

Simulink® Design Optimization™

Getting Started Guide



MATLAB® & SIMULINK®

R2023a



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.
1 Apple Hill Drive
Natick, MA 01760-2098

Simulink® Design Optimization™ Getting Started Guide

© COPYRIGHT 1993–2023 by The MathWorks, Inc.

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

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

Trademarks

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

Patents

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

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)
March 2015	Online only	Revised for Version 2.7 (Release 2015a)
September 2015	Online only	Revised for Version 2.8 (Release 2015b)
March 2016	Online only	Revised for Version 3.0 (Release 2016a)
September 2016	Online only	Revised for Version 3.1 (Release 2016b)
March 2017	Online only	Revised for Version 3.2 (Release 2017a)
September 2017	Online only	Revised for Version 3.3 (Release 2017b)
March 2018	Online only	Revised for Version 3.4 (Release 2018a)
September 2018	Online only	Revised for Version 3.5 (Release 2018b)
March 2019	Online only	Revised for Version 3.6 (Release 2019a)
September 2019	Online only	Revised for Version 3.7 (Release 2019b)
March 2020	Online only	Revised for Version 3.8 (Release 2020a)
September 2020	Online only	Revised for Version 3.9 (Release 2020b)
March 2021	Online only	Revised for Version 3.9.1 (Release 2021a)
September 2021	Online only	Revised for Version 3.10 (Release 2021b)
March 2022	Online only	Revised for Version 3.11 (Release 2022a)
September 2022	Online only	Revised for Version 3.12 (Release 2022b)
March 2023	Online only	Revised for Version 3.13 (Release 2023a)

1	Product Overview
	<hr/>
	Simulink Design Optimization Product Description 1-2
	Key Features 1-2
	Optimization Support for Simulink Models Using Third-Party Applications 1-3
	Ways to Speed Up Design Optimization Tasks 1-4
	Speed Up Using Parallel Computing 1-4
	Speed Up Using Fast Restart Mode 1-4
	Speed Up Using Accelerator Mode 1-5
	Required and Related Products 1-6

2	Parameter Estimation
	<hr/>
	Supported Data 2-2
	Prepare Data for Parameter Estimation 2-3
	About This Tutorial 2-3
	Engine Throttle Model 2-3
	Start a Parameter Estimator Session 2-4
	Create an Experiment for Parameter Estimation 2-4
	Import Data 2-5
	Analyze Data 2-5
	Extract Data for Estimation 2-6
	Replacing Outliers 2-7
	Filtering Data 2-8
	Saving the Session 2-9
	Estimate Parameters from Measured Data 2-10
	About This Tutorial 2-10
	Estimate Model Parameters Using Default Estimation Settings 2-12
	Improve Estimation Results Using Parameter Bounds 2-18
	Validate Estimated Model Parameters 2-20

3

Supported Design Requirements	3-2
Design Optimization to Meet Step Response Requirements (GUI)	3-3
Design Optimization to Meet Step Response Requirements (Code)	3-15
Design Optimization to Track Reference Signal (GUI)	3-21
Model Structure	3-21
Design Requirements	3-21
Specify Reference Signal	3-21
Specify Design Variables	3-26
Optimize Model Response	3-28
Design Optimization Using Frequency-Domain Check Blocks (GUI) ...	3-32
Full Wave Rectifier Model	3-32
Model Structure	3-32
Design Requirements	3-32
Specify Design Requirements	3-33
Specify Design Variables	3-34
Optimize Design	3-36
Time-Domain Model Verification	3-42

Optimization-Based Linear Control Design

4

When to Use Optimization-Based Linear Control Design	4-2
Types of Time- and Frequency-Domain Design Requirements for Optimization-Based Control Design	4-3
Design Optimization-Based PID Controller for Linearized Simulink Model (GUI)	4-4
Model Structure	4-4
Design Requirements	4-4
PID and Plant Model	4-4
Configure the Control System Designer App for Optimization-Based Control Design	4-5
Design an Initial PID Controller to Meet Bode Magnitude and Phase Margins Requirements	4-6
Refine the Controller Design to Meet Controller Output Bounds	4-16

Product Overview

- “Simulink Design Optimization Product Description” on page 1-2
- “Optimization Support for Simulink Models Using Third-Party Applications” on page 1-3
- “Ways to Speed Up Design Optimization Tasks” on page 1-4
- “Required and Related Products” on page 1-6

Simulink Design Optimization Product Description

Analyze model sensitivity and tune model parameters

Simulink Design Optimization provides functions, interactive tools, and blocks for analyzing and tuning model parameters. You can determine the model's sensitivity, fit the model to test data, and tune it to meet requirements. Using techniques like Monte Carlo simulation and Design of Experiments, you can explore your design space and calculate parameter influence on model behavior.

Simulink Design Optimization helps you increase model accuracy. You can preprocess test data, automatically estimate model parameters such as friction and aerodynamic coefficients, and validate the estimation results.

To improve system design characteristics such as response time, bandwidth, and energy consumption, you can jointly optimize physical plant parameters and algorithmic or controller gains. These parameters can be tuned to meet time-domain and frequency-domain requirements, such as overshoot and phase margin, and custom requirements.

Key Features

- Parameter estimation from test data
- Parameter tuning to meet time-domain, frequency-domain, and custom requirements
- Design exploration and sensitivity analysis
- Graphical requirement specification and optimization progress monitoring
- 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.

See Also

More About

- "Choosing MEX Applications"

Ways to Speed Up Design Optimization Tasks

You can use the following ways to speed up parameter estimation, response optimization, and sensitivity analysis tasks:

- Parallel computing
- Fast restart mode
- Accelerator mode

You can use a combination of these, but depending on the limitations associated with each, you may or may not see an increase in speed. For example, you can use parallel computing and fast restart together to speed up optimization. However, you do not see an increase in speed compared to using only parallel computing if the compilation phase of your model is short.

Speed Up Using Parallel Computing

You can use Parallel Computing Toolbox™ software to speed up parameter estimation, response optimization, and sensitivity analysis. 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 many parameters to optimize, and you use the `GradientDescent` or `NonLinearLeastSquares` method.
- The `PatternSearch` method is selected as the optimization method.
- The model contains many uncertain parameters and uncertain parameter values.
- The model is complex and takes a long time to simulate.

You can use parallel computing in the **Parameter Estimator**, **Response Optimizer**, and **Sensitivity Analyzer** apps, or at the command line. For more information, see “Use Parallel Computing for Parameter Estimation”, “Use Parallel Computing for Response Optimization”, and “Use Parallel Computing for Sensitivity Analysis”.

Speed Up Using Fast Restart Mode

You can use the fast restart feature of Simulink to speed up design optimization of tunable parameters of a model.

Fast restart enables you to perform iterative simulations without compiling a model or terminating the simulation each time. Using fast restart, you compile a model only once. You can then tune parameters and simulate the model again without spending time on compiling. Fast restart associates multiple simulation phases with a single compile phase to make iterative simulations more efficient. You see a speedup of design optimization tasks using fast restart in models that have a long compilation phase. See “How Fast Restart Improves Iterative Simulations”.

When you enable fast restart, you can only change tunable properties of the model during simulation. For more information about the limitations, see “Limitations”.

You can configure fast restart in the **Parameter Estimator**, **Response Optimizer**, and **Sensitivity Analyzer** apps, or at the command line. For more information see, “Improving Optimization

Performance Using Fast Restart (GUI)", "Improving Optimization Performance Using Fast Restart (Code)", "Use Fast Restart Mode During Response Optimization", or "Use Fast Restart Mode During Sensitivity Analysis".

Speed Up Using Accelerator Mode

Simulink Design Optimization software supports `Normal` and `Accelerator` simulation modes. You can accelerate the design optimization computations by changing the simulation mode of your Simulink model to `Accelerator`. For information about these modes, see "How Acceleration Modes Work".

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

In the `Accelerator` mode, Simulink Design Optimization software runs simulations during optimization with compiled C code. Using compiled C code speeds up the simulations and reduces the time to optimize the model response signals.

For information about the limitations, and how to use the `Accelerator` mode, see "Use Accelerator Mode During Simulations".

See Also

Related Examples

- "Optimizing Time-Domain Response of Simulink Models Using Parallel Computing"
- "Improving Optimization Performance Using Fast Restart (GUI)"
- "Improving Optimization Performance Using Fast Restart (Code)"

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 <https://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.
Deep Learning 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.
Statistics and Machine Learning Toolbox	Provides additional probability distributions and statistical analysis techniques for sensitivity analysis.
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.

Parameter Estimation

- “Supported Data” on page 2-2
- “Prepare Data for Parameter Estimation” on page 2-3
- “Estimate Parameters from Measured Data” on page 2-10

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 for Parameter Estimation”.
- *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 2-10.

See Also

More About

- “Time Series Objects and Collections”
- “Complex Data”

Prepare Data for Parameter Estimation

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 Estimator**:

- 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.

Engine Throttle Model

Open the engine throttle system model.

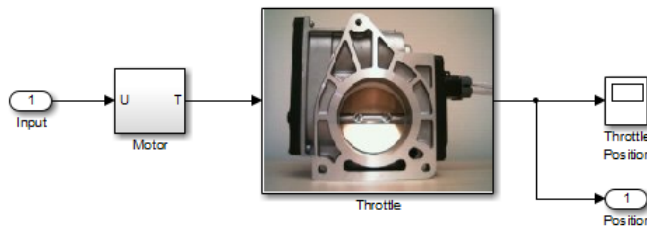
```
open_system('spe_engine_throttle1.slx')
```

Start a Parameter Estimator Session

To perform parameter estimation, you must first start a **Parameter Estimator** session.

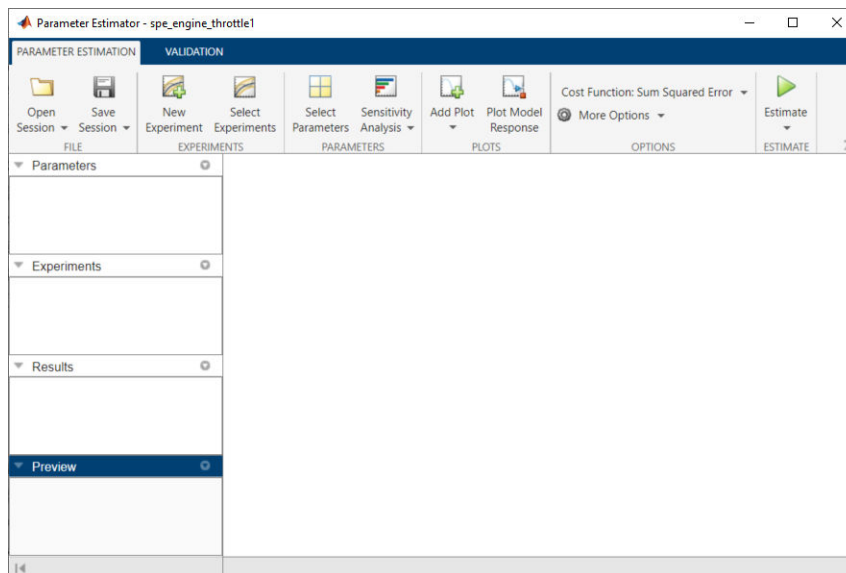
- 1 When you open the Simulink model, the model loads the data into the MATLAB workspace.

Engine Throttle Model



- 2 In the Simulink model window, on the **Apps** tab, under **Control Systems**, select **Parameter Estimator**.

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



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

Create an Experiment for Parameter Estimation

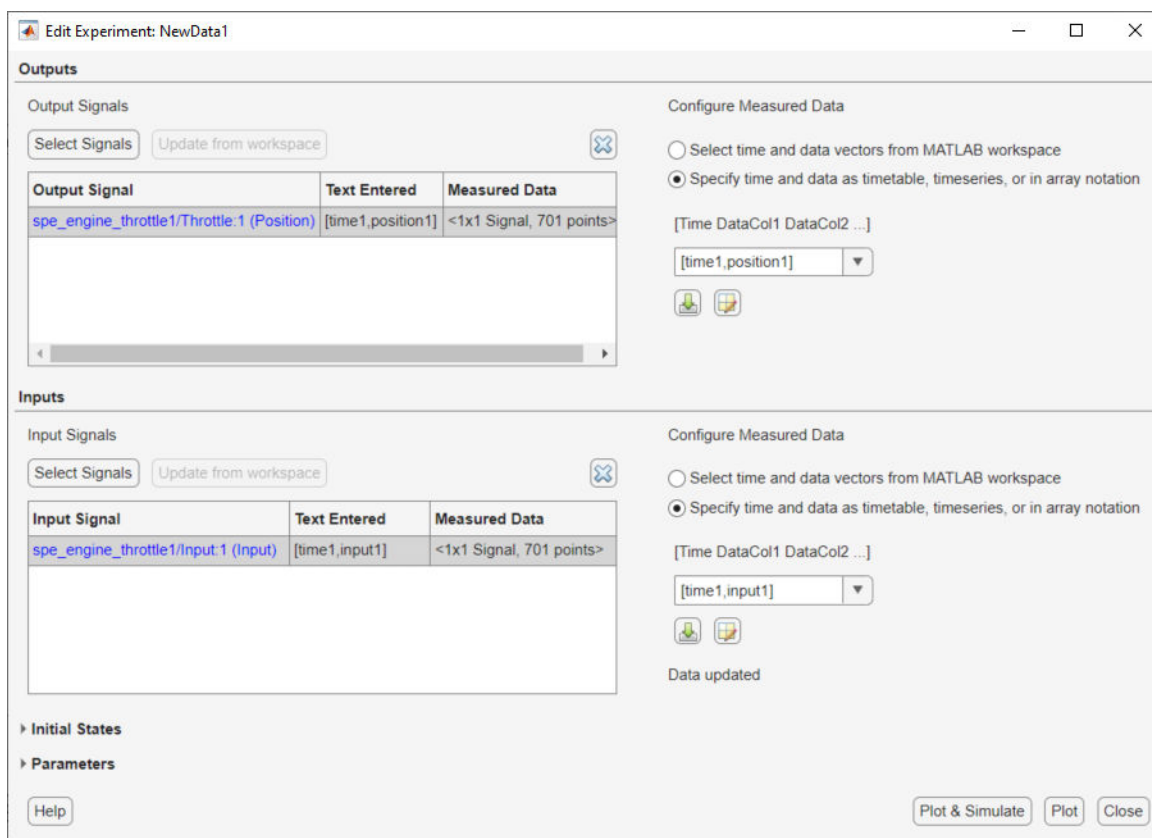
In the **Parameter Estimator** on the **Parameter Estimation** tab, click the **New Experiment** button. This will create an experiment with the name **Exp** in the **Experiments** list on the left pane. You can rename it by right-clicking and selecting **Rename** from the list. For example, call it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

Import Data

This portion of the tutorial explains how to import measured I/O data into the experiment in the **Parameter Estimator**. 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. To learn more about the blocks, see the **Inport** and **Outport** block reference pages in the Simulink documentation.

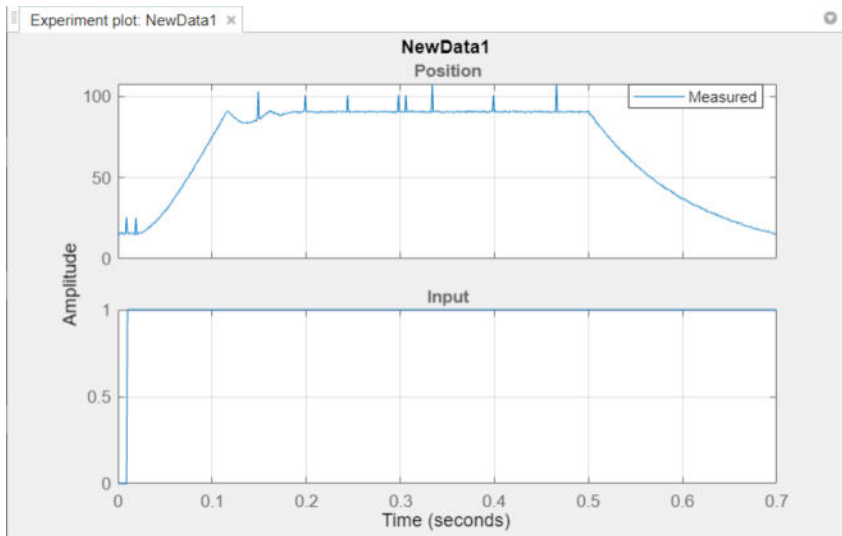
To import data into the experiment, right-click and select **Edit...** to launch the experiment editor. Import the output data by typing `[time1, position1]` in the dialog box in the **Outputs** panel. Import the input data by typing `[time1, input1]` in the dialog box in the **Inputs** panel.



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 2-3. 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 2-6 .

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 2-7 and “Filtering Data” on page 2-8.

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

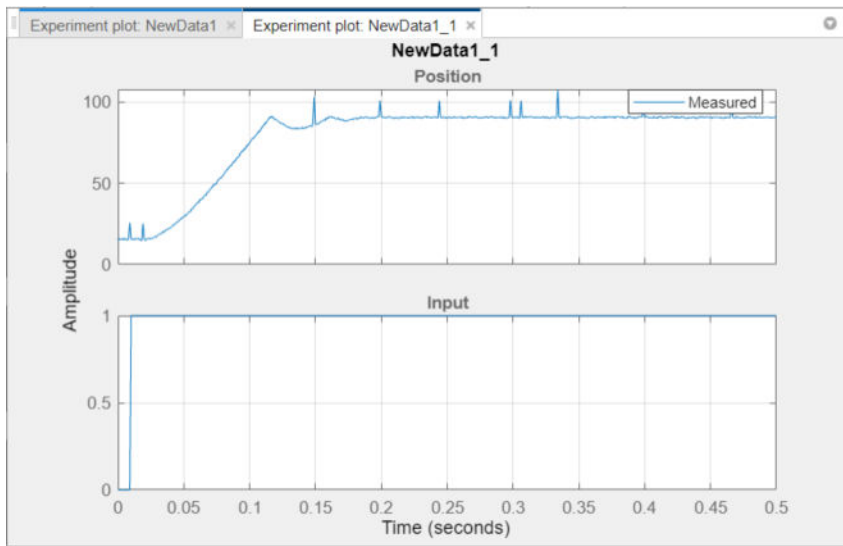
This portion of the tutorial explains how to select a subset of I/O data for estimation. As described in “Analyze Data” on page 2-5, the system is shut down at $t = 0.5$ s. To focus the estimation on the time period before $t = 0.5$ s, exclude the data samples beyond $t = 0.5$ s. This operation selects 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 2-5.

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 2-5, 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 Estimator** 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

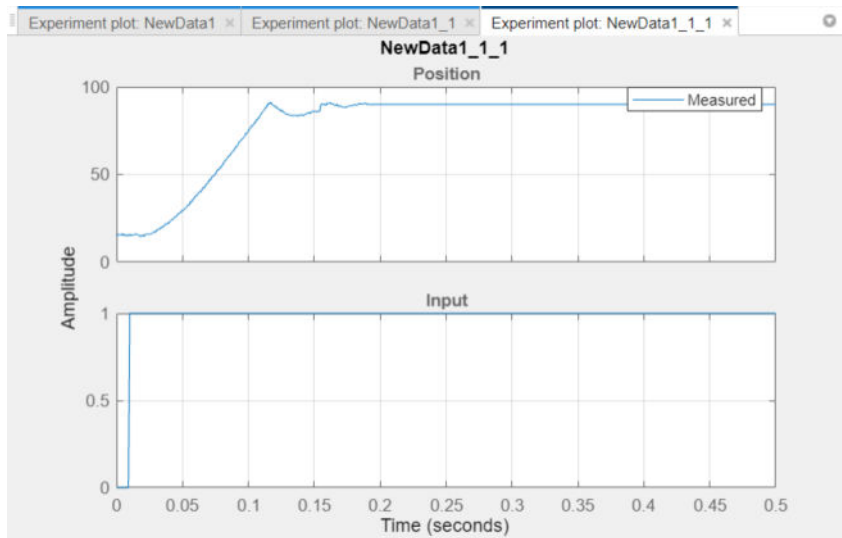
Why Replace Outliers

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 2-6, 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. 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 **Apply** section and select **Save As: Create a new experiment from the modified data. Parameter Estimator** 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.

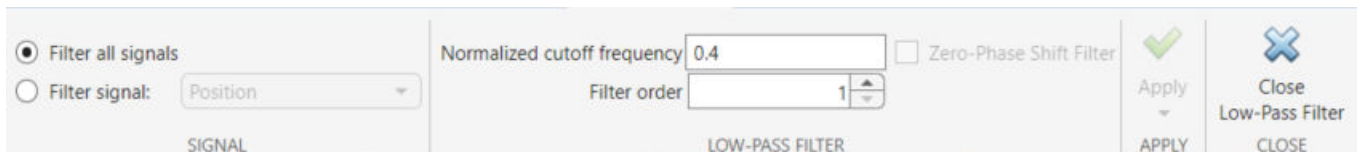


Filtering Data

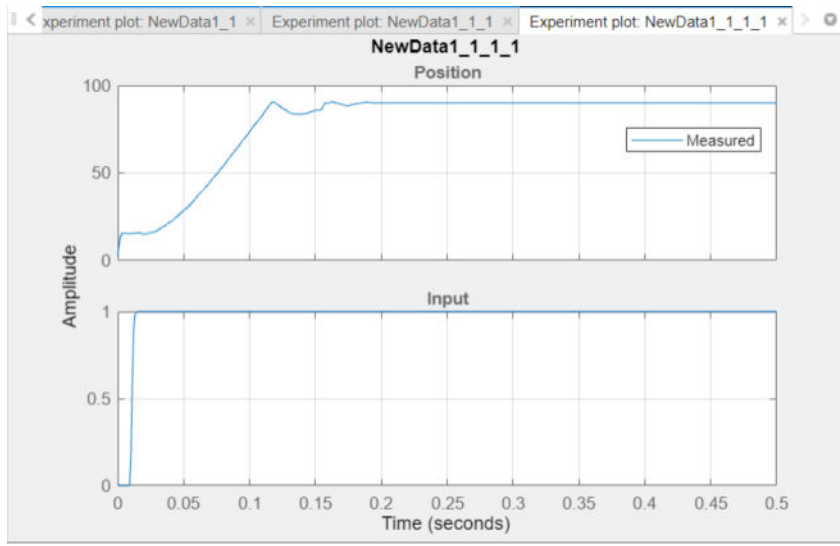
This portion of the tutorial explains how to filter the noise and remove any periodic trends from the output data. First remove the outliers from the output data, as described in “Replacing Outliers” on page 2-7.

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.



- 4 Click the arrow in the **Apply** section and select **Save As: Create a new experiment from the modified data. Parameter Estimator** 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_sdoession.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 2-10.

Estimate Parameters from Measured Data

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 Estimator**:

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

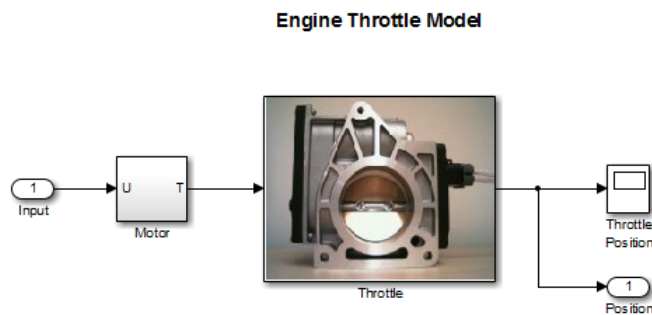
Engine Throttle Model

Open the engine throttle system model.

```
open_system('spe_engine_throttle1.slx')
```

About the Model

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

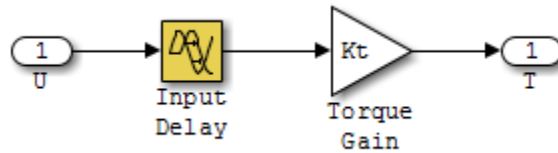


The throttle system controls the flow of air and fuel mixture to the engine cylinders. The throttle body contains a butterfly valve that opens when a driver presses the accelerator pedal. Opening this valve increases the amount of fuel mixture entering the cylinders, which increases the engine speed. 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 2-11 and “Throttle Subsystem” on page 2-11.

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

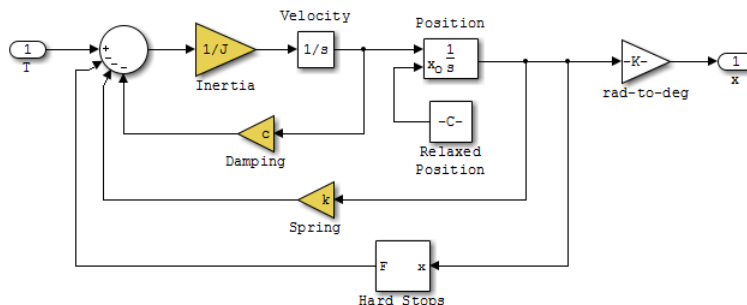
The Motor subsystem contains the DC motor model. To open the model, double-click the corresponding block.



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. t_d is the input time delay of the motor, represented by <code>input_delay</code> 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	<p>T is the torque applied by the DC motor.</p> <p>θ is the angular position of the valve, represented by x in the model.</p> <p>$T_{hardstop}$ is the torque applied by the hard stop.</p>
Parameters	<p>J is the valve inertia.</p> <p>c is the valve viscous friction.</p> <p>k is the valve spring constant.</p>
States	<p>θ is the angular position.</p> <p>$\dot{\theta}$ is the angular velocity.</p>
Equations	<p>The mathematical system for the butterfly valve is described in the following equation:</p> $J\ddot{\theta} + c\dot{\theta} + k\theta = T + T_{hardstop}$ <p>where $15^\circ \leq \theta \leq 90^\circ$, with initial conditions $\theta_0 = 15^\circ$, and $\dot{\theta}_0 = 0$.</p> <p>The torque applied by the Hard Stops block is described in the following equation:</p> $T_{hardstop} = \begin{cases} 0, & 15^\circ \leq \theta \leq 90^\circ \\ K(90^\circ - \theta), & \theta > 90^\circ \\ K(15^\circ - \theta), & \theta < 15^\circ \end{cases}$ <p>where K is the gain of the Hard Stops block.</p>
Input	T
Output	θ

Estimate Model Parameters Using Default Estimation Settings

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 **Parameter Estimator** app from the Simulink model.

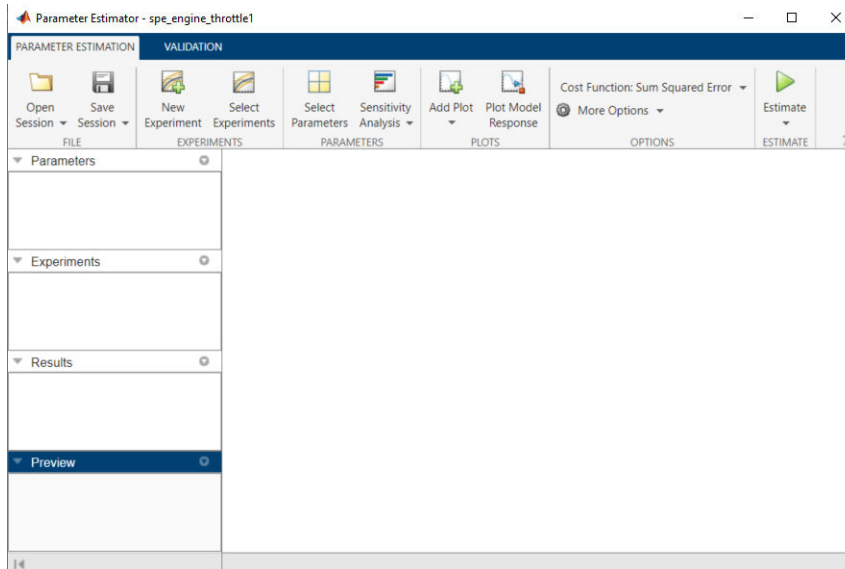
In the Simulink Toolstrip, on the **Apps** tab, under **Control Systems**, select **Parameter Estimator**.

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

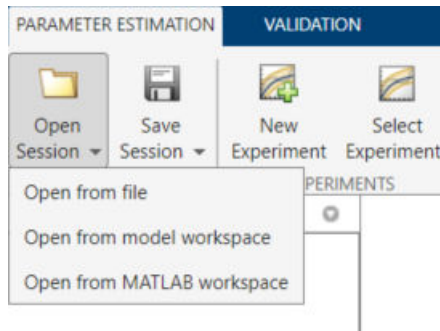
Note The Simulink model must remain open to perform parameter estimation tasks.

