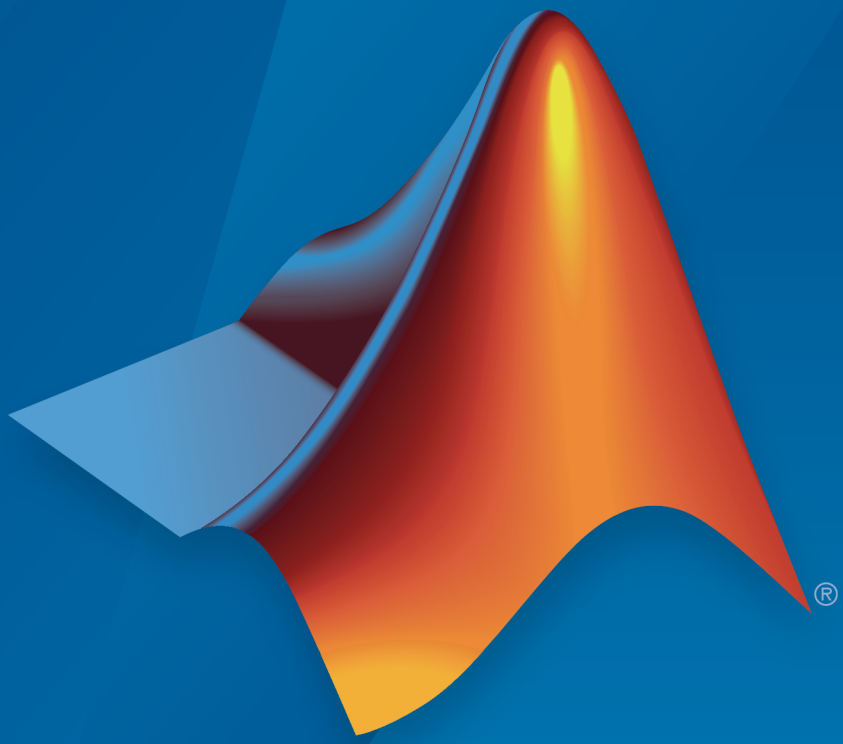
Simulink® Design Optimization™ Release Notes



MATLAB® & SIMULINK®

MathWorks®

How to Contact MathWorks

Latest news: www.mathworks.com

Sales and services: www.mathworks.com/sales_and_services

User community: www.mathworks.com/matlabcentral

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

Phone: 508-647-7000

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

Simulink® Design Optimization™ Release Notes

Contents

R2017a

R2016b

R2016a

R2015b

R2015a

R2014b

R2014a

R2013b

R2013a

R2012b

R2012a

R2011b

R2011a

Bug Fixes

R2010b

R2010a

R2009b

R2009a

R2017a

Lookup Tables: Tune and analyze lookup table data in the Response Optimization and Sensitivity Analysis tools

You can now specify additional design requirements in the Response Optimization tool and Sensitivity Analysis tool. These requirements can be imposed on any variable or signal in your model. Previously, you could not impose requirements on variables in the tools, these requirements were available at the command line only.

- **Monotonic Variable** — Impose a monotonic constraint on a variable. For example, constrain a variable to be monotonically increasing. You can use this requirement to tune the breakpoint values of a lookup table.
- **Smoothness Constraint** — Impose bounds on the gradient magnitude of a variable. For example, use this requirement on the table data of a lookup table.
- **Relational Constraint** — Impose a relational constraint between two variables. For example, require that variable a is always greater than b.
- **Vector Property** — Impose constraints on a specified property of a vector variable. For example, constrain the mean value of the variable.
- **Ellipse Region Constraint** — Impose an elliptic bound on the phase plane trajectory of two signals in your model. The phase plane trajectory is a plot of the two signals against each other.

For an example, see "Design Optimization Using Lookup Table Requirements for Gain Scheduling (GUI)".

`Method` Property of `sdo.requirements.PhasePlaneEllipse`: Specify method for evaluating a phase plane ellipse requirement

The `sdo.requirements.PhasePlaneEllipse` object has a new `Method` property. You can use this property to specify the method that `evalRequirement` uses for evaluating an elliptic bound requirement on the phase plane trajectory of two signals in your model. You specify `Method` as one of the following values:

- `'Maximum'` — `evalRequirement` command returns a scalar value that is the maximum signed distance between the phase plane trajectory and the bounding ellipse. `'Maximum'` is the default value.
- `'Residuals'` — `evalRequirement` command returns a vector with the signed distance between the ellipse and all the trajectory points.

Previously, `evalRequirement` always returned the signed distance between the ellipse and all trajectory points.

