Simulink® Response Optimization

For Use with Simulink®

- Modeling
- Simulation
- Implementation

Getting Started

Version 2

The MathWorks

Getting Started with Simulink Response Optimization

Printing History: June 2004	First printing	New for Version 2.0 (Release 14) (Renamed from *Nonlinear Control Design Blockset User's Guide*)

Contents

Introduction

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, "Tutorial," which introduces the main product features and uses two simple examples to describe their use.

To perform response optimization using functions, read Chapter 3, "Response Optimization Using Functions."

For advice on solving typical problems, read Chapter 4, "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.

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

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

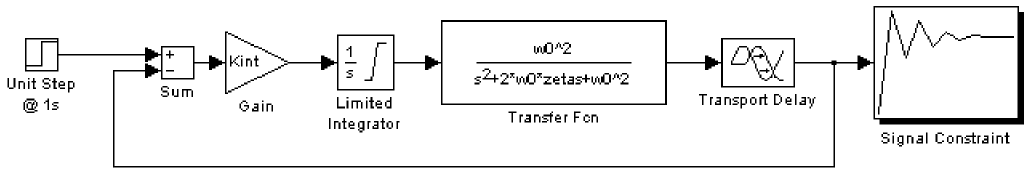
1. Make a model of your (nonlinear) system and controller using Simulink. Add input signals (e.g., steps, ramps, observed data) for which you know what the desired output should look like.
2. Attach a Signal Constraint block to the signals you want to constrain. The Simulink library `srolib` contains 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 box 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 box 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 or to the 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 box. 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 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 (`Kint`) 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.

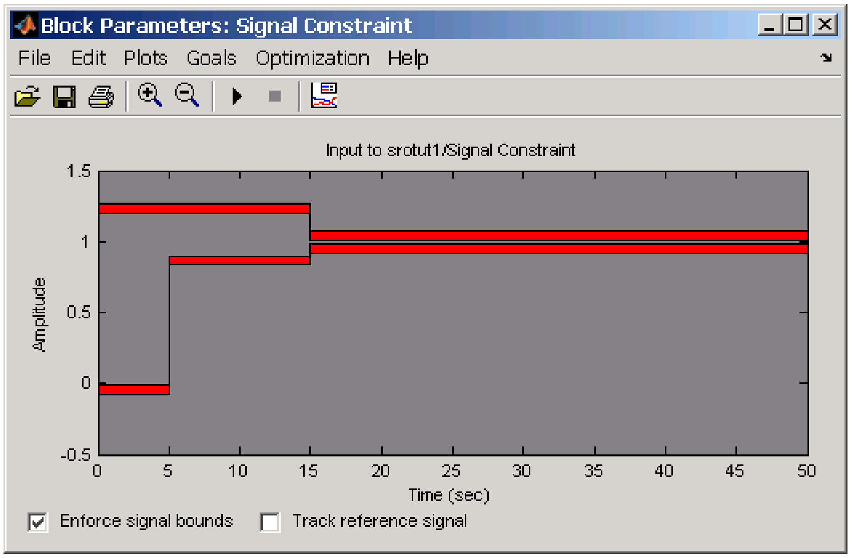
You do not need to remodel any of your present Simulink systems to use Simulink Response Optimization. You need simply to

- 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

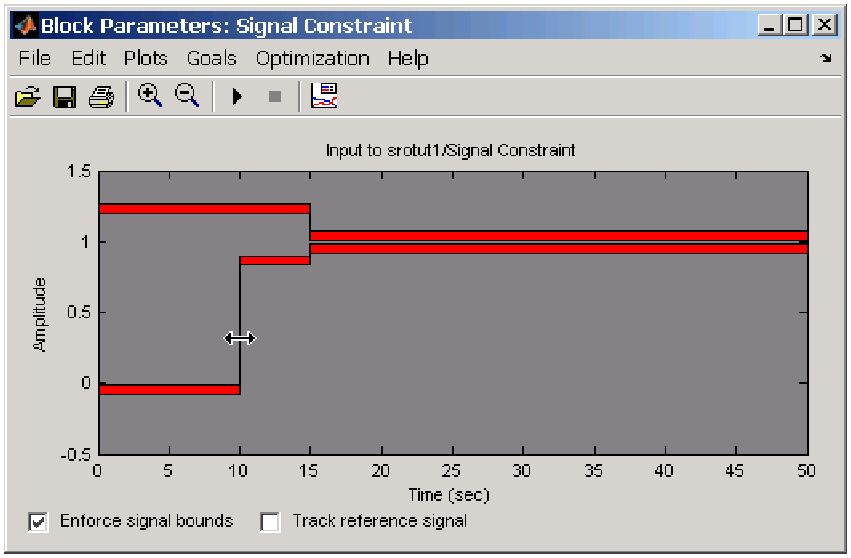
To open the Simulink Response Optimization **Signal Constraint** window, double-click on the Signal Constraint block. The window, shown in the following figure, contains an amplitude versus time axis with default upper and lower constraint bounds. To optimize the tuned parameters so that the response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


Block Parameters: Signal Constraint
File Edit Plots Goals Optimization Help
Input to srotut1/Signal Constraint
Amplitude
1.5
1
0.5
0
-0.5
0 5 10 15 20 25 30 35 40 45 50
Time (sec)
Enforce signal bounds
Track reference signal

Enforce signal bounds [✓]
Track reference signal [ ]

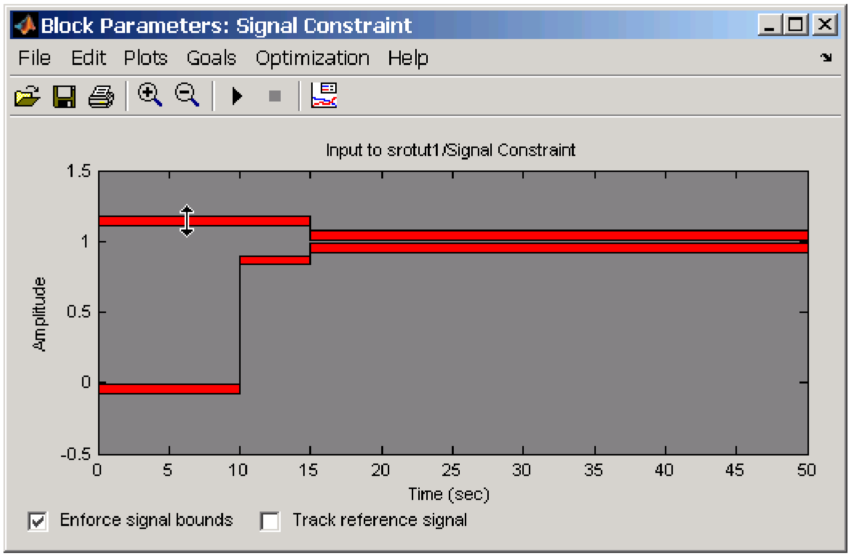
The thickness of the constraint bounds holds no significance; it merely provides visual cues for clicking and dragging the constraint bounds.

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 vertical line separating the lower bound constraint that ends at 5 seconds and the lower bound constraint that begins at 5 seconds. Press and hold down the (left) mouse button. The arrow should turn to a left/right drag cursor as shown below.

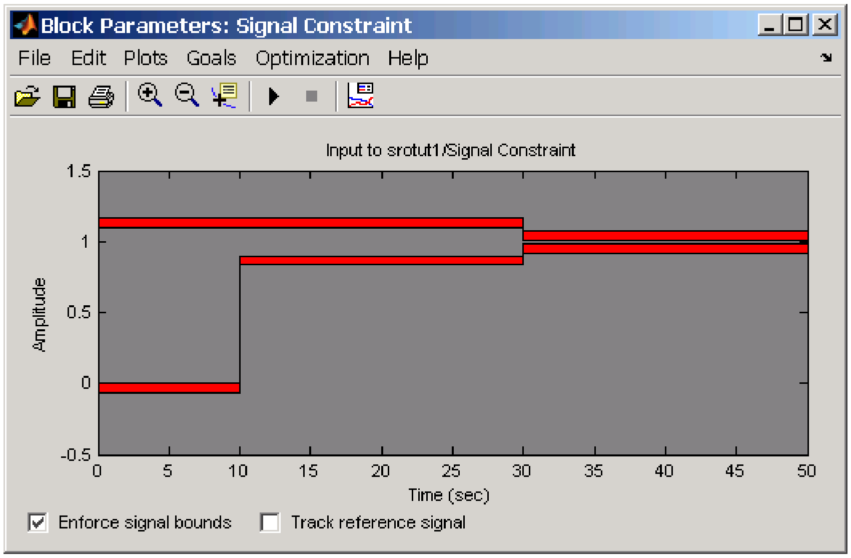

Block Parameters: Signal Constraint
File Edit Plots Goals Optimization Help
Input to srotut1/Signal Constraint
1.5
1
0.5
0
-0.5
Amplitude
0 5 10 15 20 25 30 35 40 45 50
Time (sec)
Enforce signal bounds
Track reference signal

In this mode, you can change the time boundary between two 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 axes and select **Grid** from the menu. If you need to precisely place constraint-bound segments, use the **Edit Constraint** dialog box, 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 box.