Specify Estimation Data and Parameters

- 1 Load or import the estimation data.



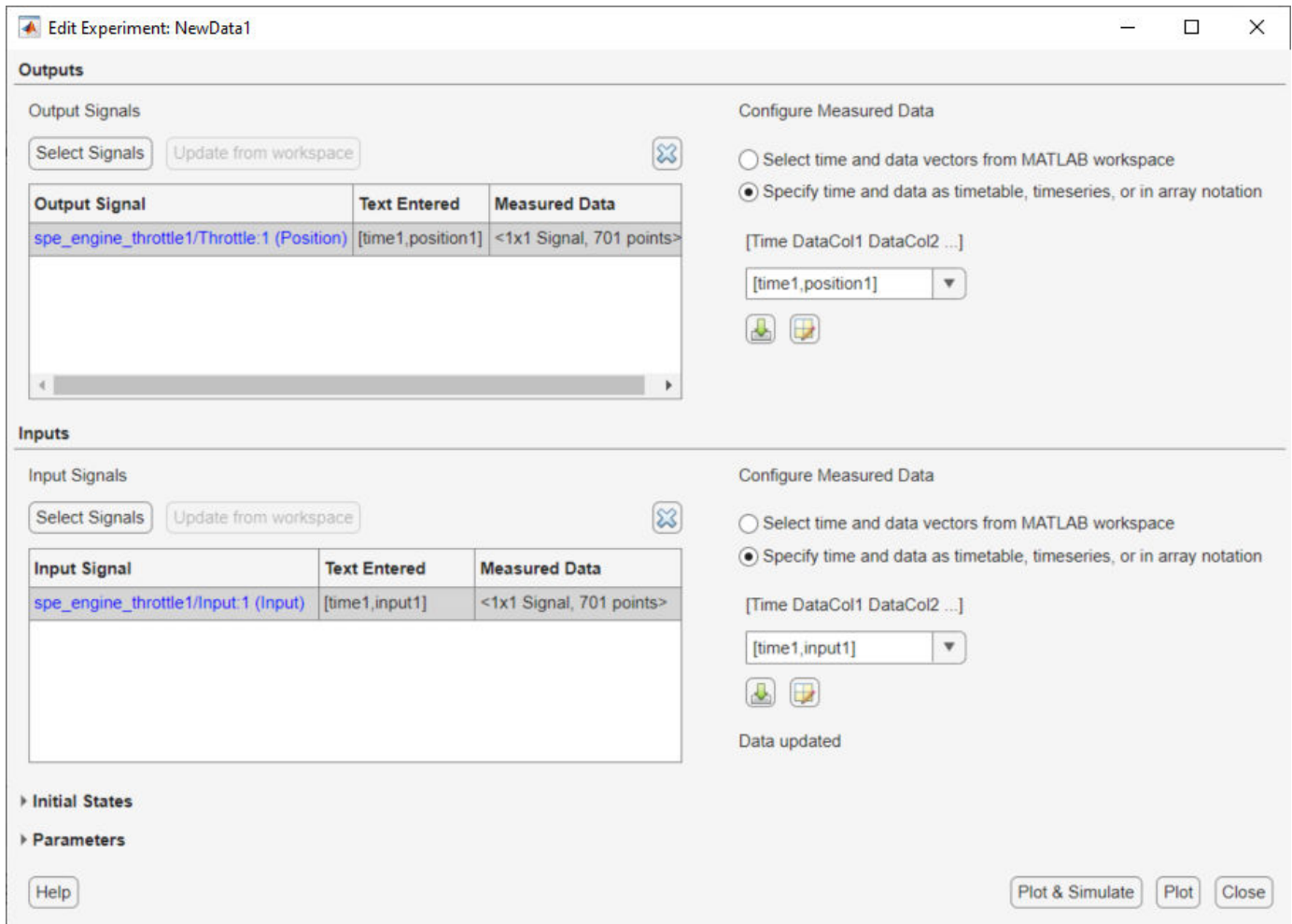
- a If you prepared data and saved your session as described in “Prepare Data for Parameter Estimation” on page 2-3, load the preconfigured session. On the **Parameter Estimation** tab, click the **Open Session** drop down list.



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**.

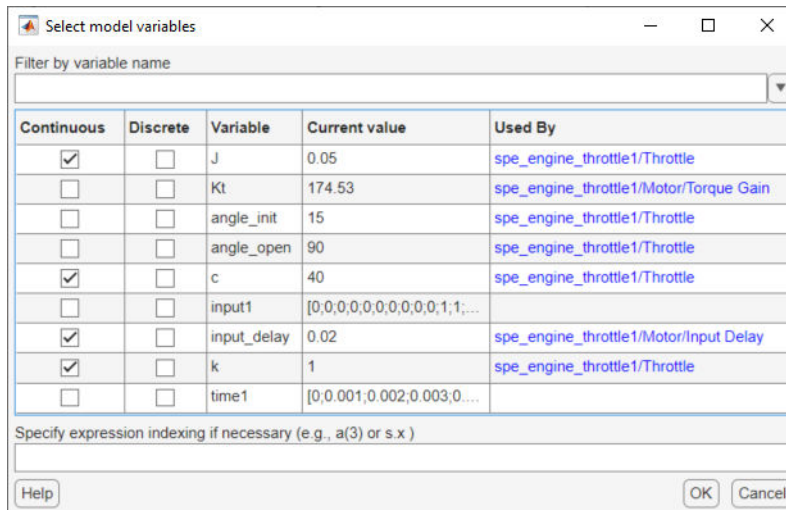
To import data into the experiment, right-click and select **Edit...** to launch the experiment editor. Import the output data by typing in the dialog box in the **Outputs** panel, for example `[time1, position1]`. Import the input data by typing in the dialog box in the **Inputs** panel, for example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



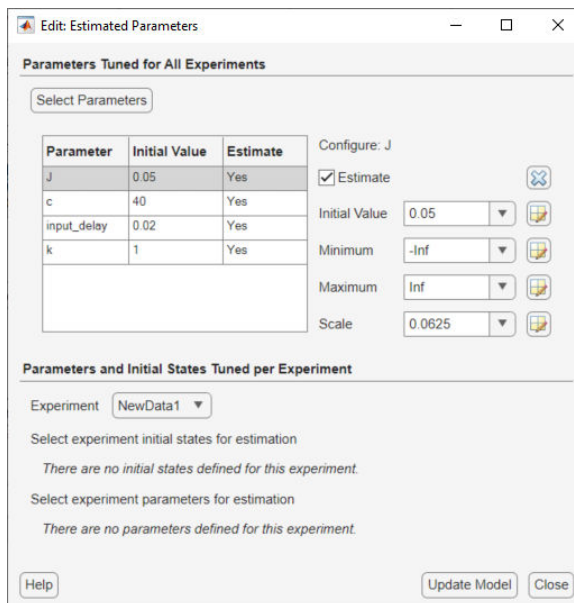
- Specify parameters for estimation. On the **Parameter Estimation** tab, click the **Select Parameters** button to open the **Edit: Estimated Parameters** dialog box. In the **Parameters Tuned for all Experiments** panel, click the **Select parameters** button to launch the **Select Model Variables**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

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

Note In your application, if the model parameters you want to estimate are not listed in the Select Model Variables dialog box, first specify these parameters as variables. See, “Add Model Parameters as Variables for Estimation”.



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

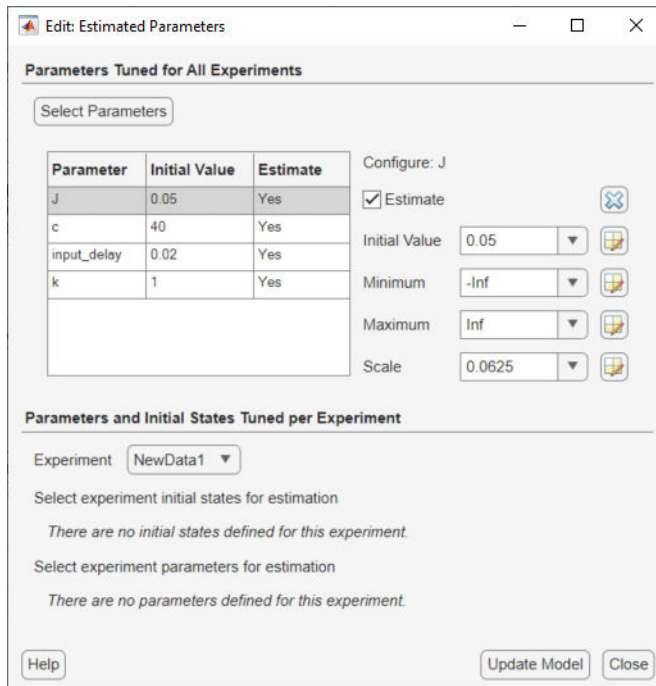


The app selects the parameters you add for estimation by default. When estimating a large number of parameters, you can first select a subset of parameters to estimate.

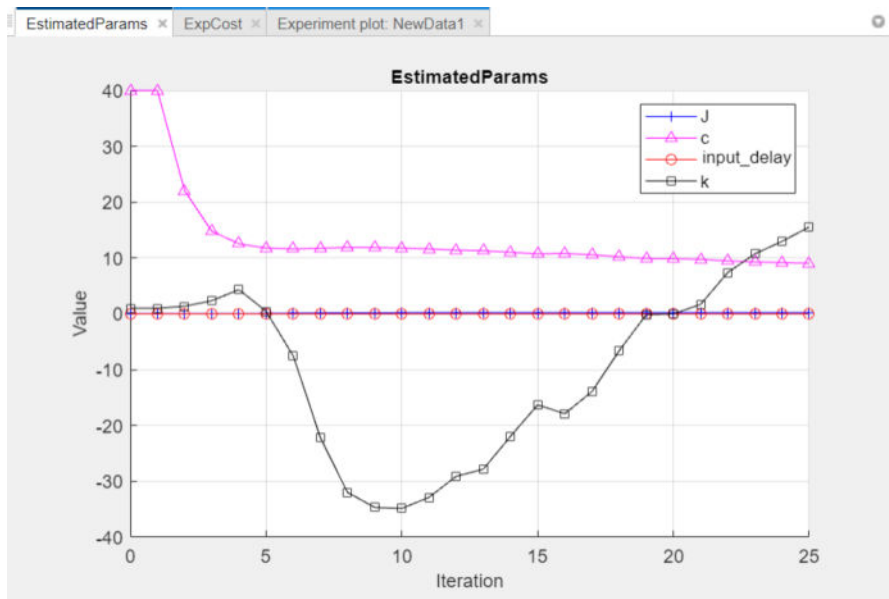
You can also first use sensitivity analysis to identify the parameters that most influence the estimation, and then specify these parameters for estimation. To open the **Sensitivity Analyzer**,

in the **Parameter Estimation** tab, click  **Sensitivity Analysis**. In the **Sensitivity Analyzer**, you can identify the model parameters that most influence the estimation problem and compute initial values for the estimation parameters.

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



- 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.
- 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)
14	135	0.0386
15	144	0.0362
16	153	0.0339
17	162	0.0314
18	171	0.0280
19	180	0.0236
20	189	0.0191
21	198	0.0153
22	207	0.0119
23	216	0.0095
24	225	0.0078
25	234	0.0070

Optimization started 2023-Jan-26, 09:42:10
 Estimation converged, 2023-Jan-26, 09:44:07
 'spe_engine_throttle1' updated with estimated parameter values
 Estimated parameter values written to 'EstimatedParams'
 Estimated experiment values written to 'NewData1'
 Estimation solver output:
 Local minimum possible.
 lsqnonlin stopped because the final change in the sum of squares relative to its initial value is less than the value of the function tolerance.

Buttons: Save Iteration..., Display Options..., Estimate

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

```

Estimation result(s):
J = 0.2277
c = 9.0612
input_delay = 0.0097504
k = 15.557

Parameters estimated using experiments:
NewData1, cost = 0.0069984

Solver output:

Cost: 0.0069984
ExitFlag: 3
FCount: 234
Date: 26-Jan-2023 09:44

Solver termination message:

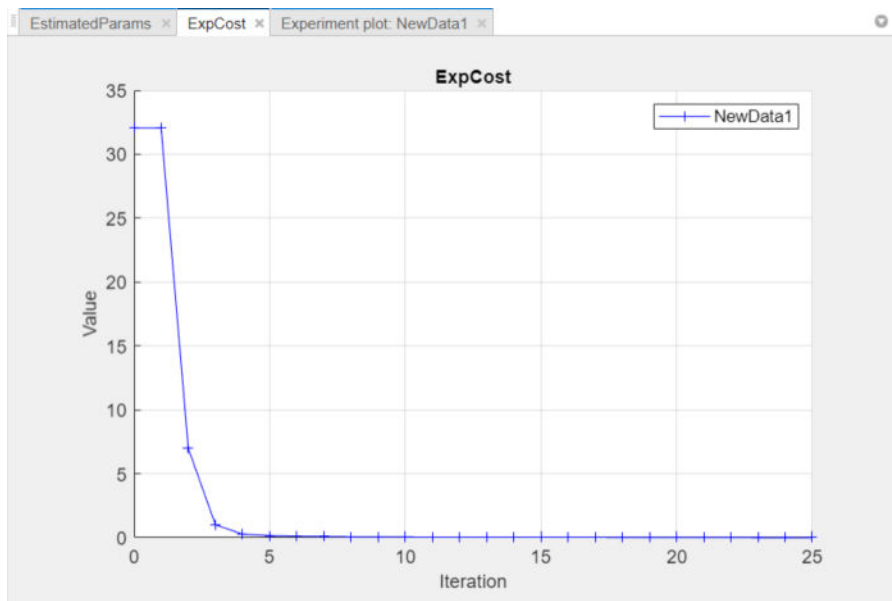
Local minimum possible.

lsqnonlin stopped because the final change in the sum of squares relative to
its initial value is less than the value of the function tolerance.

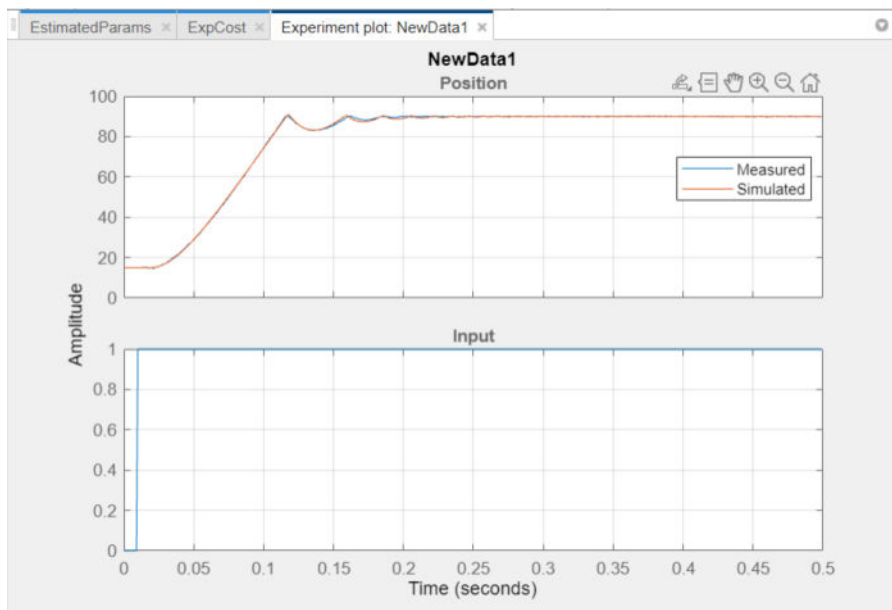
Optimization stopped because the relative sum of squares (r) is changing
by less than options.FunctionTolerance = 1.000000e-03.
  
```

Buttons: Use as initial guess, Update Model, OK

- Examine the estimated cost function graph. Cost function is the error between the simulated and measured output. During estimation, the default optimization method `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 Estimator**, click the **Select Parameters** button to specify bounds on the parameter values. For each parameter, click the right parameter 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.

Parameter	Initial Value	Estimate
J	0.227700325791114	Yes
c	9.06120677932194	Yes
input_delay	0.00975042789683614	Yes
k	15.5565068882597	Yes

Configure: k

Estimate

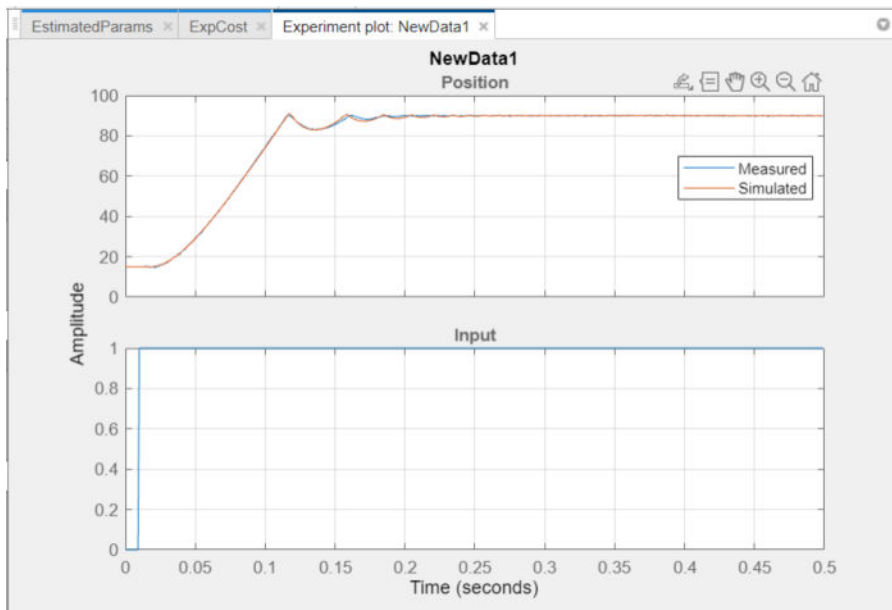
Initial Value: 15.55650...

Minimum: 0

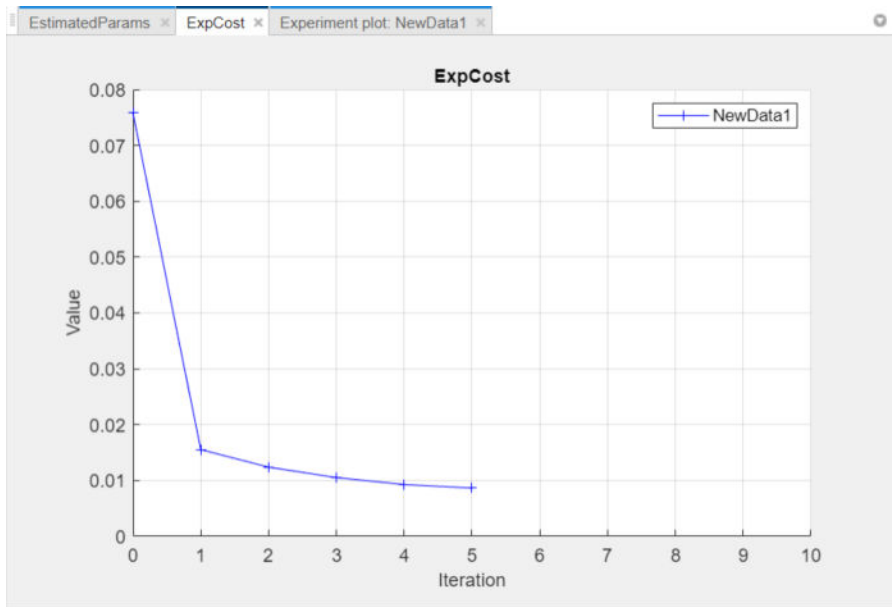
Maximum: 0.1

Scale: 0.03125

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.

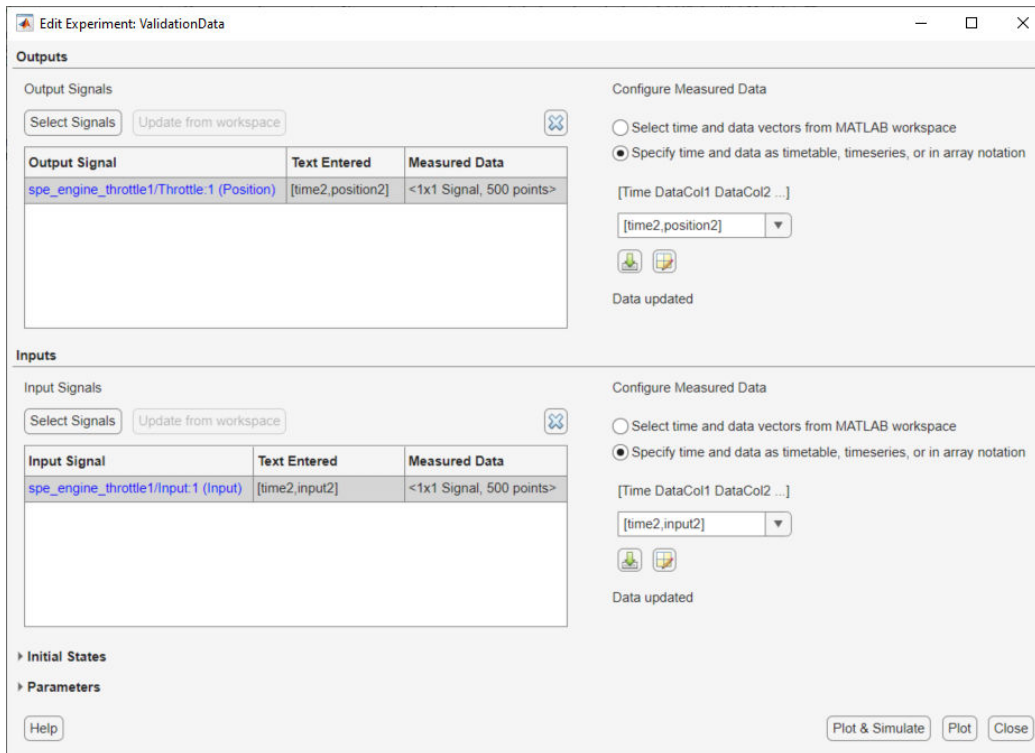


Validate Estimated Model Parameters

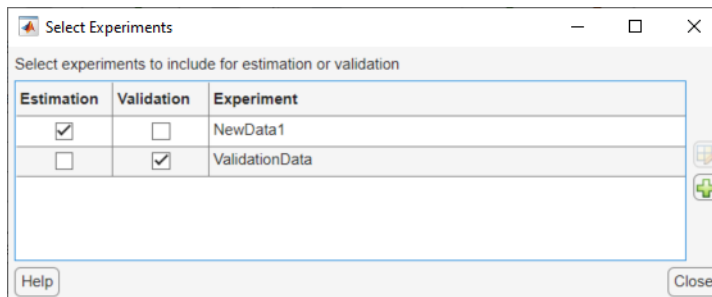
After estimating model parameters, validate the model using another data set (*validation data*). A good match between the simulated response and the validation data indicates that you have not overfitted the model.

To validate the estimated parameters using a validation data set:

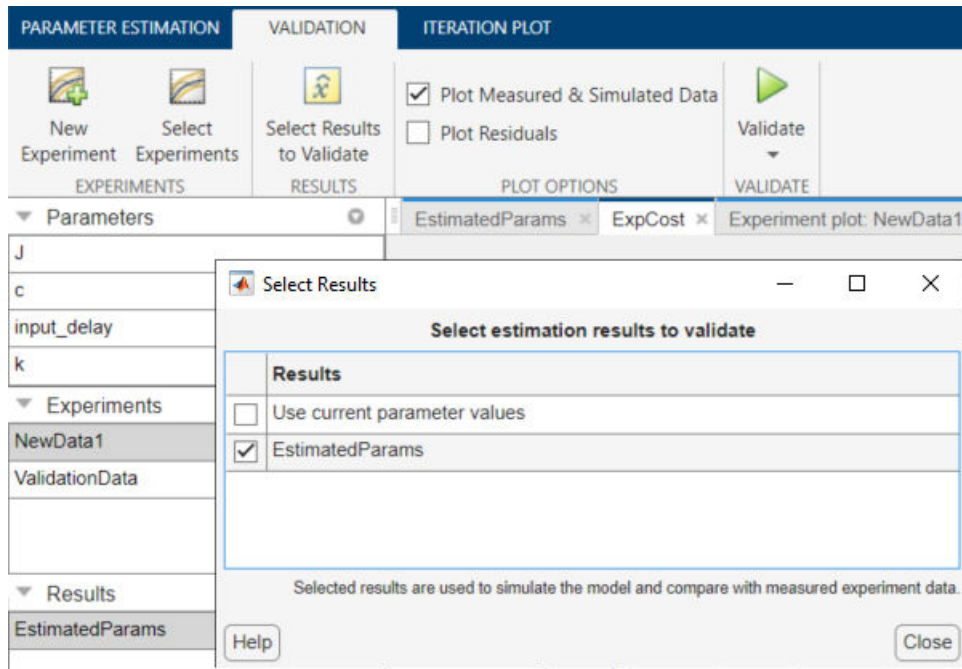
- 1 Create a new experiment to use for validation. Name it `ValidationData`. Import the validation I/O data, `input2` and `position2`, and the time vector, `time2` in the `ValidationData` experiment. To do this, in the **Parameter Estimator**, in the Experiments pane, right-click `ValidationData` and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



- 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.

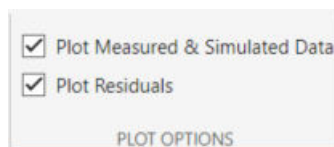


- 3 Select results to use. On the **Validation** tab, click **Select Results to Validate**.



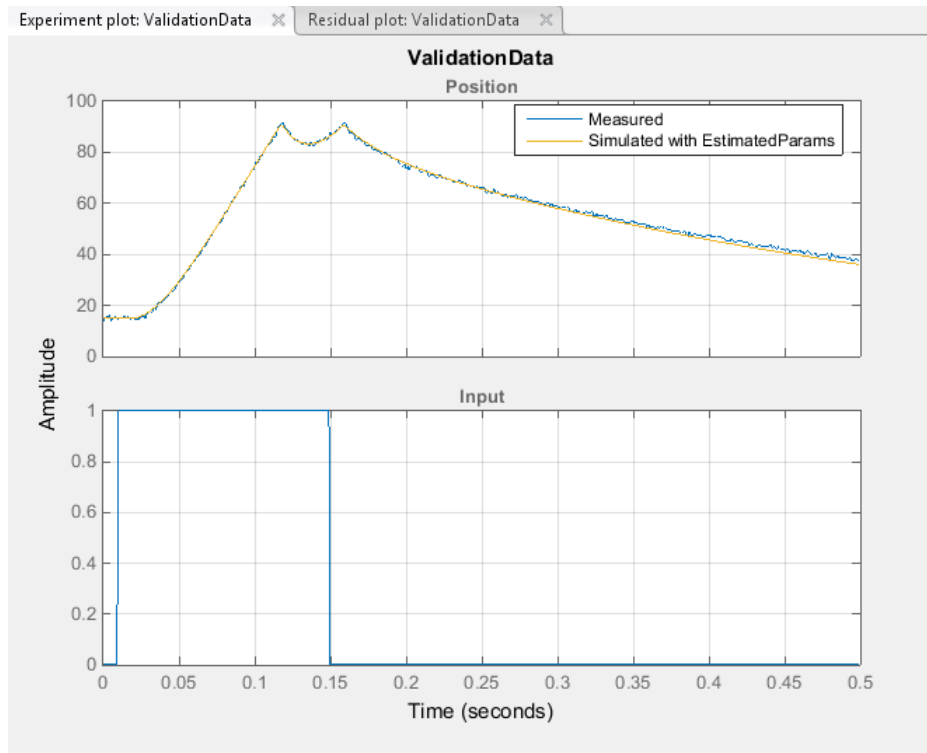
Deselect Use current parameter values and select EstimatedParams, and click **OK**.

- 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.



