

# Simulink<sup>®</sup> Control Design<sup>™</sup> Release Notes



# MATLAB<sup>®</sup>&SIMULINK<sup>®</sup>

## How to Contact MathWorks



Latest news: [www.mathworks.com](http://www.mathworks.com)  
Sales and services: [www.mathworks.com/sales\\_and\\_services](http://www.mathworks.com/sales_and_services)  
User community: [www.mathworks.com/matlabcentral](http://www.mathworks.com/matlabcentral)  
Technical support: [www.mathworks.com/support/contact\\_us](http://www.mathworks.com/support/contact_us)



Phone: 508-647-7000



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

*Simulink<sup>®</sup> Control Design<sup>™</sup> Release Notes*

© COPYRIGHT 2004–2017 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](http://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](http://www.mathworks.com/patents) for more information.

## R2017a

---

<b>Custom Constraints and Objective Functions for Trimming Simulink Models: Calculate operating points with increased flexibility</b> .....	1-2
<b>Bounds on State Derivatives During Trimming: Constrain derivatives of model states that are not at steady state</b> ..	1-2
<b>addoutputspec Command: Add output specification to multiple operating point specification objects in an array</b> .....	1-2
<b>New Syntax For <code>setBlockParam</code>: Specify multiple block parameterizations at once</b> .....	1-3

## R2016b

---

<b>Batch Trimming for Parameter Variation: Vary model parameters, and compute multiple operating points using a single model compilation</b> .....	2-2
<b>Improved LPV System Construction: Compute operating point offsets for model inputs, outputs, states, and state derivatives during linearization</b> .....	2-2
<b>operspec Command: Create array of operating point specification objects</b> .....	2-3

## R2016a

<b>Redesigned Control System Designer: Design SISO controllers for feedback systems in Simulink using improved interactive workflows</b> . . . . .	3-2
<b>Control System Tuner App and systune Command: Automatically tune single-loop and multiloop control systems in Simulink to meet design requirements</b> . . . . .	3-3
<b>TimeUnit property added to sLinearizer and slTuner</b> . . . . .	3-3

## R2015b

<b>Automatic Tuning of 2-DOF PID Controllers with Fixed Setpoint Weights</b> . . . . .	4-2
<b>getTunedValue and setTunedValue commands for accessing tuned variables within slTuner interface</b> . . . . .	4-2
<b>getBlockParam and getBlockValue return parameterizations and values in a structure</b> . . . . .	4-3
<b>Functionality being removed or changed</b> . . . . .	4-4

## R2015a

<b>Improved input disturbance rejection with the PID tuning algorithm</b> . . . . .	5-2
<b>Automatic tuning of setpoint weight coefficients in 2-DOF PID Controller block for improved disturbance rejection</b>	5-3

<b>Linear Analysis Tool enhancements for improved linear analysis workflows</b> .....	<b>5-3</b>
<b>Simplified and faster linear analysis of Simulink models across different model parameter values in Linear Analysis Tool</b> .....	<b>5-5</b>
<b>Option to provide PID gains as external inputs to PID Controller and PID Controller (2DOF) blocks</b> .....	<b>5-6</b>

## R2014b

<b>Unfiltered-derivative option in discrete-time PID Controller blocks</b> .....	<b>6-2</b>
<b>FOH and matched methods for automatic rate conversion in sITuner interface</b> .....	<b>6-2</b>
<b>Improved support for genss block parameterization in sITuner interface</b> .....	<b>6-2</b>
<b>Support for additional multiplication modes in sITuner parameterization of Gain block</b> .....	<b>6-3</b>

## R2014a

<b>sITuner interface for improved control system tuning of Simulink models with <code>sys tune</code> or <code>looptune</code> functions, including tuning of gain-scheduled controllers (with Robust Control Toolbox)</b> .....	<b>7-2</b>
<b>Redesigned PID Tuner for improved PID tuning workflow</b> .	<b>7-2</b>

<b>PID controller tuning using system identification to model the plant from simulation input-output data in the PID Tuner</b> .....	7-3
<b>Option to specify multiple substitute linearizations of a Simulink block for batch linearization</b> .....	7-4

## R2013b

<b>Enhanced linearize command, providing faster batch linearization for model parameter variations</b> .....	8-2
<b>sllinearizer interface, providing faster batch linearization for multiple I/O sets</b> .....	8-2
<b>linearizeOptions and findopOptions for specifying options for linearization and operating point search</b> .....	8-2
<b>Highlight linear analysis points in Linear Analysis Plot Blocks and Model Verification Blocks</b> .....	8-3
<b>writeBlockValue command can update Simulink model with tuned parameter values from generalized LTI model</b> ....	8-4
<b>Format of BlockData structure identical for snapshot and operating point linearization</b> .....	8-5
<b>Linear Analysis Blocks and Model Verification Blocks save operating points with computed linear systems</b> .....	8-6

## R2013a

<b>Transient behavior slider added to PID Tuner for increased control over reference tracking and disturbance rejection performance</b> .....	9-2
---	-----

<b>Linear analysis points redesigned to clarify I/O types and loop openings</b> .....	<b>9-4</b>
<b>Linear Analysis Blocks and Model Verification Blocks save data in single object with Simulink model logging output</b> .....	<b>9-6</b>

## R2012b

<b>MATLAB code generation from Linear Analysis Tool for batch estimation of model frequency responses</b> .....	<b>10-2</b>
<b>Operating point calculation (trimming) from multiple specifications with only one model compilation</b> .....	<b>10-2</b>
<b>Export and import operating point specifications in Linear Analysis Tool</b> .....	<b>10-2</b>
<b>MATLAB script or function generation from Linear Analysis Tool for repeated or batch linearization</b> .....	<b>10-3</b>
<b>Print plots to MATLAB figure in Linear Analysis Tool</b> ....	<b>10-3</b>
<b>Commands for setting and querying rate conversion methods in tunable blocks</b> .....	<b>10-3</b>
<b>“Ignore saturation when linearizing” checked by default in PID Controller and PID Controller (2DOF) blocks</b> .....	<b>10-4</b>
<b>showBlockValue renamed to showTunable</b> .....	<b>10-4</b>

## R2012a

<b>Create Linearization Input/Output Sets in the Linear Analysis Tool</b> .....	<b>11-2</b>
---	-------------

<b>Specify Feedback Sign for getLoopTransfer Without Specifying Loop Openings .....</b>	<b>11-2</b>
---	-------------

**R2011b**

<b>Redesigned Graphical Tool for Improved Linear Analysis Workflows .....</b>	<b>12-2</b>
<b>Interactive Frequency Response Estimation and Validation of Linearization Results .....</b>	<b>12-2</b>
<b>Optimization of Model Parameters to Meet Design Requirements Specified by Model Verification Blocks ..</b>	<b>12-2</b>
<b>Automatic Tuning of PID Controller Blocks in a Referenced Model .....</b>	<b>12-3</b>
<b>Control System Tuning for Simulink Models with looptune or hinfstruct Using slTunable Interface .....</b>	<b>12-3</b>
<b>Change in Default Number of Samples in frest.Chirp .....</b>	<b>12-3</b>

