

Simulink[®] Response Optimization

For Use with Simulink[®]

■ Modeling

■ Simulation

■ Implementation

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 by The MathWorks, Inc.

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

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

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.

Printing 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)

Introduction

1

What Is Simulink Response Optimization?	1-2
Applications	1-2
System Requirements	1-4
Upgrading from the Nonlinear Control Design Blockset ..	1-5
Using the Documentation	1-6
Expected Background	1-6
How to Use This Guide	1-6
Online Documentation	1-6

Response Optimization Tutorial

2

Quick Start	2-2
Simple Control Design Example	2-4
Simulink Response Optimization Startup	2-4
Adjusting Constraints	2-5
Specifying Tuned Parameters	2-8
Running the Optimization	2-10
Tracking a Signal	2-12
Adding Uncertainty	2-16
Saving the Project	2-19
Physical Modeling Example	2-20
Opening the Hydraulic Cylinder Model	2-20
Adjusting Constraints	2-22
Specifying Tuned Parameters	2-26
Running the Optimization	2-27

Changing Optimization Settings	2-28
--------------------------------------	------

Approaching Response Optimization

3

Choosing Signals to Constrain	3-2
Attaching Signal Constraint Blocks	3-2
Creating a Response Optimization Project	3-3
Saving and Reloading Response Optimization Projects ...	3-4
Saving Response Optimization Projects	3-4
Saving Additional Settings	3-5
Reloading Response Optimization Projects	3-6

Specifying the Desired Response

4

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

Initial Response	4-14
Intermediate Steps	4-14
Response Plots Property Editor	4-15
Labels Pane	4-15
Limits Pane	4-16

Choosing Optimized Parameters

5

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

Running the Optimization

6

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

Accelerating the Optimization 6-11

Response Optimization Using Functions

7

Control Design Example Using Functions 7-2

- Choosing Signals to Constrain 7-2
- Creating an Optimization Project 7-2
- Properties of a Response Optimization Project 7-3
- Running the Optimization 7-9

Troubleshooting

8

Common Questions About Response Optimization 8-2

Function Reference

9

Functions — By Category 9-2

- Response Optimization Projects 9-2
- Constraints and Parameters 9-2
- Optimization and Simulation Settings 9-2

Functions — Alphabetical List 9-3

Block Reference

10

Blocks — Alphabetical List 10-2

Index

Introduction

What Is Simulink Response Optimization? (p. 1-2)	Introduction to the key features of Simulink Response Optimization
System Requirements (p. 1-4)	Required and related products
Upgrading from the Nonlinear Control Design Blockset (p. 1-5)	Information on upgrading NCD models for use with Simulink Response Optimization
Using the Documentation (p. 1-6)	Expected background, how to use this guide, and other documentation sources

What Is Simulink Response Optimization?

Simulink® Response Optimization provides a graphical user interface (GUI) to assist in tuning and optimization of control systems and physical systems. With this product, you can tune parameters within a nonlinear Simulink model to meet time-domain performance requirements by graphically constraining signals within a time-domain window or tracking and closely matching a reference signal. You can tune any number of Simulink variables including scalars, vectors, and matrices. In addition, you can place uncertainty bounds on other variables in the model for robust design. Simulink Response Optimization makes attaining performance objectives and optimizing tuned parameters an intuitive and easy process.

To use Simulink Response Optimization, you need only to include a special block, the Signal Constraint block, in your Simulink diagram. Just connect the block to any signal in the model to signify that you want to place some kind of constraint on the signal. Simulink Response Optimization automatically converts time-domain constraints into a constrained optimization problem and then solves the problem using optimization routines taken from the Optimization Toolbox or the Genetic Algorithm and Direct Search Toolbox. The constrained optimization problem formulated by Simulink Response Optimization iteratively calls for simulations of the Simulink system, compares the results of the simulations with the constraint objectives, and uses gradient methods to adjust tuned parameters to better meet the objectives.

Two additional blocks, CRMS and DRMS, compute the continuous and discrete cumulative root mean square values of signals. Use them with the Signal Constraint block to optimize the cumulative root mean square of signals in your model.

Applications

You can use Simulink Response Optimization for a variety of applications. Possible uses include

- Designing and optimizing control systems by tuning gains.
- Designing physical systems by adjusting parameters in your system. For example, adjust the dimensions of physical devices such as robot arms or hydraulic pistons, tune properties of materials such as thermal emissivity, etc.

- Closely tracking a reference, or desired, signal.
- Optimizing responses for systems that include physical actuation limits and constraints on state/variable values.
- Minimizing the energy in a system by minimizing the root-mean-square signal.
- Including uncertainty in your parameter values to take into account imperfect knowledge of certain physical parameters in your model.

System Requirements

Simulink Response Optimization has the same system requirements as MATLAB®. Refer to the MATLAB documentation for details. For a list of required and related products, see the Simulink Response Optimization product page on the MathWorks Web site at <http://www.mathworks.com/products/simresponse/>.

Upgrading from the Nonlinear Control Design Blockset

Simulink Response Optimization replaces the Nonlinear Control Design (NCD) Blockset. For information on upgrading from the Nonlinear Control Design Blockset, refer to the Release Notes. In addition, the `ncdupdate` reference page has detailed information on updating NCD models.

Using the Documentation

Expected Background

Users of this guide should be familiar with dynamic systems design and analysis, and have experience creating Simulink models.

How to Use This Guide

To get started using Simulink Response Optimization, read Chapter 2, “Response Optimization Tutorial,” which introduces the main product features and uses two simple examples to describe their use.

To perform response optimization using functions, read Chapter 7, “Response Optimization Using Functions.”

For advice on solving typical problems, read Chapter 8, “Troubleshooting.”

If you are upgrading from the Nonlinear Control Design (NCD) Blockset, read the Release Notes to learn about new features and how to convert your NCD models for use with Simulink Response Optimization.

Online Documentation

Further documentation is available online, including function and block references, and detailed discussions on setting up and running a response optimization.

Response Optimization Tutorial

Use Simulink Response Optimization to solve a wide variety of control and physical design problems by inserting Signal Constraint blocks in your model and using the graphical user interface (GUI) to specify constraints on these signals. Specify tuned and uncertain parameters as well as optimization settings. To get started with Simulink Response Optimization, work through the two examples in this chapter.

Quick Start (p. 2-2)

A brief outline of the response optimization process

Simple Control Design Example (p. 2-4)

An example that tunes a single system parameter to optimize a response signal

Physical Modeling Example (p. 2-20)

An example that tunes multiple system parameters to optimize multiple response signals

Quick Start

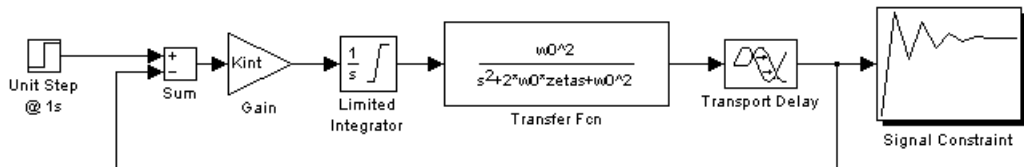
If you would like to get started using Simulink Response Optimization quickly, this section gives a quick outline of the response optimization process:

- 1** Make a model of your (nonlinear) system and controller using Simulink. Add input signals (e.g., steps, ramps, observed data) for which you know what the desired output should look like.
- 2** Attach a Signal Constraint block to the signals you want to constrain. The Simulink library `srolib` contains the Signal Constraint block. To open the library, just type `srolib` at the MATLAB prompt or select **Simulink Response Optimization** from the Simulink Library Browser.
- 3** If the model's parameters do not already exist in the MATLAB workspace, initialize these parameters in the workspace with a best first guess.
- 4** Double-click on each Signal Constraint block in your system to display the **Signal Constraint** window for each constrained output. Choose a method for constraining the response signal by selecting **Enforce signal bounds** and/or **Track reference signal** at the bottom of the window.
- 5** Within the **Signal Constraint** window, constrain each signal by positioning constraint bound segments and/or tracking a reference signal. You can position constraint bound segments by clicking and dragging the segment or right clicking on the segment and selecting **Edit** from the menu. Plot reference signals by selecting **Goals -> Desired Response** in the **Signal Constraint** window and entering vectors of data for the reference signal.
- 6** Open the **Tuned Parameters** dialog by selecting **Optimization -> Tuned Parameters** within a **Signal Constraint** window. Click the **Add** button to add parameters to the list. Select a parameter in the list to specify initial guesses and maximum and minimum values.
- 7** Optional: Open the **Uncertain Parameters** dialog by selecting **Optimization -> Uncertain Parameters** within a **Signal Constraint** window. Click the **Add** button to add parameters to the list. Choose a method and range for sampling the uncertain parameters and select which responses to optimize.

- 8** Optional: Save the project, including constraints, tuned parameters, uncertain parameters, and settings for optimization and simulation, to a file, to the MATLAB workspace, or to the model workspace by selecting **File -> Save**. You can retrieve previously saved projects by selecting **File -> Load**. To automatically save and reload the project with the Simulink model, select the check box at the bottom of the **Save** dialog. Note that when saving the project, you are saving the *entire* optimization project, which might include several Signal Constraint blocks.
- 9** Click the Start button in the toolbar or select **Start** from the **Optimization** menu to optimize the response signal by adjusting the tuned parameters. The **Optimization Progress** window displays the new, optimized parameter values.

Simple Control Design Example

Simulink Response Optimization uses time-domain constraint bounds to represent lower and upper bounds on response signals. You can stretch, move, split, or open constraint bounds in a variety of ways that are explained here and in the online documentation. This section guides you through an example of how you might perform control design using Simulink Response Optimization. In this section, you will constrain a parameter to control a second order SISO system via integral action, shown in the diagram below.



Specifically, the integral gain (K_{int}) should ensure that the closed loop system meets or exceeds the following performance specifications when you excite the system with a unit step input:

- A maximum 10 percent overshoot
- A maximum 10 second rise time
- A maximum 30 second settling time

Because of the actuator limits and system transport delay, standard linear control design techniques may not yield reliable results.

Simulink Response Optimization Startup

The Simulink model `srotut1` contains the system shown above. Open the system by typing `srotut1` at the MATLAB prompt.

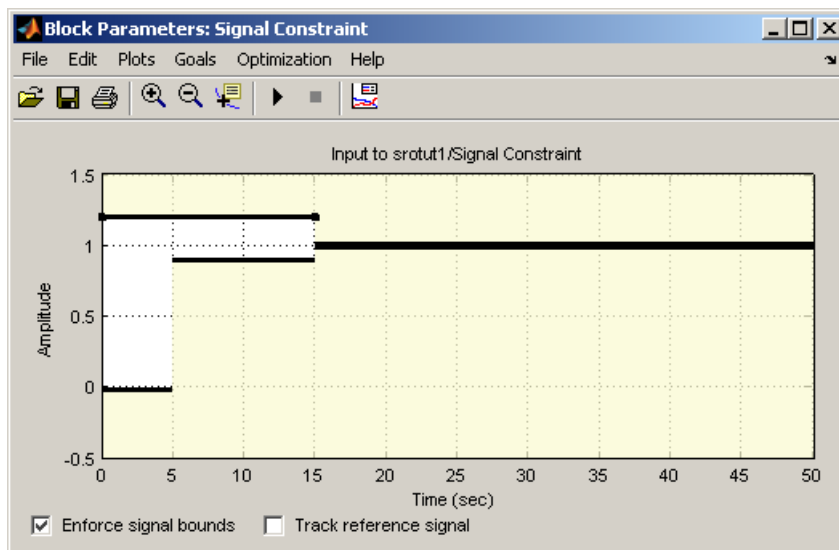
You do not need to remodel any of your present Simulink systems to use Simulink Response Optimization. You need simply to

- Attach an Signal Constraint block to all signals you want to constrain. In `srotut1`, a Signal Constraint block (square block with a step response icon) is attached to the plant output.

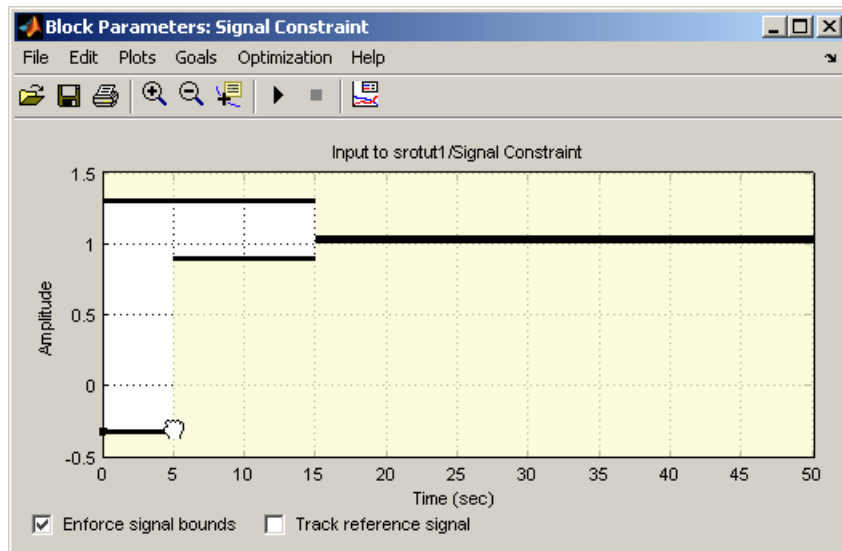
- Add input signals to the system, for which you know what the output should look like. In `srotut1`, the input is a step since the desired step response characteristics of the system are known.
- Change the model's **Stop time** to the desired value. In `srotut1`, the step response should settle within 30 seconds, so simulating for 50 seconds allows the step to go to completion. Since optimization with Simulink Response Optimization calls for many simulations of the system, you should make the simulation time as short as possible, but long enough to show dynamics of interest. You can change the **Start time** and **Stop time** through the Simulink **Simulation Parameters** dialog by selecting **Simulation -> Configuration Parameters**.

Adjusting Constraints

To open the Simulink Response Optimization **Signal Constraint** window, double-click on the Signal Constraint block. The window, shown in the following figure, contains an amplitude versus time axis with default upper and lower constraint bounds. To optimize the tuned parameters so that the response signal lies within the constraint bound segments, select the **Enforce signal bounds** check box at the bottom of the window.



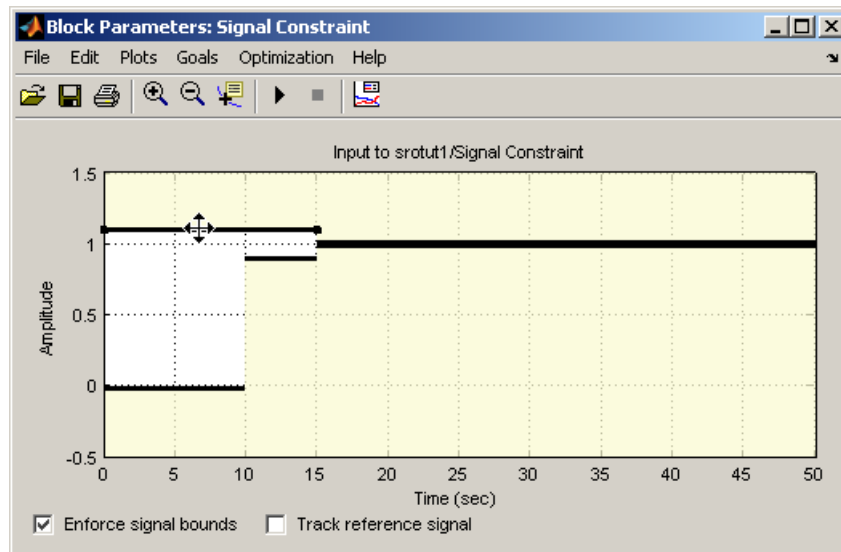
The lower and upper constraint bounds define a channel within which the signal response should lie. The default constraints effectively define a rise time of 5 seconds and a settling time of 15 seconds. These bounds must change to reflect the performance requirements proposed in the beginning of this section. To adjust the rise time constraint, position the mouse over the endpoint of the lower bound constraint that ends at 5 seconds. Press and hold down the (left) mouse button. The arrow should turn to a hand symbol as shown below.



