain parameter values for the parameters P, I, and D.

```
uset=randunc(4,'P',{1,4},'I',{0.1,0.3},'D',{30,40})
```

This returns

```
Optimized: [12x1 logical]
P: [12x1 double]
I: [12x1 double]
D: [12x1 double]
```

Scattered set with 12 parameter vectors.

View the data in detail using dot notation. For example:

```
uset.P
ans =
    1.0000
    4.0000
    1.0000
    4.0000
    1.0000
    4.0000
    1.0000
    4.0000
    1.0000
    4.0000
    3.5155
    2.7042
    2.1112
    3.1082
```

To optimize responses based on all the parameter combinations within `uset`, enter the following command.

```
uset.Optimized(1:end)=true
```

See Also

`gridunc`, `setunc`

Purpose Specify parameter uncertainty in optimization project

Syntax `setunc(proj,unc_settings)`

Description `setunc(proj,unc_settings)` sets the parameter uncertainty specifications for the response optimization project, `proj`. Use the function `gridunc` or `randunc` to specify the uncertainty settings, `unc_settings`.

Example Create a response optimization project.

```
proj=newsro('srotut1','Kint');
```

Specify uncertain parameter settings using `gridunc`.

```
uset=gridunc('zeta',[0.9,1,1.1],'w0',[0.95,1,1.05])
```

Set the uncertain parameters in the project.

```
setunc(proj,uset)
```

See Also `gridunc`, `randunc`

simget

Purpose Retrieve current simulation settings

Syntax `simoptions=simget('proj')`

Description `simoptions=simget('proj')` returns a object containing the current simulation options, `simoptions`, used by the response optimization project, `proj`. To modify the project's simulation settings, use the function `simset`.

For a detailed list of simulation options and the possible values they can take, see the reference page for the Simulink function `simset`. The default values of the simulation options for the project are the same as those used by the Simulink model the project is associated with. Changes that are made to the project's simulation settings will only be used during simulations that are run as part of the optimization and will not affect the simulation settings for the model.

Example Create a response optimization project for the `srotut1` model.

```
proj=newsro('srotut1','Kint')
```

Get the simulation settings for this project

```
simget(proj)
```

This returns

```
ans =  
    AbsTol: 1.0000e-006  
    FixedStep: 'auto'  
    InitialStep: 'auto'  
    MaxStep: 'auto'  
    MinStep: 'auto'  
    RelTol: 1.0000e-003  
    Solver: 'ode45'  
    ZeroCross: 'on'  
    StartTime: '0.0'  
    StopTime: '50'
```

See Also `optimget, optimset, simset`

simset

Purpose Modify simulation settings

Syntax `simset(proj, 'setting1', value1, 'setting2', value2, ...)`

Description `simset(proj, 'setting1', value1, 'setting2', value2, ...)` modifies the simulation settings within the response optimization project, `proj`. The value of the simulation setting, `setting1`, is set to `value1`, `setting2` is set to `value2`, etc.

For a detailed list of simulation options and the possible values they can take, see the reference page for the Simulink function `simset`. The default values of the simulation options for the project are the same as those used by the Simulink model the project is associated with. Changes that are made to the project's simulation settings will only be used during simulations that are run as part of the optimization and will not affect the simulation settings for the model.

Example Create a response optimization project for the `srotut1` model.

```
proj=newsro('srotut1', 'Kint')
```

Get the simulation settings for this project.

```
simget(proj)
```

This returns

```
ans =  
    AbsTol: 1.0000e-006  
    FixedStep: 'auto'  
    InitialStep: 'auto'  
    MaxStep: 'auto'  
    MinStep: 'auto'  
    RelTol: 1.0000e-003  
    Solver: 'ode45'  
    ZeroCross: 'on'  
    StartTime: '0.0'  
    StopTime: '50'
```


Use `simset` to change the solver type to `ode23` and the absolute tolerance to `1e-7`.

```
simset(proj, 'Solver', 'ode23', 'AbsTol', 1e-7)
```

Check the new values:

```
sim_settings=simget(proj);  
sim_settings.Solver
```

This shows that the solver is now set to `ode23`.

```
ans =  
ode23
```

Check the absolute tolerance:

```
sim_settings.AbsTol
```

This value is now set to `1e-7`.

```
ans =  
1.0000e-007
```

See Also

`optimget`, `optimset`, `simget`

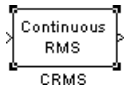
Blocks — Alphabetical List

CRMS

Purpose Compute the continuous-time, cumulative root mean square (CRMS) of a signal

Library Simulink Response Optimization

Description Attach the CRMS block to a signal to compute its continuous-time, cumulative root mean square value. Use in conjunction with the Signal Constraint block to optimize the signal energy.