**R2011a**

<b>Ability to Select Individual Bus Elements as Linearization Input and Output Points .....</b>	<b>13-2</b>
<b>Enhanced LINLFT Command Optionally Returns Linearization of Excluded Blocks .....</b>	<b>13-2</b>
<b>Access to Current Linearization of a Simulink Block for Specifying Custom Linearization .....</b>	<b>13-2</b>
<b>Enhanced PID Controller Blocks Display Compensator Formula in Block Dialog Box .....</b>	<b>13-3</b>



<b>New Blocks for Plotting and Verifying Linear System Characteristics of Simulink Models</b> .....	14-2
Plotting Linear System Characteristics of Simulink Models .	14-2
Verifying Linear System Characteristics of Simulink Models .....	14-2
<b>New Tools for Identifying Time-Varying Source Blocks for Frequency Response Estimation</b> .....	14-3
<b>Tuning Tools Update Workspace Variables That Define Parameters of Tuned Blocks</b> .....	14-4
<b>Enhanced PID Tuner Including New Response Plots</b> .....	14-4
<b>New Demo Illustrating Control Design for a Plant That Has Parameter Variations</b> .....	14-5

<b>New Parallel Computing Support For Frequency Response Estimation</b> .....	15-2
<b>New Commands Support Recomputing Frequency Response Estimation Results at Specific Frequencies</b> .....	15-2
<b>New frest.simcompare Output Argument Returns Simulation Output Data From Linear System</b> .....	15-2
<b>New Options in Simulink Results Viewer GUI for Viewing Frequency Response Estimation Results</b> .....	15-2
<b>New Option for Labeling Bus Signal I/O Names in the SISO Design Task</b> .....	15-3
<b>Existing Simulink Blocks Now Have Analytic Jacobians</b> ..	15-3

<b>Change in Format of Time Series in frestimate Output . . .</b>	<b>15-3</b>
---	-------------

## **R2009b**

<b>New GUI for Tuning New PID Controller Blocks . . . . .</b>	<b>16-2</b>
<b>New Automated PID Tuning Algorithm . . . . .</b>	<b>16-2</b>
<b>Ability to Compute Frequency Response of Simulink Models . . . . .</b>	<b>16-2</b>
<b>Ability to Specify the Linearization of Simulink Blocks . . .</b>	<b>16-2</b>
<b>Ability to Design Compensators for Plant Models With Time Delays . . . . .</b>	<b>16-2</b>
<b>New Commands to More Efficiently Compute Multiple Linearizations . . . . .</b>	<b>16-3</b>
<b>Ability to Set Default Plot Type for Linear Analysis Results from GUI . . . . .</b>	<b>16-3</b>

## **R2009a**

<b>Ability to Generate MATLAB Code from the GUI for Creating Operating Points and Linearizing Models . . . . .</b>	<b>17-2</b>
<b>Ability to Tune Additional Blocks . . . . .</b>	<b>17-2</b>
<b>New Option for Labeling Bus Signal I/O Names in Linearization Results . . . . .</b>	<b>17-2</b>

## R2008b

<b>New Upsampling Option for Rate Conversion When Linearizing Simulink Models</b> .....	18-2
<b>Ability to Specify State Order of Linearized Models from the Command Line</b> .....	18-2
<b>Ability to Filter the Linearization Inspector to Show Blocks in the Linearization Path</b> .....	18-2
<b>Ability to Disable the Calculation of Linearization Diagnostics and Inspector Data in the GUI</b> .....	18-2

## R2008a

<b>New Diagnostic Messages Help You Troubleshoot Linearization Results</b> .....	19-2
<b>Ability to Find Operating Points for Simscape Models</b> ....	19-2
<b>Updated Error and Warning Message System</b> .....	19-2

## R2007b

<b>Ability to Linearize Models with Model-Reference Blocks by Any Linearization Method</b> .....	20-2
<b>Ability to Design Compensators for Models Containing Model-Reference Blocks</b> .....	20-2
<b>Ability to Generate Linearized Models with Exact Time-Delay Representations</b> .....	20-2

<b>Ability to Linearize Periodic Function-Call Subsystems . . .</b>	<b>20-3</b>
---	-------------

## **R2007a**

---

<b>Ability to Linearize Using an Operating Point Specified Directly in a Model . . . . .</b>	<b>21-2</b>
<b>Ability to Capture Linearization Snapshots in GUI . . . . .</b>	<b>21-2</b>
<b>Ability to Perform Control Design at Snapshots in GUI . . .</b>	<b>21-2</b>
<b>Ability to Retrieve Stored Compensator Designs . . . . .</b>	<b>21-2</b>

## **R2006b**

---

**Bug Fixes**

## **R2006a**

---

<b>Compensator Design in Simulink Is Now Supported . . . . .</b>	<b>23-2</b>
--	-------------

## **R14SP3**

---

<b>Control and Estimation Tools Manager Enhanced . . . . .</b>	<b>24-2</b>
<b>Support for Operating Point Search and Linearization of Models with Model Reference Blocks . . . . .</b>	<b>24-2</b>

<b>Context-Sensitive Help Added</b> .....	25-2
<b>View Linearizations in the Control and Estimation Tools Manager</b> .....	25-2
<b>Discretization Methods Added</b> .....	25-2
<b>List of Blocks with Preprogrammed Analytic Jacobians Added</b> .....	25-2
<b>Block Name Readability Improved</b> .....	25-2



# R2017a

**Version: 4.5**

**New Features**

**Bug Fixes**

## **Custom Constraints and Objective Functions for Trimming Simulink Models: Calculate operating points with increased flexibility**

You can now add custom constraints and objective functions when trimming Simulink<sup>®</sup> models. You can specify either custom constraints, a custom objective function, or both. During trimming, the `findop` command applies the custom constraints and objective function in addition to the standard constraints and objective function.

For complex models, you can simplify your custom constraints and custom objective function by specifying a custom mapping. This mapping defines the subsets of model states, inputs, and outputs required for computing the constraints and objective function. Creating a mapping function involves finding the indices of the required states, inputs, and outputs in the operating point specification for a model. You can find these indices using the new `getStateIndex`, `getInputIndex`, and `getOutputIndex` commands.

For more information, see “Compute Operating Points Using Custom Constraints and Objective Functions”.

## **Bounds on State Derivatives During Trimming: Constrain derivatives of model states that are not at steady state**

When trimming a model at the command line, you can now constrain the derivatives of model states that are not at steady state during trimming. Using such constraints, you can trim the state derivatives to known nonzero values or specify derivative tolerances for states that cannot reach steady state.

For more information, see `operspec` and “Compute Operating Points from State Specifications at Command Line”.

## **addoutputspec Command: Add output specification to multiple operating point specification objects in an array**

You can now simultaneously add an output specification for a Simulink model to multiple operating point specification objects in an array with one call to `addoutputspec`. To do so, all of the specification objects must have the same `Model` property. Previously, you had to add the output specification to each operating point specification individually, which required compiling the model multiple times.

For more information, see the `addoutputspec` reference page.