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 an up/down drag cursor. In this mode, you can drag a constraint vertically 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.


Block Parameters: Signal Constraint
File Edit Plots Goals Optimization Help
Input to srotut1/Signal Constraint
Amplitude
1.5
1
0.5
0
-0.5
0 5 10 15 20 25 30 35 40 45 50
Time (sec)
☑ Enforce signal bounds
☐ Track reference signal

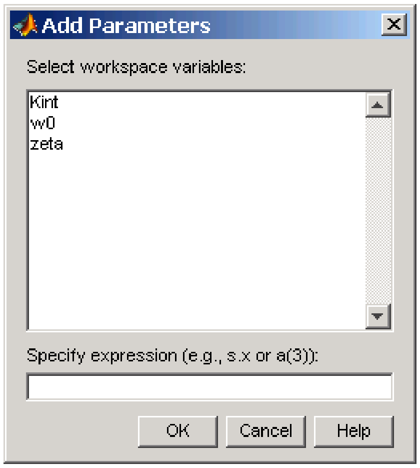
Finally, the settling time constraints require adjustment. Position the mouse button just within the left edge of the upper bound constraint extending from 15 to 50 seconds. Press and hold down the (left) mouse button and notice that the constraint becomes selected and that the pointer changes to a four-headed arrow. In this mode, you can stretch the end of the constraint at any angle. While still holding the mouse button down, drag the constraint so that the settling-time constraint begins at 30 seconds. Adjust the lower bound constraint so that it too defines a 30 second settling-time constraint. The constraint figure should now look the one shown below.


Block Parameters: Signal Constraint
File Edit Plots Goals Optimization Help
Input to srotut1/Signal Constraint
Amplitude
Time (sec)
1.5
1
0.5
0
-0.5
0 5 10 15 20 25 30 35 40 45 50
Enforce signal bounds
Track reference signal

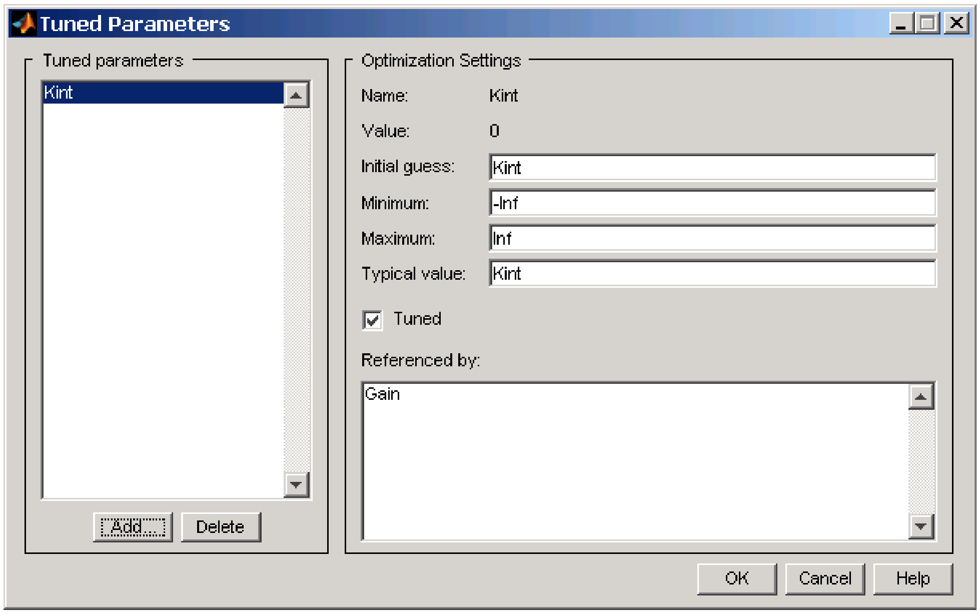
Alternatively, you could set the signal constraints using the **Desired Response** dialog by selecting **Goals -> Desired Response** from the **Signal Constraint** window. For more information on setting signal constraints, see the online documentation.

Specifying Tuned Parameters

Before beginning the optimization, you must tell Simulink Response Optimization which variables the optimization should tune. Open the **Tuned Parameters** dialog box 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 box with a list of all model parameters that are currently in the workspace.


Add Parameters
Select workspace variables:
Kint
w0
zeta
Specify expression (e.g., s.x or a(3)):
OK
Cancel
Help

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



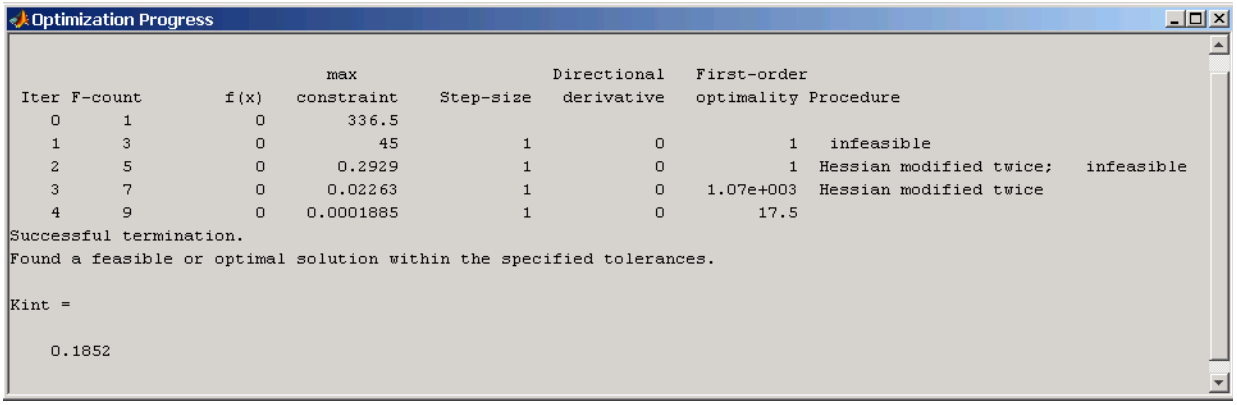
Name: Kint
Value: 0
Initial guess: Kint
Minimum: -Inf
Maximum: Inf
Typical value: Kint
[✓ Tuned]
Referenced by:
Gain

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 box, 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**.

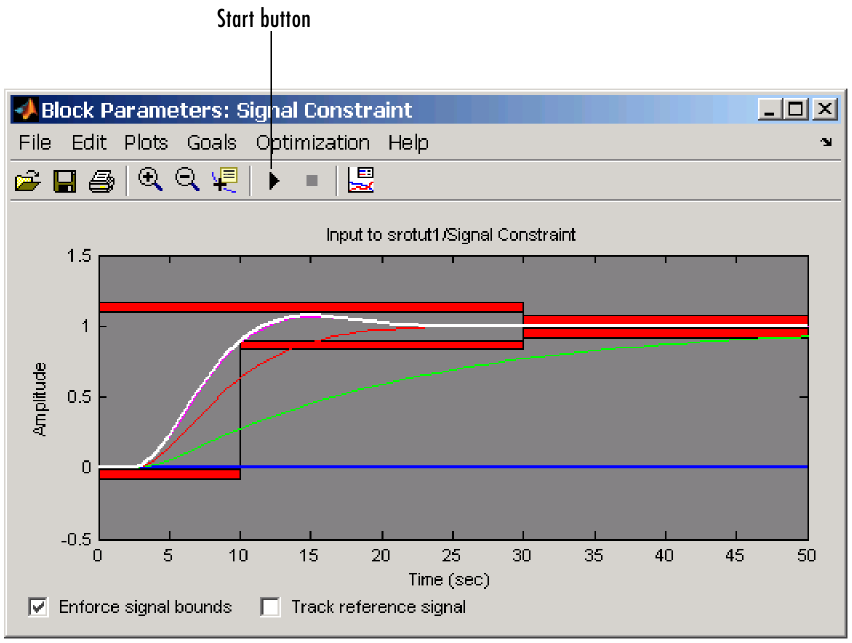
Simulink Response Optimization automatically converts the constraint bound data and tuned parameter information into a constrained optimization problem that it solves using functions from the Optimization Toolbox or the Genetic Algorithm and Direct Search Toolbox. It attempts to satisfy the constraints 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.



```
                                max                    Directional   First-order
 Iter F-count        f(x)   constraint    Step-size    derivative   optimality Procedure
    0      1            0        336.5
    1      3            0           45            1            0            1   infeasible
    2      5            0       0.2929            1            0            1  Hessian modified twice;   infeasible
    3      7            0      0.02263            1            0    1.07e+003  Hessian modified twice
    4      9            0    0.0001885            1            0         17.5
Successful termination.
Found a feasible or optimal solution within the specified tolerances.

Kint =

    0.1852
```

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.