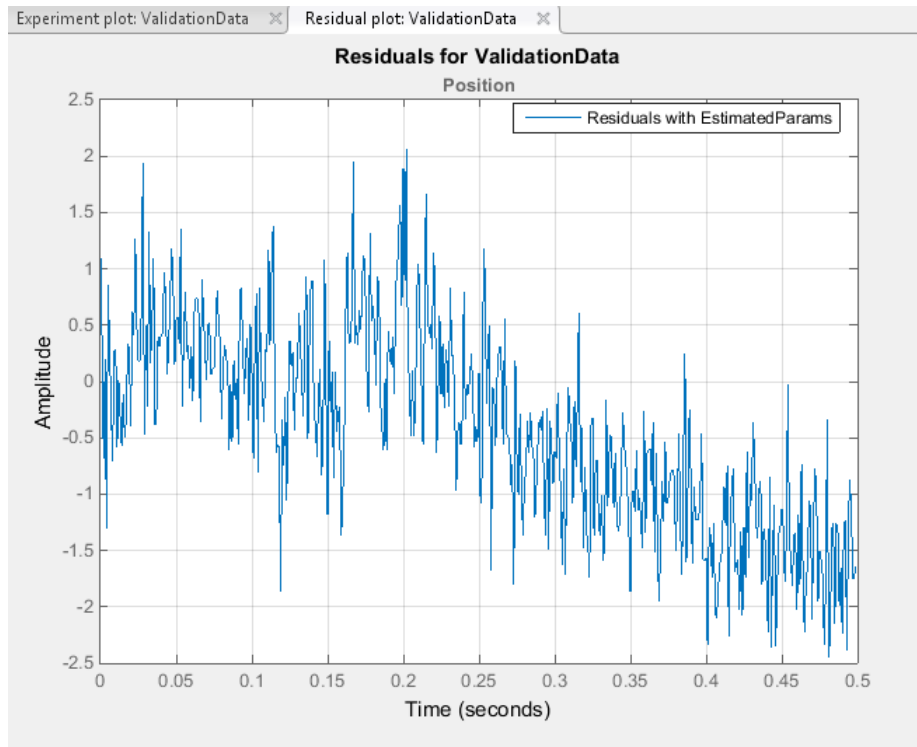
On the **Validation** tab, click **Validate** to start validation.

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



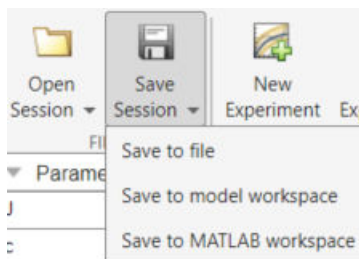
The simulated response as shown in light brown on the top experiment plot is overlaid on the measured output data, and closely matches the measured validation data.

- 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” on page 3-2
- “Design Optimization to Meet Step Response Requirements (GUI)” on page 3-3
- “Design Optimization to Meet Step Response Requirements (Code)” on page 3-15
- “Design Optimization to Track Reference Signal (GUI)” on page 3-21
- “Design Optimization Using Frequency-Domain Check Blocks (GUI)” on page 3-32
- “Time-Domain Model Verification” on page 3-42

Supported Design Requirements

You can optimize response of Simulink models to meet time-domain 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” (Simulink Control Design) blocks of the Simulink Control Design software library.

Related Examples

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

“Design Optimization to Meet Step Response Requirements (Code)” on page 3-15

“Design Optimization to Track Reference Signal (GUI)” on page 3-21

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

“Design Optimization Using Frequency-Domain Check Blocks (GUI)” on page 3-32

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

More About

“How the Optimization Algorithm Formulates Minimization Problems”

“Specify Time-Domain Design Requirements in the App”

“Specify Frequency-Domain Design Requirements in the App”

Design Optimization to Meet Step Response Requirements (GUI)

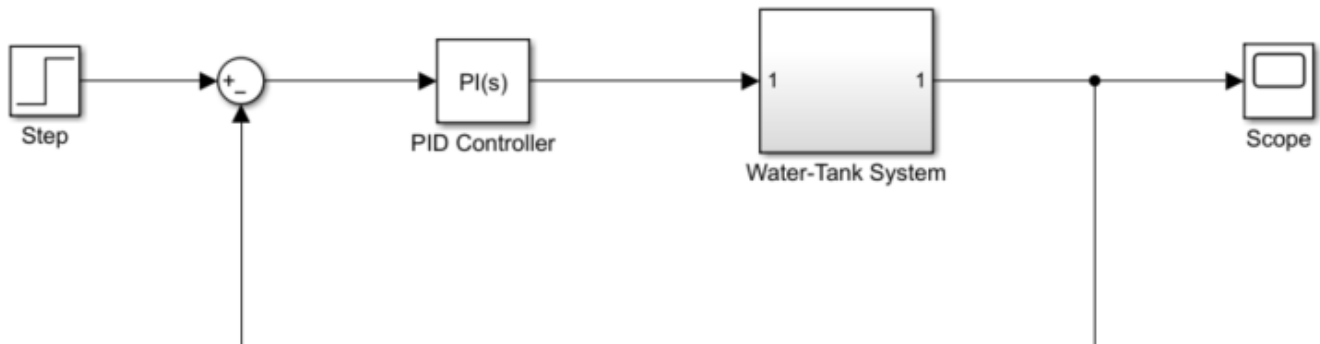
This example shows how to optimize controller parameters to meet step response design requirements using the **Response Optimizer** app. You specify the design requirements in a Check Step Response Characteristics block.

Model Structure

This example uses the `watertank_stepinput` model. This model includes the nonlinear Water-Tank System plant and a PI controller in a single-loop feedback system.

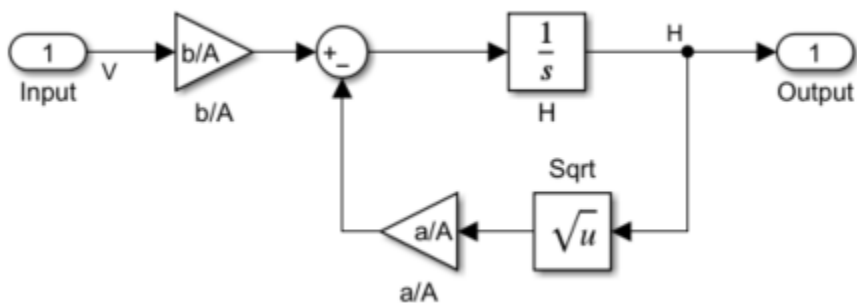
Open the model.

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

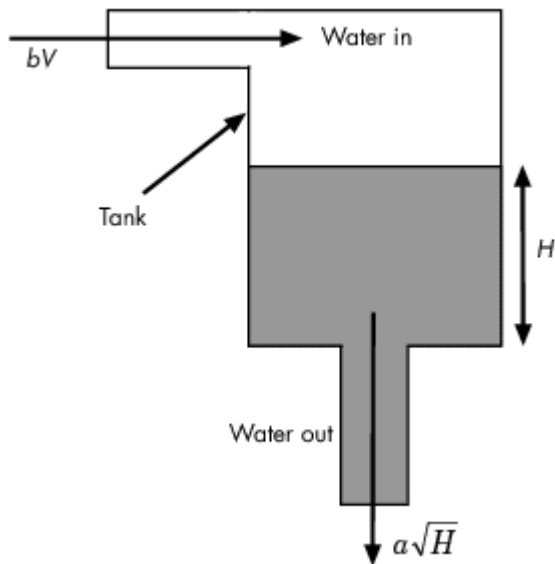


Copyright 2004-2014 The MathWorks, Inc.

To view the water tank model, open the Water-Tank System subsystem.



This model represents the following water tank system.



Here:

- Vol is the volume of water in the tank.
- A is the cross-sectional area of the tank.
- H is the height of water in the tank.
- V is the voltage applied to the pump.
- b is a constant related to the flow rate into the tank.
- a is a constant related to the flow rate out of the tank.

Water enters the tank at the top at a rate proportional to the valve opening. The valve opening is proportional to the voltage, V , applied to the pump. The water leaves through an opening in the tank base at a rate that is proportional to the square root of the water height, H . The presence of the square root in the water flow rate results in a nonlinear plant. Based on these flow rates, the rate of change of the tank volume is:

$$\frac{d}{dt}Vol = A\frac{d}{dt}H = bV - a\sqrt{H}$$

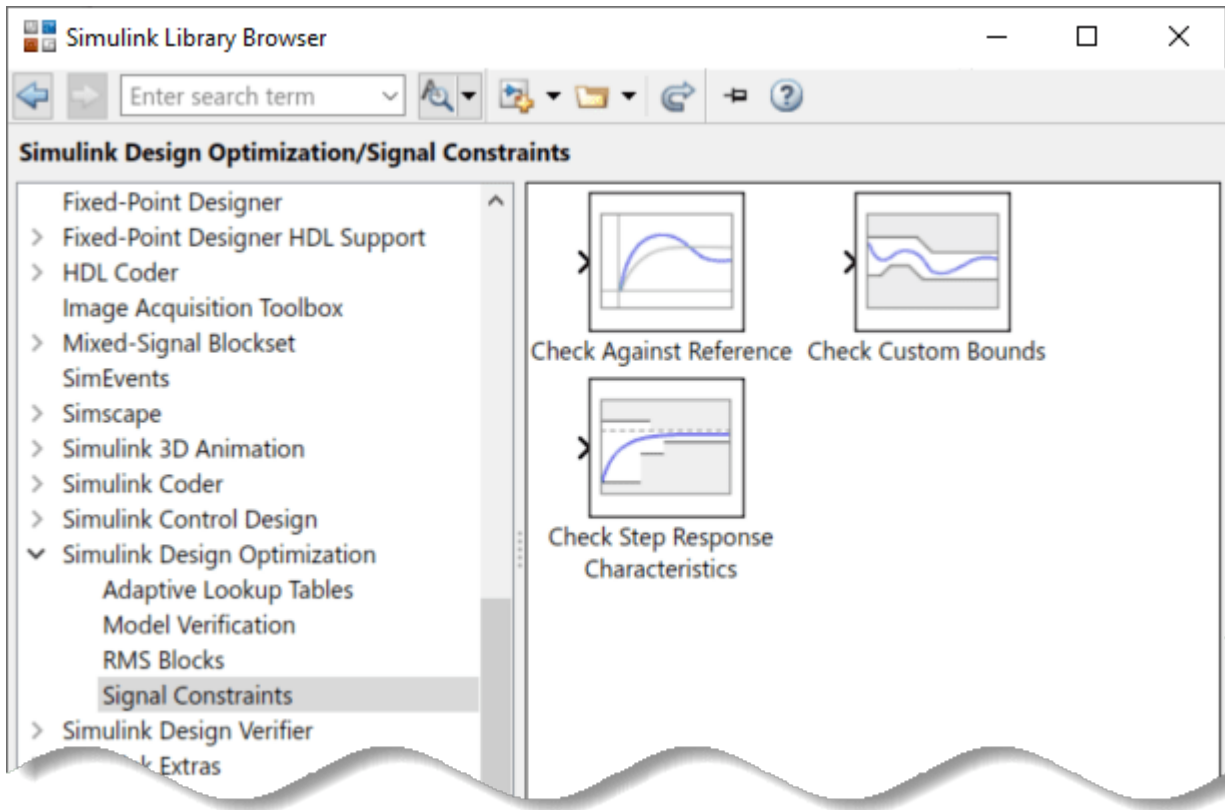
Design Requirements

The height of water in the tank, H , must meet the following step response requirements.

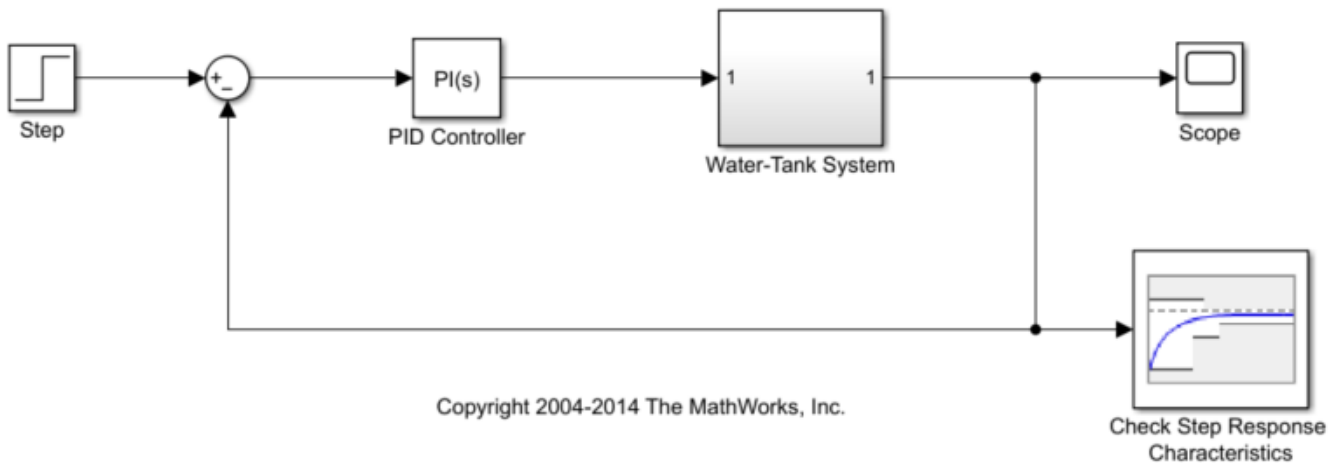
- Rise time less than 2.5 seconds
- Settling time less than 20 seconds
- Overshoot less than 5%

Specify Step Response Requirements

To specify step response requirements, add a Check Step Response Characteristics block to the model. To do so, in the Simulink® model window, on **Simulation** tab, click **Library Browser**. In the **Simulink Design Optimization** list, select **Signal Constraints**.



Drag and drop the Check Step Response Characteristics block into the model window and connect the block to the output. The block is connected to the signal for which you want to specify design requirements.



Double-click the Check Step Response Characteristics block to open the block parameters dialog and specify the following requirements:

- In **Rise time (seconds)**, enter 2.5.

- In **Settling time (seconds)**, enter 20.
- In **% Overshoot**, enter 5.
- In **Initial value**, enter 1.
- In **Final value**, enter 2.

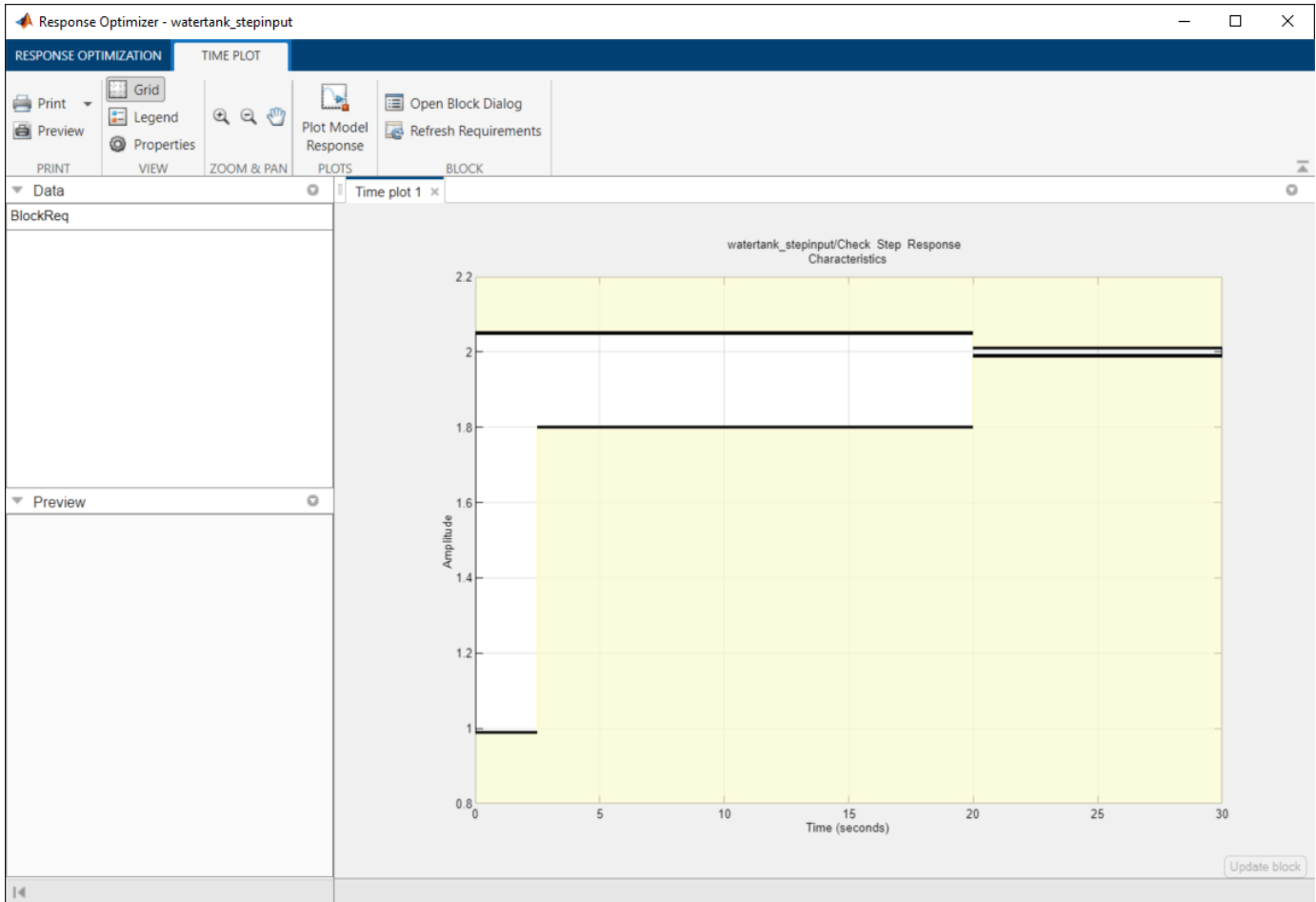
Click **OK**.

Instead of specifying time-domain requirements in the Check blocks, you can specify them in the **Response Optimizer** without adding blocks. For an example that uses this approach, see “Design Optimization to Track Reference Signal (GUI)” on page 3-21.

Specify Design Variables

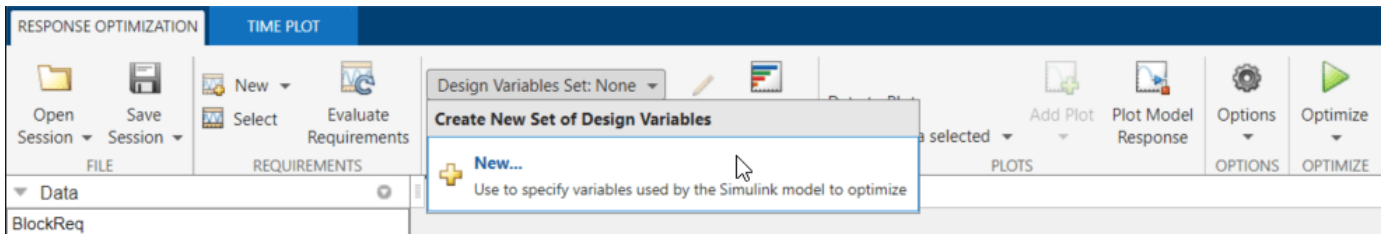
When you optimize the model response, the software modifies the design variable values to meet the design requirements. You specify which model parameters the software can modify.

To open a **Response Optimizer** session for this model, in the Simulink model window, in the **Apps** gallery, click **Response Optimizer**. Alternatively, in the Check Step Response Characteristics block parameters dialog, click **Response Optimization**.

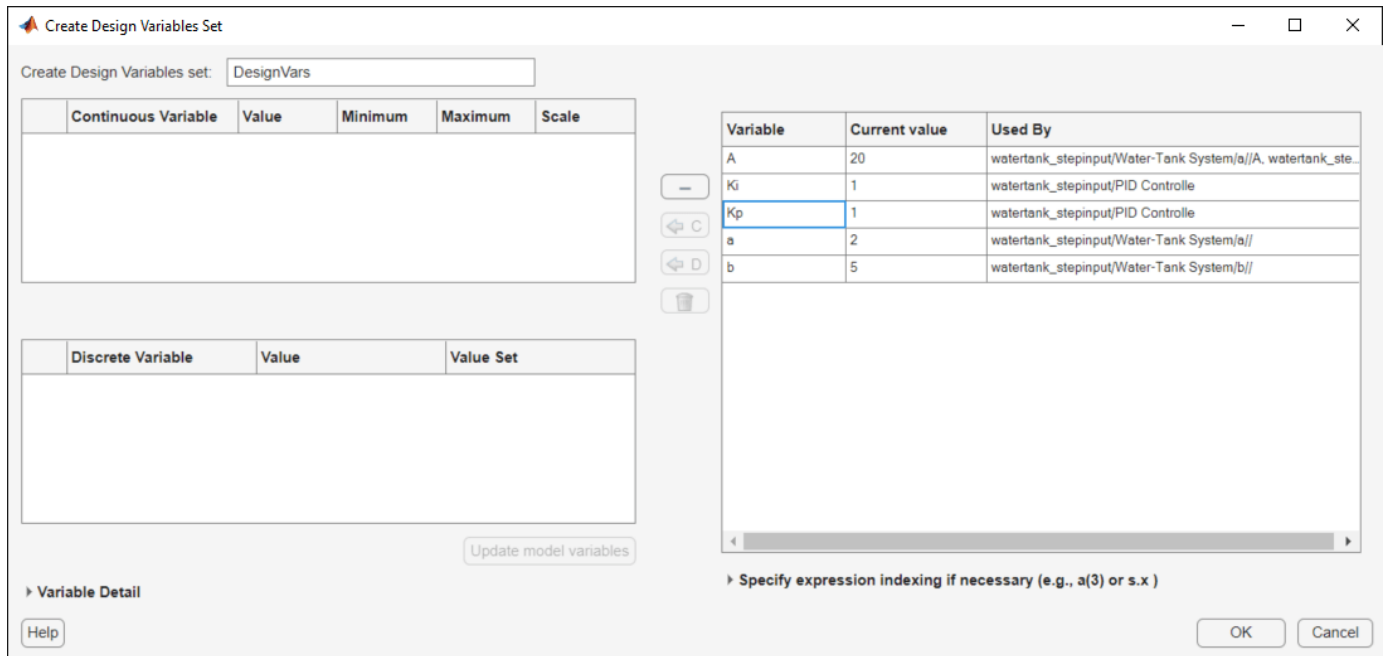



The region bounded by black line segments in **Time plot 1** shows the step response requirements that you specified in the Check Step Response Characteristics block.

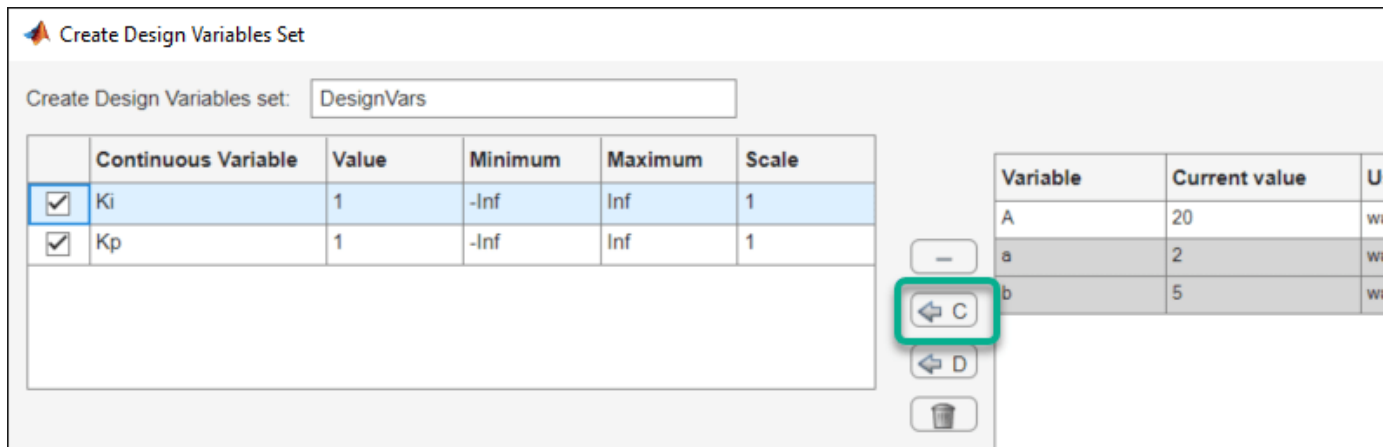
To create a set of design variables, in the **Design Variables Set** list, select **New**.



The Create Design Variables Set dialog box shows model parameters that you can use as design variables and indicates their locations within the model.



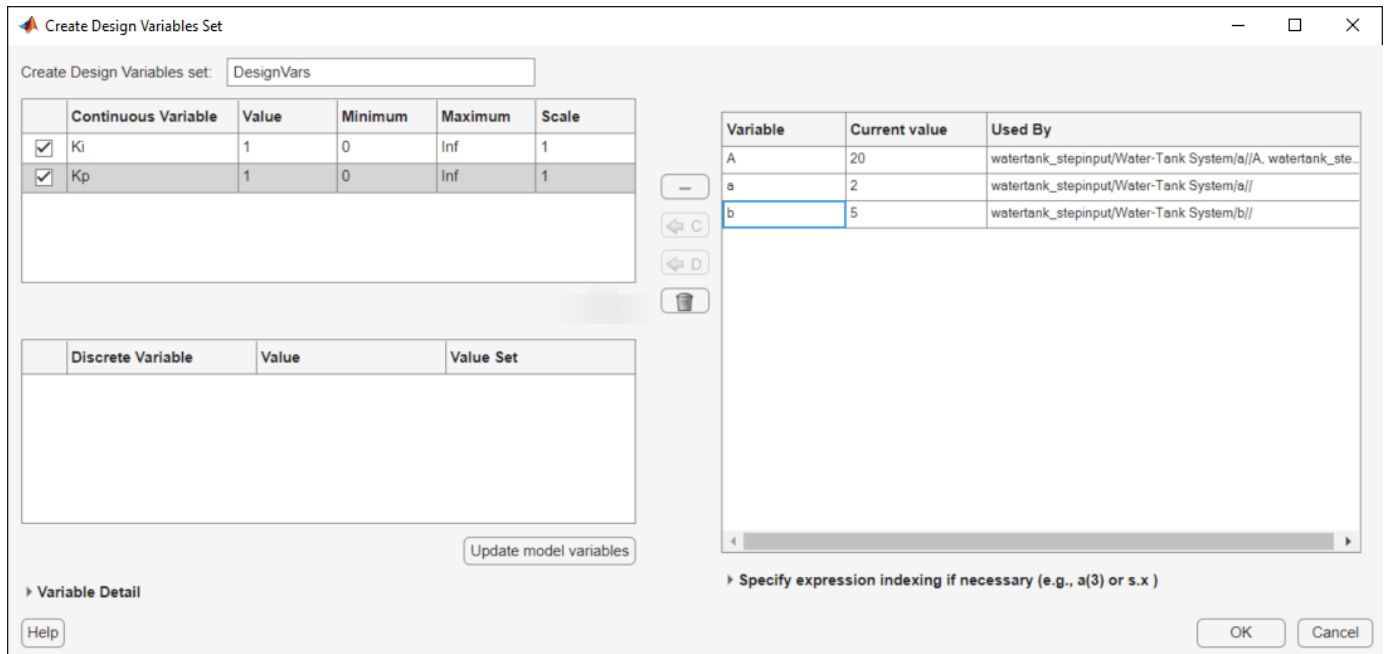
To add parameters to the design variable set, select Ki and Kp, and click .



The design variables list displays the following variable settings:

- **Variable** — Variable name
- **Value** — Current variable value
- **Minimum** and **Maximum** — Variable bounds
- **Scale** — Scaling factor for the variable

Limit the design variables to positive values. To do so, enter 0 for the minimum value of each variable in the corresponding **Minimum** field and press Enter on your keyboard.