For an example that specifies the new `Method` property, see "Evaluate Elliptical Bound on Phase Plane Trajectory".

Compatibility Considerations

If you evaluate an `sdo.requirements.PhasePlaneEllipse` requirement object, the `evalRequirement` command now returns a scalar with the maximum signed distance between the phase plane trajectory and the bounding ellipse. To see the signed distance for all trajectory points, specify the `Method` property of the requirement as `'Residuals'`, and then evaluate the requirement.

If you load a requirement object that is saved in a `.mat` file, the software automatically sets the `Method` property to `'Residuals'` and there is no compatibility issue.

Specify Latin hypercube sampling method without Statistics and Machine Learning Toolbox license

You can now specify the Latin hypercube sampling method for generating parameter samples for sensitivity analysis without requiring a Statistics and Machine Learning Toolbox™ license. You specify the method in the Sensitivity Analysis tool or at the command line using `sdo.SampleOptions`.

For information about Latin hypercube sampling, see "Generate Random Parameter Values".

Functionality Being Removed or Changed

Functionality	Result	Use This Instead	Compatibility Considerations
Evaluating `sdo.requirements.Pha` requirement object.	Still works	Not Applicable	The default output of `evalRequirement` method of `sdo.requirements.PhasePlaneEllipse` has changed. To recover the results from previous releases, update your scripts as described in

Functionality	Result	Use This Instead	Compatibility Considerations
			"`Method` Property of `sdo.requirements.PhasePlaneEllipse` Specify method for evaluating a phase plane ellipse requirement" on page 1-2.

R2016b

Version: 3.1

New Features

Bug Fixes

Command-Line Requirements for Lookup Table Tuning: Impose constraints on table data, and apply pairwise signal bounds to lookup table inputs

You can now specify additional requirements when performing parameter estimation, response optimization, or sensitivity analysis at the command line. To specify the new requirements, use the corresponding requirement objects.

Requirement	Description	Requirement Object
Monotonic constraint	Impose a monotonic constraint on a variable. For example, constrain a variable to be monotonically increasing.	sdo.requirements.MonotonicVariable
Smoothness constraint	Impose bounds on the gradient magnitude of a variable.	sdo.requirements.SmoothnessConstraint
Relational constraint	Impose a relational constraint between two variables. For example, require that variable *a* is always greater than *b*.	sdo.requirements.RelationalConstraint
Function matching constraint	Impose a linear or quadratic function matching constraint on the values of a variable.	sdo.requirements.FunctionMatching
Phase plane ellipse constraint	Impose an elliptic bound on the phase plane trajectory of two signals in your model. The phase plane trajectory is a plot of the two signals against each other.	sdo.requirements.PhasePlaneEllipse
Phase plane region constraint	Impose a piecewise-linear bound on the phase plane trajectory of two signals.	sdo.requirements.PhasePlaneRegion

After creating the requirement, use the `evalRequirement` method of the requirement object to evaluate the cost and constraints associated with the requirement. A positive output implies that the requirement has been violated.

The requirements can be imposed on any variable or signal in your model. If you have lookup tables in your model, you can impose the requirements on the lookup table inputs and data. For an example, see Design Optimization Using Lookup Table Requirements for Gain Scheduling (Code).

Generate Parameter Samples for Sensitivity Analysis: Specify additional probability distributions and sampling methods

You can now specify additional probability distributions and sampling methods for generating random parameter samples for sensitivity analysis.

- The new probability distributions available in the Sensitivity Analysis tool and at the command line are multinomial, piecewise linear, and triangular distributions. Previously, these distributions required Statistics and Machine Learning Toolbox software. At the command line, use makedist to specify the probability distributions.
- At the command line, you can now also generate truncated probability distributions using the truncate command. Previously, you required the Statistics and Machine Learning Toolbox software to generate truncated distributions.
- The new sampling methods available in the tool and at the command line are the Sobol and Halton quasirandom methods. These methods require Statistics and Machine Learning Toolbox software. At the command line, use sdo.SampleOptions to specify the methods.

R2016a

Sensitivity Analysis Tool: Determine the most influential Simulink model parameters using design of experiments, Monte Carlo simulations, and correlation analysis

You can now use the Sensitivity Analysis tool to understand how the parameters and states of a Simulink® model influence the model output or model design requirements. Previously, sensitivity analysis was available only at the command line.

Using the tool, you can:

- Sample model parameters using design of experiments.
- Perform Monte Carlo simulations to evaluate your design requirement at selected parameter values.
- Analyze and visualize model sensitivity to parameters.

You can accelerate evaluation of design requirements using parallel computing and Simulink fast restart.

For examples of performing sensitivity analysis in the tool, see Identify Key Parameters for Estimation (GUI) and Design Exploration using Parameter Sampling (GUI).