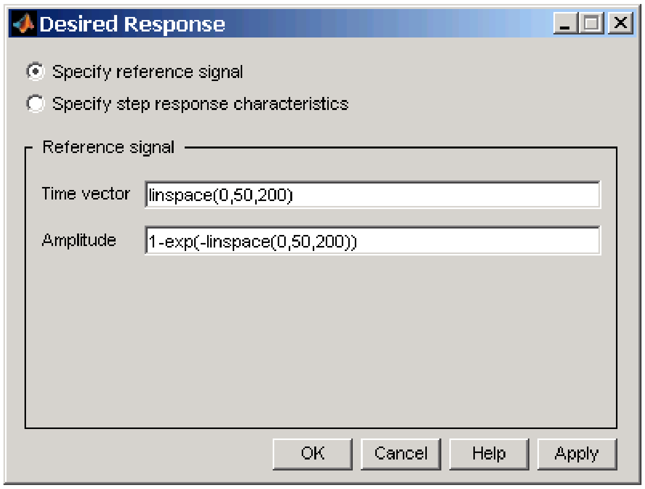
The blue line shows the initial response and the white line shows the current, or final, response. As you can see, the final response signal lies within the constraint bounds.

The new value of the tuned parameter `Kint` appears in the **Optimization Progress** window and is also changed in the MATLAB workspace. In this case the new value of `Kint` 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** window, as shown in the following figure.



[✓ Specify reference signal] Specify step response characteristics

Reference signal

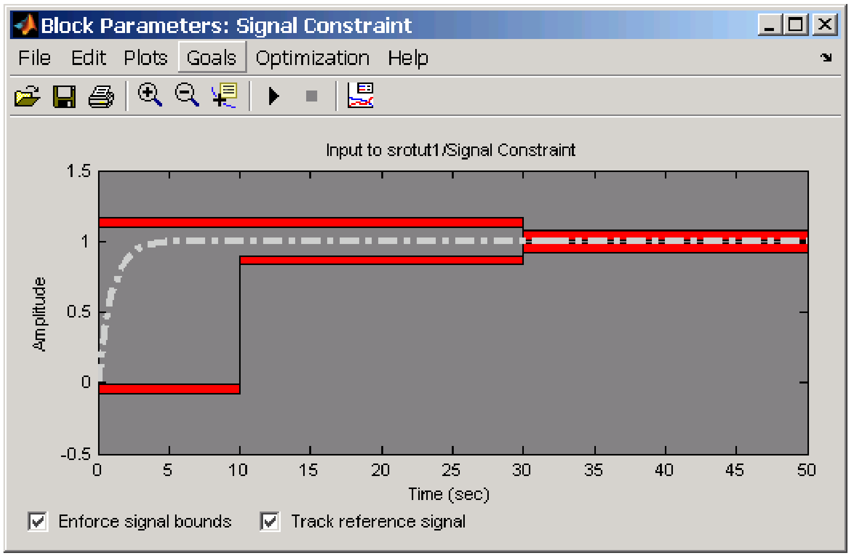
Time vector: linspace(0,50,200)

Amplitude: 1-exp(-linspace(0,50,200))

For this example, use the reference signal given by

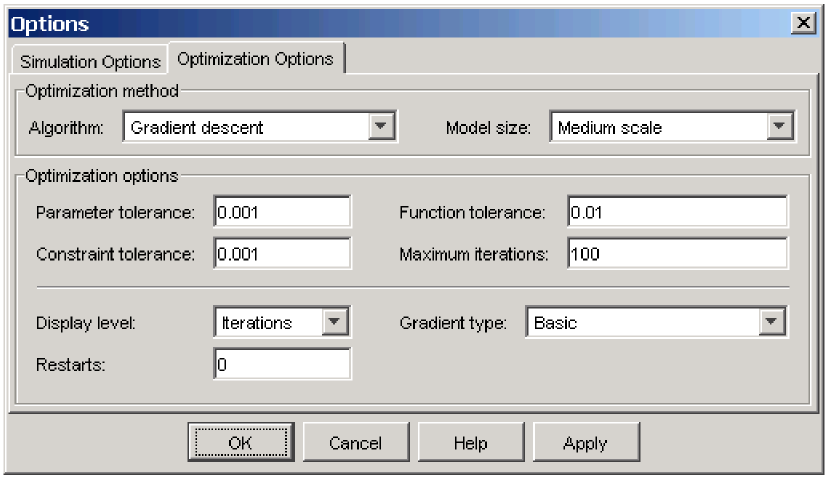
$$y = 1 - e^{-t}$$

Enter time and amplitude vectors for the reference signal in the **Desired Response** window 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**.


Block Parameters: Signal Constraint
File Edit Plots Goals Optimization Help
Input to srotut1/Signal Constraint
Amplitude
Time (sec)
1.5
1
0.5
0
-0.5
0 5 10 15 20 25 30 35 40 45 50