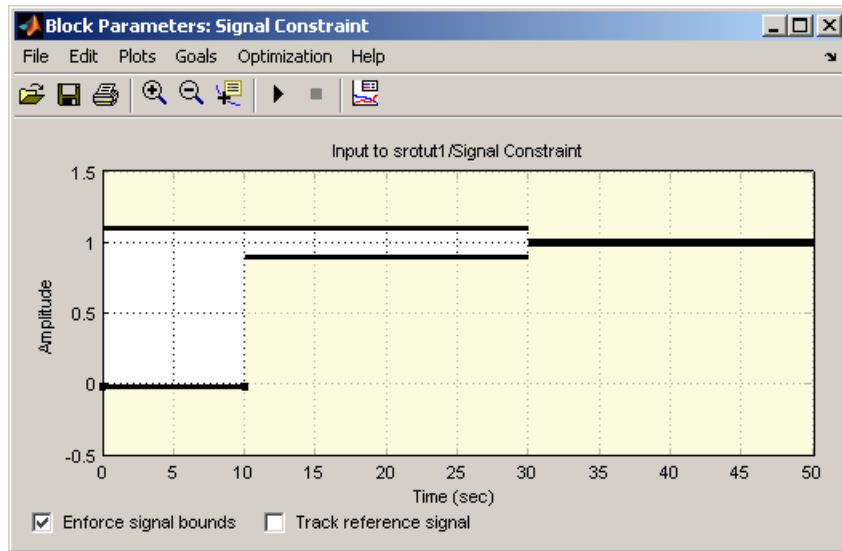
In this mode, you can change the time boundary between two constraints as well as change the slope of constraints. While still holding the mouse button down, drag the constraint boundary to the right. Release the mouse button after positioning the boundary as close as possible to 10 seconds. You may find it helpful to enable axis gridding while placing constraints. To turn axis gridding on or off, right-click anywhere within the white space of the **Signal Constraint** window and select **Grid** from the menu. If you need to precisely place constraint-bound segments, use the **Edit Constraint** dialog, which appears when you right-click on a constraint-bound segment and select **Edit** from the menu. See the online documentation for more information on the **Edit Constraint** dialog.

To adjust the overshoot constraint, press and hold the (left) mouse button somewhere in the middle of the upper bound constraint bound segment that

extends from 0 to 15 seconds. Notice that the pointer becomes a four way arrow. In this mode, you can drag the entire constraint edge anywhere within the axes. While still holding the mouse button down, drag the constraint until its lower boundary is at a height of 1.1 as shown below.



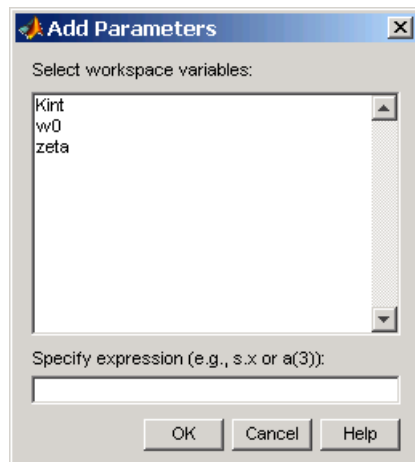
Finally, the settling time constraints require adjustment. Position the mouse button on the right edge of the lower constraint edge extending from 10 to 15 seconds. Press and hold down the (left) mouse button and notice that the constraint becomes selected and that the pointer changes to a hand symbol. In this mode, you can stretch the end of the constraint or change its angle. While still holding the mouse button down, drag the constraint so that the settling-time constraint begins at 30 seconds. Adjust the upper bound constraint so that it too defines a 30 second settling-time constraint. The constraint figure should now look the one shown below.



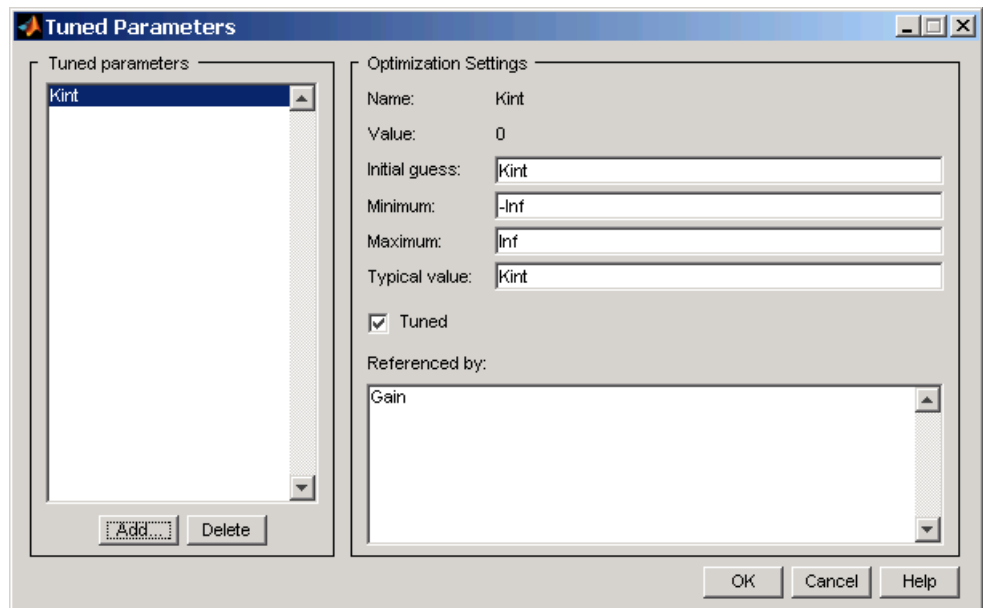
Alternatively, you could set the signal constraints using the **Desired Response** dialog by selecting **Goals -> Desired Response** from the **Signal Constraint** window. For more information on setting signal constraints, see the online documentation.

Specifying Tuned Parameters

Before beginning the optimization, you must tell Simulink Response Optimization which variables the optimization should tune. Open the **Tuned Parameters** dialog by selecting **Optimization -> Tuned Parameters** within the **Signal Constraint** window. Add the parameter **Kint** to the **Tuned parameters** list by clicking the **Add** button. This opens the **Select Parameters** dialog with a list of all model parameters that are currently in the workspace.



Select **Kint** in this list and click **OK**. This adds **Kint** to the **Tuned parameters** list as shown in the following figure.

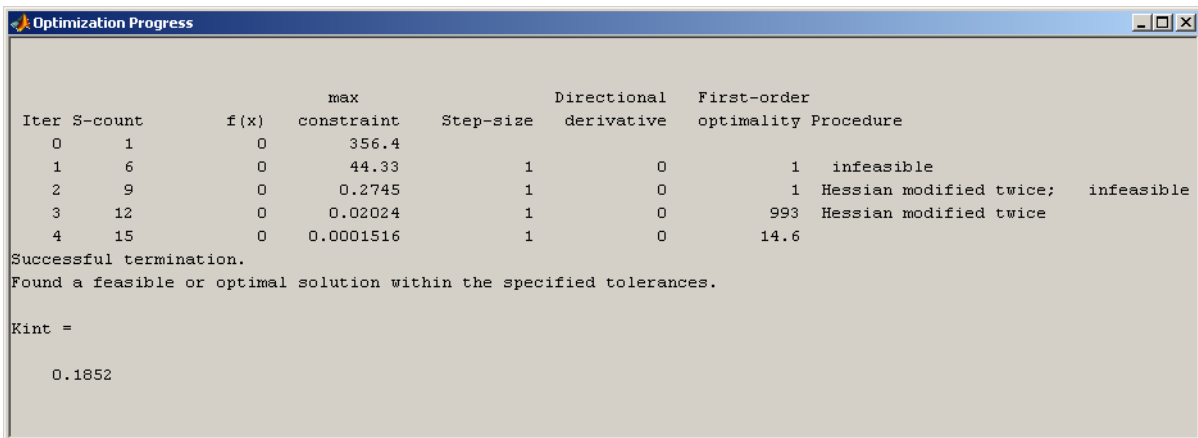


To display the optimization settings for this parameter, click on **Kint** in the **Tuned parameters** list. The settings appear under **Optimization settings** on the right. To constrain **Kint** to be positive, enter 0 as the **Minimum** value. For more information on the various tuned parameter settings, see the online documentation.

Running the Optimization

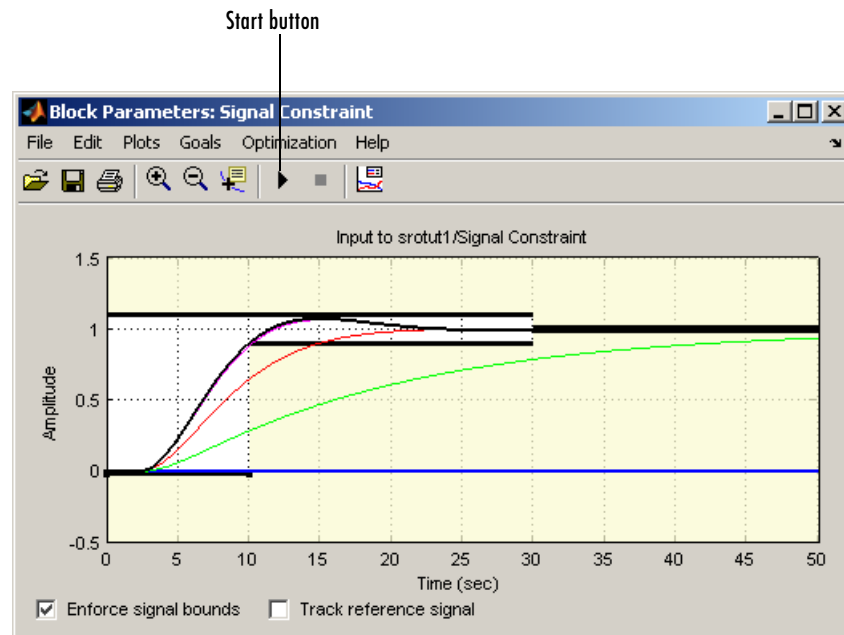
After adjusting the constraint bounds in the **Signal Constraint** window and specifying the tuned parameters using the **Tuned Parameters** dialog, you are ready to begin the optimization. You can start an optimization by clicking the **Start** button in the **Signal Constraint** window or by selecting **Optimization -> Start**.

Simulink Response Optimization automatically converts the constraint bound data and tuned parameter information into a constrained optimization problem that it solves using functions from the Optimization Toolbox or the Genetic Algorithm and Direct Search Toolbox. It attempts to satisfy the constraints on the response signals by adjusting the tuned parameters. The results of each iteration appear in the **Optimization Progress** window shown in the following figure. The number of iterations necessary for the optimization to converge or terminate, will depend on the initial guess for the tuned parameters, the specific positioning of the constraints, and the optimization settings.



In this case the optimization converges after four iterations. For more information about the results shown in the **Optimization Progress** window, see the online documentation.

The **Signal Constraint** window, shown in the following figure, plots the responses at each iteration.



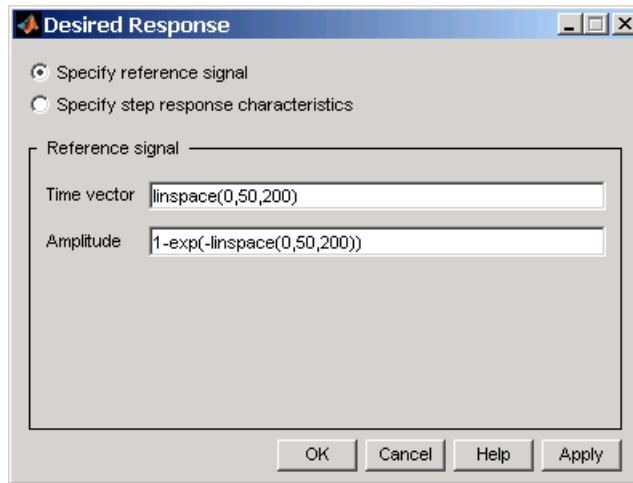
The blue line shows the initial response and the black line shows the current, or final, response. As you can see, the final response signal lies within the constraint bounds.

The new value of the tuned parameter K_{int} appears in the **Optimization Progress** window and is also changed in the MATLAB workspace. In this case the new value of K_{int} is 0.1852

Because of different numerical precision, the results of the optimization may differ slightly across different platforms.

Tracking a Signal

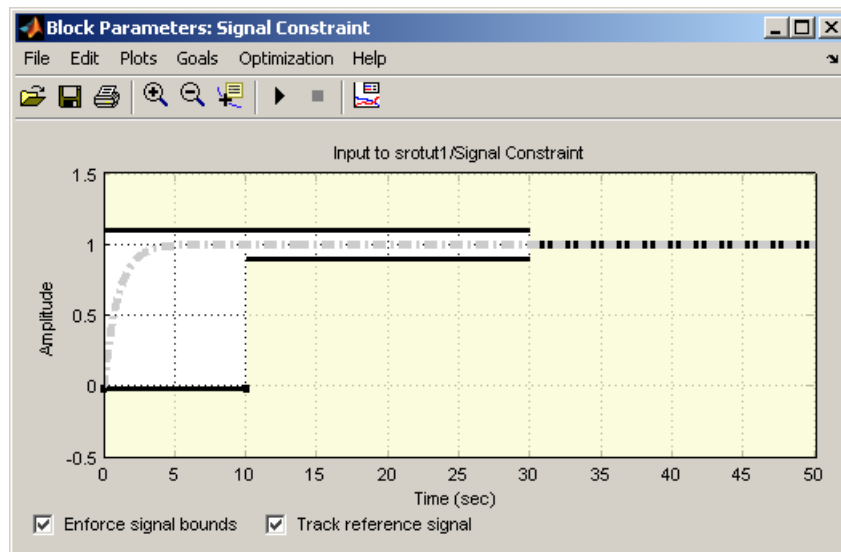
In addition to constraining the signal by constraint bounds, you can constrain the signal by requiring it to closely match a reference signal. To do this, select the **Track reference signal** option at the bottom of the **Signal Constraint** window. To enter the reference signal, select **Goals -> Desired Response**. This displays the **Desired Response** dialog, as shown in the following figure.



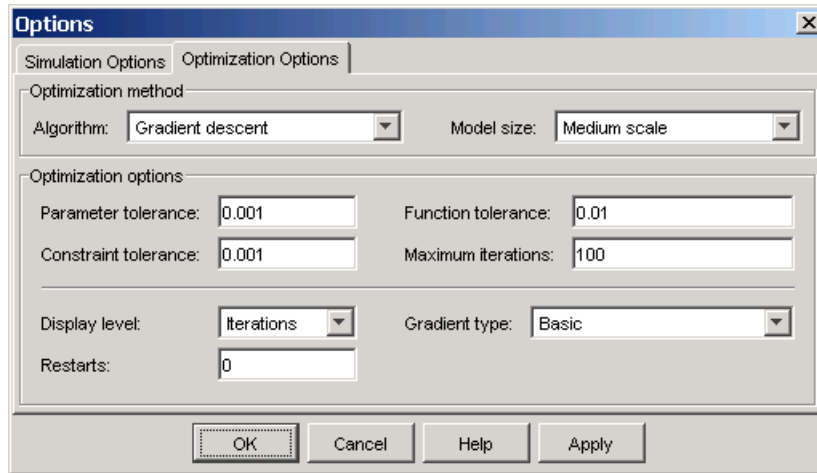
For this example, use the reference signal given by

$$y = 1 - e^{-t}$$

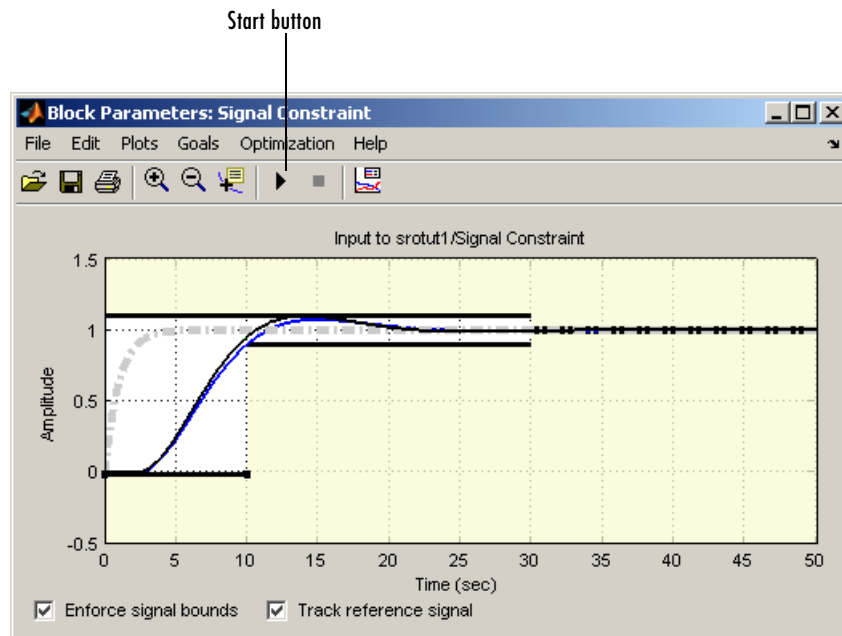
Enter time and amplitude vectors for the reference signal in the **Desired Response** dialog as shown above, and then click **OK**. This displays the following plot in the **Signal Constraint** window. Note, to remove the plots of the optimized response signals, as was done in the figure, select **Plots -> Clear Plots**.



