Simulink® Design Optimization™

Getting Started Guide

R2014**b**

MATLAB® & SIMULINK®

MathWorks®

How to Contact MathWorks

Latest news: www.mathworks.com

Sales and services: www.mathworks.com/sales_and_services

User community: www.mathworks.com/matlabcentral

Technical support: www.mathworks.com/support/contact_us

Phone: 508-647-7000

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

Simulink® Design Optimization™ Getting Started Guide

Revision History

March 2009	Online only	New for Version 1 (Release 2009a)
September 2009	Online only	Revised for Version 1.1 (Release 2009b)
March 2010	Online only	Revised for Version 1.1.1 (Release 2010a)
September 2010	Online only	Revised for Version 1.2 (Release 2010b)
April 2011	Online only	Revised for Version 1.2.1 (Release 2011a)
September 2011	Online only	Revised for Version 2.0 (Release 2011b)
March 2012	Online only	Revised for Version 2.1 (Release 2012a)
September 2012	Online only	Revised for Version 2.2 (Release 2012b)
March 2013	Online only	Revised for Version 2.3 (Release 2013a)
September 2013	Online only	Revised for Version 2.4 (Release 2013b)
March 2014	Online only	Revised for Version 2.5 (Release 2014a)
October 2014	Online only	Revised for Version 2.6 (Release 2014b)

Contents

1 Product Overview

2 Parameter Estimation

3 Response Optimization

4 Optimization-Based Linear Control Design

Product Overview

Simulink Design Optimization Product Description

Estimate and optimize Simulink model parameters

Simulink Design Optimization provides interactive tools, functions, and Simulink blocks for estimating and tuning Simulink model parameters using numerical optimization. An interactive tool lets you automatically estimate model parameters such as friction and aerodynamic coefficients from test data to increase model accuracy. You can preprocess test data, select model parameters to estimate, start an optimization, and validate estimation results.

You can also automatically tune design parameters in a Simulink model to meet objectives such as improved system performance and minimized energy consumption. Using design optimization techniques, you can meet both time- and frequency-domain constraints such as overshoot and phase margin. You can also jointly optimize physical plant parameters and algorithmic or controller gains to maximize overall system performance.

Key Features

- Model parameter estimation from test data
- Simultaneous optimization of time- and frequency-domain responses of Simulink models (with Simulink Control Design™)
- Graphical specification of response requirements and visual monitoring of the optimization progress
- Optimization of parameters to meet requirements specified by Model Verification blocks
- Custom constraints and cost functions for response optimization
- Scripting interface for programmatic specification of design optimization problems
- Robust design optimization, accounting for parameter variation or uncertainty

Optimization Support for Simulink Models Using Third-Party Applications

You can use Simulink Design Optimization software to optimize Simulink models that invoke third-party simulation tools or contain legacy simulation code. To do so, use the S-function block in Simulink. When using the command-line functions, use MATLAB® MEX functions.

References

Cherian, V., Shenoy, R., Stothert, A., Shriver, J. et al., "Model-Based Design of a SUV Anti-rollover Control System" SAE Technical Paper 2008-01-0579, 2008, doi:10.4271/2008-01-0579.

More About

- "Introducing MEX-Files"

Speeding Up Design Optimization Using Parallel Computing

When you have the Parallel Computing Toolbox™ software, you can use parallel computing to speed up parameter estimation and response optimization. When you use parallel computing, the software distributes the independent simulations on multiple MATLAB sessions. Thus, the simulations run in parallel which reduces the optimization time.

Using parallel computing may reduce the optimization time in the following cases:

- The model contains a large number of parameters to optimize, and the `Gradient descent` or `Nonlinear least squares` method is selected.
- The`Pattern search` method is selected as the optimization method.
- The model contains a large number of uncertain parameters and uncertain parameter values.
- The model is complex and takes a long time to simulate.

Required and Related Products

Simulink Design Optimization software requires MATLAB, Simulink, and Optimization Toolbox™ software.

The following table summarizes MathWorks® products that extend and complement the Simulink Design Optimization software. For current information about these and other MathWorks products, visit http://www.mathworks.com/products/.

Product	Description
Control System Toolbox	Enables you to design controllers for linear time-invariant (LTI) models using optimization methods.
Global Optimization Toolbox	Provides genetic algorithms, and direct search methods to estimate and optimize model parameters.
Neural Network Toolbox	Provides Simulink models of neural networks for optimization-based control design.
Parallel Computing Toolbox	Enables parallel computing on multicore processors and multiprocessor networks to speed up estimation and optimization.
Simulink Control Design	Lets you linearize Simulink models. Use Simulink Design Optimization software to design controllers for linearized models using optimization methods.
System Identification Toolbox	Lets you estimate linear and nonlinear models from measured data. Import the estimated model into Simulink software, and use Simulink Design Optimization software for optimization-based control design.

2

Parameter Estimation

Supported Data

From signal data, you can estimate model parameters and initial conditions of single or multiple input and output Simulink models.

Simulink Design Optimization software lets you estimate model parameters from the following types of data:

- *Time-domain* data — Data with one or more input variables *u(t)* and one or more output variables *y(t)*, sampled as a function of time. See “Import Data”.
- *Time-series* data — Data stored in time-series objects. See “Time-Series Data”.

Simulink Design Optimization software estimates model parameters by comparing the measured signal data with simulation data generated from the Simulink model. Using optimization techniques, the software estimates the parameters and initial conditions of states to minimize a user-selected cost function. The cost function typically calculates a least-square error between the measured and simulated data. To learn more, see “Estimate Parameters from Measured Data” on page 2-16.

More About

- “Time Series Objects”
- “Complex Data”

Prepare Data for Parameter Estimation

In this section...
"About This Tutorial" on page 2-3
"Start a Parameter Estimation Tool Session" on page 2-4
"Create an Experiment for Parameter Estimation" on page 2-6
"Import Data" on page 2-7
"Analyze Data" on page 2-7
"Extract Data for Estimation" on page 2-9
"Replacing Outliers" on page 2-10
"Filtering Data" on page 2-12
"Saving the Session" on page 2-14

About This Tutorial

Objectives

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

Note: Simulink Design Optimization software estimates parameters from real, time-domain data only.

Perform the following tasks using the Parameter Estimation tool:

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

About the Sample Data

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

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

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

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

`spe_engine_throttle1` contains the Simulink model of the engine throttle system.

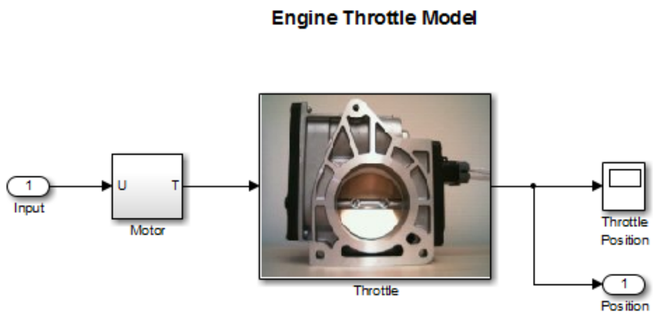
Start a Parameter Estimation Tool Session

To perform parameter estimation, you must first start a Parameter Estimation tool session.

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

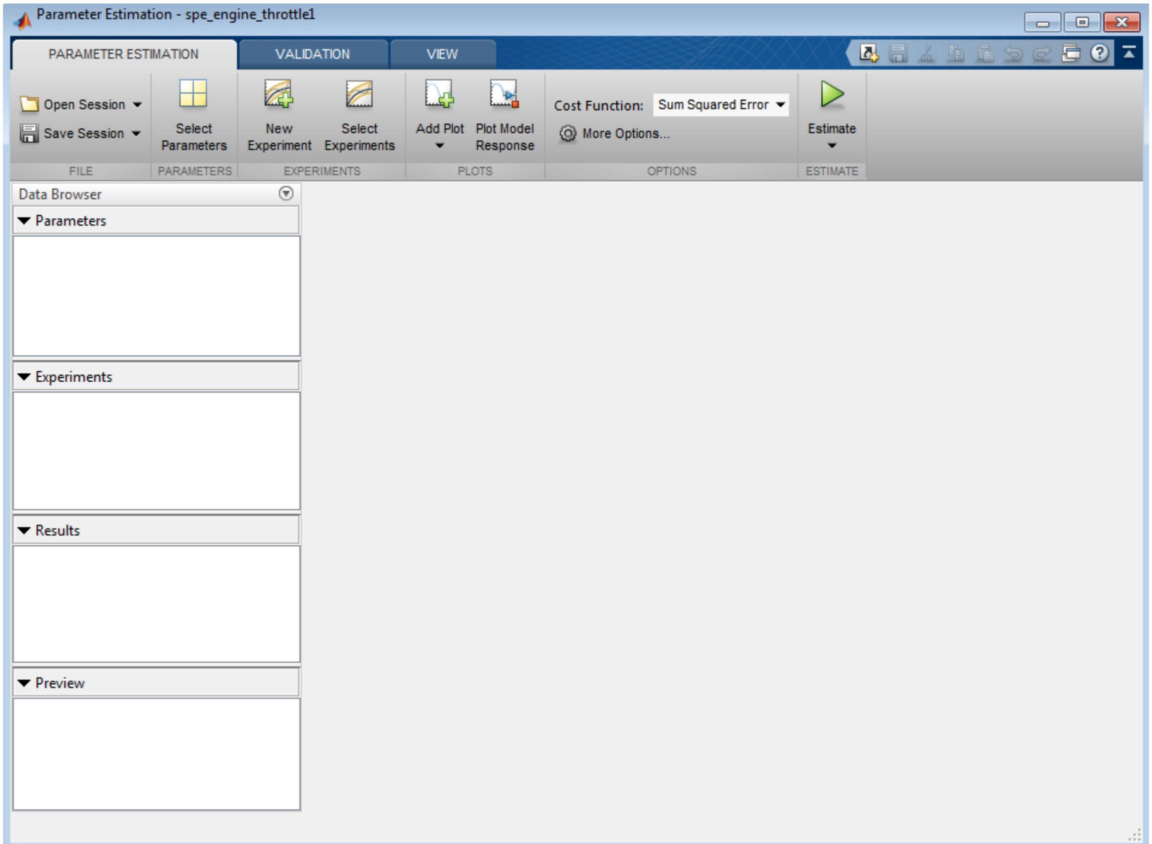
   ```
   spe_engine_throttle1
   ```

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



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

This action opens a new session, **Parameter Estimation - spe_engine_throttle1**, in the Parameter Estimation tool.



Note: The Simulink model must remain open to perform parameter estimation tasks.

Create an Experiment for Parameter Estimation

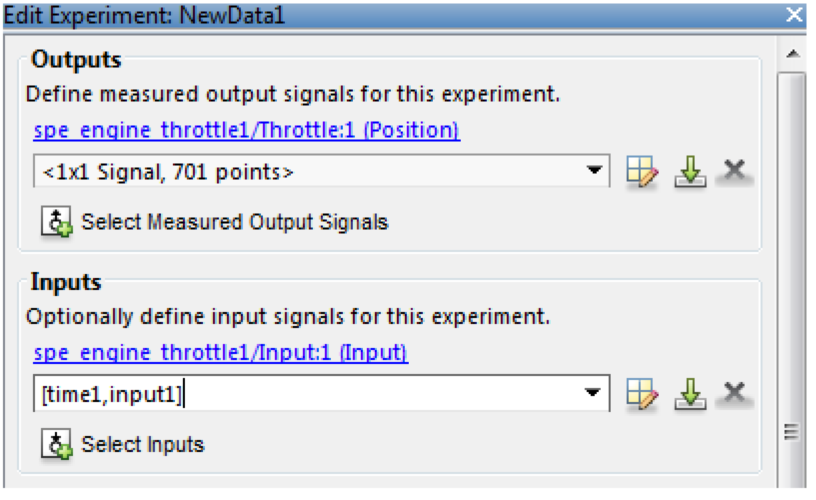
In the Parameter Estimation tool on the **Parameter Estimation** tab, click the **New Experiment** button. This will create an experiment with the name `Exp` in the **Experiments** list on the left pane. You can rename it by right-clicking and selecting **Rename** from the list. For example, call it `NewData1`.

Import Data

This portion of the tutorial explains how to import measured I/O data into the experiment in the Parameter Estimation tool. Importing data assigns the data to the corresponding model input and output signals.

The model input and output signals are designated with the Inport `Input` and Outport `Position` blocks, respectively, as shown in the figure. To learn more about the blocks, see the “Inport” and “Outport” block reference pages in the Simulink documentation.

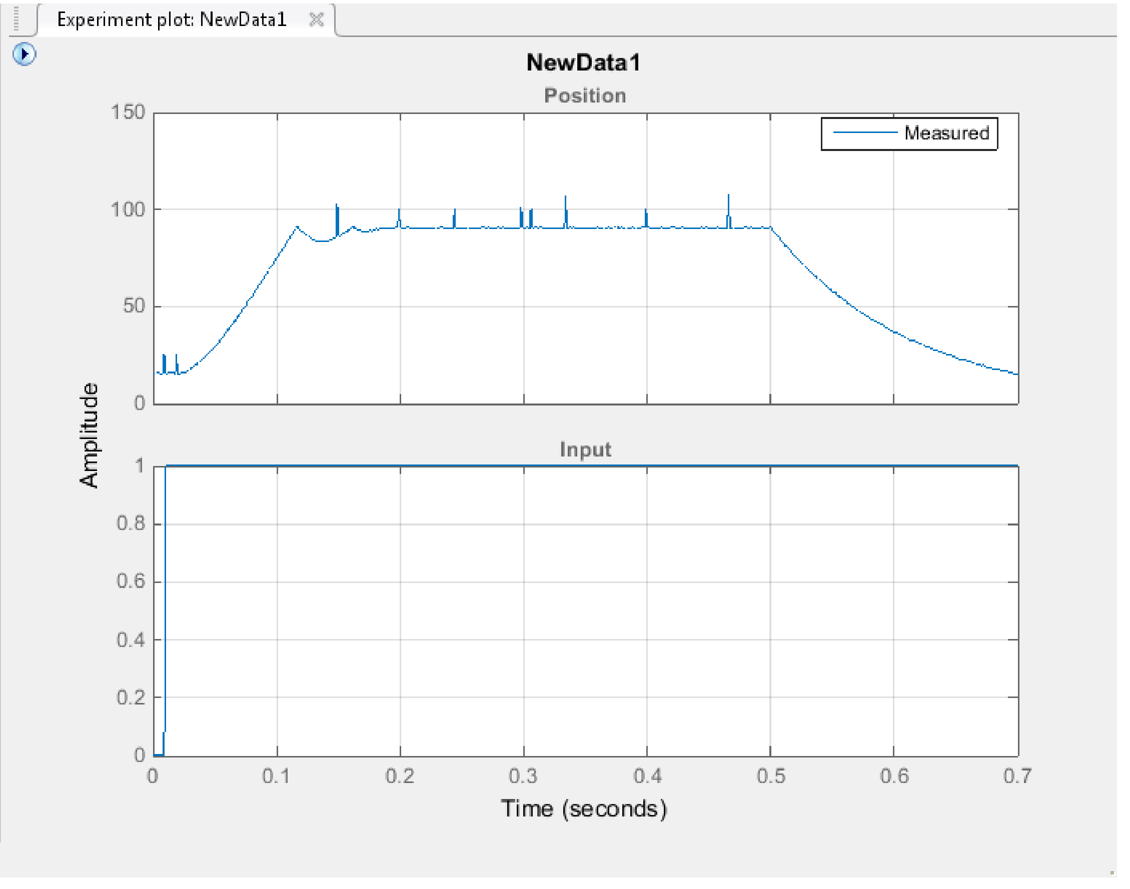
To import data into the experiment, right-click and select **Edit...** to launch the experiment editor. Import the output data by typing `[time1,position1]` in the dialog box in the **Outputs** panel. Import the input data by typing `[time1,input1]` in the dialog box in the **Inputs** panel.



Analyze Data

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

To create an experiment plot, click **Add Plot** on the **Parameter Estimation** tab and select the experiment name, for example, `NewData1` under **Experiment Plots**.



The time plot shows the output data in response to a step input, as described in “About the Sample Data” on page 2-4. The plot shows a rapid decrease in the response after t = 0.5 s because the system is shut down. To focus parameter estimation on the time period when the system is active, select the data samples between t = 0 s and t = 0.5 s, as in “Extract Data for Estimation” on page 2-9 .

The spikes in the data indicate *outliers*, defined as data values that deviate from the mean by more than three standard deviations. They might be caused by measurement errors or sensor problems. The response also contains noise. Before estimating model parameters from this data, remove the outliers and filter the noise, as described in "Replacing Outliers" on page 2-10and "Filtering Data" on page 2-12.

You can also plot the experiment data by right-clicking the experiment, for example `NewData1`, and selecting **Plot measured experiment data** from the list.

Extract Data for Estimation

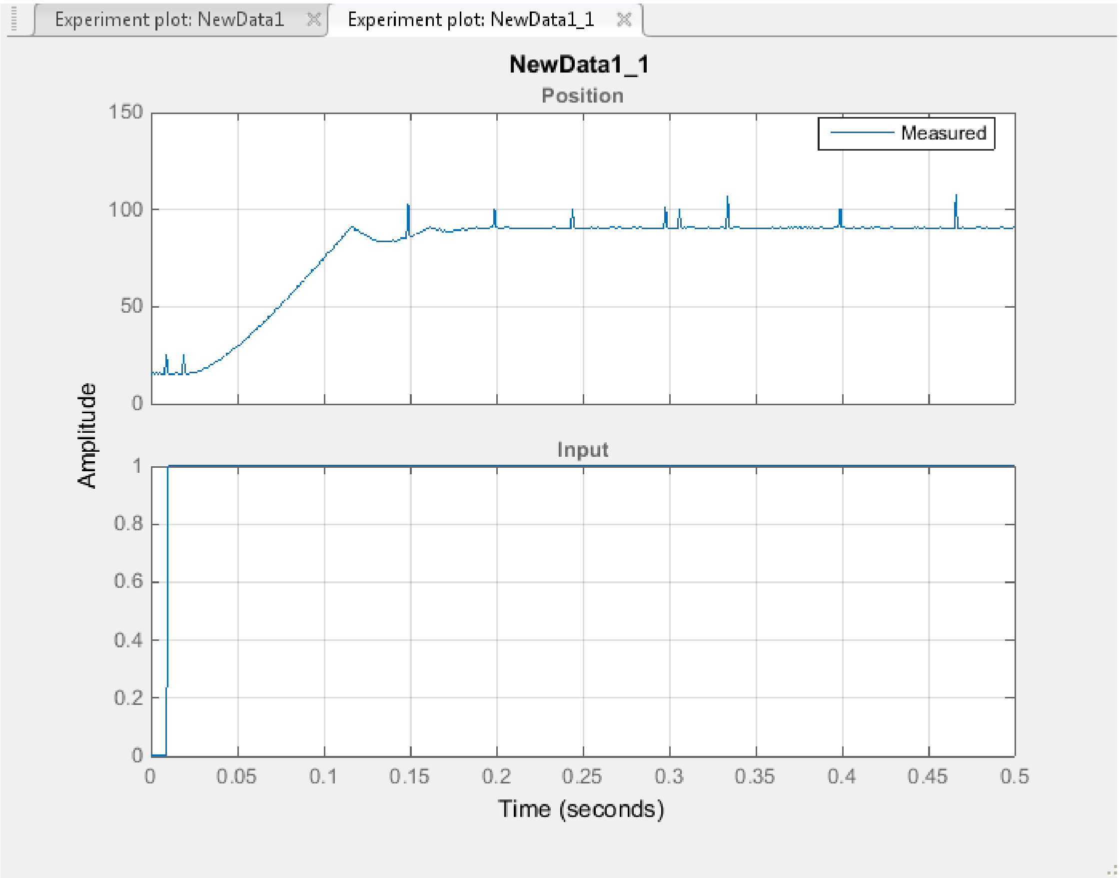
This portion of the tutorial explains how to select a subset of I/O data for estimation. As described in "Analyze Data" on page 2-7, the system is shut down at t = 0.5 s. To focus the estimation on the time period before t = 0.5 s, exclude the data samples beyond t = 0.5 s. This operation selects the data between t = 0 s and t = 0.5 s for estimation.

First, import the data into the experiment, as described in "Import Data" on page 2-7.

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

1. Plot the measured data as described in "Analyze Data" on page 2-7, to have access to the **Experiment Plot** tab.
2. On the **Experiment Plot** tab,click **Extract Data** to launch the **Extract Data** tab.
3. Enter 0 in the **Start Time:** field and 0.5 in the **End Time:** field.
4. Click **Save As** to save data in a new experiment, for example, `NewData1_1`.

The Parameter Estimation tool extracts the corresponding input data. To plot the new data, click on **Add Plot** on the **Parameter Estimation** tab. Select the experiment name, for example, `NewData1_1` in the **Experiment Plots** list to display the experiment plot of the data from t = 0 s to t = 0.5 s.



Replacing Outliers

- "Why Replace Outliers" on page 2-10
- "How to Replace Outliers" on page 2-11

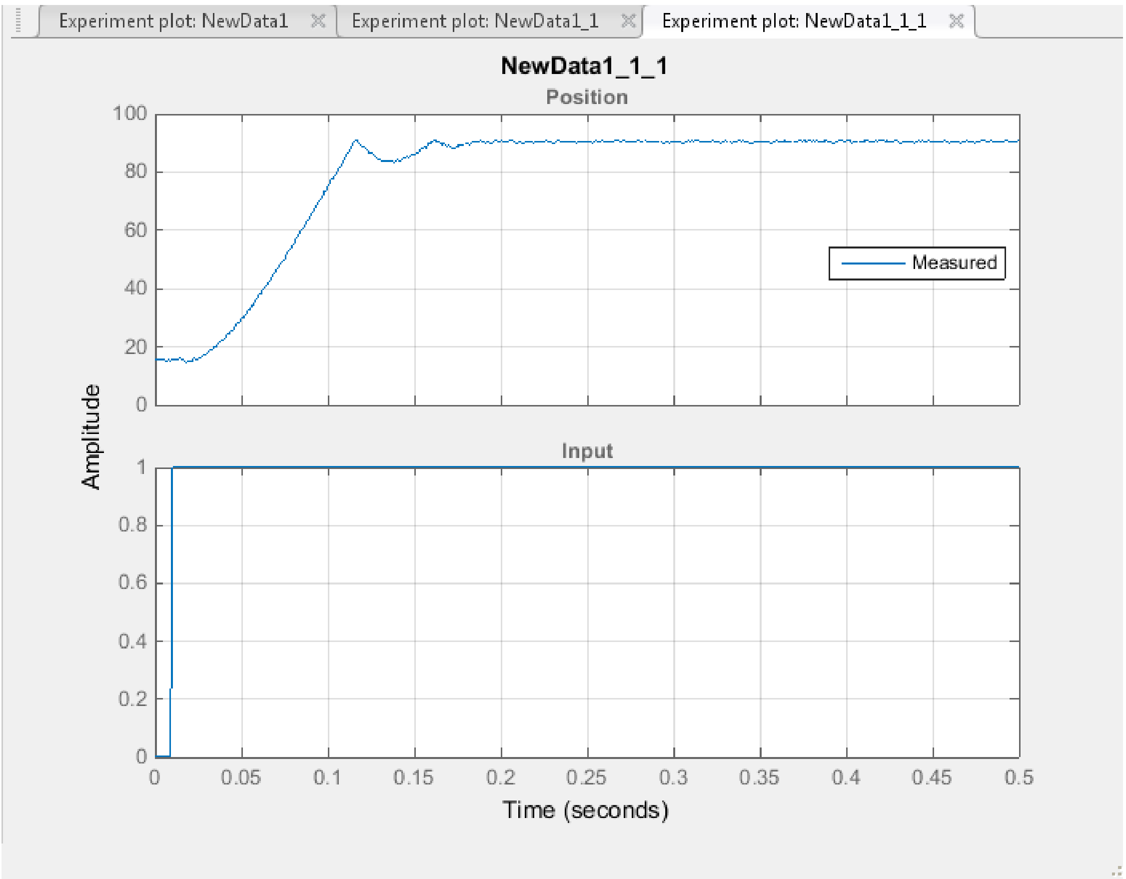
Why Replace Outliers

Outliers are data values that deviate from the mean by more than three standard deviations. When estimating parameters from data containing outliers, the results might not be accurate. Hence, you might choose to replace the outliers in the data before you estimate the parameters.

How to Replace Outliers

In the experiment plot of the data extracted as in "Extract Data for Estimation" on page 2-9, you can visually detect the data points that seem to be outliers. To replace these points:

1. In the **Experiment Plot** tab, click **Replace data** to launch the **Replace data** tab. The experiment plot shows the preview data, which is in light brown. On the preview, select the data point that you want to replace.
2. On the **Replace Data** tab, click **Replace data** and select the constant value. For example, replace the output signal data that correspond to time points 0.00899 and 0.0189 with 15, that corresponds to the time point 0.149 with 86, and the rest of the outlier data points with 90.
3. Click the arrow in the **Update** section and select **Save As: Create a new experiment from the modified data**. Parameter Estimation tool saves the modified data in the new experiment, for example, `NewData1_1_1`.
4. Click **Add Plot** on the **Parameter Estimation** tab and select the new experiment, for example, `NewData1_1_1`. This creates an experiment plot of the modified data. The spikes, that indicated outliers, no longer appear on the time plot.



Filtering Data

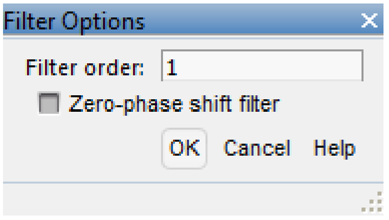
This portion of the tutorial explains how to filter the noise and remove any periodic trends from the output data. First remove the outliers from the output data, as described in “Replacing Outliers” on page 2-10.

Click the experiment plot for the new experiment, for example, `NewData1_1_1`. On the **Experiment Plot** tab, click **Low-Pass Filter**.

1 On the **Low-Pass Filter** tab select **Filter all signals**.

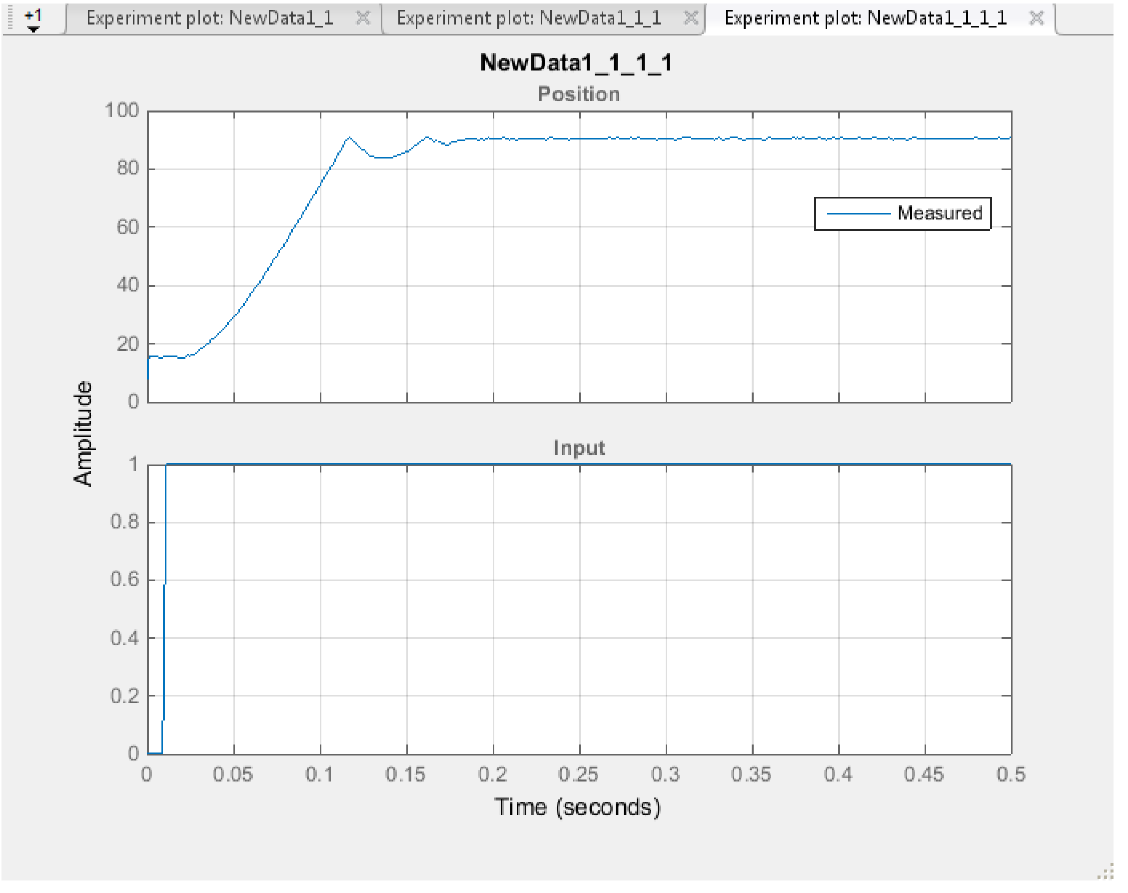
2 Enter 0.4 in the **Normalized cutoff frequency** field.

3 Click **Options**. Enter 1 in the **Filter order** field and click **OK**.



4 Click the arrow in the **Update** section and select **Save As: Create a new experiment from the modified data**. Parameter Estimation tool saves the modified data in the new experiment, for example, `NewData1_1_1_1`.

5 Click **Add Plot** on the **Parameter Estimation** tab and select the new experiment, `NewData1_1_1_1`. This creates an experiment plot of the modified data. The noise is filtered and the output data appears smooth.



Saving the Session

After you prepare the data, delete the data in the older experiments, for example, `New Data1`, `New Data1_1`, `New Data1_1_1`. You can rename the last experiment, for example, `NewData1_1_1_1` as `NewData1`, and save the session.

To delete the experiments, right-click the experiment name in the **Experiments** pane, and select **Delete** from the list.

To save the session, click **Save Session** on the **Parameter Estimation** tab to select where to save the session. Specify a name for the session, for example, `spe_engine_throttle1_sdosession.mat` in the **File name** or **Session** field, and then click **Save** or **OK**. This saves your parameter estimation session as a MAT-file.

To learn how to estimate parameters from this data, see "Estimate Parameters from Measured Data" on page 2-16.

Estimate Parameters from Measured Data

In this section...
"About This Tutorial" on page 2-16
"Estimate Model Parameters Using Default Estimation Settings" on page 2-19
"Improve Estimation Results Using Parameter Bounds" on page 2-29
"Validate Estimated Model Parameters" on page 2-32

About This Tutorial

Objectives

This tutorial shows how to estimate parameters of a single-input single-output (SISO) Simulink model from measured input and output (I/O) data.

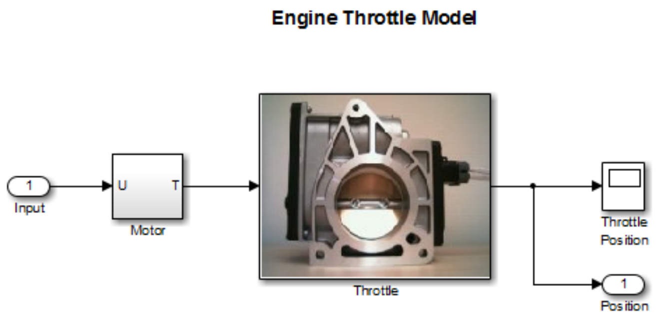
Note: Simulink Design Optimization software estimates parameters from real, time-domain data only.

You can perform the following tasks using the Parameter Estimation tool:

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

About the Model

This tutorial uses the `spe_engine_throttle1` Simulink model, which represents an engine throttle system.

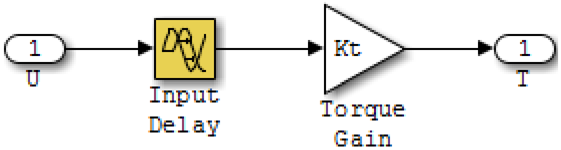


The throttle system controls the flow of air and fuel mixture to the engine cylinders. The throttle body contains a butterfly valve that opens when a driver presses the accelerator pedal. Opening this valve increases the amount of fuel mixture entering the cylinders, which increases the engine speed. A DC motor controls the opening angle of the butterfly valve in the throttle system. The models for these components are described in "Motor Subsystem" on page 2-17 and "Throttle Subsystem" on page 2-18.

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

Motor Subsystem

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

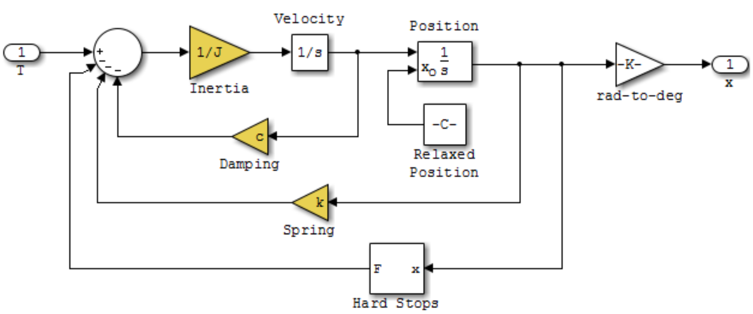


Components of the `Motor` subsystem	Description
Variables	*U* is the input current to the motor. *T* is the torque applied by the motor.
Parameters	K_t is the torque gain of the motor, represented by `Kt` in the model.

Components of the Motor subsystem	Description
	t_d is the input time delay of the motor, represented by input_delay in the model.
Equation	The torque applied by the motor is described in the following equation: $$T(t) = K_t U(t - t_d)$$ where t is time.
Input	U
Output	T

Throttle Subsystem

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



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

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

Components of the Throttle subsystem	Description
Variables	T is the torque applied by the DC motor. θ is the angular position of the valve, represented by x in the model.

Components of the `Throttle` subsystem	Description
	$T_{hardstop}$ is the torque applied by the hard stop.
Parameters	`J` is the valve inertia. `c` is the valve viscous friction. `k` is the valve spring constant.
States	θ is the angular position. $\dot{\theta}$ is the angular velocity.
Equations	The mathematical system for the butterfly valve is described in the following equation: $$J\ddot{\theta} + c\dot{\theta} + k\theta = T + T_{hardstop}$$ where $15^\circ \le \theta \le 90^\circ$, with initial conditions $\theta_0 = 15^\circ$, and $\dot{\theta}_0 = 0$. The torque applied by the `Hard Stops` block is described in the following equation: $$T_{hardstop} = \begin{cases} 0, & 15^\circ \le \theta \le 90^\circ \\ K(90^\circ - \theta), \theta > 90^\circ & \\ K(15^\circ - \theta), \theta < 15^\circ & \end{cases}$$ where `K` is the gain of the `Hard Stops` block.
Input	T
Output	θ

Estimate Model Parameters Using Default Estimation Settings

- "Overview of the Estimation Process" on page 2-20
- "Specify Estimation Data and Parameters" on page 2-20

Overview of the Estimation Process

Simulink Design Optimization software uses optimization techniques to estimate model parameters. In each optimization iteration, it simulates the model with the current parameter values. It computes and minimizes the error between the simulated and measured output. The estimation is complete when the optimization method finds a local minimum.

To start the estimation process, first open the engine throttle system Simulink model by typing the following at the MATLAB prompt:

```
spe_engine_throttle1
```

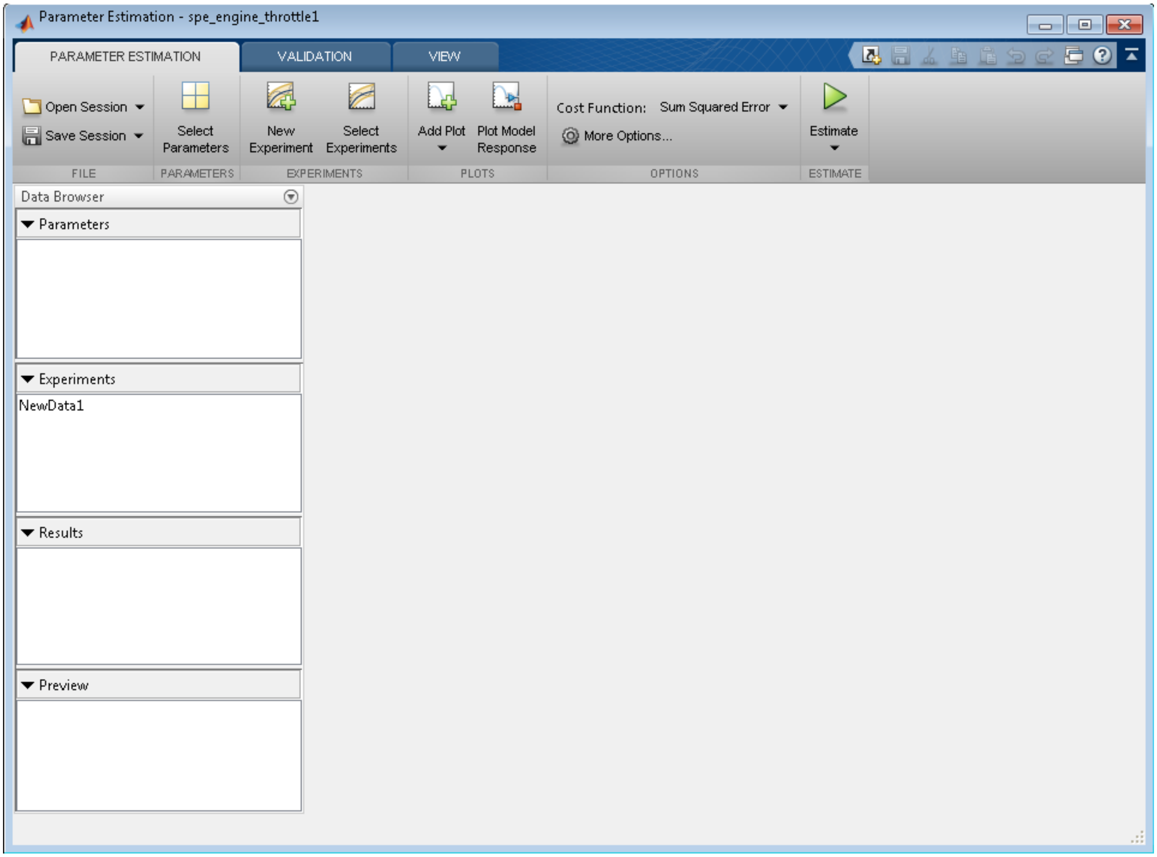
In the Simulink Editor, select **Analysis** > **Parameter Estimation**.