Specification of `UseParallel` property of `sdo.OptimizeOptions` and `sdo.EvaluateOptions`

Specify the `UseParallel` property of sdo.OptimizeOptions and sdo.EvaluateOptions as either `true`, or `1`, or `false`, or `0`. The default value is `0`. Previously, you specified the property as `'always'` or `'never'`. The software now converts `'always'` and `'never'` to `1` and `0`, respectively.

Compatibility Considerations

If your code reads the value of `UseParallel`, modify it to expect `1` or `0` instead of `'always'` and `'never'`, respectively.

Functionality Being Removed or Changed

Functionality	Result	Use This Instead	Compatibility Considerations
Code that reads the `UseParallel` property	Error	Modify code to expect `1` or `0` instead of	For more information, see “Specification of

Functionality	Result	Use This Instead	Compatibility Considerations
of sdo.OptimizeOptions or sdo.EvaluateOptions.		`'always'` and `'never'` from the `UseParallel` property.	`UseParallel` property of `sdo.OptimizeOptions` and `sdo.EvaluateOptions`" on page 3-2
Code that specifies invalid `MethodOptions` for optimization solver specified in `Method` property of sdo.OptimizeOptions.	Error	Do not specify optimization solver options that are invalid for your specified optimization solver.	For details about the compatible optimization options, see the sdo.OptimizeOptions reference page.

R2015b

Design Optimization with Simulink Fast Restart: Speed up parameter estimation, response optimization, and sensitivity analysis tasks

You can now use the Fast Restart feature of Simulink to speed up optimization, estimation, and evaluation of tunable parameters of a model. You see a speedup of using fast restart in models that have a long compilation time. You can configure fast restart in the tool or at the command line. For more information, see:

- Speed Up Using Fast Restart Mode
- Improving Optimization Performance using Fast Restart (GUI)
- Improving Optimization Performance using Fast Restart (Code)

`SystemLoggingInfo` property of `sdo.SimulationTest` for specifying linear systems to log

For response optimization problems that include frequency-domain requirements, the model is linearized using Simulink Control Design™. At the command line, you can now use the new `SystemLoggingInfo` property of sdo.SimulationTest to specify linear systems to log. If you specify this property, the sim method of sdo.SimulationTest linearizes the model during simulation. You can use `SystemLoggingInfo` and sim instead of using the linearize command to compute the linear systems.

For an example of how to specify model linearization information at the command line, see Design Optimization to Meet Frequency-Domain Requirements (Code).

You can continue to use the linearize command to compute linear systems. However, to use fast restart, you must use the `SystemLoggingInfo` property and `sim` instead.

`info.Log` output of `sdo.evaluate` for storing additional evaluation information

If your cost function output includes a `Log` field, that information is now returned in the `Log` field of the `info` output of `sdo.evaluate`.

For more information, see the sdo.evaluate reference page.

R2015a

Version: 2.7

New Features

Bug Fixes

Improved workflow for setting up parameter estimation, response optimization, and sensitivity analysis on a distributed computing cluster

The workflow for specifying model dependencies for parallel computing has been streamlined:

- Simulink model variables that are in the MATLAB® base workspace, now get automatically copied to the remote workers. The change ensures that the model on the workers has access to the variables.

  You no longer need to either add the variables to the model workspace, or add a script to the model `PreloadFcn` or `PostloadFcn` callbacks.
- You can now specify file dependencies of your model in the tool and command line. The files get copied to the remote workers.

  - In the Parameter Estimation or Response Optimization tool:

    In the **Parallel Options** tab of the **Options** dialog box, select **Use the parallel pool during optimization**. This option checks for model dependencies and displays the file dependencies in the **Model file dependencies** tab. The file dependencies check may be incomplete. Enter any undetected file dependencies manually, or click **Add file dependency**, and select the file to add.
  - At the command line, use sdo.getModelDependencies to get the file and path dependencies of the model.

  For more information, see How to Use Parallel Computing for Parameter Estimation, How to Use Parallel Computing for Response Optimization, and How to Use Parallel Computing for Sensitivity Analysis.

MATLAB code generation from Parameter Estimation tool for automatically scripting tasks, including batch estimation and objective function customization

You can now generate MATLAB code to perform batch estimation and objective function customization tasks using the Parameter Estimation tool.

For an example, see Generate MATLAB Code for Parameter Estimation Problems (GUI).