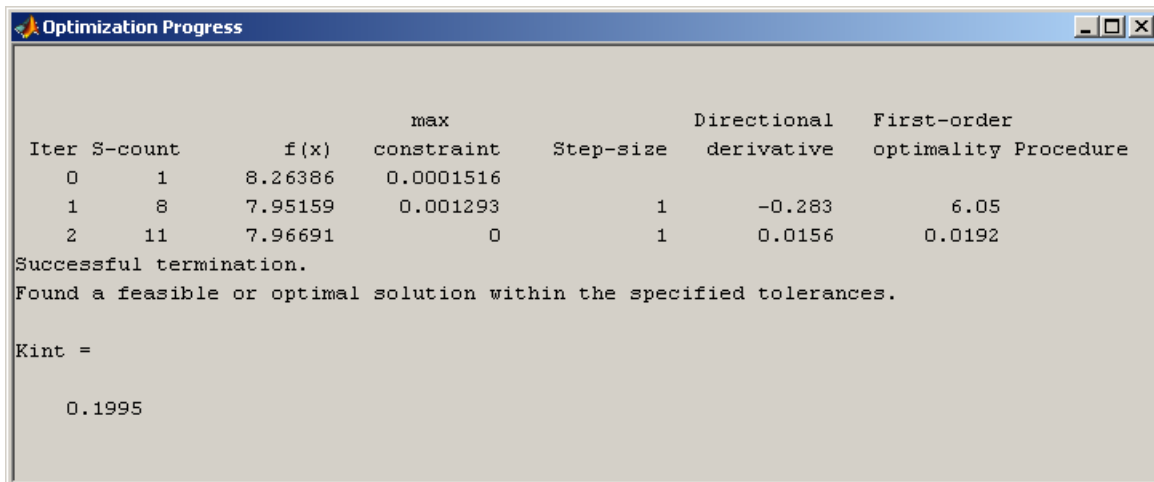
Before running the optimization, change the function tolerance to 0.01 within the **Options** dialog. This ensures that the new objective function meets the imposed tolerances. To do this, open the **Options** dialog by selecting **Optimization -> Optimization Options** from the menu in the **Signal Constraint** window and then change the **Function tolerance** value to 0.01 within the **Optimization Options** pane of the **Options** dialog, as shown in the following figure.



To run the optimization, select **Optimization -> Start** or use the Start button on the toolbar in the **Signal Constraint** window. The results, shown below, are similar to the previous ones when only constraint bounds were used.



The **Optimization Progress** window shows the final value of K_{int} .



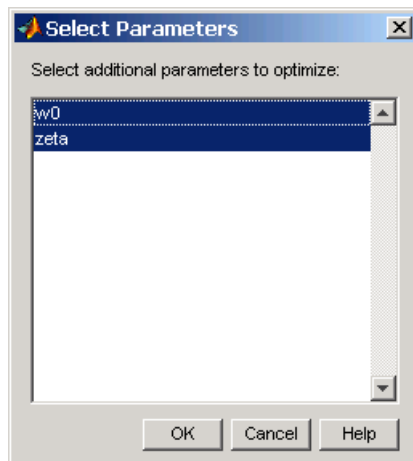
Before proceeding to the next section, make sure to clear the **Track reference signal** option at the bottom of the **Signal Constraint** window.

Adding Uncertainty

In your particular problem, you might not have a precise plant model. 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, assume that the plant parameter zeta varies 5% about its nominal value and w0 varies between 0.7 and 1.45.

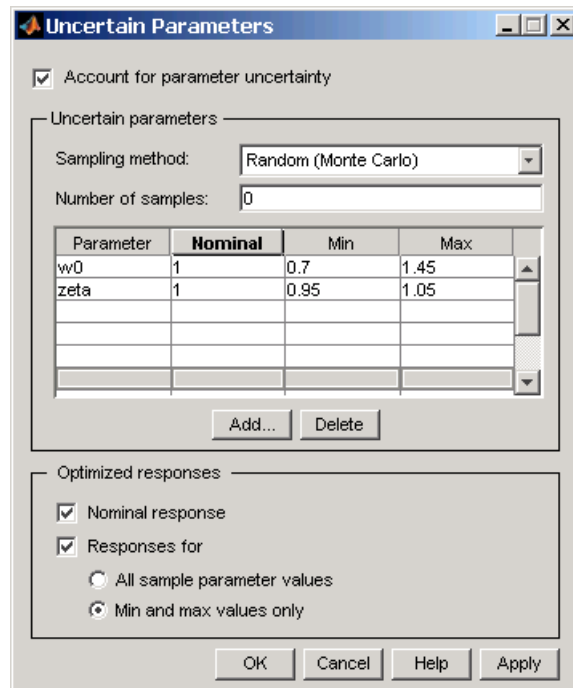
Simulink Response Optimization allows you to optimize response signals in the face of this uncertainty by specifying uncertainty in parameter values.

Open the **Uncertain Parameters** dialog by selecting **Optimization -> Uncertain Parameters** in the **Signal Constraint** window. To add zeta and w0 as uncertain parameters, click the **Add** button, which displays the **Select Parameters** dialog. This dialog contains all parameters from the model that are currently in the workspace and are *not* selected as tuned parameters. Select zeta and w0 in this list, and then click **OK**.



The parameters appear within the **Uncertain Parameters** dialog with their default uncertainty settings. Their nominal values are the current parameter values, and the uncertainty range is 10% on either side of the nominal value.

Change the uncertainty ranges to those given previously: 0.95 to 1.05 for zeta and 0.7 to 1.45 for w_0 . The **Uncertain Parameters** dialog should now look like that in the following figure.



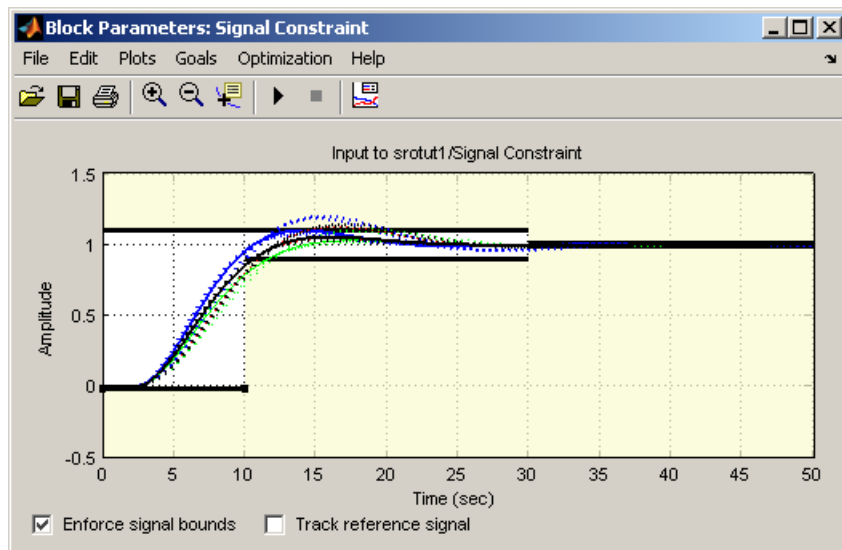
In this example, the sampling method for uncertainty is Random (Monte Carlo). The other choice is Grid in which case you would enter a vector of sample values for each parameter instead of minimum and maximum values. The number of samples indicates how many parameter value combinations to use other than the nominal, minimum, and maximum values.

By default, Simulink Response Optimization optimizes the nominal response along with all responses corresponding to combinations of minimum and maximum values of w_0 and zeta (five responses total). To speed up to optimization you can optimize the nominal response only. To include more uncertainty, you can optimize all sample parameter values. Select which responses to optimize in the **Optimized responses** section of the **Uncertain Parameters** dialog. If you choose not to optimize all uncertain responses, you can still use them for analysis by plotting them in the **Signal Constraint**

window after the optimization. For more information on specifying uncertainty in your models, see the online documentation.

Click **OK** to save your uncertain parameter information, and then run the optimization again.

Simulink Response Optimization plots the uncertain responses as dashed lines. This time, the optimization does not find a feasible solution due to the uncertain response signals violating the constraints as shown in the following figure.



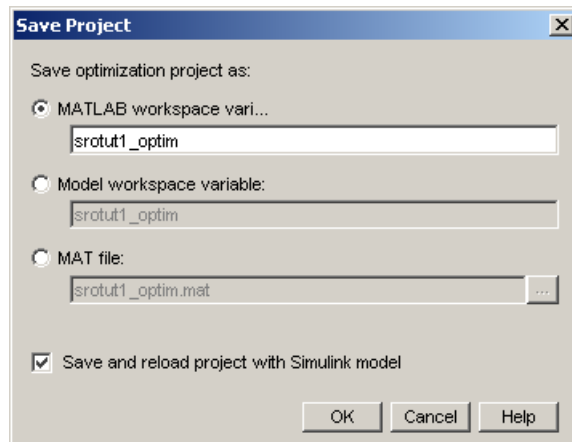
To get the optimization to converge, you might try a different initial guess, relax the constraints slightly, or change the uncertain parameter values. If the uncertain response signals do not violate the constraints by a large amount, the result might be acceptable for your design needs.

To turn off the uncertain response plots, right-click within the figure axes and select **Show -> Uncertainty**. Turn them back on again by repeating this step. To clear all plots, right-click within the figure axes and select **Clear plots**.

Saving the Project

A response optimization project includes the constraints (from all **Signal Constraint** windows in the model), tuned parameter settings, uncertain parameter settings, and settings for the optimization and simulation. After creating a project, you can save it for later use.

To save this project, select **File -> Save** in the **Signal Constraint** window. This opens the **Save Project** dialog as shown below.



You can save the project to either a MATLAB workspace variable, model workspace variable, or a MAT-file. By selecting the **Save and reload project with Simulink model** option, the project will automatically reload when you reopen the model. If the model cannot find the saved project when you reopen it, it gives a warning.

For more information on saving and reloading projects, see the online documentation.

Physical Modeling Example

The following example demonstrates further techniques for using Simulink Response Optimization. It uses the model `srotut2` to show how to constrain more than one signal and tune multiple parameters.

Specifically, you will design a hydraulic system that extends a piston by 0.07m while keeping the pressure from the pump below 12×10^5 Pa. The physical system consists of a hydraulic pump supplying pressure to a hydraulic cylinder. The cylinder contains a piston, which extends and compresses a spring. This system is modeled in the Simulink model `srotut2`. The system parameters that you can adjust, or tune, to achieve the required design characteristics are the cross-sectional area of the piston, A_c , and the maximum pump flow-rate, Q_{max} .

The specifications for the response of the piston position signal are

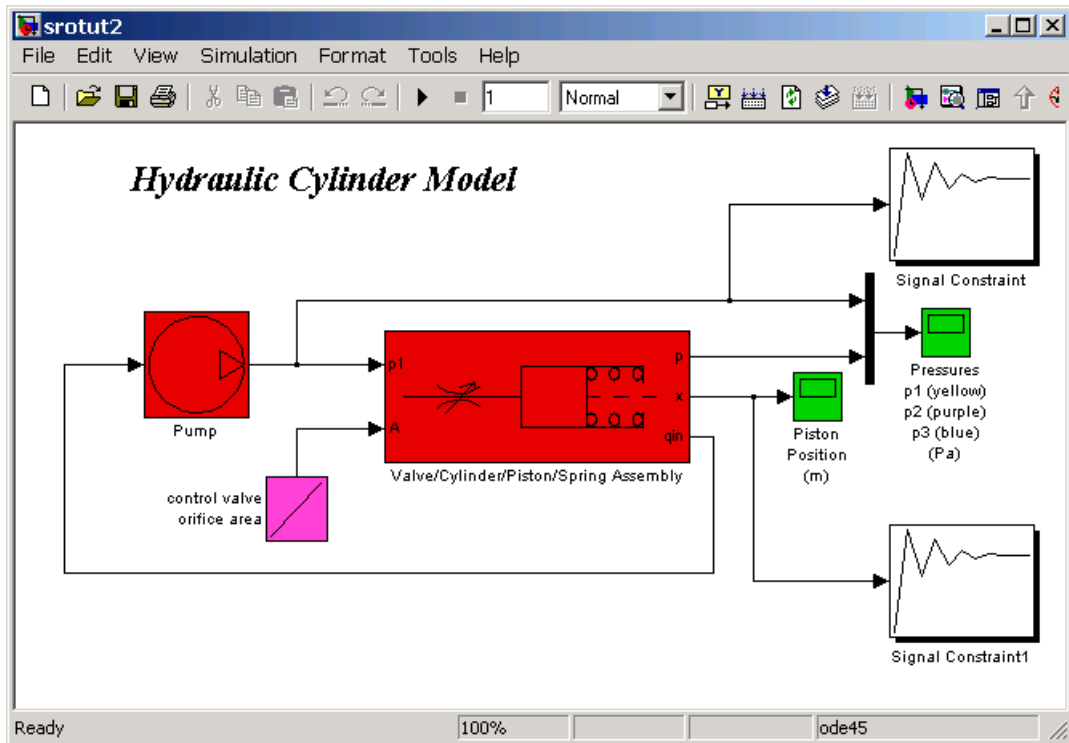
- Maximum overshoot of 14%
- Rise time of 25 seconds
- Settle to within 14% of final value after 25 seconds
- Final value of 0.07m

The specifications for the response of the pump pressure signal are

- Lower bound of zero (Pressures below this value are not physically possible.)
- Upper bound of 12×10^5 Pa

Opening the Hydraulic Cylinder Model

The Simulink system `srotut2` contains a block diagram representation of the hydraulic system described above. You can open the system by typing `srotut2` at the MATLAB prompt.



Notice that the system contains two Signal Constraint blocks. These blocks define constraints on the piston position and pump pressure signals. Simulink Response Optimization allows you to constrain multiple responses including responses within subsystems of your model. For each signal that you want to constrain, you need to attach a Signal Constraint block. Within each Signal Constraint block, set constraint bounds and/or reference signals for each signal. The tuned and uncertain parameters, along with optimization and simulation options are set for the *whole system* within any one of the Signal Constraint blocks.

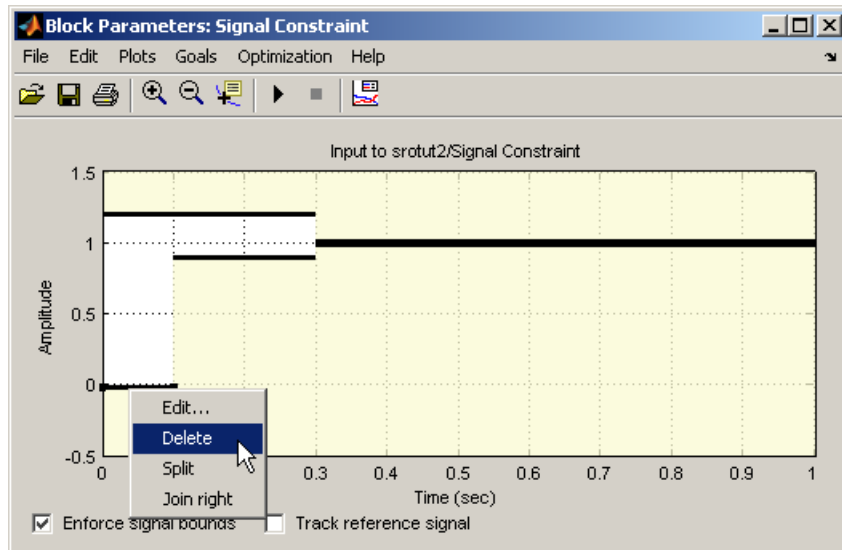
Adjusting Constraints

In this section you will adjust the signal constraints in the system.