This action opens a new session with the name **Parameter Estimation - spe_engine_throttle1** in the Parameter Estimation tool.

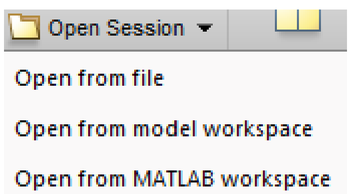
Note: The Simulink model must remain open to perform parameter estimation tasks.

Specify Estimation Data and Parameters

1 Load or import the estimation data.



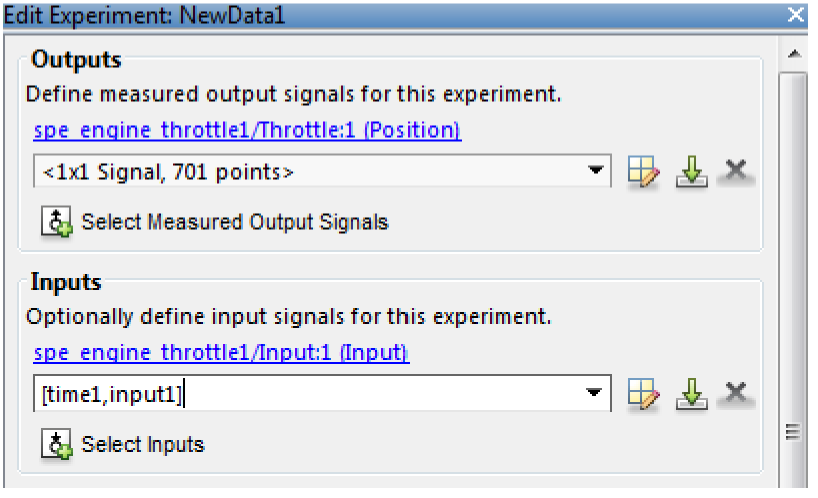
a If you prepared data and saved your session as described in “Prepare Data for Parameter Estimation” on page 2-3, load the preconfigured session. On the **Parameter Estimation** tab, click the `Open Session` drop down list.



Select the correct option to browse to the location of your saved session, for example, `Open from file`. Then select the MAT-file.

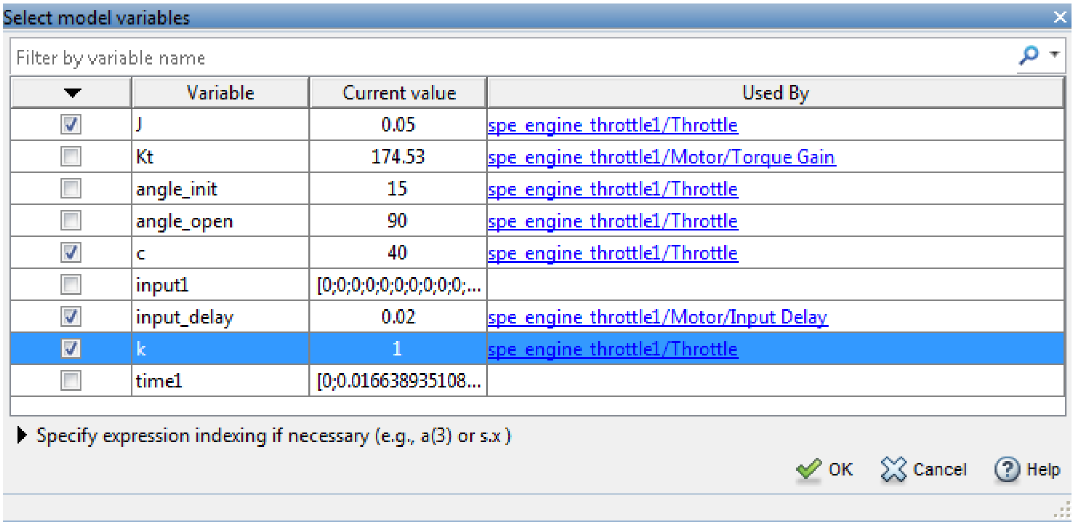
b If you do not have a previously saved session, create a new experiment. on the **Parameter Estimation** tab, click **New Experiment** . In the **Experiments** list on the left pane. You can rename it by right-clicking and selecting **Rename** from the list. For example, call it `NewData1`.

To import data into the experiment, right-click and select **Edit...** to launch the experiment editor. Import the output data by typing in the dialog box in the **Outputs** panel, for example `[time1,position1]`. Import the input data by typing in the dialog box in the **Inputs** panel, for example `[time1,input1]`.

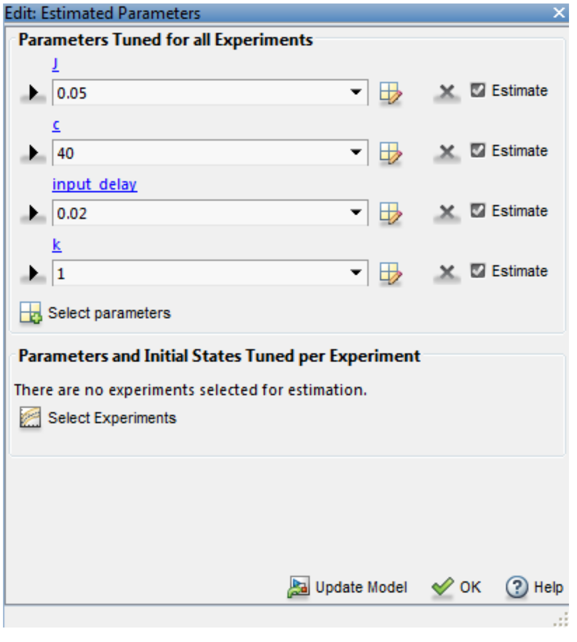


2 Specify parameters for estimation. On the **Parameter Estimation** tab, click the **Select Parameters** button to open the **Edit:Estimated Parameters** dialog box. In the **Parameters Tuned for all Experiments** panel, click the **Select parameters** button to launch the **Select Model Variables** dialog box.

Select the parameters `J`, `c`, `input_delay`, and `k`, and click **OK**.

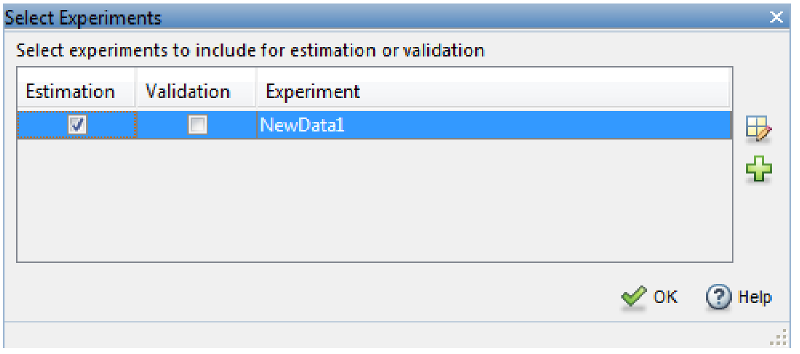


The **Edit:Estimated Parameters** window now looks as follows.



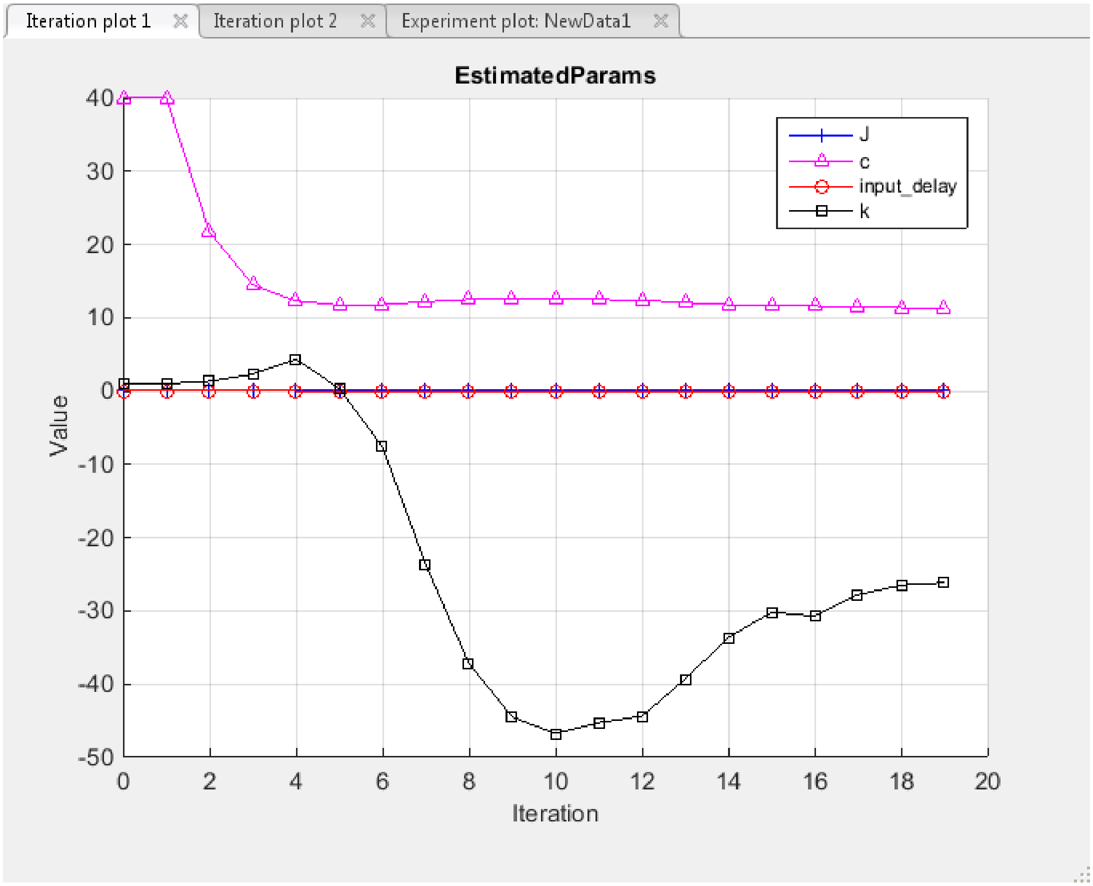
The tool selects the parameters you add for estimation by default. When estimating a large number of parameters, you can first select a subset of parameters to estimate.

3 Specify an experiment for estimation. On the **Parameter Estimation** tab, click **Select Experiments**, and select the box under the **Estimation** column. Click **OK**.

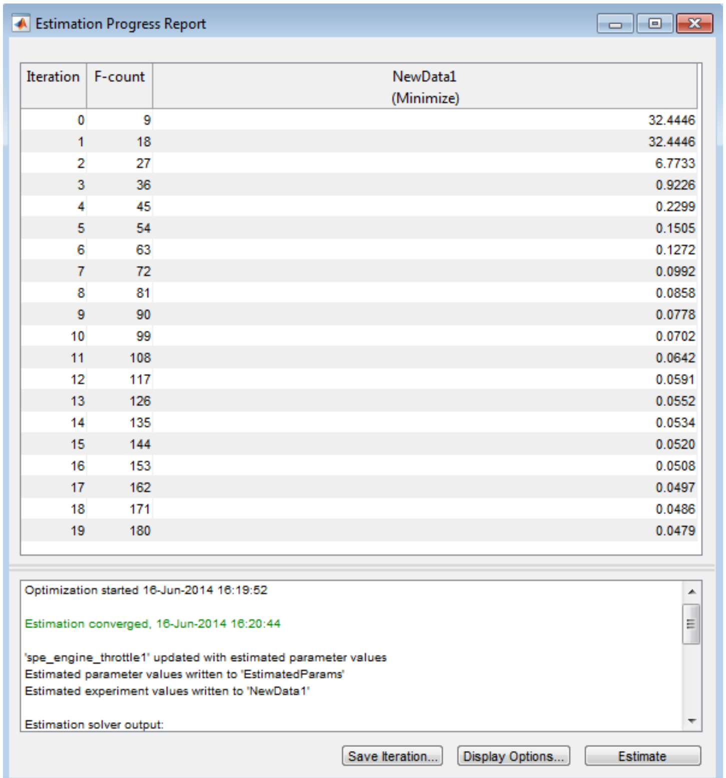


4 To add progress plots, click **Add Plot** on the **Parameter Estimation** tab. Here you can choose the **Parameter Trajectory** and **Estimation Cost** iteration plots. You can also choose an experiment plot of measured and simulated data for `NewData1`.

5 Estimate the parameters using the default settings. On the **Parameter Estimation** tab, click **Estimate** to open the **Parameter Trajectory** plot and **Estimation Progress Report** window and estimate the parameters. The **Parameter Trajectory** plot shows the change in the parameter values at each iteration.

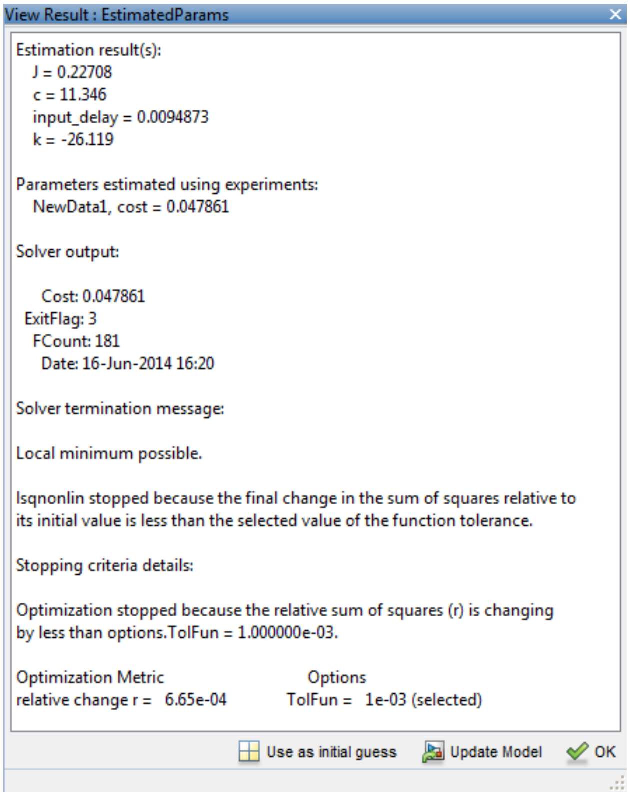


The **Estimation Progress Report** shows the iteration number, number of times the objective function is evaluated, and the value of the cost function at the end of each iteration. After the estimation converges, the **Estimation Progress Report** looks like this figure.

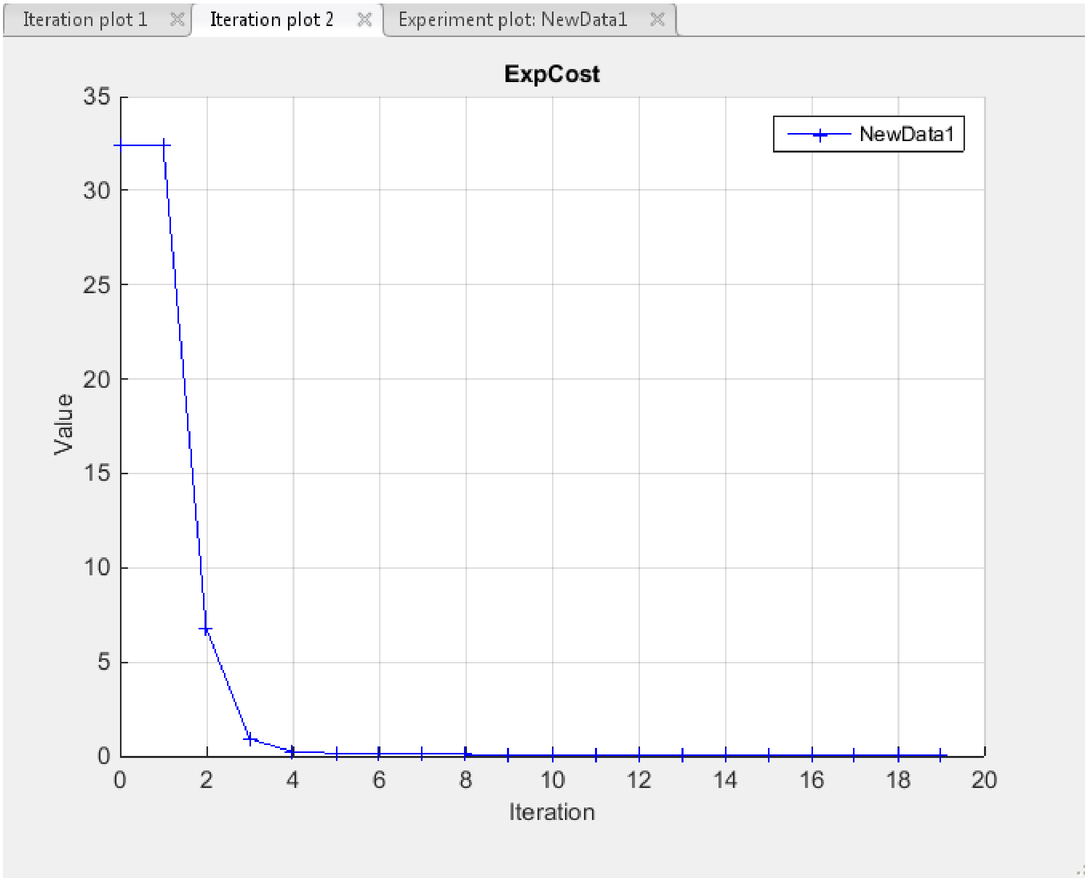


Iteration	F-count	NewData1 (Minimize)
0	9	32.4446
1	18	32.4446
2	27	6.7733
3	36	0.9226
4	45	0.2299
5	54	0.1505
6	63	0.1272
7	72	0.0992
8	81	0.0858
9	90	0.0778
10	99	0.0702
11	108	0.0642
12	117	0.0591
13	126	0.0552
14	135	0.0534
15	144	0.0520
16	153	0.0508
17	162	0.0497
18	171	0.0486
19	180	0.0479

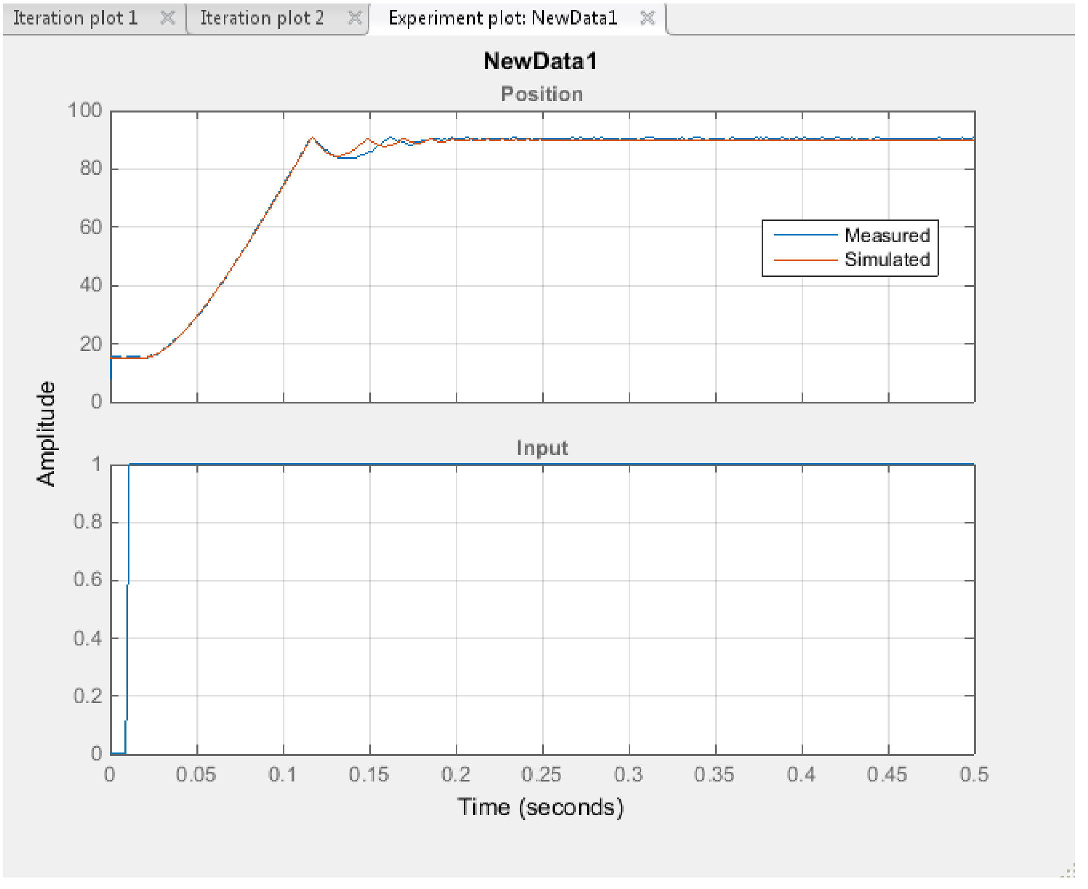
The estimated parameters are saved in the Parameter Estimation tool, in the **Results** section of the **Data Browser** pane, as `EstimatedParams`. Right-click `EstimatedParams`, and select **Open...** to view the results.



6 Examine the estimated cost function graph. Cost function is the error between the simulated and measured output. During estimation, the default optimization method `Nonlinear least squares`, "lsqnonlin", minimizes the cost function by changing the parameter values. The following figure displays the change in the expected cost during iterations.



7 Examine the simulated response plot to see how well the simulated output matches the measured output. The experiment plot shows that the output simulated using the estimated parameters is close to the measured outputs.



Improve Estimation Results Using Parameter Bounds

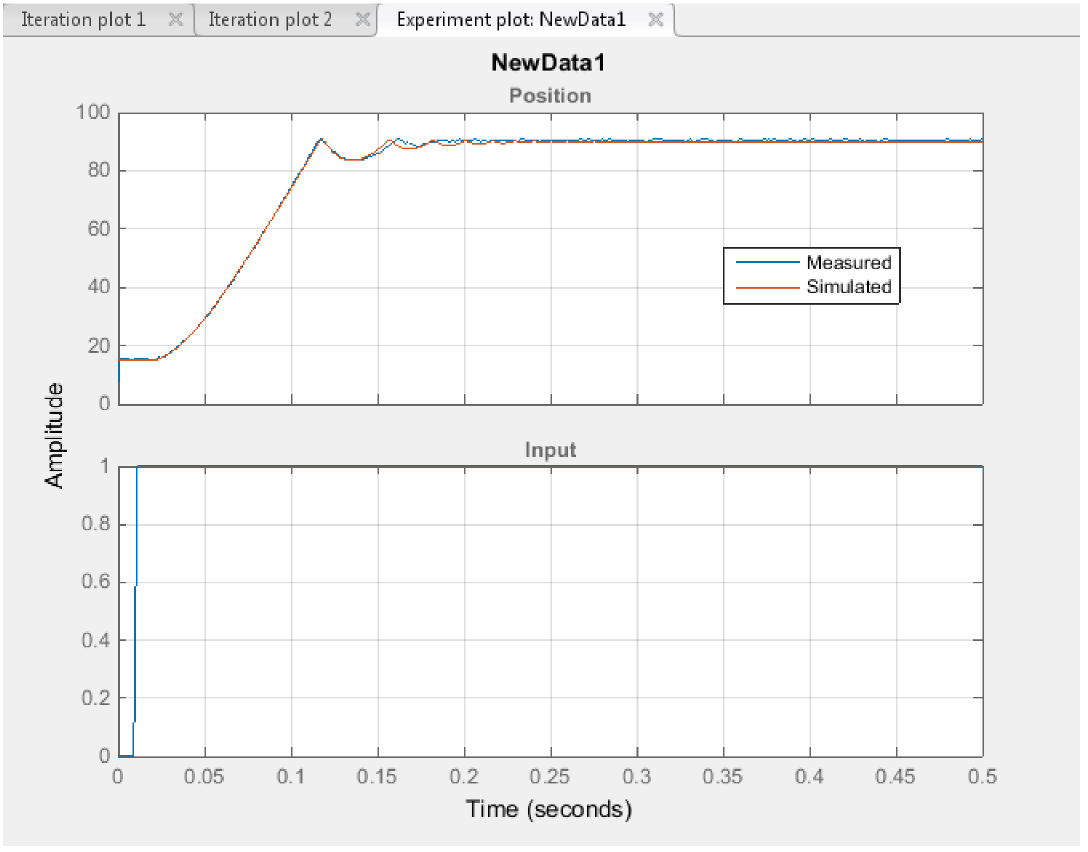
You can improve the accuracy of estimation by specifying bounds on parameter values. This technique restricts the region in which the optimization method searches for a local minima.

The engine throttle system has these characteristics:

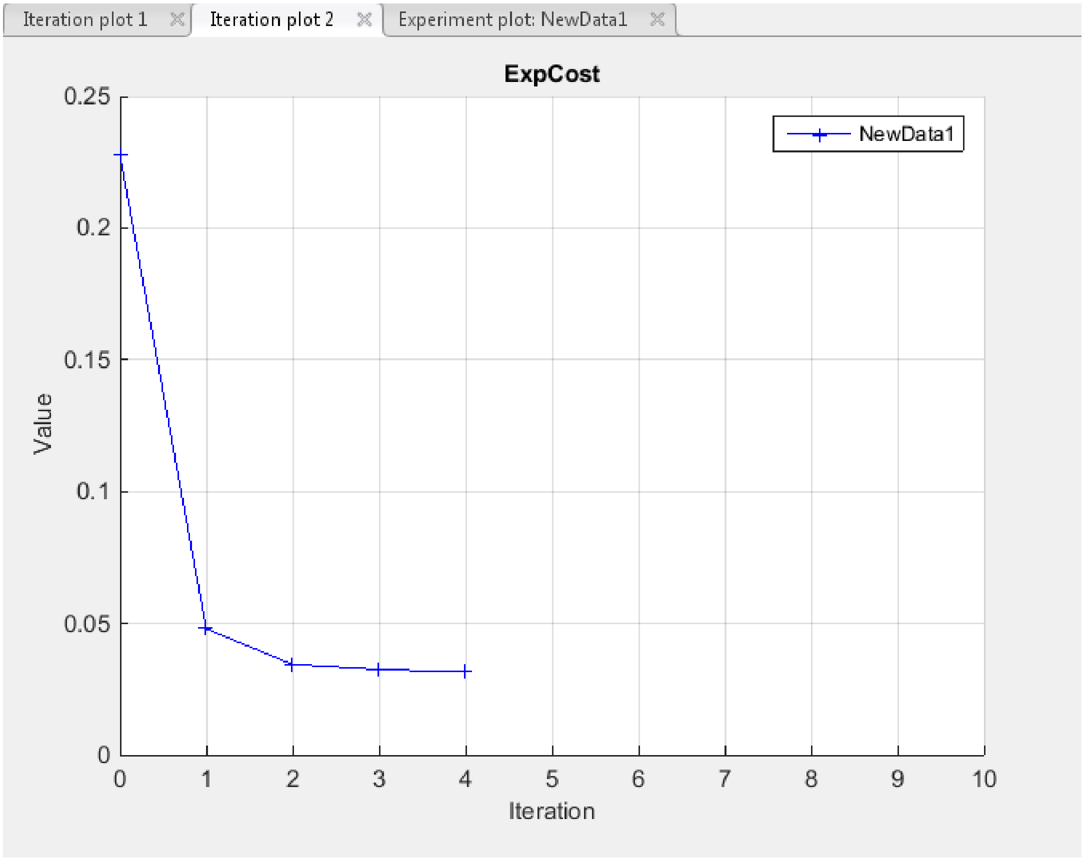
- All parameter values are positive.
- Maximum time delay of the system, represented by `input_delay`, is 0.1 s.

Therefore, specify 0 as the minimum value for all parameters, and 0.1 as the maximum value of `input_delay`. In the Parameter Estimation tool, click the **Select Parameters** button to specify bounds on the parameter values. For each parameter, click the right arrow toggle to display the minimum, maximum, and scale fields. Specify the minimum value for each parameter by replacing `-Inf` with `0` in the **Minimum** field. Specify the maximum value for `input_delay` by replacing `+Inf` with `0.1` in the corresponding **Maximum** field.

After estimating the parameters, analyze the results using the experiment plot and the plot for expected cost.



The data simulated using the estimated parameter values agree better with the measured data than when the parameter limits were not specified.



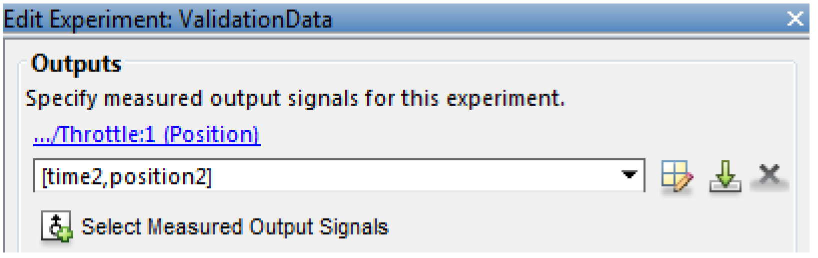
Validate Estimated Model Parameters

After estimating model parameters, validate the model using another data set *(validation data)*. A good match between the simulated response and the validation data indicates that you have not overfitted the model.

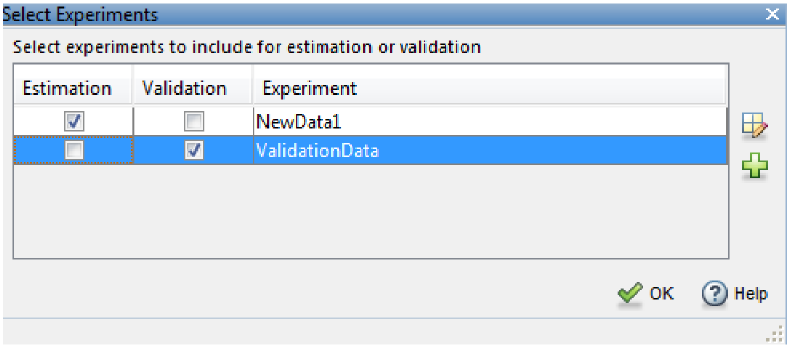
To validate the estimated parameters using a validation data set:

1 Create a new experiment to use for validation. Name it `ValidationData`. Import the validation I/O data, `input2` and `position2`, and the time vector, `time2` in the `ValidationData` experiment. To do this, in the Parameter Estimation tool, in the Experiments pane, right-click `ValidationData` and select **Edit...** to open the

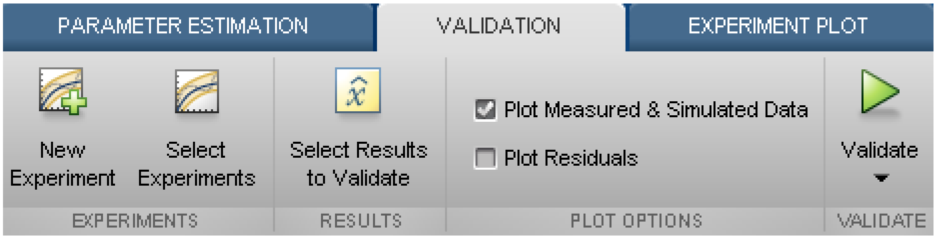
experiment editor. Then, type `[time2,position2]` in the output dialog box and `[time2,input2]` in the input dialog box. For more information, see “Import Data”.



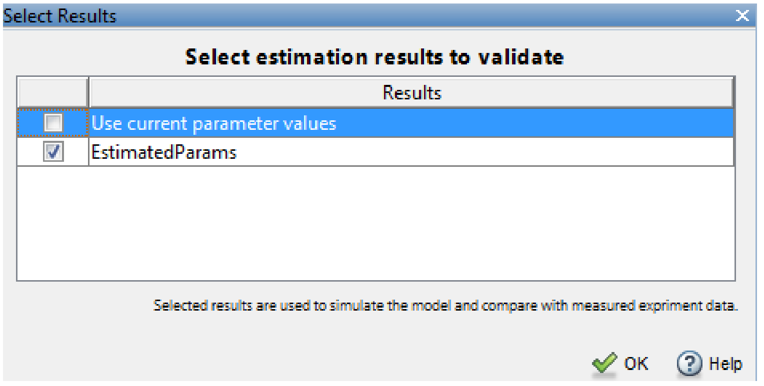
2 Select the experiment for validation. On the **Parameter Estimation** tab, click **Select Experiments**. By default, the `ValidationData` experiment is selected for estimation. Deselect the check box that corresponds to `ValidationData` for estimation and select the check box for validation.



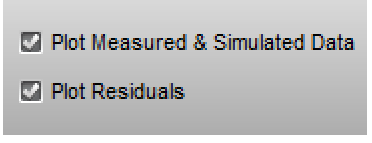
3 Select results to use. On the **Validation** tab, click **Select Results to Validate**.



Deselect `Use current parameter values` and select `EstimatedParams`, and click **OK**.



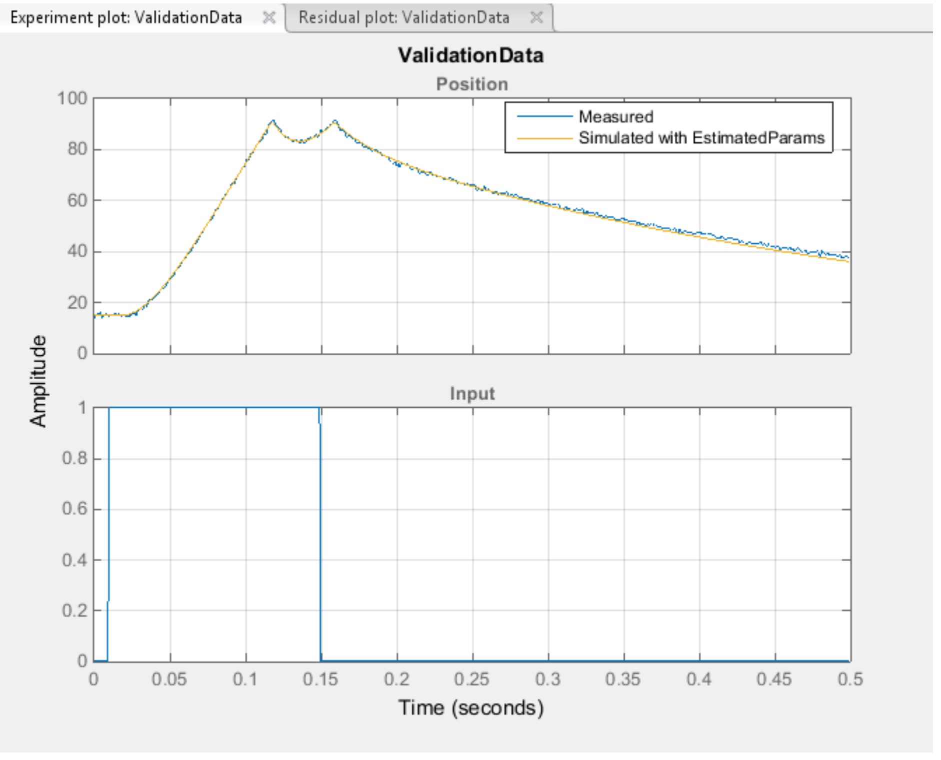
4 Select the plots for measured and simulated data, and residuals on the **Validation** tab. You can assess how much the data simulated using the estimated parameters agrees with the measured data using these plots.



On the **Validation** tab, click **Validate** to start validation.