R2014b

Version: 2.6

New Features

Bug Fixes

Redesigned Parameter Estimation tool for improved parameter estimation workflows

A redesigned Parameter Estimation tool streamlines and improves parameter estimation workflows. You can:

- Open legacy projects or create new sessions
- Preprocess data
- Create experiments to store estimation or validation data
- Select parameters to estimate
- Select estimation options
- Validate estimation results
- Create estimation progress, measured versus simulated, parameter trajectory, and residual plots
- Save sessions

For more information on using the tool, see the Prepare Data for Parameter Estimation and Estimate Parameters from Measured Data examples.

Per-experiment parameter estimation in Parameter Estimation tool, for estimating parameters with values that vary from experiment to experiment

You can now estimate parameters on a per-experiment basis in the Parameter Estimation tool. You can use multiple experiments to estimate a mix of model parameter values, some that are estimated using all the experiments and others that are estimated using individual experiments. For more information, see the Estimate Model Parameters Per Experiment (GUI) example.

R2014a

Parameter sensitivity analysis interface that supports design of experiments, Monte Carlo evaluation, and correlation analysis for Simulink models

Use new commands to analyze the sensitivity of the optimization cost function to model parameters or states. This analysis is also referred to as *sensitivity analysis*. The new commands support global sensitivity analysis. You sample parameters, specifying distributions such as normal or uniform. (If you have a Statistics Toolbox™ license, you can use its other distributions and Latin hypercube sampling.) Then, evaluate the optimization cost function at each sample point. You can plot the cost function output for the samples to visually analyze trends. You can also quantitatively analyze the relation between the evaluation function and the samples. Analysis methods include correlation, partial correlation (requires a Statistics Toolbox license), and standardized regression. You can configure each analysis method to use either the raw or ranked data.

Examples of using sensitivity analysis:

- Before optimization — Determine the influence of the parameters of a Simulink model on the output. Use sensitivity analysis to rank parameters in order of influence so that you can determine the most influential parameters. Optimize the model by tuning the most influential parameters or perform experiments to better characterize those parameters.
- After optimization — Test how robust the cost function is to small changes in the values of optimized parameters.

The new commands include sdo.sample, sdo.evaluate, sdo.analyze, and sdo.scatterPlot. For examples of using these commands, see Design Exploration using Parameter Sampling (Code) or Identify Key Parameters for Estimation (Code).

`RobustCost` property for reducing influence of outliers in signal tracking

Use the new `RobustCost` property of `sdo.requirements.SignalTracking` objects for robust treatment of outliers when the software evaluates the requirements. This option reduces the influence of outliers on the estimation without you manually modifying your data.

For more information, see sdo.requirements.SignalTracking.

R2013b

Version: 2.4

Bug Fixes

R2013a

Sequential Quadratic Programming is default for fmincon (Gradient Descent) algorithm

The default algorithm for the `fmincon` method is Sequential Quadratic Programming (SQP). Previously, the default algorithm was Active Set.

SQP is better suited than Active Set for problems that specify both an objective function and constraints. SQP ensures that every iterate satisfies the specified upper and lower bounds. When an objective or constraint function returns `Inf`, `NaN`, or complex values, the algorithm takes a smaller step, and continues. If a constraints-only problem is not successfully solved by SQP, use Active Set instead.

When performing design optimization or parameter estimation programmatically, you can specify optimization options using sdo.OptimizeOptions. The default value of the `MethodOptions.Algorithm` property of the options object is `'sqp'`. In the Design Optimization tool, the default **Algorithm** for `Gradient Descent` is `Sequential Quadratic Programming`.

For more information, see:

- fmincon SQP Algorithm in the Optimization Toolbox™ documentation
- sdo.OptimizeOptions
- Optimization Options

Compatibility Considerations

- Design Optimization tool sessions created in previous releases retain their saved optimization settings.
- The results of your code, written in a previous release, may be affected if you use the default solver with the default algorithm. That is, your code may yield different results if you:

  - Call sdo.optimize with only two inputs.
  - Specify an optimization options set with the default values of the `Method` and `MethodOptions.Algorithm` properties.

  You can revert the optimization settings by using an options object with `sdo.optimize`. This object must specify the `Method` and `MethodOptions.Algorithm` properties as `'fmincon'` and `'active-set'`.

Note: Your results may be affected by SQP and Active Set treating the constraint function tolerance differently. SQP treats this tolerance as a relative bound, proportional to the initial constraint violation, while Active Set treats it as an absolute bound. To specify a value for this tolerance at the command line, use the `MethodOptions.TolCon` property of the optimization options set.

Example of design optimization with uncertain variables

The new Design Optimization with Uncertain Variables (Code) example shows how to programmatically optimize a design when there are uncertain variables.

Example of specifying custom signal objectives with uncertain variables

The new Specify Custom Signal Objective with Uncertain Variable (GUI) example shows how to specify a custom objective function for a model signal in the Design Optimization tool.