---

## **New Syntax For `setBlockParam`: Specify multiple block parameterizations at once**

You can now specify custom parameterizations for multiple blocks in an `sITuner` interface with one call to `setBlockParam`. The following syntax assigns the parameterizations `tunable_md11`, `tunable_md12`, ..., `tunable_md1N` to blocks `blk1`, `blk2`, ..., `blkN`, respectively. The blocks are designated as tuned blocks in an `sITuner` interface `st`.

```
setBlockParam(st,blk1,tunable_md11,blk2,tunable_md12,...,blkN,tunable_md1N)
```

For more information, see the `setBlockParam` reference page.



# R2016b

**Version: 4.4**

**New Features**

**Bug Fixes**

## **Batch Trimming for Parameter Variation: Vary model parameters, and compute multiple operating points using a single model compilation**

You can now efficiently batch trim a model for variations in model parameters using the `findop` command. If the varying parameters are tunable, the software compiles the model only once, making batch trimming faster, especially for models that are expensive to compile.

To batch trim your model for a single varying parameter, specify the new `param` input argument as a structure with the following fields:

- **Name** — Parameter name, specified as a character vector or MATLAB<sup>®</sup> expression
- **Value** — Parameter sample values, specified as a double array

To vary the value of multiple parameters, specify an array of such structures.

For more information, see [Batch Compute Steady-State Operating Points for Parameter Variation](#).

## **Improved LPV System Construction: Compute operating point offsets for model inputs, outputs, states, and state derivatives during linearization**

You can now compute operating point offsets for model inputs, outputs, states, and state derivatives when linearizing Simulink models.

To construct a linear parameter-varying (LPV) system, you linearize your model for a grid of operating points in the scheduling parameter space. Each grid point contains offset information about model inputs, outputs, states, and state derivatives (continuous-time) or updates (discrete-time). Starting in R2016b, you no longer manually construct the offset structure for each grid point. Instead, linearization commands can now return this offset information.

To obtain operating point offsets, first create a `linearizeOptions` option set, and set the new `StoreOffsets` option to `true`.

You can then linearize your model and obtain the operating point offsets using the:

- `linearize` command.
- `getIOTransfer`, `getLoopTransfer`, `getSensitivity`, and `getCompSensitivity` linearization commands for an `sLinearizer` interface.

- 
- Linearization commands for an slTuner interface. In this case, use an sltunerOptions option set.

For each of these linearization functions, the new `info` output argument is a structure with an `Offsets` field. The new `getOffsetsForLPV` command extracts the linearization offsets from the `info` and converts them into the required format for the LPV System block.

For an example of LPV construction using operating point offsets, see [Approximating Nonlinear Behavior Using an Array of LTI Systems](#). For more information on LPV systems, see [Linear Parameter-Varying Models](#).

## **operspec Command: Create array of operating point specification objects**

You can now create an array of operating point specification objects using the `operspec` command. You can modify the individual specification objects based on your trimming requirements and batch-compute operating points using a single model compilation.

For an example that uses an array of operating point specification objects, see [Designing a Family of PID Controllers for Multiple Operating Points](#).



# R2016a

**Version: 4.3**

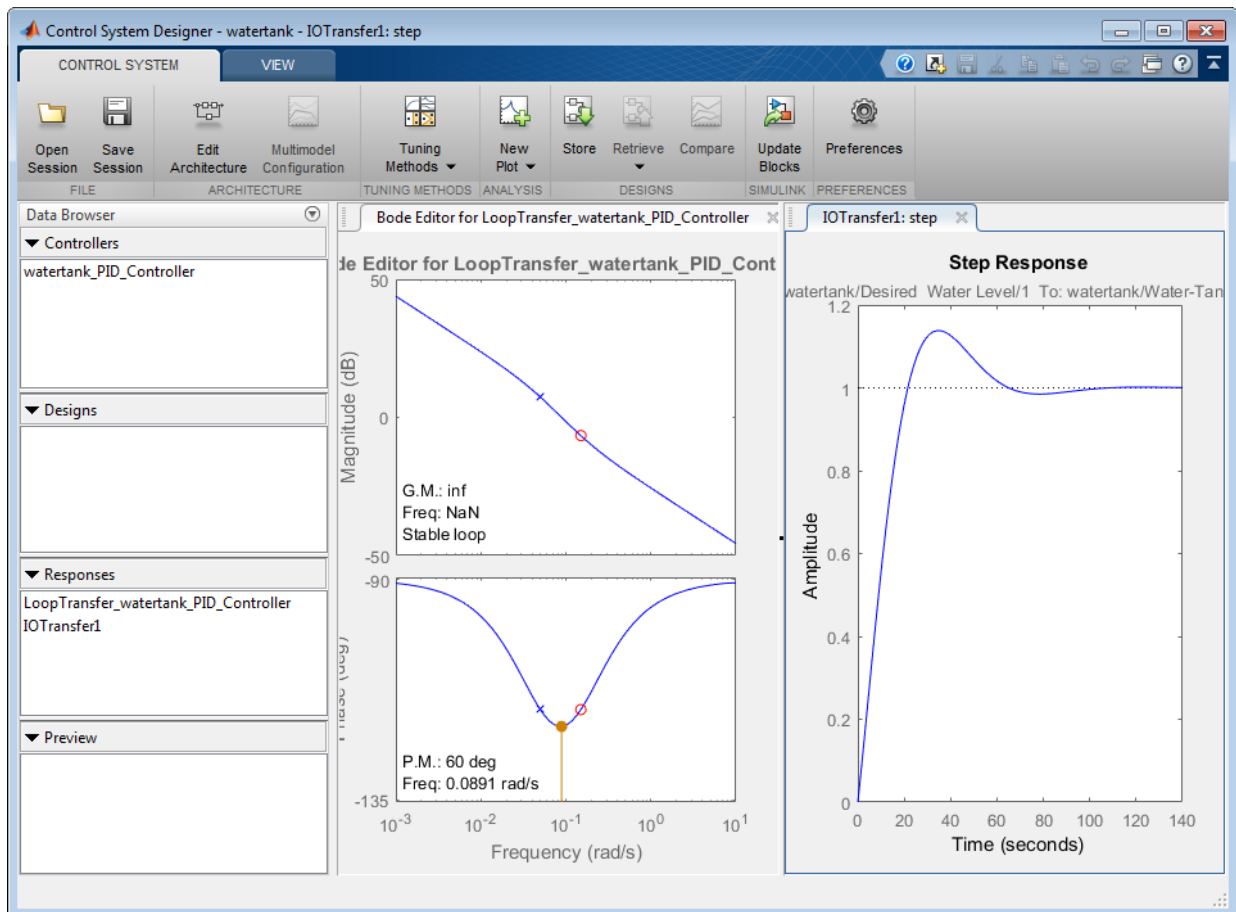
**New Features**

**Bug Fixes**

**Compatibility Considerations**

## Redesigned Control System Designer: Design SISO controllers for feedback systems in Simulink using improved interactive workflows

The redesigned **Control System Designer** app streamlines workflows for designing controllers for SISO feedback control systems in Simulink using graphical and automated tuning methods.