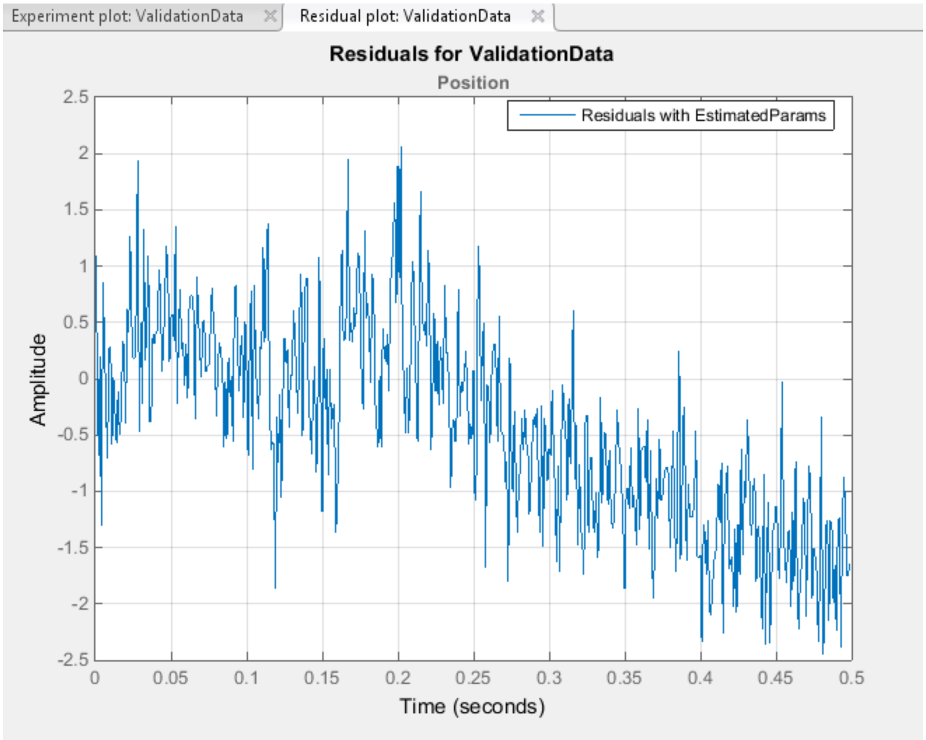
5 Examine the plots.

   a Examine the experiment plot to see how well the simulated output matches the output data.



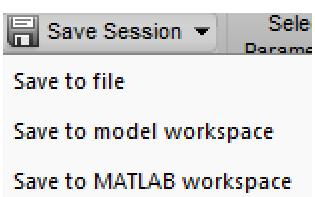
The simulated response as shown in light brown on the top experiment plot is overlaid on the measured out put data, and closely matches the measured validation data.

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



The difference between the simulated and measured data varies between 2 and -2.5. The residuals lie within 6% of the maximum output variation and do not display any systematic patterns. This indicates a good fit between the simulated output and measured data.

6 Save the session. On the **Parameter Estimation** tab, click **Save Session**.



From the drop-down list select where to save the session. Specify the file name, and click **Save** or **OK** to save your parameter estimation session as a MAT-file.

Response Optimization

Supported Design Requirements

You can optimize response of Simulink models to meet time- and frequency-domain design requirements.

Simulink Design Optimization software optimizes model response by formulating the requirements into a constrained optimization problem. It then solves the problem using optimization methods.

- For time-domain requirements, the software simulates the model during optimization, compares the current response with the requirement and uses gradient methods to modify design variables (model parameters) to meet the objectives.

  You can specify time-domain requirements either in blocks from the **Signal Constraints** library or without adding blocks to the model. You can also include requirements specified in Check Static Range, Check Static Lower Bound and Check Static Upper Bound blocks from the Simulink **Model Verification** library.
- For frequency-domain requirements, the software linearizes the portion of the model between specified linearization inputs and outputs, compares the linear system with the requirement and uses gradient methods to modify the design variables to meet the objectives.

  If you have Simulink Control Design software, you can optimize the model to meet frequency-domain requirements, such as Bode magnitude and gain and phase margin bounds. You can specify the frequency-domain requirements without adding blocks to the model or by using the “Model Verification” blocks of the Simulink Control Design software library.

Related Examples

“Design Optimization to Meet Step Response Requirements (GUI)” on page 3-4

“Design Optimization to Meet Step Response Requirements (Code)” on page 3-18

“Design Optimization to Track Reference Signal (GUI)” on page 3-25

“Design Optimization to Meet Frequency-Domain Requirements (GUI)”

“Design Optimization Using Frequency-Domain Check Blocks (GUI)” on page 3-40

"Design Optimization to Meet Time- and Frequency-Domain Requirements (GUI)"

More About

"How the Optimization Algorithm Formulates Minimization Problems"

"Specify Time-Domain Design Requirements"

"Specify Frequency-Domain Design Requirements"

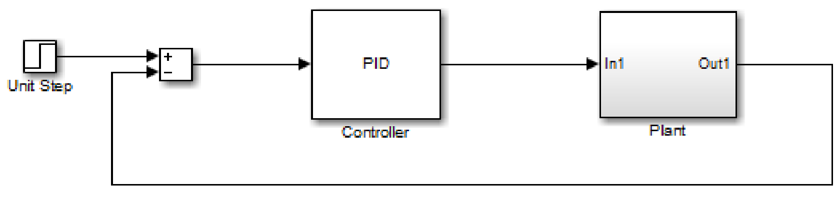
Design Optimization to Meet Step Response Requirements (GUI)

In this section...

This example shows how to optimize controller parameters to meet step response design requirements using the Design Optimization tool. You specify the design requirements in a Check Step Response Characteristics block.

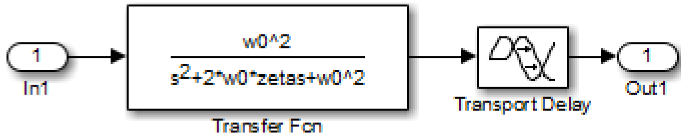
Model Structure

The model `sldo_model1` includes the following blocks:



- **Controller block**, which is a PID controller. This block controls the output of the `Plant` subsystem.
- **Unit Step block** applies a step input and produces the model output that should meet step response requirements.

  You can also use other types of inputs, such as ramp, to optimize the response to meet step response requirements generated by such inputs.
- **Plant subsystem** is a second-order system with delay. It contains “Transfer Function” and “Transport Delay” blocks.



Design Requirements

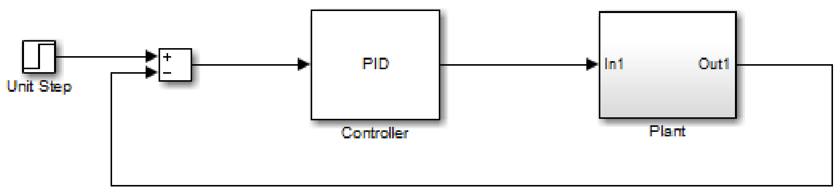
The plant output must meet the following step response requirements:

- Rise time less than 2.5 seconds
- Settling time less than 30 seconds
- Overshoot less than 5%

Specify Step Response Requirements

1 Open Simulink model.

```
sys = 'sldo_model1';
open_system(sys);
```

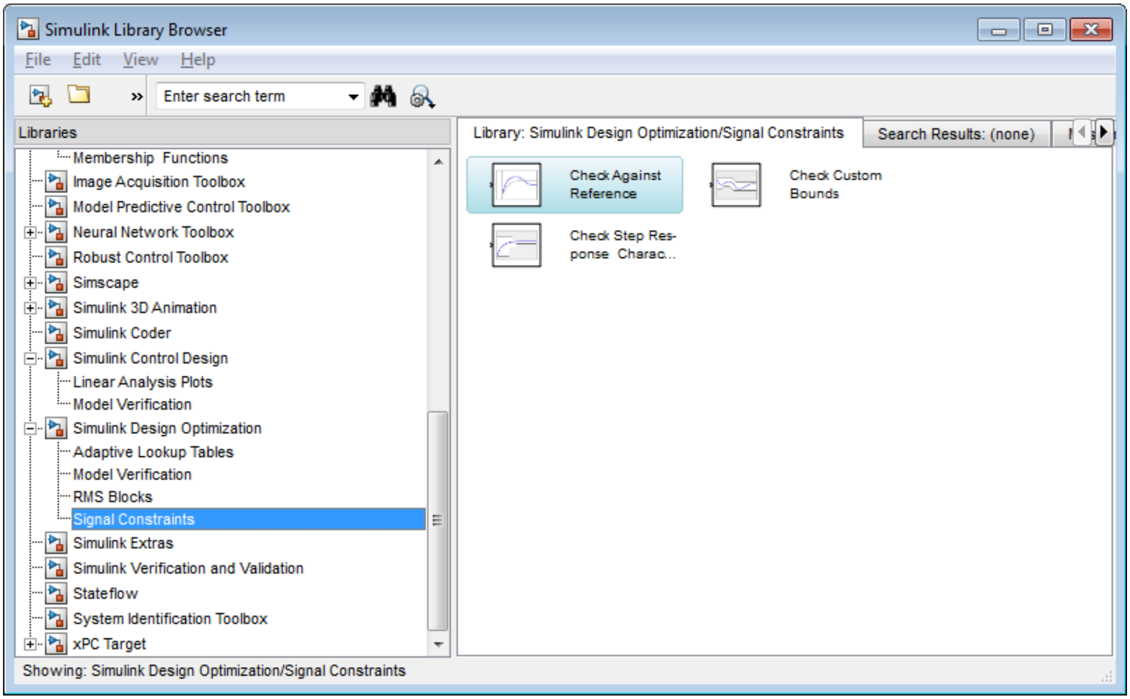


To learn more about the model, see "Model Structure" on page 3-4.

2 Add a Check Step Response Characteristics block to the model.

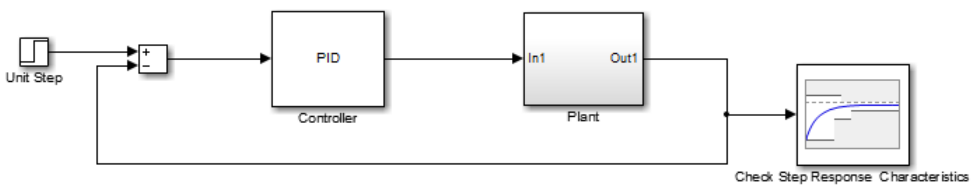
a In the Simulink model window, select **View** > **Library Browser**.

b In the **Libraries** pane, expand the **Simulink Design Optimization** node and select **Signal Constraints**.



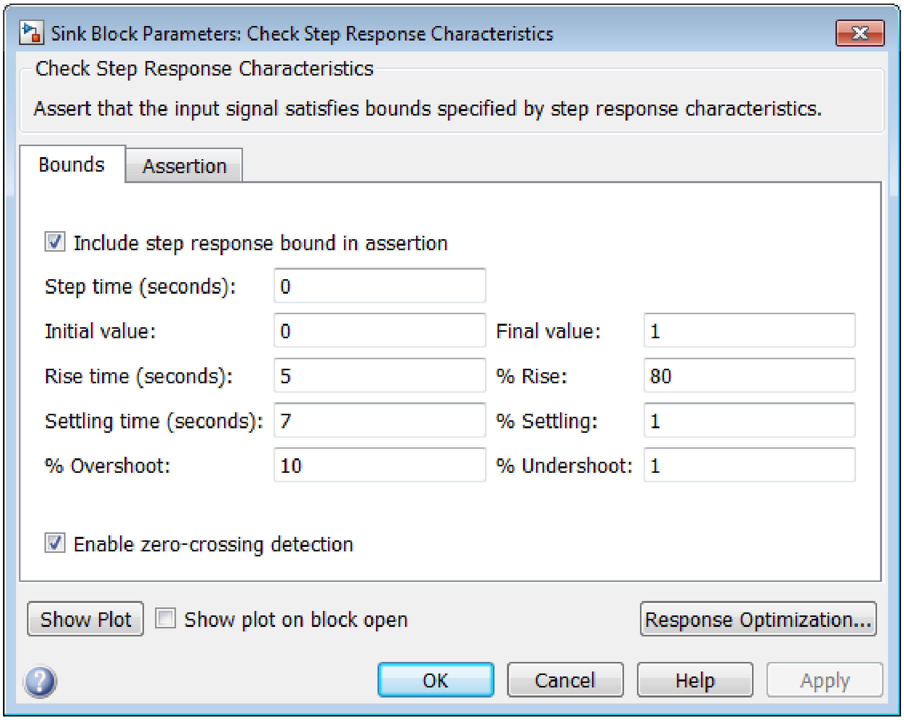
c Drag and drop the Check Step Response Characteristics block into the model window.

d Connect the block to the output.



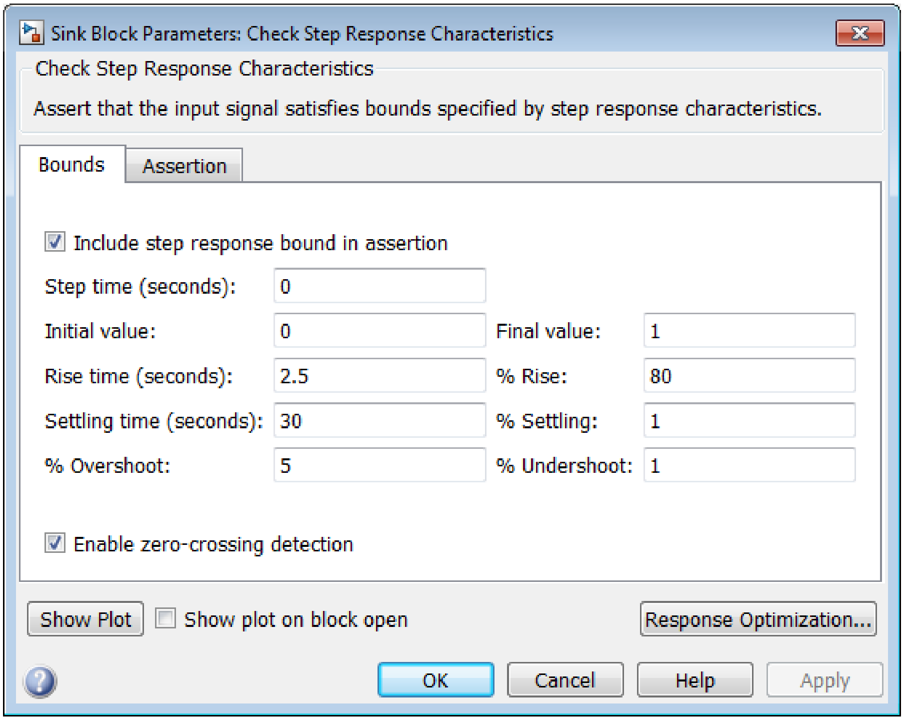
You must connect the block to the signal on which you want to specify design requirements.

3 Double-click the Check Step Response Characteristics block to open the Sink Block Parameters: Check Step Response Characteristics dialog box.



4 Specify step response requirements:

- In **Rise time (seconds)**, enter `2.5`
- In **Settling time (seconds)**, enter `30`
- In **% Overshoot**, enter `5`



Click **OK**.

Instead of specifying time-domain requirements in the Check blocks, you can specify them in the Design Optimization tool without adding blocks. For an example, see "Specify Reference Signal" on page 3-26.

Specify Design Variables

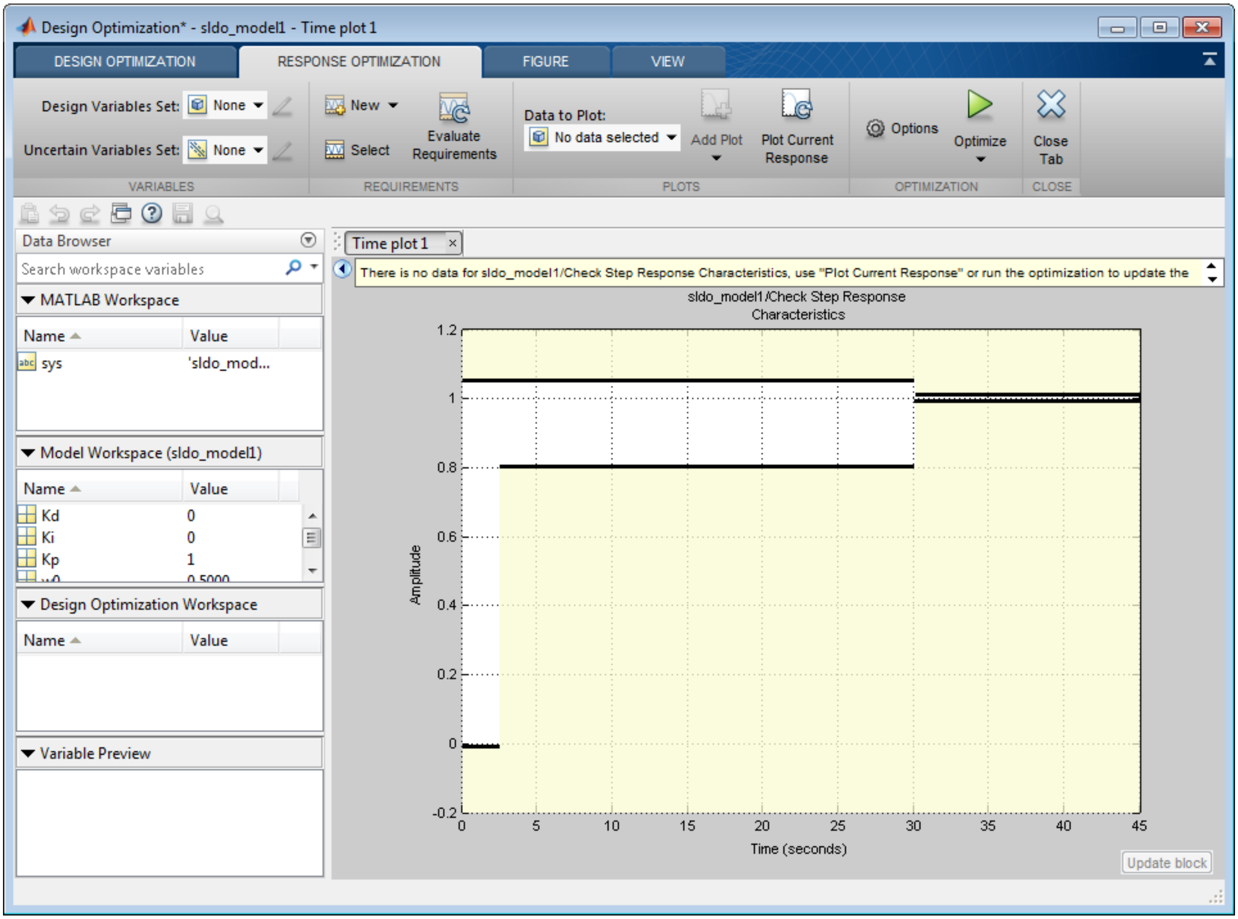
Before you begin this task, specify the design requirements as described in "Specify Step Response Requirements" on page 3-5.

When you optimize the model response, the software modifies the design variable (parameter) values to meet the design requirements.

1 In the Simulink model window, select **Analysis** > **Response Optimization**.

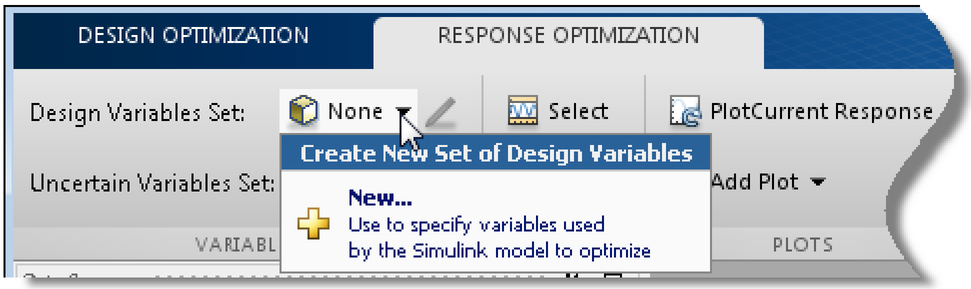
   Alternatively, click **Response Optimization** in the Block Parameters dialog box.

A Design Optimization tool for the model opens.

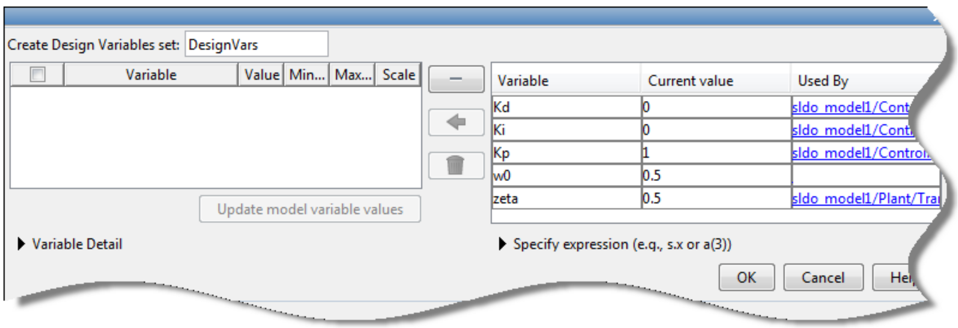


The amplitude versus time plot graphically shows the step response requirements specified in the Check Step Response Characteristics block.

2 Select **New** in the **Design Variables Set** drop-down list.

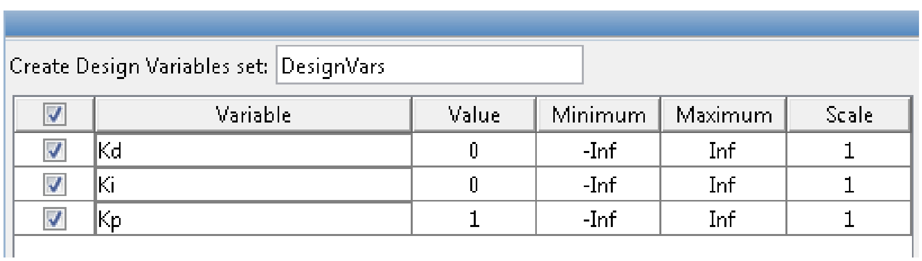


A window opens where you specify design variables.



3 Click Kd, Ki and Kp to select them.

4 Click  to add the selected parameters to a design variables set.



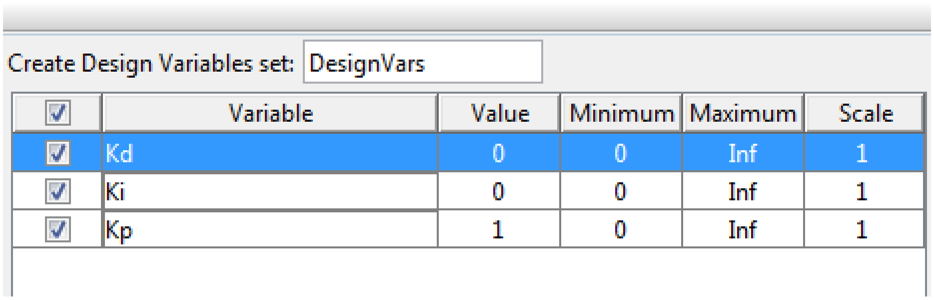
The software displays the following parameter settings:

- **Variable** — Parameter name

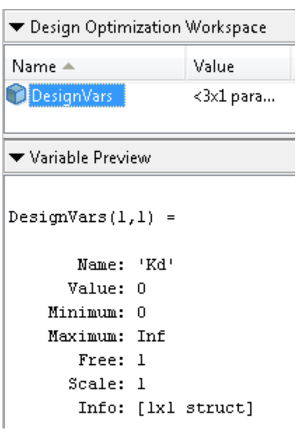
- **Value** — Current parameter value
- **Minimum** and **Maximum** — Parameter bounds
- **Scale** — Scaling factor for the parameter

The check-box indicates that the parameter is included in the design variable set. The default design variable set name is `DesignVars`.

5 To limit the parameters to positive values, enter the minimum value of each parameter as `0` in the corresponding **Minimum** field and press **Enter**.



Click **OK**. A new variable `DesignVars` appears in **Design Optimization Workspace** of the Design Optimization tool. You can click the variable to view its contents in the **Variable Preview** area.

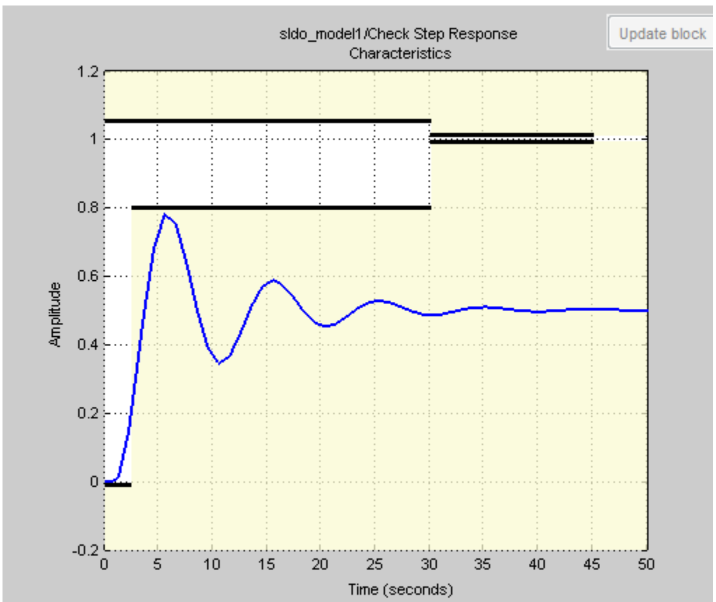


Optimize Model Response

Before you begin this task, you must have already specified the design requirements and design variables as described in “Specify Step Response Requirements” on page 3-5, and “Specify Design Variables” on page 3-8, respectively.

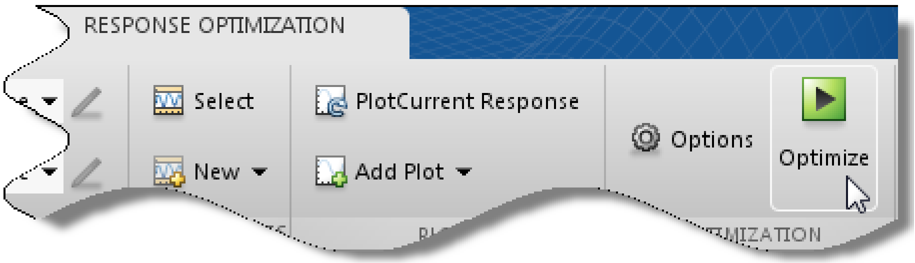
1 (Optional) View the current response of the model. Click 

.

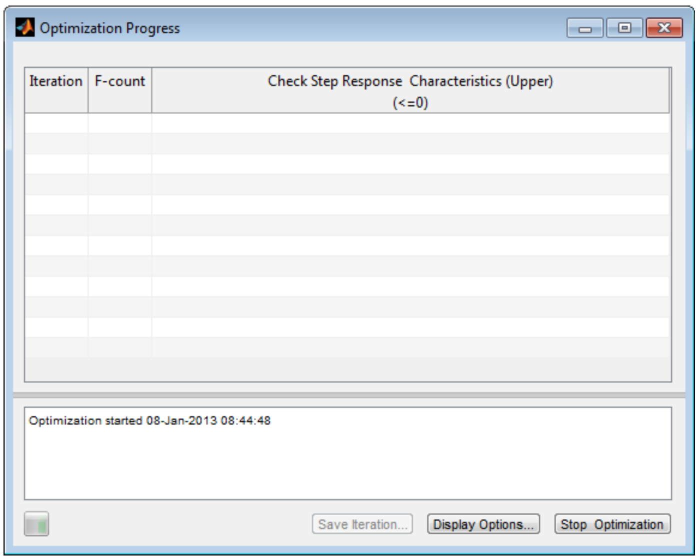


The plot shows that the model output does not meet the specified step response requirements.

2 Click **Optimize**.



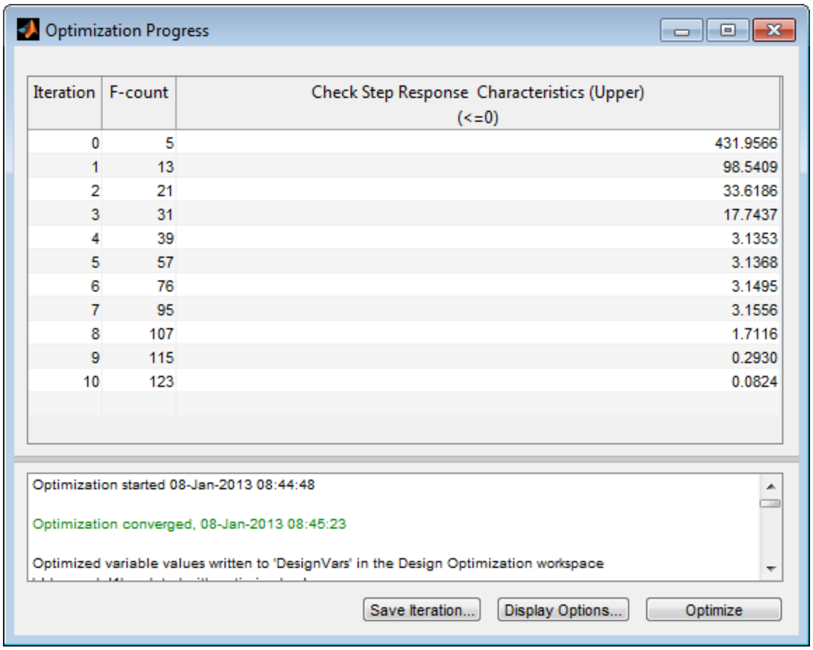
An Optimization Progress window opens.



Tip To view the model response and optimization progress windows simultaneously, tile them using the plot layout area .

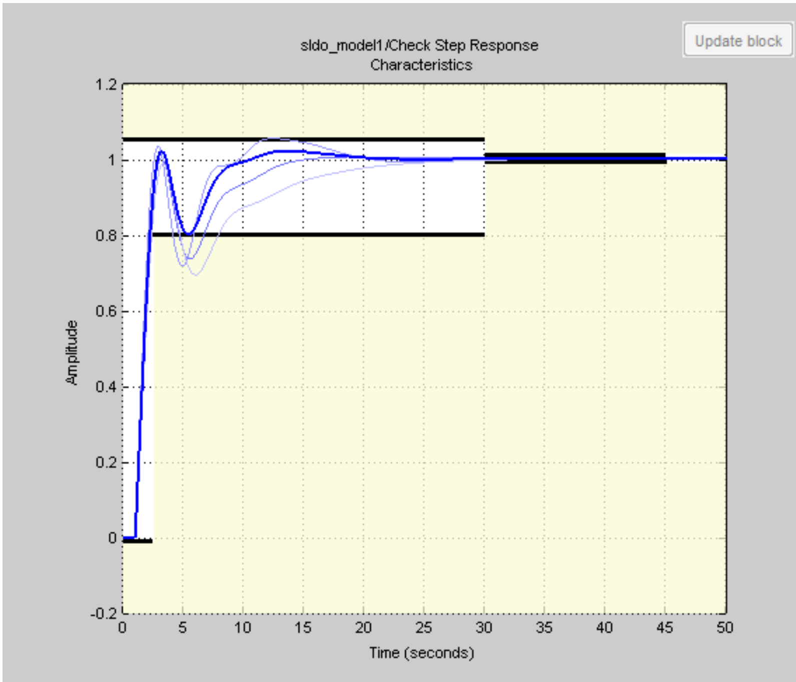
At each optimization iteration, the software simulates the model, and the default optimization solver `Gradient descent` (`fmincon`) modifies the design variables to reduce the distance between the simulated response and the design requirement line segments. For more information, see “Selecting Optimization Methods” and “How the Optimization Algorithm Formulates Minimization Problems”.

After the optimization completes, the optimization progress window resembles the next figure.



The message `Optimization converged` indicates that the optimization solver found a solution that meets the design requirements within the tolerances and parameter bounds. For more information about the outputs displayed in the optimization, see “Iterative Display” in the Optimization Toolbox documentation.

3 Verify that the model output meets the step response requirements.



The plot displays the last five iterations. The final response using the optimized parameter values appears as the thick line.

The optimized response lies in the white region bounded by the design requirement line segments and thus meets the requirements.

4 View the optimized parameter values. Click `DesignVars` in **Design Optimization Workspace** and view the updated values in the **Variable Preview** area.

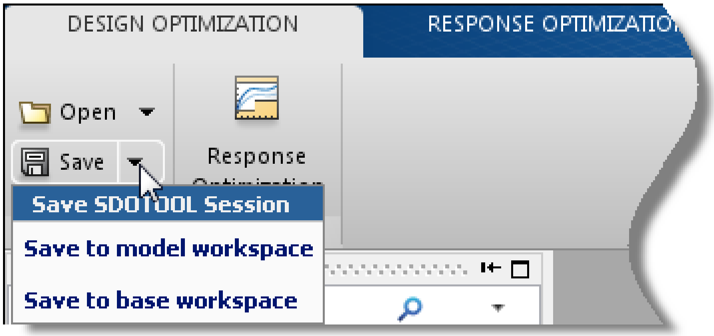
The optimized values of the design variables are automatically updated in the Simulink model.

Save the Session

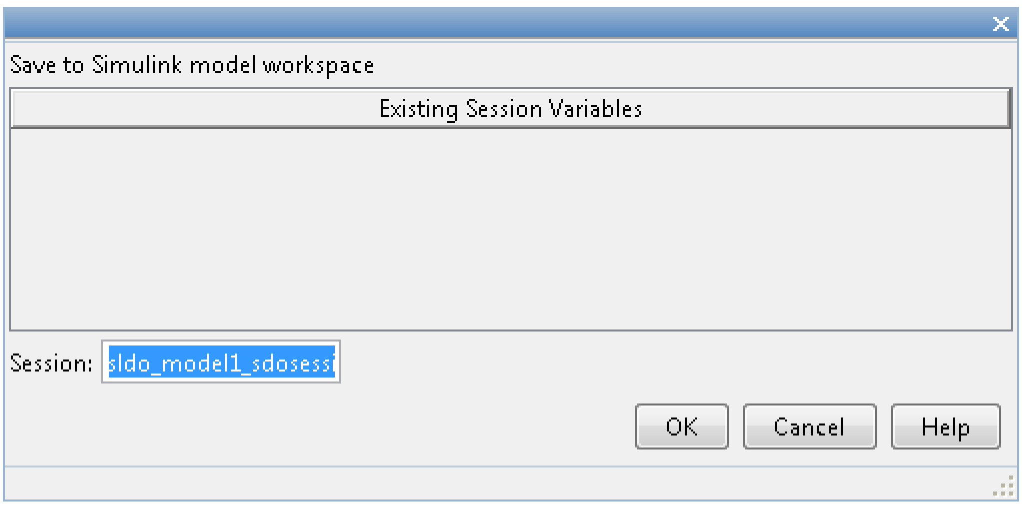
After you optimize the model response to meet design requirements, you can save the Design Optimization tool session which includes the optimized parameter values.

1 Click the **Design Optimization** tab of the Design Optimization tool.

2 In the **Save** drop-down list, select **Save to model workspace**.



A window opens where you specify the session name.



3 Specify a session name in the **Session** field.

Click **OK**.

Tip To open the saved session, click the **Open from model workspace** option in the **Open** drop-down list of the Design Optimization tool for the model.

Related Examples

- “Design Optimization to Meet Step Response Requirements (Code)”
- “Design Optimization to Track Reference Signal (GUI)” on page 3-25

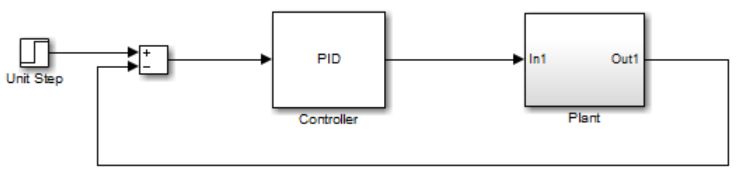
Design Optimization to Meet Step Response Requirements (Code)

In this section...

This example shows how to programmatically optimize controller parameters to meet step response requirements using `sdo.optimize`.

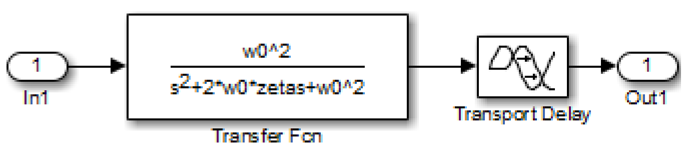
Model Structure

The model `sldo_model1` includes the following blocks:



- **Controller block**, which is a PID controller. This block controls the output of the `Plant` subsystem.
- **Unit Step block** applies a step input and produces the model output that should meet step response requirements.

  You can also use other types of inputs, such as ramp, to optimize the response to meet step response requirements generated by such inputs.
- **Plant subsystem** is a second-order system with delay. It contains "Transfer Function" and "Transport Delay" blocks.



Design Requirements

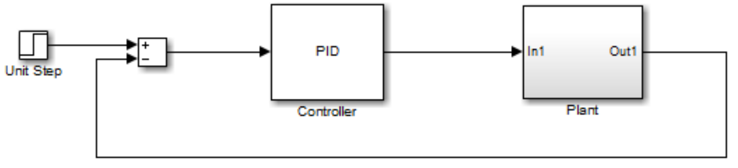
The plant output must meet the following step response requirements:

- Rise time less than 2.5 seconds
- Settling time less than 30 seconds
- Overshoot less than 5%

Specify Step Response Requirements

1 Open the Simulink model.

```
sys = 'sldo_model1';
open_system(sys);
```



2 Log the model output signal.

Design requirements require logged model signals. During optimization, the model is simulated using the current value of the model parameters and the logged signal is used to evaluate the design requirements.

```
PlantOutput = Simulink.SimulationData.SignalLoggingInfo;
PlantOutput.BlockPath               = [sys '/Plant'];
PlantOutput.OutputPortIndex         = 1;
PlantOutput.LoggingInfo.NameMode    = 1;
PlantOutput.LoggingInfo.LoggingName = 'PlantOutput';
```

3 Store the logging information.

```
simulator = sdo.SimulationTest(sys);
simulator.LoggingInfo.Signals = PlantOutput;
```

`simulator` is a `sdo.SimulationTest` object that you also use later to simulate the model.