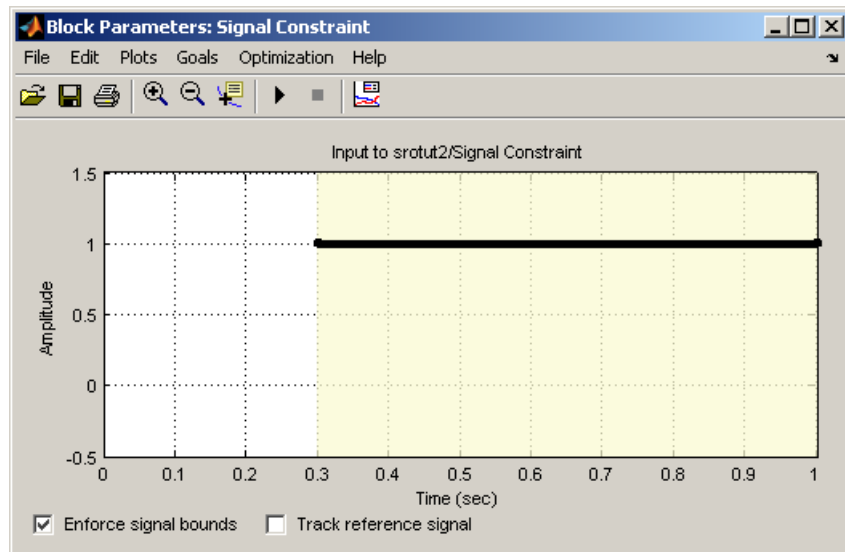
Adjusting Pump Pressure Signal Constraints

Double-click the block labeled Signal Constraint, in the upper-right corner of the model diagram. This block defines the constraints on the pump pressure signal.

First, delete some constraint bound segments so that there is just a single upper bound and a single lower bound. To do this, right click the left-most lower segment and select **Delete** from the menu.

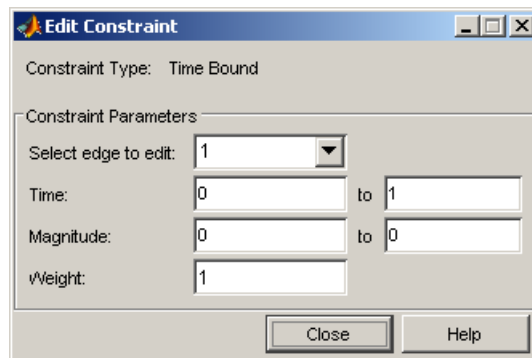


Repeat for one additional lower segment and one upper segment. The figure window should now look something like that in the following figure. Note that the thick black constraint line is actually two constraint edges positioned very close to each other.



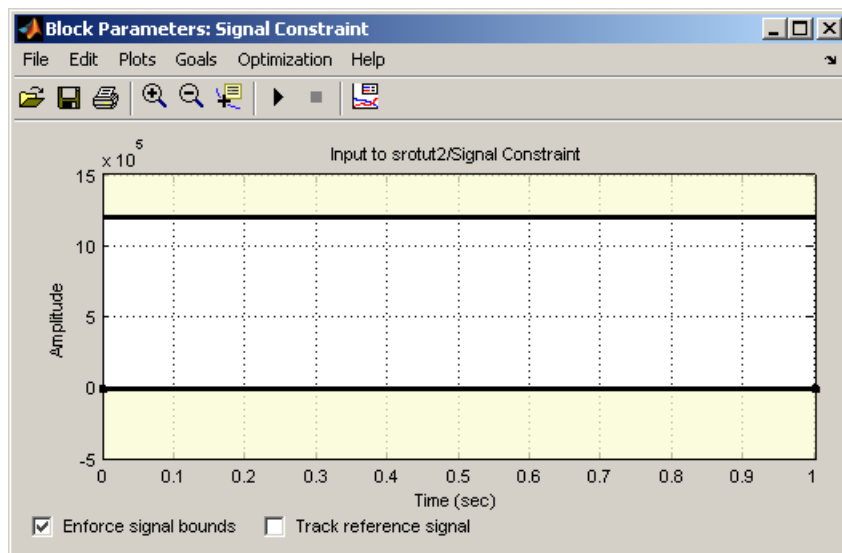
Reposition the constraint bound segments using the **Edit Constraint** dialog. To move the lower constraint, right-click anywhere on the lower constraint segment and select **Edit** from the menu. Within the **Edit Constraint** dialog, enter the start and end time of the constraint edge (0 and 1 respectively) as well as the magnitude of the constraint edge (since it should lie flat, both ends have the same magnitude of 0).

The following figure shows the **Edit Constraint** dialog with the position settings for the lower constraint segment.



Repeat for the upper constraint, moving the constraint edge so that it begins at 0, ends at 1, and has a magnitude of $12e5$. See the online documentation for more information on the **Edit Constraint** dialog.

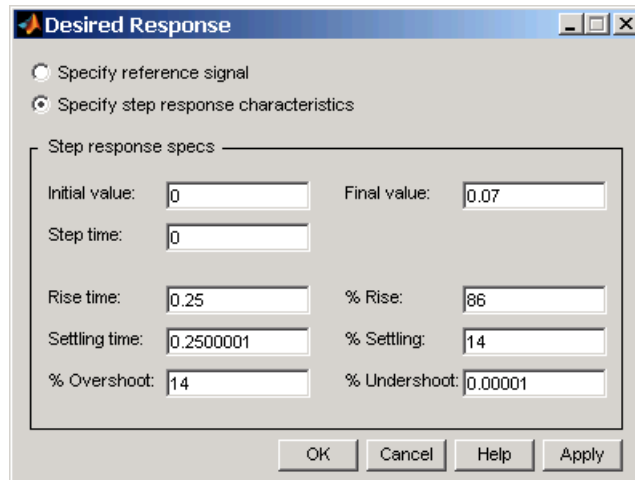
The new constraints are positioned at the edge of the axes and cannot be seen. To adjust the axes so that the constraints lie within the axes limits, select **Edit -> Axes Properties** in the **Signal Constraint** window. Change the **Y-limits** to $-5e5$ and $15e5$. The **Signal Constraint** window should now look like that in the following figure.



Adjusting Piston Position Signal Constraints

Double-click the block labeled Signal Constraint1, in the lower-right corner of the model diagram. This block defines the constraints on the piston position signal.

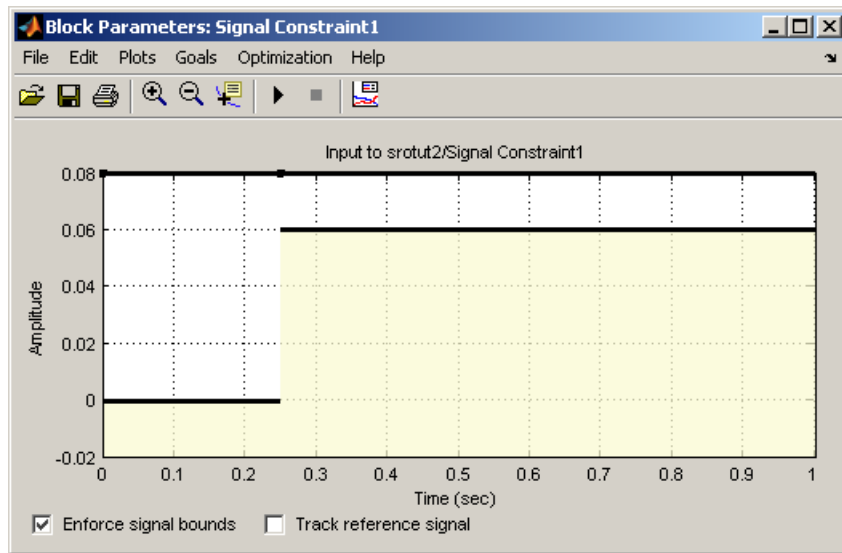
Reposition the constraint bound segments using the **Desired Response** dialog. Open the dialog by selecting **Goals -> Desired Response** from the **Signal Constraint** window. Within the **Desired Response** dialog, select **Specify step response characteristics**. The list at the beginning of “Physical Modeling Example” on page 2-20 specifies the step response characteristics. Enter the information shown in the following figure, and then click **OK**.



These step response characteristics require that

- The signal must settle in approximately the same time it rises.
- After 25 seconds, the signal must rise to 86% of the final value (or 0.06m).
- The signal must settle to within 14% of the final value (or between 0.06m and 0.08m).
- The signal has a maximum overshoot of 14% (or up to 0.08m).
- The signal must have almost no under shoot.

The constraint-bound segments should now look like those in the following figure.



See the online documentation for more information on step response settings.

Specifying Tuned Parameters

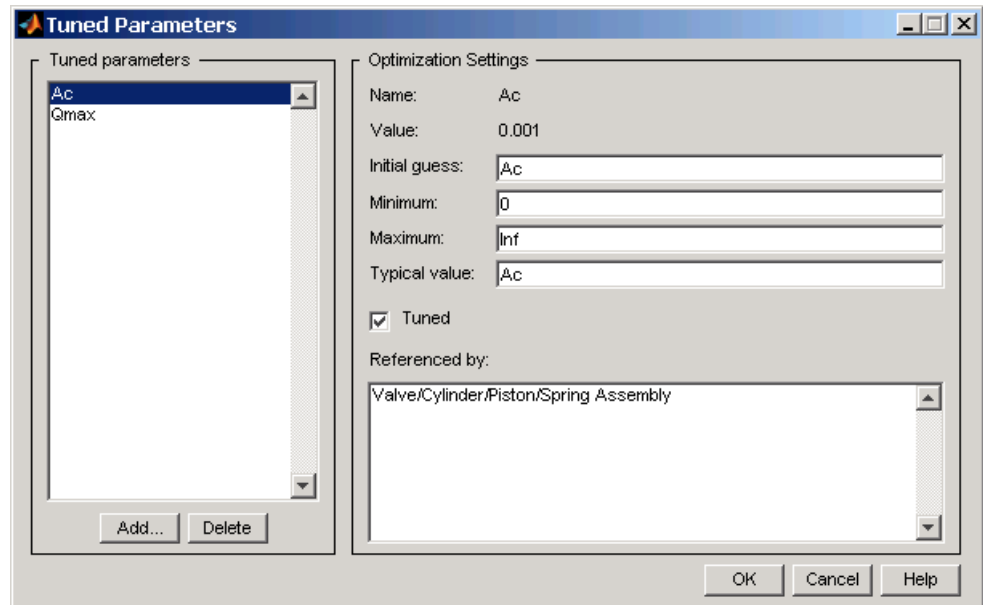
To optimize the signal responses in this example, you will tune two parameters: the cross-sectional area of the piston, A_c , and the maximum pump flow rate, Q_{max} . Although there are two constrained signals and two tuned parameters, you could have any number of tuned parameters in your response optimization.

You can specify tuned parameters within any Signal Constraint block in the model; the tuned parameter settings are for the *whole system*, not just for a particular optimized signal.

To specify the tuned parameters for this example, open any Signal Constraint block in the model and select **Optimization -> Tuned Parameters**. Within the **Tuned Parameters** dialog, click **Add**, and then select A_c and Q_{max} from the list in the **Add Parameters** dialog. Hold down the **Ctrl** key to select multiple parameters from the list. Click **OK** to continue.

In the **Tuned Parameters** dialog, enter specifications for each tuned parameter by selecting the parameter in the list on the left, and then entering information such as initial guesses, minimum values, and maximum values on

the right. For A_c , enter 0 as a **Minimum** value since it does not make sense for the cross-sectional area to be negative. Similarly, enter 0 as a **Minimum** value for Q_{max} to avoid negative flow rates, which imply that the flow reverses direction through the system.

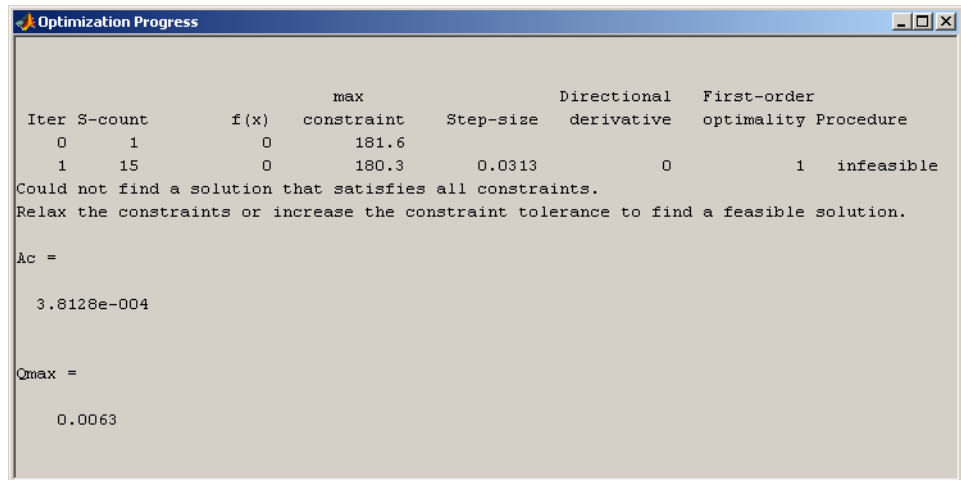


See the online documentation for more information on tuned parameter settings.

Running the Optimization

After adjusting the constraints and defining tuned parameters, start the optimization by selecting **Optimization -> Start**, or by clicking the Start button below the menu bar in a **Signal Constraint** window.

In this case the optimization algorithm is not able to find a solution that satisfies all the constraints, as seen in the **Optimization Progress** window below.



```
Optimization Progress

Iter S-count      f(x)      max      constraint      Step-size      Directional      First-order
                                derivative      optimality Procedure
0         1         0         181.6
1        15         0         180.3         0.0313         0         1  infeasible

Could not find a solution that satisfies all constraints.
Relax the constraints or increase the constraint tolerance to find a feasible solution.

Ac =

3.8128e-004

Qmax =

0.0063
```

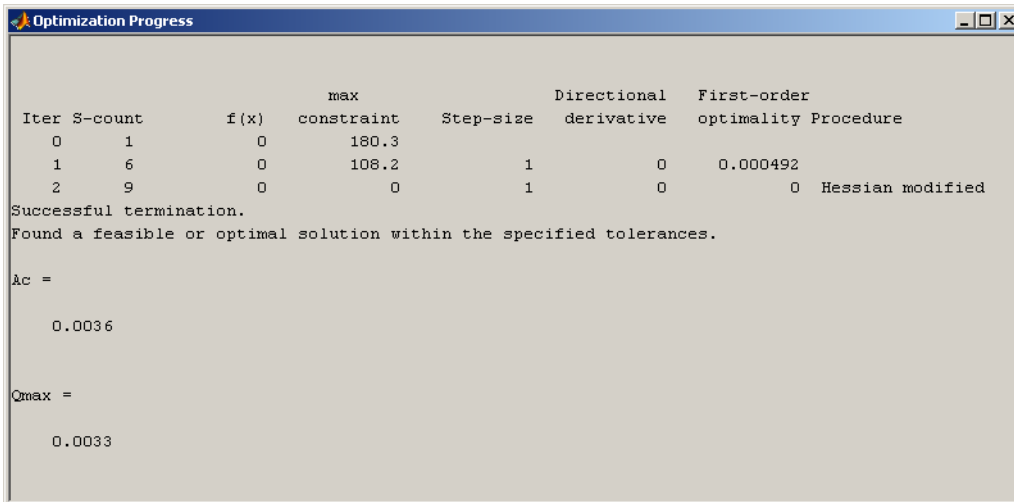
Changing Optimization Settings

When the optimization algorithm does not find a successful solution, you can often find a better solution by changing some of the optimization settings and running the optimization again. Useful things to try are

- Changing the **Gradient type** to **Refined**. This is more accurate for some models.
- Increasing the tolerances.
- Using a different optimization algorithm.

For this example, change the gradient type to Refined by selecting **Optimization -> Optimization Options** in the **Signal Constraint** window, and then selecting Refined as the **Gradient type**. Click **OK** to save the changes and close the **Options** dialog. See the online documentation for more information on optimization settings.