For more information on using **Control System Designer** in Simulink, see:

- [Control System Designer](#)



- 
- Control System Designer Tuning Methods
  - Design Compensator Using Automated PID Tuning and Graphical Bode Design
  - Analyze Designs Using Response Plots

## **Control System Tuner App and `systemune` Command: Automatically tune single-loop and multiloop control systems in Simulink to meet design requirements**

A Robust Control Toolbox™ license is no longer required to use the `systemune` or `looptune` commands or to use Control System Tuner. The Control System Tuner app and the `systemune` command automatically tune control systems from high-level design goals you specify, such as reference tracking, disturbance rejection, and stability margins. You can now use these tools to tune control systems modeled in Simulink with a Simulink Control Design™ license.

To tune a control system, you specify the tunable elements of your control system. You then capture your design requirements using the library of tuning goals. The software jointly tunes all the free parameters of your control system regardless of control system architecture, the number of feedback loops it contains, or whether it is modeled in MATLAB or Simulink.

For information about using these tools, see:

- Tuning with Control System Tuner
- Programmatic Tuning

You can also use the `systemune` command to tune gain-scheduled controllers for control systems in which plant dynamics change with operating conditions or time. For more information, see Gain Scheduling.

## **TimeUnit property added to `sLinearizer` and `sTuner`**

You can now specify time units for `sTuner` and `sLinearizer` interfaces using the new `TimeUnit` property. This property specifies the time units for linearized models returned by `getIOTransfer`, `getLoopTransfer`, `getSensitivity`, and `getCompSensitivity`.

Starting in R2016a, for both `sLinearizer` and `sTuner` interfaces, specify time units using dot notation. For example:

```
sllin = sLinearizer('scdcascade',{ 'r', 'u1', 'u2', 'y1', 'y2', 'y1m', 'y2m' });  
sllin.TimeUnit = 'minutes';
```

When loading `sLinearizer` and `sTuner` interfaces that were saved using a previous release, the `TimeUnit` property is set to `'seconds'` by default.

# R2015b

**Version: 4.2.1**

**New Features**

**Bug Fixes**

**Compatibility Considerations**

## Automatic Tuning of 2-DOF PID Controllers with Fixed Setpoint Weights

When you use PID Tuner to tune a PID Controller (2DOF) block, new options let you select controller types with fixed setpoint weights on the proportional and derivative terms,  $b$  and  $c$ . These new options include commonly-used fixed-weight controller configurations such as I-PD ( $b = 0, c = 0$ ) and PI-D ( $b = 1, c = 0$ ).

In R2015a, PID Tuner treated all PID coefficients, including setpoint weights, as free parameters when you tuned a PID Controller (2DOF) block. In previous releases, you could only tune 2-DOF controllers using the setpoint-weight values that you set manually in the block.

For more information about tuning 2-DOF PID controllers, see Design Two-Degree-of-Freedom PID Controllers. For more information about using PID Tuner, see Introduction to Automatic PID Tuning.

## getTunedValue and setTunedValue commands for accessing tuned variables within sITuner interface

To access tuned variables within an `sITuner` interface, use the new `getTunedValue` and `setTunedValue` commands. Tuned variables are any Control Design Blocks involved in the parameterization of a tuned Simulink block, either directly or through a generalized parametric model.

For Simulink blocks parameterized by a generalized model or a tunable surface:

- `getBlockValue` now provides access only to the overall value of the block parameterization. To access the values of the tuned variables within the block parameterization, use `getTunedValue`.
- `setBlockValue` can no longer be used to modify the block value. Use `setTunedValue` to modify the values of tuned variables within the block parameterization.

For Simulink blocks parameterized by a Control Design Block, the block itself is the tuned variable. To modify the block value, you can use either `setBlockValue` or `setTunedValue`. Similarly, you can retrieve the block value using either `getBlockValue` or `getTunedValue`.

Also, the syntax `setBlockValue(st,M)`, where  $M$  is a generalized model, has been replaced. To set the values of tunable parameters within a custom parameterization using a generalized model, use the new `setTunedValue(st,M)` syntax. The `setBlockValue(st,M)` syntax will continue to work in future releases.

---

## Compatibility Considerations

- If your code uses `getBlockValue` or `setBlockValue` to access the values of tunable elements within blocks parameterized by a generalized model, modify your code to use the new `getTunedValue` and `setTunedValue` commands. For example, when accessing or modifying the tunable parameter 'Ki' in the `s1Tuner` interface `st`, make the following code substitutions:

Old Syntax	Result	New Syntax
<code>KiTuned = getBlockValue(st, 'Ki')</code>	Error	<code>KiTuned = getTunedValue(st, 'Ki')</code>
<code>setBlockValue(st, 'Ki', Error)</code>	Error	<code>setTunedValue(st, 'Ki', 10)</code>

- If your code uses `setBlockValue(st, M)` to set the values of tunable parameters within an `s1Tuner` interface, consider modifying your code to use the new `setTunedValue(st, M)` syntax.

## getBlockParam and getBlockValue return parameterizations and values in a structure

New syntax of the `getBlockParam` command returns a structure that contains all block parameterizations of an `s1Tuner` interface. The syntax returns a structure, `S`, whose field names are the names of the tunable blocks in the `s1tuner` interface, `st`.

```
S = getBlockParam(st)
```

For more information, see the `getBlockParam` reference page.

Similarly, a new syntax of the `getBlockValue` command returns a structure that contains the current values of all block parameterizations of an `s1Tuner` interface. The syntax returns a structure, `S`, whose field names are the names of the tunable blocks in the `s1tuner` interface, `st`.

```
S = getBlockValue(st)
```

You can use this syntax to transfer the tuned values from one `s1Tuner` interface to another `s1Tuner` interface with the same tuned block parameterizations:

```
S = getBlockValue(st1);  
setBlockValue(st2, S);
```

For more information, see the `getBlockValue` reference page.

## Compatibility Considerations

Previously, the syntax `getBlockParam(st)` returned the current block parameterizations of `st` as a vector.

```
[Param1,Param2,...] = getBlockParam(st)
```

Now, using this syntax causes an error. You can still obtain block parameterizations in a list by specifying the block names as input arguments, as follows:

```
[Param1,Param2,...] = getBlockParam(st,Blkname1,Blkname2,...)
```

Similarly, the syntax `getBlockValue(st)` previously returned the current values of the block parameterizations of `st` as a vector.

```
[Val1,Val2,...] = getBlockValue(st)
```

Now, using this syntax causes an error. You can still obtain block parameterization values in a list by specifying the block names as input arguments, as follows:

```
[Val1,Val2,...] = getBlockValue(st,Blkname1,Blkname2,...)
```

## Functionality being removed or changed

Functionality	Result	Use This Instead	Compatibility Considerations
<code>val = getBlockValue(st,b)</code> with <code>blk</code> representing a tunable element in <code>sITuner</code> interface <code>st</code> .	Error	<code>val = getTunedValue(st,</code>	See “ <code>getTunedValue</code> and <code>setTunedValue</code> commands for accessing tuned variables within <code>sITuner</code> interface” on page 4-2 for more information.
<code>setBlockValue(st,b)</code> with <code>blk</code> representing a tunable element in <code>sITuner</code> interface <code>st</code> .	Error	<code>setTunedValue(st,</code>	See “ <code>getTunedValue</code> and <code>setTunedValue</code> commands for accessing tuned variables within <code>sITuner</code> interface” on page 4-2 for more information.
<code>setBlockValue(st,M)</code>	Still Work	<code>setTunedValue(st,</code>	See “ <code>getTunedValue</code> and <code>setTunedValue</code> commands for accessing tuned variables within <code>sITuner</code> interface” on page 4-2 for more information.