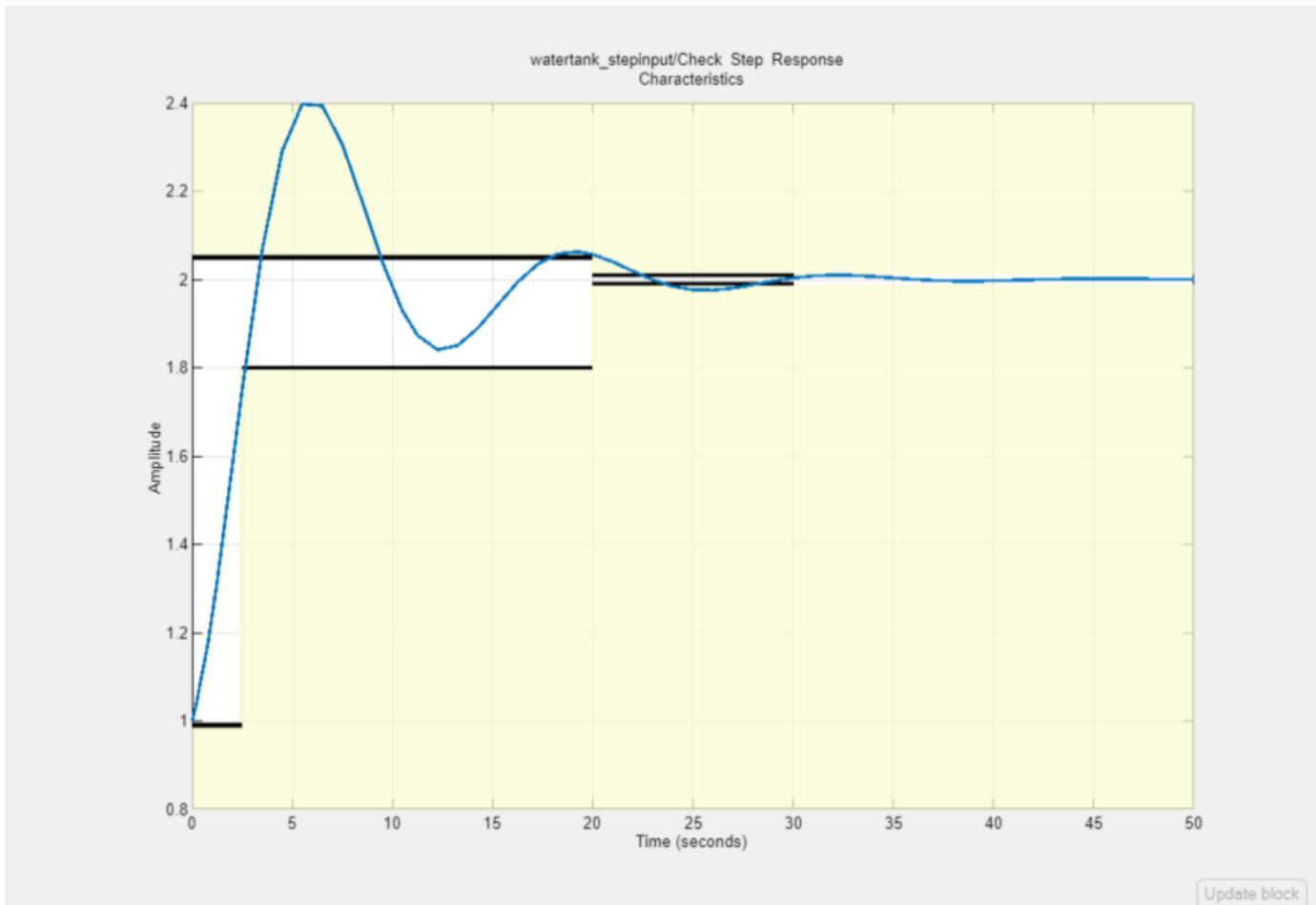
Click **OK**. A new design variable **DesignVars** is created and appears in the **Data** area of **Response Optimizer**. You can click the variable to view its contents in the **Variable Preview** area.



If your model has many parameters, you can first use sensitivity analysis to determine the most influential parameters to optimize, or to obtain initial guesses for the design variables. For more information, see “What is Sensitivity Analysis?” Using the **Sensitivity Analyzer** app, you can explore the response optimization design space by altering the design variables, identify the parameters that most influence the optimization problem, and compute initial values.

Optimize Model Response

To view the current response of the model, click **Plot Model Response**.



The plot shows the model output, depicted by the blue line, lies outside the region of the specified step response.

To optimize the model response, click **Optimize**. The default optimization solver Gradient descent (`fmincon`) modifies the design variables at each iteration so that the simulated response lies within the design requirement line segments. For more information, see “How the Optimization Algorithm Formulates Minimization Problems”.

The screenshot shows a window titled "Optimization Progress Report" with a table and a log area. The table has three columns: "Iteration", "F-count", and "Check Step Response Characteristics (Upper) (<=0)". The log area contains the following text:

```

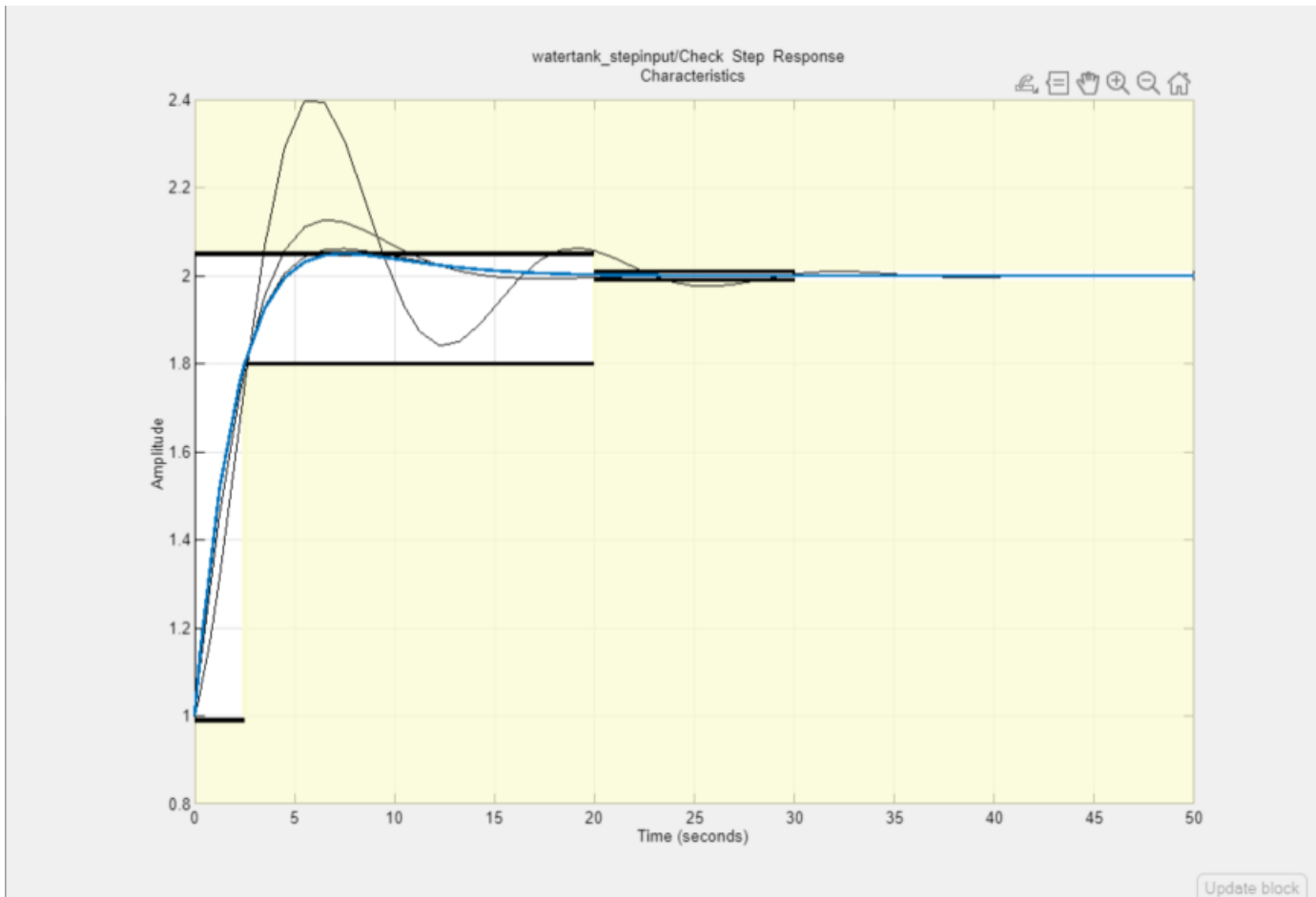
Optimization started 2022-Dec-05, 15:32:05
Optimization converged, 05-Dec-2022 15:32:29
Optimized variable values written to 'DesignVars' in the Design Optimization workspace
'watertank_stepinput' updated with optimized values
Optimized requirement values written to 'ReqValues' in the Design Optimization workspace
Optimization solver output:
Local minimum found that satisfies the constraints.
    
```

At the bottom of the window are three buttons: "Save Iteration...", "Display Options...", and "Optimize".

Iteration	F-count	Check Step Response Characteristics (Upper) (<=0)
0	5	0.8017
1	10	0.0376
2	15	0.0053
3	20	0.0001
4	25	0.0000

The message `Optimization converged` in the Optimization Progress Report 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”.

Verify that the model output meets the step response requirements.



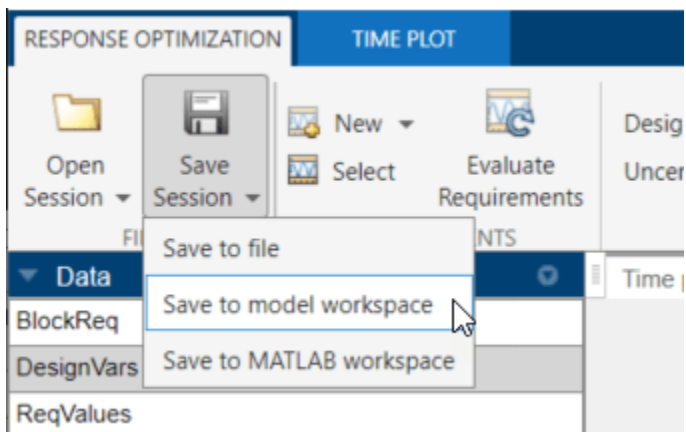
The plot displays the last five iterations. The final response using the optimized variable parameter appears as the thick blue line. The optimized response lies in the white region bounded by the design requirement line segments and thus meets the requirements.

To view the optimized parameter values, click **DesignVars** in **Model 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 **Response Optimizer** session which includes the optimized parameter values.

In the **Response Optimizer**, in the **Response Optimization** tab, in the **Save Session** drop-down list, select **Save to model workspace**.



In the Save Session window, specify the session name in the **Session** field.

To open the saved session, in the **Response Optimizer** for the model, in the **Open Session** drop-down list, click the **Open from model workspace** option.

See Also

Related Examples

- “Design Optimization to Meet Step Response Requirements (Code)” on page 3-15
- “Design Optimization to Track Reference Signal (GUI)” on page 3-21

Design Optimization to Meet Step Response Requirements (Code)

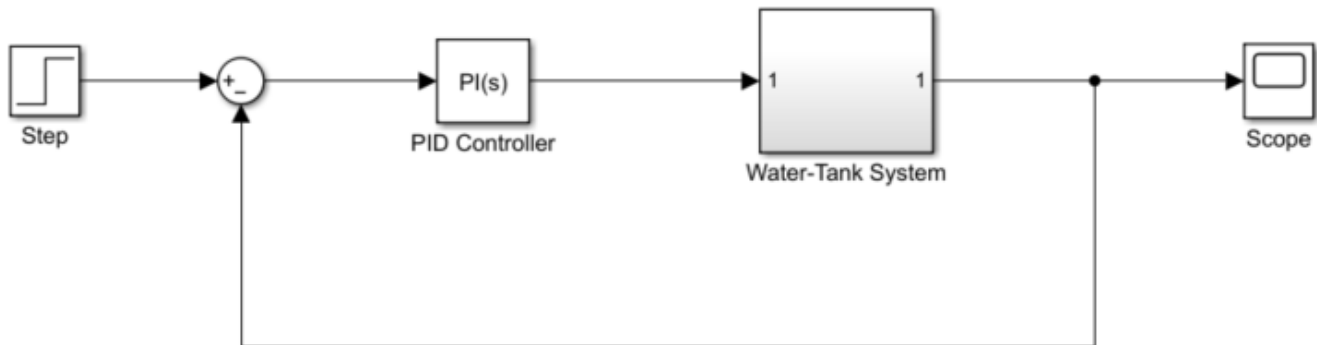
This example shows how to programmatically optimize controller parameters to meet step response requirements using the `sdo.optimize` function.

Model Structure

This example uses the `watertank_stepinput` model. This model includes the nonlinear Water-Tank System plant and a PI controller in a single-loop feedback system.

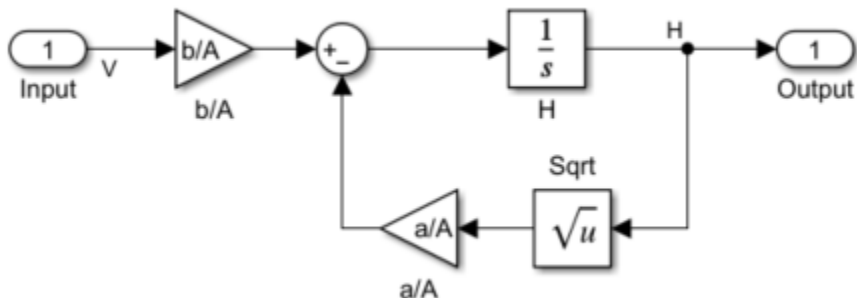
Open the model.

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

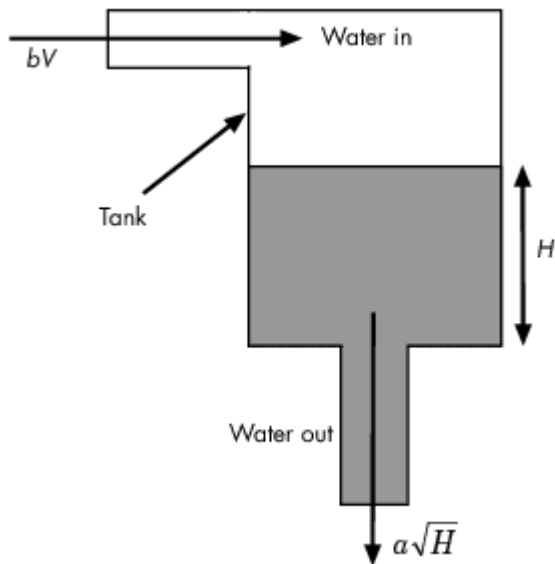


Copyright 2004-2014 The MathWorks, Inc.

To view the water tank model, open the Water-Tank System subsystem.



This model represents the following water tank system.



Here:

- Vol is the volume of water in the tank.
- A is the cross-sectional area of the tank.
- H is the height of water in the tank.
- V is the voltage applied to the pump.
- b is a constant related to the flow rate into the tank.
- a is a constant related to the flow rate out of the tank.

Water enters the tank at the top at a rate proportional to the valve opening. The valve opening is proportional to the voltage, V , applied to the pump. The water leaves through an opening in the tank base at a rate that is proportional to the square root of the water height, H . The presence of the square root in the water flow rate results in a nonlinear plant. Based on these flow rates, the rate of change of the tank volume is:

$$\frac{d}{dt}Vol = A \frac{d}{dt}H = bV - a\sqrt{H}$$

Design Requirements

The height of water in the tank, H , must meet the following step response requirements:

- Rise time less than 2.5 seconds
- Settling time less than 20 seconds
- Overshoot less than 5%

Specify Step Response Requirements

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.

For this model, log the water level, H .

```
PlantOutput = Simulink.SimulationData.SignalLoggingInfo;
PlantOutput.BlockPath = [sys '/Water-Tank System'];
PlantOutput.OutputPortIndex = 1;
PlantOutput.LoggingInfo.NameMode = 1;
PlantOutput.LoggingInfo.LoggingName = 'PlantOutput';
```

Next, create a `sdo.SimulationTest` object to store the logging information. You also use this later to simulate the model.

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

Specify the step response requirements.

```
StepResp = sdo.requirements.StepResponseEnvelope;
StepResp.RiseTime = 2.5;
StepResp.SettlingTime = 20;
StepResp.PercentOvershoot = 5;
StepResp.FinalValue = 2;
StepResp.InitialValue = 1;
```

`StepResp` is a `sdo.requirements.StepResponseEnvelope` object. The values assigned to `StepResp.FinalValue` and `StepResp.InitialValue` correspond to a step change in the water tank height from 1 to 2.

Specify Design Variables

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

Select model parameters to optimize. Here, optimize the parameters of the PID controller.

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

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

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

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

Optimize Model Response

Create a design function to evaluate the system performance for a set of parameter values.

```
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. To view this function, type `edit sldo_model1_design` at command line.

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

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

Examine the nonlinear inequality constraints.

```
initDesign.Cleq
```

```
ans = 8×1
    0.1739
    0.0169
   -0.0002
   -0.0101
   -0.0229
    0.0073
   -0.0031
    0.0423
```

Some `Cleq` values are positive, beyond the specified tolerance, which indicates the response using the current parameter values violate the design requirements.

Specify optimization options.

```
opt = sdo.OptimizeOptions;
opt.MethodOptions.Algorithm = 'sqp';
```

The software configures `opt` to use the default optimization method, `fmincon`, and the sequential quadratic programming algorithm for `fmincon`.

Optimize the response.

```
[p0opt,optInfo] = sdo.optimize(evalDesign,p,opt);
```

```
Optimization started 2023-Mar-03, 15:14:41
```

Iter	F-count	f(x)	max constraint	Step-size	First-order optimality
0	5	0	0.1739		
1	10	0	0.03411	1	0.81
2	15	0	0	0.235	0.0429
3	15	0	0	2.71e-19	0

```
Local minimum found that satisfies the constraints.
```

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

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 “How the Optimization Algorithm Formulates Minimization Problems”.

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”.

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 `Cleq` and `exitflag` fields.

`Cleq` shows the optimized nonlinear inequality constraints.

```
optInfo.Cleq
```

```
ans = 8x1
    -0.0001
    -0.0028
    -0.0050
    -0.0101
    -0.0135
    -0.0050
    -0.0050
    -0.0732
```

All values satisfy $C_{leq} \leq 0$ within the optimization tolerances, which indicates that the step response requirements are satisfied.

`exitflag` identifies why the optimization terminated.

```
optInfo.exitflag
```

```
ans = 1
```

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

View the optimized parameter values.

```
p0pt
```

```
p0pt(1,1) =
```

```
    Name: 'Kp'
    Value: 2.0545
    Minimum: 0
    Maximum: Inf
    Free: 1
    Scale: 1
    Info: [1x1 struct]
```

```
p0pt(2,1) =
```

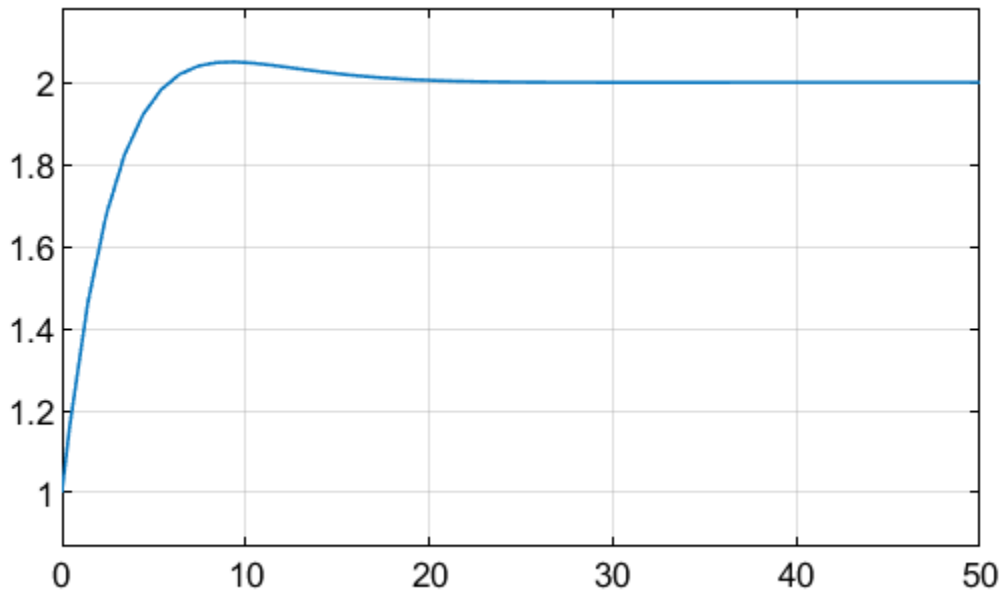
```
    Name: 'Ki'
    Value: 0.3801
    Minimum: 0
    Maximum: Inf
    Free: 1
    Scale: 1
    Info: [1x1 struct]
```

```
2x1 param.Continuous
```

Simulate the model with the optimized values.

```
sdo.setValueInModel(sys,p0pt);
sim(sys);
```

Verify that the model output meets the step response requirements.



Close the model.

