Release Notes for Simulink® Design Optimization™

How to Contact MathWorks

`www.mathworks.com`	Web
`comp.soft-sys.matlab`	Newsgroup
`www.mathworks.com/contact_TS.html`	Technical Support

`suggest@mathworks.com`	Product enhancement suggestions
`bugs@mathworks.com`	Bug reports
`doc@mathworks.com`	Documentation error reports
`service@mathworks.com`	Order status, license renewals, passcodes
`info@mathworks.com`	Sales, pricing, and general information

508-647-7000 (Phone)

508-647-7001 (Fax)

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

For contact information about worldwide offices, see the MathWorks Web site.

Release Notes for Simulink® Design Optimization™

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.

Contents

R2013a

R2012b

R2012a

R2011b

R2011a

R2010b

R2010a

R2009b

R2009a

R2013a

Version: 2.3
New Features: Yes
Bug Fixes: Yes

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

Compatibility Considerations: Yes

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

Version: 2.2
New Features: Yes
Bug Fixes: Yes

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

Compatibility Considerations: Yes

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 more information, see Skip Model Simulation Based on Parameter Constraint Violation.

For an example, see Prevent Invalid Model Evaluation (GUI) .

R2012a

Version: 2.1
New Features: Yes
Bug Fixes: Yes

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.

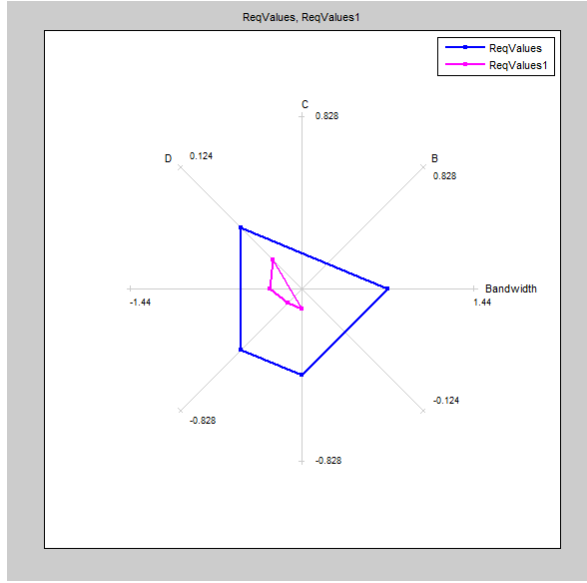
The type of frequency-domain requirements that you can specify include:

- 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

Version: 2.0
New Features: Yes
Bug Fixes: Yes

Redesigned Graphical Tool and Commands for Improved Response Optimization Workflows

Compatibility Considerations: Yes

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 `sdoupdate('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 Open and Use the Verification Manager.

Functionality Being Removed or Changed

Compatibility Considerations: Yes

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 Response Optimization Workflows" on page 18.
`newsro`	Warns	Not applicable	See "Redesigned Graphical Tool and Commands for Improved Response Optimization Workflows" on page 18.
`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 18.
`findconstr`	Warns	`getbounds`	Replace all instances of `findconstr` with `getbounds`. See "Redesigned Graphical Tool and Commands for Improved Response Optimization Workflows" on page 18.
`findpar`	Warns	`sdo.getParameterFromModel`	Replace all instances of `findpar` with `sdo.getParameterFromModel`. See "Redesigned Graphical Tool and Commands for Improved Response Optimization Workflows" on page 18.

Functionality	What Happens When you Use This Functionality?	Use This Instead	Compatibility Considerations
`initpar`	Warns	Not applicable	See "Redesigned Graphical Tool and Commands for Improved Response Optimization Workflows" on page 18.
`finddepend`	Warns	`sdo.getModelDependencies`	Replace all instances of `finddepend` with `sdo.getModelDependencies`. See "Redesigned Graphical Tool and Commands for Improved Response Optimization Workflows" on page 18.
`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 18.
`setunc`	Warns	Not applicable	See "Redesigned Graphical Tool and Commands for Improved Response Optimization Workflows" on page 18.
`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 18.

Functionality	What Happens When you Use This Functionality?	Use This Instead	Compatibility Considerations
`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 18.
`GradientType` optimization setting	Errors	Not applicable	See "Redesigned Graphical Tool and Commands for Improved Response Optimization Workflows" on page 18.
`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
New Features: No
Bug Fixes: Yes

R2010b

Version: 1.2
New Features: Yes
Bug Fixes: Yes

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

Compatibility Considerations: Yes

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

Compatibility Considerations: Yes

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: • "Support for Initial State Estimation of Model References, SimHydraulics, SimMechanics, SimPowerSystems, and Simscape Blocks" on page 32

R2010a

Version: 1.1.1
New Features: Yes
Bug Fixes: Yes

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: Yes
Bug Fixes: Yes

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

Compatibility Considerations: Yes

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 38
`LargeScale`	Errors	`Algorithm`	See "New Algorithm Option for fmincon (Gradient descent) and lsqnonlin (Nonlinear least squares) Methods, LargeScale (Model size) Option Removed" on page 38

R2009a

Version: 1.0
New Features: Yes
Bug Fixes: Yes

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

Compatibility Considerations: Yes

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`.