Functionality	Result	Use This Instead	Compatibility Considerations
[Param1,Param2,...] = getBlockParam(st)	Error	S = getBlockParam(st)	getBlockParam(st) now returns a structure that contains the current parameterizations of all tunable blocks in st. Update scripts and functions that use getBlockValue(st) to use an output structure.
[Val1,Val2,...] = getBlockValue(st)	Error	S = getBlockValue(st)	getBlockValue(st) now returns a structure that contains the current values of the parameterizations of all tunable blocks in st. Update scripts and functions that use getBlockValue(st) to use an output structure.





# R2015a

**Version: 4.2**

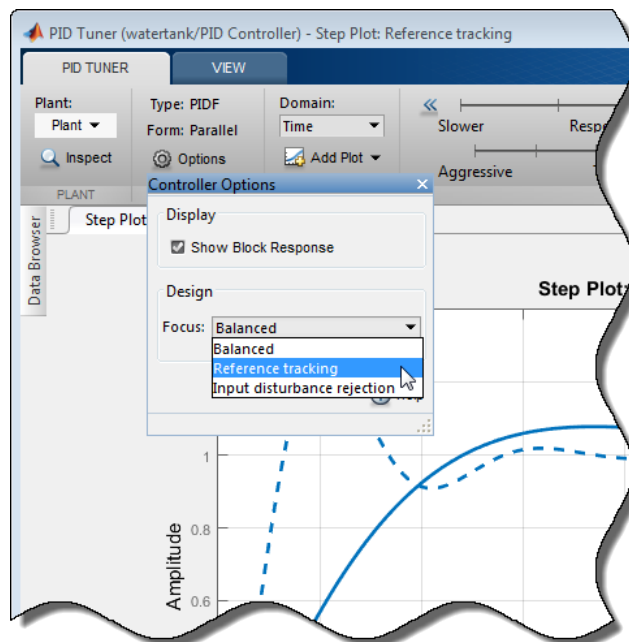
**New Features**

**Bug Fixes**

## Improved input disturbance rejection with the PID tuning algorithm

Controller tuning with the PID Tuner app now yields better disturbance rejection by default. For a given target phase margin, the tuning algorithm selects PID coefficients that achieve a balance between reference tracking and input disturbance rejection. If you require more disturbance rejection or better reference tracking than the default algorithm provides, PID Tuner has a new Design Focus option. Use this option to alter the balance that the tuning algorithm sets between reference tracking and input disturbance rejection. For instance, setting the design focus to reference tracking improves the reference tracking performance of the tuned controller, with some cost to disturbance rejection. Similarly, setting the design focus to input disturbance rejection improves rejection with some cost to reference tracking. Changing design focus is most effective when tuning PID controllers, rather than controllers with fewer free parameters, such as PI.

To use the Design Focus option in PID Tuner, click **Options** and select a design focus from the **Focus** menu.



---

You can still use the **Response Time** and **Transient Behavior** sliders to further adjust the balance between reference tracking and input disturbance rejection.

For more information about using the design focus option, see [Tune PID Controller to Favor Reference Tracking or Disturbance Rejection](#).

For more information about using PID Tuner, see [Introduction to Automatic PID Tuning](#).

## **Automatic tuning of setpoint weight coefficients in 2-DOF PID Controller block for improved disturbance rejection**

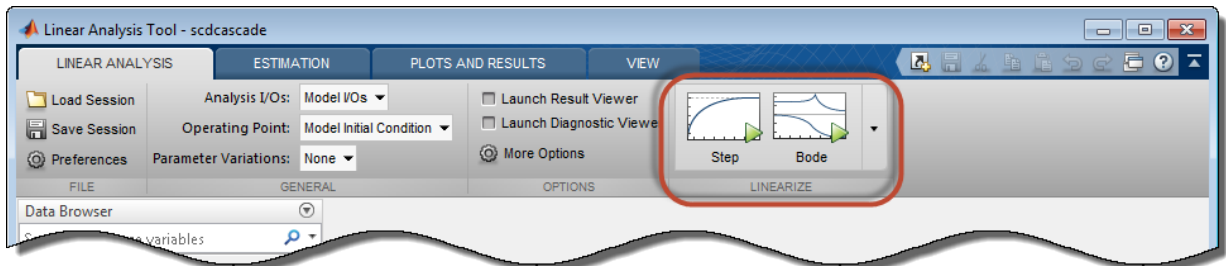
You can now use PID Tuner to tune all parameters of the PID Controller (2DOF) block, including the setpoint weights  $b$  and  $c$ . When you open the PID Tuner app from a PID Controller (2DOF) block in your Simulink model, the software automatically tunes all parameters of the block to achieve a balance between performance and robustness. When you use the Response Time and Transient Behavior sliders to adjust that balance, PID Tuner adjusts all parameters, including  $b$  and  $c$  if necessary. Previously, the software tuned only the PID gains and filter coefficients, and you had to adjust the setpoint weights manually.

For more information about 2-DOF PID controllers, see [Design Two-Degree-of-Freedom PID Controllers](#). For more information about using PID Tuner, see [Introduction to Automatic PID Tuning](#).

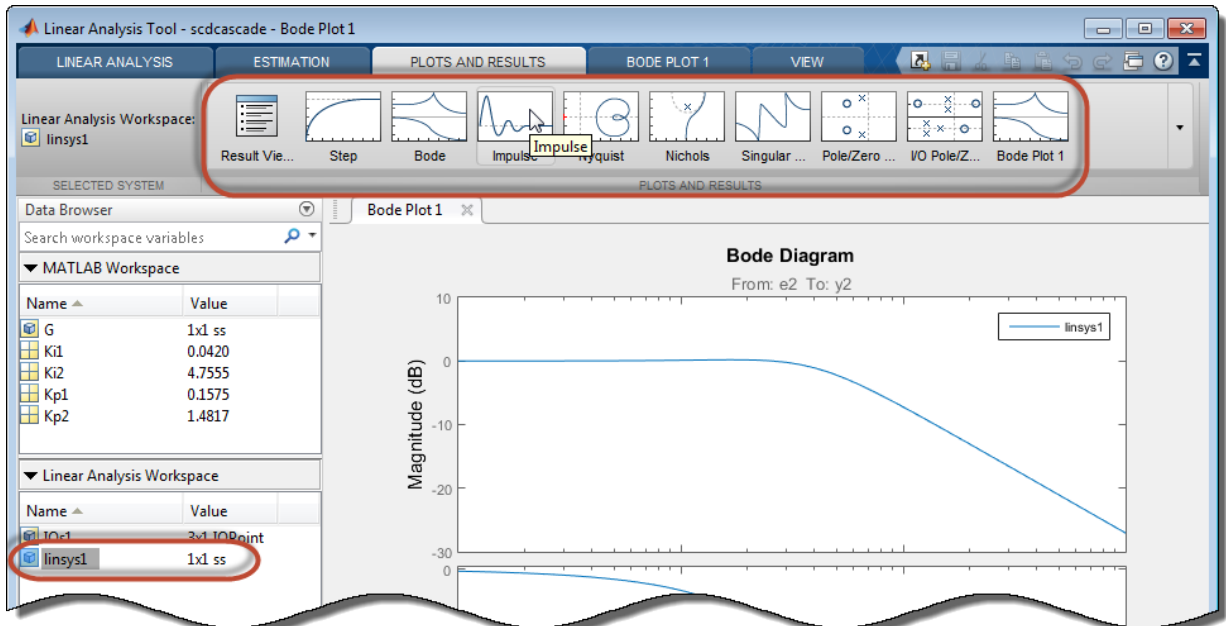
## **Linear Analysis Tool enhancements for improved linear analysis workflows**