☑ Enforce signal bounds ☑ Track reference signal

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



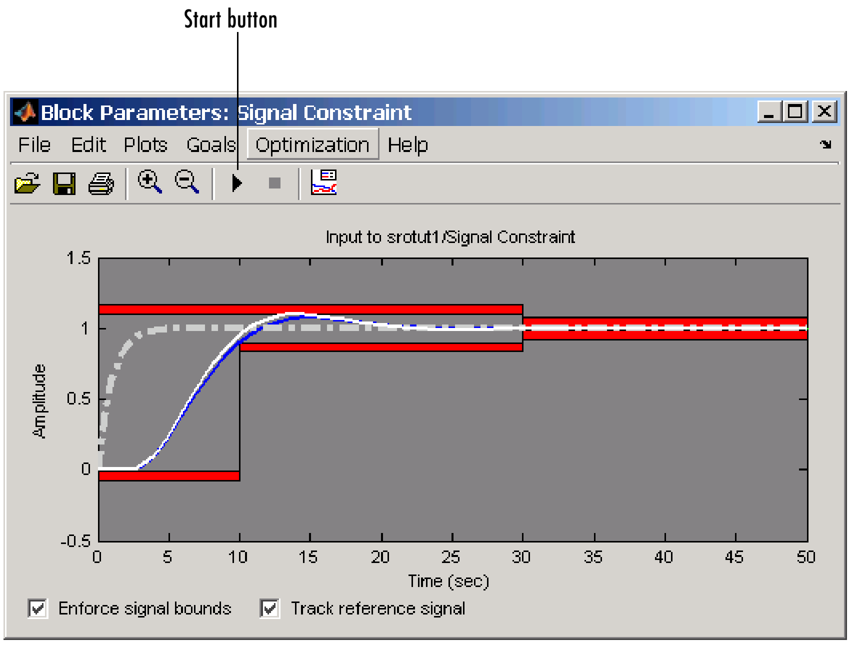
Optimization method

Algorithm: Gradient descent
Model size: Medium scale

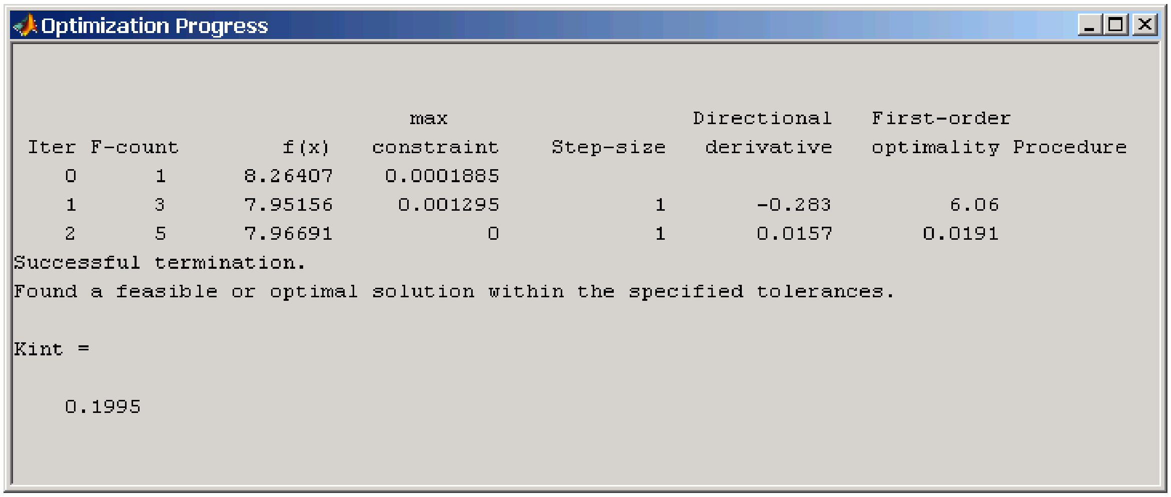
Optimization options

Parameter tolerance: 0.001
Function tolerance: 0.01
Constraint tolerance: 0.001
Maximum iterations: 100
Display level: Iterations
Gradient type: Basic
Restarts: 0

To run the optimization, select **Optimization -> Start** or use the Start button below the menu bar. 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 `Kint`.



```
                                  max                      Directional   First-order
 Iter F-count        f(x)     constraint    Step-size    derivative   optimality Procedure
    0      1       8.26407    0.0001885
    1      3       7.95156     0.001295            1        -0.283         6.06
    2      5       7.96691            0            1        0.0157       0.0191
Successful termination.
Found a feasible or optimal solution within the specified tolerances.

Kint =

    0.1995
```

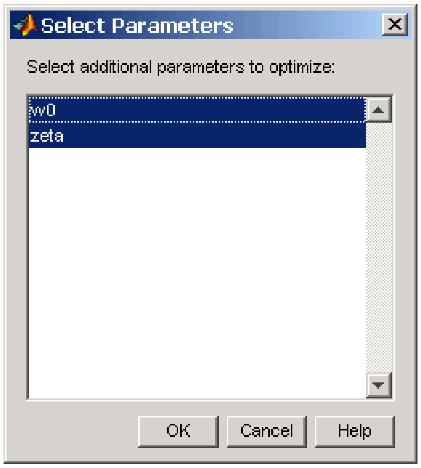
Before proceeding to the next section, make sure to clear the **Track reference signal** option at the bottom of the **Signal Constraint** window.

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.

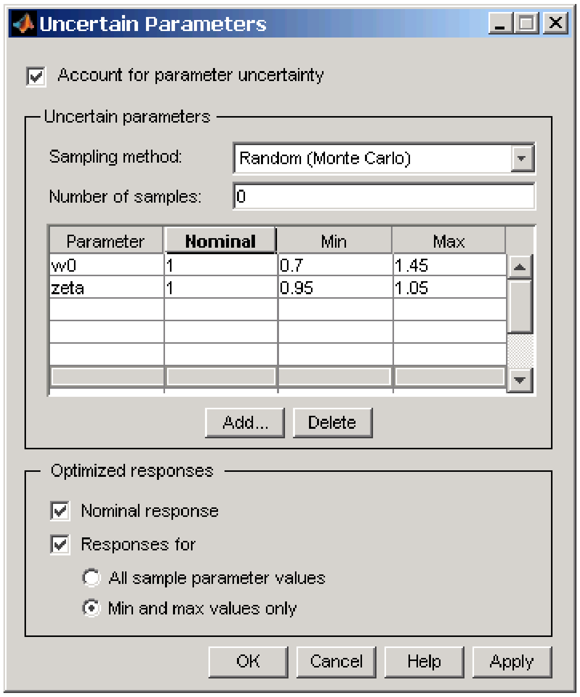
Simulink Response Optimization allows you to optimize response signals in the face of this uncertainty by specifying uncertainty in parameter values.

Open the **Uncertain Parameters** dialog box 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 box. This dialog box 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** window 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 `w0`. The **Uncertain Parameters** window should now look like that in the following figure.



Uncertain Parameters

[✓] Account for parameter uncertainty

Uncertain parameters

Sampling method: Random (Monte Carlo)

Number of samples: 0

Parameter	Nominal	Min	Max
w0	1	0.7	1.45
zeta	1	0.95	1.05

Optimized responses

[✓] Nominal response

[✓] Responses for

○ All sample parameter values

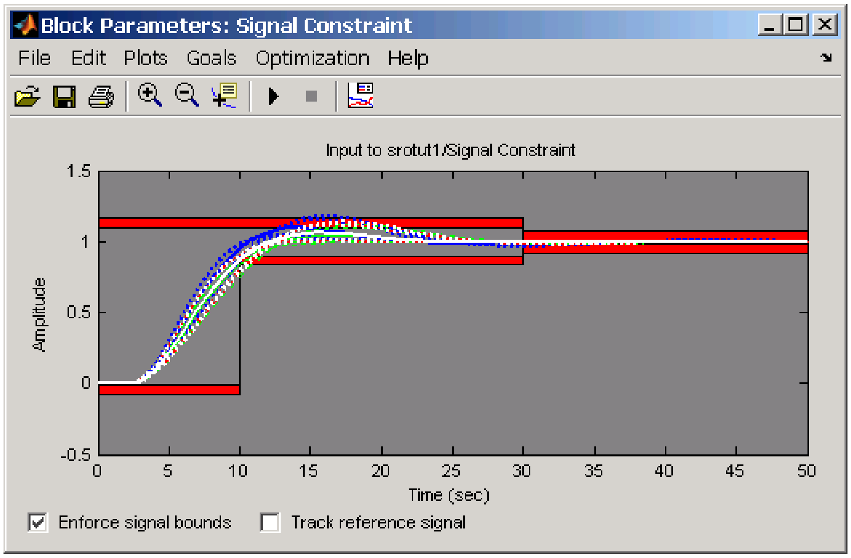
● Min and max values only

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 `w0` 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** window. 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.


Block Parameters: Signal Constraint
File Edit Plots Goals Optimization Help
Input to srotut1/Signal Constraint
Amplitude
Time (sec)
1.5
1
0.5
0
-0.5
0
5
10
15
20
25
30
35
40
45
50
Enforce signal bounds
Track reference signal

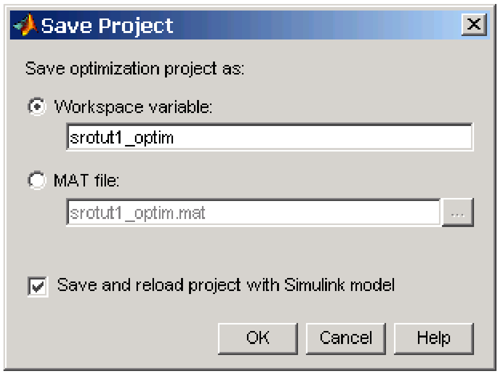
To get the optimization to converge, you might try a different initial guess, relax the constraints slightly, or change the uncertain parameter values. In this example the uncertain response signals do not violate the constraints by a large amount so the result might be acceptable for your design needs.

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 box as shown below.



Save optimization project as:

(●) Workspace variable: srotut1_optim

( ) MAT file: srotut1_optim.mat

[✓] Save and reload project with Simulink model

You can save the project to either a 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 12×10^5Pa. 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, `Ac`, and the maximum pump flow-rate, `Qmax`.