Run the optimization again with these new settings. The optimization algorithm should now find a feasible solution as shown in the following figure.



```

Optimization Progress
-----
Iter S-count      f(x)      max      Directional      First-order
      constraint Step-size derivative optimality Procedure
0      1          0      180.3
1      6          0      108.2          1          0      0.000492
2      9          0          0          1          0          0 Hessian modified

Successful termination.
Found a feasible or optimal solution within the specified tolerances.

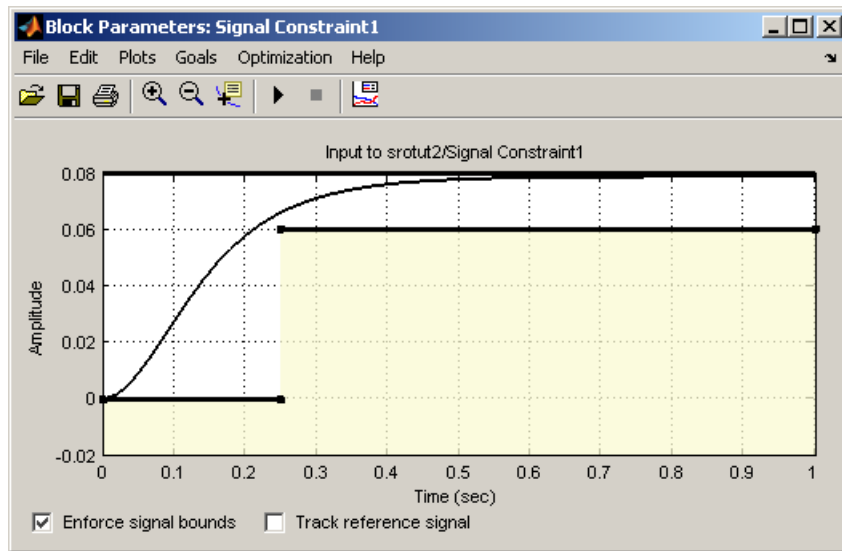
Ac =
    0.0036

Qmax =
    0.0033

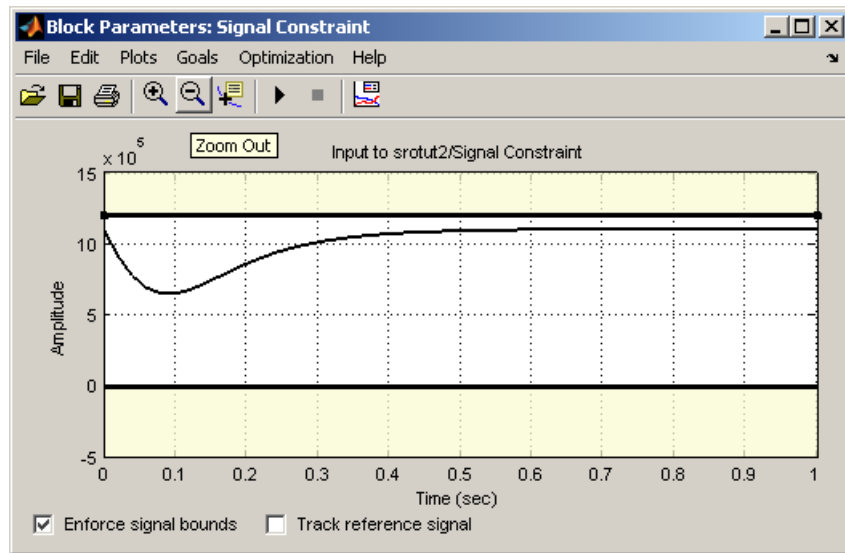
```

The **Optimization Progress** window shows the new optimized parameter values of 0.0036 m^2 for the cross-sectional area of the piston, and $0.0033 \text{ m}^3/\text{s}$ for the pump flow rate. These values are also changed within the MATLAB workspace.

The **Signal Constraint** windows also show that the optimized response signals satisfy both sets of constraints. To see this more clearly, first remove all response signals from the axes by selecting **Plots -> Clear Plots**. Next, plot the current response, based on the optimized parameter values, by selecting **Plots -> Plot Current Response**. The **Signal Constraint** windows should now look like those in the following two figures for piston position and pump pressure, respectively.



Piston Position



Pump Pressure

See Chapter 8, "Troubleshooting" for more advice on adjusting the response optimization to achieve the desired results.

Approaching Response Optimization

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

Choosing Signals to Constrain (p. 3-2)

Inserting Signal Constraint blocks

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

Components of a response optimization project and how to create one

Saving and Reloading Response Optimization Projects (p. 3-4)

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 inport 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 4, “Specifying the Desired Response”
- Chapter 5, “Choosing Optimized Parameters”

To save the project for use in a later session, see “Saving and Reloading Response Optimization Projects” on page 3-4.

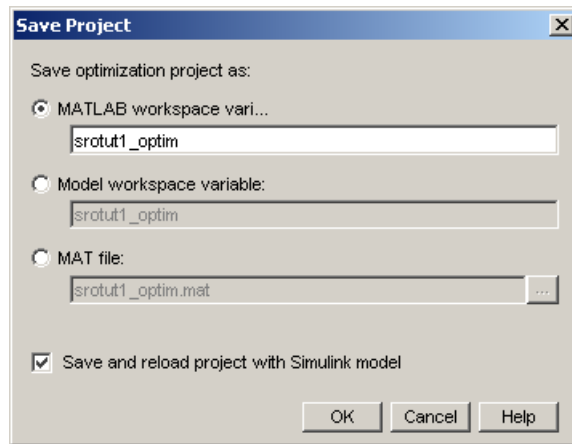
Saving and Reloading Response Optimization Projects

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 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 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

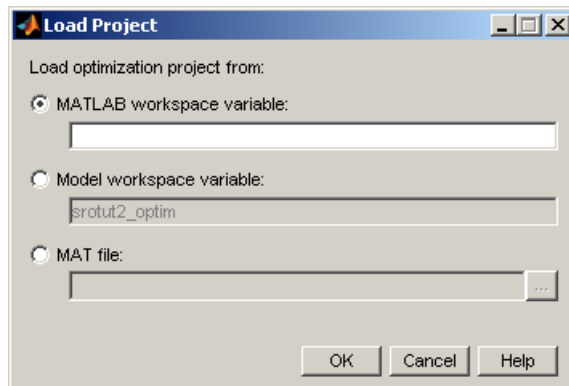
In addition to settings that you save with the response optimization project, there are several other settings that you save with the *model*. These settings include the following:

- 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 filename 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 **Filename** 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. 4-2)

Enforce signal bounds by positioning and editing signal constraints.

Tracking Reference Signals (p. 4-13)

Specify the desired signal response by plotting and tracking a reference signal.

Plotting Responses in the Signal Constraint Window (p. 4-14)

Choose which types of response signals to plot

Response Plots Property Editor (p. 4-15)

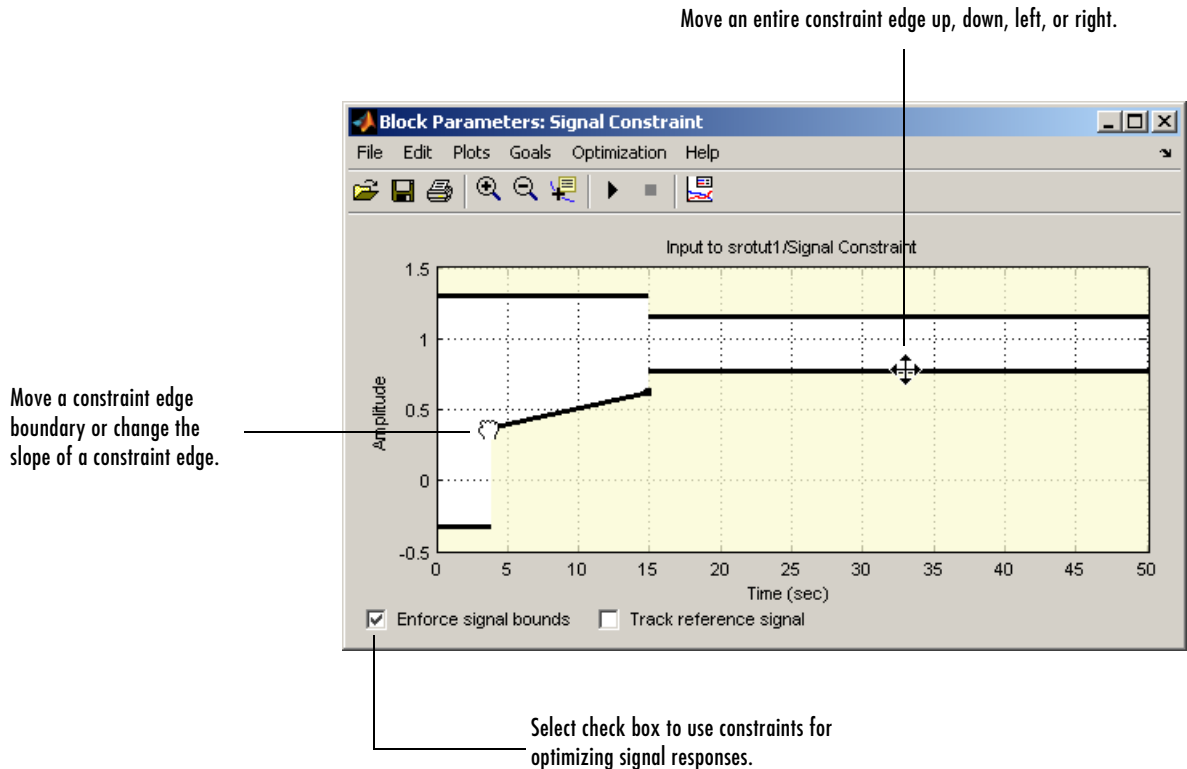
Change axes limits and labels

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.

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.

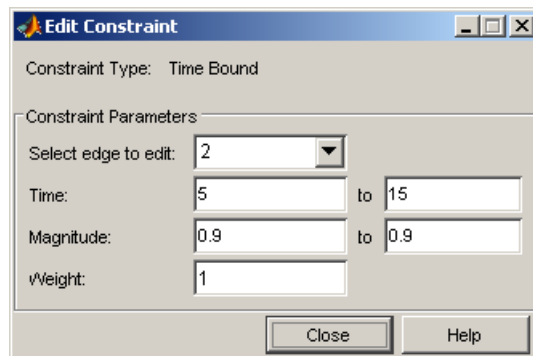
Note It is possible to move a lower bound constraint edge above an upper bound constraint edge, or vice versa, but this will produce an error when you attempt to run the optimization.

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**.

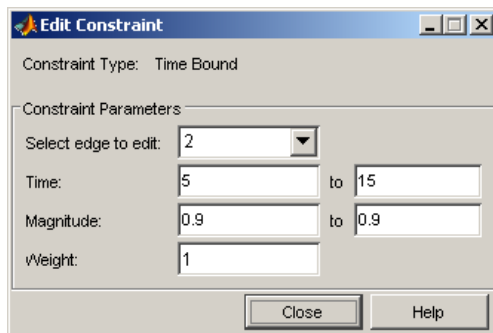
Positioning Constraints Exactly

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 4-5.



Adjusting Constraint Weightings

To change the weight of a constraint edge, position the pointer over the segment you want to weight 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 4-5.

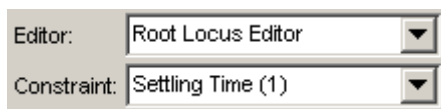


Edit Constraint Dialog

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.



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 and 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.

Table 4-1: 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.

Table 4-1: Edit Constraint Dialog Constraint Parameters

Constraint Parameter	Found in	Description
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. 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.
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.

Table 4-1: Edit Constraint Dialog Constraint Parameters

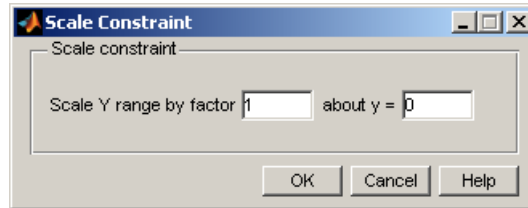
Constraint Parameter	Found in	Description
Natural Frequency	SISO Tool Root Locus Editor	Defines a constraint edge for a particular natural frequency. To specify the constraint, choose at least or at most from the menu, and then specify the natural frequency of interest.
Real	SISO Tool Root Locus Editor	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.
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.

Table 4-1: Edit Constraint Dialog Constraint Parameters

Constraint Parameter	Found in	Description
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

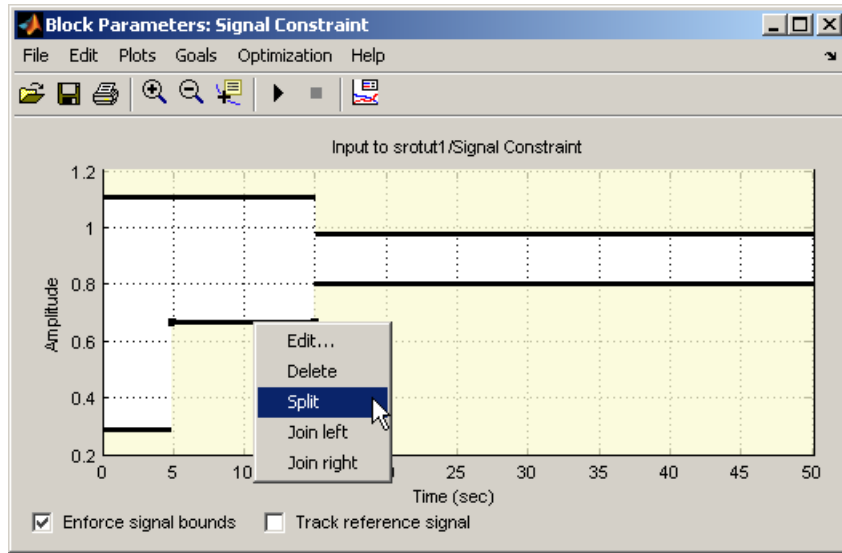
Instead of clicking and dragging the constraints to their new positions, you can scale the constraints. To scale the constraints, select **Edit -> Scale Constraint** in the **Signal Constraint** window. This displays the **Scale Constraint** dialog.



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 the radio button labeled **Specify step response characteristics** to display the step response specifications as shown below.

Desired Response

Specify reference signal

Specify step response characteristics

Step response specs

Initial value: Final value:

Step time:

Rise time: % Rise:

Settling time: % Settling:

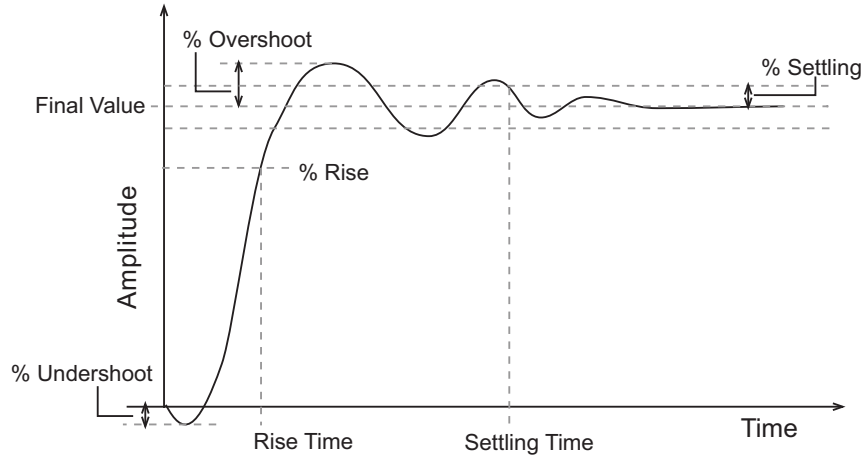
% Overshoot: % Undershoot:

OK Cancel Help Apply

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 certain percentage of the final step value.
- **% Rise:** The percentage of the final step value used in the rise time.
- **Settling time:** The time taken before the response signal remains within certain percentage bounds of the final step value.
- **% Settling:** The percentage of the final step value used in the settling time.
- **% Overshoot:** The percentage of the final step value by which the response signal is allowed to exceed the final step value.
- **% Undershoot:** The percentage of the initial step value by which the response signal is allowed to be less than the initial step value.

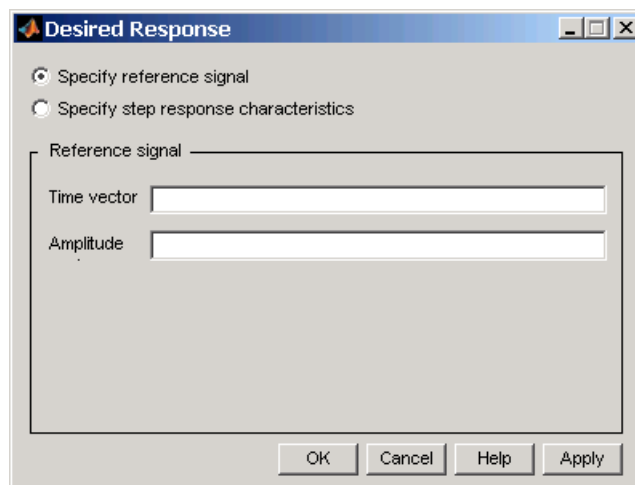
Enter values for the response specifications in the **Response Specifications** dialog, based on the requirements of your model, and then click **OK**. The constraint edges will now reflect the constraints specified.