The redesigned Linear Analysis Tool streamlines workflows for linear analysis tasks such as finding steady-state operating points (trimming), linearization, and frequency-response estimation. Improvements include:

- The **Linear Analysis** and **Estimation** tabs are enhanced with a plot gallery for quick, visual selection of plot types for viewing linearized or estimated system responses.



- The new **Plots and Results** tab also provides a gallery of plot types for creating additional plots of system responses. In the **Plots and Results** tab, select a dynamic system model from the MATLAB Workspace or the Linear Analysis Workspace. The plot gallery updates to reflect the available plots for the type of system you have selected. If the system you select was created in the Linear Analysis Tool session, the result viewer is also available in the plot gallery.



---

To access the Linear Analysis Tool from a Simulink model, in the Simulink Editor, select **Analysis > Control Design > Linear Analysis**. See the following topics for more information about using Linear Analysis Tool for:

- Computing operating points — Steady-State Operating Points (Trimming) from Specifications
- Exact linearization — Linearize Simulink Model at Model Operating Point
- Frequency Response Estimation — Estimate Frequency Response Using Linear Analysis Tool

## **Simplified and faster linear analysis of Simulink models across different model parameter values in Linear Analysis Tool**

You can now use the Linear Analysis Tool to efficiently batch linearize a model at varying plant or controller parameter values. *Batch linearization* refers to extracting multiple linearizations from a model for various combinations of model parameters. Using the new parameter-variation capability of Linear Analysis Tool, you can obtain multiple linearizations for varying values of any parameter in your Simulink model. For example, you can linearize your system at multiple values of plant coefficients, controller gains, or controller sample times.

Use Linear Analysis Tool response plots to examine time-domain and frequency-domain responses of the resulting linear models. For instance, you can compute and plot linearizations for the plant model, overall closed-loop transfer function, and plant disturbance rejection for varying plant and controller parameter values.

In cases where the varying parameters are all tunable, Linear Analysis Tool requires only one model compilation to compute multiple open-loop and closed-loop transfer functions for varying parameter values. This efficiency is especially advantageous for models that are expensive to compile repeatedly.

This new functionality brings the batch linearization capability of the `slLinearizer` interface and the `linearize` command to the Linear Analysis Tool. Any parameter variations you specify in Linear Analysis Tool are included in MATLAB code you generate from the session.

For more information about performing varying parameters for linearization with the Linear Analysis Tool, see [Batch Linearize Model for Parameter Value Variations Using Linear Analysis Tool](#).

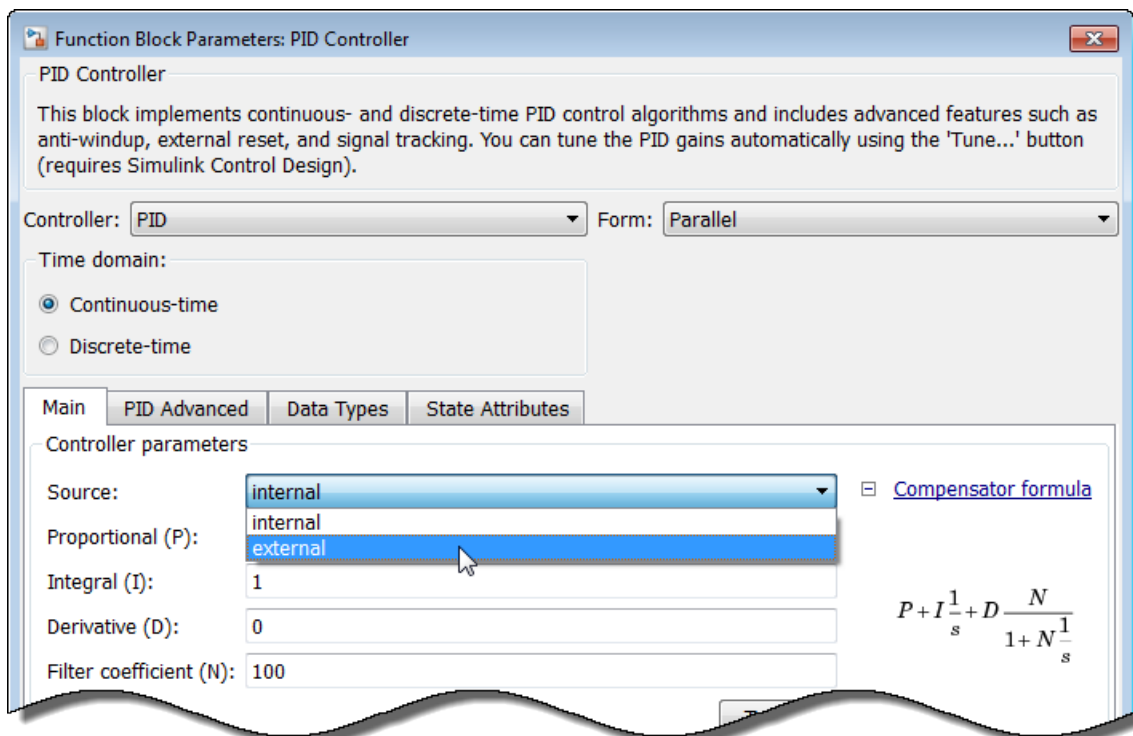
For more information about batch linearization, see [What Is Batch Linearization?](#)

## Option to provide PID gains as external inputs to PID Controller and PID Controller (2DOF) blocks

A new option in the PID Controller and PID Controller (2DOF) blocks adds signal inputs for the PID gains and filter coefficients. Previously the PID parameters had to be entered in the block dialog box as numerical values or MATLAB expressions. Enabling external inputs for the parameters allows you to compute PID gains and filter coefficients externally to the block and provide them to the block as signal inputs.

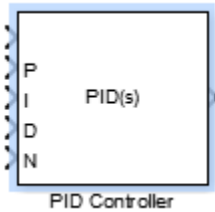
External gain input is useful, for example, when you want to map a different PID parameterization to the PID gains of the block. You can also use external gain input to implement gain-scheduled PID control, in which controller gains are determined by logic or other calculation in the Simulink model and passed to the block.

To enable external inputs for the PID coefficients, in the block dialog box, in the **Controller parameters** section, in the **Source** menu, select **external**.