R2012b

Redesigned commands for parameter estimation, enabling custom cost functions, parameter constraints, and estimation of parameters per experiment

Redesigned commands and objects streamline the programmatic parameter estimation workflow. You can now:

- Estimate parameters and initial conditions on a per experiment basis.
- Specify custom parameter constraints, such as enforcing that the static friction coefficient be greater than or equal the dynamic friction coefficient for a simple friction model.

  Similarly, you can specify custom initial condition constraints.
- Specify custom cost functions, such as log-likelihood and weighted sum square. Previously, you could specify sum of squared errors (SSE), or sum of absolute errors (SAE) cost functions.

Redesigned commands and objects include:

- `sdo.Experiment` for specifying measured input/output data, parameter values and initial-states for estimation.

  For more information, see sdo.Experiment.
- `sdo.getStateFromModel` for returning an object that parameterizes the initial-state of a Simulink model you are estimating.

  For more information, see sdo.getStateFromModel.
- `InitialState` property of `sdo.SimulationTest` for specifying the model initial-state.

  For more information, see sdo.SimulationTest.
- `ref` input of `sdo.requirements.SignalTracking.evalRequirement` for specifying the reference signal for evaluating this requirement using the new input of this method.

  For more information, see sdo.requirements.SignalTracking.evalRequirement.

For information on how the estimation is computed, see Computing the Estimation Error (Code).

For examples of programmatic parameter estimation, see:

- Estimate Model Parameter Values (Code)
- Estimate Model Parameters and Initial States (Code)
- Estimate Model Parameters Per Experiment (Code)
- Estimate Model Parameters using Multiple Experiments (Code)
- Estimate Model Parameters with Parameter Constraints (Code)

Compatibility Considerations

Parameter estimation commands from previous releases now warn and will be removed in a future version. Use the new parameter estimation commands instead.

For information regarding parameter estimation commands from previous releases, see Estimate Parameters Using Parameter Estimation Objects.

MATLAB code generation from Design Optimization tool for batch optimization of model responses

You can now generate MATLAB code to perform batch optimization of model responses using the Design Optimization Tool.

For an example, see Generate MATLAB Code for Design Optimization Problems (GUI) .

Skip model simulation based on parameter constraint violation

This release introduces functionality in the Design Optimization tool to prevent model evaluation with parameters that lead to a simulation error.

For an example, see “Skip Model Simulation Based on Parameter Constraint Violation (GUI)”.

R2012a

Formulating and Solving Response Optimization Problems with Frequency Domain Requirements Without Adding Blocks

You can now specify frequency-domain requirements, without adding blocks to the model, using the Design Optimization tool. This feature requires Simulink Control Design software.

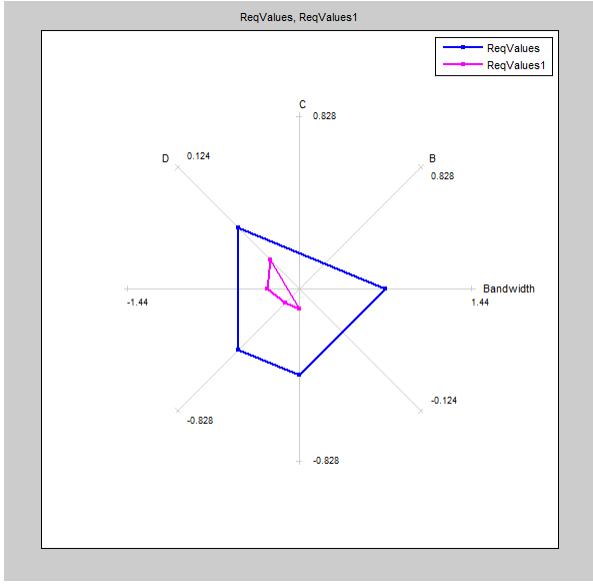
The type of frequency-domain requirements that you can specify include:

- Bounds on the gain and phase margin of a linear system
- Piecewise linear bounds on the Bode magnitude of the system response
- Bounds on the closed-loop peak response of a system
- Bounds on the damping ratio of the poles of a linear system
- Bounds on the natural frequency of the poles of a linear system
- Bounds on the location of the poles of a linear system such that an equivalent second order system would have a specified settling time
- Piecewise linear bound on the singular values of a linear system
- Bounds on the step response of a linear system

For more information, see how to specify requirements in Design Optimization to Meet Frequency-Domain Requirements.