The specifications for the response of the piston position signal are

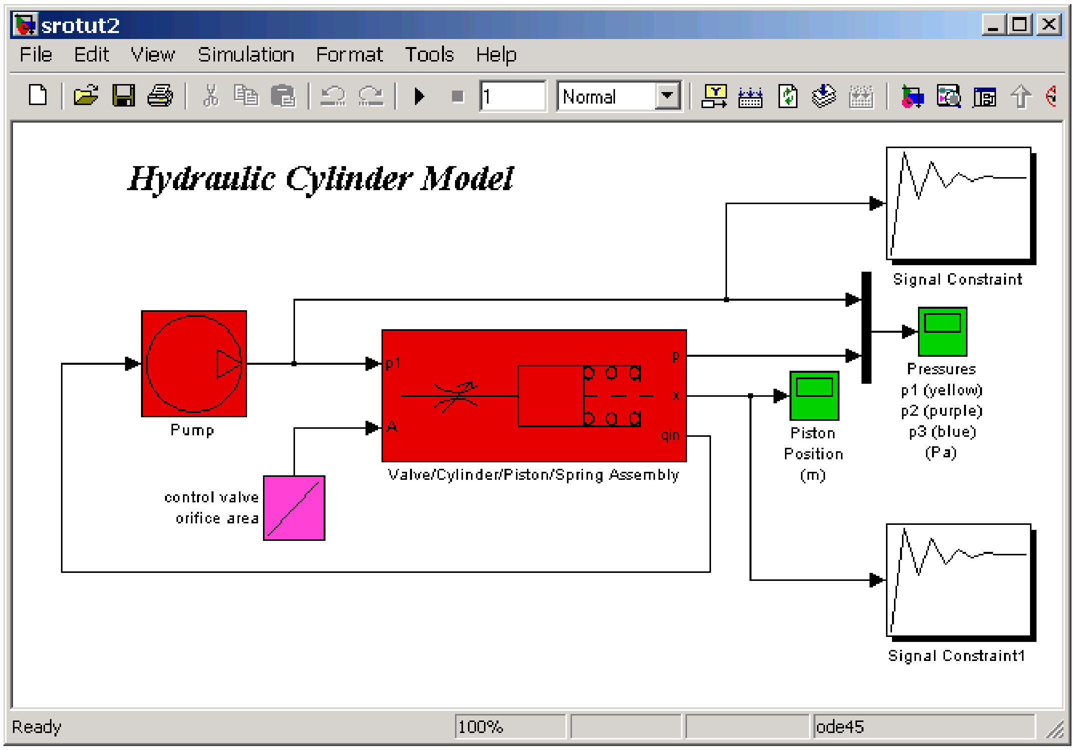
- Maximum overshoot of 14%
- Rise time of 25 seconds
- Settle to within 14% of final value after 25 seconds
- Final value of 0.07m

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 12×10^5Pa

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.

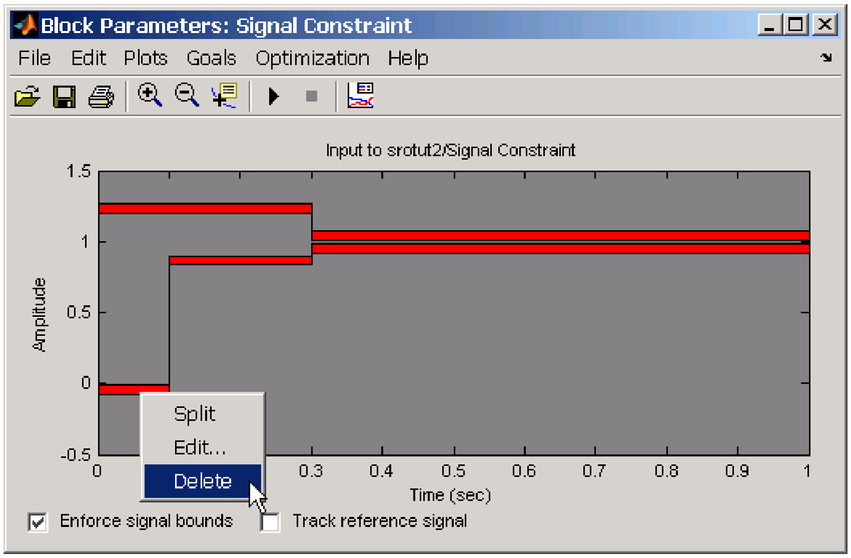
Adjusting Constraints

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

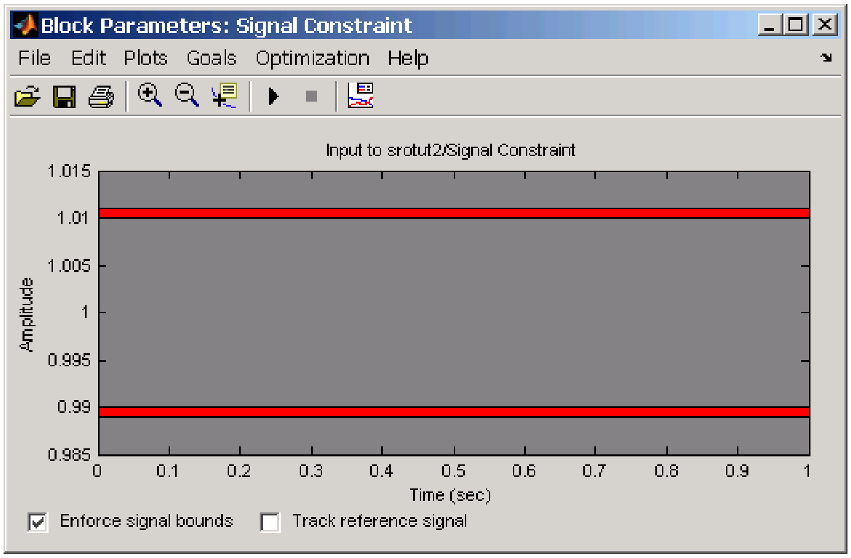
Adjusting Pump Pressure Signal Constraints

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.



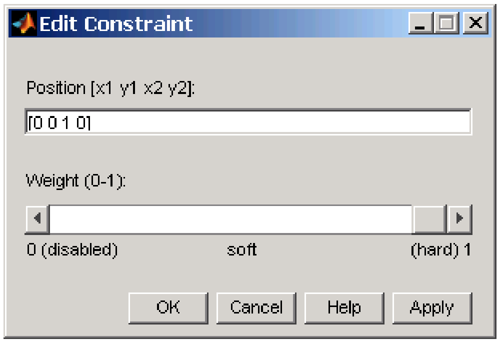
Repeat for one additional lower segment and one upper segment. The figure window should now look like that in the following figure.


Block Parameters: Signal Constraint
File Edit Plots Goals Optimization Help
Input to srotut2/Signal Constraint
Amplitude
Time (sec)
1.015
1.01
1.005
1
0.995
0.99
0.985
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

[✓] Enforce signal bounds [ ] Track reference signal

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 a vector describing the new position of the constraint segment.

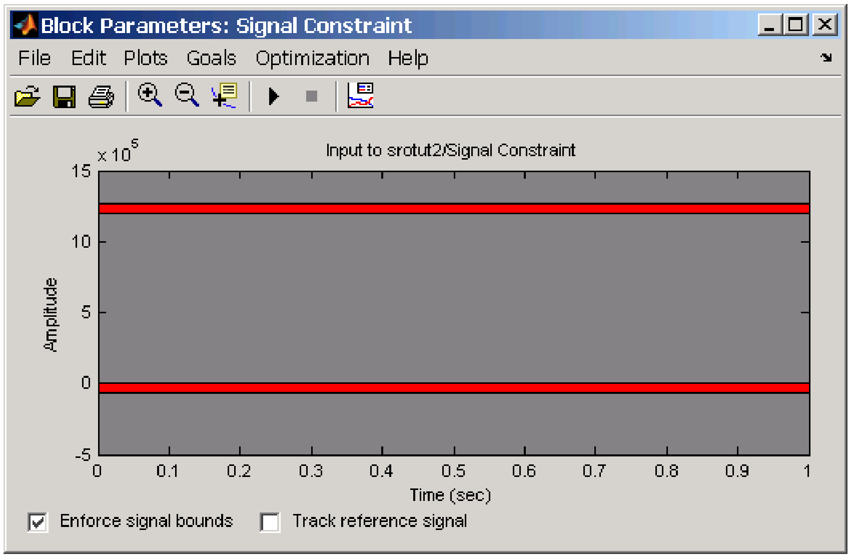
The position vector must have four elements, representing the starting (left) *x*-position of the segment, the starting *y*-position, the ending (right) *x*-position, and the ending *y*-position, respectively. The following figure shows the **Edit Constraint** dialog with the position vector for the lower constraint segment.