---

When you click **OK** or **Apply**, the new inputs appear on the block in the Simulink model.



For an example illustrating the use of external gain inputs for gain scheduling, see [Implement Gain-Scheduled PID Controllers](#). For more information about using the PID controller blocks, see the [PID Controller](#) and [PID Controller \(2DOF\)](#) block reference pages.





# R2014b

**Version: 4.1**

**New Features**

**Bug Fixes**

## Unfiltered-derivative option in discrete-time PID Controller blocks

You can now specify an unfiltered derivative term in the discrete-time PID Controller and PID Controller (2DOF) blocks. Previously, these blocks required a finite derivative filter constant on the derivative term.

To specify an unfiltered derivative, in the **Main** pane of the block dialog box, uncheck **Use filtered derivative**. Unchecking this option replaces the derivative filter with a discrete differentiator. The option is checked by default for compatibility with previous versions.

For more information about the blocks, see PID Controller and PID Controller (2DOF).

## FOH and matched methods for automatic rate conversion in sITuner interface

When you use `system` to tune a Simulink model, block tuning is performed at the rate specified in the `sITuner` interface to the model (see the `Ts` property of `sITuner`). When you write the tuned values back to Simulink, if the sample time of the Simulink block differs from the sample time used for tuning, a rate conversion takes place automatically. You can use the `setBlockRateConversion` command to specify the rate conversion method.

In addition to the ZOH (zero-order hold) and Tustin methods, `setBlockRateConversion` now supports FOH (first-order hold) and matched methods. The matched method is only available for SISO blocks.

For more information, see the `setBlockRateConversion` reference page.

## Improved support for genss block parameterization in sITuner interface

When you assign a custom parameterization of a tunable block in an `sITuner` interface to a Simulink model, you can now use the `setBlockValue` command to set the value of a particular tunable element within the custom parameterization of the block. Similarly, you can now use `getBlockValue` to query the value of a tunable element within the block.

Previously, `setBlockValue` and `getBlockValue` could only set or query the current value of the entire parameterization of a Simulink block. There was no way to set or query the current value of a single tunable element of a custom parameterization.

---

For more information, see the reference pages for `setBlockValue` and `getBlockValue`.

## Support for additional multiplication modes in sITuner parameterization of Gain block

When you create an `sITuner` interface to a Simulink model that contains a Gain block with vector inputs and outputs, a parameterization is assigned to the block for any of the following multiplication modes of the block:

- `Element-wise (K.*u)`
- `Matrix (K*u)`
- `Matrix (u*K)`

(The multiplication mode is specified in the **Main** tab of the block dialog box.) Previously, the `sITuner` interface could only assign a parametrization for `Element-wise (K.*u)` and `Matrix (K*u)` modes.

You can now set parameter values for the Gain block in the `sITuner` interface using `setBlockValue` for all these multiplication modes. Previously, `setBlockValue(ST,blockID,value)` worked only for `Matrix (K*u)` mode.



# R2014a

**Version: 4.0**

**New Features**

**Bug Fixes**

**Compatibility Considerations**

## **sITuner interface for improved control system tuning of Simulink models with `systeme` or `looptune` functions, including tuning of gain-scheduled controllers (with Robust Control Toolbox)**

Use the new sITuner interface for tuning control systems in Simulink models. This interface replaces sITunable. The sITuner interface allows you to:

- Tune model blocks and subsystems to meet tuning goals using the `systeme` and `looptune` functions. (Both functions require a Robust Control Toolbox license.)
- Perform robust tuning of a controller against a set of plant models using `systeme`. You can configure an sITuner interface to vary model parameter values and operating points. When you call `systeme` for the interface, the software returns a controller that satisfies the tuning goals for all the specified model variations.
- Validate the controller design by examining the transfer function for relevant I/O sets using the `getIOTransfer`, `getLoopTransfer`, `getSensitivity`, and `getCompSensitivity` functions.

sITuner, similar in design to sILinearizer, simplifies I/O management in the controller tuning and validation workflow. You specify signals of interest as *analysis points*. You can use these analysis points to configure design requirements and specify linearization inputs/outputs when you extract transfer functions.

For more information on command-line tuning of Simulink models with sITuner, see [Command-Line Control System Tuning](#).

### **Compatibility Considerations**

The sITunable interface will continue to work for backward compatibility. However, only the sITuner interface will be supported and enhanced in future releases. Therefore, adoption of the sITuner interface is strongly recommended.

For documentation of the sITunable interface, see sITunable in the R2013b documentation.

### **Redesigned PID Tuner for improved PID tuning workflow**

The redesigned PID Tuner streamlines workflows for automatic and interactive tuning of PID controllers. Improvements include:

- 
- Additional options to import a plant into the PID Tuner. These options are especially useful if your Simulink plant model linearizes to zero. You can:
    - Relinearize the model using a simulation snapshot operating point that you find based on the controller error signal.
    - Identify an LTI plant model from simulated input-output data. This option requires a System Identification Toolbox™ license.

For more information about this option, see “PID controller tuning using system identification to model the plant from simulation input-output data in the PID Tuner” on page 7-3.

- Ability to compare the response of multiple plant models to the same controller. For example you can test the robustness of a controller to plant model uncertainty. Suppose you have tuned a controller for one plant model. You can import variations of the plant model into the PID Tuner. Then, you can plot the controller response for all the plant models in a single figure to compare the controller performance.
- Ability to display multiple response plots simultaneously. For example, you can tune a controller while simultaneously monitoring the reference tracking and output disturbance rejection plots.

To access the PID Tuner, in the PID block dialog box, click **Tune**.

For more information about using the PID Tuner, see PID Controller Tuning.

## **PID controller tuning using system identification to model the plant from simulation input-output data in the PID Tuner**

If you have System Identification Toolbox software, you can use the PID Tuner to identify a linear plant model from data obtained by simulating the Simulink model. You use the identified model to tune your PID Controller block. For example, suppose you want to tune the PID controller block in a model that contains a Triggered Subsystem block. The analytical block-by-block linearization algorithm does not support event-based subsystems and therefore the model linearizes to zero. Now, you can simulate the plant response for a specified input and use this simulated data to identify the plant model. The PID Tuner automatically tunes the PID controller for the identified model. You can then interactively adjust the performance of the tuned control system, and save the identified plant and tuned controller.

For an example showing how to use system identification to model a plant for PID tuning, see Interactively Estimate Plant from Measured or Simulated Response Data.

## Option to specify multiple substitute linearizations of a Simulink block for batch linearization

The method for specifying substitute linearizations for blocks and model subsystems is simplified. Also, you can specify multiple substitute linearizations for a block and obtain a linearization for each substitution (batch linearization). Use this functionality, for example, to study the effects of varying the linearization of a Saturation block on the model dynamics.