4 Specify step response requirements.

```
StepResp = sdo.requirements.StepResponseEnvelope;
StepResp.RiseTime = 2.5;
StepResp.SettlingTime = 30;
StepResp.PercentOvershoot = 5;
```

`StepResp` is a `sdo.requirements.StepResponseEnvelope` object.

Specify Design Variables

When you optimize the model response, the software modifies parameter (design variable) values to meet the design requirements.

1 Select model parameters to optimize.

```
p = sdo.getParameterFromModel(sys,{'Kp','Ki','Kd'});
```

`p` is an array of 3 `param.Continuous` objects.

2 To limit the parameters to positive values, set the minimum value of each parameter to `0`.

```
p(1).Minimum = 0;
p(2).Minimum = 0;
p(3).Minimum = 0;
```

Optimize Model Response

1 Create a design function.

```
evalDesign = @(p) sldo_model1_design(p,simulator,StepResp);
```

`evalDesign` is an anonymous function that calls the cost function `sldo_model1_design`. The cost function simulates the model and evaluates the design requirements.

Tip Type `edit sldo_model1_design` to view this function.

2 (Optional) Evaluate the current response.

a Compute the model response using the current values of the design variables.

```
initDesign = evalDesign(p);
```

During simulation, the `Step Response` block throws assertion warnings at the MATLAB prompt which indicate that the requirements specified in the block are not satisfied.

b Examine the nonlinear inequality constraints.

```
initDesign.Cleq
```

```
ans =

   -0.2571
   -0.4968
   -0.5029
   -1.0000
    0.7060
    0.5092
    0.4964
  201.4682
```

Some `Cleq` values are positive, beyond the specified tolerance, indicating that the response using the current parameter values violates the design requirements.

3 Specify optimization options.

```
opt = sdo.OptimizeOptions;
opt.MethodOptions.Algorithm = 'active-set';
```

The software configures `opt` to use the default optimization method, `fmincon`. However, because the optimization problem specifies only constraints, you specify the Active Set algorithm for `fmincon`. The Active Set algorithm is good for solving feasibility (constraints only) optimization problems.

4 Optimize the response.

```
[pOpt,optInfo] = sdo.optimize(evalDesign,p,opt);
```

At each optimization iteration, the software simulates the model, and the default optimization solver `fmincon` modifies the design variables to meet the design requirements. For more information, see "Selecting Optimization Methods" and "How the Optimization Algorithm Formulates Minimization Problems".

After the optimization completes, the command window displays the following results:

```
                                max       Step-size    First-order
 Iter F-count        f(x)   constraint                 optimality
    0      5            0        201.5
    1     12            0        167.5        0.687            0
    2     19            0       0.8462         1.22         5.86
    3     29            0       0.8169        0.133            0
    4     37            0       0.7819        0.115            0
    5     44            0        3.121        0.265         3.29
    6     52            0        3.328         2.92            0
    7     59            0       0.1413         1.26         2.81
    8     66            0      0.02418        0.443       0.0029
    9     73            0    8.125e-05       0.0563     1.03e-05
Local minimum found that satisfies the constraints.

Optimization completed because the objective function is non-decreasing in
feasible directions, to within the selected value of the function tolerance,
and constraints are satisfied to within the selected value of the constraint tolerance.
```

The message `Local minimum found that satisfies the constraints` indicates that the optimization solver found a solution that meets the design requirements within specified tolerances. For more information about the outputs displayed during the optimization, see "Iterative Display" in the Optimization Toolbox documentation.

5 Examine the optimization termination information, contained in the `optInfo` output argument. This information helps you verify that the response meets the step response requirements.

For example, check the following fields:

- `Cleq`, which shows the optimized nonlinear inequality constraints.

  ```
  optInfo.Cleq
  ```

  ```
  ans =

     0.0001
    -0.0099
    -0.0099
    -1.0000
     0.0004
    -0.0100
    -0.0101
    -0.1997
  ```

  All values satisfy `Cleq` ≤ `0`, within the optimization tolerances, which indicates that the step response requirements are satisfied.

- `exitflag`, which identifies why the optimization terminated.

The value is 1 which indicates that the solver found a solution which was less than the specified tolerances on the function value and constraint violations.

6 View the optimized parameter values.

```
pOpt
```

```
pOpt(1,1) =

       Name: 'Kp'
      Value: 1.2364
    Minimum: 0
    Maximum: Inf
       Free: 1
      Scale: 1
       Info: [1x1 struct]


pOpt(2,1) =

       Name: 'Ki'
      Value: 0.3901
    Minimum: 0
    Maximum: Inf
       Free: 1
      Scale: 1
       Info: [1x1 struct]


pOpt(3,1) =

       Name: 'Kd'
      Value: 2.5937
    Minimum: 0
    Maximum: Inf
       Free: 1
      Scale: 1
       Info: [1x1 struct]
```

7 Simulate the model with the optimized values.

a Update optimized parameter values in the model.

```
sdo.setValueInModel(sys,pOpt);
```

b Simulate the model.

```
sim(sys);
```

See Also

`sdo.requirements.StepResponseEnvelope` | `param.Continuous` | `sdo.getParameterFromModel` | `sdo.optimize` | `sdo.OptimizeOptions` | `sdo.SimulationTest` | `Simulink.SimulationData.SignalLoggingInfo`

Related Examples

- "Design Optimization to Meet Step Response Requirements (GUI)" on page 3-4
- "Design Optimization to Track Reference Signal (GUI)" on page 3-25

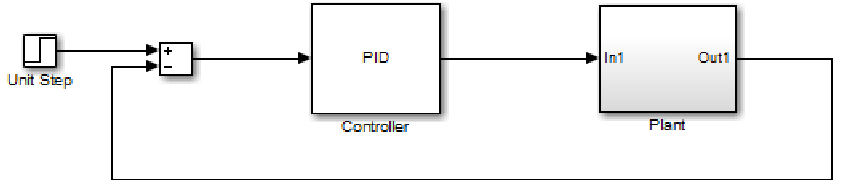
Design Optimization to Track Reference Signal (GUI)

In this section...

This example shows how to optimize controller parameters to track a reference signal using the Design Optimization tool. You specify the reference signal without adding any Check blocks to the model.

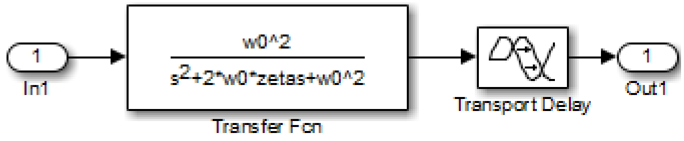
Model Structure

The model `sldo_model1` includes the following blocks:



- **Controller block**, which is a PID controller. This block controls the output of the `Plant` subsystem.
- **Unit Step block** applies a step input and produces the model output that should meet step response requirements.

  You can also use other types of inputs, such as ramp, to optimize the response to meet step response requirements generated by such inputs.
- **Plant subsystem** is a second-order system with delay. It contains "Transfer Function" and "Transport Delay" blocks.



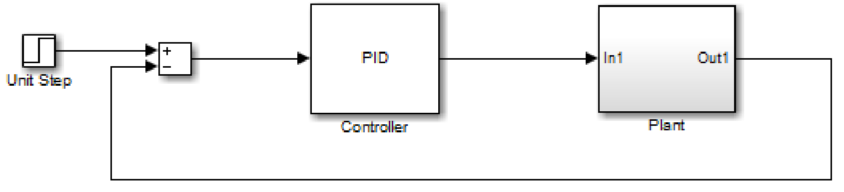
Design Requirements

The model output must track a reference signal $y = 1 - \exp(-0.1 \times t)$, where t is time.

Specify Reference Signal

1 Open Simulink model.

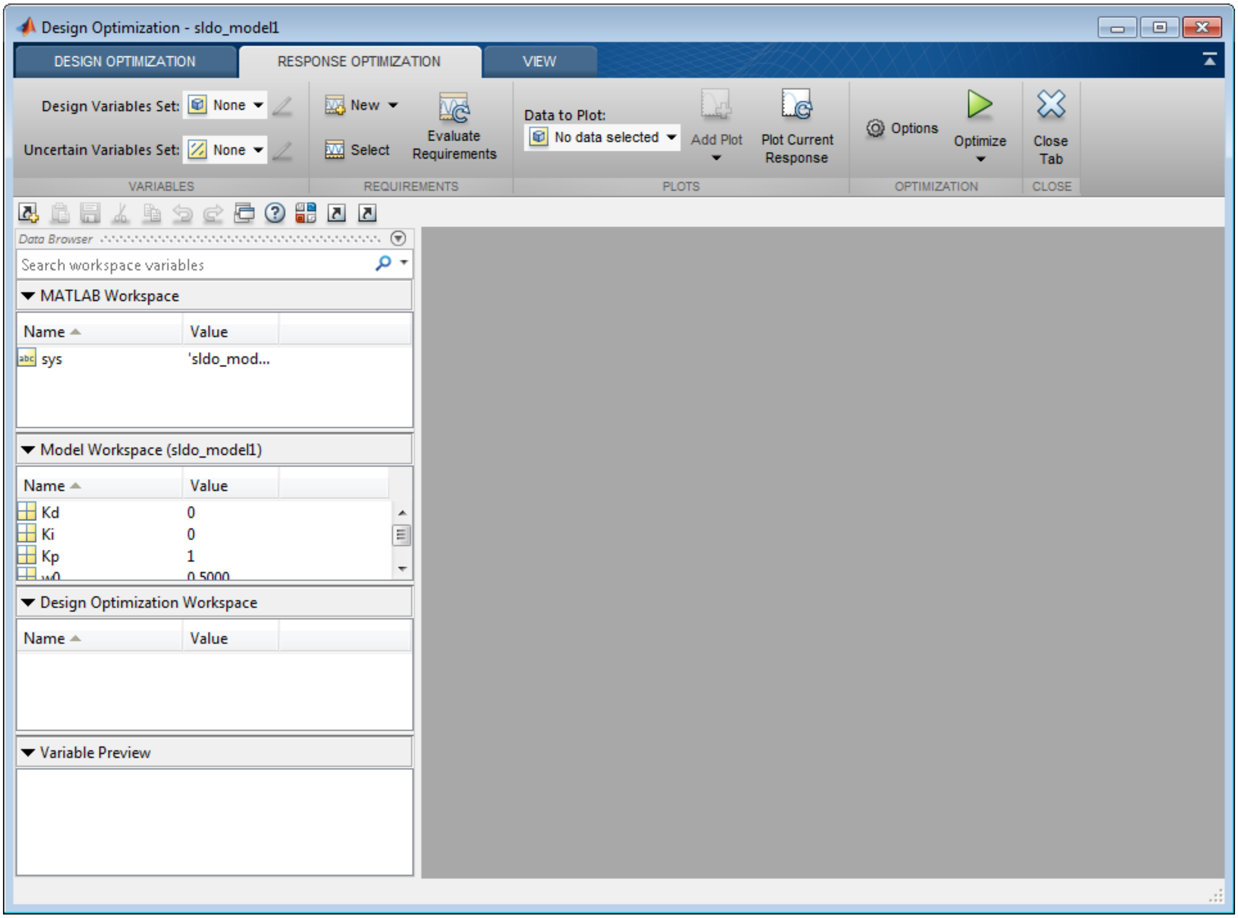
```
sys = 'sldo_model1';
open_system(sys);
```



To learn more about the model, see "Model Structure" on page 3-25.

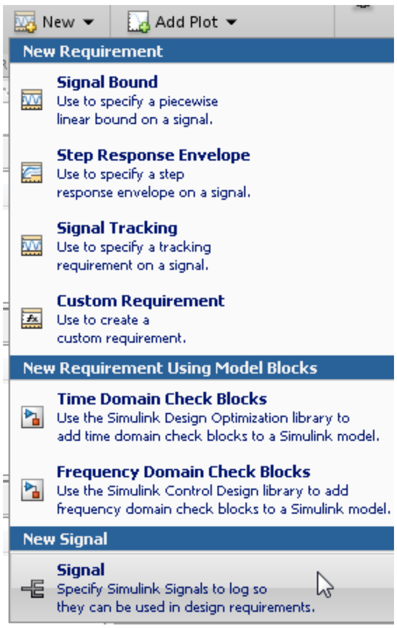
2 In the Simulink model window, select **Analysis** > **Response Optimization**.

A Design Optimization tool for the model opens.

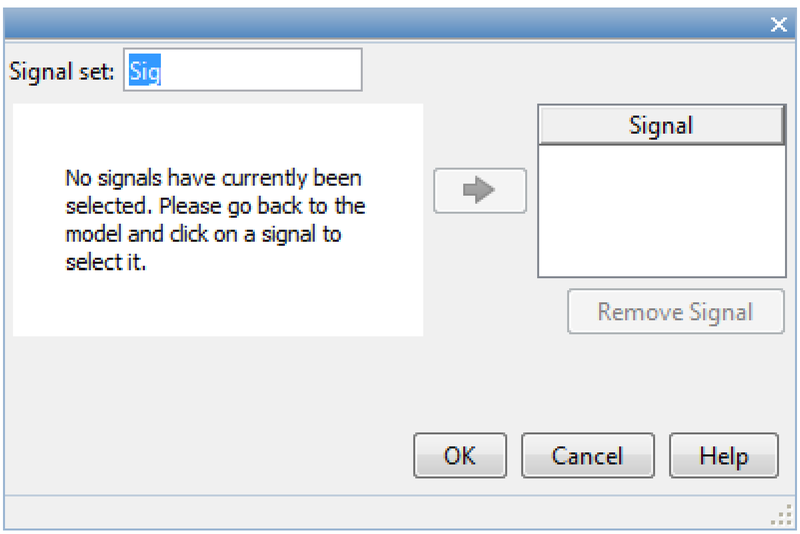


3 Select the model signal which must track the reference signal.

a In the **New** drop-down list, select **Signal**.

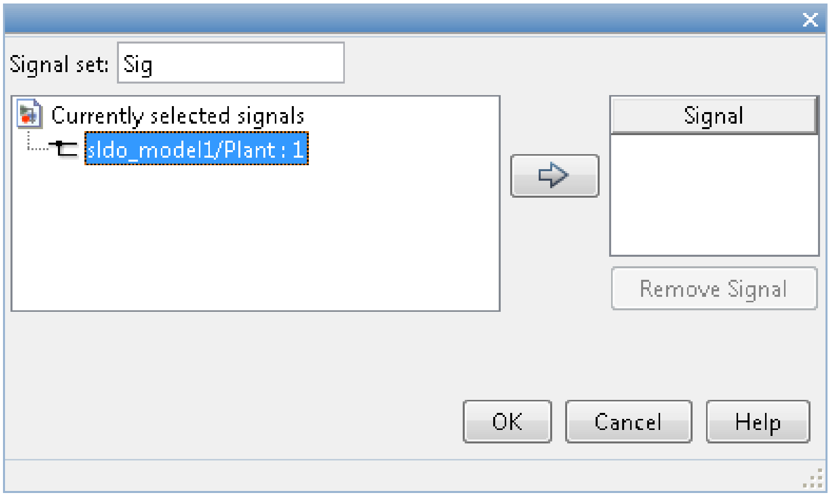


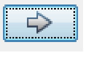
A window opens where you select a model signal.

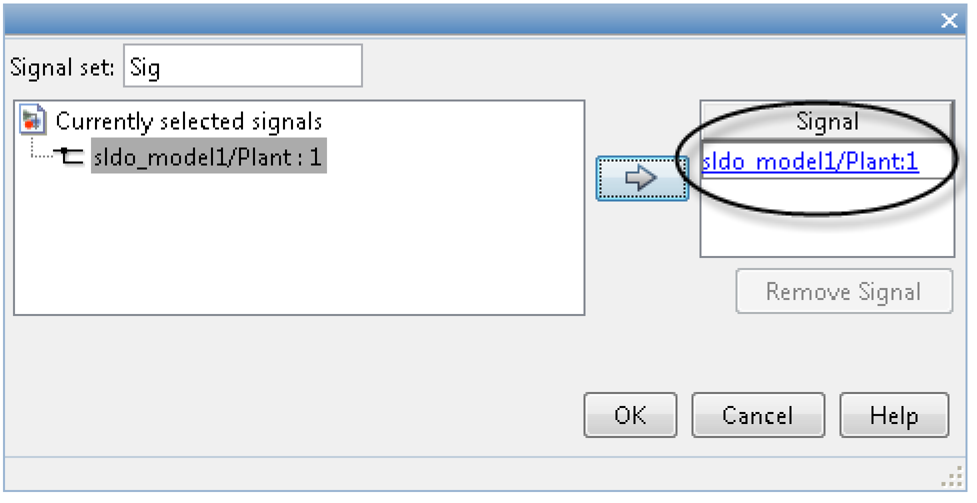


b In the Simulink model window, click the output of the `Plant` block.

The window updates to display the selected signal.



c Select the signal and click  to add it to the signal set.

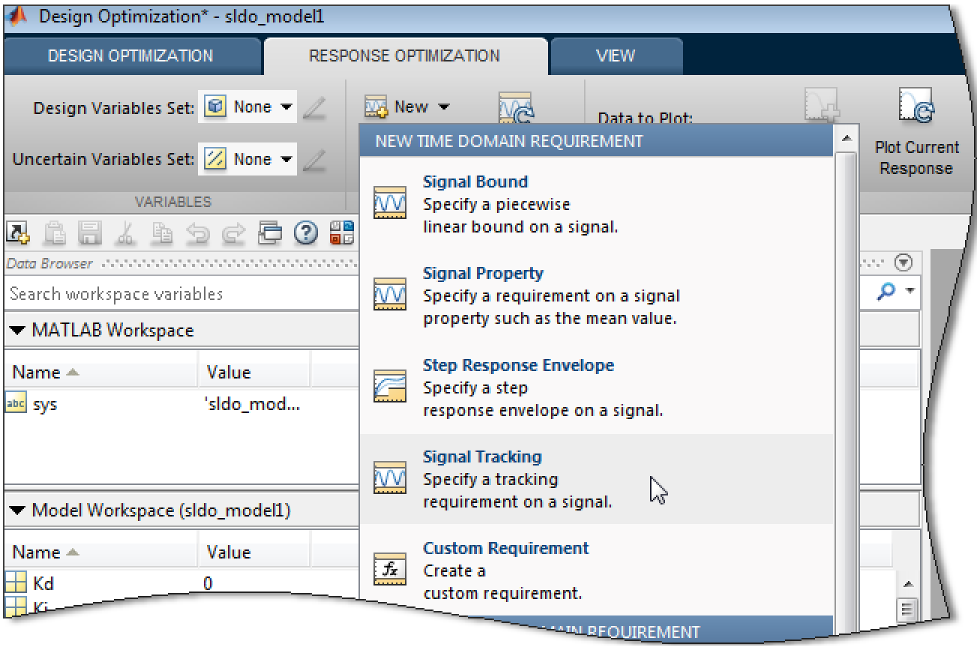


d In **Signal set**, enter `PlantOutput` as the selected signal name.

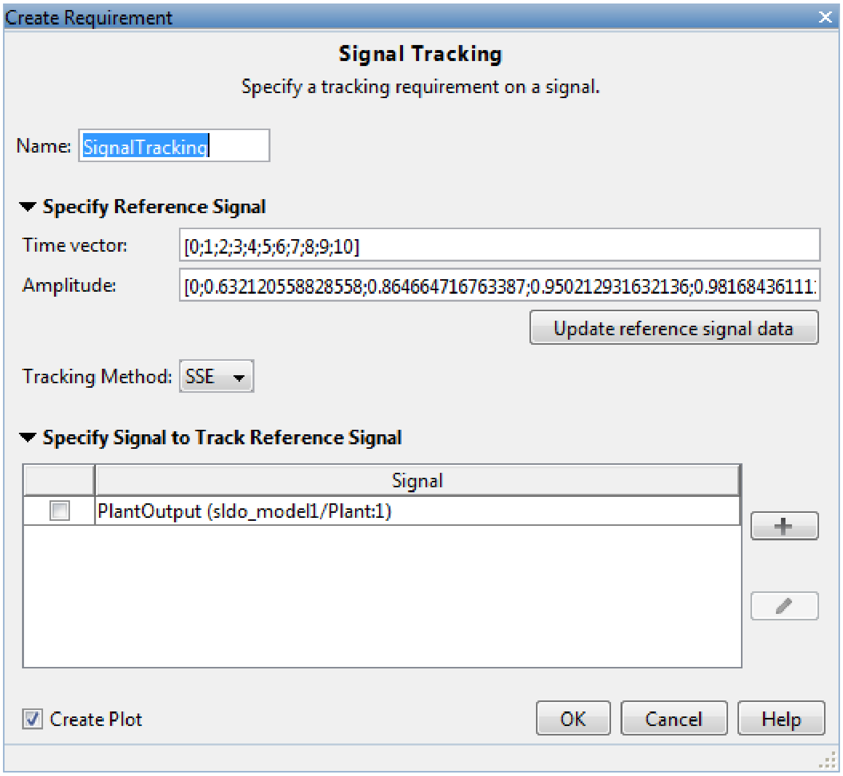
Click **OK**. A new variable `PlantOutput` appears in the **Design Optimization Workspace** of the Design Optimization tool.

4 Specify the reference signal that the model output must track.

a In the **New** drop-down list, select **Signal Tracking**.



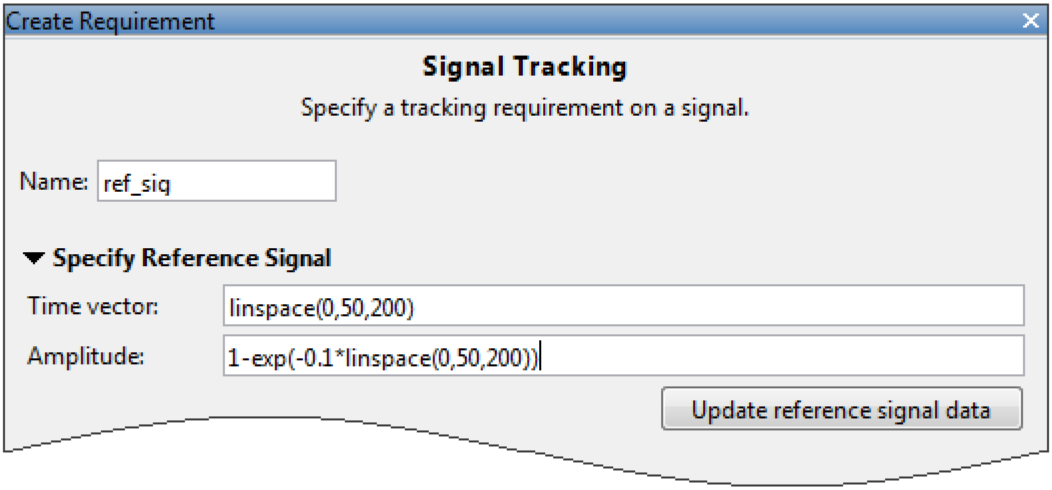
A window opens where you specify the reference signal.



b In the **Name** edit box, enter `ref_sig`.

c In the **Time vector** edit box, enter `linspace(0,50,200)`

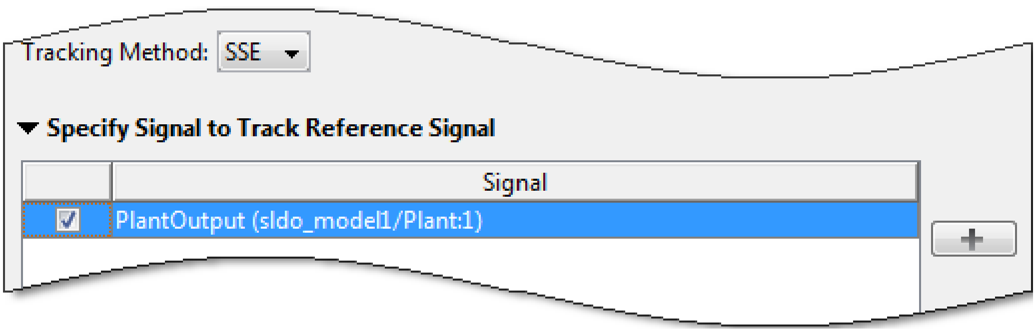
d In the **Amplitude** edit box, enter `1-exp(-0.1*linspace(0,50,200))`.



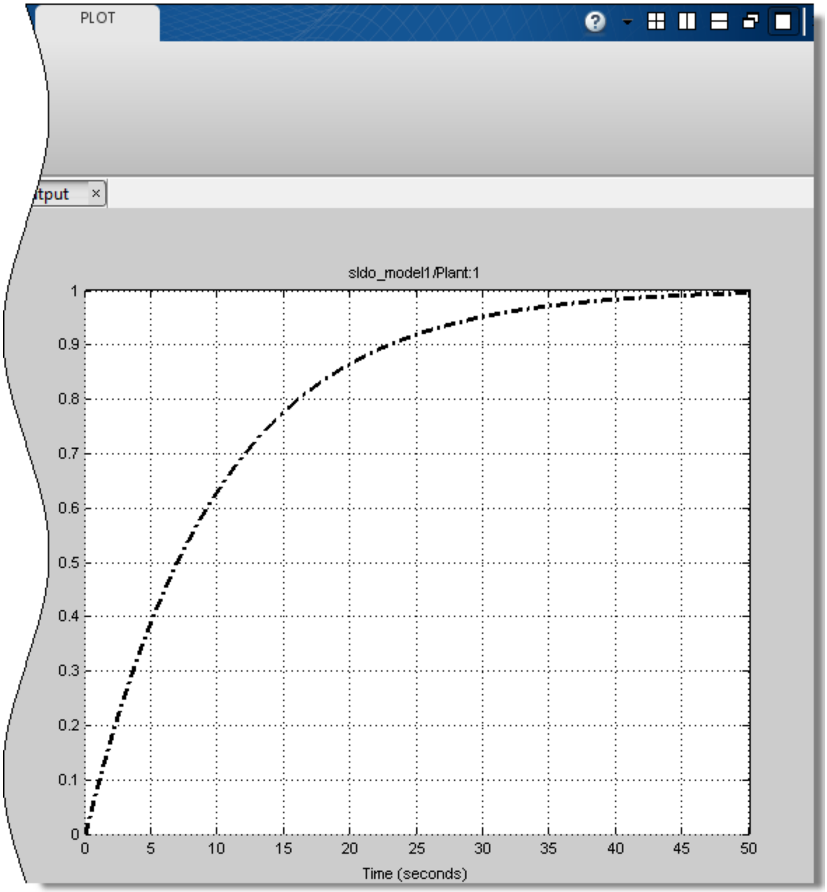
The **Tracking Method** is `SSE`, which means that at each optimization iteration, the solver attempts to reduce the sum of squared errors between the simulated output and reference signal.

e Click **Update reference signal data**.

f Select the check-box corresponding to the signal you selected in the previous step in the **Specify Signal to Track Reference Signal** area.



Click **OK**. A new variable `ref_sig` appears in the **Design Optimization Workspace** and the Design Optimization window updates to plot the reference signal.



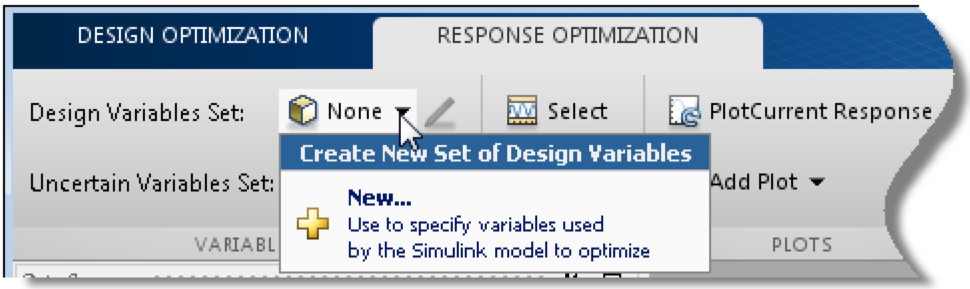
Specify Design Variables

Before you begin this task, specify the reference signal to track as described in "Specify Reference Signal" on page 3-26.

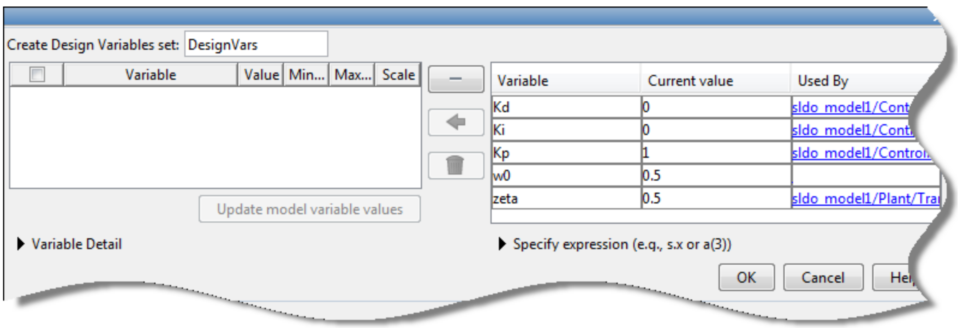
When you optimize the model response, the software modifies the design variable (model parameter) values to meet the design requirements.

In the **Response Optimization** tab:

1 Select **New** in the **Design Variables Set** drop-down list.

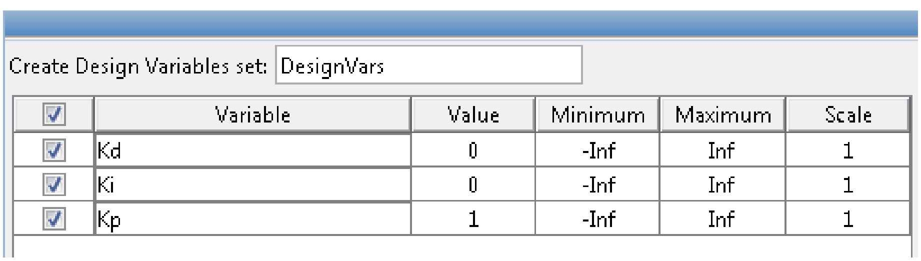


A window opens where you specify design variables.



2 Click Kd, Ki and Kp to select them.

3 Click  to add the selected parameters to a design variables set.



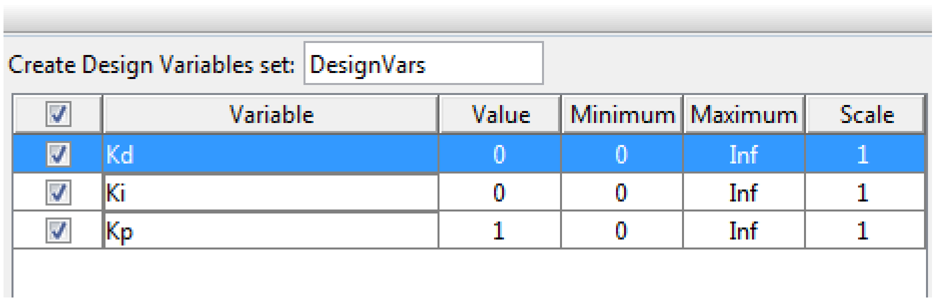
The software displays the following parameter settings:

- **Variable** — Parameter name

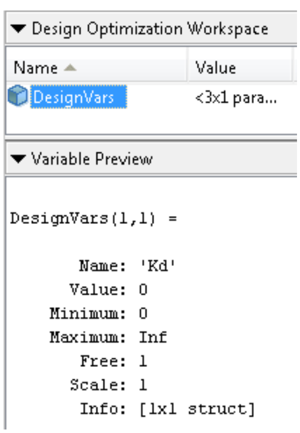
- **Value** — Current parameter value
- **Minimum** and **Maximum** — Parameter bounds
- **Scale** — Scaling factor for the parameter

The check-box indicates that the parameter is included in the design variable set. The default design variable set name is `DesignVars`.

4 To limit the parameters to positive values, enter the minimum value of each parameter as `0` in the corresponding **Minimum** field and press **Enter**.



Click **OK**. A new variable `DesignVars` appears in **Design Optimization Workspace** of the Design Optimization tool. You can click the variable to view its contents in the **Variable Preview** area.

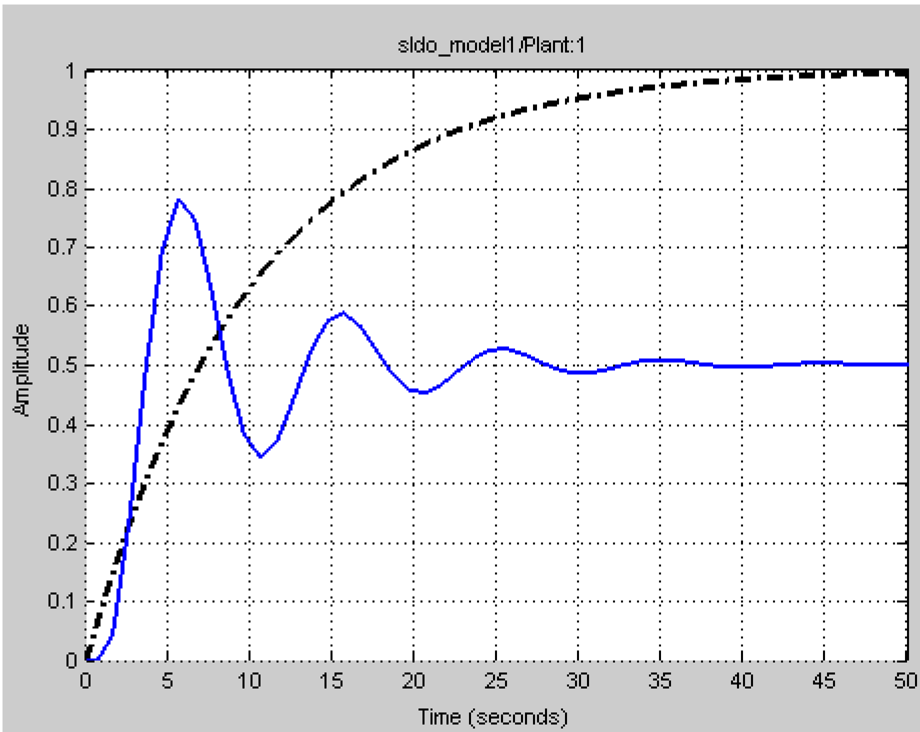


Optimize Model Response

Before you begin this task, you must have specified the reference signal to track and design variables, as described in “Specify Reference Signal” on page 3-26 and “Specify Design Variables” on page 3-33, respectively.

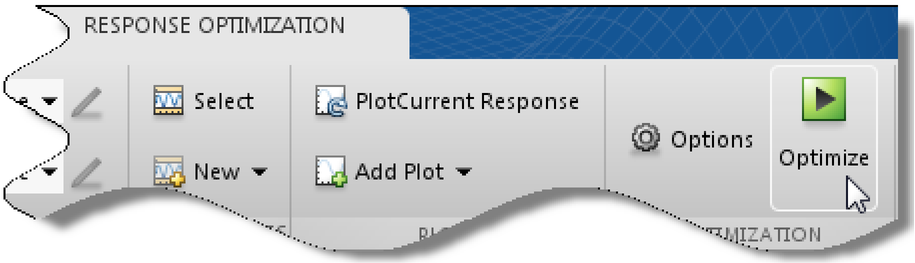
1 (Optional) View the current model response. Click 

.



The plot shows that the response does not track the reference signal.

2 Click **Optimize**.

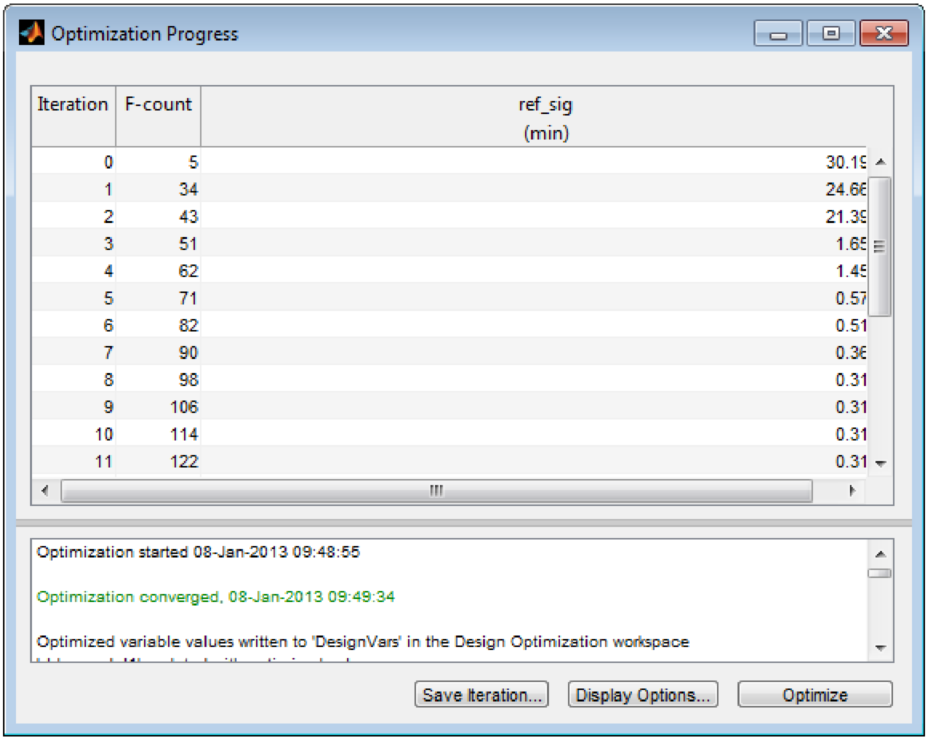


An optimization progress window opens.