```
close_system(sys,0);
```

See Also

[sdo.optimize](#) | [Simulink.SimulationData.SignalLoggingInfo](#) | [sdo.SimulationTest](#) | [sdo.getParameterFromModel](#) | [sdo.requirements.StepResponseEnvelope](#) | [param.Continuous](#) | [sdo.OptimizeOptions](#)

Related Examples

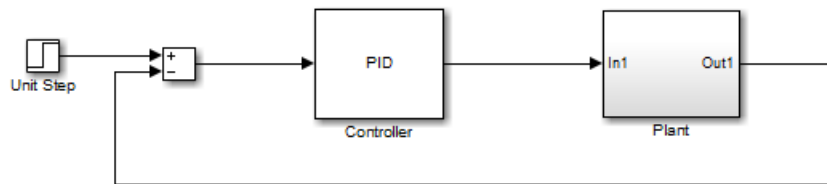
- “Design Optimization to Meet Step Response Requirements (GUI)” on page 3-3
- “Design Optimization to Track Reference Signal (GUI)” on page 3-21
- “Discrete-Valued Variables in Response Optimization (Code)”

Design Optimization to Track Reference Signal (GUI)

This example shows how to optimize controller parameters to track a reference signal using the **Response Optimizer**. You specify the reference signal without adding any Check blocks to the model.

Model Structure

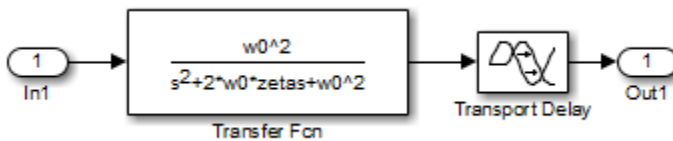
The model `sldo_model1` includes these blocks:



- Controller block, a PID controller, controls the output of the Plant subsystem.
- Unit Step block applies a step input.

You can also use other types of inputs, such as ramp, to optimize the response generated by such inputs.

- Plant subsystem is a second-order system with delay. It contains Transfer Function and Transport Delay blocks.



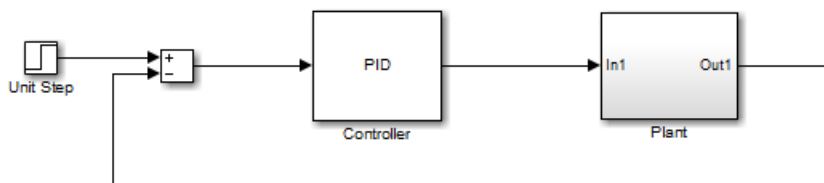
Design Requirements

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

Specify Reference Signal

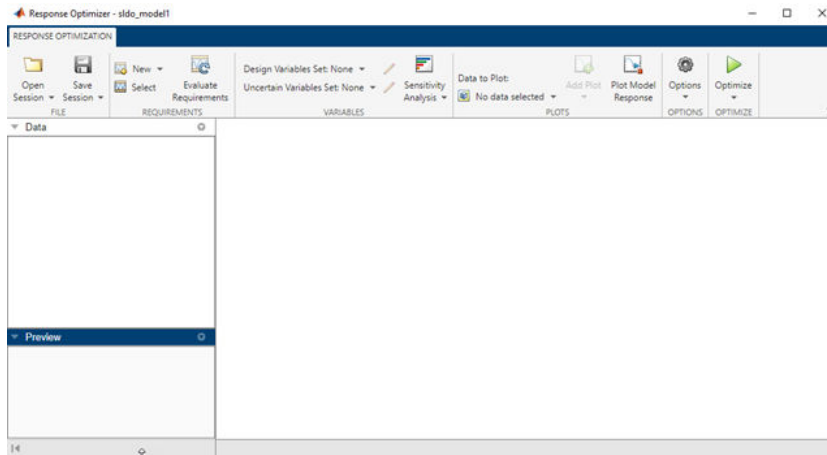
- 1 Open the Simulink model.

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

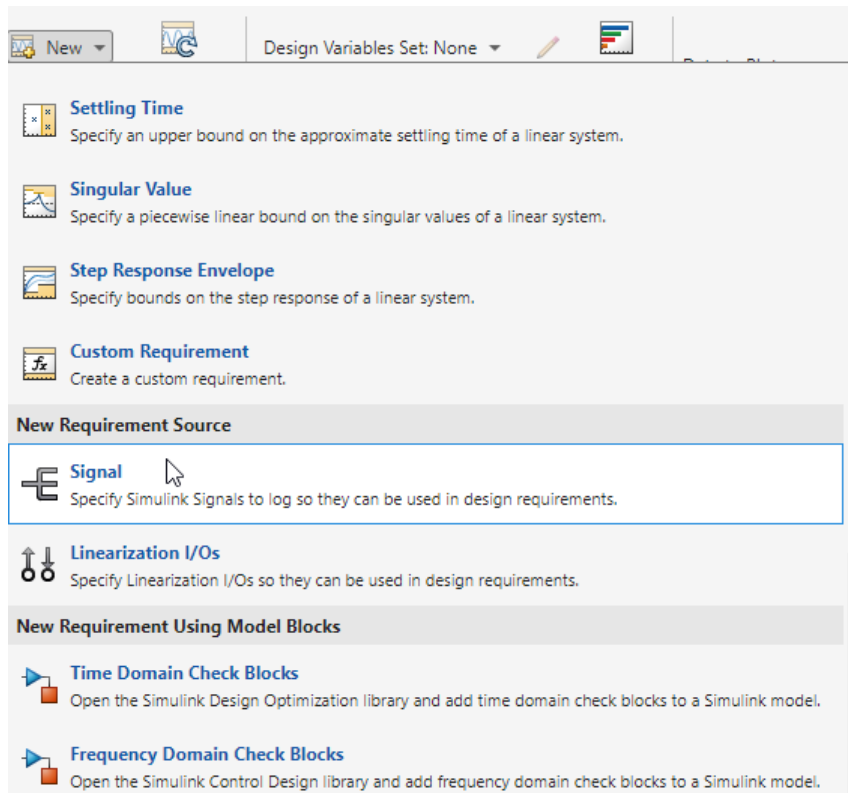


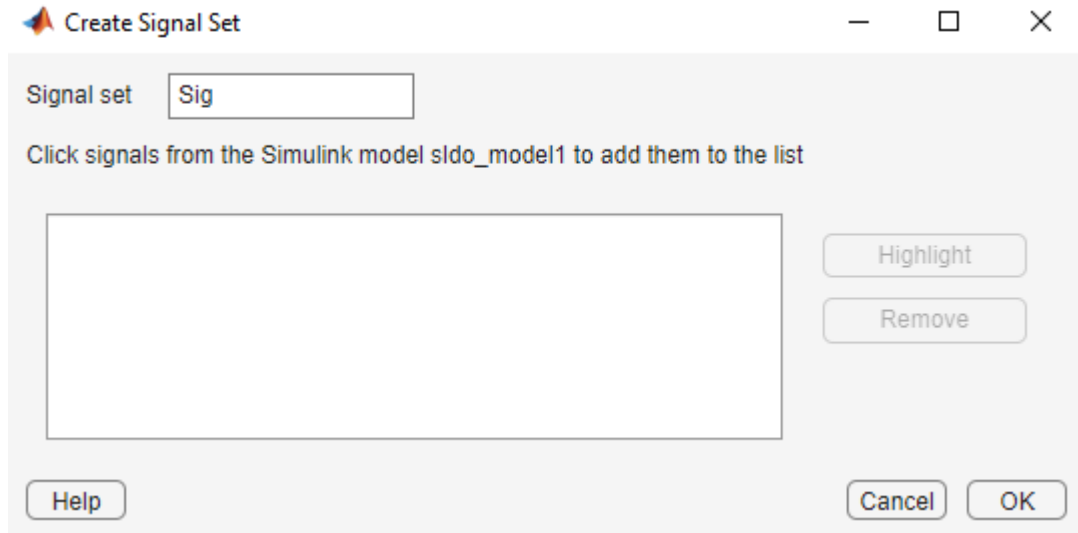
To learn more about the model, see “Model Structure” on page 3-21.

- 2 To open the **Response Optimizer**, in the Simulink model window, from the **Apps** tab, in the gallery, under **Control Systems**, select **Response Optimizer**.

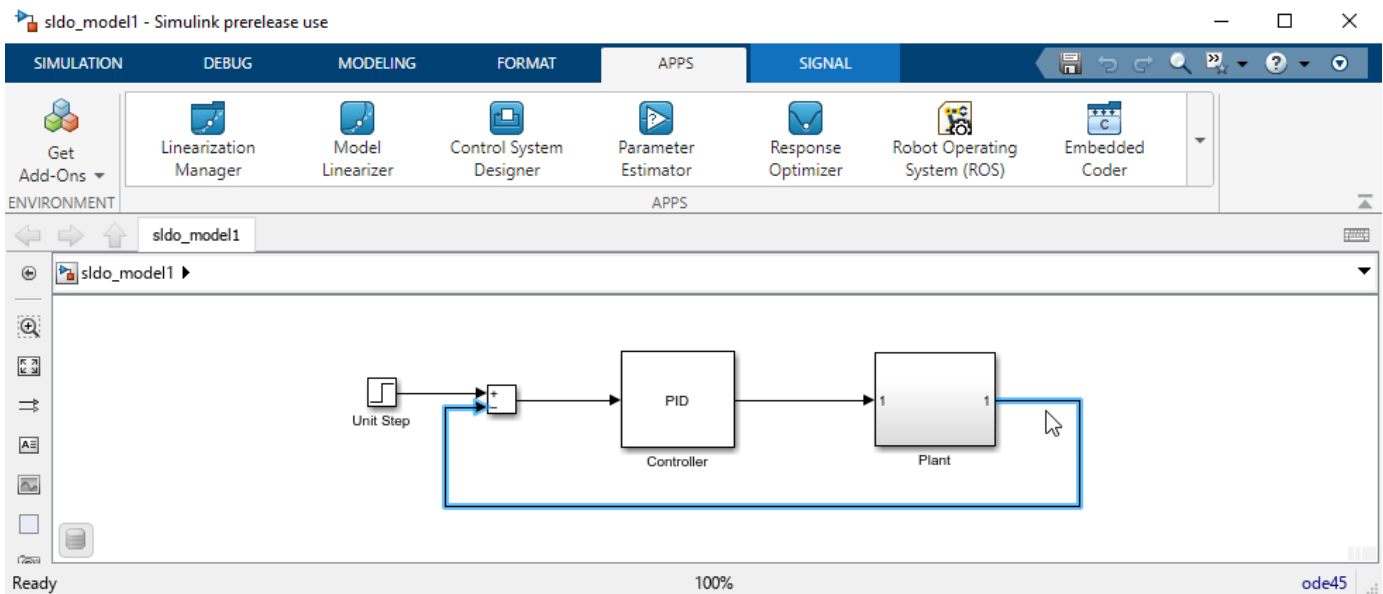


- 3 Select the model signal to track the reference signal.
 - a In the **New** drop-down list, select **Signal** to open the Create Signal Set window.

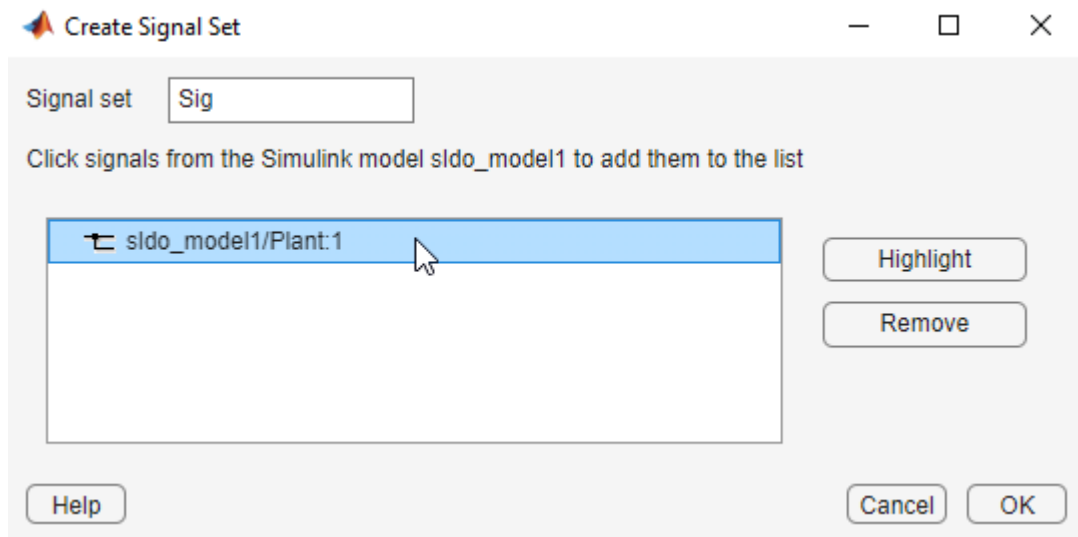




- b** To display the signal in the window, click the output of the Plant block in the Simulink model window.



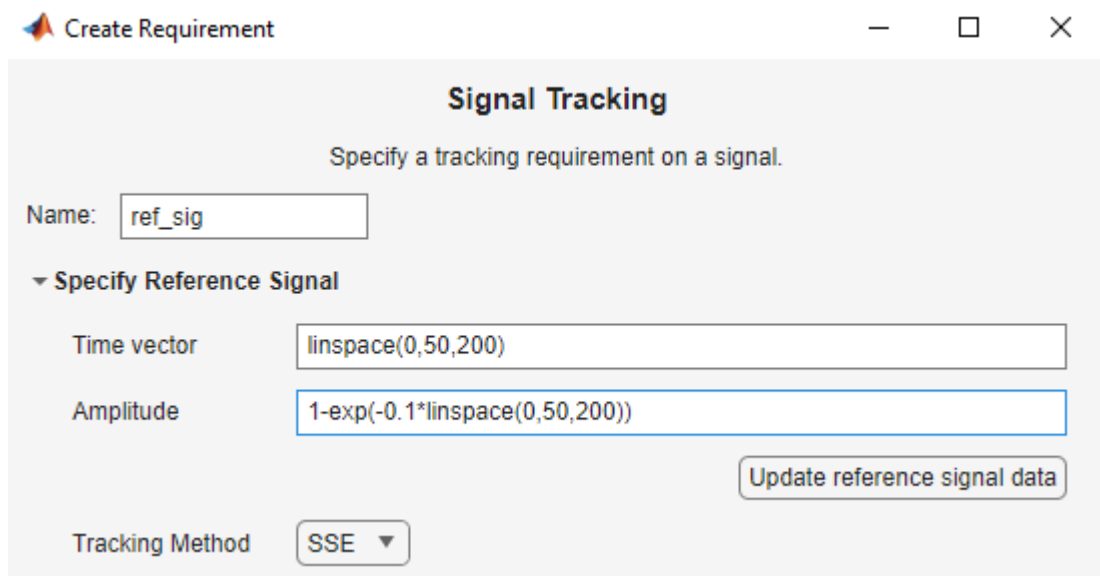
- c** Select the signal to add it to the signal set.



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

Click **OK** to create the variable PlantOutput. It appears in the **Data** area of the **Response Optimizer**.

- 4 Specify the reference signal for the model output to track.
- In the **New** drop-down list, select **Signal Tracking** to open a Create Requirement window.
 - In the **Name** edit box, enter `ref_sig`.
 - In the **Time vector** edit box, enter `linspace(0,50,200)`
 - In the **Amplitude** edit box, enter `1-exp(-0.1*linspace(0,50,200))`.



Leave the **Tracking Method** as SSE which means, 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 In the **Specify Signal to Track Reference Signal** area, select the check-box corresponding to the signal you selected in the previous step, and click **OK**.

Signal Tracking

Specify a tracking requirement on a signal.

Name:

▼ **Specify Reference Signal**

Time vector

Amplitude

Tracking Method

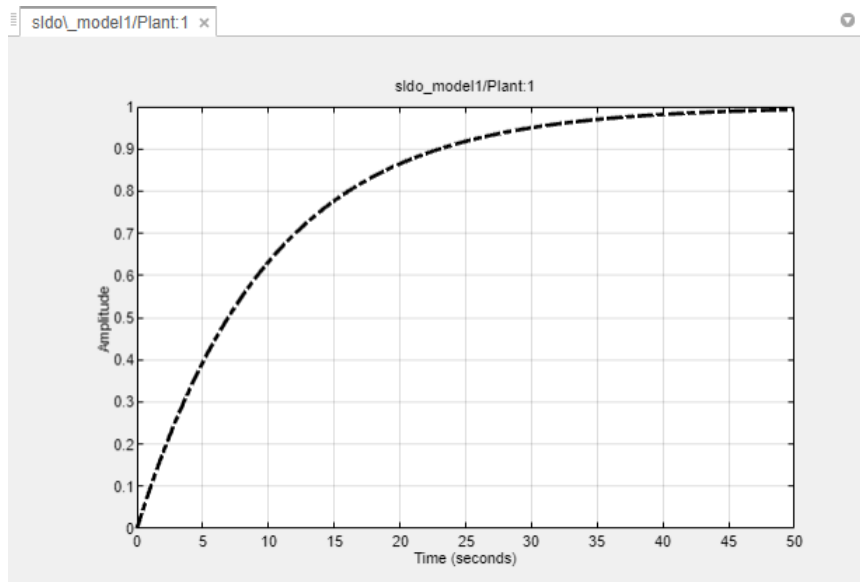
Use robust cost

▼ **Specify Signal to Track Reference Signal**

Create a signal logging definition so that it can be used in requirements.

	Signal	
<input checked="" type="checkbox"/>	PlantOutput (sldo_model1/Plant:1)	<input type="button" value="+"/>
		<input type="button" value="✎"/>

A new reference signal `ref_sig` is created and appears in the **Data** area. The Response Optimization window updates to plot the reference signal.



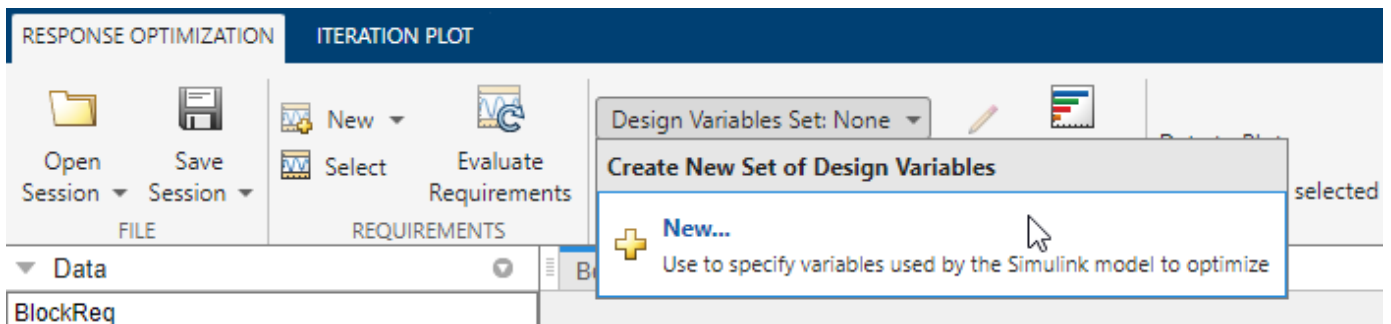
Specify Design Variables

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

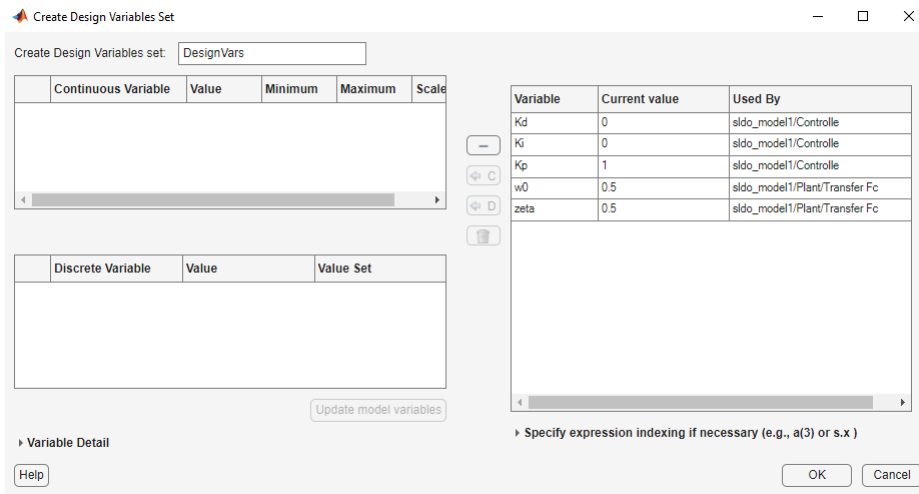
In the **Response Optimization** tab:

- 1 Create a new set of design variables.

In the **Design Variables Set** drop-down list select **New**.

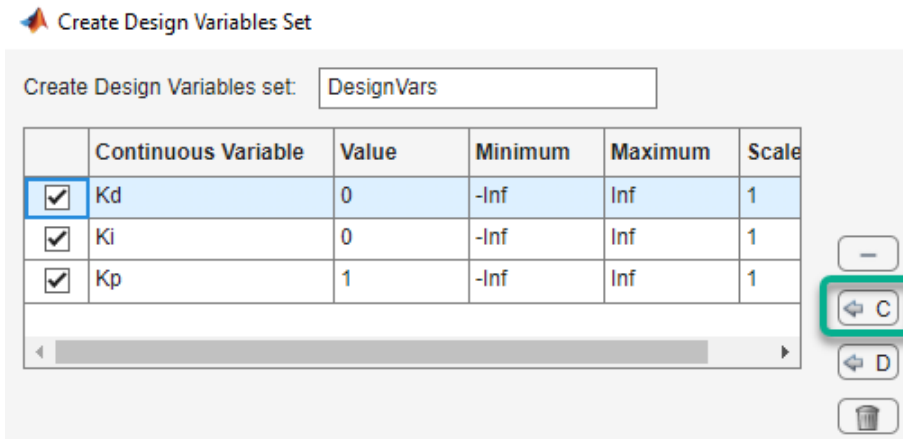


The Create Design Variables Set window shows model parameters that you can use as design variables and indicates their locations within the model subsystems.



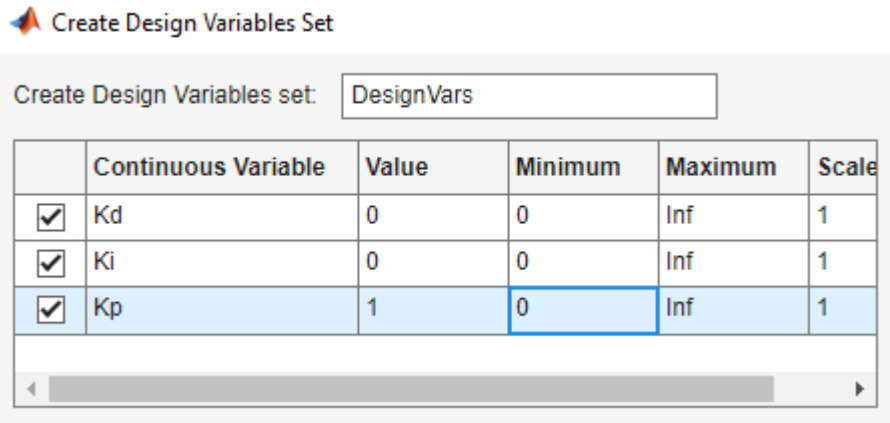
- 2 Add parameters to the design variables set.

Select Kd, Ki, and Kp, and click  to add the selected parameters.

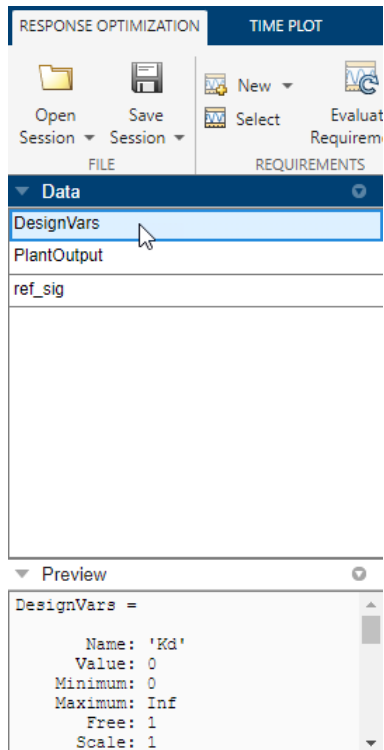


The design variables list displays the following parameter settings:


- **Variable** — Parameter name
 - **Value** — Current parameter value
 - **Minimum** and **Maximum** — Parameter bounds
 - **Scale** — Scaling factor for the parameter
- 3 Limit the parameters to positive values. To do so, enter 0 for the minimum value of each parameter in the corresponding **Minimum** field, and press **Enter** on your keyboard.

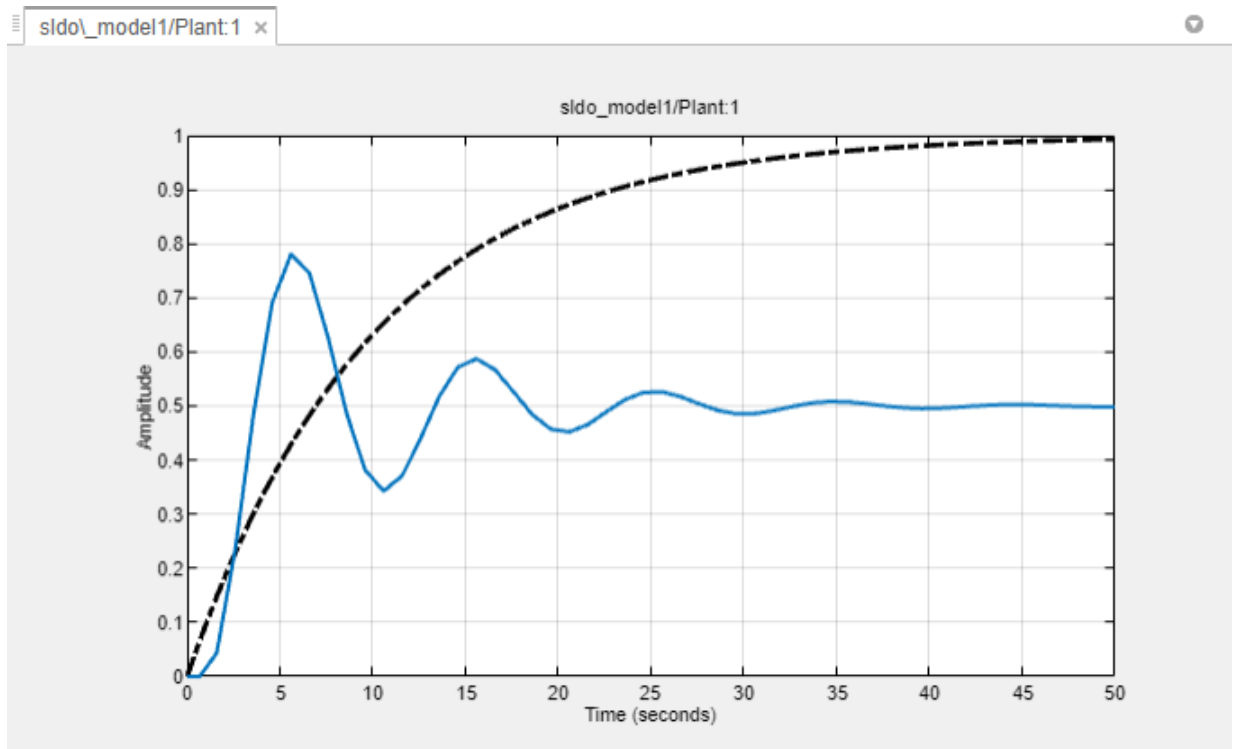


Click **OK**. A new design variable DesignVars is created and appears in the **Data** area of the **Response Optimizer**.



Optimize Model Response

- 1 To view the current model response, click  **Plot Model Response**.



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

2

Click  **Optimize**.

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”.

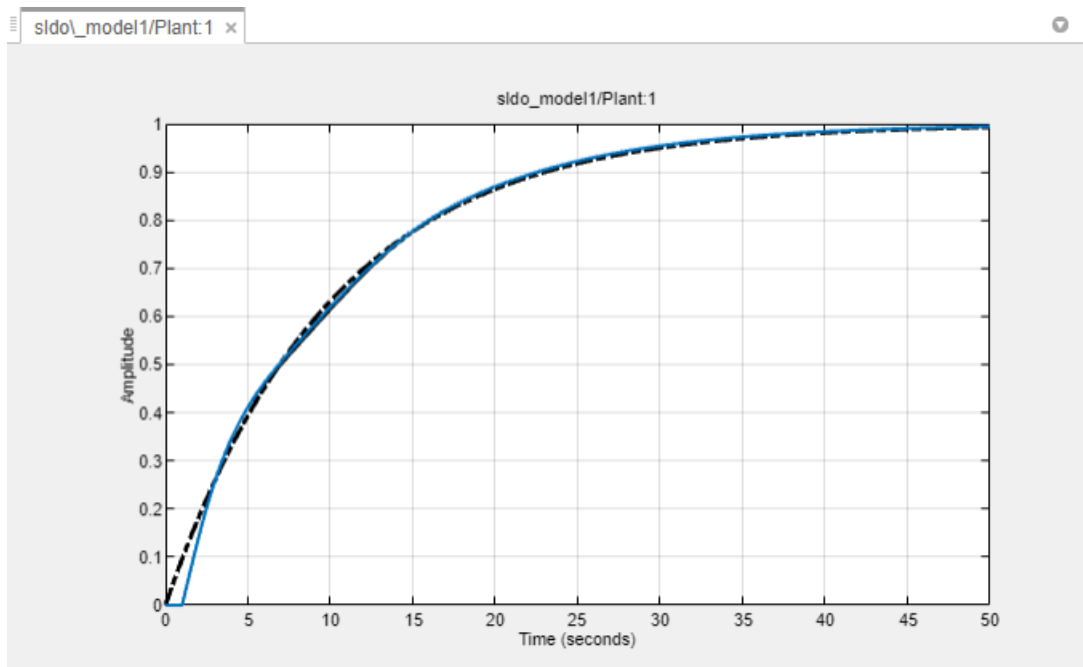
The message **Optimization converged** in the Optimization Progress Report indicates that the optimization method found a solution that tracks the reference signal within the tolerances and parameter bounds. For more information about the outputs displayed in the Optimization Progress Report, see “Iterative Display”.

Iteration	F-count	ref_sig (Minimize)
16	146	0.0942
17	153	0.0509
18	160	0.0431
19	167	0.0418
20	174	0.0417
21	181	0.0416
22	188	0.0416

Optimization started 2022-Dec-01, 15:54:33
 Optimization converged, 01-Dec-2022 15:55:32
 Optimized variable values written to 'DesignVars' in the Design Optimization workspace
 'sldo_model1' updated with optimized values
 Optimized requirement values written to 'ReqValues' in the Design Optimization workspace
 Optimization solver output:

Save Iteration... Display Options... Optimize

- Verify that the response tracks the reference signal by observing the amplitude versus time plot.



The optimized response closely tracks the reference signal.

- To view the optimized parameter values, click **DesignVars** in the **Data** area of the **Response Optimizer**. View the updated values in the **Variable Preview** area.

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

See Also

Related Examples

- “Design Optimization to Meet Step Response Requirements (GUI)” on page 3-3
- “Design Optimization to Meet Step Response Requirements (Code)” on page 3-15

Design Optimization Using Frequency-Domain Check Blocks (GUI)

This example shows how to optimize model parameters to meet frequency-domain requirements using the **Response Optimizer**. 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. You optimize rectifier filter parameters to meet gain and bandwidth requirements by minimizing a custom objective.

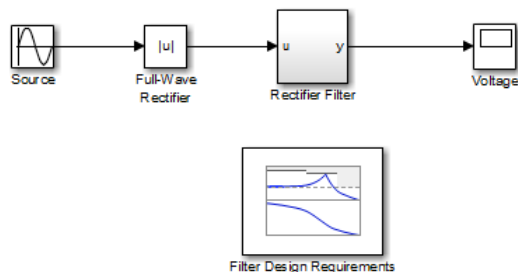
Full Wave Rectifier Model

Open the model.