Tracking Reference Signals

You can specify 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 4-13.

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.

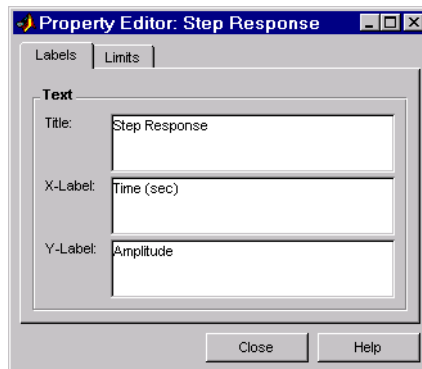
Intermediate Steps

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.



The Property Editor for the 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** 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** box again.

4 Specifying the Desired Response

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.

Specifying Tuned Parameters in the Model (p. 5-2)

Setting the tuned parameters and their characteristics

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

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

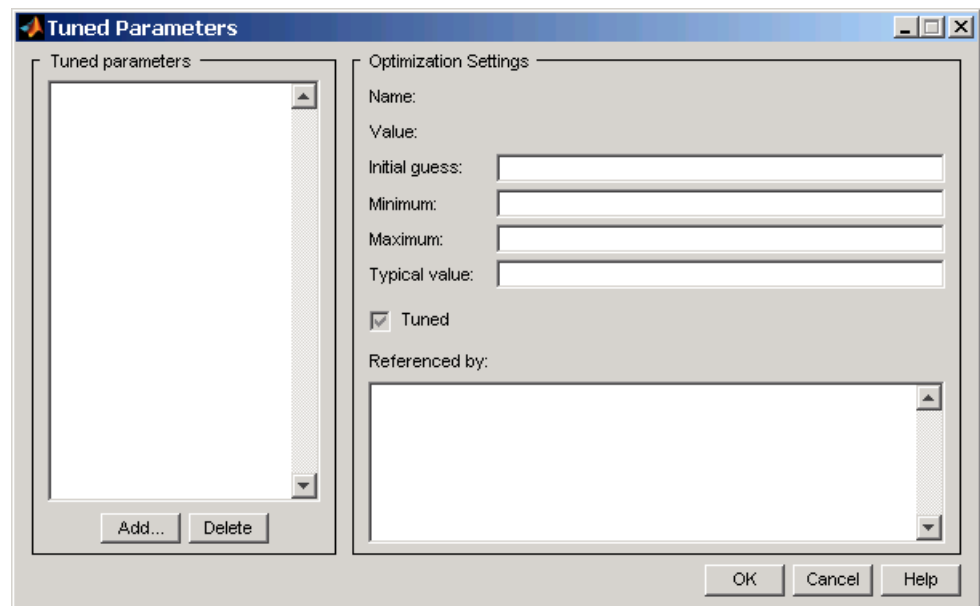
Including Independent Parameters (p. 5-9)

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.

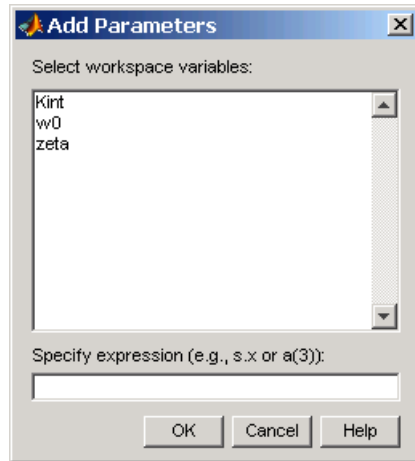
Note When you have more than one Signal Constraint block in your model, you need to specify the tuned parameters in only one window as these settings apply to all constrained signals within the model.



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

Setting	Description	Default
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.
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

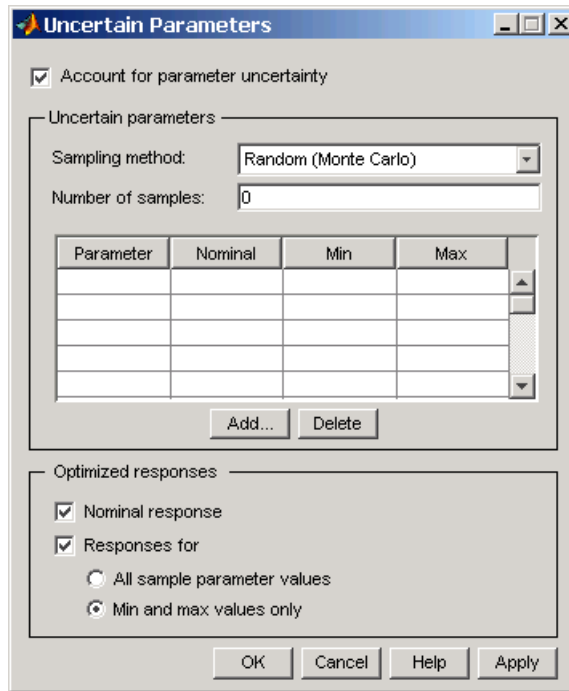
As discussed in “Simple Control Design Example” on page 2-4, a precise plant model might not be known for your particular problem. Instead you might know what the nominal plant should be and have some idea of the uncertainty inherent in various components of the plant. For example, in “Adding Uncertainty” on page 2-16, 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.

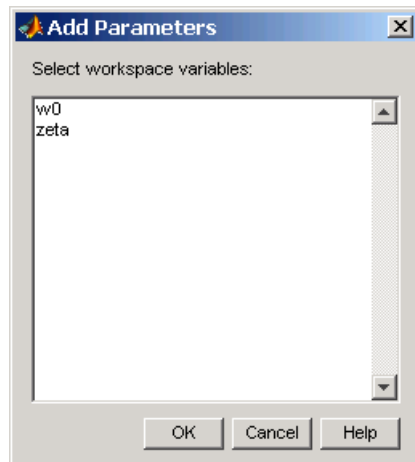
Note When you have more than one Signal Constraint block in your model, you need to specify the uncertain parameters in only one window as these settings apply to all constrained signals within the model.



By default, the **Account for parameter uncertainty** check box is checked when you open the **Uncertain Parameters** dialog. This indicates that you want to include parameter uncertainty in your optimization. By unchecking 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, 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.

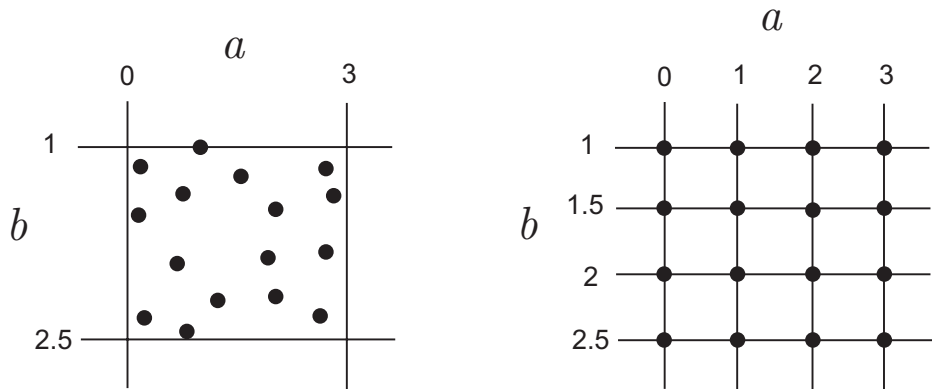
- **Random (Monte Carlo)** computes the optimization at several random parameter values within the range of uncertainty. When this method is selected, you must also enter a value for **Number of samples** which indicates the number of random parameters that Simulink Response Optimization uses. For each 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, check the **Nominal response** check box. To include parameter values other than the nominal value in the optimization, check 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 or Uncertain parameters. Add these parameters to the list of tuned parameters to tune them during the response optimization.

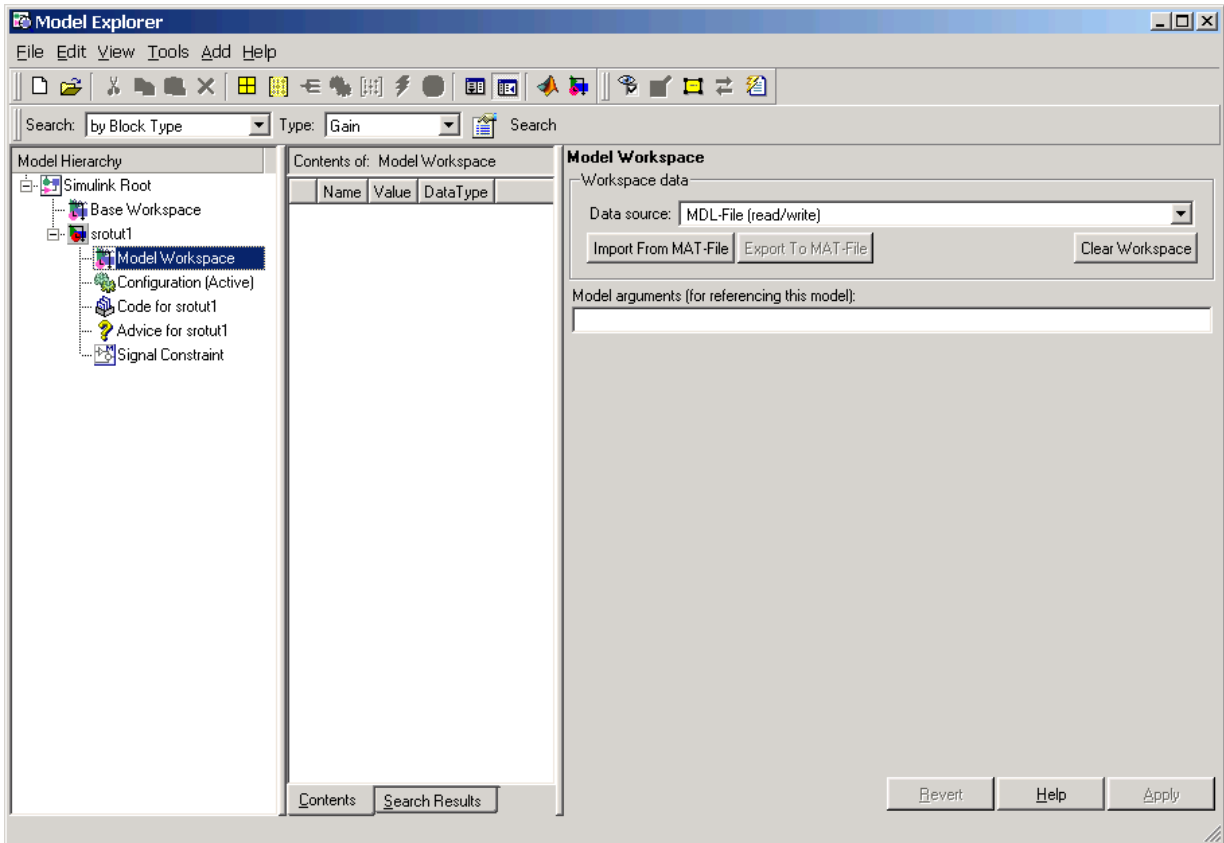
Caution Avoid adding independent parameters together with their corresponding dependent parameters to the lists of tuned and uncertain parameters. Otherwise the optimization could give incorrect results. For example, when a parameter x depends on the parameters a and b , avoid adding all three parameters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

Example:

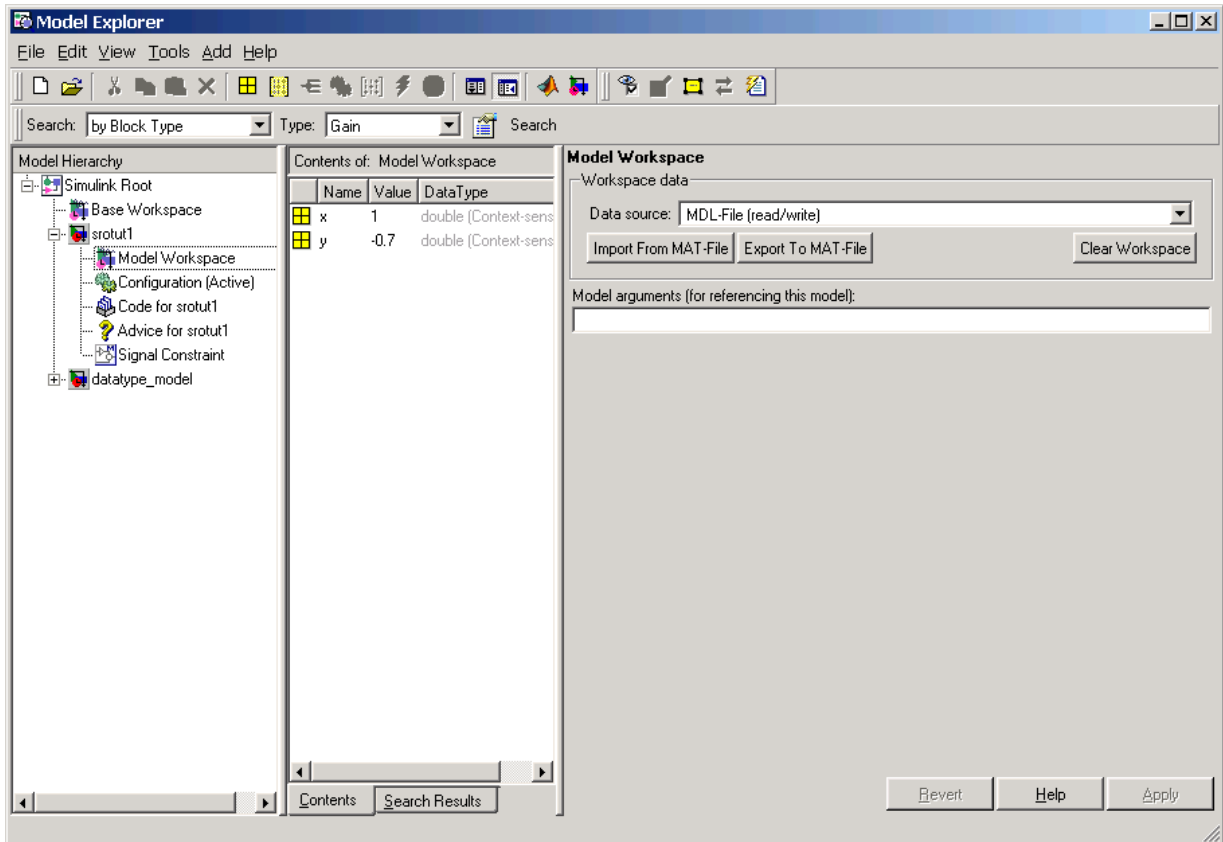
Assume that the parameter K_{int} in the model `srotut1` is related to the parameters x and y according to the relationship $K_{int}=x+y$. Also assume that the initial values of x and y are 1 and -0.7 respectively. To tune x and y instead of K_{int} , first define these parameters in the model workspace.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

5 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.

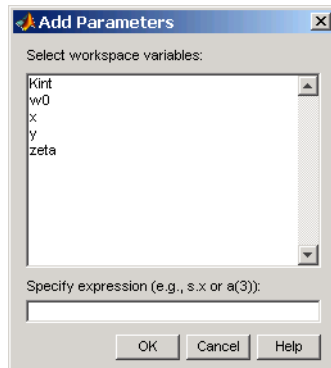


- 5 To add the Simulation Start function defining the relationship between Kint and the independent parameters x and y, select **File -> Model Properties** in the srotut1 window, then select **Callbacks** in the **Model Properties** dialog.
- 6 Under **Simulation start function**, enter the name of a 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.