Tip To view the model response and optimization progress windows simultaneously, tile them using the plot layout area .

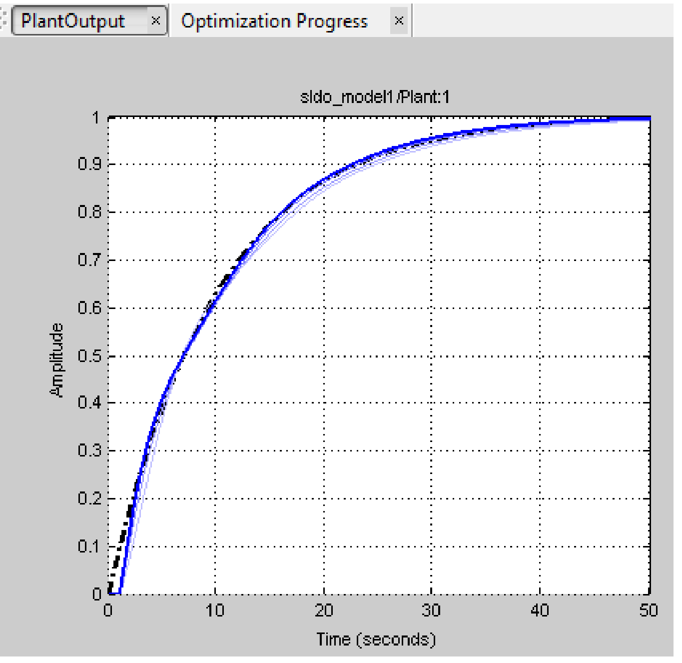
At each iteration, the optimization solver `Gradient descent` (`fmincon`) modifies the controller parameters to minimize the error between the simulated response and the reference signal. To learn more, see "How the Optimization Algorithm Formulates Minimization Problems".

After the optimization completes, the optimization progress window resembles the following figure.



The message `Optimization converged` indicates that the optimization method found a solution that tracks the reference signal within the tolerances and parameter bounds. For more information about the outputs displayed in the optimization progress window, see "Iterative Display" in the Optimization Toolbox documentation.

3 Verify that the response tracks the reference signal.



The optimized response closely tracks the reference signal.

4 View the optimized parameter values. Click `DesignVars` in **Design Optimization Workspace** and view the updated values in the **Variable Preview** area.

The optimized values of the design variables are automatically updated in the Simulink model.

Related Examples

- “Design Optimization to Meet Step Response Requirements (GUI)” on page 3-4
- “Design Optimization to Meet Step Response Requirements (Code)” on page 3-18

Design Optimization Using Frequency-Domain Check Blocks (GUI)

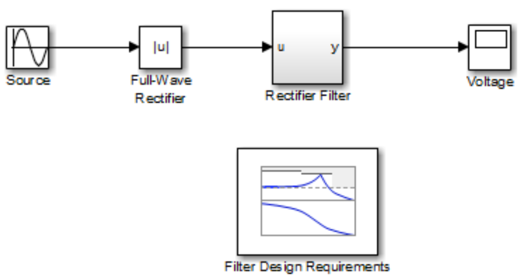
In this section...

This example shows how to optimize a design to meet frequency-domain requirements using the Design Optimization tool. Simulink Control Design software must be installed to optimize a design to meet frequency-domain design requirements.

In this example, you specify the design requirements in a Check Bode Characteristics block and optimize a rectifier filter parameters to meet gain and bandwidth requirements, and minimize a custom objective.

Model Structure

The model `sdorectifier` includes the following blocks:



- **Full-Wave Rectifier block** — An Abs block
- **Rectifier Filter subsystem** — RLC filter implemented using integrator and gain blocks
- **Filter Design Requirements block** — Check Bode Characteristics block that specifies the gain and bandwidth design requirements

Design Requirements

The design optimization problem is multi-objective. The design must:

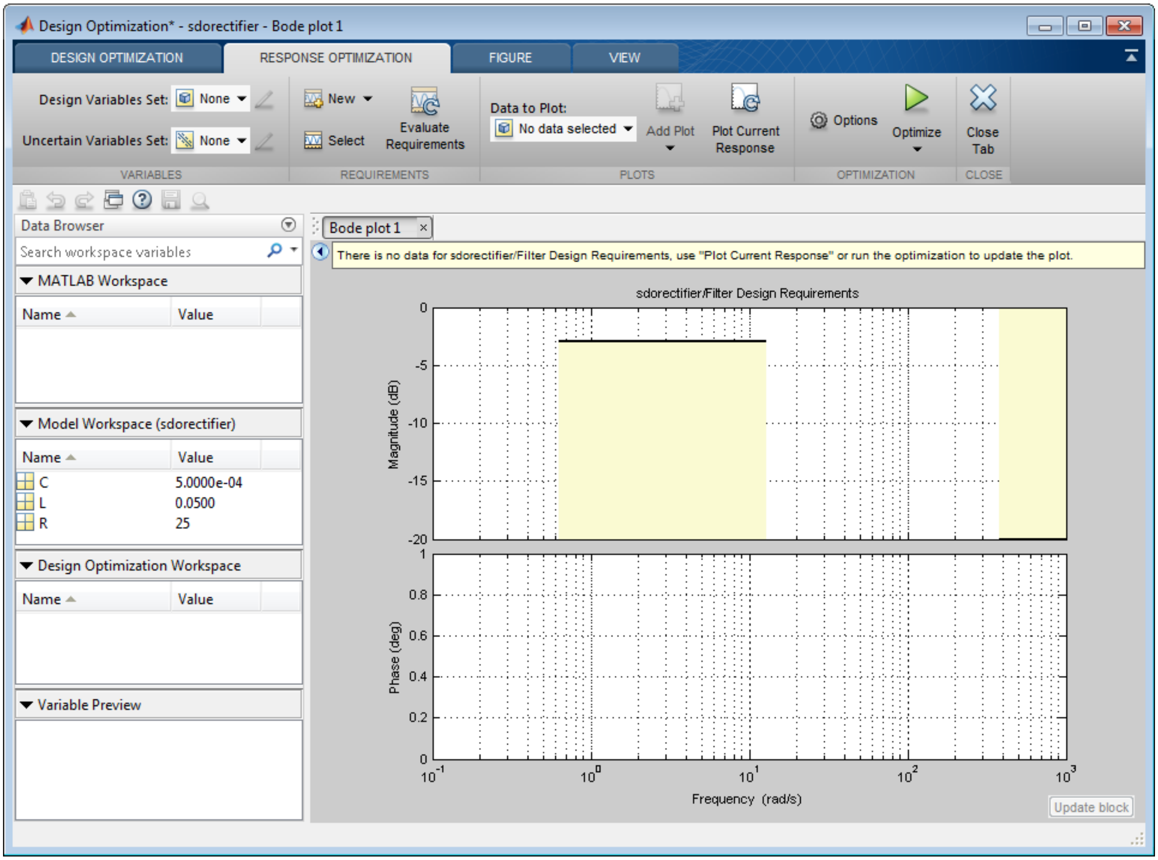
- Have a –3db bandwidth of at least 2Hz
- Limit the gain across the frequency range 2Hz—60Hz to at most 0db
- Limit the gain above 60Hz to at most –20db
- Maximize the filter resistance R
- Minimize the filter inductance L

The requirements ensure that the rectifier filter combination has minimal high frequency content, responds quickly to voltage changes, and limits filter currents.

Specify Design Requirements

1. Open the Design Optimization tool for the model.

   ```
   sdotool('sdorectifier')
   ```



The plot shows the gain and bandwidth requirements specified in the `Filter Design Requirements` block. See their values in the **Bounds** tab of the Block Parameters dialog box.

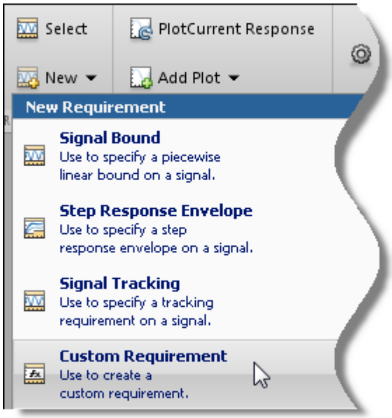
2 Specify a custom objective to minimize the filter inductance and maximize the resistance.

The custom objective is specified in the `sdorectifier_cost` function. The function accepts the design variables *R* and *L*, and returns the objective to be minimized.

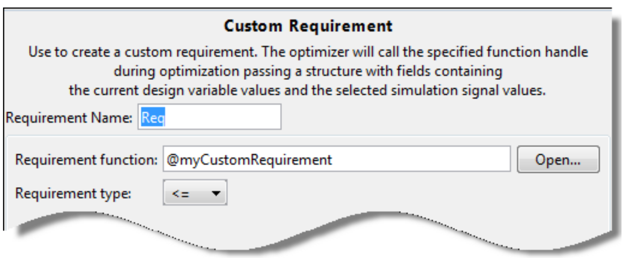
Tip Type `edit sdorectifier_cost` to view this function.

---

a Select **Custom Requirement** in the **New** drop-down list.

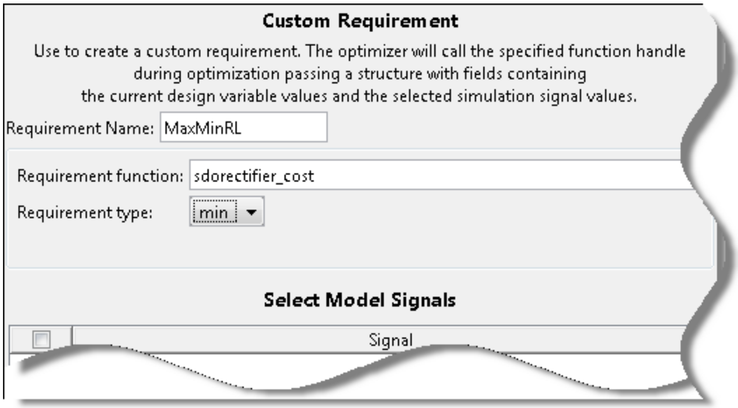


A window opens where you specify the custom requirement.



b Specify the following:

- Enter `MaxMinRL` in **Requirement Name**
- Enter `@sdorectifier_cost` in **Requirement function**. The optimization solver calls the specified function handle.
- Select `min` in **Requirement type**



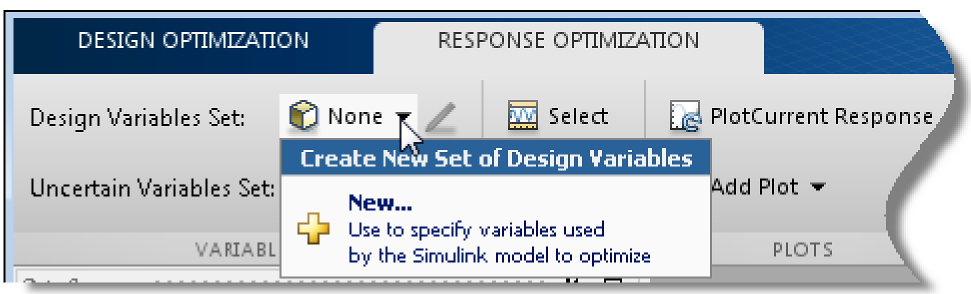
Click **OK**. A new variable `MaxMinRL` appears in **Design Optimization Workspace** of the Design Optimization tool.

Specify Design Variables

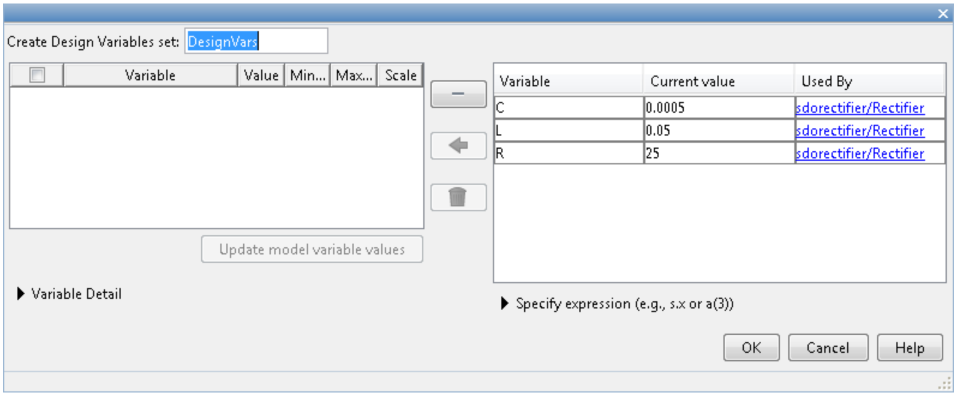
Before you begin this task, specify the design requirements as described in "Specify Design Requirements" on page 3-41.

When you optimize the model response, the software modifies the design variable (parameter) values to meet the design requirements.

1 Select **New** in the **Design Variables Set** drop-down list.



A window opens where you specify design variables.

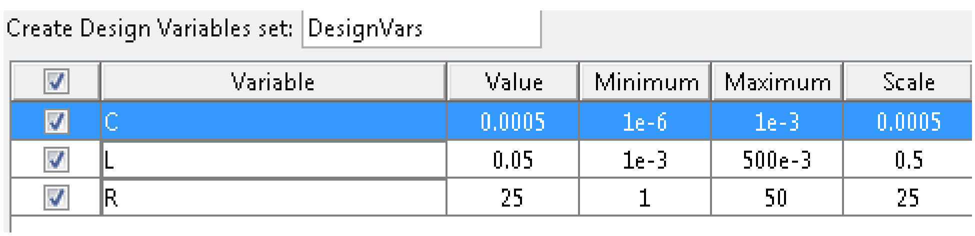


2 Click R, L and C to select them.

3 Click  to add the selected parameters to a design variables set.

4 Specify the value range for each design variable:

- R in the range 1–50 ohms
- L in the range 1–500mH
- C in the range 1µF–1mF
- Because the variables are different orders of magnitude, specify scaling factors in the corresponding **Scale** column of the variables:
  - R by 25
  - L by 0.05
  - C by 0.0005



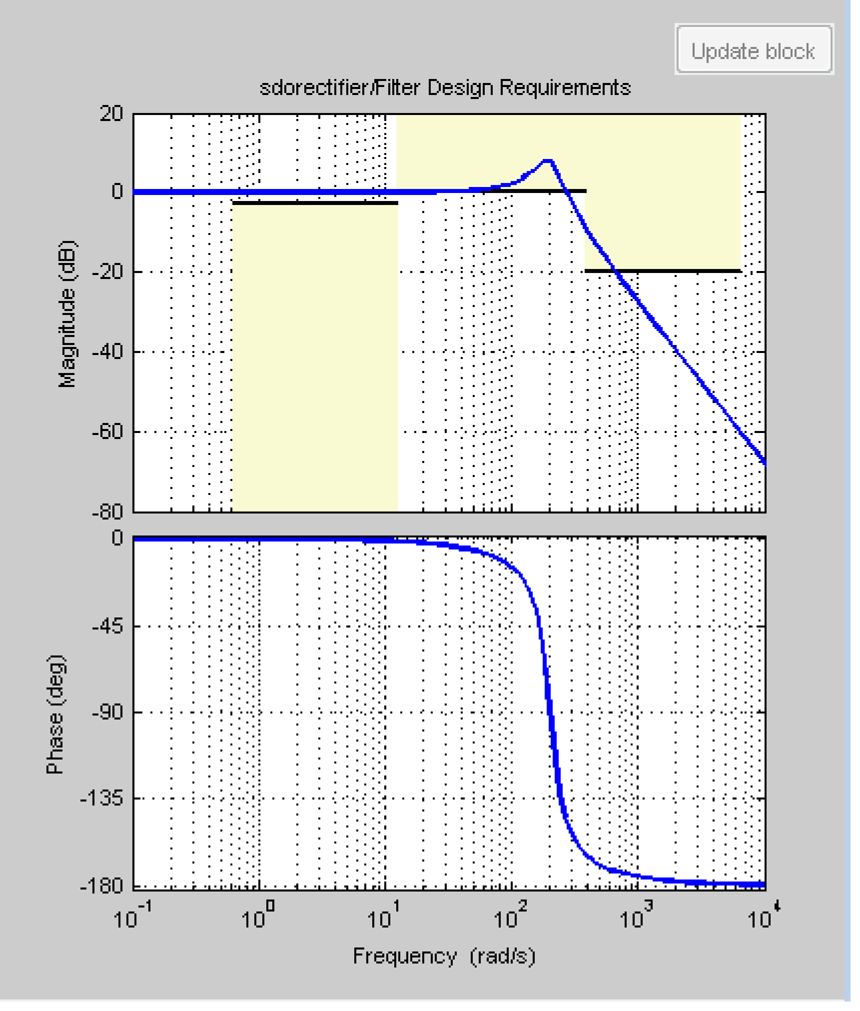
Click **OK**. A new variable `DesignVars` appears in **Design Optimization Workspace** of the Design Optimization tool.

Optimize Design

Before you begin this task, you must have already specified the design requirements and design variables, as described in “Specify Design Requirements” on page 3-41 and “Specify Design Variables” on page 3-44, respectively.

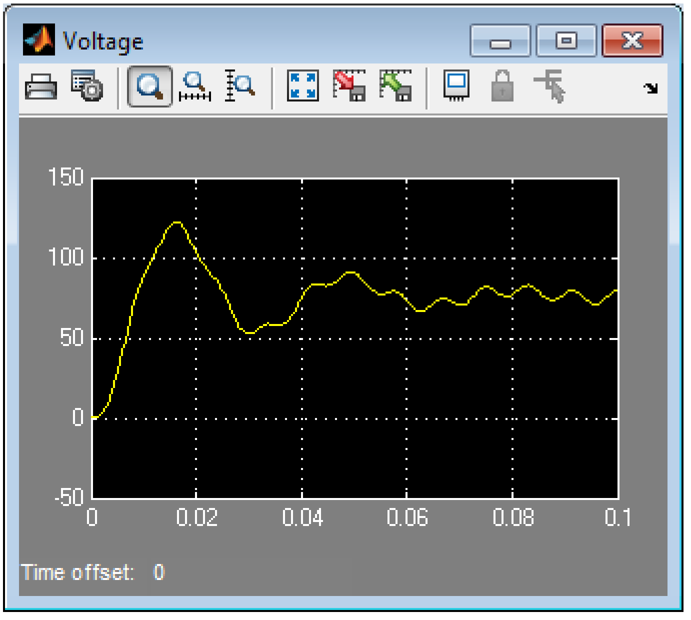
1. (Optional) View the current response of the model. Click 

.

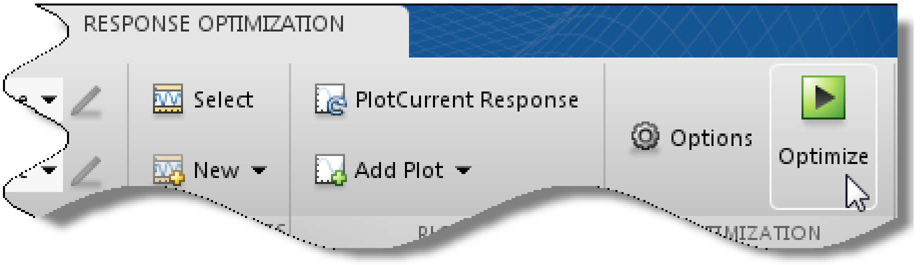


The plot shows that the model does not meet the specified gain and bandwidth requirements.

You also see that the filter voltage signal overshoots its steady-state value and contains significant harmonic content in the `Voltage` scope window.

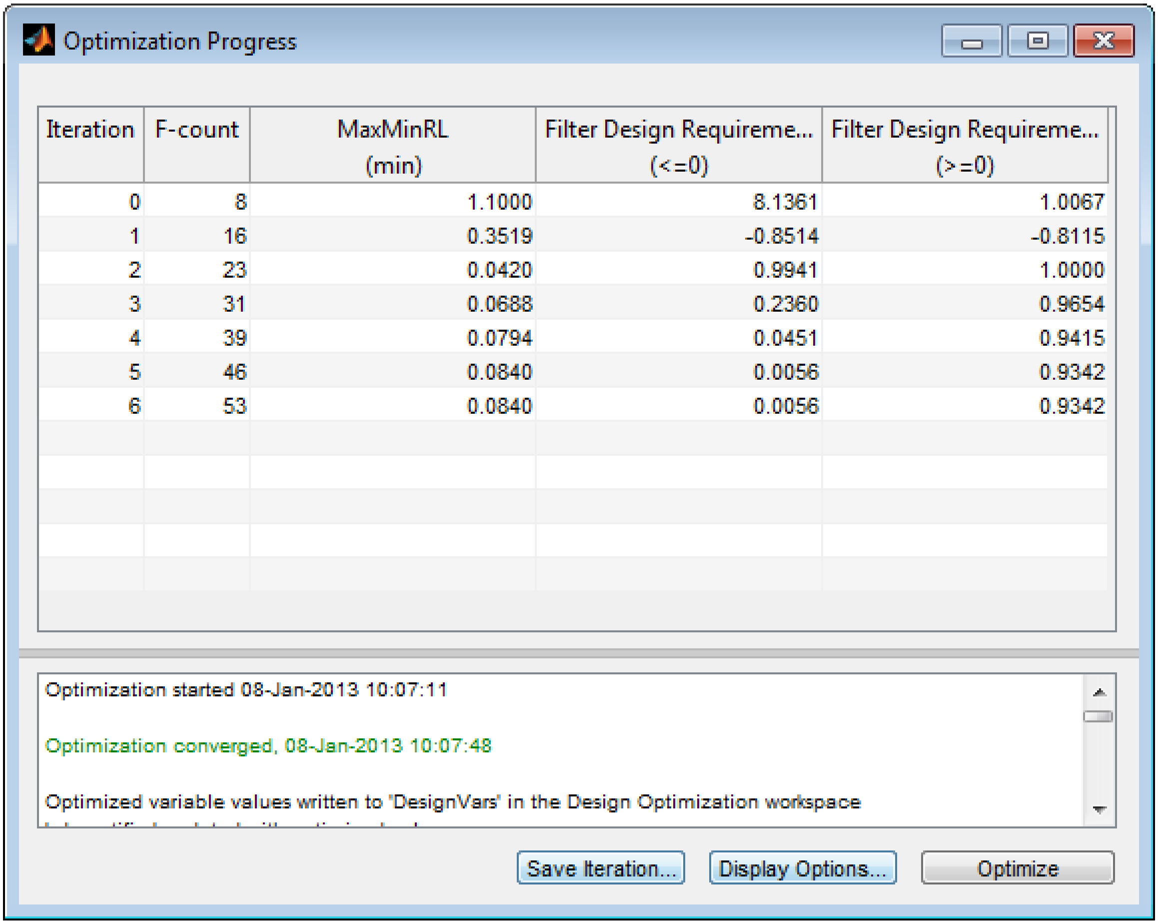


2 Click **Optimize**.



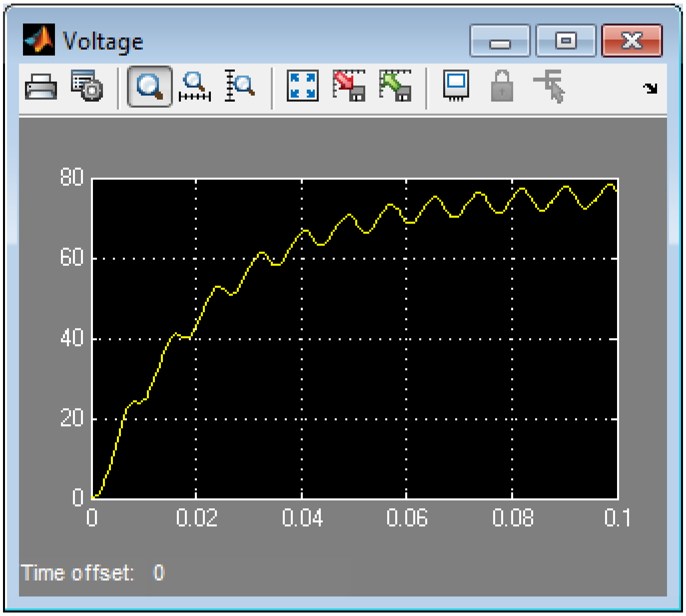
An optimization progress window opens.

After the optimization completes, the optimization progress window resembles the next figure. After a few iterations, the optimization converges to satisfy the filter bandwidth requirements.

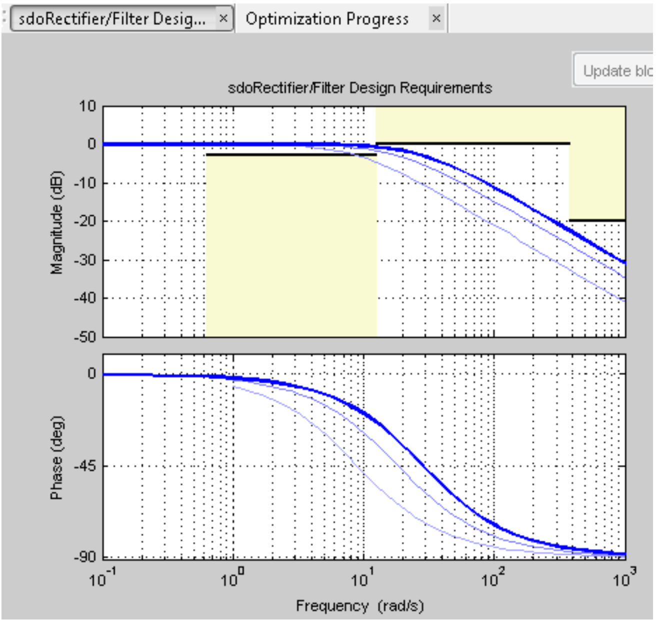


Iteration	F-count	MaxMinRL (min)	Filter Design Requireme... (<=0)	Filter Design Requireme... (>=0)
0	8	1.1000	8.1361	1.0067
1	16	0.3519	-0.8514	-0.8115
2	23	0.0420	0.9941	1.0000
3	31	0.0688	0.2360	0.9654
4	39	0.0794	0.0451	0.9415
5	46	0.0840	0.0056	0.9342
6	53	0.0840	0.0056	0.9342

The filter voltage signal in the `Voltage` scope window also settles to within its steady-state value in around 0.08 seconds without any overshoot and the harmonic content is significantly reduced from the initial design.



3 Verify that the model meets the gain and bandwidth requirements.



The plot displays the last five iterations. The final response using the optimized parameter values appears as the thick line.

The optimized response lies in the white region bounded by the design requirement line segments and thus meets the requirements.

4. Click `DesignVars` in **Design Optimization Workspace** and view the updated values in the **Variable Preview** area.

   The optimized values of the design variables are automatically updated in the Simulink model.

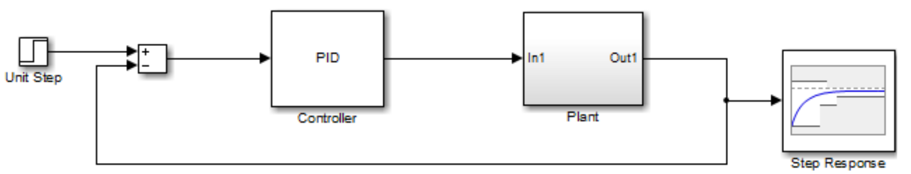
Time-Domain Model Verification

This example shows how to perform time-domain model verification using Simulink Design Optimization **Model Verification** blocks. During time-domain verification, the software monitors a signal to check if it meets time-domain characteristics such as step response characteristics and upper and lower amplitudes, or tracks a reference signal.

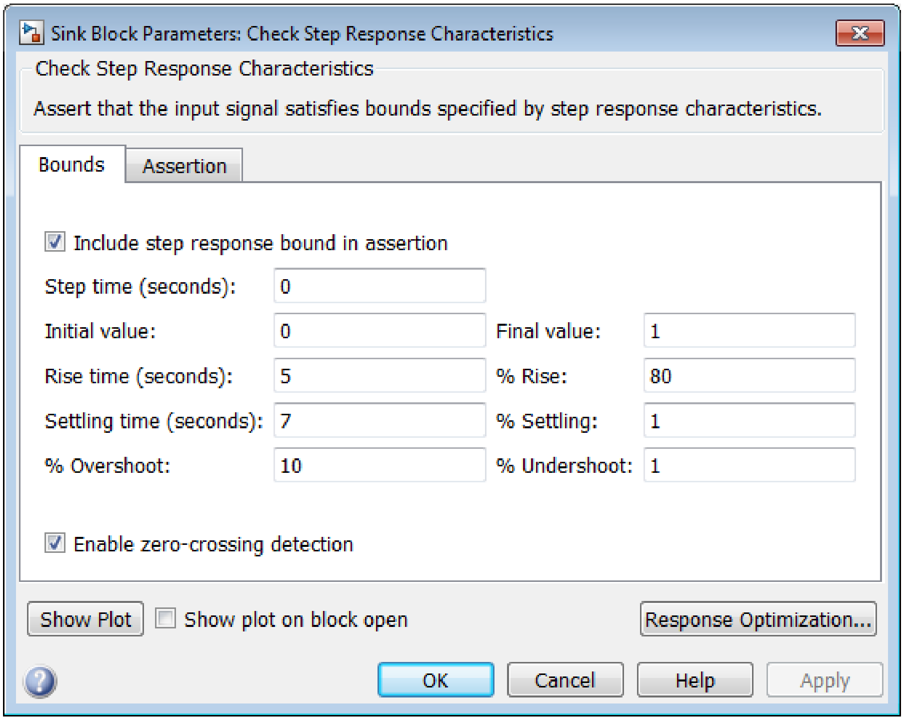
You can also use blocks from Simulink and Simulink Control Design **Model Verification** libraries to design complex assertion logic for time- and frequency-domain verification, and signal monitoring. If you have Simulink Verification and Validation software, you can construct simulation tests for your model using the Verification Manager tool in the Signal Builder.

1 Open Simulink model.

```
sys = 'sldo_model1_stepblk';
open_system(sys);
```



The model includes a Step Response block which is a Check Step Response Characteristics block from the Simulink Design Optimization **Model Verification** library and has default step response bounds.



2 In the Simulink Editor, click **Simulation** > **Run**.

The block asserts multiple times during simulation because the signal to which the block is connected violates the specified bounds. Assertion warnings appear in the MATLAB command window.

You can optimize model parameters to satisfy the bounds and eliminate assertion warnings. See “Design Optimization to Meet Step Response Requirements (GUI)” on page 3-4.

Optimization-Based Linear Control Design

When to Use Optimization-Based Linear Control Design

When you have Control System Toolbox software installed, you can design and optimize control systems for LTI models by optimizing controller parameters in the SISO Design Tool. To use optimization methods for linear control design, also known as *optimization-based tuning*, you must already have an initial controller. You can then use optimization-based tuning to refine the controller design to meet additional design requirements. For more information on designing controllers, see the Control System Toolbox documentation.

Note: Optimization-based tuning only changes the value of the controller parameters and not the controller structure itself.

Optimization-based tuning provides flexibility in terms of specifying additional design requirements for the controller. When you have a large number of design requirements, you can first design an initial controller by selecting a subset of requirements and subsequently select additional requirements to refine the design.

Optimization-based tuning also provides flexibility in terms of selecting a subset of controller parameters to optimize, and specifying bounds on the controller parameters.

To design linear controllers for Simulink models using optimization-based tuning, you must first linearize the model using the Simulink Control Design software. For more information on linearizing Simulink models, see the Simulink Control Design documentation.

Types of Time- and Frequency-Domain Design Requirements for Optimization-Based Control Design

When you design linear controllers for LTI or Simulink models using the Simulink Design Optimization software, you can specify both time- and frequency-domain requirements on the system response. You can specify design requirements on the following plots:

- Root Locus plot
- Open-Loop and Prefilter Bode plots
- Open-Loop Nichols plot
- Step/Impulse Response plots

For more information, see "Time- and Frequency-Domain Requirements in SISO Design Tool".

Simulink Design Optimization software uses the frequency-domain requirements to compute the frequency response of the system. It then uses optimization methods to reduce the distance between the current response and the requirements by modifying the controller parameters. The software does not change the controller structure when optimizing the controller parameters.

Quick Start — Optimization-Based Linear Control Design

In this quick start, you get an overview of the typical tasks for optimization-based linear control design using the SISO Design Tool:

1. Open a SISO Design Tool session.
2. Configure a project for optimization-based control design.
3. Specify the controller parameters to design.
4. Specify the design requirements.
5. Design the controller.
6. Evaluate the controller design.

Note: The same workflow applies to optimization-based control design for LTI models created at the command line using Control System Toolbox software. To learn how to create LTI models, see "Linear (LTI) Models" in the Control System Toolbox documentation.

Prerequisites for optimization-based linear control design include:

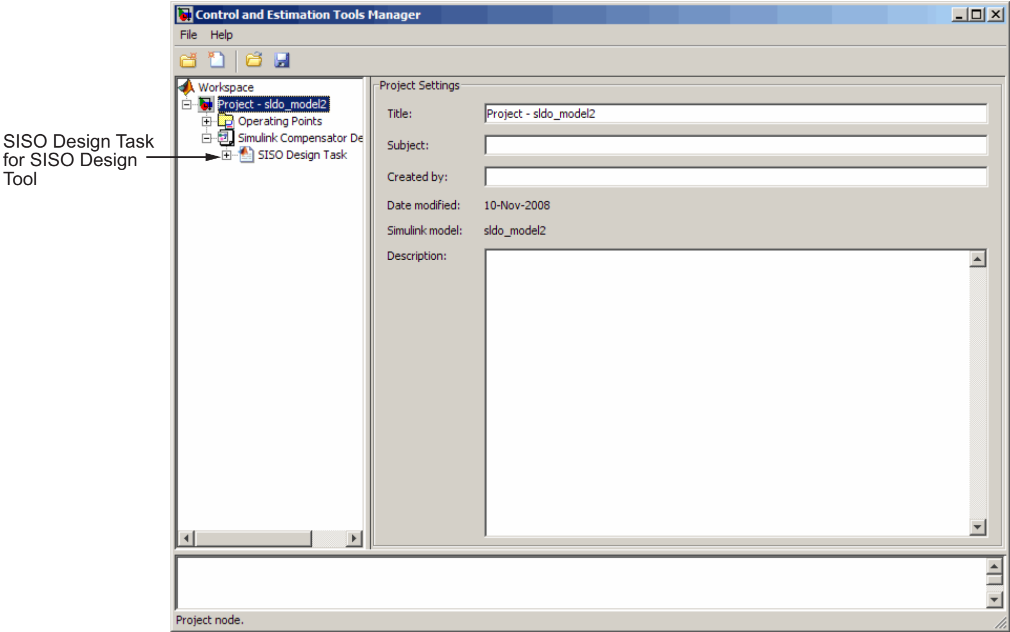
- Simulink Compensator Design Task that contains a linearized version of the Simulink model and, optionally, any response plots you configure.

  For more information on how to linearize a Simulink model for control design, see the Simulink Control Design documentation.
- Time- and frequency-domain design requirements

To design a controller using optimization methods:

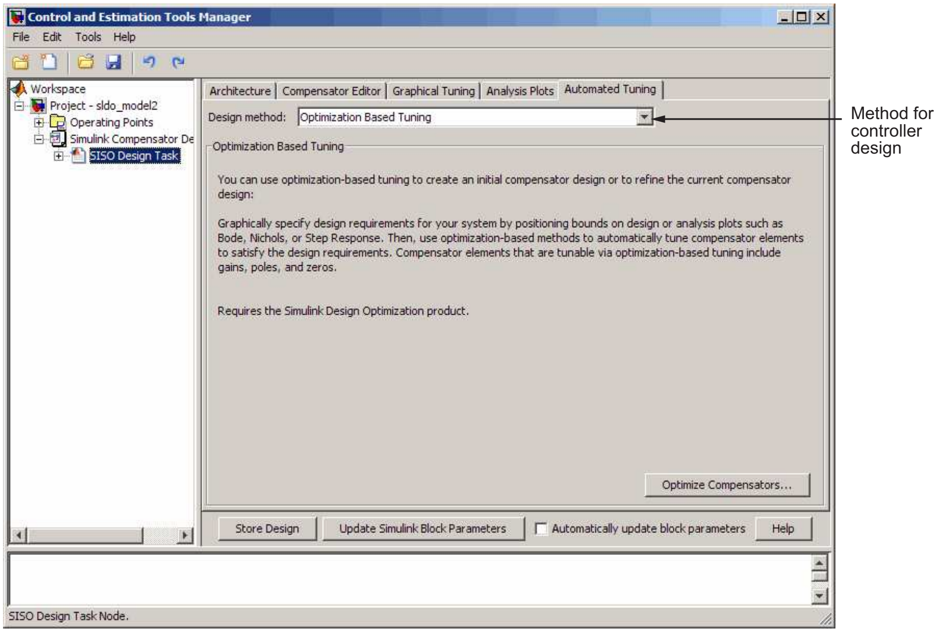
1. Open a SISO Design Tool session by typing the following command at the MATLAB prompt:

   ```
   sisotool('projectname.mat')
   ```

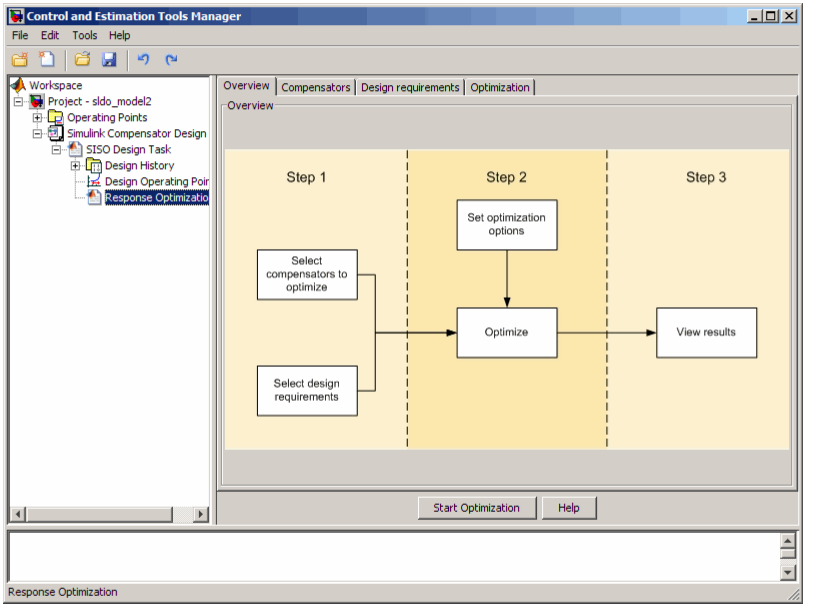


The command also opens a SISO Design for SISO Design Task window by default and any response plots you configured when you linearized the Simulink model using Simulink Control Design software.