Position [x1 y1 x2 y2]: [0 0 1 0]

Weight (0-1):

Repeat for the upper constraint, moving the constrain segment to the position specified by the vector `[0 12e5 1 12e5]`. The figure window should now look like that in the following window.

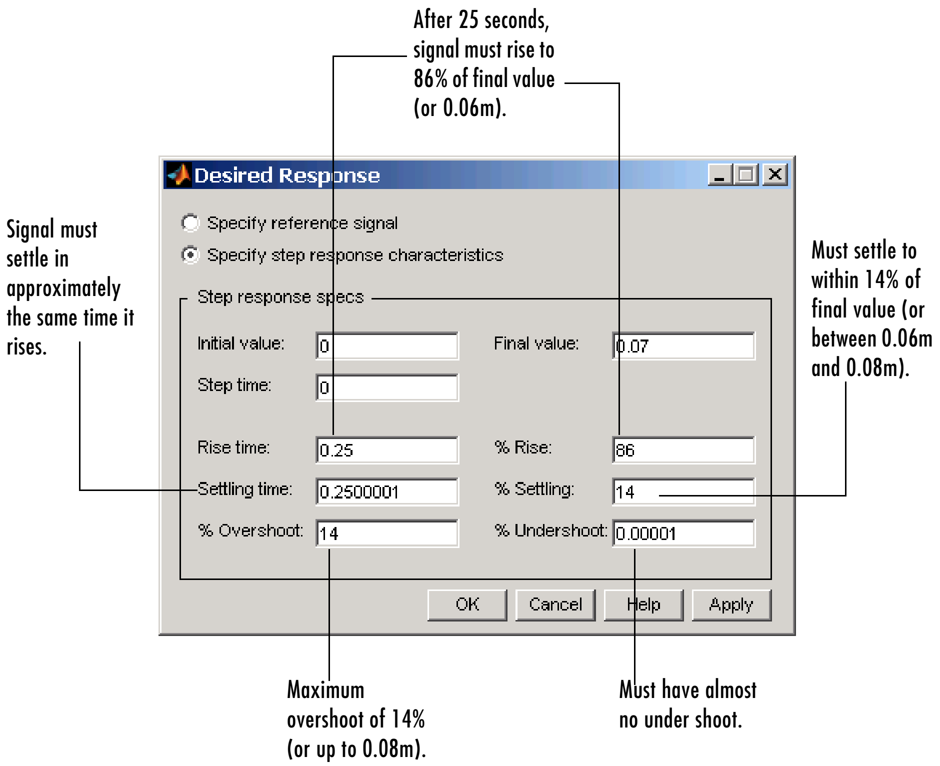


See the online documentation for more information on the **Edit Constraint** dialog.

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 this chapter specifies the step response characteristics. Enter the following information, and then click **OK**.



After 25 seconds, signal must rise to 86% of final value (or 0.06m).

Signal must settle in approximately the same time it rises.

Must settle to within 14% of final value (or between 0.06m and 0.08m).

Maximum overshoot of 14% (or up to 0.08m).

Must have almost no under shoot.

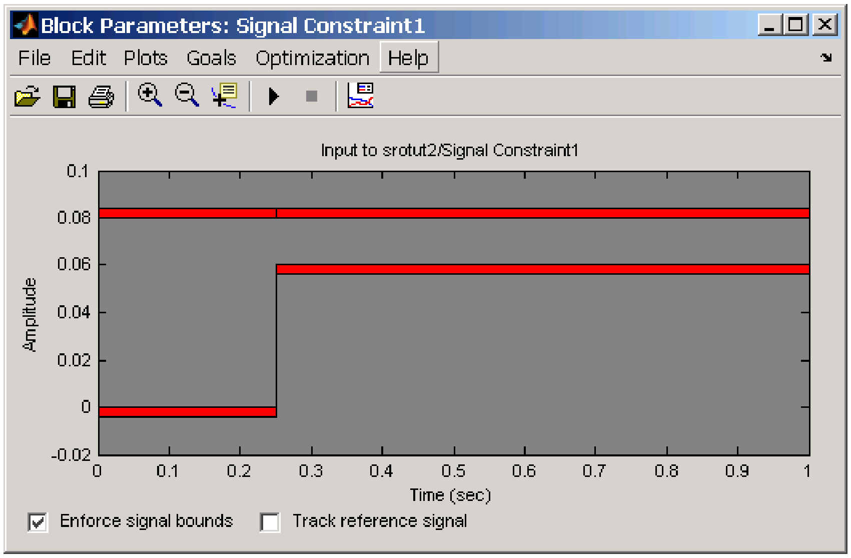
Desired Response

Specify reference signal [✓ Specify step response characteristics]

Step response specs

Initial value:	0	Final value:	0.07
Step time:	0		
Rise time:	0.25	% Rise:	86
Settling time:	0.2500001	% Settling:	14
% Overshoot:	14	% Undershoot:	0.00001

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



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

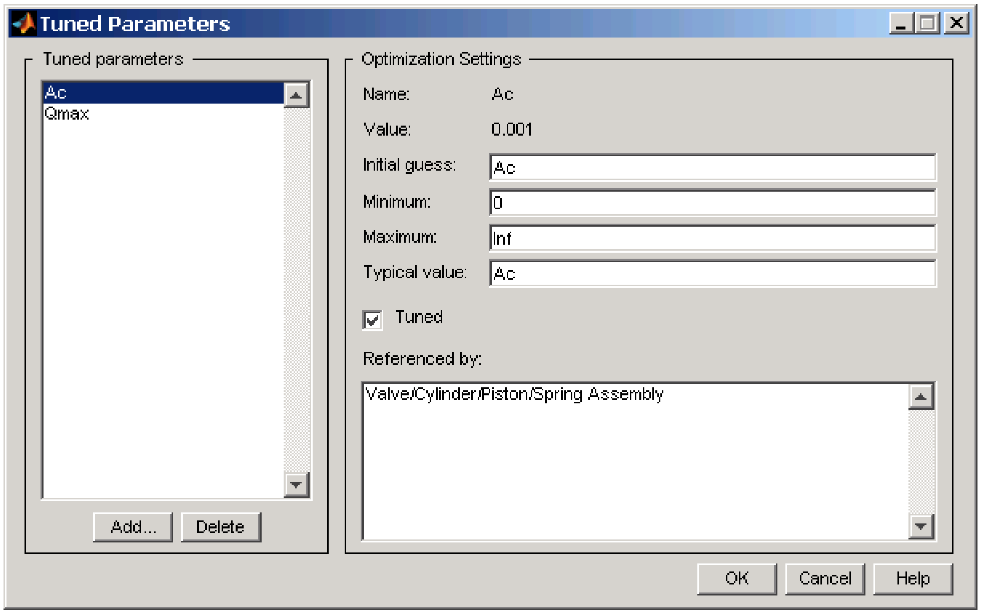
Specifying Tuned Parameters

To optimize the signal responses in this example, you will tune two parameters: the cross-sectional area of the piston, `Ac`, and the maximum pump flow-rate, `Qmax`. 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 `Ac` and `Qmax` from the list in the **Add Parameter**s 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 both Ac, 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 Qmax to avoid negative flow-rates, which imply that the flow reverses direction through the system.