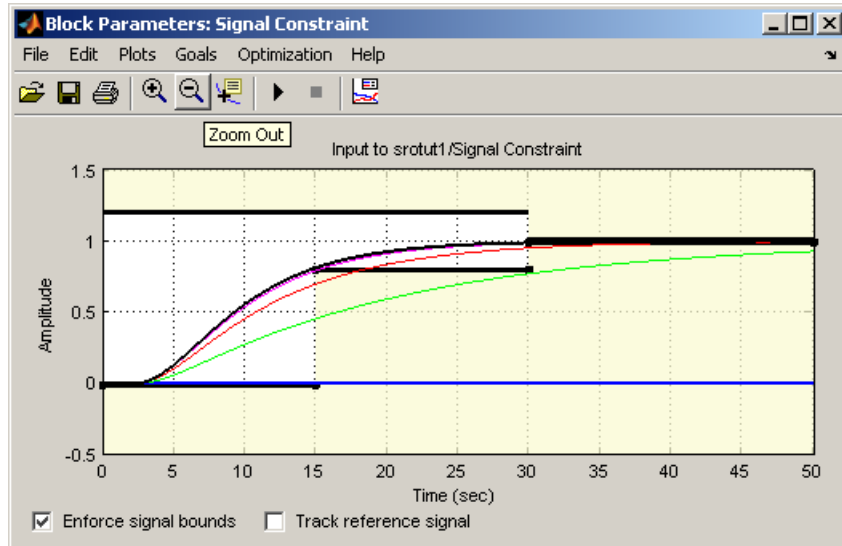
Running the Optimization (p. 6-2)	How to run a response optimization.
Tuning the Optimization Results (p. 6-4)	Description of the various optimization options and suggestions for their use.
Setting Options for the Simulation (p. 6-8)	Description of the various simulation options and suggestions for their use.
Accelerating the Optimization (p. 6-11)	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.

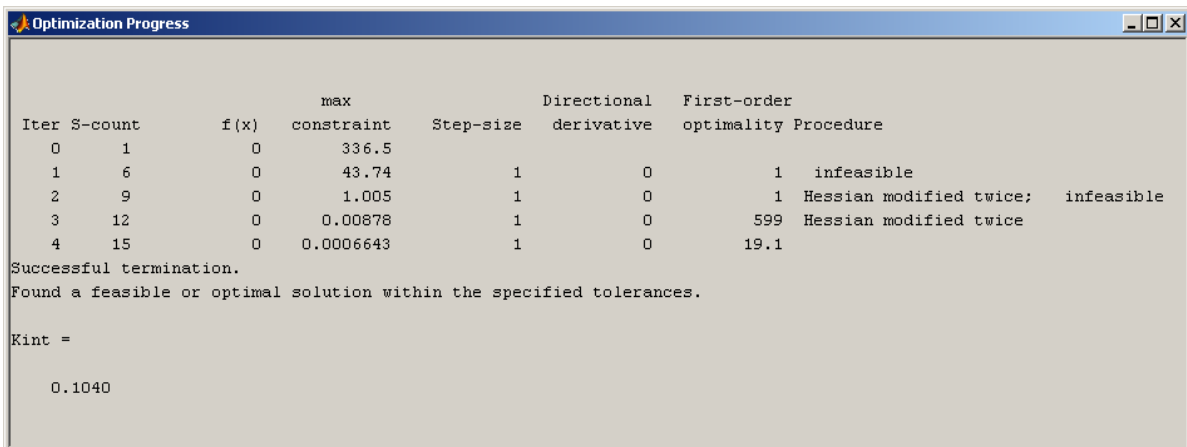


Simulink Response Optimization changes the values of the tuned parameters within the MATLAB workspace and displays the final value in the

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
          constraint  Step-size  Directional  First-order
          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

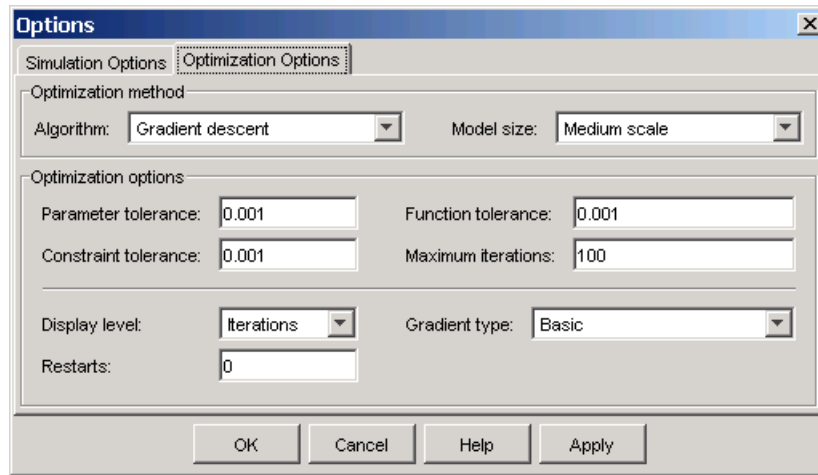
Successful termination.
Found a feasible or optimal solution within the specified tolerances.

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 6-6.

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.



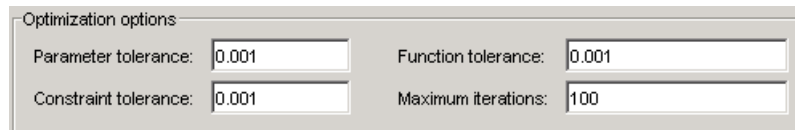
For the **Algorithm** parameter, the three options are

- **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.



Optimization options

Parameter tolerance:	<input type="text" value="0.001"/>	Function tolerance:	<input type="text" value="0.001"/>
Constraint tolerance:	<input type="text" value="0.001"/>	Maximum iterations:	<input type="text" value="100"/>

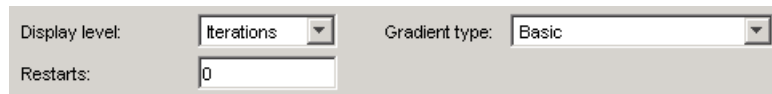
- **Parameter tolerance:** when using the Simplex Search algorithm, the optimization will terminate when successive parameter values change by less than this number. For more details, refer to the discussion of the parameter `TolX` in the reference page for the Optimization Toolbox function `fmincon`.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

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'.

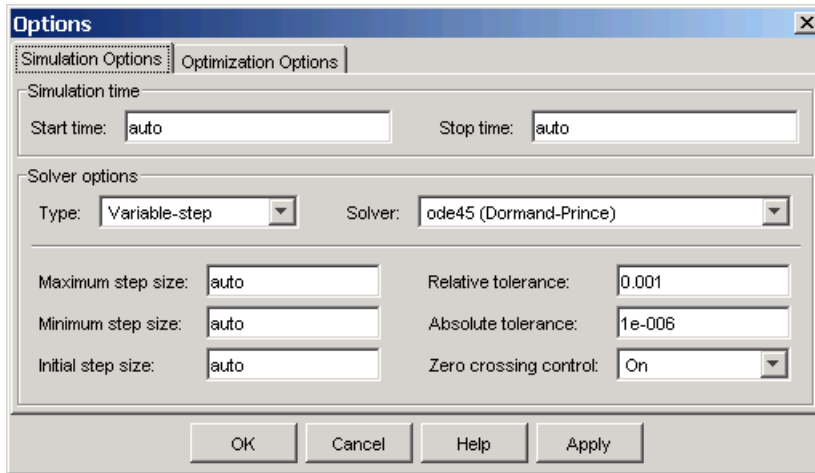
Additional options for optimization include

- **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 end-point 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.

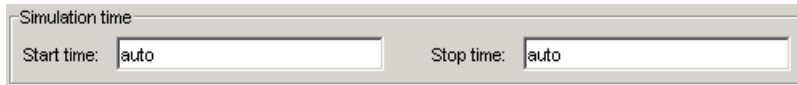
- **Gradient Type:** When using Gradient descent as the **Algorithm**, Simulink Response Optimization calculates gradients based on finite difference methods. The default method for computing the gradients is Basic. The Refined method offers a more robust and less noisy gradient calculation method than 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, enter them under **Simulation Time**.

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

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 step-size 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 non-smooth 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.

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.

Response Optimization Using Functions

In addition to the graphical user interface, Simulink Response Optimization has a command-line interface that lets you use functions to optimize the responses of signals within a Simulink model. This chapter includes an example outlining the use of these functions. A function reference is provided in the online documentation.

Control Design Example Using
Functions (p. 7-2)

An example outlining the response optimization process
using functions.

Control Design Example Using Functions

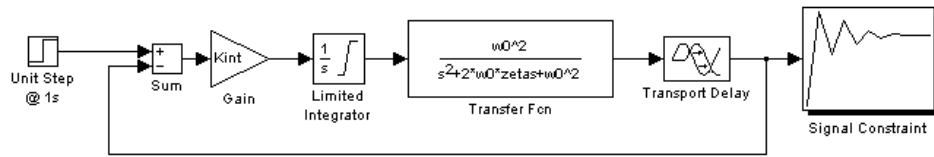
This example uses functions to optimize the response of a signal in the model `srotut1`. For a more detailed description of these functions, refer to the “Function Reference” in the online documentation.

Choosing Signals to Constrain

Open the Simulink model `srotut1` by typing

```
srotut1
```

at the MATLAB prompt. The model opens and should look like that in the following figure.



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. In `srotut1` there is already a Signal Constraint block attached to a signal. Refer to the online documentation for more information on the Signal Constraint block.

Creating an Optimization Project

After attaching Signal Constraint blocks to the appropriate signals within your Simulink model, you need to create a Simulink Response Optimization project. This project is an object that contains information about the response optimization. There are two options for creating the project object:

- To create a default project with all project properties set to the default settings, use the `newsro` function.
- To create a project based on the current response optimization settings in the model, use the `getsro` function.

Creating a Default Project

Create a new project with default settings using the following command.

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

The first input to the `newsro` function is the model name. The second input is a cell array of the tuned parameters. In this case `Kint` is the only tuned parameter.

This returns the following object.

```
Name: 'srotut1'  
Parameters: [1x1 ResponseOptimizer.Parameter]  
OptimOptions: [1x1 ResponseOptimizer.OptimOptions]  
Tests: [1x1 ResponseOptimizer.SimTest]  
Model: 'srotut1'
```

```
Simulink Response Optimization Project.
```

Creating a Project Based on Current Response Optimization Settings

Create a new project based on the current response optimization settings in the model using the following command. This is useful when you previously saved a project for the model and would like to optimize this project at the command line or when you have set the project up with the graphical interface and would like to continue the analysis at the command line.

```
proj2=getsro('srotut1')
```

This command returns a project object with the same properties as the object returned with `newsro`, although these properties will possibly have different current values. The model must already be open to use the `getsro` function.

Properties of a Response Optimization Project

Within the project object you can set characteristics of the tuned parameters, specify uncertain parameters, position constraint bound segments, and set options for the optimization and simulations.

Specifying Tuned Parameter Attributes

To specify bounds and initial guesses for the tuned parameters, first extract the tuned parameters from the project object with the `findpar` function.

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

This returns a tuned parameter object.

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

Tuned parameter.

Next, use dot notation to edit these properties to specify maximum values, minimum values, initial guesses, etc. For example, to set the initial guess to 0.4 and the minimum value to 0, use the following commands.

```
param.InitialGuess=0.4;  
param.Minimum=0
```

The new parameter settings are shown below.

```
      Name: 'Kint'  
      Value: 0  
InitialGuess: 0.4000  
      Minimum: 0  
      Maximum: Inf  
TypicalValue: 0  
ReferencedBy: {0x1 cell}  
Description: ''  
      Tuned: 1
```

Tuned parameter.

For more information on specifying tuned parameters, see the online documentation.

Adjusting Signal Constraints

To define the constrain bound segments within which the optimized signal response must lie, first extract a signal constraint object from the project, using the function `findconstr`.

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

This function takes the project object and the location of a Signal Constraint block in the model as inputs and creates the following signal constraint object.

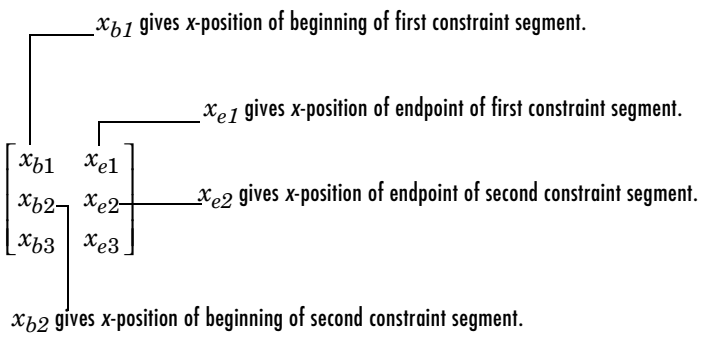
```
ConstrEnable: 'on'  
isFeasible: 1  
CostEnable: 'off'  
Enable: 'on'  
Name: 'Signal Constraint'  
SignalSize: [1 1]  
LowerBoundX: [3x2 double]  
LowerBoundY: [3x2 double]  
LowerBoundWeight: [3x1 double]  
UpperBoundX: [2x2 double]  
UpperBoundY: [2x2 double]  
UpperBoundWeight: [2x1 double] 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

Signal Constraint.

To move the constraints, edit the `LowerBoundX`, `LowerBoundY`, `UpperBoundX`, and `UpperBoundY` matrices. These matrices specify the x and y positions of the endpoints of each segment. For example:

LowerBoundX gives x-positions of all lower bound constraint segments.

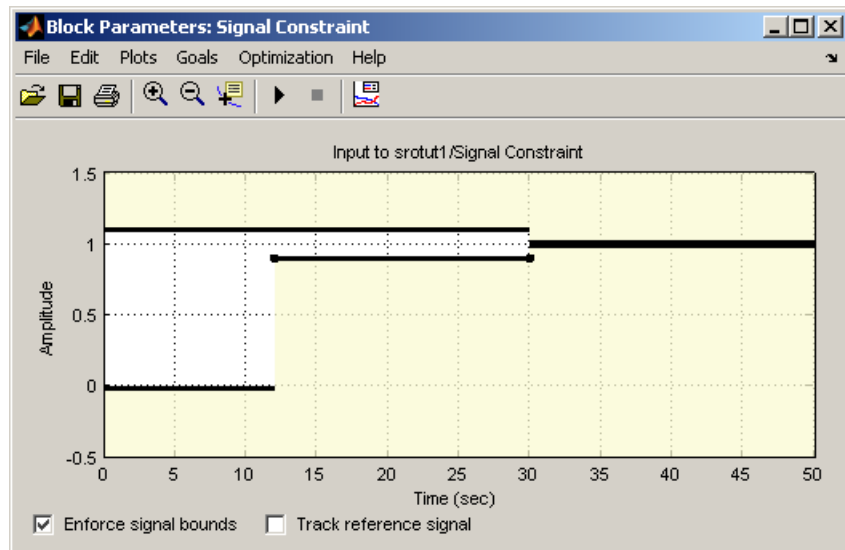
$$\text{LowerBoundX} = \begin{bmatrix} x_{b1} & x_{e1} \\ x_{b2} & x_{e2} \\ x_{b3} & x_{e3} \end{bmatrix}$$



To move the constraint bounds to the positions shown in the following figure, use the following commands.

```

constr.LowerBoundX=[0 12;12 30;30 50];
constr.LowerBoundY=[0 0;0.9 0.9;0.99 0.99];
constr.UpperBoundX=[0 30;30 50];
constr.UpperBoundY=[1.1 1.1;1.01 1.01];
    
```



For more information on constraint bound segments, see the online documentation.

Setting Optimization Options

Before running the optimization, it is sometimes useful to change or set some optimization options. These options define the algorithms and methods Simulink Response Optimization uses to optimize the signals. For a list of all options and their uses, see “Tuning the Optimization Results” in the online documentation or the documentation for the MATLAB function `optimset`.

Get the current optimization options settings with the `optimget` function.

```
optimget(proj)
```

This returns a list of the options 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 the `optimset` function to change any values within the optimization options object. For example, to change the tolerance on the parameter values, `TolX`, to $1e-4$ use the following command.

```
optimset(proj, 'TolX', 1e-4)
```

Setting Simulation Options

In addition to specifying optimization options, it is sometimes useful to specify simulation options. These options define the solvers, simulation times, and other settings that Simulink Response Optimization uses when simulating the models during the response optimization. For a list of all options and their uses, see “Setting Options for the Simulation” in the online documentation or the documentation for the Simulink function `simset`.

Get the current simulation options settings with the `simget` function.

```
simget(proj)
```

This returns a list of the options and their current values.

```
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'
```

To change values of within the simulation options object, use the `simset` function. For example, change the solver to `ode23` with the following command.

```
simset(proj, 'Solver', 'ode23')
```

Specifying Uncertain Parameters

When some parameters in your model are not known exactly, but you do know the nominal plant and the level of uncertainty surrounding this model, you can include this uncertainty in your response optimization by specifying uncertain parameters in your model.

In this example, assume that w_0 varies between 0.8 and 1.2 while ζ varies between 0.95 and 1.05. First, create a set of sample parameter values within these ranges using the `randunc` function.