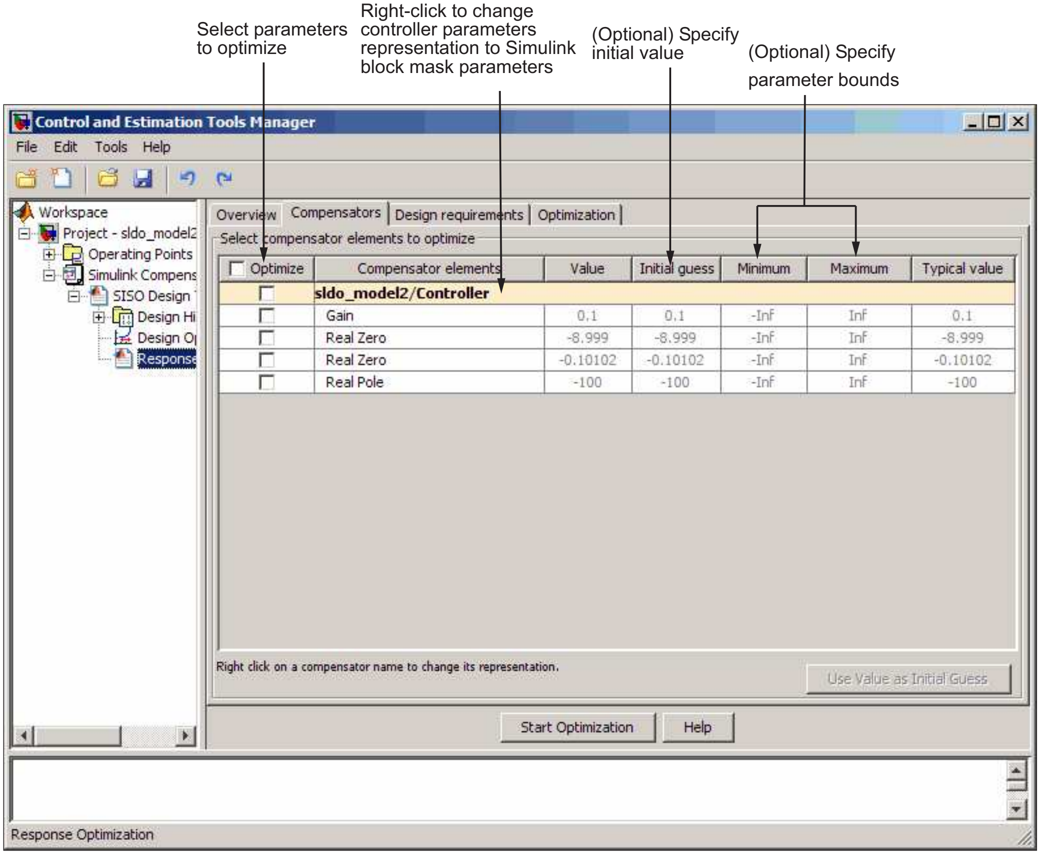
2 Configure a project for optimization-based control design by clicking **Optimize Compensators** in the Automated Tuning tab of the **SISO Design Task**.



This action creates a new **Response Optimization** node in the Control and Estimation Tools Manager.

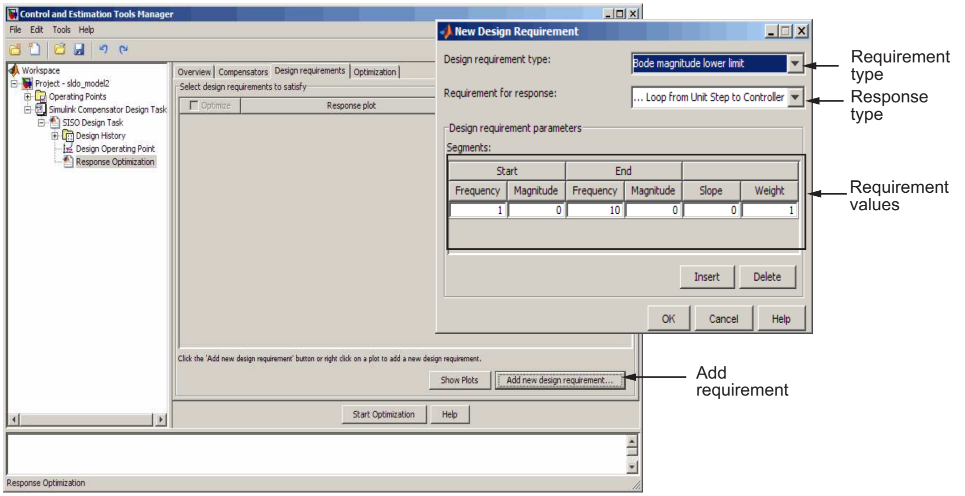


3 Specify the controller parameters to design in the **Compensators** tab.

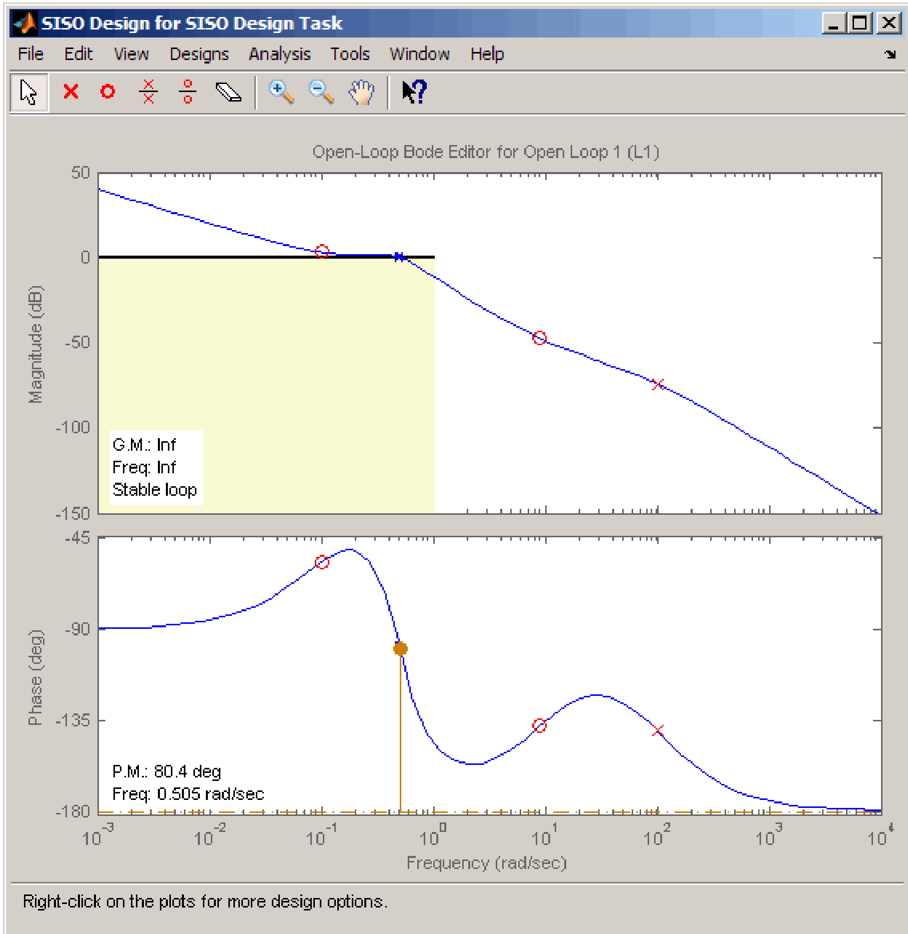


4 Specify the design requirements.

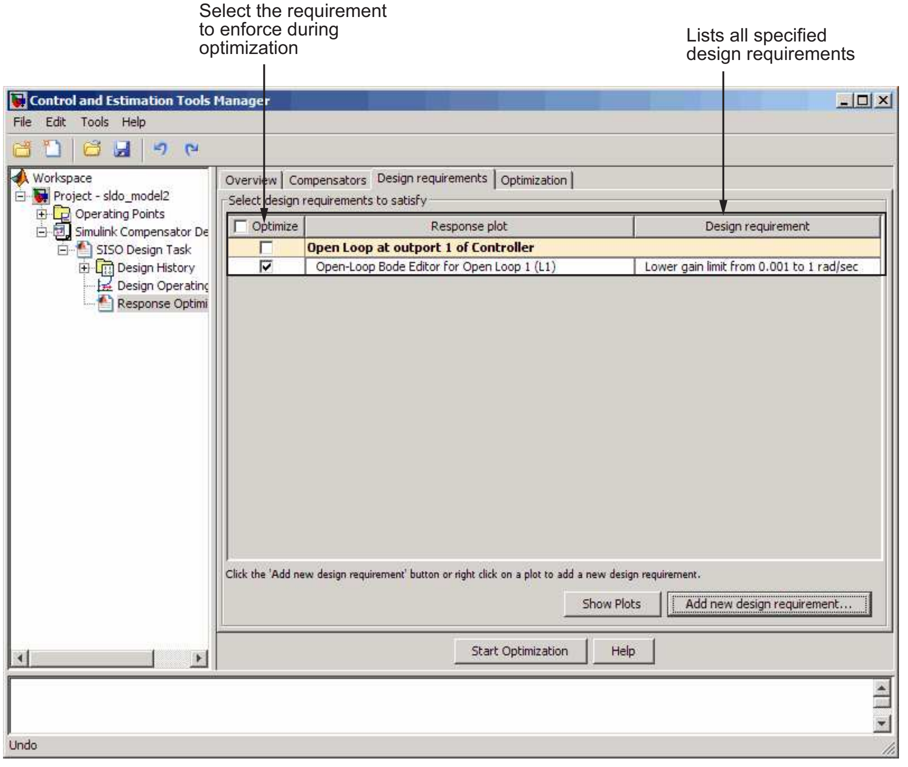
a In the **Design requirements** tab, click **Add new design requirement**. Specify the design requirements, for example Bode magnitude lower limit, in the New Design Requirement dialog box.



In the SISO Design window, the yellow region with the black line segment represents the design requirement on the response plot.

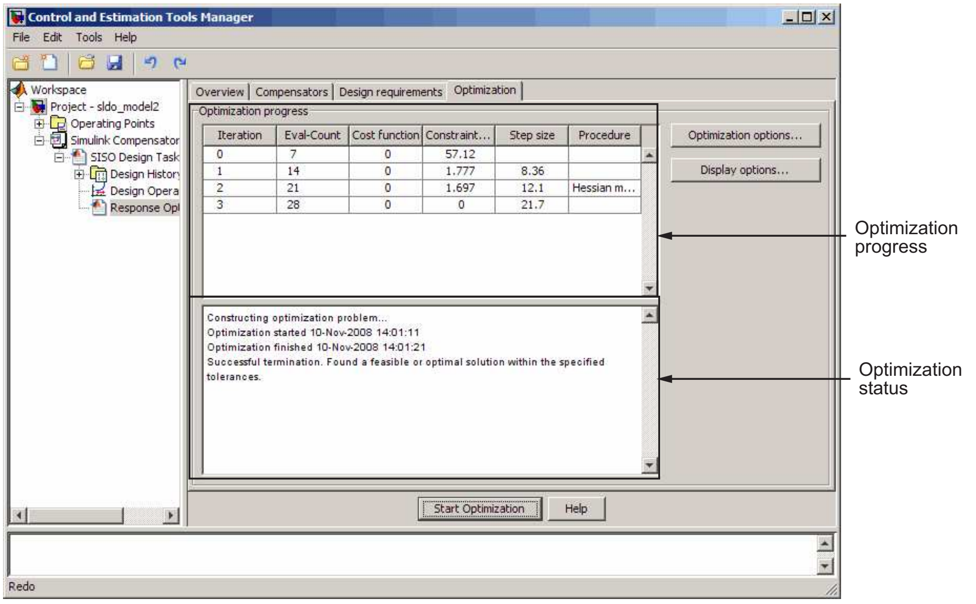


The **Design Requirements** tab also lists the design requirement.



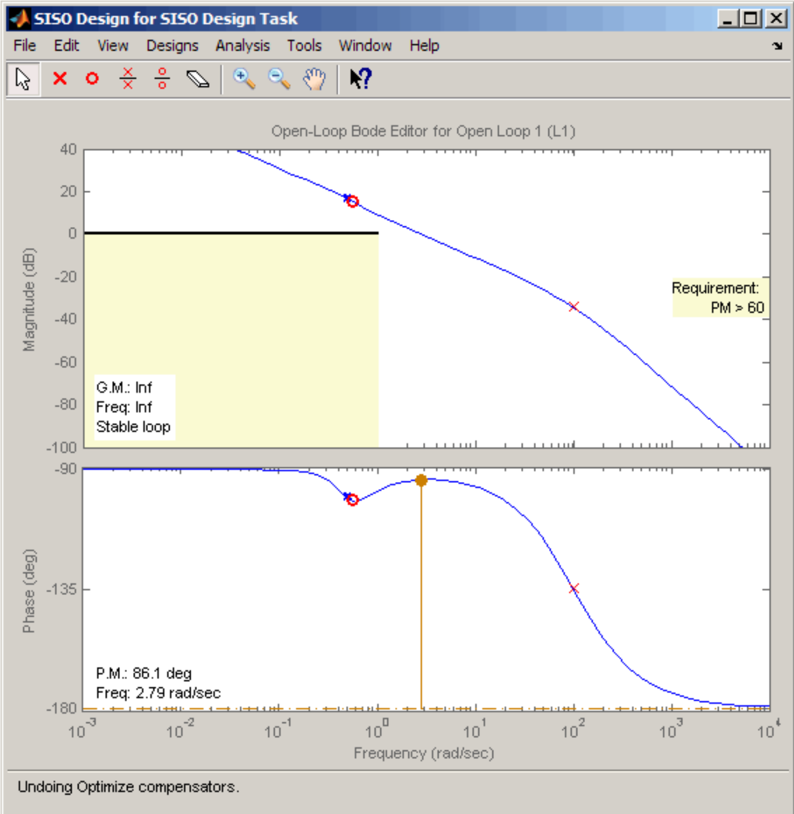
b Repeat step a to specify additional time- and frequency-domain requirements.

5 Start the optimization to design the controller by clicking **Start Optimization** in the **Optimization** tab.

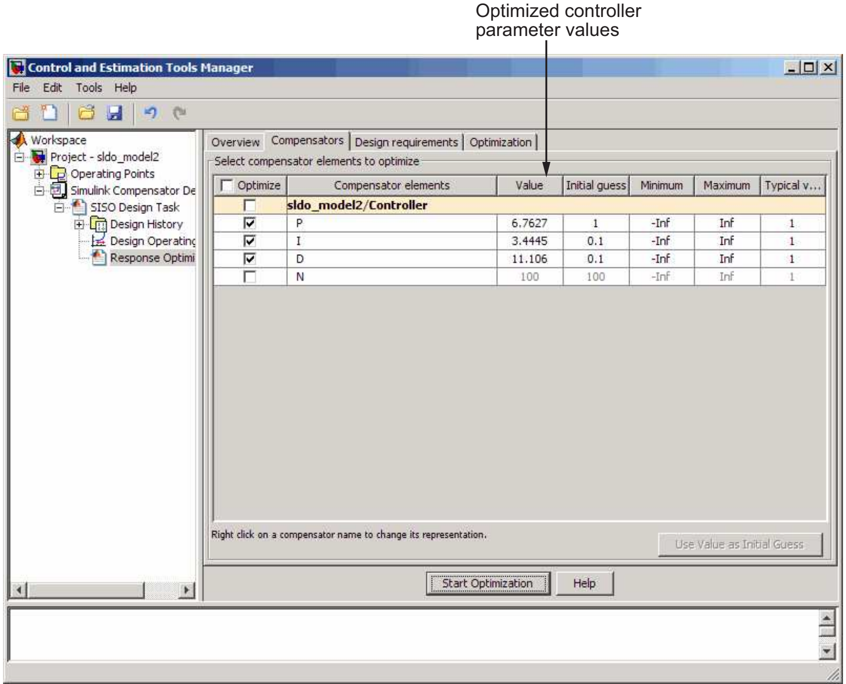


6 Evaluate the controller design.

a Examine the system's response in the response plot, for example the Bode plot, to see if it meets the requirements. The system's response must lie in the white region in order to meet the design requirement.



b Examine the controller parameter values in the **Compensator** tab.



7 Write the controller parameter values into the Simulink model. To do so, click **Update Simulink Block Parameters** in the **SISO Design Task** node.

See Also: "Design Optimization-Based PID Controller for Linearized Simulink Model (GUI)".

Design Optimization-Based PID Controller for Linearized Simulink Model (GUI)

In this section...

About This Tutorial

Objectives

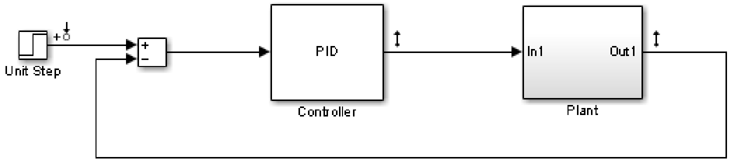
In this tutorial, you learn to use optimization methods to design a PID controller to meet frequency-domain design requirements on a system's response.

You accomplish the following tasks using the GUI:

- Specify frequency-domain Bode magnitude and phase margin requirements.
- Design an initial controller to meet the frequency-domain requirements.
- Refine the initial controller design to limit the controller's output signal.

About the Model

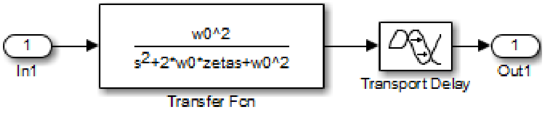
In this tutorial, you use the Simulink model named `sldo_model2`, as shown in the next figure.



The model contains a `Controller` block, which is a PID Controller. This block controls the output of the `Plant` subsystem.

Using the Simulink Control Design software has linearized the Simulink model at the operating point specified in the model. The `sldo_model2.mat` file contains a preconfigured SISO Design Tool session saved after linearizing the model. To learn more about linearizing Simulink models for control design, see “Control Design”.

Double-click the `Plant` subsystem to open it. The plant is modeled as a second-order system with delay. It contains Transfer Function and Transport Delay blocks, as shown in the next figure.



To learn more about the blocks, see “Transfer Fcn” and “Transport Delay” block reference pages.

Design Requirements

The compensator you design in this tutorial must meet the following design requirements:

- Bode lower magnitude bound of 0 in the frequency range 1e-3 to 1 rad/sec
- Phase margin greater than 60 degrees
- Controller output bounds in the range [-250 550]

Configuring a Project for Optimization-Based Control Design

To design a linear controller for a Simulink model, you must first configure a Control and Estimation Tools Manager project.

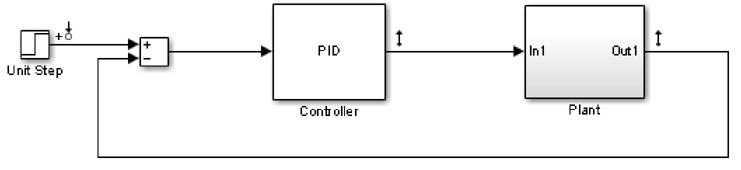
1 Open a SISO Design Tool session for the linearized Simulink model by typing the following command at the MATLAB prompt:

```
sisotool('sldo_model2.mat')
```

Note: `sldo_model2.mat` file contains a preconfigured SISO Design Tool session. This session was saved after Simulink Control Design software linearized the `sldo_model2` Simulink model.

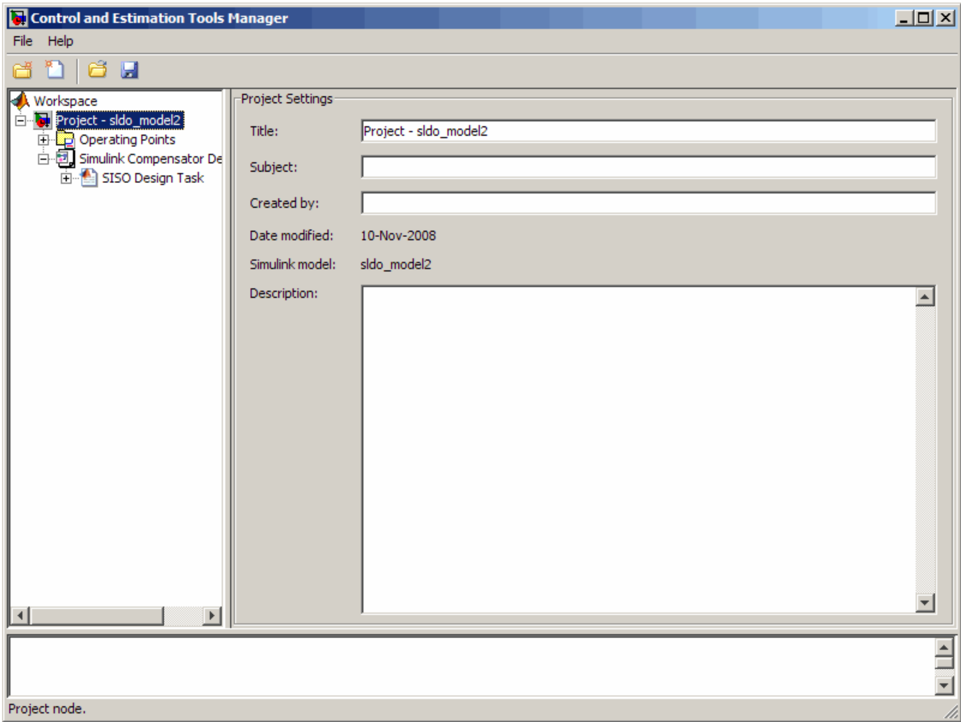
This command opens the following windows:

- Simulink model

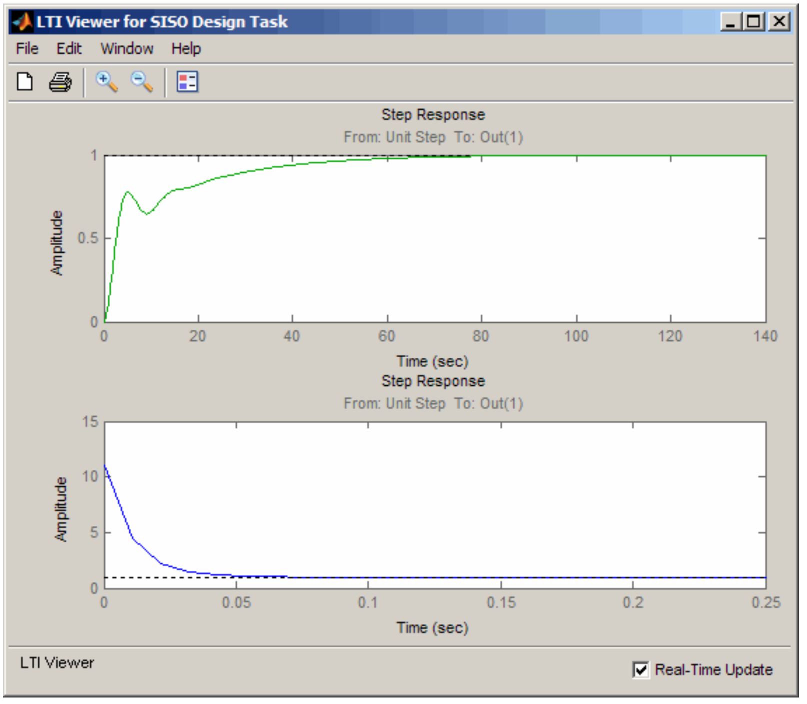


  To learn more about the model, see "About the Model" on page 4-15.

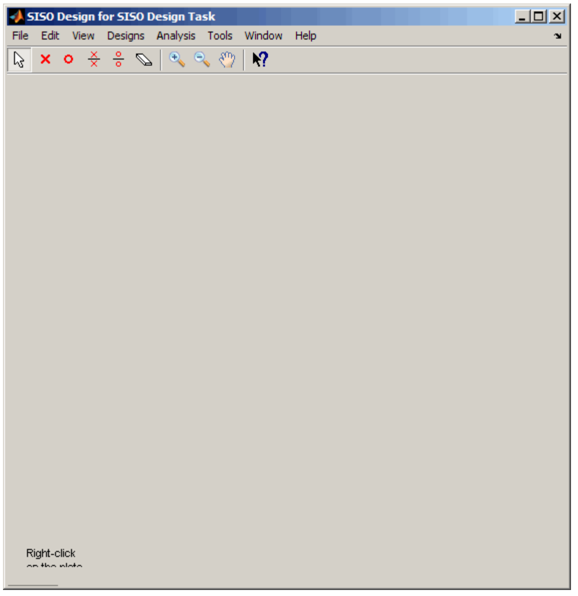
- Control and Estimation Tools Manager GUI, which contains a **SISO Design Task** node under the **Simulink Compensator Design Task** node.



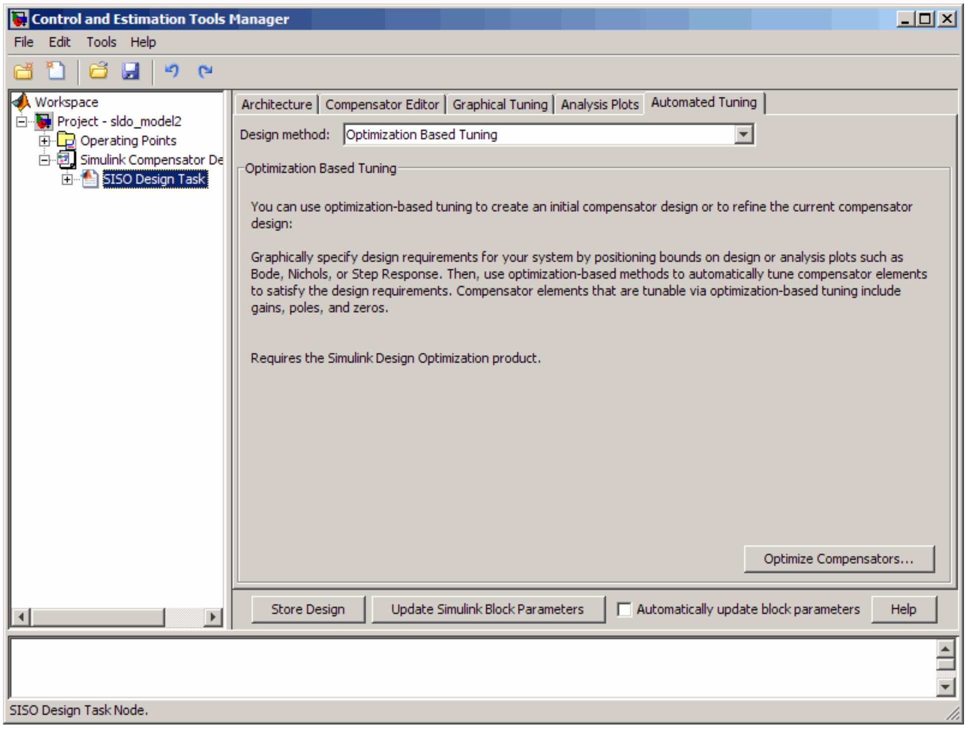
- LTI Viewer for SISO Design Task window, which contains the following plots:
  - Closed-loop step response of the system in the top plot
  - Output of the `Controller` block in the bottom plot



- A blank SISO Design for SISO Design Task window

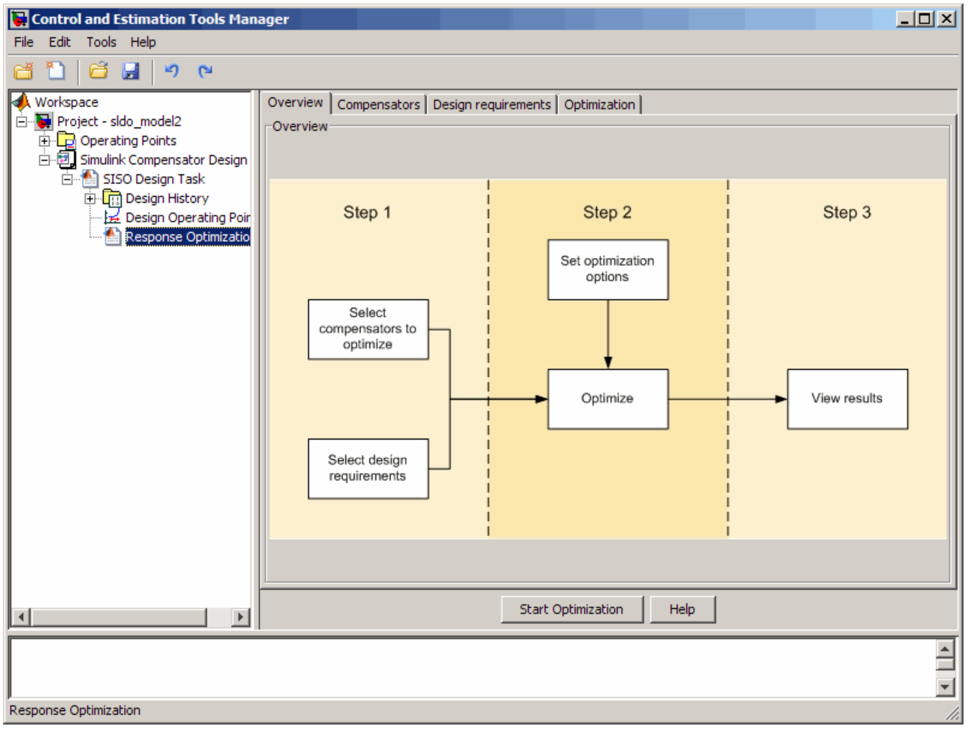


2 In the Control and Estimation Tools Manager GUI, select the **Automated Tuning** tab in the **SISO Design Task** node.



3 Click **Optimize Compensators**.

This action creates a new **Response Optimization** node.



Designing an Initial PID Controller to Meet Bode Magnitude and Phase Margins Requirements

- "Specifying the Controller Parameters" on page 4-22
- "Specifying Bode Magnitude and Phase Margin Design Requirements" on page 4-26
- "Designing the Controller" on page 4-36

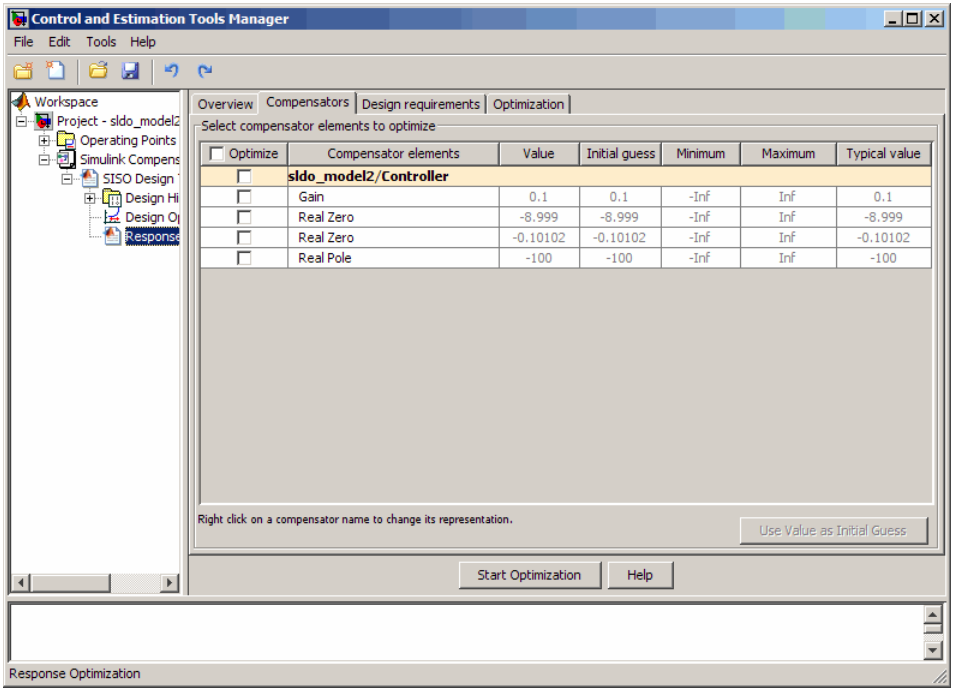
Specifying the Controller Parameters

In this portion of the tutorial, you specify the controller parameters to design.

You must have already configured a project for control design, as described in "Configuring a Project for Optimization-Based Control Design" on page 4-16.

To specify the controller parameters to design:

1 In the Control and Estimation Tools Manager GUI, select the **Compensators** tab in the **Response Optimization** node.



The controller parameters appear as poles and zeros in the **Compensator elements** column and represent the following:

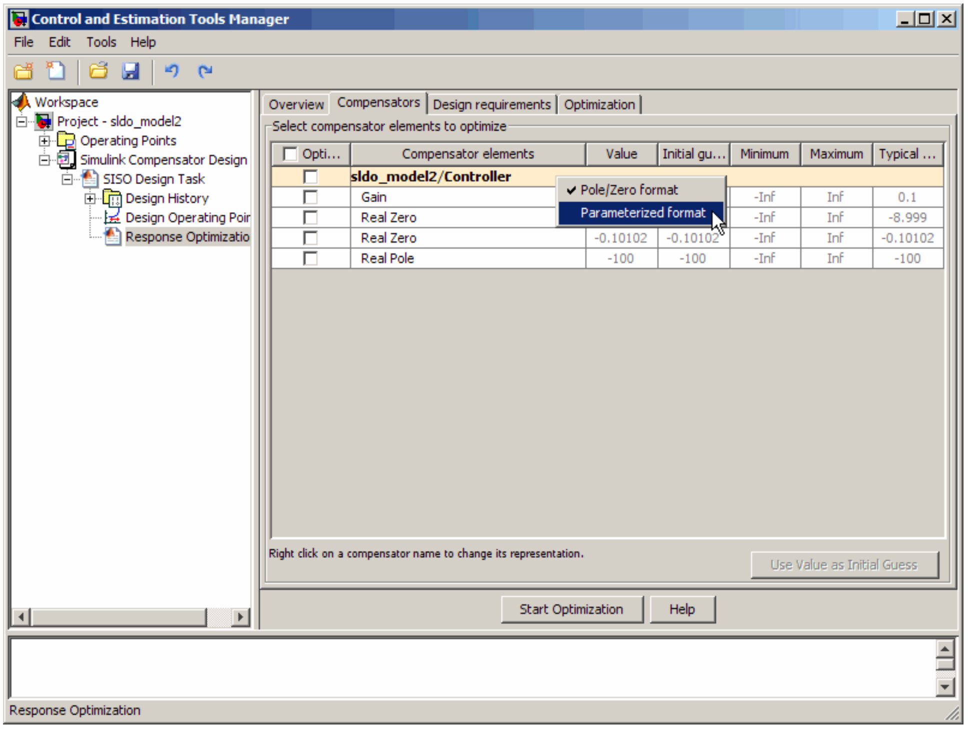
- Gain — Overall gain of the controller
- Real zeros — Zeros resulting from the differentiator and integrator
- Real pole — Pole resulting from the low-pass filter of the differentiator

---

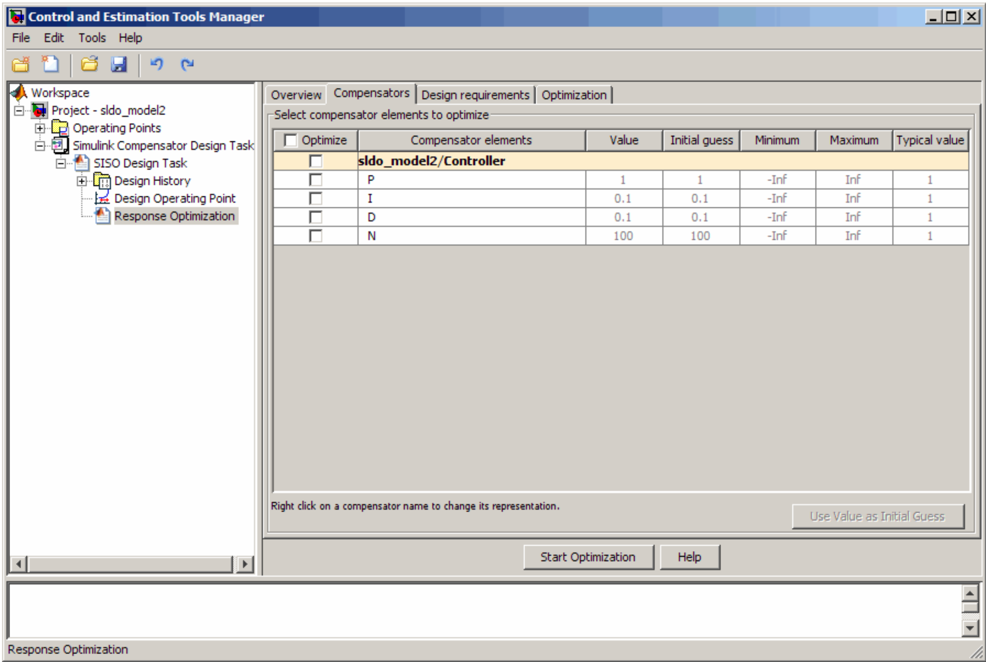
Tip To view the structure of the `Controller` block, right-click the block and select **Mask > Look Under Mask**.