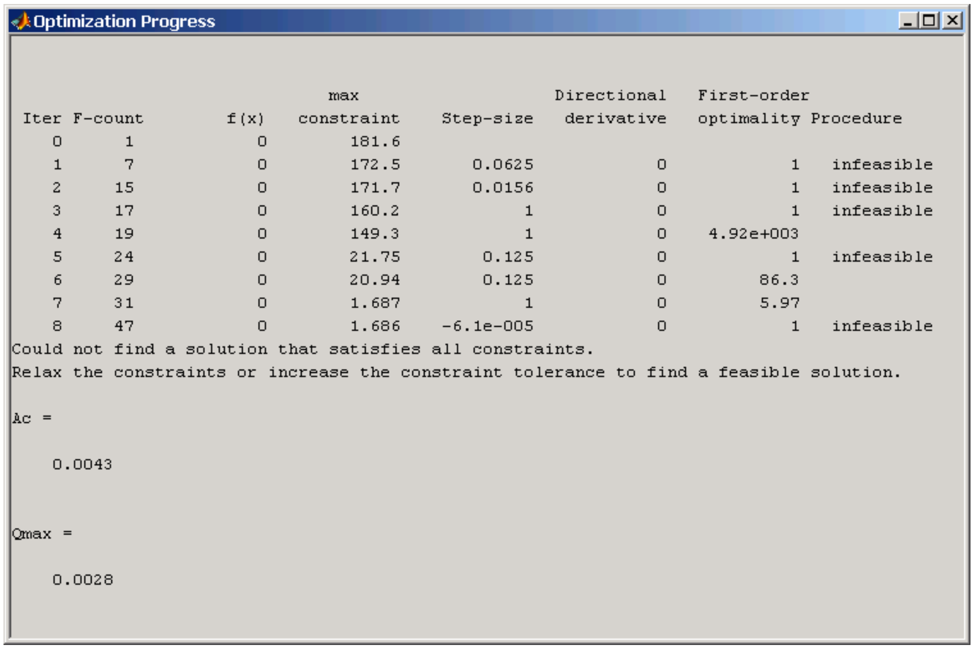
Name: Ac
Value: 0.001
Initial guess: Ac
Minimum: 0
Maximum: Inf
Typical value: Ac
Referenced by: Valve/Cylinder/Piston/Spring Assembly

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

Running the Optimization

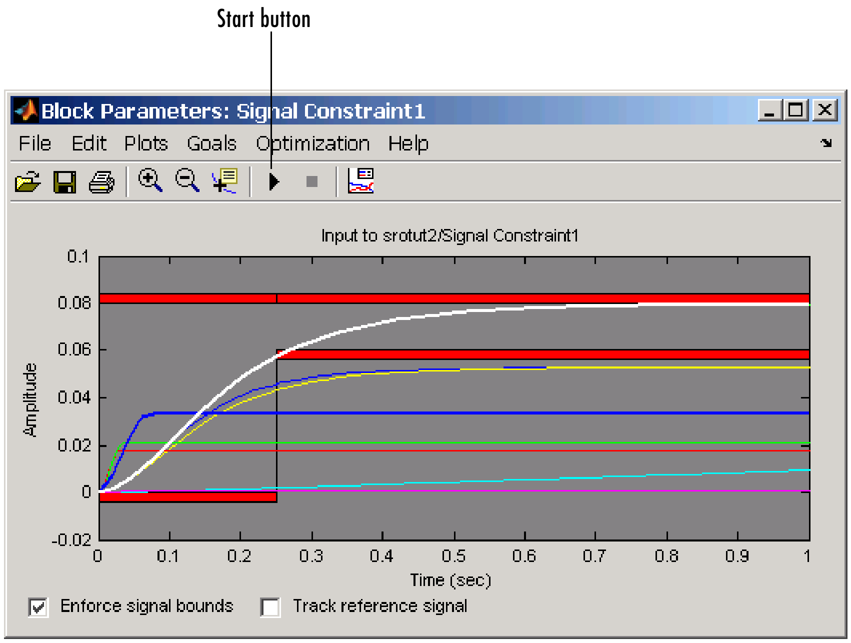
After adjusting the constraints and defining tuned parameters, start the optimization by selecting **Optimization -> Start**, or by clicking the Start button below the menu bar in a **Signal Constraint** window.

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.



```
                                  max                     Directional   First-order
 Iter F-count        f(x)     constraint    Step-size      derivative   optimality Procedure
    0       1           0          181.6
    1       7           0          172.5       0.0625            0             1    infeasible
    2      15           0          171.7       0.0156            0             1    infeasible
    3      17           0          160.2            1            0             1    infeasible
    4      19           0          149.3            1            0      4.92e+003
    5      24           0          21.75        0.125            0             1    infeasible
    6      29           0          20.94        0.125            0          86.3
    7      31           0          1.687            1            0          5.97
    8      47           0          1.686     -6.1e-005           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 =

    0.0043


Qmax =

    0.0028
```

However, the **Signal Constraint** window for the piston position shows that the solution is actually very close to satisfying the constraints.


Start button
Block Parameters: Signal Constraint1
File Edit Plots Goals Optimization Help
Input to srotut2/Signal Constraint1
Amplitude
Time (sec)
Enforce signal bounds
Track reference signal

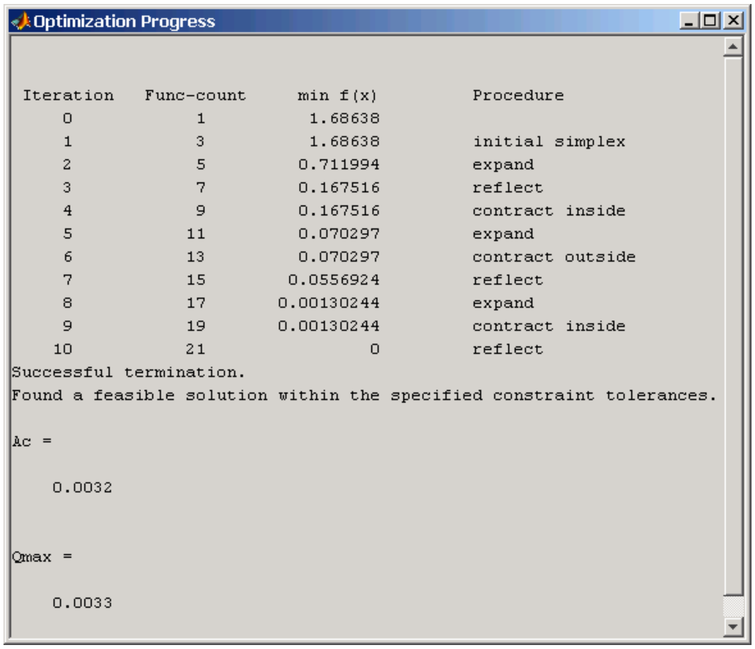
Changing Optimization Settings

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 optimization algorithm to `Simplex search` by selecting **Optimization -> Optimization Options** in the **Signal Constraint** window, and then selecting `Simplex search` as the **Algorithm**. Click **OK** to save the changes and close the **Options** dialog. See the online documentation for more information on optimization settings.

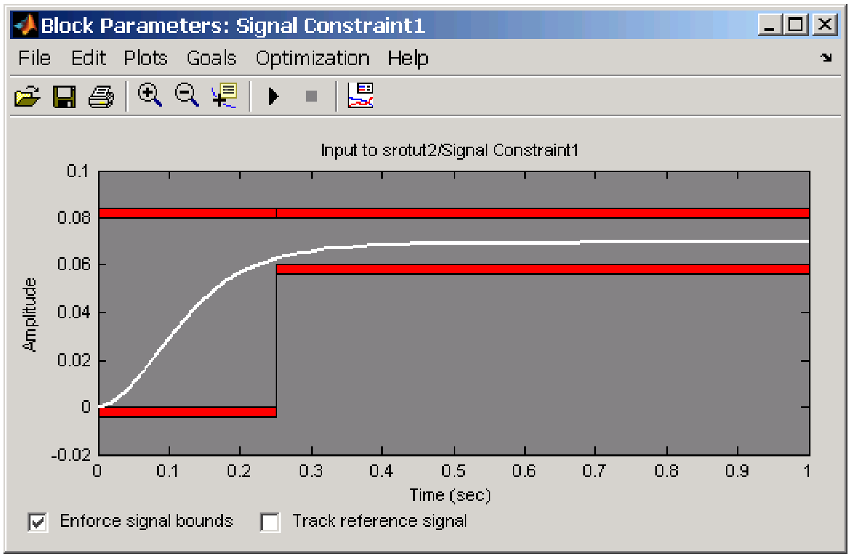
Run the optimization again with these new settings. The optimization algorithm should now find a feasible solution as shown in the following figure.