```
open_system('sdorectifier.slx')
```

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

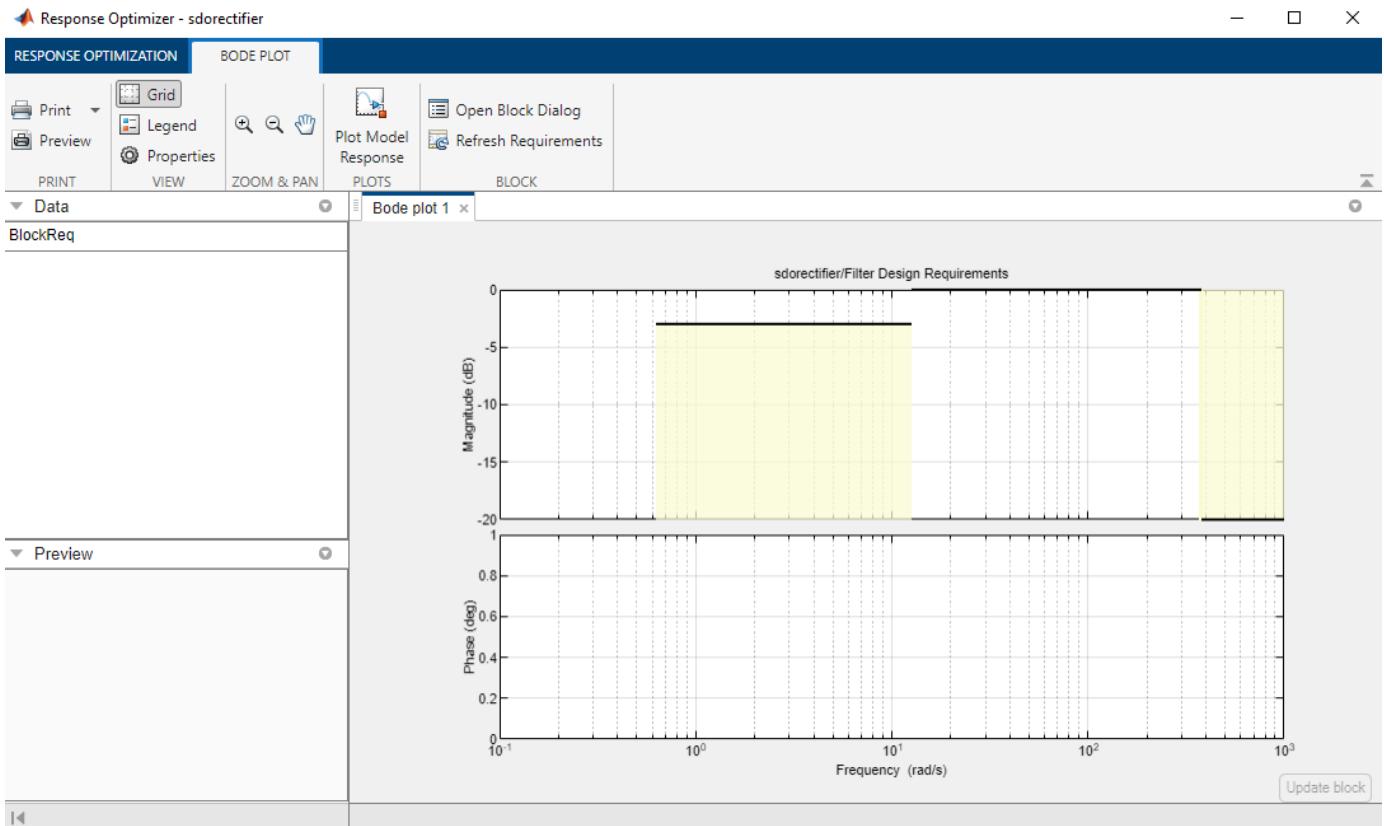
The design optimization problem has several objectives. The design must:

- Have a -3 dB bandwidth of at least 2 Hz
- Limit the gain across the frequency range 2 Hz - 60 Hz to at most 0 dB
- Limit the gain above 60 Hz to at most -20 dB
- 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 To open the app, in the **Apps** gallery, click **Response Optimizer**.



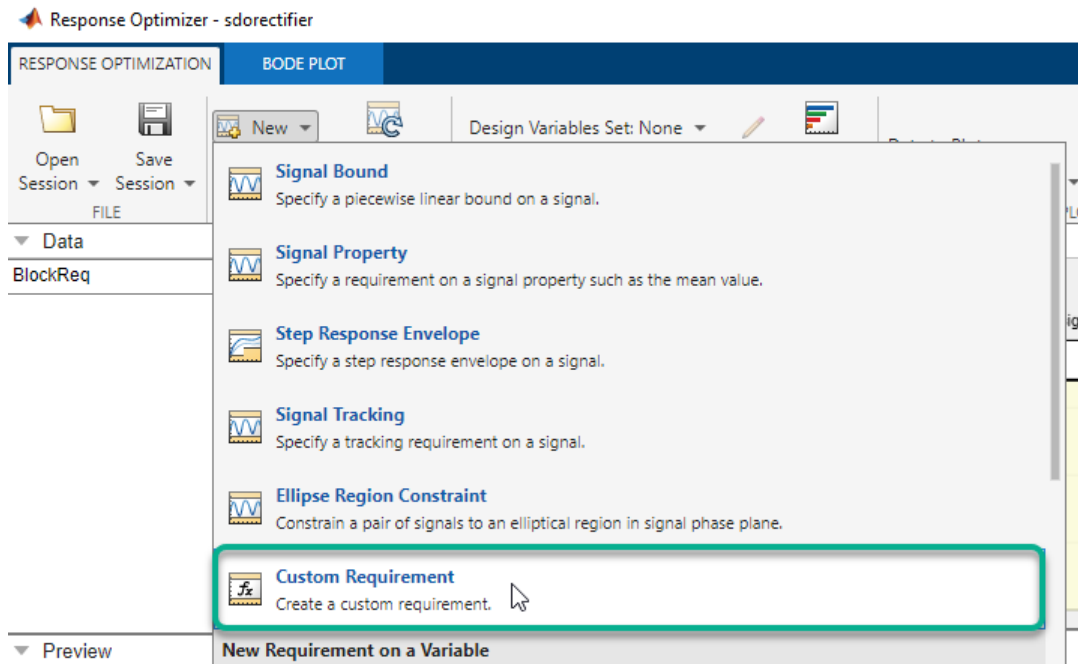
The **Bode plot 1** tab shows the gain and bandwidth requirements specified in the Filter Design Requirements block in the model. To see their values, double-click the block to open the Block Parameters dialog box, and select the **Bounds** tab.

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

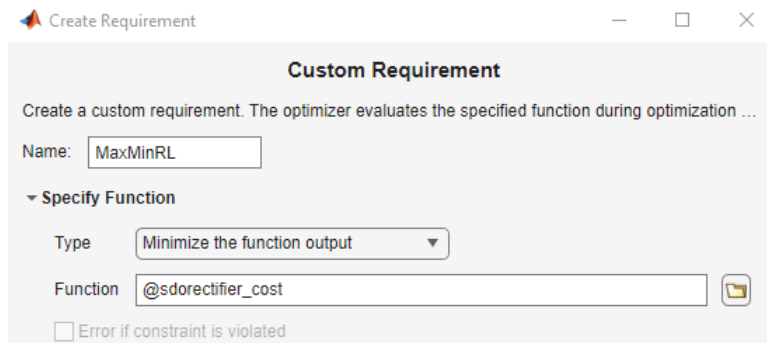
The custom objective is already defined in the `sdorectifier_cost` function. The function accepts the design variables R and L and returns the objective to be minimized.

Tip Type `edit sdorectifier_cost` in the command line to view this function.

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



- b Specify the following values in the Create Requirement window, and click **OK**:
- In the **Name** edit box, enter MaxMinRL.
 - In the **Type** edit box, select Minimize the function output
 - In the **Function** edit box, enter @sdrectifier_cost. The optimization solver calls the specified function handle.

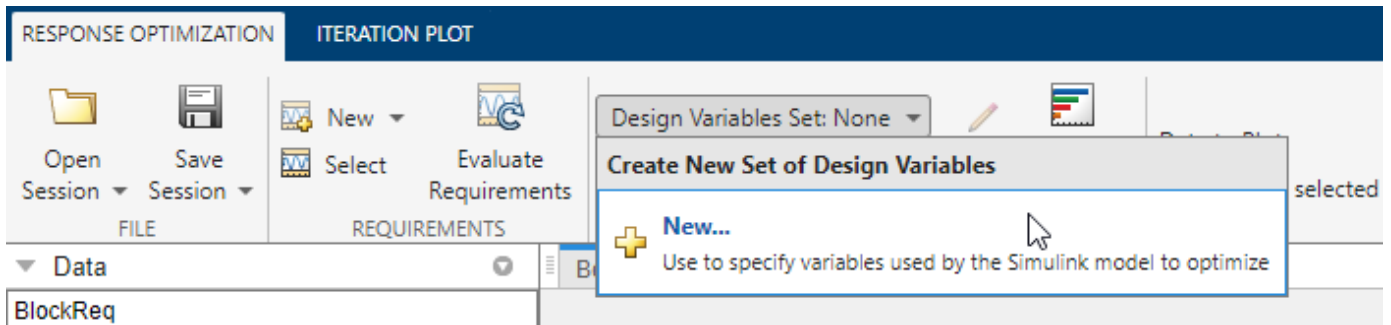



A new requirement variable MaxMinRL is created, and appears in the **Data** area in the **Response Optimizer**. The **Iteration plot 1** tab shows the value of MaxMinRL at each iteration during the optimization.

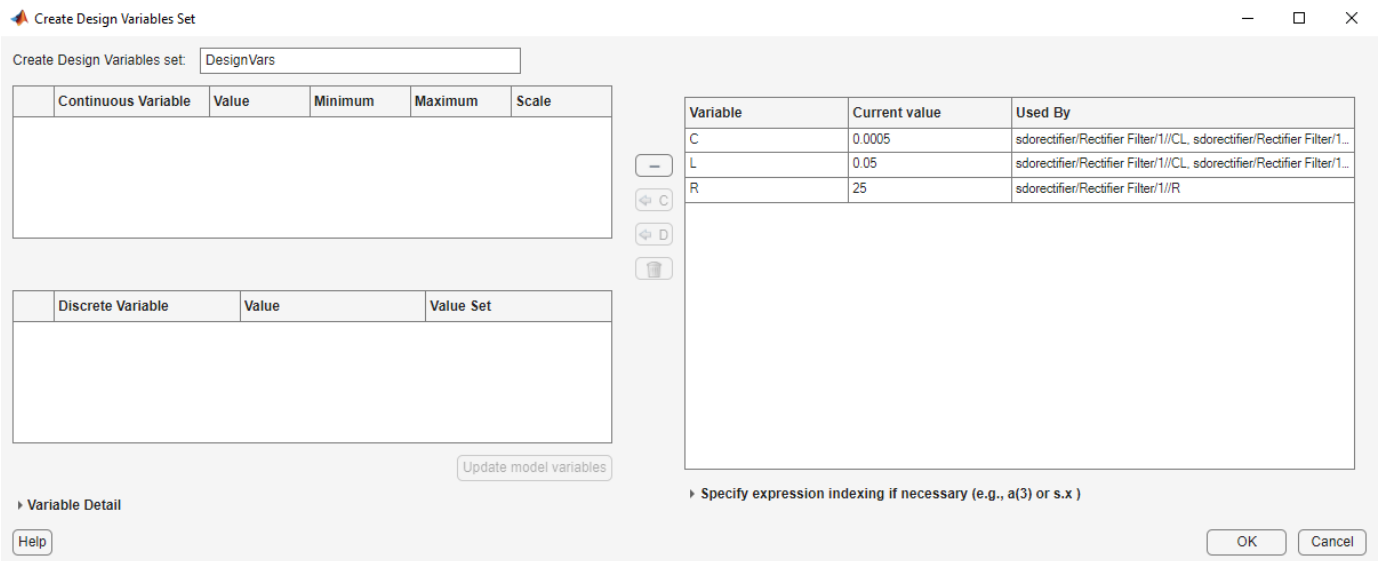
Specify Design Variables

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

- 1 In the **Design Variables Set** drop-down list, select **New**.



Select C, L, and R in the Create Design Variable Set window. Click  to add the selected parameters to a design variables set.



2 Specify the value range for each design variable, and click **OK**:

- C in the range 1 μ F-1 mF
- L in the range 1-500 mH
- R in the range 0.01-50 ohms

Create Design Variables Set

Create Design Variables set:

	Continuous Variable	Value	Minimum	Maximum	Scale
<input checked="" type="checkbox"/>	C	0.0005	1e-06	0.001	0.0009765625
<input checked="" type="checkbox"/>	L	0.05	0.001	0.5	0.0625
<input checked="" type="checkbox"/>	R	25	0.01	50	32

	Discrete Variable	Value	Value Set


Variable

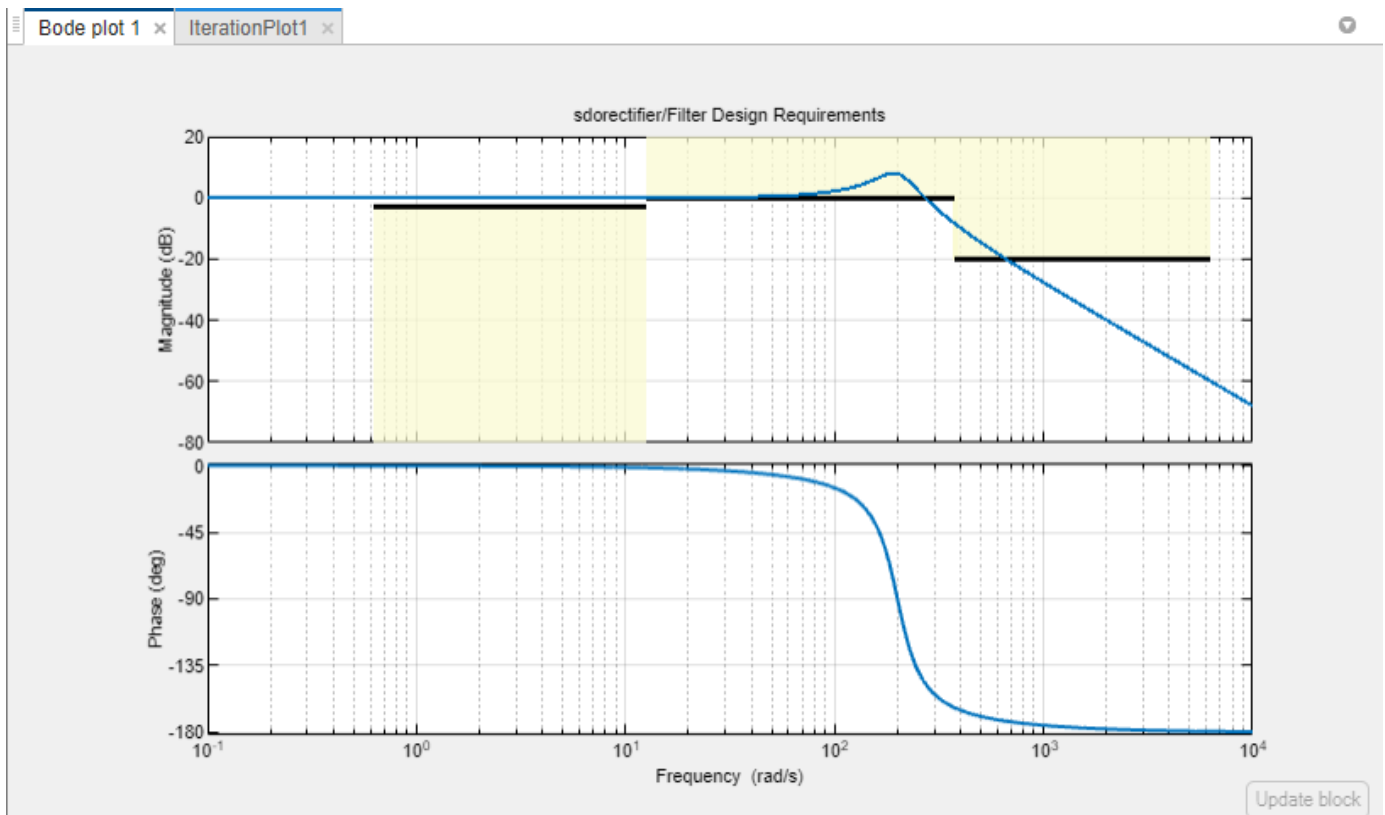
Variable	Current value	Used By

Specify expression indexing if necessary (e.g., a(3) or s.x)

A new variable DesignVars is created, and appears in the **Data** area of the **Response Optimizer**.

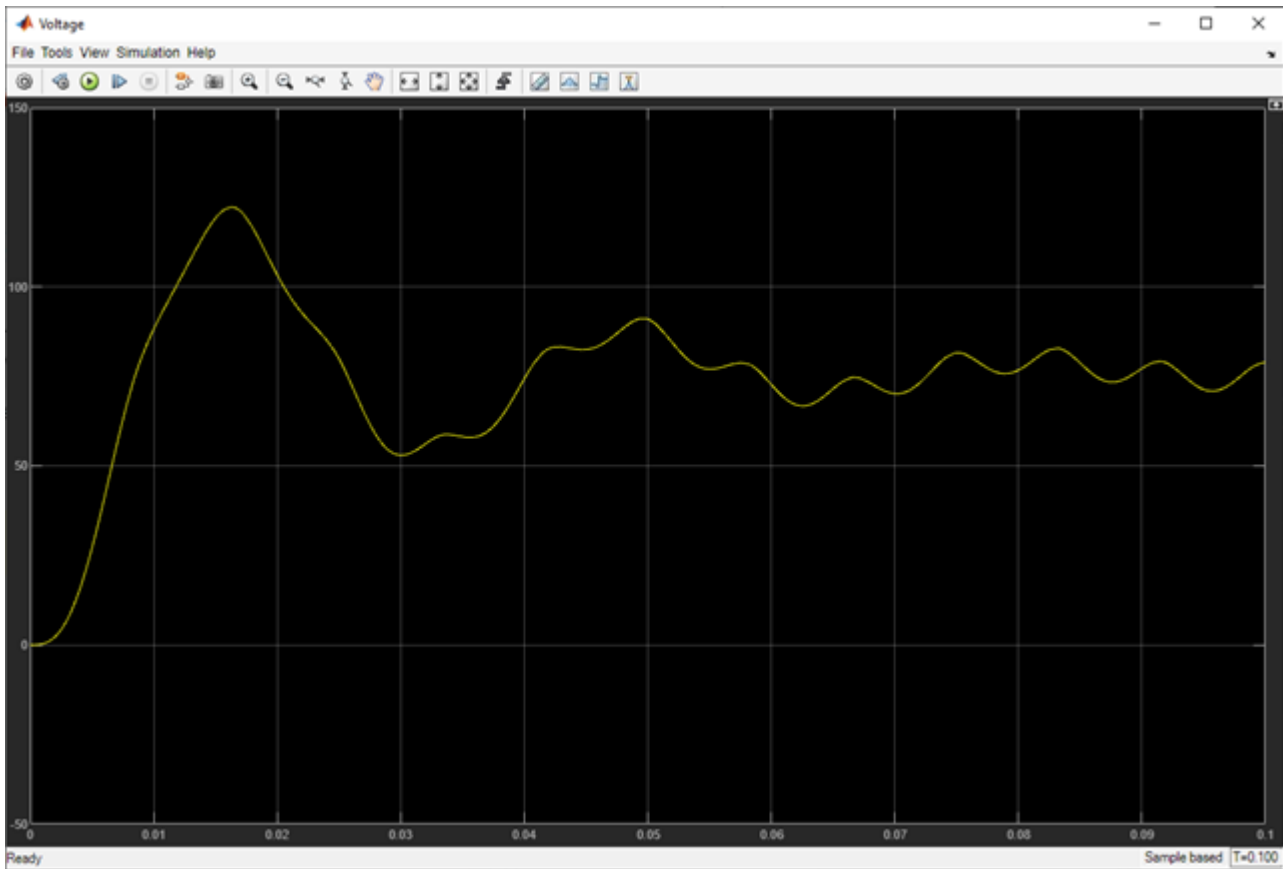
Optimize Design

- 1 To view the current response of the model, click  **Plot Model Response**.



The **Bode plot 1** window in the **Response Optimizer** shows that the model output goes out of the region bounded by the design requirement line segments.

In the **Voltage** scope window, you see that the filter voltage signal overshoots its steady-state value and contains significant harmonic content.



2 Click  **Optimize**.

The **Optimization converged** message in the Optimization Progress Report indicates that the optimization method found a solution to satisfy the filter bandwidth requirements.

Optimization Progress Report

Iteration	F-count	IterationPlot1 (Minimize)	Filter Design Requirements (Upper) (≤ 0)	Filter De (≥ 0)
3	30	1.2432	-0.1784	
4	37	0.7129	0.0128	
5	45	0.7266	-0.0483	
6	55	0.3845	0.0254	
7	71	0.3687	0.0108	
8	78	0.0414	-0.3041	
9	79	0.0414	-0.3041	

Optimization started 2022-Nov-30, 14:44:05

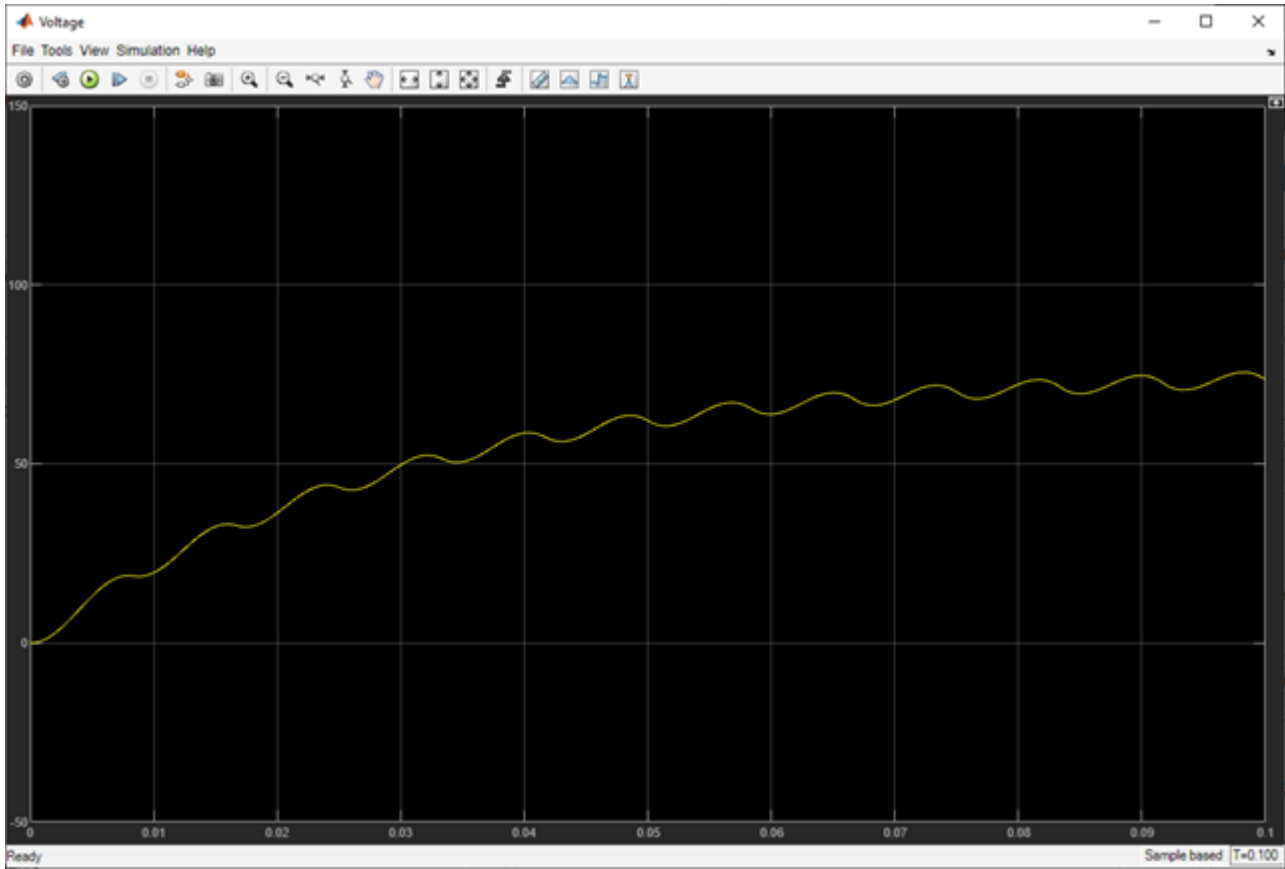
Optimization converged, 30-Nov-2022 14:45:30

Optimized variable values written to 'DesignVars' in the Design Optimization workspace
'sdorectifier' updated with optimized values
Optimized requirement values written to 'ReqValues' in the Design Optimization workspace

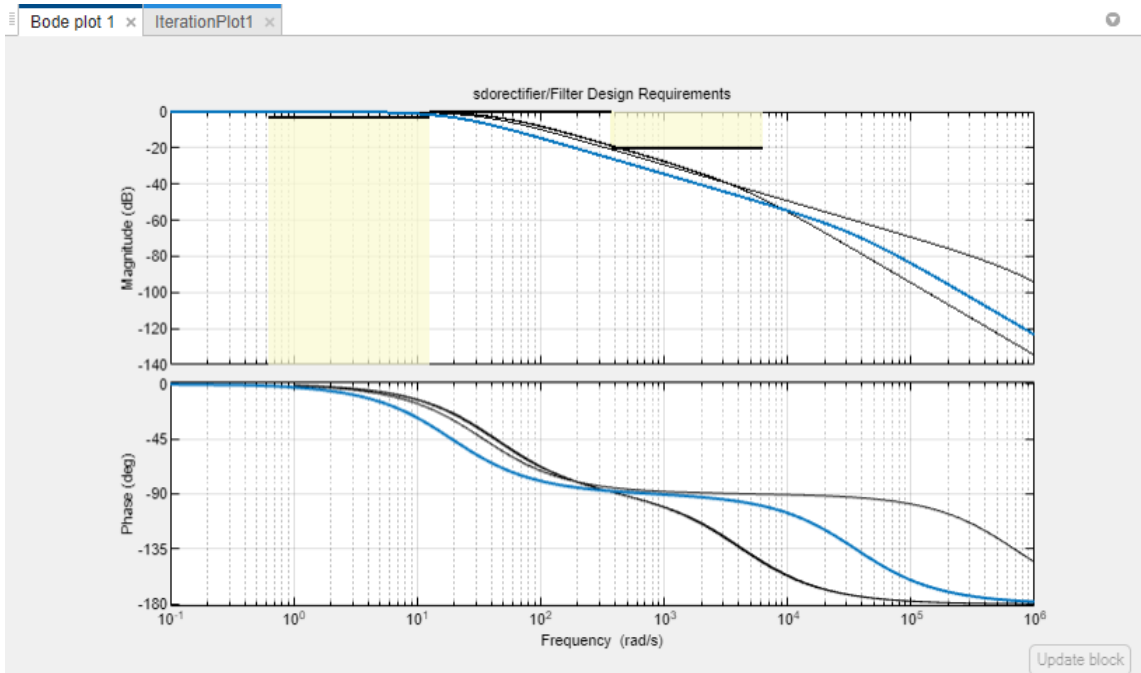
Optimization solver output:

Save Iteration... Display Options... Optimize

The harmonic content in the filter voltage signal is reduced from the initial design.



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



The plot displays the output of the last five iterations. The final response using the optimized parameter values appears as the thick blue 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 the **Data** area and view the updated values in the **Variable Preview** area.

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

See Also

Check Bode Characteristics

Related Examples

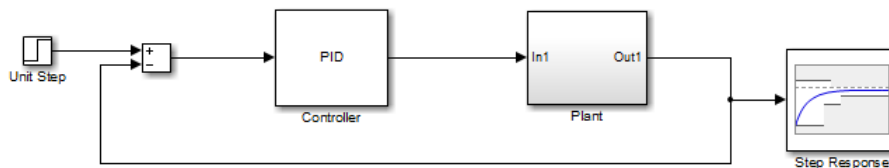
- “Design Optimization to Meet Frequency-Domain Requirements (GUI)”

Time-Domain Model Verification

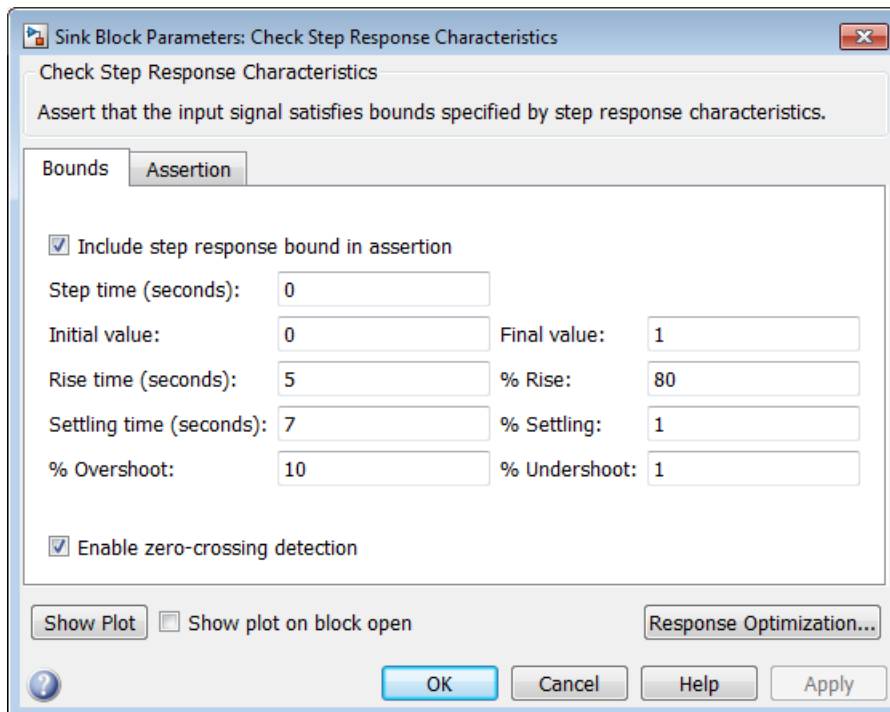
This example shows how to perform time-domain model verification using Simulink Design Optimization **Model Verification** blocks. During time-domain verification, the software monitors a signal to check if it meets time-domain characteristics such as step response characteristics and upper and lower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

- 1 For example, consider the following Simulink model.



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



- 2 When you simulate such a model and the block asserts multiple times during simulation that 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 3-3.

Optimization-Based Linear Control Design

- “When to Use Optimization-Based Linear Control Design” on page 4-2
- “Types of Time- and Frequency-Domain Design Requirements for Optimization-Based Control Design” on page 4-3
- “Design Optimization-Based PID Controller for Linearized Simulink Model (GUI)” on page 4-4

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 **Control System Designer** app. 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.

See Also

Related Examples

- “Optimize LTI System to Meet Frequency-Domain Requirements”
- “Design Optimization-Based PID Controller for Linearized Simulink Model (GUI)” on page 4-4

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 Control System Designer App”.

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.

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

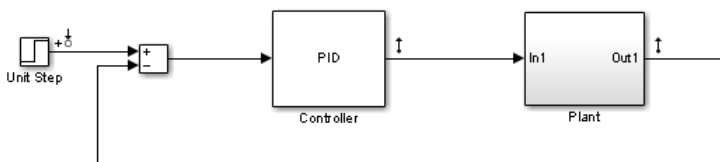
This example shows how to perform optimization-based control design in the **Control System Designer** app when you have Control System Toolbox software. You design a PID controller for a linearized Simulink model.

You accomplish the following tasks:

- 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 output signal.

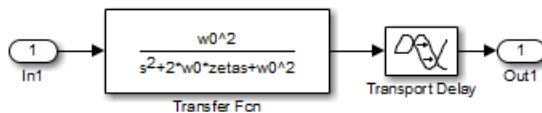
Model Structure

The Simulink model, `sldo_model2`, contains a `Controller` block, which is a PID Controller. This block controls the output of the `Plant` subsystem.



Using the Simulink Control Design software, the model has been linearized at the operating point specified in the model. The `sldo_model2.mat` file contains a preconfigured **Control System Designer** app session, saved after linearizing the model. To learn more about linearizing Simulink models for control design, see “Control System Design and Tuning” (Simulink Control Design).

The `Plant` subsystem is modeled as a second-order system with delay. It contains Transfer Function and Transport Delay blocks.



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

Design Requirements

The compensator you design 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]

PID and Plant Model

Open a **Control System Designer** app session for the model.