---

2 For convenience, change the PID controller parameters to Simulink block mask parameters format. To do so, right-click the **sldo_model2/Controller** column, and select **Parameterized format**.



This action displays the controller parameters as Simulink block mask parameters `P`, `I`, and `D`, as shown in the next figure. To learn more about mask parameters, see "Mask Parameters" in the Simulink documentation.



The **Compensators** tab displays the following parameter settings:

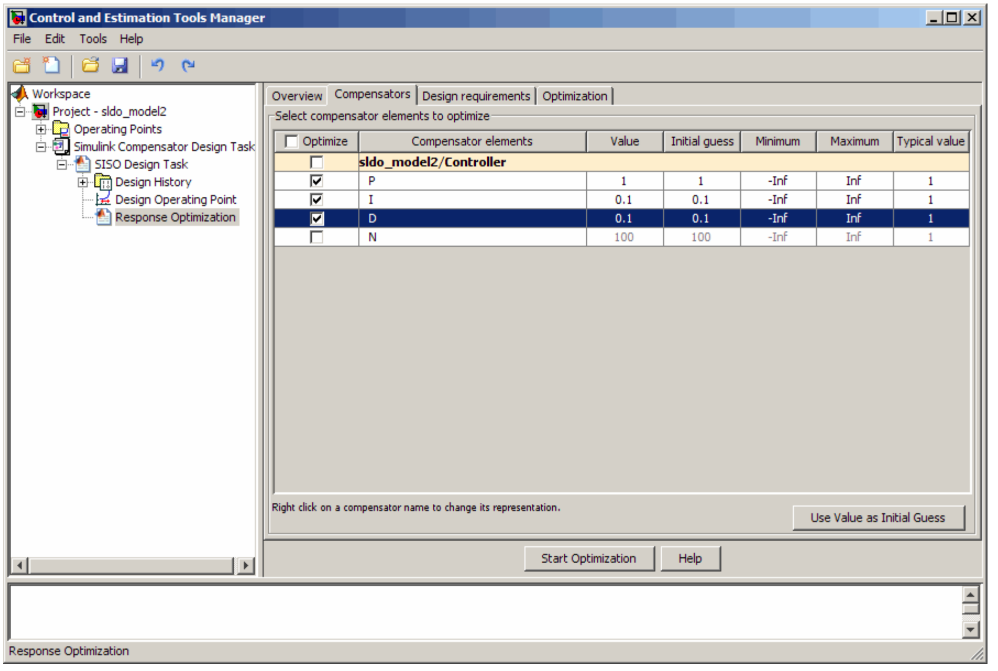
- **Value** — Current controller parameter value
- **Initial Guess** — Initial controller parameter value
- **Minimum** and **Maximum** — Controller parameter bounds
- **Typical Value** — Scaling factor for the controller parameter

---

Note: Compensator elements or parameters cannot have uncertainty when used with frequency-domain based response optimization.

---

3 In the **Optimize** column, select the check boxes for `P`, `I`, and `D`.



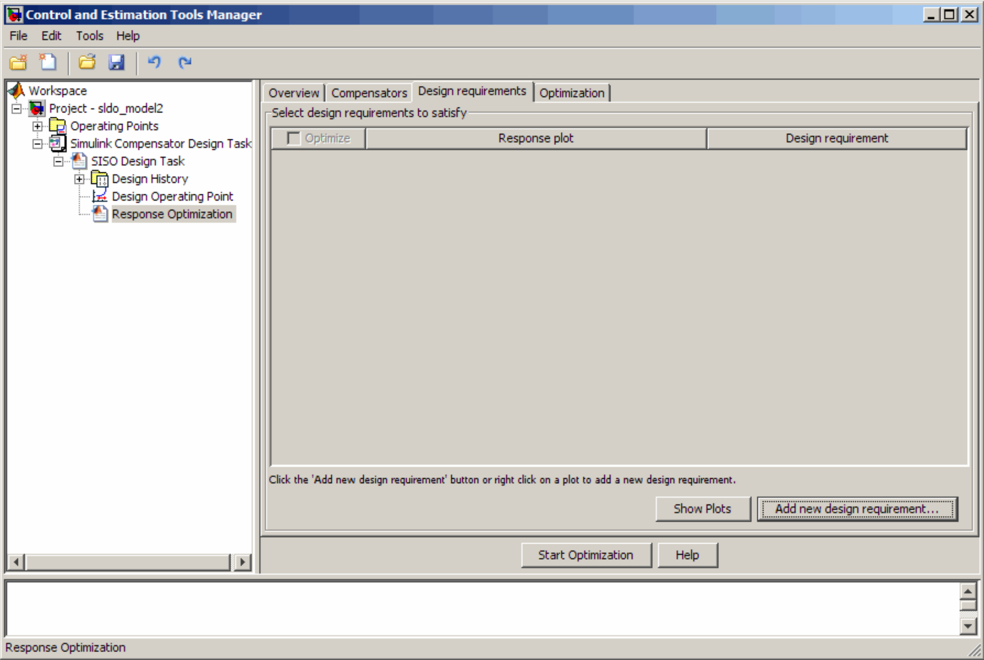
Specifying Bode Magnitude and Phase Margin Design Requirements

In this portion of the tutorial, you specify the Bode magnitude and phase margin requirements that the controller must satisfy.

Before you specify the design requirements, you can specify the controller parameters to design, as described in "Specifying the Controller Parameters" on page 4-22.

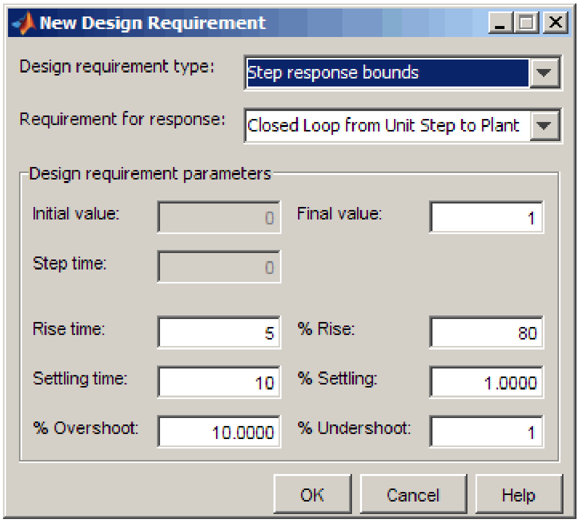
To specify the Bode design requirements:

1 In the **Response Optimization** node of the Control and Estimation Tools Manager GUI, select the **Design requirements** tab.



2 Click **Add new design requirement**.

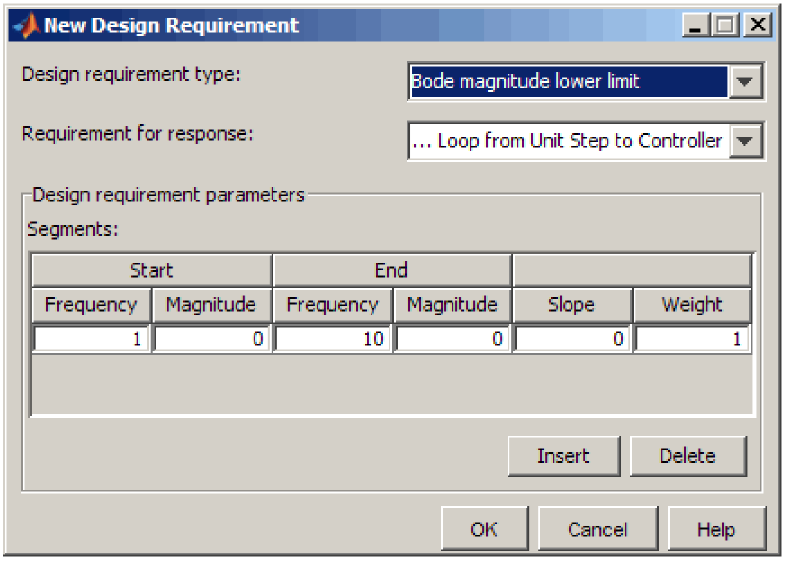
This action opens the New Design Requirement dialog box.



3 Specify the Bode magnitude lower limit requirement:

a In the New Design Requirement dialog box, select `Bode magnitude lower limit` from the **Design requirement type** drop-down list.

This action updates the New Design Requirement dialog box, as shown in the next figure.

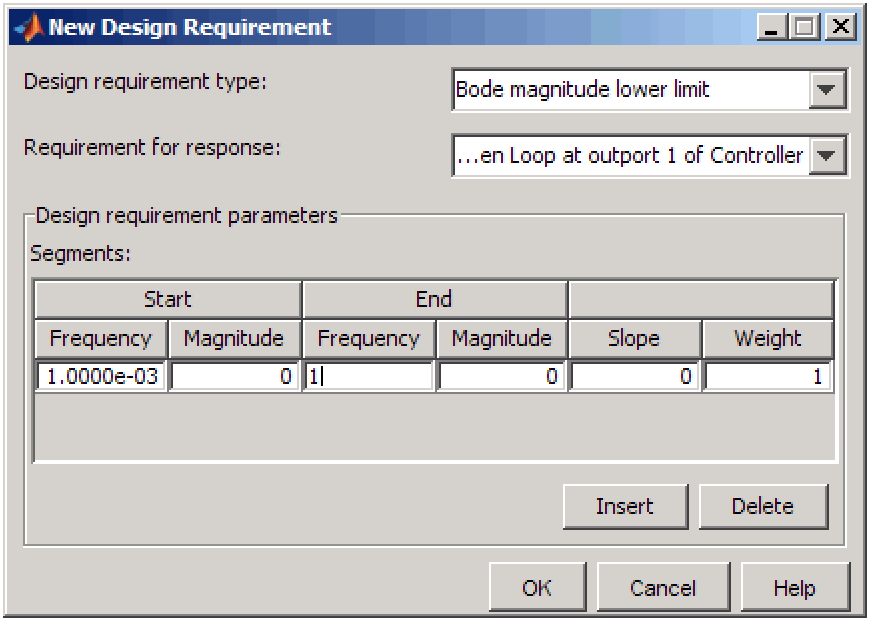


b Select `Open Loop at outport 1 of Controller` from the **Requirement for response** drop-down list.

c In the **Frequency** field of the **Start** column, enter `1e-3`.

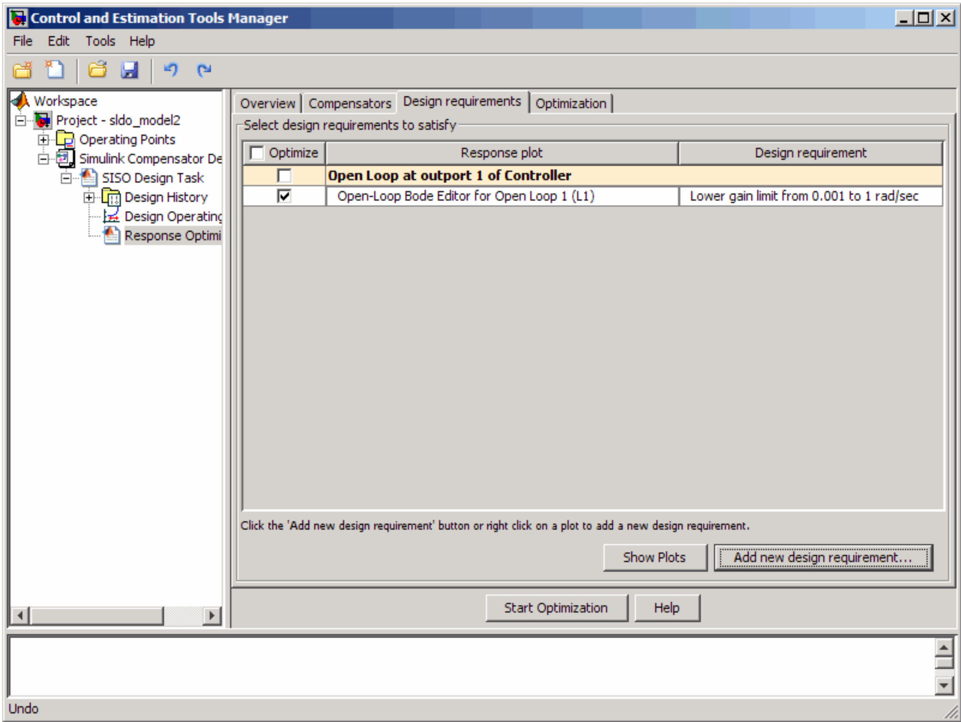
d In the **Frequency** field of the **End** column, enter `1`.

The New Design Requirement dialog box resembles the following figure.

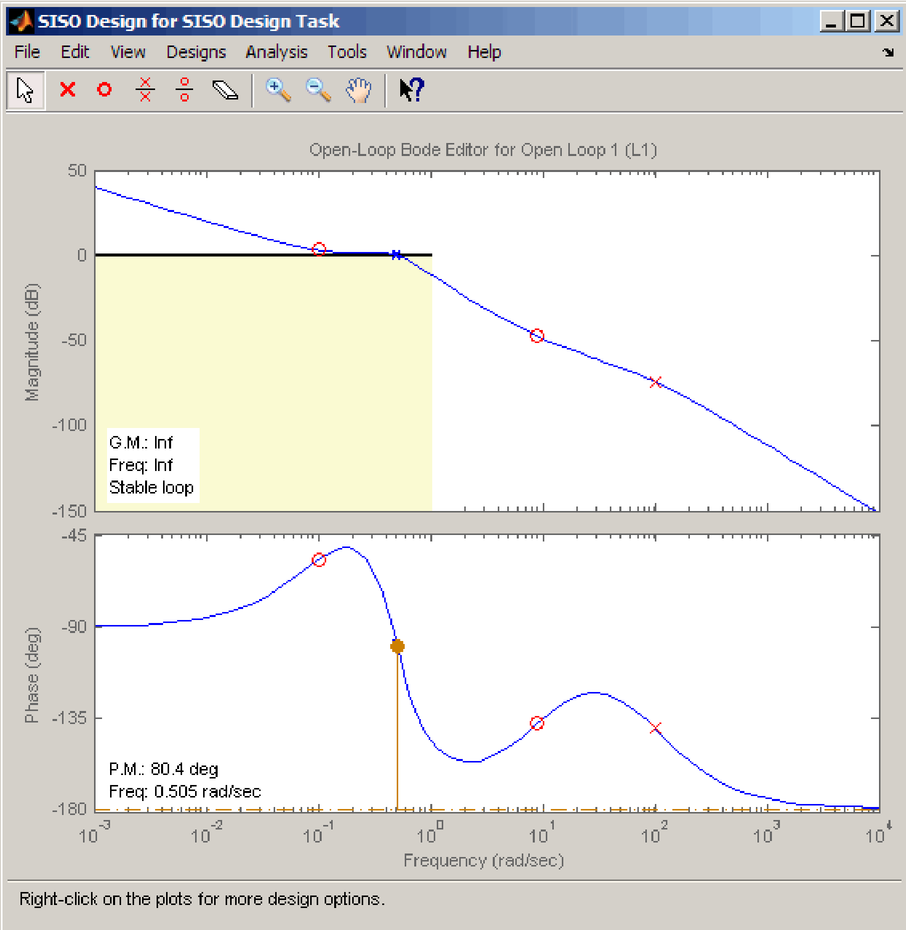


e Click **OK** to close the New Design Requirement dialog box.

The Bode lower magnitude limit is added to the **Design requirements** tab, as shown in the next figure.

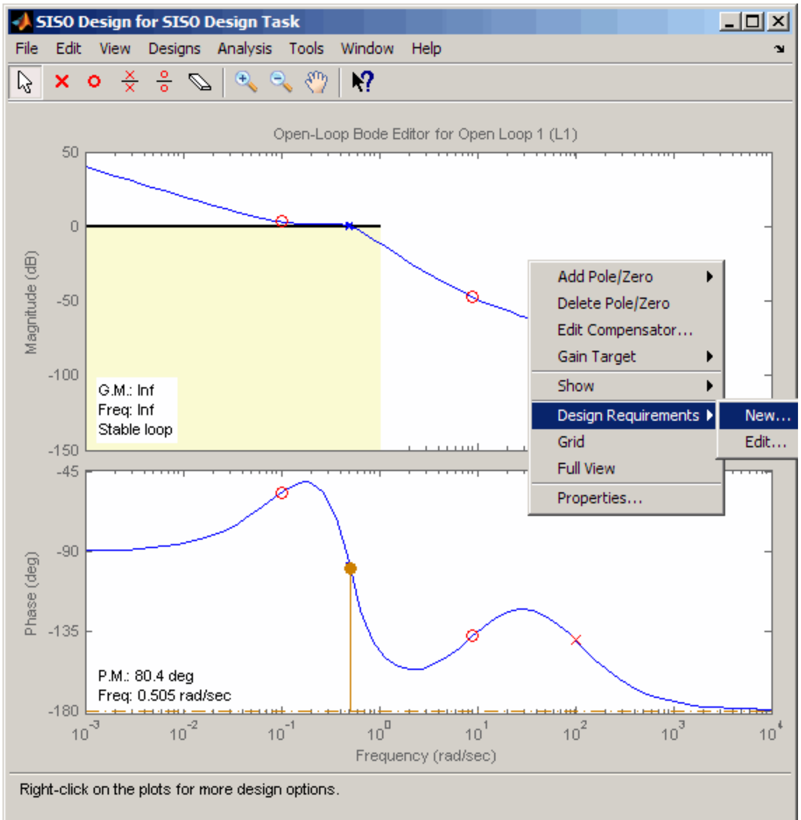


The SISO Design for SISO Design task window also updates to show the Bode plot with the design requirement displayed as the black line segment.

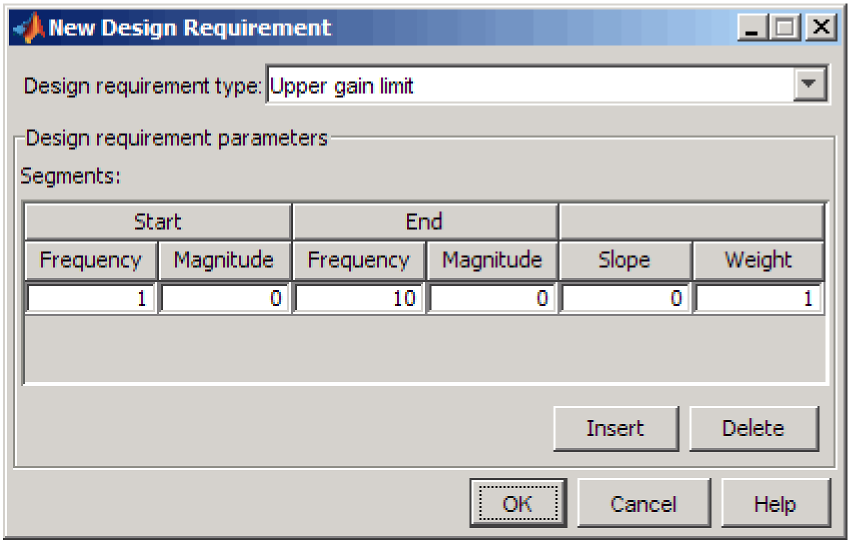


4 Add the phase margin requirement:

a In the SISO Design window, right-click the white area on the top plot, and select **Design requirement** > **New**.

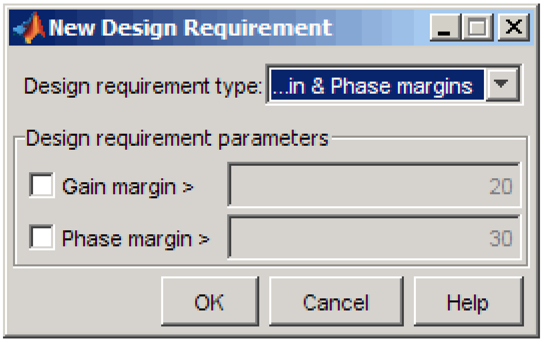


This action opens the New Design Requirement dialog box, as shown in the next figure.

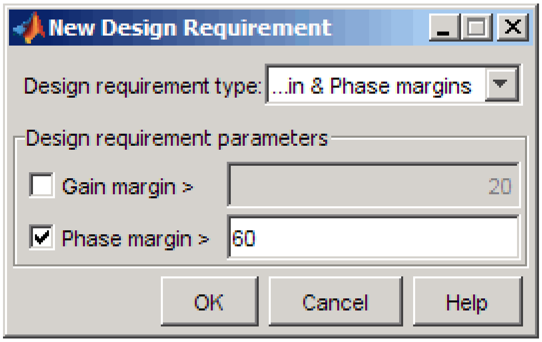


b In the New Design Requirement dialog box, select `Gain & phase margins` from the **Design requirement type** drop-down list.

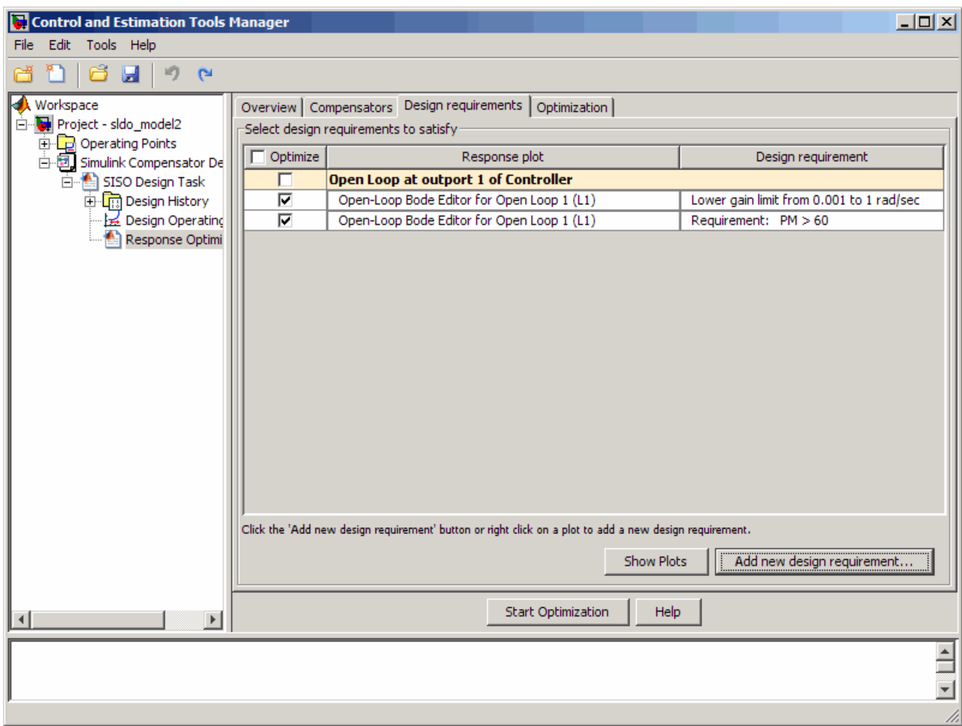
The New Design Requirement dialog box updates to display the **Gain Margin >** and **Phase Margin >** options, as shown in the next figure.



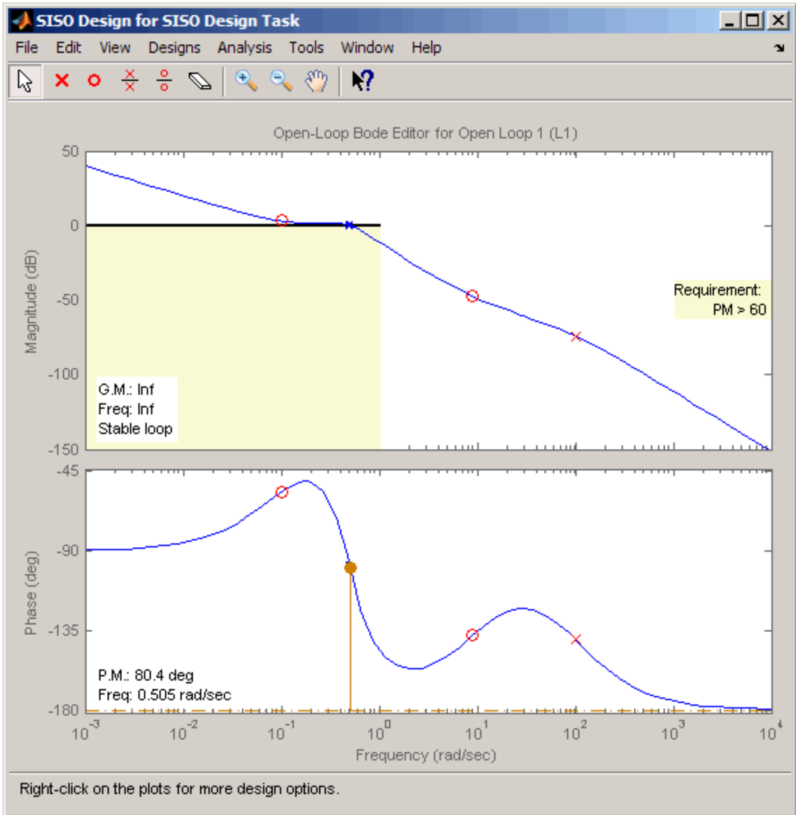
c Select the **Phase margin >** check box, and enter `60` in the adjacent field. Then, click **OK** to close the dialog box.



The **Design requirements** tab in the Control and Estimation Tools Manager GUI updates. It now displays the phase margin requirement, as shown in the next figure.



The SISO Design window also updates to display the phase margin requirement.



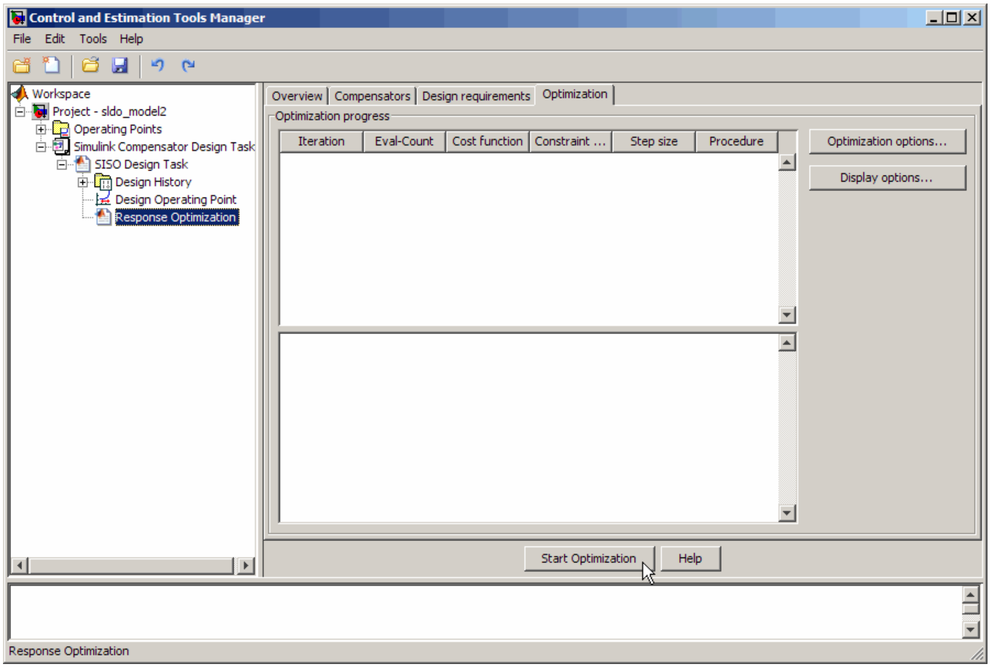
Designing the Controller

In this portion of the tutorial, you design the controller to meet the Bode magnitude and phase requirements.

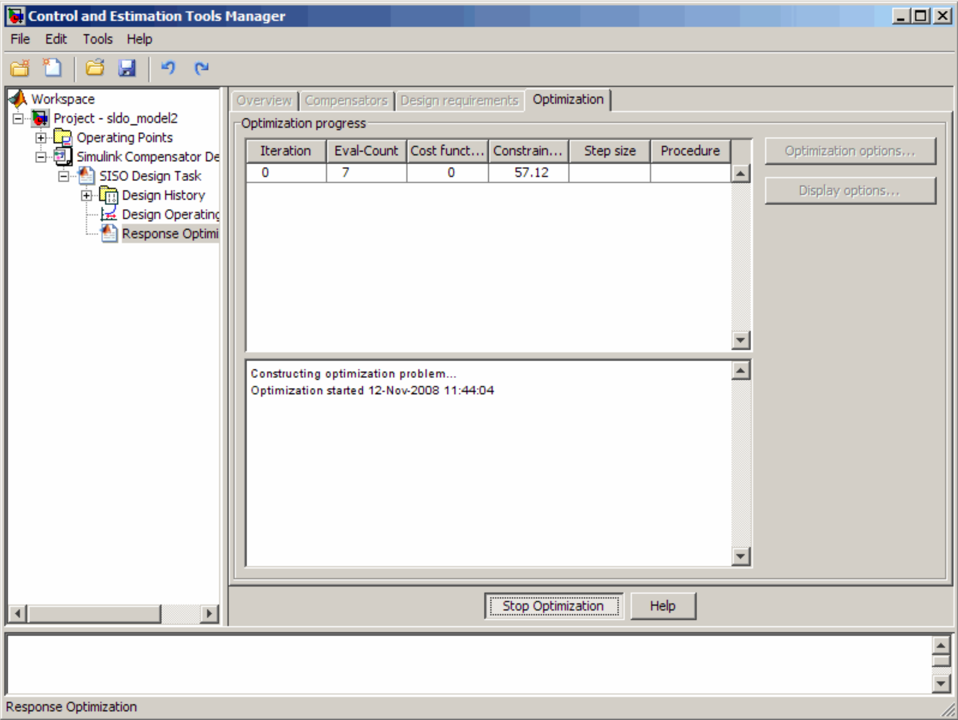
You must have already specified the controller parameters to design, as described in "Specifying the Controller Parameters" on page 4-22, and the design requirements, as described in "Specifying Bode Magnitude and Phase Margin Design Requirements" on page 4-26.

To design the controller:

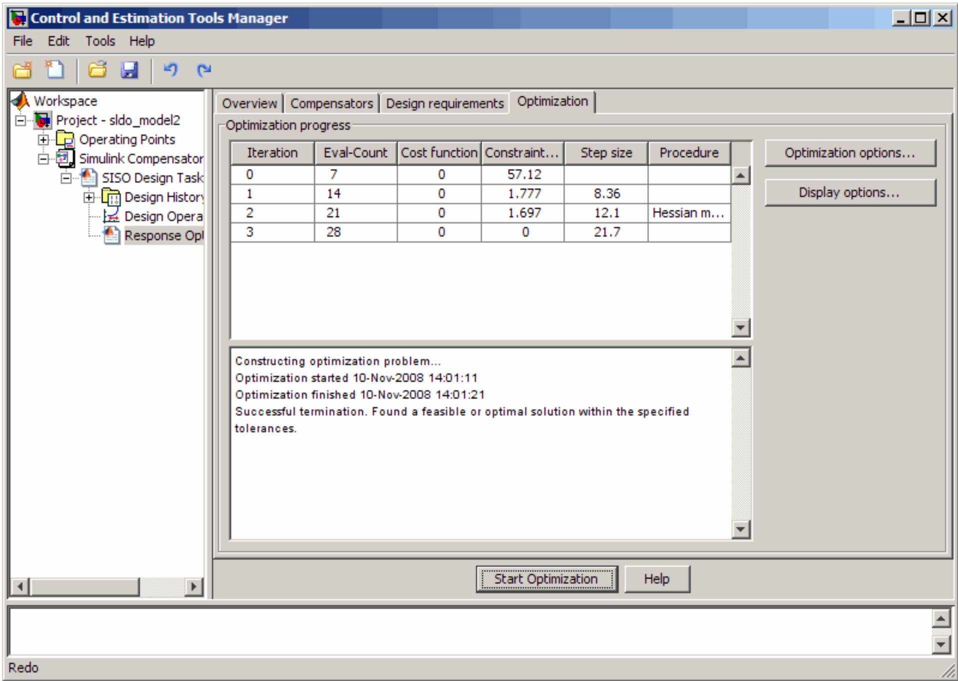
1 In the **Optimization** tab, click **Start Optimization**.



The **Optimization** tab updates as shown in the next figure.



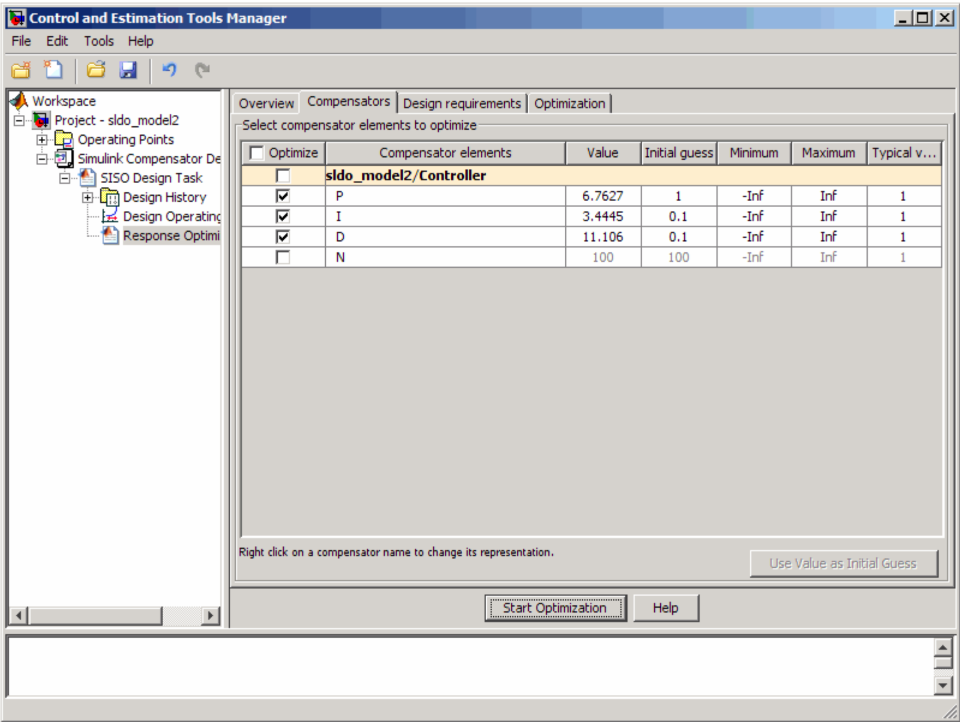
- At every optimization iteration, the default optimization method `Gradient descent` reduces the distance between the current response and the magnitude requirement line segment by modifying the controller parameters. Simultaneously, the software also computes the phase margin and reduces the distance between the current response and the phase margin. To learn more about the optimization method, see "Selecting Optimization Methods".
- After the optimization completes, the **Optimization** tab displays the optimization iterations and status, as shown in the next figure.



The message `Successful termination` indicates that the optimization method found a solution that meets the design requirements. For more information about the outputs displayed in the **Optimization progress** table, see "Iterative Display" in the Optimization Toolbox documentation.