```
unc_rand=randunc(2, 'w0', {0.8 1.2}, 'zeta', {0.95 1.05});
```

The `randunc` function creates combinations of parameter values based on the endpoints of their uncertainty ranges. In addition, it creates several random parameter value combinations. The first argument to the function specifies the number of random parameter value combinations that the function creates. The remaining input arguments specify the uncertain parameters and the ranges over which they vary. This particular example creates two random combinations of w_0 and ζ values in addition to the combinations of parameter values at the endpoints.

When you would rather specify uncertain parameter values on a grid, use the `gridunc` function.


```
unc_grid=gridunc('w0',{0.8 0.9 1.0 1.1 1.2},'zeta',{0.95 1  
1.05});
```

The `gridunc` function creates combinations of the given parameters at values specified in the cell-arrays of parameter values.

For more information on creating sets of uncertain parameter value combinations, see the reference pages for `randunc` and `gridunc`.

By default, when adjusting the tuned parameters, the response optimization algorithm does not take responses based on these uncertain parameter values into account. To include an uncertain parameter combination in the optimization, you must set its `Optimized` property to `true`.

For example, to include all the parameter combinations within `unc_rand`, enter the following command.

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

After creating the set of uncertain parameter values, and choosing to include some or all of them in the optimization, add these values to the project object with the `setunc` function.

```
setunc(proj,unc_rand)
```

For more information on specifying uncertain parameter values, see the online documentation.

Running the Optimization

To run the optimization for this project, enter

```
optimize(proj)
```

The results appear in the MATLAB window after each iteration. See the online documentation for more information on the optimization results.

During the optimization, Simulink Response Optimization changes the tuned parameter values in the MATLAB workspace. The optimization also displays the new, optimized parameter values after terminating. In this example, the optimized value of `Kint` is 0.1539.

Troubleshooting

Where possible, Simulink Response Optimization provides visual cues to help you formulate problems and inform you about the progress of an optimization. However, sometimes problems can occur when optimizing signal responses. This chapter contains a list of common problems along with recommendations for dealing with them.

Common Questions About Response
Optimization (p. 8-2)

Solutions for commonly encountered response
optimization problems

Common Questions About Response Optimization

The following list of questions represent commonly encountered problems with response optimization. For each question, solutions, advice and tips are given.

How do I quit an optimization and revert to my initial parameter values?

Click the Stop button or select **Optimization -> Stop** in a **Signal Constraint** window to stop the optimization, and then select **Edit -> Undo Optimize Parameters** to revert to your initial parameter values.

The responses and parameter values do not change at all.

- The optimization problem you formulated might be nonsmooth. This means that small parameter changes have no effect on the amount by which response signals satisfy or violate the constraints and only large changes will make a difference. Try switching to a search-based algorithm such as simplex search or pattern search. Alternatively, look for initial guesses outside of the dead zone where parameter changes have no effect. You could also try removing nonlinear blocks such as the Quantizer or Dead Zone block.
- If you are using the Refined option for **Gradient type** with the Gradient descent algorithm, try the Basic option for **Gradient type** instead. The gradient model that the Refined option uses might be invalid for your problem.

The optimization does not get close to an acceptable solution.

- If you're using Gradient descent, the default algorithm, try the Refined option for **Gradient type**. This option yields more accurate gradient estimates when using variable-step solvers and can facilitate convergence.
- If you're using Pattern search, check that you have specified appropriate maximum and minimum values for all your tuned parameters. The pattern search algorithm looks inside these bounds for a solution. When they are set to their default values of Inf and -Inf, the algorithm searches within $\pm 100\%$ of the initial values of the parameters. In some cases this region is not large enough and changing the maximum and minimum values can expand the search region.
- Your optimization problem might have local minima. Consider running one of the search-based algorithms first to get closer to an acceptable solution.

- Reduce the number of tuned parameters by removing from the **Tuned Parameters** list those parameters that you know only mildly influence the optimized responses. Once you identify reasonable values for the key parameters, add the fixed parameters back to the tunable list and restart the optimization using these reasonable values as initial guesses.

The optimization terminates before exceeding the maximum number of iterations, with a solution that does not satisfy all the constraints.

- It might not be possible to achieve your specifications. Try relaxing the constraints that the response signals violate the most. Once you find an acceptable solution to the relaxed problem, tighten some constraints again and restart the optimization.
- The optimization might have converged to a local minimum that is not a feasible solution. Restart the optimization from a different initial guess and/or use one of the search-based methods to identify another local minimum that satisfies the constraints.

The optimization drives the tuned parameters to undesirable values.

- When you know that a tuned parameter should remain positive, or when its value is physically constrained to a given range, enter this information in the **Tuned Parameters** dialog as lower and upper bounds (**Minimum** and **Maximum**). This information helps guide the optimization algorithm toward a reasonable solution.
- In the **Tuned Parameters** dialog, specify initial guesses that are within the range of desirable values.

The optimization is close to a solution but is taking a long time to converge.

- Use the Stop button, or select **Optimization -> Stop**, in a **Signal Constraint** window to interrupt the optimization when you think the current optimized response signals are acceptable.
- If you use the Gradient descent algorithm, try restarting the optimization. This resets the Hessian estimate and might speed up convergence.
- Increase the convergence tolerances in the **Optimization Options** dialog to force earlier termination.

- Relax some of the constraints to increase the size of the feasibility region.

The response signal becomes unstable and does not recover.

While the optimization formulation has explicit safeguards against unstable or divergent response signals, the optimization can sometimes venture into an unstable region where simulation results become erratic and gradient methods fail to find a way back to the stable region. Remedies include

- Add or tighten the lower and upper bounds on parameter values. Instability often occurs when you allow some parameter values to become too large.
- Use a search-based algorithm to find parameter values that stabilize the response signals and then start the gradient-based algorithm using these initial values.

The optimization stalls.

Certain parameter combinations can make the simulation stall for models with strong nonlinearities or frequent mode switching. In these cases the ODE solvers take smaller and smaller step sizes. Stalling can also occur when the model's ODEs become too stiff for some parameter combinations. A symptom of this behavior is when the Simulink model status is Running and clicking the Stop button fails to interrupt the optimization. Remedies include

- Switch to a different ODE solver, especially one of the stiff solvers.
- Specify a minimum step size.
- Disable zero crossing detection if chattering is occurring.
- Tighten the lower and upper bounds on parameters that cause simulation difficulties. In particular, eliminate regions of the parameter space where some model assumptions are invalid and the model behavior can become erratic.

How do I accelerate the optimization?

- Since the time it takes to simulate the model dominates the optimization time, using the Simulink Accelerator can dramatically reduce the optimization time.
- The choice of ODE solver can also significantly affect the overall optimization time. Use a stiff solver when the simulation takes many small steps, and use a fixed-step solver when such solvers yield accurate enough

simulations for your model. (These solvers must be accurate in the entire parameter search space.)

- Reduce the number of tuned parameters and constrain their range to narrow the search space.
- When specifying parameter uncertainty, keep the number of sample values small since the number of simulations grows proportionally with the number of samples. For example, a grid of 3 parameters with 10 sample values for each parameter requires $10^3=1000$ simulations per iteration.

How should I pick the start and stop time?

By default, the start and stop time are inherited from the Simulink model. However, you can change them with the **Simulation Options** dialog. Choose a stop time that captures enough of the desired response's characteristics. When you want the response to settle to a final value, use at least 10-20% of the simulation time for constraining the steady-state response. This ensures the proper weighting of requirements on the final value and overall stability.

Should I worry about the scale of my responses and how constraints are discretized?

No. Simulink Response Optimization automatically normalizes constraint and response data and, unlike its predecessor, the Nonlinear Control Design Blockset, it does not discretize the constraints.

Function Reference

- Functions — By Category (p. 9-2) A list of available functions, sorted by category
- Functions — Alphabetical List (p. 9-3) A list of available functions, sorted alphabetically

Functions – By Category

Response Optimization Projects

<code>getsro</code>	Take snapshot of current optimization project
<code>ncdupdate</code>	Update old NCD Outport blocks
<code>newsro</code>	Create new optimization project for Simulink model
<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>	Define 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>simget</code>	Retrieve current simulation settings
<code>simset</code>	Modify simulation settings
<code>optimget</code>	Retrieve current optimizer settings
<code>optimset</code>	Modify optimizer settings

Functions — Alphabetical List

This section contains function reference pages listed alphabetically.

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 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.

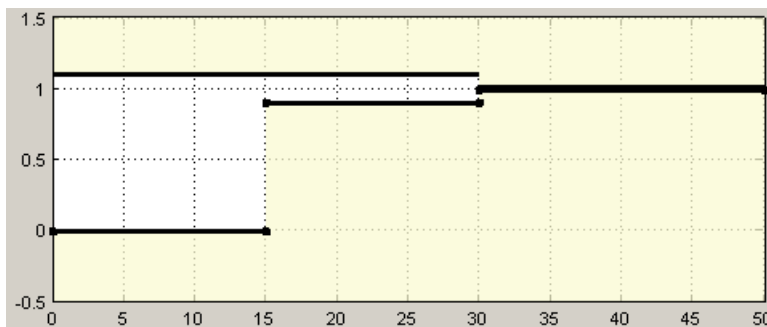
Change the positioning of the constraint bounds by editing the upper and lower bound matrices.

```

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:

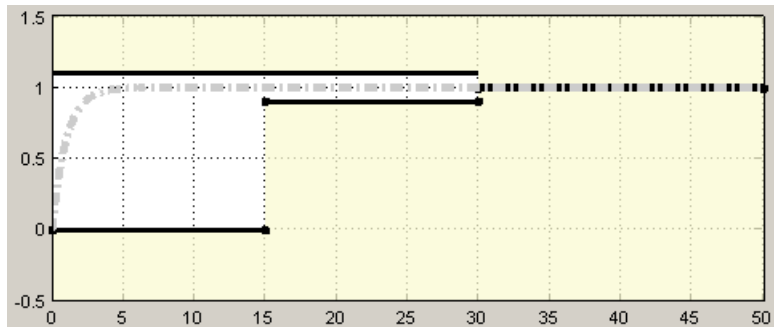
```

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

```

This defines the reference signal shown in the following figure.

findconstr



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.

Example Create a response optimization project for `srotut1`.

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

Find the tuned parameters object for the parameter `Kint`.

findpar

```
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 Response Optimization blocks. Use the optimization 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 Constructs 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 multi-dimensional 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 `initpar(proj)`

Description `initpar(proj)` sets the `InitialGuess` value of tuned parameters in the project, `proj`, with the values of the parameters that are currently in the model or base workspace.

See Also `findpar`

Purpose Upgrade models with old NCD blocks

Syntax `ncdupdate('modelName')`

Description `ncdupdate('modelName')` 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 `ncdupdate`. NCD is the version of Simulink Response Optimization that existed before Release 14.

When your model automatically loads its NCD settings from an `ncdStruct` 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.

When your NCD settings are stored in an `ncdStruct` variable, but do not automatically load with the model, first load the `ncdStruct` variable into the workspace before calling `ncdupdate`, then resave the variable afterwards.

To retain the upgraded blocks, make sure you also save the model after running `ncdupdate`.

See Also `supdate`

newsro

Purpose Create a default Simulink Response Optimization project

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

Description `proj=newsro('modelName',params)` creates a new 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 optimization 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 project `proj`. Use `optimset` to modify the optimization options.

For a list of all possible settings and the values they can take, see the documentation 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.

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

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: []

```

See Also `optimset`, `simget`, `simset`

optimize

Purpose Run response optimization project

Syntax `result=optimize(proj)`

Description `result=optimize(proj)` optimizes the responses specified in the project, `proj`, with the constraints, parameters and settings. The 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.

A results object, `result`, is also returned. The properties of this object are:

- **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.

For more information on the results properties, see the documentation for the Optimization Toolbox functions `fmincon` and `fminsearch` and the Genetic Algorithm and Direct Search Toolbox function `patternsearch`.

Example Open the `pitchrate_demo` model.

```
pitchrate_demo
```

Create a response optimization project based on the current settings in the model.

```
proj=getstro('pitchrate_demo');
```

Run the optimization with the following command.

```
results=optimize(proj)
```

The results are displayed as follows.

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

getsro, newsro, findpar, findconstr, optimset, 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.

For a list of all possible settings and the values they can take, see the documentation 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)
```

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

randunc

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

$$\text{Min} \leq p \leq \text{Max}$$

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

$$\text{Min}(i,j) \leq p(i,j) \leq \text{Max}(i,j)$$

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.

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
    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`

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 functions `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`

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

Block Reference

Blocks — Alphabetical List (p. 10-2) A list of available blocks, sorted alphabetically

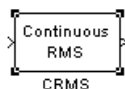
Blocks – Alphabetical List

This section contains block reference pages listed alphabetically.

Purpose Compute the continuous-time, cumulative root mean square 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.

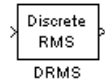
See Also DRMS, Signal Constraint

DRMS

Purpose Compute the discrete-time, cumulative root mean square 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

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

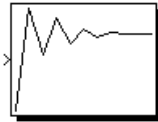
The R.M.S value gives a measure of the average energy in the signal.

See Also CRMS, Signal Constraint

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, “Introduction.”

See Also CRMS, DRMS

Signal Constraint

A

- algorithms
 - response optimization 6-4
 - response optimization example 2-28

C

- common questions
 - response optimization 8-1
- constraint bounds 4-2
 - deleting 2-22
 - example 2-5
 - positioning exactly 4-4
 - specifying using functions 7-5
 - splitting 4-9
- constraint edges
 - moving 4-2
- constraints
 - positioning bounds 4-2
 - scaling 4-9
 - weightings 4-5
- CRMS block 10-3
- cumulative root mean square blocks 1-2
- current responses
 - plotting 4-14

D

- desired responses
 - constraint bounds 4-2
 - reference signals 4-13
 - step responses 4-10
- DRMS block 10-4

E

- Edit Constraint dialog 4-4

- example 2-23

F

- findconstr function 9-4
 - example 7-5
- findpar function 9-7
 - example 7-3

G

- getsro function 9-9
 - example 7-3
- gridlines
 - response plots 4-4
- gridunc function 9-10
 - example 7-8

I

- initial responses
 - plotting 4-14
- initpar function 9-12
- intermediate responses
 - plotting 4-14

L

- loading
 - response optimization projects 3-6

N

- ncdupdate function 9-13
- newsro function 9-14
 - example 7-3
- Nonlinear Control Design Blockset

upgrading 1-5

O

optimget function 9-15

example 7-7

optimize function 9-16

example 7-9

optimset function 9-18

example 7-7

P

parameters

tuned 5-2

uncertain 5-5

R

randunc function 9-20

example 7-8

reference signals 2-12

plotting 4-14

specifying 4-13

tracking 4-13

response optimization

accelerating 6-11

choosing signals 3-2

creating projects 3-3

example with control design 2-4

example with functions 7-2

example with physical modeling 2-20

gradient type 6-5

optimization options using functions 7-7

quick start 2-2

running 6-2

running with functions 7-9

simulation options 6-8

simulation options using functions 7-7

simulation solvers 6-9

starting 6-2

termination criteria 6-5

tolerances 6-5

response optimization options 2-28

example 2-13

response optimization projects

creating with functions 7-2

example 2-19

objects 7-3

properties 3-3

reloading 3-6

saving 3-4

response optimization results

example 2-10

optimization progress dialog 6-3

plots 6-2

tuning 6-4

response plots

editing properties 4-15

S

saving

response optimization projects 3-4

setunc function 9-22

example 7-9

Signal Constraint block 10-5

signal constraints

objects 7-5

signal responses

plotting 4-14

signal tracking

example 2-12

simget function 9-23

- example 7-7
- using in response optimization 5-5
- simset function 9-24
 - example 7-8
- simulation options
 - response optimization 6-8
- simulation solvers
 - response optimization 6-9
- step responses
 - example 2-24
 - specifications 4-10
- system requirements 1-4

T

- troubleshooting
 - response optimization 8-1
- tuned parameters
 - adding 5-2
 - example 2-8
 - multiple 2-26
 - object properties 7-3
 - results 6-2
 - specifications 5-3
 - specifying 5-2
 - specifying with functions 7-3
 - undoing 6-3

U

- uncertain parameters
 - adding 5-6
 - example 2-16
 - grid 5-7
 - random 5-7
 - specifications 5-7
 - specifying 5-5
 - specifying with functions 7-8