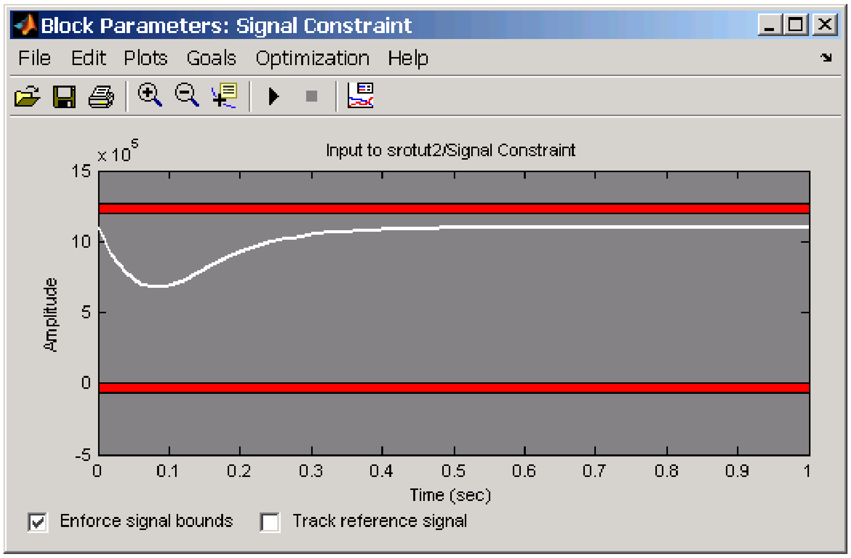
```
 Iteration   Func-count     min f(x)         Procedure
     0            1          1.68638
     1            3          1.68638         initial simplex
     2            5         0.711994         expand
     3            7         0.167516         reflect
     4            9         0.167516         contract inside
     5           11         0.070297         expand
     6           13         0.070297         contract outside
     7           15        0.0556924         reflect
     8           17       0.00130244         expand
     9           19       0.00130244         contract inside
    10           21                0         reflect
Successful termination.
Found a feasible solution within the specified constraint tolerances.

Ac =

    0.0032


Qmax =

    0.0033
```

The **Optimization Progress** window shows the new optimized parameter values of 0.0032 m^2 for the cross-sectional area of the piston, and 0.0033 m^3/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.



☑ Enforce signal bounds ☐ Track reference signal



☑ Enforce signal bounds ☐ Track reference signal

In this case the simplex search algorithm was able to find a solution when gradient descent could not. However, starting with simplex search would not have found a feasible solution as the initial response signals were not close enough to the constrained responses. The approach used in this example, of getting close to a feasible solution with one method, then using another method to find the actual solution is a commonly used approach with response optimization.

See Chapter 4, “Troubleshooting” for more advice on adjusting the response optimization to acheive the desired results.

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

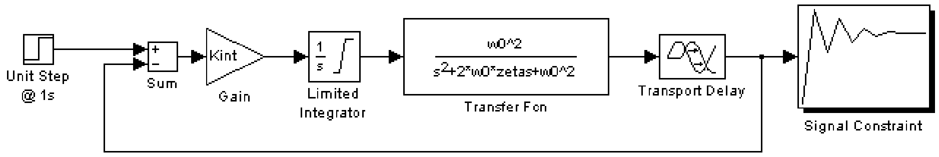
This example uses functions to optimize the response of a signal in the model `srotut1`. For a more detailed description of these functions, refer to the "Function Reference" in the online documentation.

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]
     OptimStatus: 'idle'
           Tests: [1x1 ResponseOptimizer.SimTest]
           Model: 'srotut1'
```

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: 0.3000
     InitialGuess: 0.3000
          Minimum: -Inf
          Maximum: Inf
     TypicalValue: 0.3000
     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: 0.3000
     InitialGuess: 0.4000
          Minimum: 0
          Maximum: Inf
     TypicalValue: 0.3000
     ReferencedBy: {0x1 cell}
      Description: ''
            Tuned: 1

  Tuned parameter.
```

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

Adjusting Signal Constraints

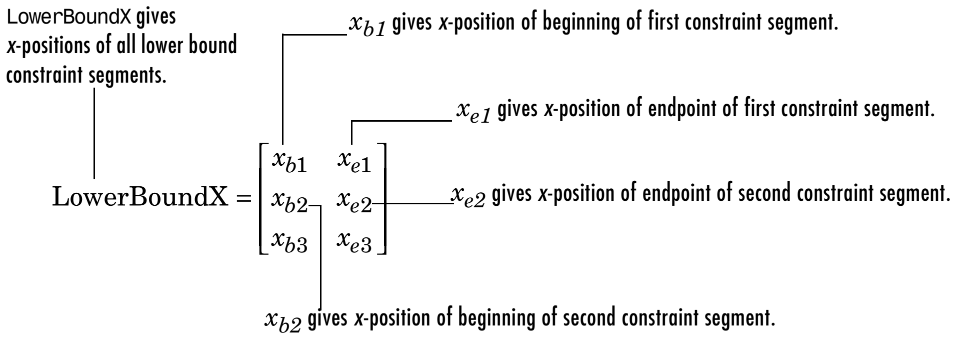
To define the constrain bound segments within which the optimized signal response must lie, first extract a signal constraint object from the project, using the function `findconstr`.

```
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.

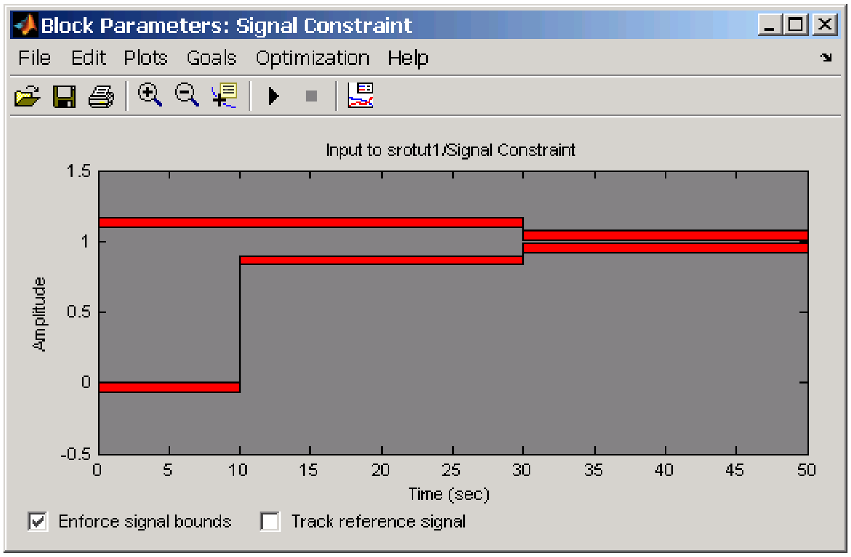
```
      ConstrEnable: 'on'
        CostEnable: 'off'
            Enable: 'on'
      SignalSource: [1x1 ResponseOptimizer.SimSource]
        SignalSize: [1 1]
       LowerBoundX: [3x2 double]
       LowerBoundY: [3x2 double]
  LowerBoundWeight: [3x1 double]
       UpperBoundX: [2x2 double]
       UpperBoundY: [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.
x_{b1} gives x-position of beginning of first constraint segment.
x_{e1} gives x-position of endpoint of first constraint segment.
$\text{LowerBoundX} = \begin{bmatrix} x_{b1} & x_{e1} \\ x_{b2} & x_{e2} \\ x_{b3} & x_{e3} \end{bmatrix}$
x_{e2} gives x-position of endpoint of second constraint segment.
x_{b2} gives x-position of beginning of second constraint segment.

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

```
constr.LowerBoundX=[0 10;10 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];
```


Block Parameters: Signal Constraint
File Edit Plots Goals Optimization Help
Input to srotut1/Signal Constraint
Amplitude
1.5
1
0.5
0
-0.5
0 5 10 15 20 25 30 35 40 45 50
Time (sec)
Enforce signal bounds
Track reference signal

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
```

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
                Debug: 'off'
           Decimation: 1
         DstWorkspace: 'current'
       FinalStateName: ''
            FixedStep: 'auto'
         InitialState: []
          InitialStep: 'auto'
             MaxOrder: 5
           SaveFormat: 'Array'
        MaxDataPoints: 0
              MaxStep: 'auto'
              MinStep: 'auto'
         OutputPoints: 'all'
      OutputVariables: 'txy'
```

```
                   Refine: 1
                   RelTol: 1.0000e-003
                   Solver: 'ode45'
             SrcWorkspace: 'base'
                    Trace: ''
                ZeroCross: 'on'
       ExtrapolationOrder: 4
   NumberNewtonIterations: 1
```

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 `w0` varies between 0.7 and 1.45 while `zeta` 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',{0.7 1.45},'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 `w0` and `zeta` 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',{0.7 0.9 1.1 1.3 1.45},'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 `0.1848`.

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

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 ±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.

Index