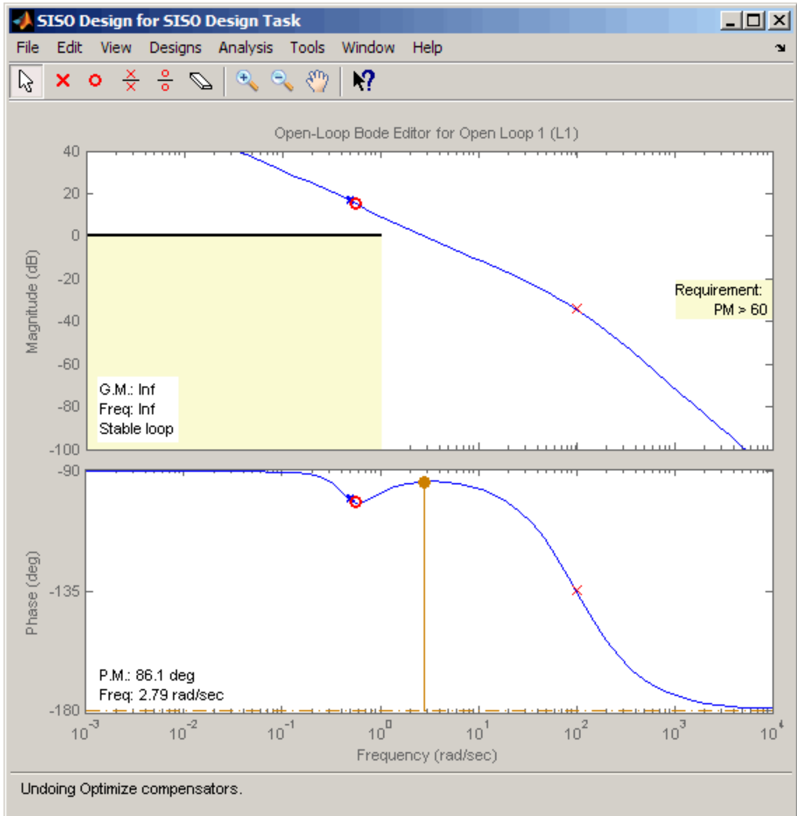
2 Examine the controller parameters and the system's response:

   a View the optimized parameter values in the **Value** field of the **Compensator** tab.



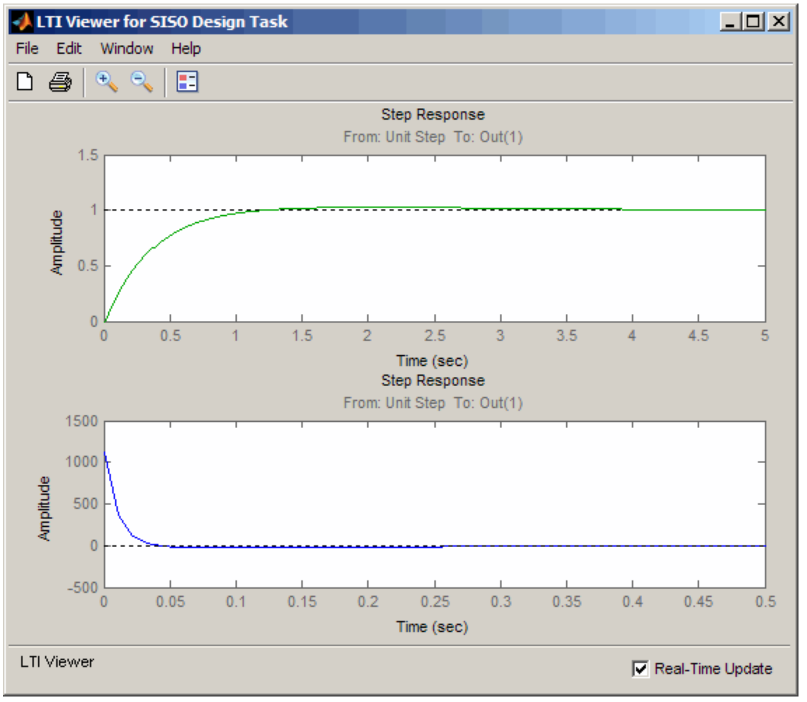
b Examine the system's response on the following plots:

- The SISO Design window



- The top plot shows that the magnitude of the system, displayed as the blue curve, lies outside the yellow region. This plot indicates that the system has met the Bode magnitude requirement.
- The bottom plot displays the phase margin (P.M.) value of 86.1 degrees. This indicates that the system has met the phase margin design requirement of >60 degrees.

- The LTI Viewer



- The top plot shows that the closed-loop response of the system is stable. The system with the designed controller thus meets both the magnitude and phase margin requirements.
- The bottom plot in the LTI Viewer shows that the peak value of the controller's output is 1000, which is very large and can cause damage to the plant. To limit the controller output, you apply lower and upper bounds on the signal, as described in "Refining the Controller Design to Meet Controller Output Bounds" on page 4-42.

Refining the Controller Design to Meet Controller Output Bounds

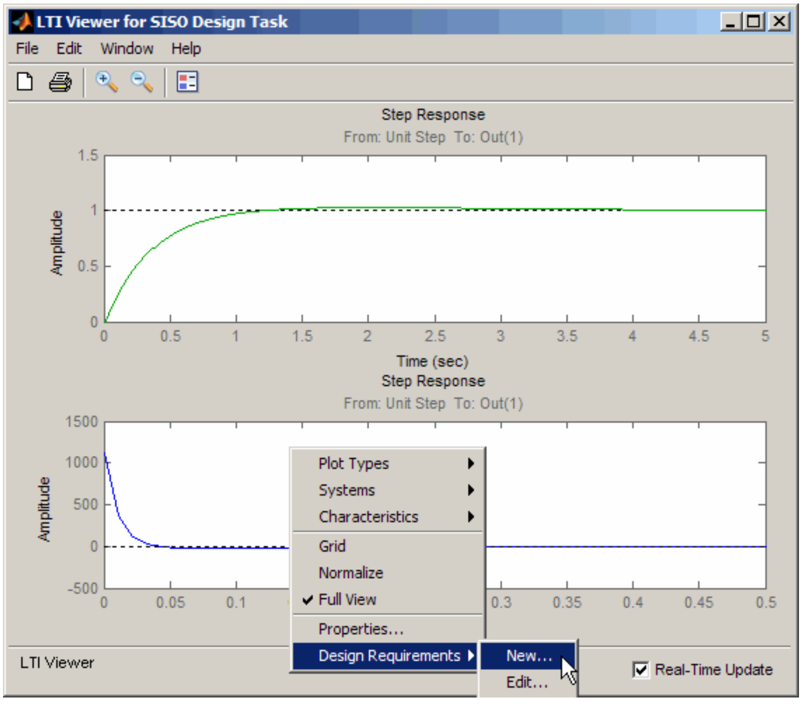
In this portion of the tutorial, you refine the controller to satisfy bounds on the controller's output.

You must have already designed an initial controller, as described in “Designing an Initial PID Controller to Meet Bode Magnitude and Phase Margins Requirements” on page 4-22.

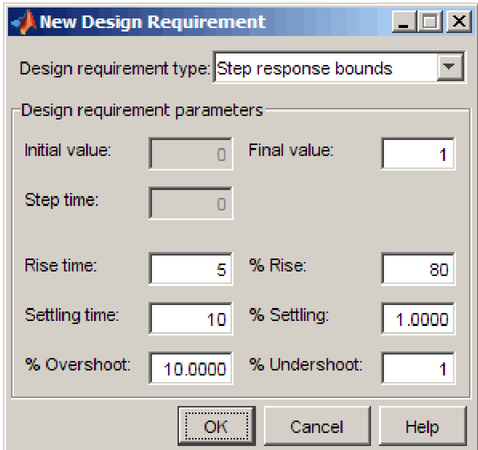
To tune the compensator parameters to meet the bounds on the controller's signal:

1 Add the upper-bound on the controller's output:

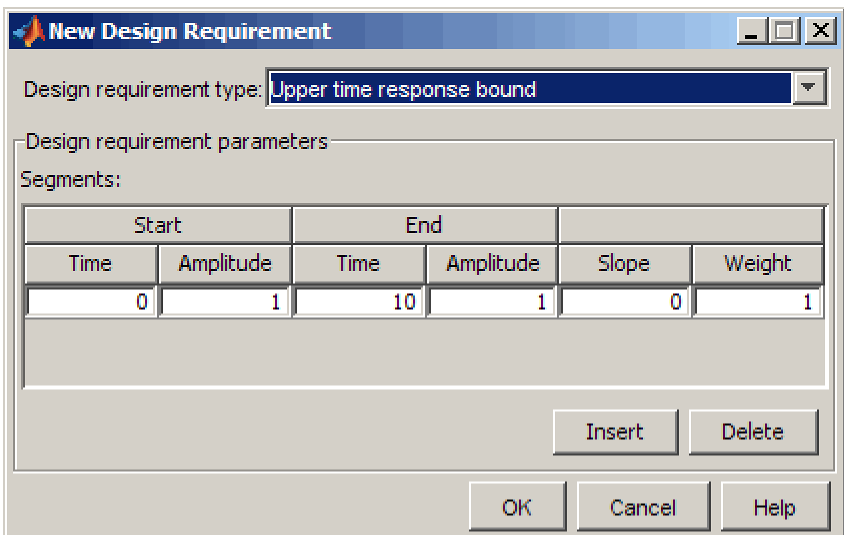
   a In the LTI Viewer, right-click the white area on the bottom plot, and select **Design requirement** > **New**.



   This action opens the New Design Requirement dialog box.



b In the New Design Requirement dialog box, select `Upper time response bound` from the **Design requirement type** drop-down list.

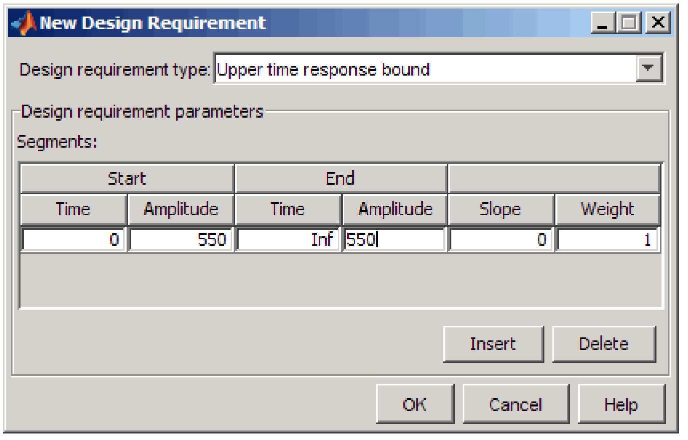


c In the **Time** field of the **End** column, enter `Inf`.

d In the **Amplitude** field of the **Start** column, enter `550`.

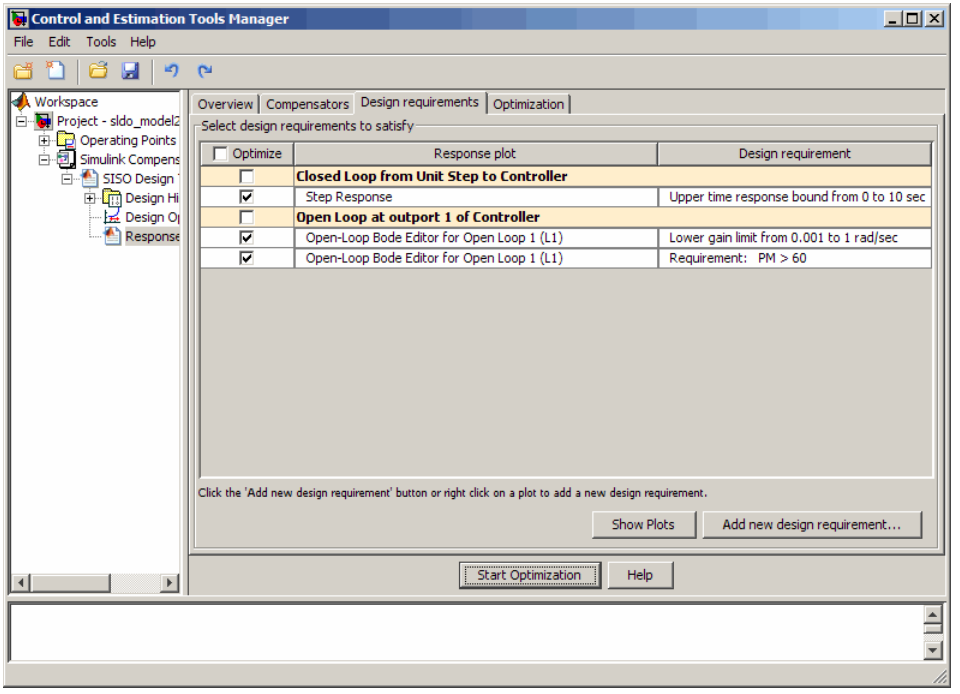
e In the **Amplitude** field of the **End** column, enter `550`.

The New Design Requirements dialog box resembles the following figure.

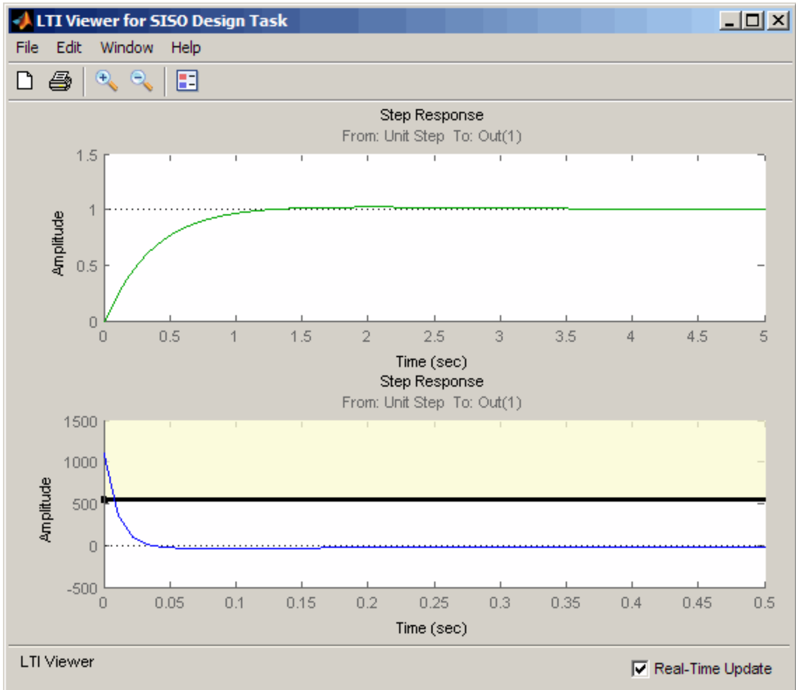


f Click **OK** to close the dialog box.

The **Design requirements** tab in the Control and Estimation Tools Manager GUI updates. It now displays the upper-bound requirement, as shown in the next figure.



The LTI Viewer also updates to show the design requirement, as shown in the next figure.



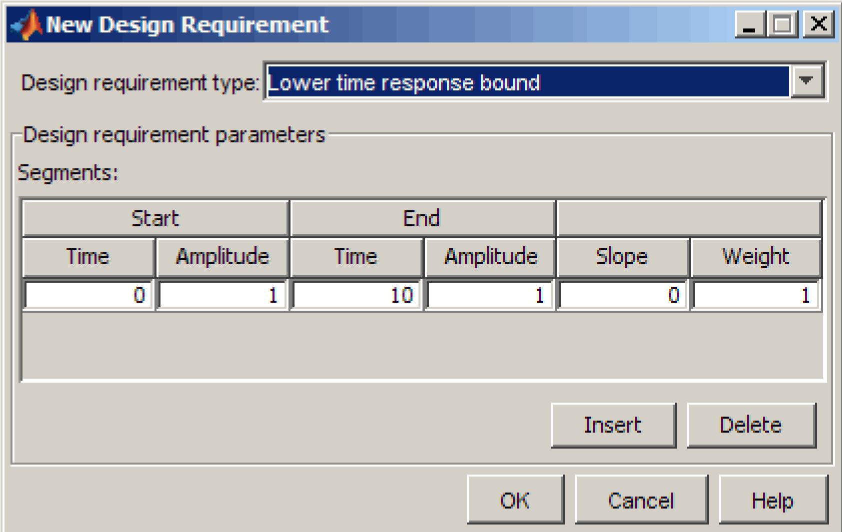
2 Add the lower-bound on the controller's output:

a Right-click the white area in the bottom plot in the LTI Viewer, and select **Design requirement** > **New**.

This action opens the New Design Requirement dialog box.

b Select `Lower time response bound` from the **Design requirement type** drop-down list.

The New Design Requirement dialog box updates to show the lower bound, as shown in the next figure.

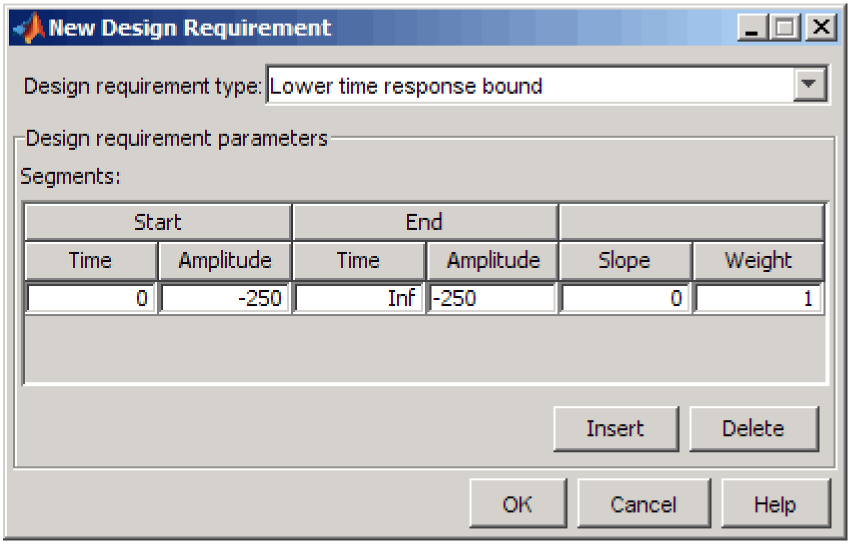


c In the **Time** field of the **End** column, enter `Inf`.

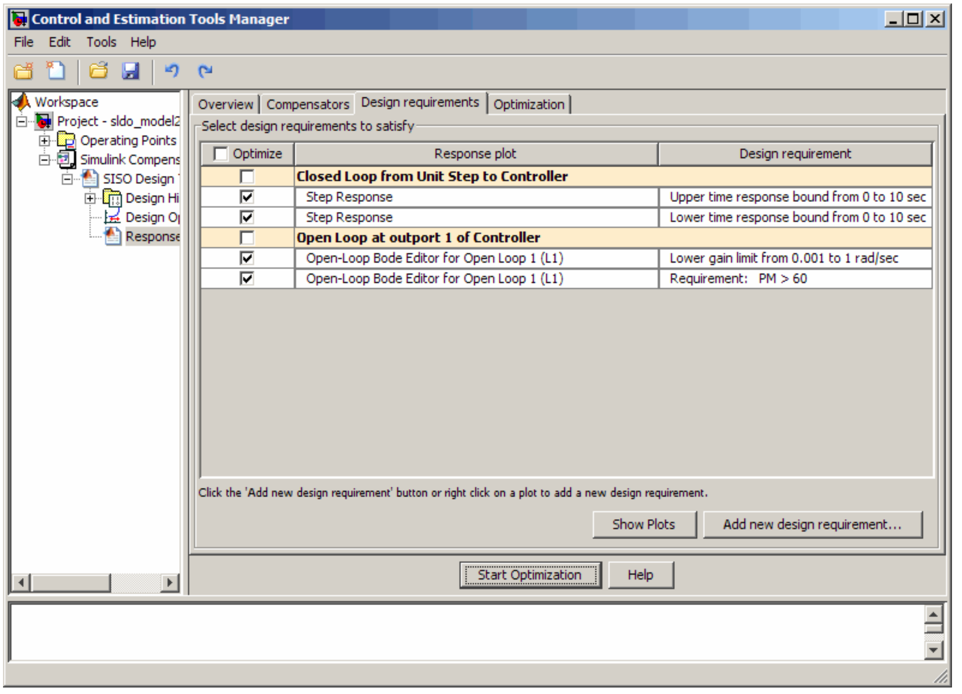
d In the **Amplitude** field of the **Start** column, enter `-250`.

e In the **Amplitude** field of the **End** column, enter `-250`.

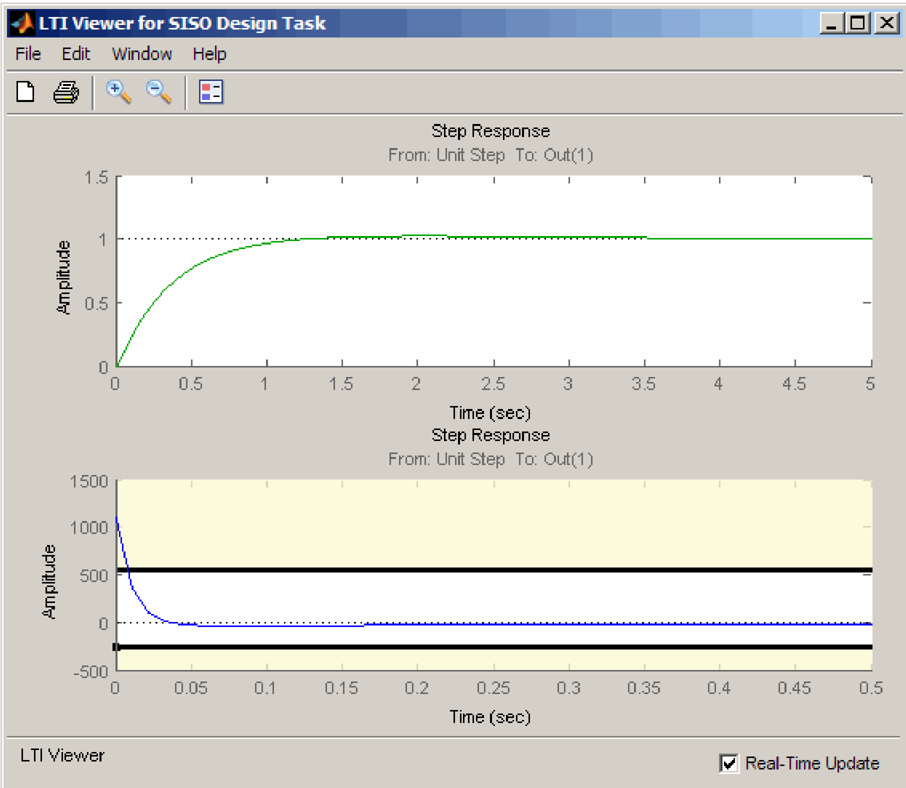
The New Design Requirement dialog box resembles the next figure. Click **OK** to close the dialog box.



The **Design requirements** tab in the Control and Estimation Tools Manager GUI updates. It now displays the lower-limit requirement, as shown in the next figure.

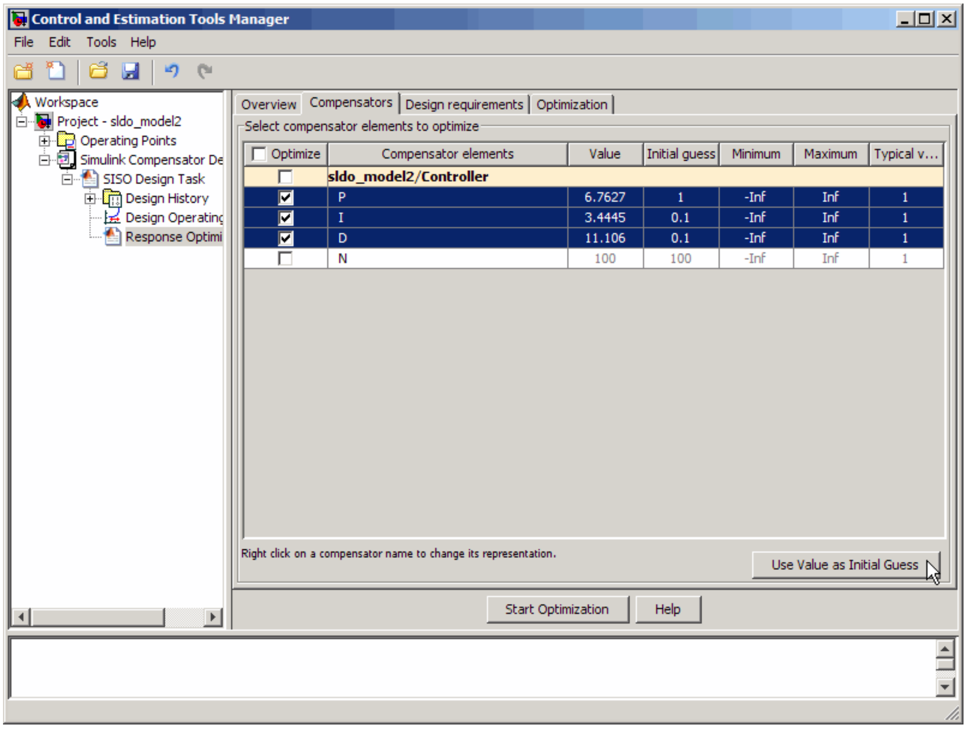


The LTI Viewer also updates to show the lower-bound on the controller's output, as shown in the next figure.



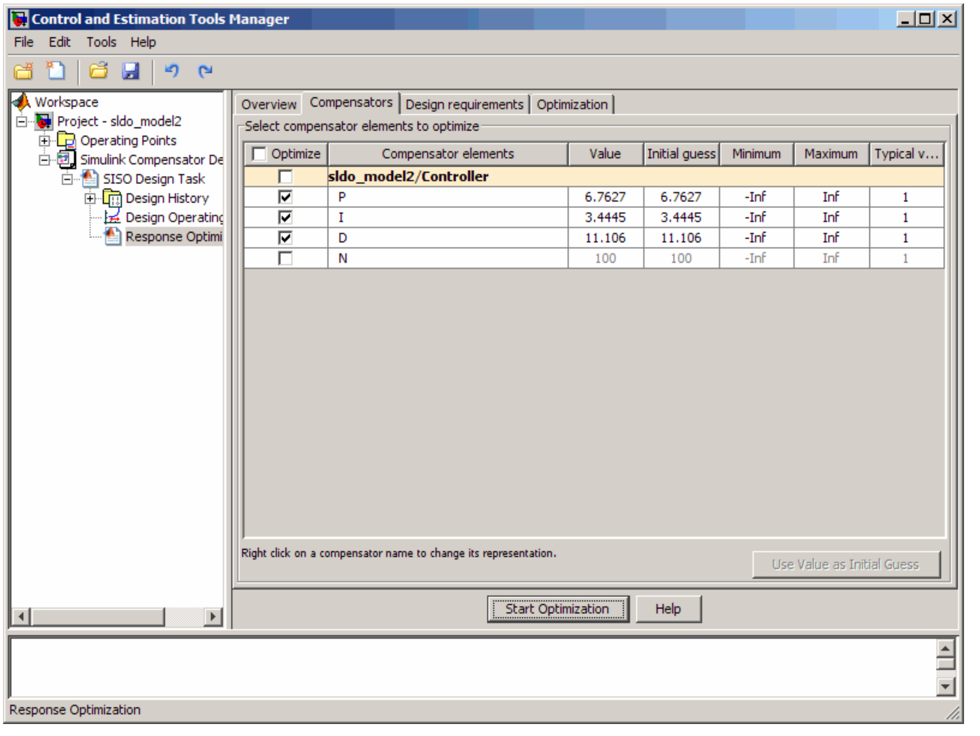
3 Optimize the parameters to meet the design requirements on the controller output:

a In the **Compensators** tab, select the rows containing `P`, `I`, and `D`, and click **Use Value as Initial Guess**.



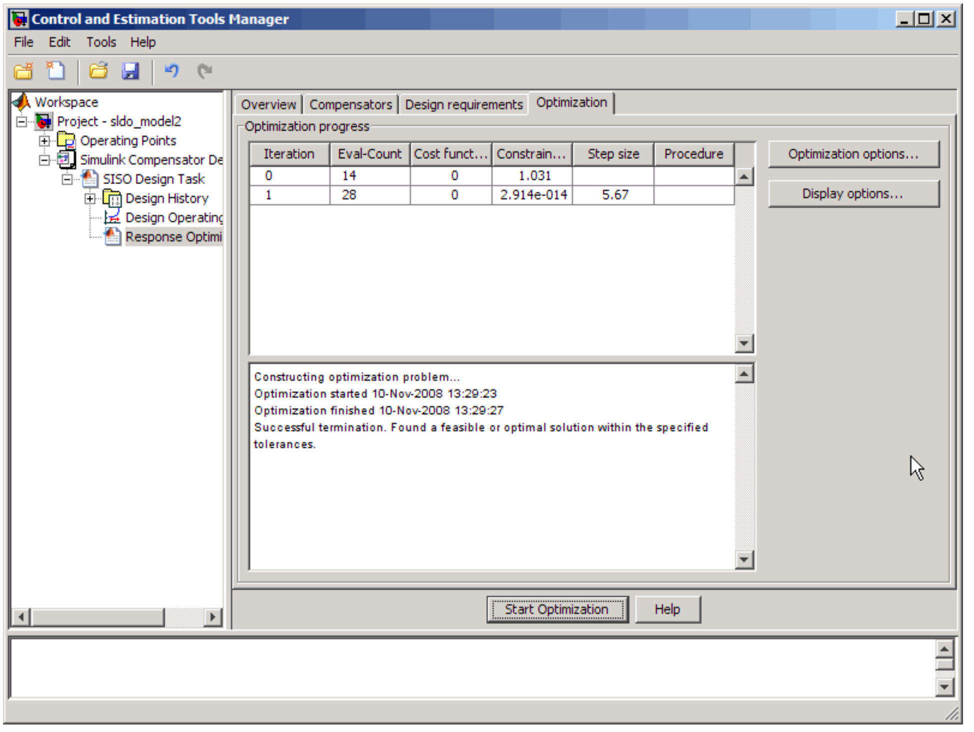
When you run the optimization again, the optimization method uses the updated parameter values as the starting point for refining the values.

Clicking **Use Value as Initial Guess** updates the values in the **Initial Guess** column, as shown in the next figure.



b In the **Optimization** tab, click **Start Optimization**.

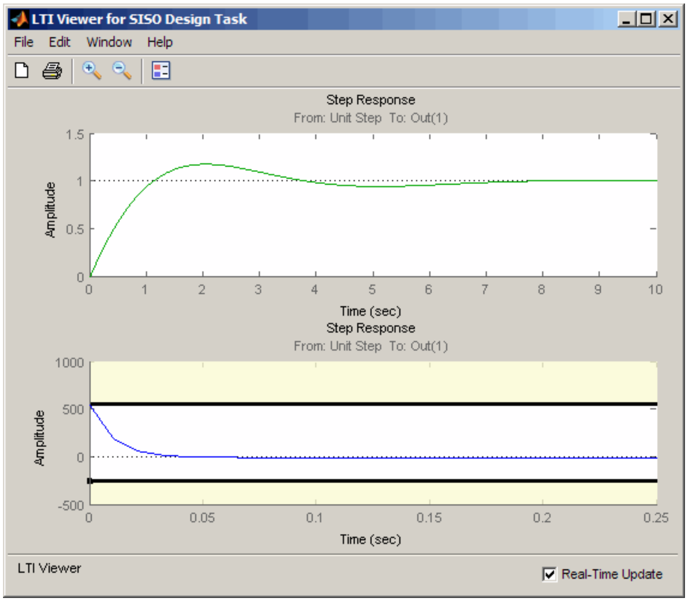
- At every optimization iteration, the optimization method reduces the distance between the current response and the upper and lower bounds on the signal.
- After the optimization completes, the **Optimization** tab resembles the next figure.



4 Examine the parameter values refined controller and the system's response:

a Examine the following response plots:

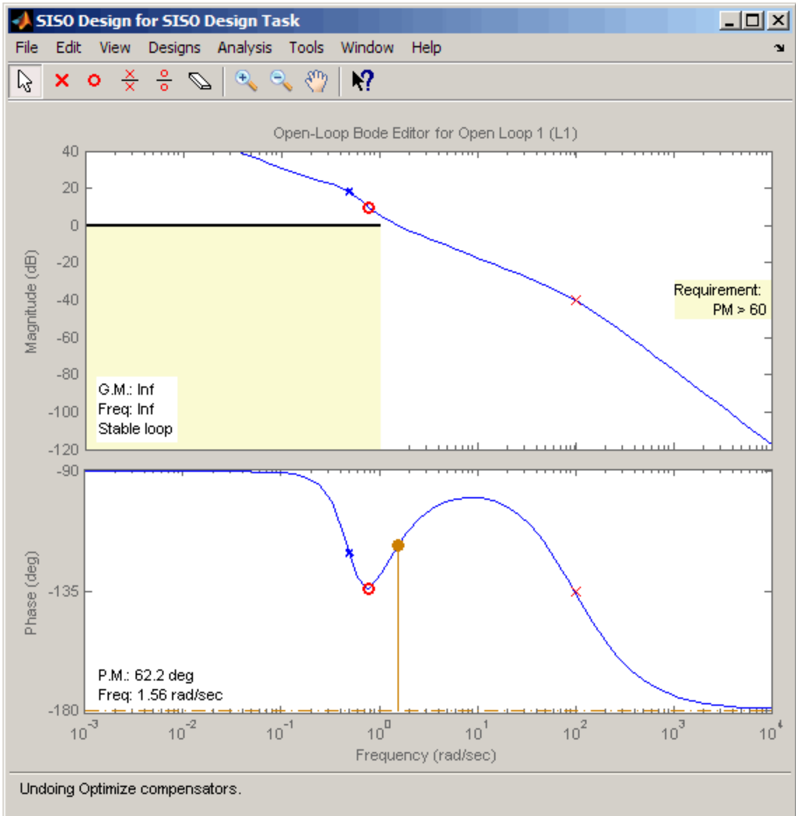
- The LTI viewer



The bottom plot shows that the controller output lies between 550 and -250, and thus meets the design requirement on the controller's output.

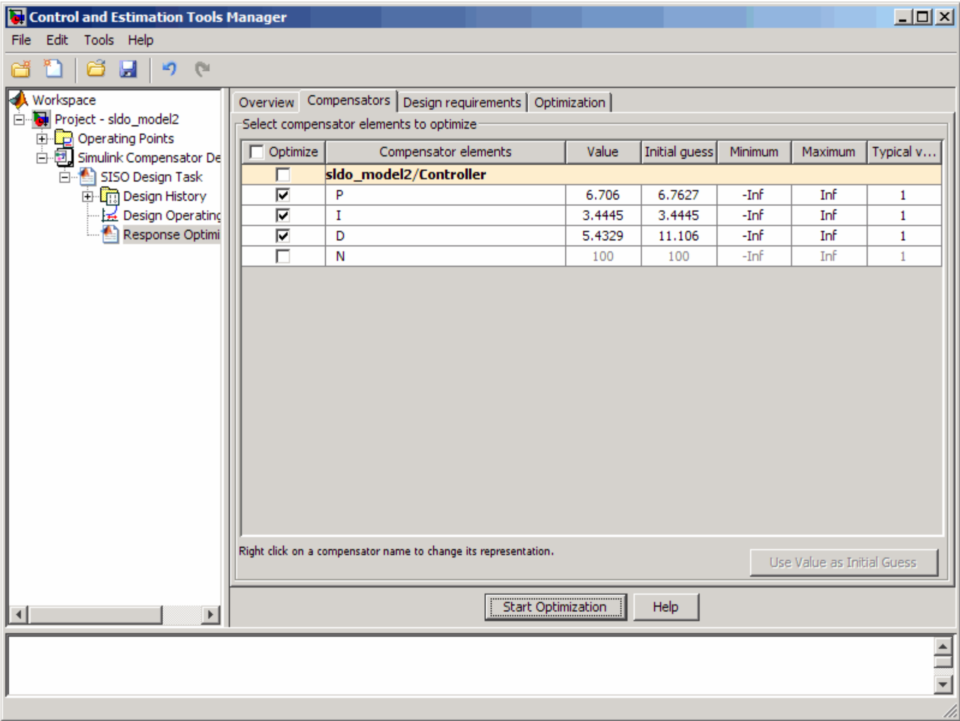
Note: You must also check that the closed-loop response, shown in the top plot, remains stable after refining the controller design.

- The SISO Design window



The plots show that after refining the design, the system continues to meet the magnitude and phase margin requirements specified in “Design Requirements” on page 4-16.

b View the optimized controller parameter values in the **Value** field in the **Compensators** tab.



5 Select the **SISO Design Task** node, and click **Update Simulink Block Parameters**.

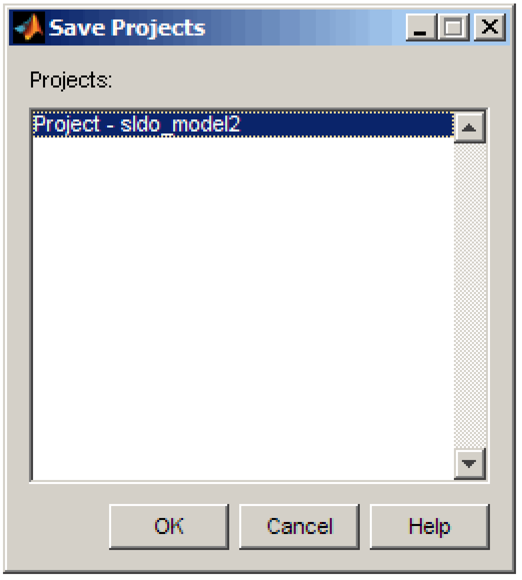
This action writes the optimized controller parameter values to the `Controller` block in the Simulink model.

Saving the Project

To save a project with the optimized controller parameters:

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

This action opens the Save Projects dialog box.



2 In the Save Projects dialog box, select **Project - sldo_model2**, and click **OK** to open the Save Projects window .

3 In the Save Projects window, enter `sldo_model2_optimized.mat` in the **File name** field, and click **Save**.

The action saves the project as a MAT-file.

Tip You can reload this project by typing `sisotool('sldo_model2_optimized')` at the MATLAB prompt.