The continuous-time, cumulative root mean square value of a signal $u(t)$, is defined as

$$R.M.S = \sqrt{\frac{1}{T} \int_0^T \|u(t)\|^2 dt}$$

The R.M.S value gives a measure of the average energy in the signal.

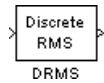
See Also DRMS, Signal Constraint

Purpose

Compute the discrete-time, cumulative root mean square (DRMS) of a signal

Library

Simulink Response Optimization

Description

Attach the DRMS block to a signal to compute its discrete-time, cumulative root mean square value. Use in conjunction with the Signal Constraint block to optimize the signal energy.

The discrete-time, cumulative root mean square value of a signal $u(t_i)$, is defined as

$$R.M.S = \sqrt{\frac{1}{N} \sum_{i=1}^N \|u(t_i)\|^2}$$

The R.M.S value gives a measure of the average energy in the signal.

See Also

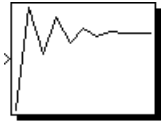
CRMS, Signal Constraint

Signal Constraint

Purpose Specify desired signal response

Library Simulink Response Optimization

Description



Signal Constraint

Attach a Signal Constraint block to a signal to optimize the response of the signal to known inputs. Simulink Response Optimization tunes parameters in the model to meet specified constraints. The constraints include bounds on signal amplitudes and matching of reference signals. The constraints are applicable to vector- and matrix-valued ports, in which case the signal bounds and reference signals apply to all entries of the signal/matrix.

For a complete discussion of the use of this block, see the Simulink Response Optimization documentation starting with Chapter 1, “Approaching Response Optimization”.

See Also CRMS, DRMS

A

algorithms
 response optimization 4-4

C

constraint bounds 2-2
 positioning exactly 2-3
 splitting 2-9
constraint edges
 moving 2-2
constraints
 positioning bounds 2-2
 scaling 2-8
 weightings 2-4
CRMS block 7-2
current responses
 plotting 2-14

D

desired responses
 constraint bounds 2-2
 reference signals 2-13
 step responses 2-10
DRMS block 7-3

E

Edit Constraint dialog 2-3

F

findconstr function 6-2
findpar function 6-5

G

getsro function 6-7
gridlines
 response plots 2-3

gridunc function 6-8

I

initial responses
 plotting 2-14
initpar function 6-10
intermediate responses
 plotting 2-14

L

loading
 response optimization projects 1-6

N

ncdupdate function 6-11
newsro function 6-12

O

optimget function 6-13
optimize function 6-17
optimset function 6-19

P

parameters
 tuned 3-2
 uncertain 3-5
positioning constraints
 snapping to horizontal 2-3
 snapping to vertical 2-3

R

randunc function 6-23
reference signals
 plotting 2-14
 specifying 2-13

- tracking 2-13
- response optimization
 - accelerating 4-12
 - choosing signals 1-2
 - creating projects 1-3
 - gradient type 4-5
 - maximally feasible solutions 4-5
 - running 4-2
 - simulation options 4-8
 - simulation solvers 4-9
 - starting 4-2
 - termination criteria 4-5
 - tolerances 4-5
- response optimization projects
 - properties 1-3
 - reloading 1-6
 - saving 1-4
- response optimization results
 - Optimization Progress dialog 4-3
 - plots 4-2
 - tuning 4-4
- response plots
 - editing properties 2-15

S

- saving
 - response optimization projects 1-4
- setunc function 6-25
- Signal Constraint block 7-4

- signal responses
 - plotting 2-14
- simget function 6-26
- simset function 6-28
- simulation options
 - response optimization 4-8
- simulation solvers
 - response optimization 4-9
- step responses
 - specifications 2-10

T

- tuned parameters
 - adding 3-2
 - results 4-2
 - specifications 3-3
 - specifying 3-2
 - undoing 4-3

U

- uncertain parameters
 - adding 3-6
 - grid 3-7
 - random 3-7
 - specifications 3-7
 - specifying 3-5
 - using in response optimization projects 3-5