Spider Plot for Comparing Design Variables Before and After Optimization

You can now compare design variable values and requirement values using a spider plot in the Design Optimization tool. Also known as radar charts, spider plots depict multivariate data using an axis for each variable. The various axes share a starting point, as this example plot shows:



For more information, see:

- Spider Plots
- Compare Requirements and Design Variables Using Spider Plot

R2011b

Redesigned Graphical Tool and Commands for Improved Response Optimization Workflows

A redesigned Design Optimization tool and new commands streamline and improve response optimization workflows. You can now:

- Create multiple sets of design and uncertain variables and time- and frequency-domain design requirements. This enables you to optimize the design using different combinations of variable sets and requirements.
- Specify design requirements without blocks, using Check blocks from the **Signal Constraints** library, or a combination of both.
- Monitor optimization progress using design variable values plot.
- Access the MATLAB and Simulink workspaces using the **Data Browser** area of the Design Optimization tool.

For more information, see:

- Optimize Controller Parameters to Meet Step Response Requirements (GUI)
- Optimize Controller Parameters to Meet Step Response Requirements (Code)
- Optimize Controller Parameters to Track Reference Signal (GUI)
- Design Optimization to Meet Frequency-Domain Requirements (GUI, with Check Block)
- Design Optimization to Meet Time- and Frequency-Domain Requirements
- Optimize Parameters for Robustness (GUI)

Compatibility Considerations

- Commands from previous releases now warn and will be removed in a future version. Use the new Response Optimization commands instead.
- The Signal Constraint block has been removed from the block library. Use `sdoupdatesdoupdate('modelname')` to automatically update your model to use the equivalent Check blocks from the **Signal Constraints** library.
- Simulation options, such as start and stop times and solver type, can no longer be set using Simulink Design Optimization™ software. Use the Configuration Parameters Dialog Box in Simulink instead.

Formulation and Solving of Response Optimization Problems Without Adding Blocks to the Model

You can now specify time-domain requirements without adding Check blocks to the Simulink model. You can do so from the Design Optimization tool or programmatically using requirement objects such as `sdo.requirements.StepResponseEnvelope`, `sdo.requirements.SignalBound` and `sdo.requirements.SignalTracking`.

Similarly, you can programmatically specify frequency-domain requirements without adding Check blocks from the Simulink Control Design library to the model. Frequency-domain requirement objects include `sdo.requirements.BodeMagnitude` and `sdo.requirements.GainPhaseMargin`.

For more information, see how to specify requirements in the following topics:

- Optimize Controller Parameters to Track Reference Signal (GUI)
- Design Optimization to Meet Custom Signal Requirements (GUI)
- Design Optimization to Meet a Custom Objective at the Command Line

Optimization of Model Parameters to Meet Frequency-Domain Requirements

If your Simulink model has Simulink Control Design Model Verification blocks, you can optimize the model response to meet the frequency-domain requirements specified in them. For example, you can optimize the model response to meet Bode magnitude requirements. You can also include time-domain requirements such as step response characteristics for optimization.

For more information, see:

- Design Optimization to Meet Frequency-Domain Requirements (GUI, with Check Block)
- Design Optimization to Meet Time- and Frequency-Domain Requirements

Optimization of Model Parameters to Meet Design Requirements Specified by Model Verification Blocks

You can optimize model response to meet requirements specified in Check Static Gap, Check Static Lower Bound and Check Static Upper Bound blocks from the Simulink

Model Verification library. The Design Optimization tool automatically includes the design requirements when you open the tool.

Custom Constraints and Cost Functions for Optimizing Model Response

This release provides functionality to specify custom requirements such as minimizing a cost function, an inequality constraint or an equality constraint. You write a function describing the custom requirement that you include for optimization either from the graphical user interface or programmatically.

For more information, see:

- Design Optimization to Meet a Custom Objective Using the GUI
- Design Optimization to Meet a Custom Objective at the Command Line
- Design Optimization to Meet Custom Signal Requirements (GUI)

Assertion Detection by Blocks During Time-Domain Model Verification

The Check Custom Bounds, Check Step Response Characteristics and Check Against Reference blocks in the **Model Verification** library detect assertions during simulation. Use these blocks to verify the time-domain characteristics of a nonlinear Simulink model satisfy specified bounds during simulation. For example, you can verify whether a model signal satisfies upper and lower bounds on its values. See Time-Domain Model Verification.

You can also use these blocks with the Model Verification blocks from Simulink and Simulink Control Design libraries to include frequency-domain bounds and build complex logic for model verification.

If you have Simulink Verification and Validation™ software, you can construct simulation tests for your model using the Verification Manager. For more information, see Verification Manager.