```
controlSystemDesigner('sldo_model2.mat')
```

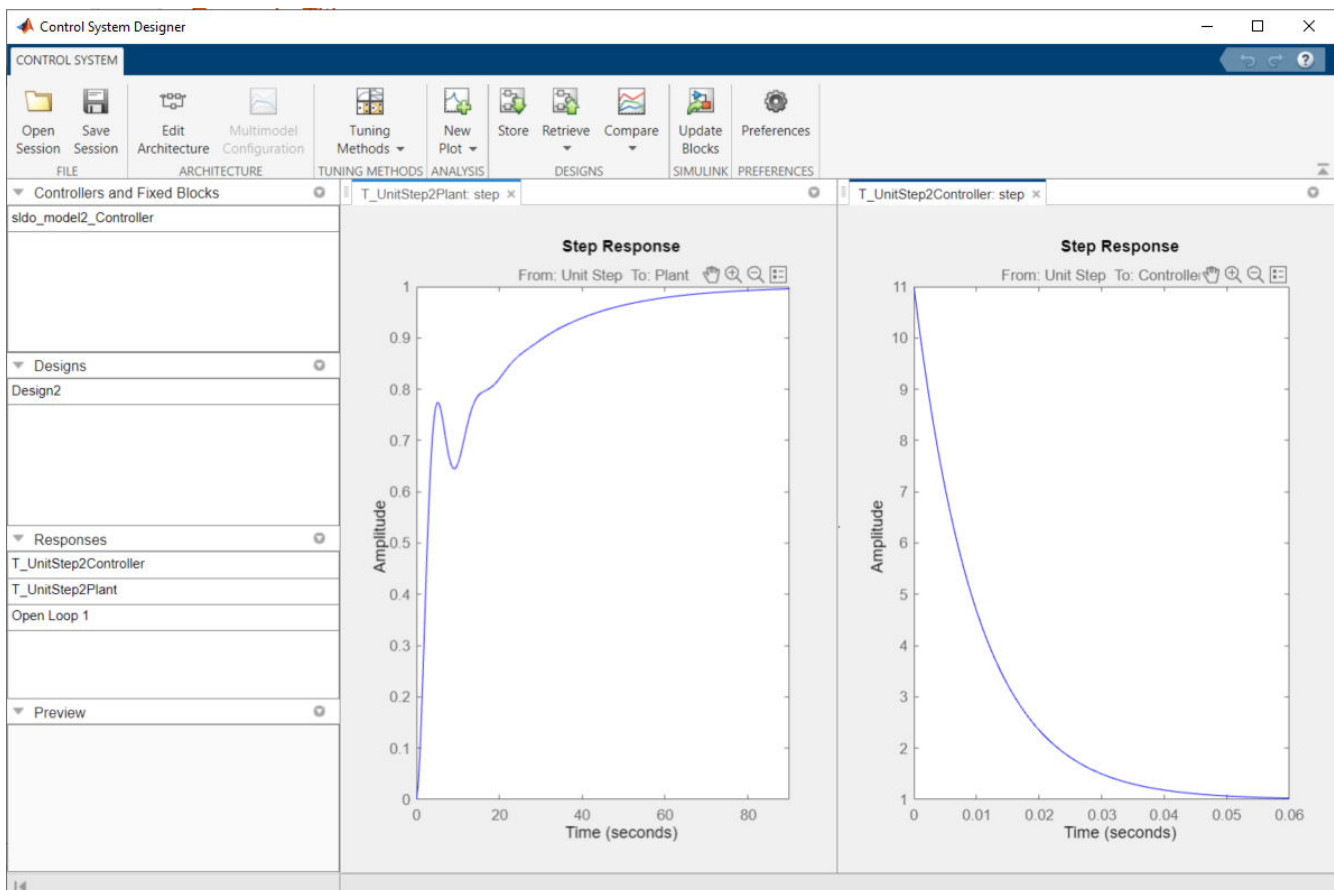
Configure the Control System Designer App for Optimization-Based Control Design

To design a linear controller for a Simulink model, first configure a **Control System Designer** app session.

- 1 `sldo_model2.mat` file contains a preconfigured **Control System Designer** app session. This session was saved after Simulink Control Design software linearized `sldo_model2`.

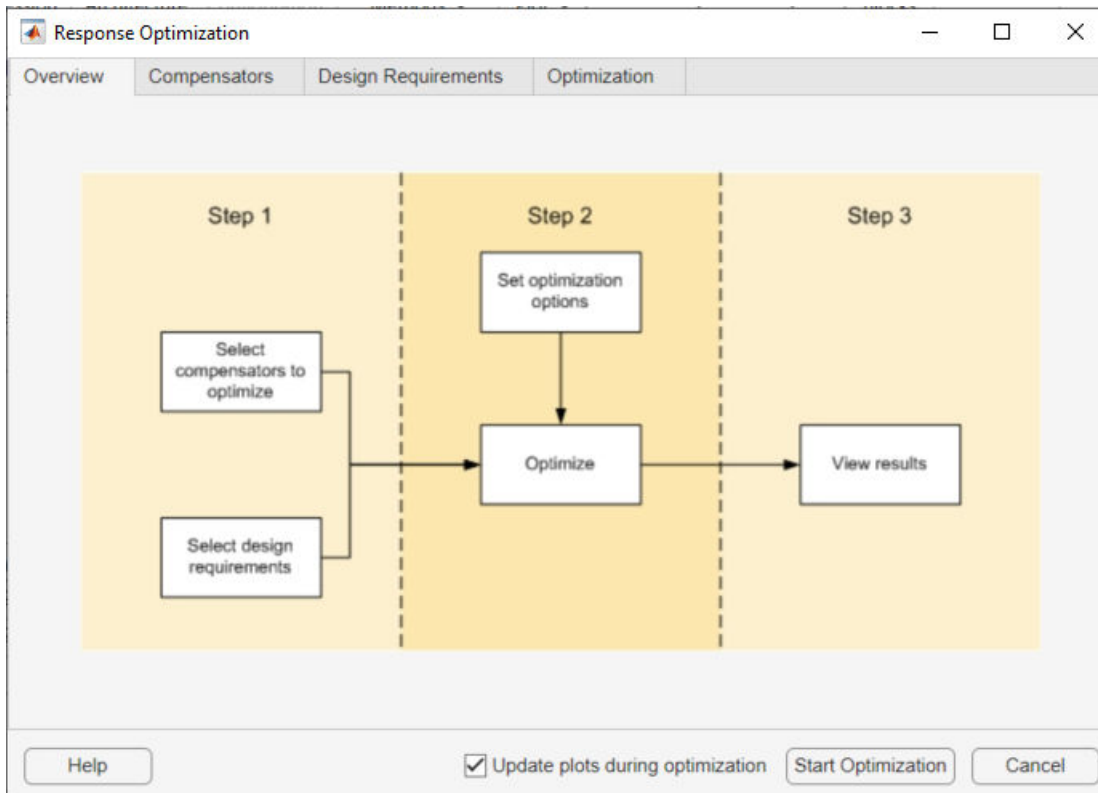
The **Control System Designer** app opens with the following plots:

- Closed-loop step response of the system
- Output of the Controller block



- 2 To perform response optimization, in the **Tuning Methods** drop-down list, select **Optimization Based Tuning**.

In the Response Optimization window, you can specify controller parameters and design requirements, and perform optimization.



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

Specify the Controller Parameters

To specify the controller parameters that are to be optimized:

- 1 In the Response Optimization window, select the **Compensators** tab.

The screenshot shows the 'Response Optimization' window with the 'Compensators' tab selected. A table displays the parameter settings for the 'sldo_model2_Controller'.

<input type="checkbox"/>	Compensator Ele...	Value	Initial Gu...	Minimum	Maximum	Typical V...
<input type="checkbox"/>	Gain	0.1	0.1	-Inf	Inf	0.1
<input type="checkbox"/>	Real Zero	-8.999	-8.999	-Inf	Inf	-8.999
<input type="checkbox"/>	Real Zero	-0.10102	-0.10102	-Inf	Inf	-0.10102
<input type="checkbox"/>	Real Pole	-100	-100	-Inf	Inf	-100

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.

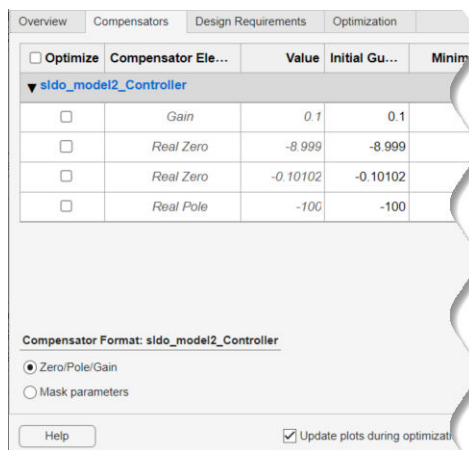
The controller parameters appear as poles and zeros in the **Compensator elements** column:

- 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

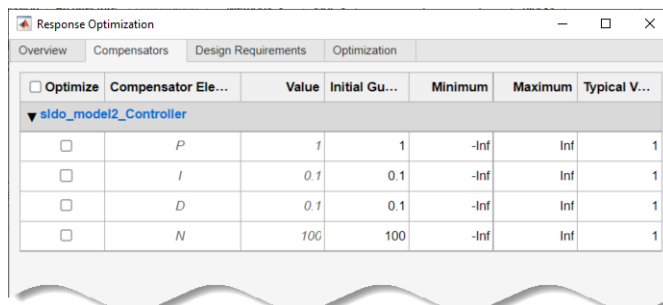
Tip To view the structure of the Controller block, right-click the block in the model, and select **Mask > Look Under Mask**.

- 2 Change the PID controller parameters to Simulink block mask parameters format.

Click the **sldo_model2/Controller** row, and select **Mask parameters** as **Compensator Format**.

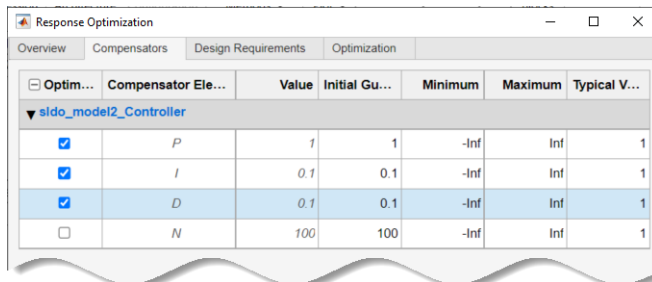


The controller parameters now display as Simulink block mask parameters, P, I, and D. For more information, see “Design Linear Controllers for Simulink Models”. To learn more about mask parameters, see “Mask Parameters”.



- 3 Select the controller parameters to optimize.

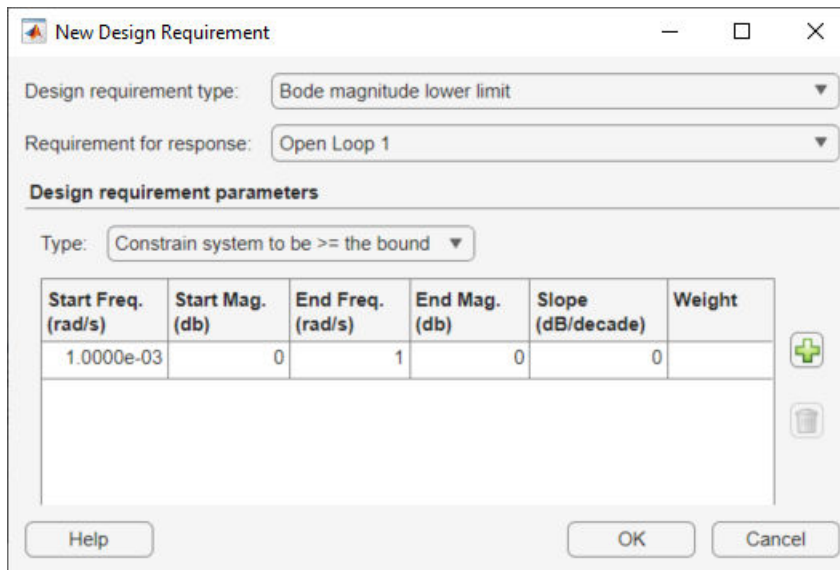
In the **Optimize** column, select P, I, and D.



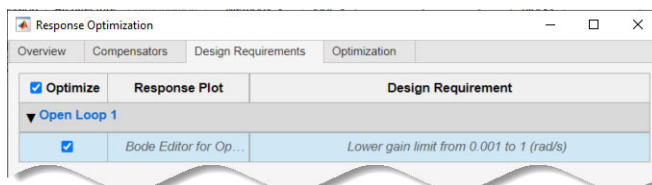
Specify Bode Magnitude and Phase Margin Design Requirements

Specify the Bode magnitude lower limit requirement:

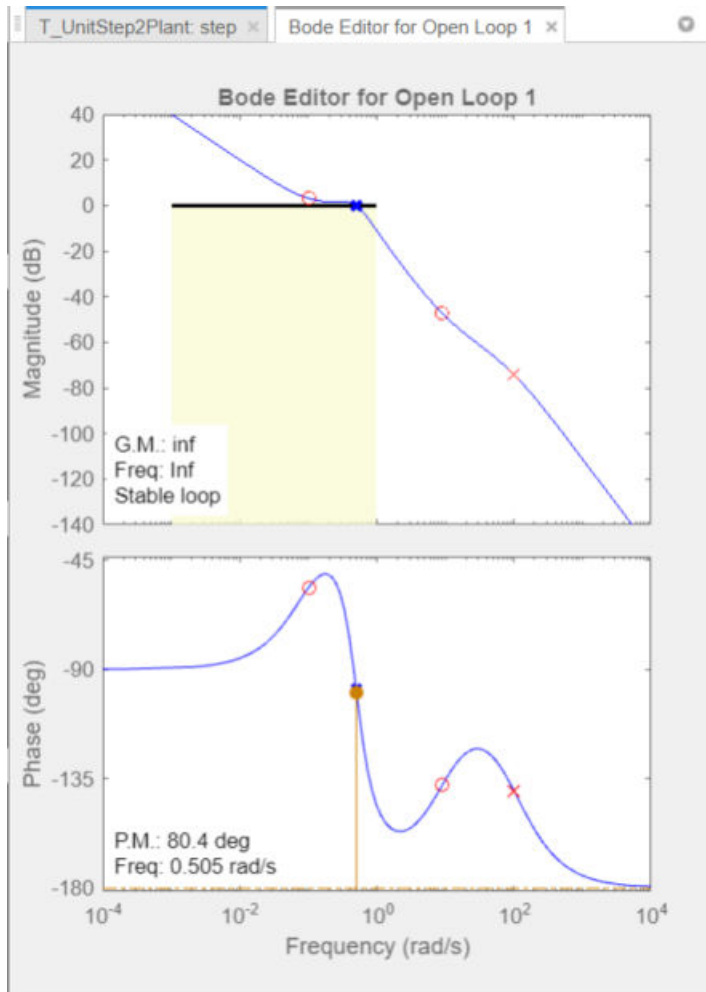
- 1 In the **Design requirements** tab, click **Add new design requirement**. A New Design Requirement dialog box opens.
- 2 In the New Design Requirement dialog box, in the **Design requirement type** drop-down list, select Bode magnitude lower limit.
- 3 In the **Requirement for response** drop-down list, select Open Loop 1.
- 4 Specify the **Frequency** range as $1e-3$ to 1 .
- 5 Specify the **Magnitude** range as 0 to 0 .
- 6 Click **OK**.



The Bode lower magnitude limit is added to the **Design requirements** tab.

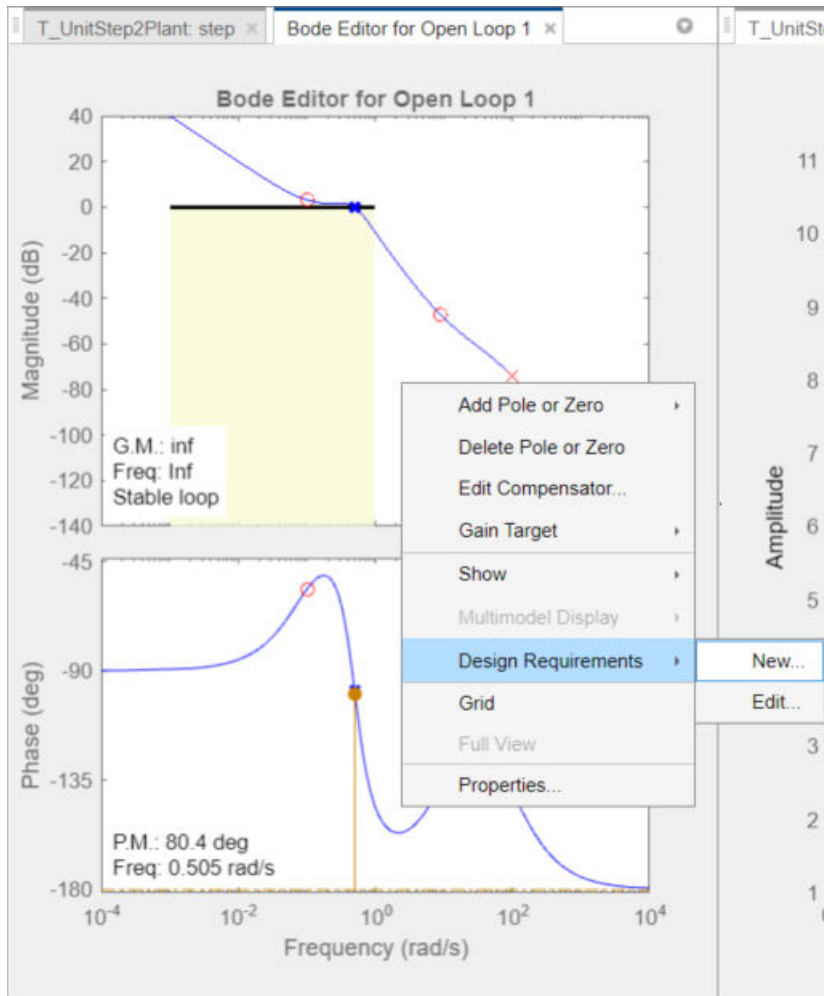


The **Control System Designer** app window updates to show the Bode plot in a **Bode Editor**. The design requirement is displayed as the black line segment.

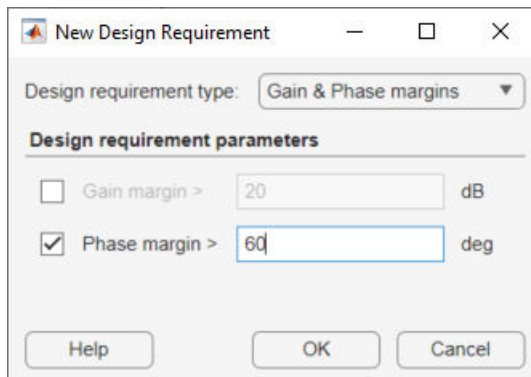


Specify the phase margin requirement:

- 1 Right-click within the white space of the Bode plot, and select **Design Requirements** > **New** to open the New Design Requirement dialog box.

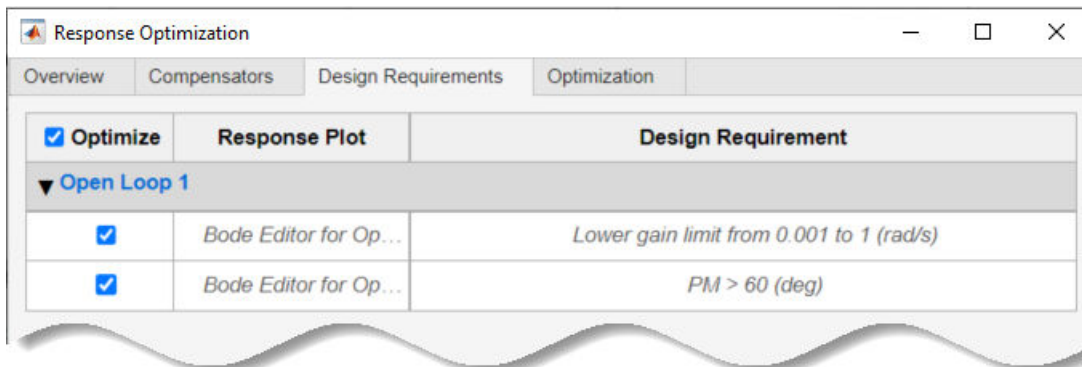


- 2 In the New Design Requirement dialog box, in the **Design requirement type** drop-down list, select Gain & phase margins.
- 3 Select the **Phase margin** check box, and specify the phase margin as 60.

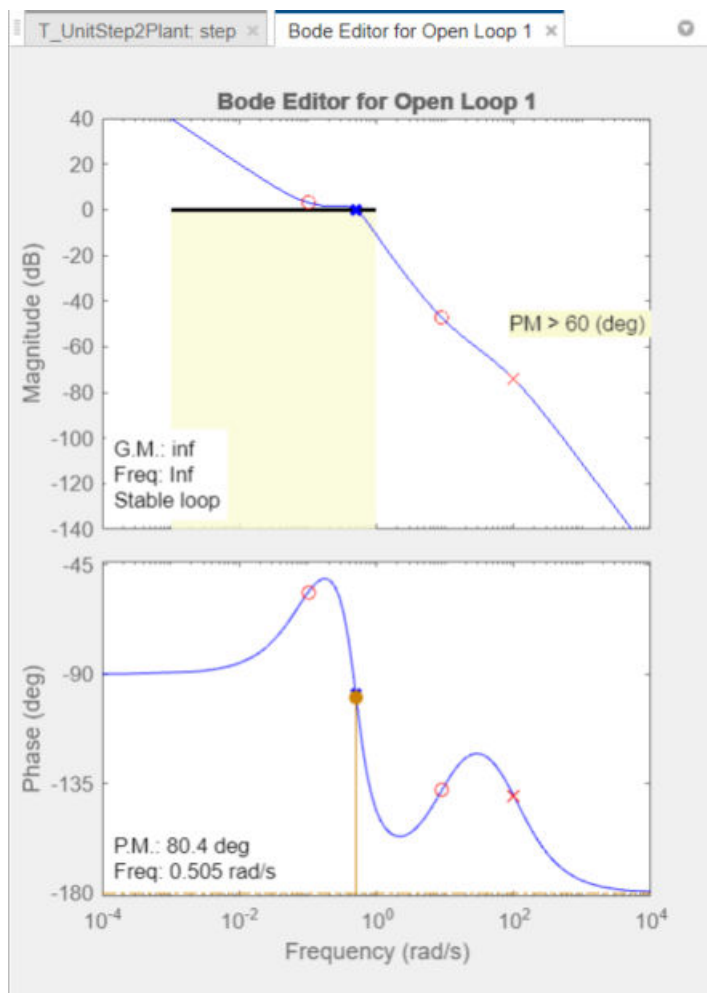


- 4 Click **OK**.

In the Response Optimization window, the **Design requirements** tab updates to display the phase margin requirement.



In the app, in the Bode Editor, the plot updates to display the phase margin requirement.



Design the Controller

To design the controller with specified design requirements:

- 1 In the Response Optimization window, in the **Optimization** tab, click **Start Optimization**.

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 available optimization methods, click **Optimization Options**, and then click **Help** in the Options dialog box.

After the optimization completes, the **Optimization** tab displays the optimization iterations and status.

The screenshot shows the 'Response Optimization' dialog box with the 'Optimization' tab selected. The dialog contains a table with the following data:

Iteration	Eval-Count	Cost Function	Constraint Violation	Step Size	Procedure
0	7	0	57.1200		
1	14	0	1.7770	8.3600	
2	21	0	1.6970	12.1000	Hessian mod
3	28	0	0	21.7000	

Below the table is a status message box containing the following text:

```
Constructing optimization problem...
Optimization started 26-Jan-2023 12:51:14
Optimization finished 26-Jan-2023 12:51:39
Successful termination.
Found a feasible or optimal solution within the specified tolerances.
```

At the bottom of the dialog, there are several buttons and options:

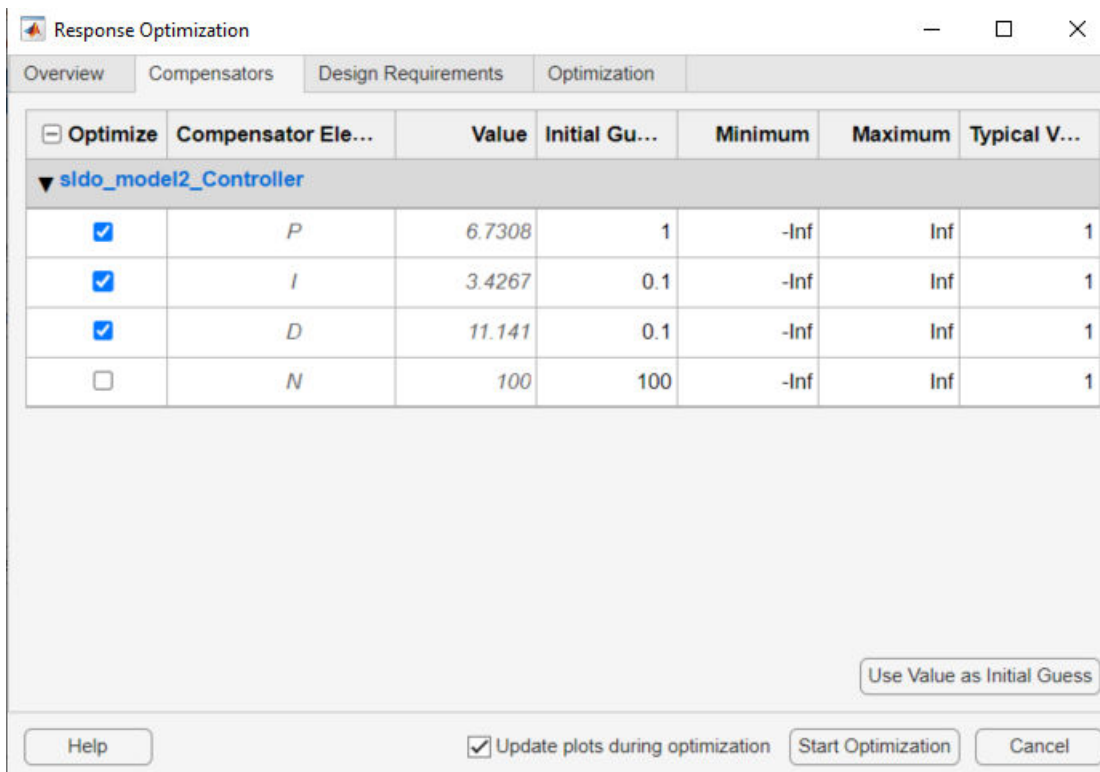
- Help** button
- Update plots during optimization**
- Start Optimization** button
- Cancel** button

On the right side of the dialog, there is an 'Optimization Options' section with a 'Display Options' sub-section containing the following checked items:

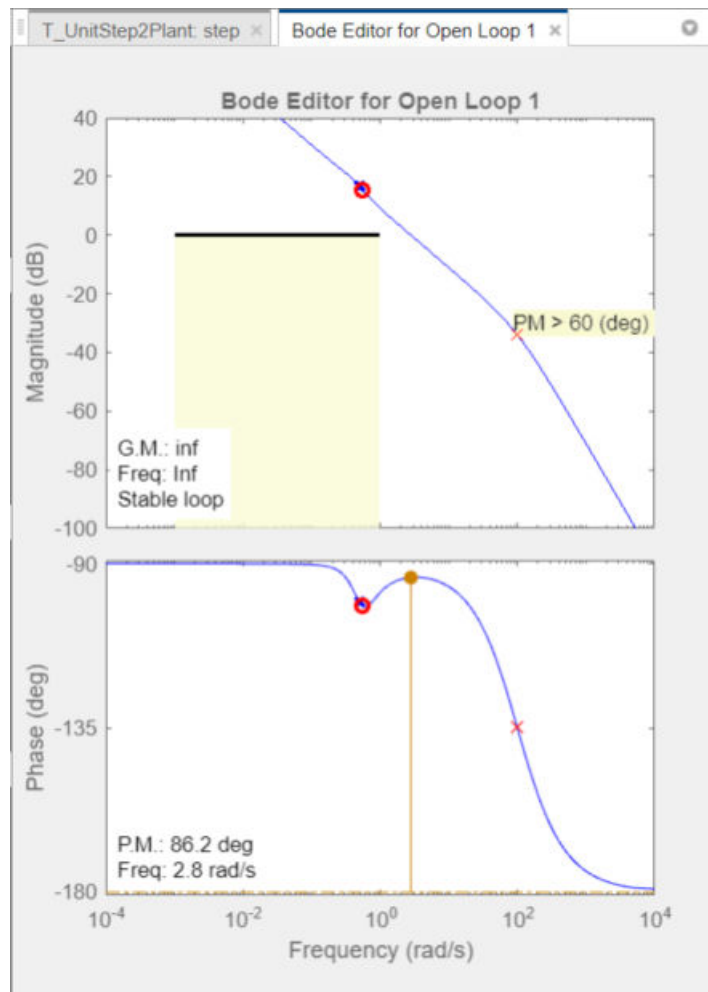
- Iteration
- Eval-Count
- Cost Function
- Constraint Violation
- Step Size
- Procedure

The status 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".

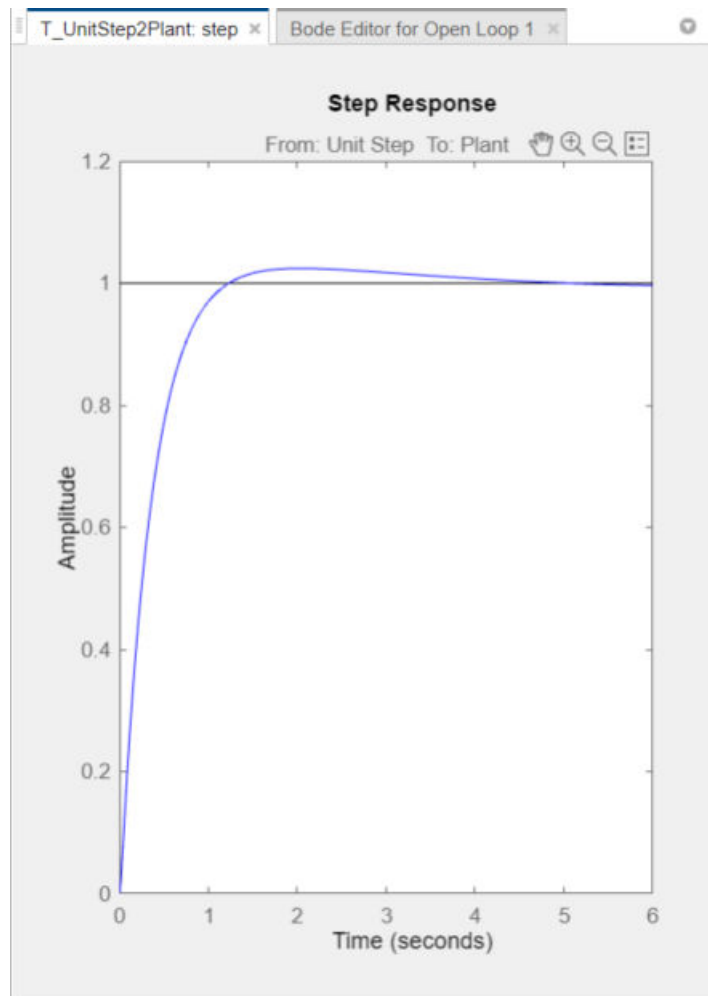
- 2 Examine the controller parameters and the system response:
 - a In the **Compensator** tab, view the optimized parameter values in the **Value** column.



- b** Examine the system response on the following plots:
- The Bode plot:

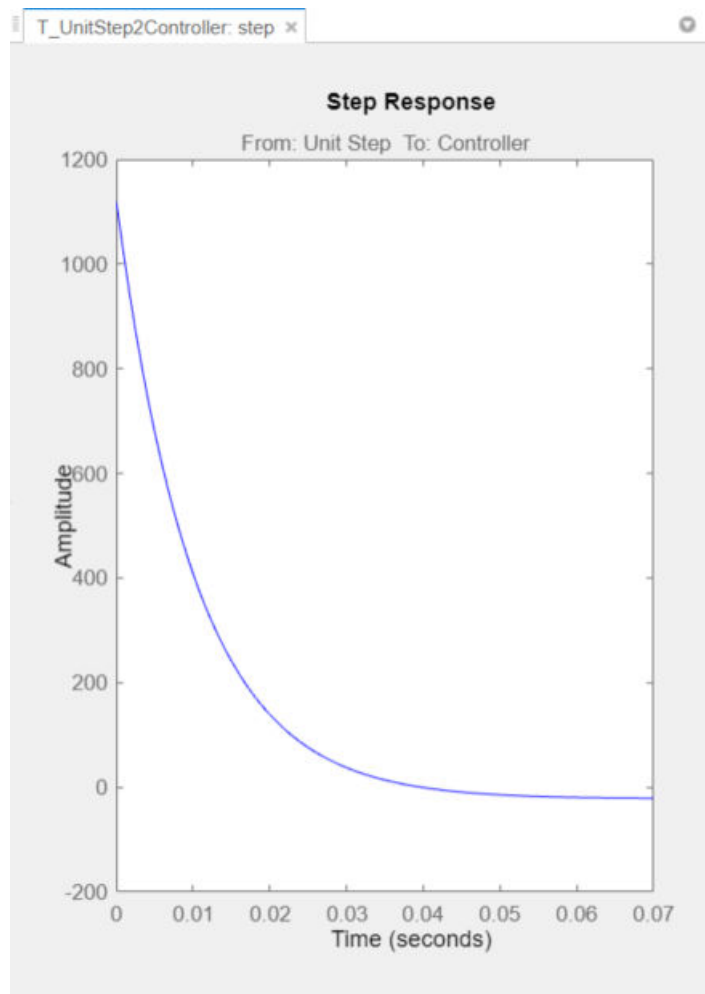


- The magnitude of the system, displayed as the blue curve in the top plot, lies outside the yellow region. This indicates that the system has met the Bode magnitude requirement.
- The phase plot displays the phase margin (P.M.) value of 86.1 degrees. This indicates that the system has met the phase margin design requirement of greater than 60 degrees.
- Closed-loop step response of the system:



The 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.

- Output of the Controller block:

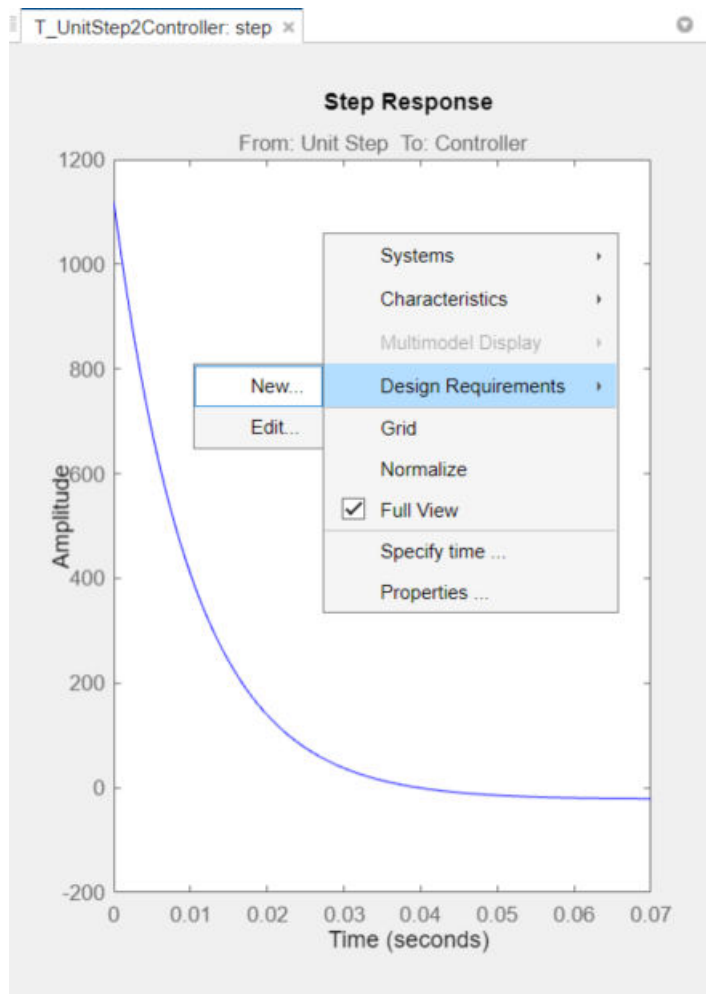


The plot shows that the peak value of the controller output is about 1100, which is large and can damage the plant. To limit the controller output, apply lower and upper bounds on the signal, as specified in “Design Requirements” on page 4-4.

Refine the Controller Design to Meet Controller Output Bounds

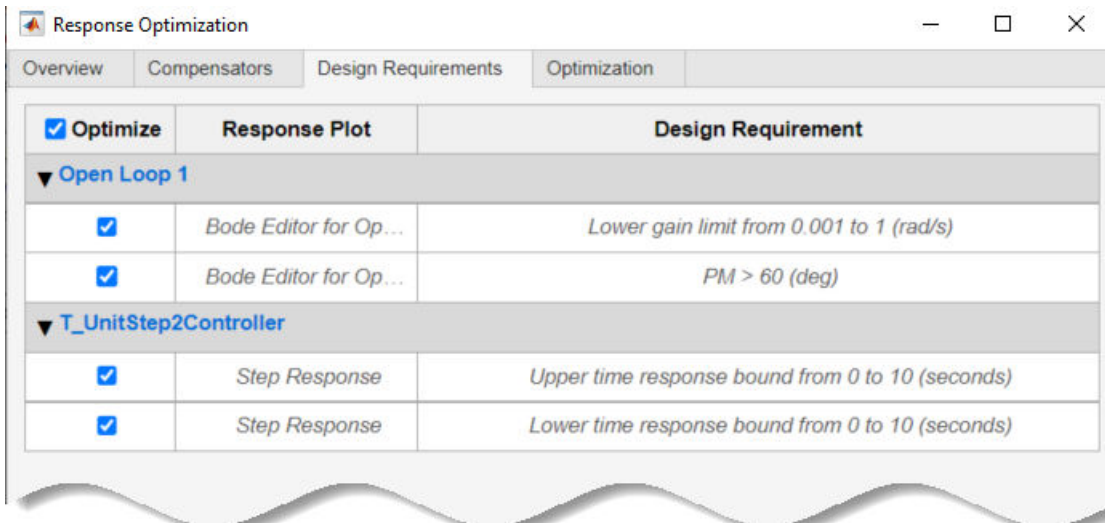
To tune the compensator parameters to meet the bounds on the controller output:

- 1 Add an upper-bound on the controller output:
 - a In the controller output plot, right-click the white area, and select **Design requirement** > **New**.

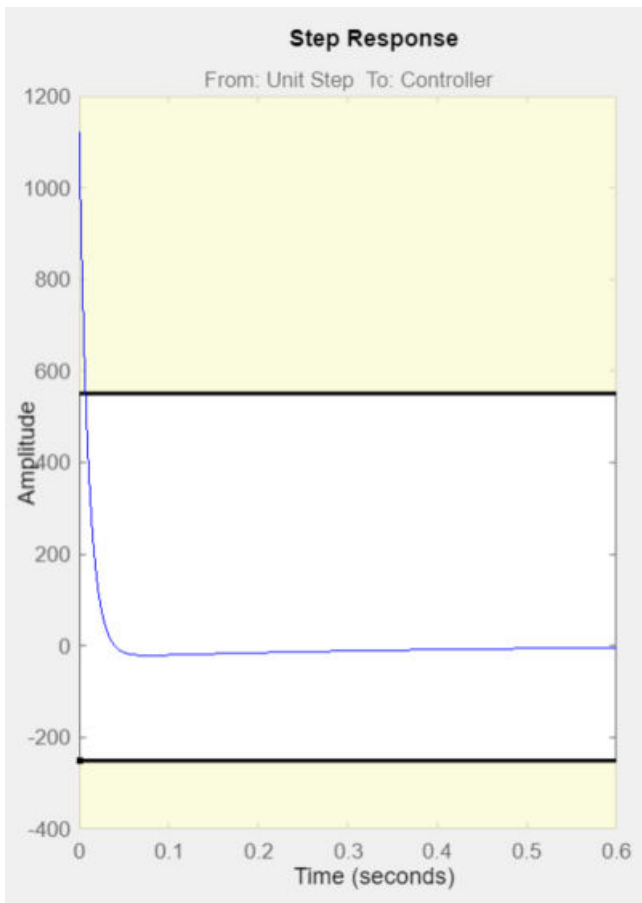


- b In the New Design Requirement dialog box, in the **Design requirement type** drop-down list, select Upper time response bound.
 - c Specify the **Time** range as 0 to Inf.
 - d Specify the **Amplitude** range as 550 to 550.
 - e Click **OK**.
 - 2 Add a lower-bound on the controller output:
 - a In controller output plot, right-click the white area, and select **Design requirement > New**.
 - b In the New Design Requirement dialog box, in the **Design requirement type** drop-down list, select Lower time response bound.
 - c Specify the **Time** range as 0 to Inf.
 - d Specify the **Amplitude** range as -250 to -250.
 - e Click **OK**.

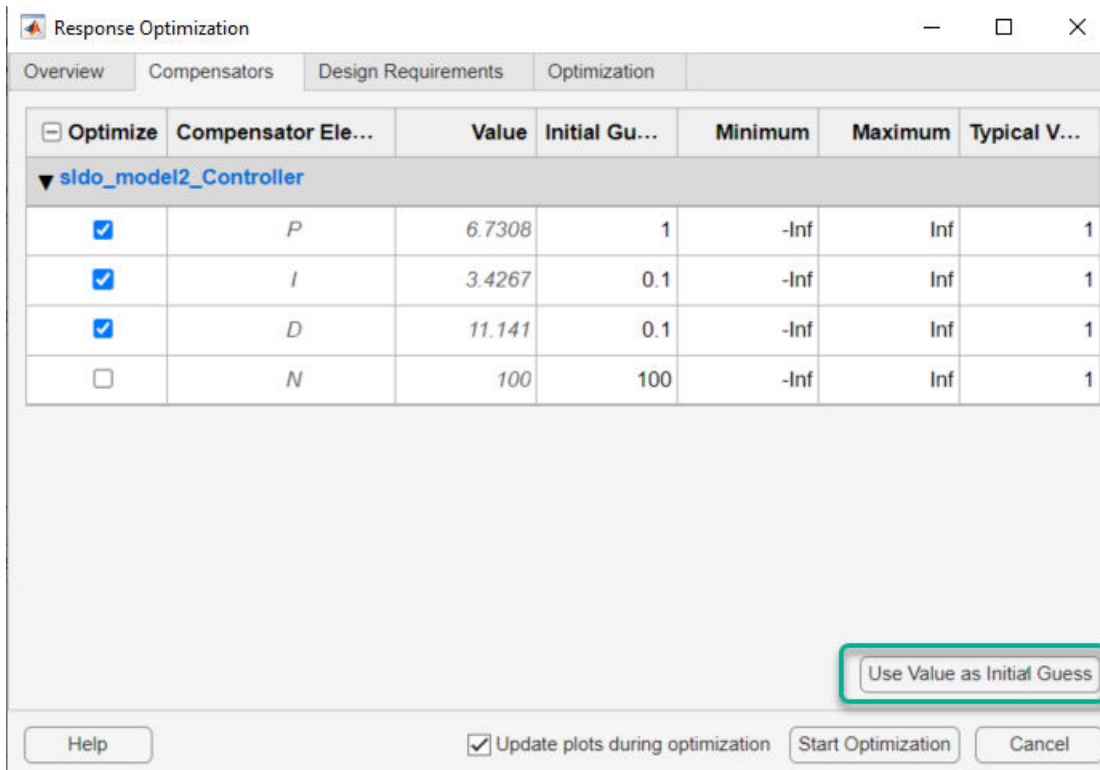
In the Response Optimization window, the **Design requirements** tab updates to display the bounds on the controller output.



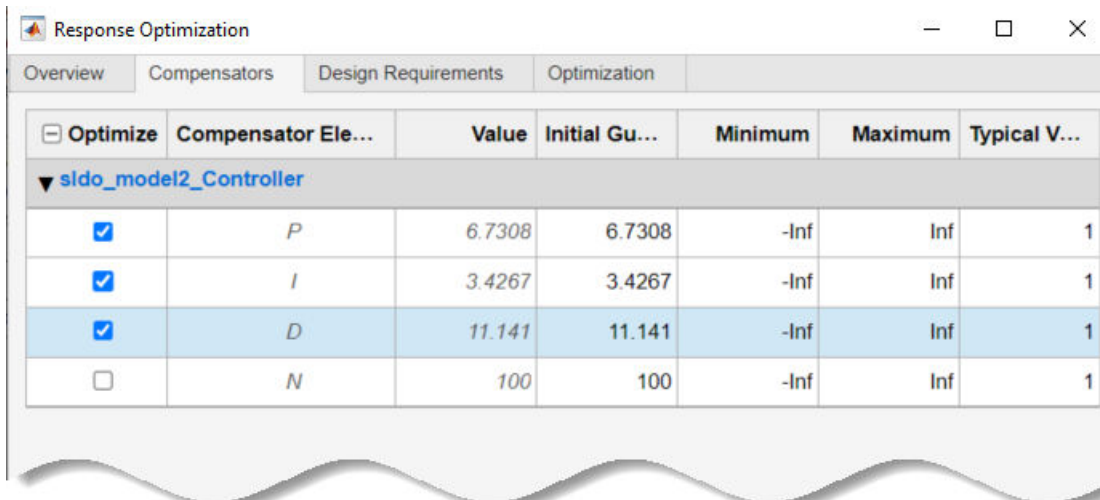
The plot of the output of the Controller block displays the new design requirements.



- 3 Optimize the parameters to meet the design requirements on the controller output:
 - a In the Response Optimization window, in the **Compensators** tab, select the rows containing P, I, and D, and click **Use Value as Initial Guess**.



The values in the **Initial Guess** column update. When you run the optimization again, the optimization method uses the updated parameter values as the starting point for refining the values.

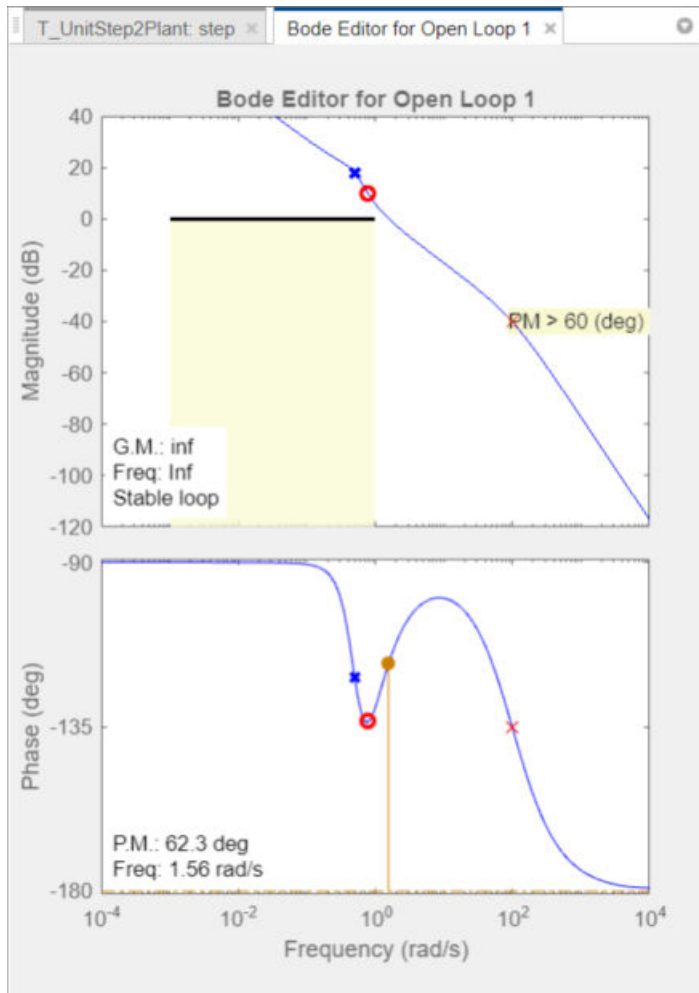


- b** In the **Optimization** tab, click **Start Optimization**. At every optimization iteration, the optimization method reduces the distance between the current response and the upper and lower bounds on the signal. After the optimization completes, the **Optimization** tab displays the optimization iterations and status.

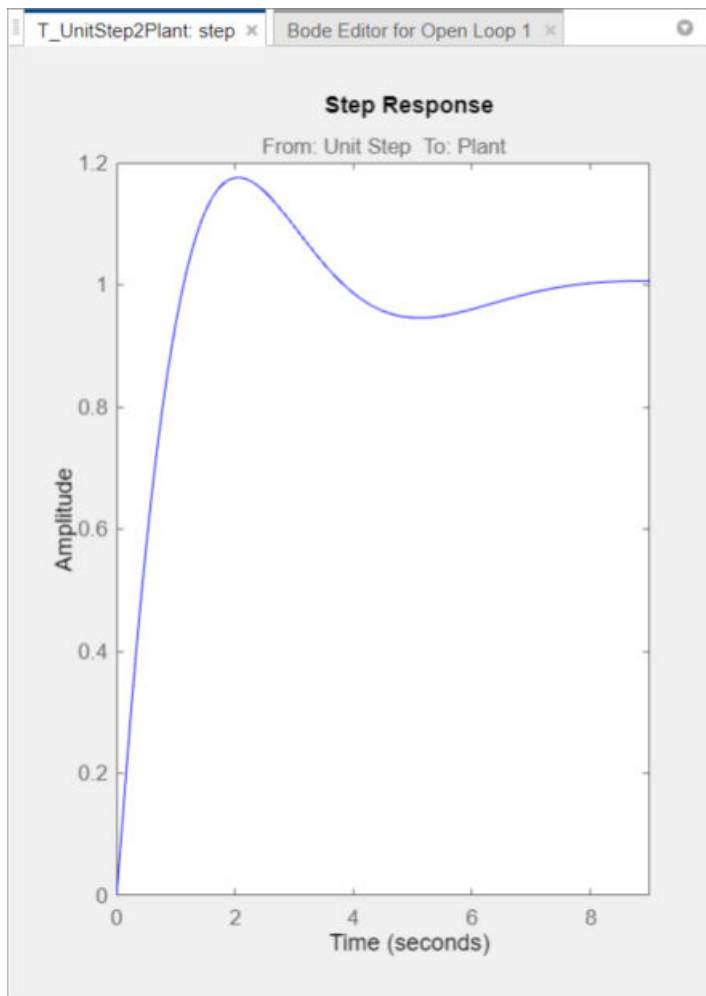
The status message, Successful termination, indicates that the optimization method found a solution that meets the design requirements.

4 Examine the response plots.

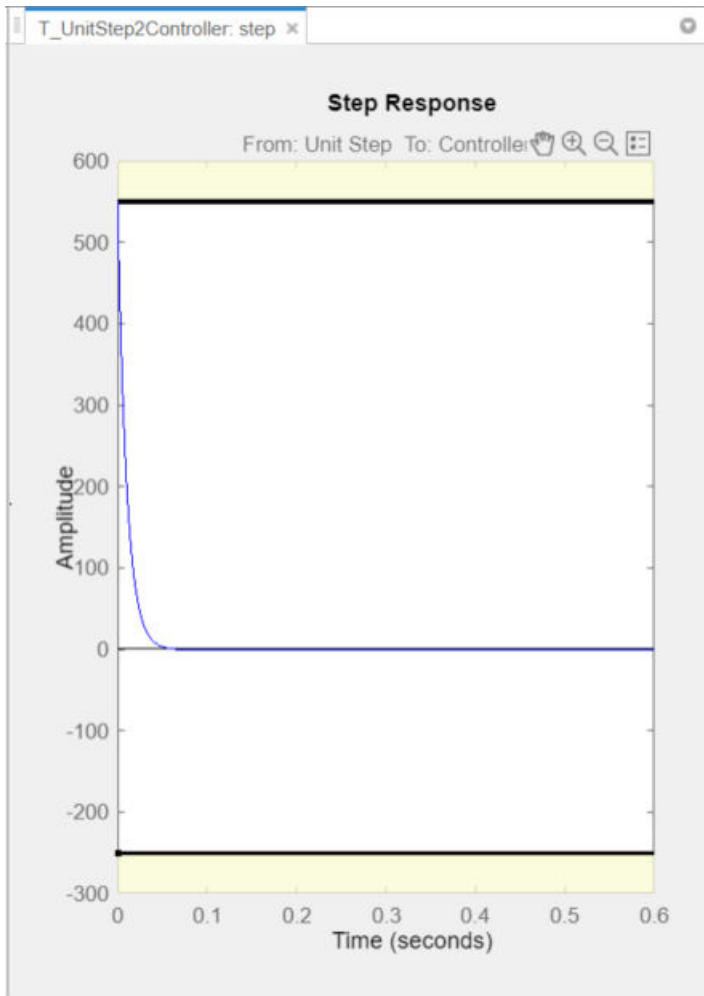
The Bode plots show that after refining the design, the system continues to meet the magnitude and phase margin requirements specified in “Design Requirements” on page 4-4.



Verify that the closed-loop response of the system remains stable after refining the controller design.



The plot of the output of the Controller block shows that the output lies between 550 and -250, and thus meets the design requirement on the bounds of the controller output.



5 Examine the parameter values of the optimized controller.

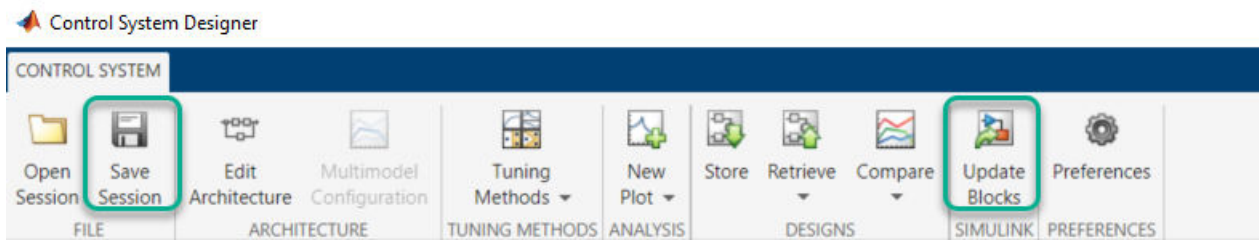
In the Response Optimization window, in the **Compensators** tab, view the optimized controller parameter values in the **Value** column.

The screenshot shows the "Response Optimization" window with the "Compensators" tab selected. The table below displays the optimized parameter values for the "slido_model2_Controller".

Optimize	Compensator Ele...	Value	Initial Gu...	Minimum	Maximum	Typical V...
<input checked="" type="checkbox"/>	<i>P</i>	6.6738	6.7308	-Inf	Inf	1
<input checked="" type="checkbox"/>	<i>I</i>	3.4267	3.4267	-Inf	Inf	1
<input checked="" type="checkbox"/>	<i>D</i>	5.4333	11.141	-Inf	Inf	1
<input type="checkbox"/>	<i>N</i>	100	100	-Inf	Inf	1

6 Write the optimized controller parameter values to the Controller block in the Simulink model.

In the **Control System Designer** app, click **Update Blocks**.



7 Save a session with the optimized controller parameters.

In the **Control System Designer** app, select **Save Session**, and specify a name for the session.

See Also

More About

- “Design Linear Controllers for Simulink Models”
- “Optimize LTI System to Meet Frequency-Domain Requirements”