Functionality Being Removed or Changed

Functionality	What Happens When you Use This Functionality?	Use This Instead	Compatibility Considerations
`getsro`	Warns	Not applicable	See “Redesigned Graphical Tool and Commands for Improved

Functionality	What Happens When you Use This Functionality?	Use This Instead	Compatibility Considerations
			Response Optimization Workflows" on page 12-2.
`newsro`	Warns	Not applicable	See "Redesigned Graphical Tool and Commands for Improved Response Optimization Workflows" on page 12-2.
`optimize`	Warns	`sdo.optimize`	Replace all instances of `optimize` with `sdo.optimize`. See "Redesigned Graphical Tool and Commands for Improved Response Optimization Workflows" on page 12-2.
`findconstr`	Warns	`getbounds`	Replace all instances of `findconstr` with `getbounds`. See "Redesigned Graphical Tool and Commands for Improved Response Optimization Workflows" on page 12-2.
`findpar`	Warns	`sdo.getParameterFromM`	Replace all instances of `findpar` with `sdo.getParameterFromModel`. See "Redesigned Graphical Tool and Commands for Improved Response Optimization Workflows" on page 12-2.
`initpar`	Warns	Not applicable	See "Redesigned Graphical Tool and Commands for Improved Response Optimization Workflows" on page 12-2.

Functionality	What Happens When you Use This Functionality?	Use This Instead	Compatibility Considerations
`finddepend`	Warns	`sdo.getModelDependen`	Replace all instances of `finddepend` with `sdo.getModelDependencies`. See "Redesigned Graphical Tool and Commands for Improved Response Optimization Workflows" on page 12-2.
`gridunc`	Warns	Not applicable	No replacement, see Design Optimization Using the Command Line.
`randunc`	Warns	Not applicable	See "Redesigned Graphical Tool and Commands for Improved Response Optimization Workflows" on page 12-2.
`setunc`	Warns	Not applicable	See "Redesigned Graphical Tool and Commands for Improved Response Optimization Workflows" on page 12-2.
`optimget`	Warns	`sdo.OptimizeOptions`	Replace all instances of `optimget` with `sdo.OptimizeOptions`. See "Redesigned Graphical Tool and Commands for Improved Response Optimization Workflows" on page 12-2.
`optimset`	Warns	`sdo.OptimizeOptions`	Replace all instances of `optimset` with `sdo.OptimizeOptions`. See "Redesigned Graphical Tool and Commands for Improved Response Optimization Workflows" on page 12-2.
`GradientType` optimization setting	Errors	Not applicable	See "Redesigned Graphical Tool and Commands for Improved Response Optimization Workflows" on page 12-2.

Functionality	What Happens When you Use This Functionality?	Use This Instead	Compatibility Considerations
`simget`	Warns	Not applicable	Use the Configuration Parameters Dialog Box
`simset`	Warns	Not applicable	Use the Configuration Parameters Dialog Box
`ncdupdate`	Warns	`sdoupdate`	Replace all instances of `ncdupdate` with `sdoupdate`.

R2011a

Version: 1.2.1

Bug Fixes

R2010b

Version: 1.2

New Features

Bug Fixes

Compatibility Considerations

Support for Initial State Estimation of Model References, SimHydraulics, SimMechanics, SimPowerSystems, and Simscape Blocks

You can now estimate the initial states of:

- Model references
- SimHydraulics® blocks
- SimMechanics™ blocks
- SimPowerSystems™ blocks
- Simscape™ blocks

You can perform initial state estimation either using the GUI or from the command-line interface. For more information, see:

- Estimate Initial States
- Estimate Parameters (Code)

Compatibility Considerations

- Previously, you represented the states of an Integrator block having multiple state names by using one `StateData` or `State` object. Now, *each* state name requires one `StateData` or `State` object. Therefore, estimating the states of such an Integrator block errors. Instead, create a TransientExperiment or Estimation object to automatically create `StateData` and `State` objects, respectively.
- Previously, the `Domain` property of the `State Data` and `State` objects was used to track SimMechanics and SimPowerSystems blocks with states. This property is no longer required and has been removed.

Functions and Function Elements Being Removed

Function or Function Element Name	What Happens When you Use the Function or Element?	Use This Instead	Compatibility Considerations
`Domain` property of the `State Data` and `State` objects	Errors	Not applicable	See the **Compatibility Considerations** subheading for this change:

Function or Function Element Name	What Happens When you Use the Function or Element?	Use This Instead	Compatibility Considerations
			• "Support for Initial State Estimation of Model References, SimHydraulics, SimMechanics, SimPowerSystems, and Simscape Blocks" on page 14-2

R2010a

Version: 1.1.1

New Features

Bug Fixes

New Engine Design and Cost Tradeoffs Demo

The new Engine Design and Cost Tradeoffs demo shows how to use the Simulink Design Optimization software to optimize a design for performance and cost.

R2009b

Version: 1.1

New Features

Bug Fixes

Compatibility Considerations

New Algorithm Option for fmincon (Gradient descent) and lsqnonlin (Nonlinear least squares) Methods, LargeScale (Model size) Option Removed

This version of the product includes changes at the command line to make the `fmincon` and `lsqnonlin` methods to be consistent with the Optimization Toolbox software:

- `Algorithm` property renamed to `Method`.
- New `Algorithm` option.
- `LargeScale` option removed.

The following table summarizes values of the new `Algorithm` option.

Method	`Algorithm` Values
`fmincon`	• `'active-set'` (default) • `'trust-region-reflective'` • `'interior-point'`
`lsqnonlin`	• `'trust-region-reflective'` (default) • `'levenberg-marquardt'`

Previously, to specify the algorithm at the command line, you set the `LargeScale` option to `'on'` or `'off'`. If you used `LargeScale='on'` in a previous release, use `Algorithm='trust-region-reflective'` instead. If you used `LargeScale='off'`, use the following instead:

- For `fmincon` – Use `Algorithm='active-set'`.
- For `lsqnonlin` – Use `Algorithm='levenberg-marquardt'`.

For more information about these options, see the *Optimization Toolbox User's Guide*.

The Options dialog box includes the following updates to the `Gradient descent` and `Nonlinear least squares` methods that correspond to the command-line changes.

- **Algorithm** option is renamed to **Method**.
- **Model size** option is deprecated and replaced by **Algorithm**.

  When you load a saved project, the software uses the **Model size** value to update the **Algorithm** value automatically.

When you optimize parameters using the `Gradient Descent` method, an **Algorithm** value other than the default value of `Active-Set` can lead to a slightly different result.

For more information on how to specify the method and its algorithm, see Estimation Options and Optimization Options.

Support for Optimization-Based Compensator Design for Plants with Delays or Specified as Frequency-Response Data

You can now use optimization-based compensator design for frequency-response data (FRD) plants or plants with exact time delays in the SISO Design Tool. For more information, see Designing Optimization-Based Controllers for LTI Systems and Designing Linear Controllers for Simulink Models in the *Simulink Design Optimization User's Guide*.

Functions and Properties Being Removed

Function or Property Name	What Happens When You Use Function or Property?	Use This Instead	Compatibility Considerations
`Algorithm`	Errors	`Method`	See "New Algorithm Option for fmincon (Gradient descent) and lsqnonlin (Nonlinear least squares) Methods, LargeScale (Model size) Option Removed" on page 16-2
`LargeScale`	Errors	`Algorithm`	See "New Algorithm Option for fmincon (Gradient descent) and lsqnonlin (Nonlinear least squares) Methods, LargeScale (Model size) Option

Function or Property Name	What Happens When You Use Function or Property?	Use This Instead	Compatibility Considerations
			Removed” on page 16-2

R2009a

Version: 1.0

New Features

Bug Fixes

Compatibility Considerations

Simulink Parameter Estimation and Simulink Response Optimization Merged into New product

As of R2009a, Simulink Parameter Estimation™ and Simulink Response Optimization™ functionality are merged into a new product, Simulink Design Optimization. Simulink Parameter Estimation and Simulink Response Optimization are no longer available.

New Parallel Computing Support for Estimating Model Parameters

If you have the Parallel Computing Toolbox™ software installed, you can use parallel computing to speed up estimating parameters of a Simulink model. The parallel computing option is available in the `Nonlinear least squares`, `Gradient descent` and `Pattern search` algorithms. You can enable this option from either the GUI or at the command line.

Using parallel computing can speed up the estimation time in the following situations:

- The model contains a large number of parameters to estimate.
- The model is complex and takes a long time to simulate.

For more information about using parallel computing for estimating model parameters, see Speedup Using Parallel Computing in the Simulink Design Optimization documentation.

Updated Demos

The Simulink Design Optimization demos have been categorized into the following new categories:

- Parameter Estimation in Simulink
- Response Optimization in Simulink
- Response Optimization in SISO Design Tool
- Design Optimization Using Parallel Computing
- Adaptive Lookup Tables

To open the Simulink Design Optimization demos, type

```
demo simulink 'simulink design optimization'
```
at the MATLAB prompt.

Upgrading from Nonlinear Control Design Blockset Software

Prior to R14, Simulink Response Optimization software was called Nonlinear Control Design Blockset software. If you are upgrading from Nonlinear Control Design Blockset software, your models will not work with Simulink Design Optimization software. To make the models compatible with Simulink Design Optimization software, use `ncdupdate`.