You can specify a substitute linearization as an input to the `linearize` command and as the `BlockSubstitutions` property of the `sLinearizer` and `sTuner` interfaces. Use a structure with the following fields:

- **Name** — Full blockpath, specified as a string. For example, 'scdenginectrlpidblock/valve timing'.
- **Value** — Substitute linearization, specified as one of the following:
  - Double, for example 1. Use for SISO models only. For models having either multiple inputs or multiple outputs, or both, use an array of doubles. For example, `[0 1]`. Each array entry specifies a linearization for the corresponding I/O combination.
  - LTI model, with I/Os that match the block specified by **Name**. For example, `zpk([], [-10 -20], 1)`.
  - Array of LTI models. For example, `[zpk([], [-10 -20], 1); zpk([], [-10 -50], 1)]`.

If you vary model parameter values, then the LTI model array size must match the grid size.

- Structure with fields: **Specification**, **Type**, and **ParameterNames**, **ParameterValues**. For an example, see [Specifying Linearization for Model Components Using System Identification](#).

Previously, the software supported only this method of specifying a substitute linearization.

To specify substitute linearizations for multiple blocks, create an array of the described structure.

This improvement is also applicable in the Simulink editor when you right-click a block and select **Linear Analysis > Specify Selected Block Linearization**. Select the



---

**Specify block linearization using one of the following** check box. When you select `MATLAB Expression` in the list, you can now specify a double, an array of doubles, an LTI model, or an array of LTI models in the text box.



# R2013b

**Version: 3.8**

**New Features**

**Bug Fixes**

**Compatibility Considerations**

## Enhanced `linearize` command, providing faster batch linearization for model parameter variations

`linearize` now allows you to efficiently batch linearize a model for variations in model parameter values. Use this feature to study the effects of varying a model parameter value. For example, you can vary plant parameter values and analyze the controller robustness to plant model uncertainty. The software uses only one model compilation, making batch linearization faster, especially for models that are expensive to compile repeatedly.

For more information, see [Batch Linearize Model for Parameter Value Variations Using `linearize` and Specify Parameter Samples](#).

## `sLinearizer` interface, providing faster batch linearization for multiple I/O sets

Use the `sLinearizer` interface to efficiently batch linearize a model. The interface includes linearization commands that you use to extract any open-loop or closed-loop transfer function for varying operating points and parameter values. For instance, you can extract linearizations for the plant model, overall closed-loop transfer function, and plant disturbance rejection, at multiple operating points, for varying plant/controller parameter values.

`sLinearizer` extends the functionality of `linearize` and uses only one model compilation to compute multiple open-loop and closed-loop transfer functions for varying operating points and parameter values. This efficiency is especially advantageous for models that are expensive to compile repeatedly.

For examples that illustrate how to use `sLinearizer`, see:

- [Vary Parameter Values and Obtain Multiple Transfer Functions Using `sLinearizer`](#)
- [Vary Operating Points and Obtain Multiple Transfer Functions Using `sLinearizer`](#).

## `linearizeOptions` and `findopOptions` for specifying options for linearization and operating point search

The new options commands `linearizeOptions` and `findopOptions` replace the command `linoptions`. This change simplifies the use of options for linearization and operating

---

point search by separating the available options for each task. See the reference pages for these options commands for more information about how to use them.

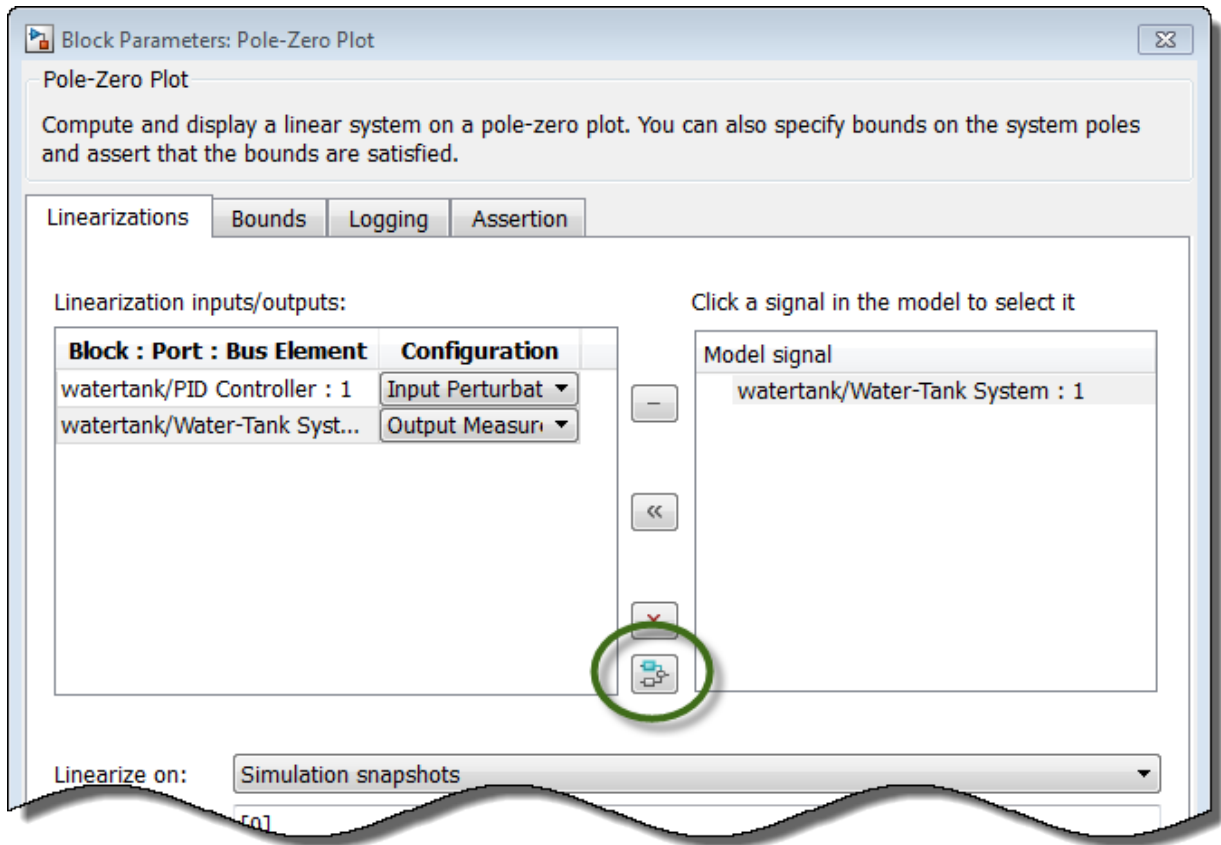
## Compatibility Considerations

The `linoptions` command will be removed in a future release, and issues a warning as of this release. If you have scripts or functions that use `linoptions`, consider updating them to instead use `linearizeOptions` for linearization and `findopOptions` for operating point search.

## Highlight linear analysis points in Linear Analysis Plot Blocks and Model Verification Blocks

The Linear Analysis Plot Blocks and Model Verification Blocks now include a button that highlights signals in your Simulink model which you have selected as linearization inputs or outputs. This highlighting makes it easier to identify linear analysis signals in your model when you are working in the block dialog box.

In the Linearizations tab of the block, select a signal from the **Linearization inputs/outputs** table. Then click the highlight button:



The block that originates the selected signal is highlighted in the Simulink model.

## **writeBlockValue** command can update Simulink model with tuned parameter values from generalized LTI model

The `writeBlockValue` method of `sITunable` has new syntaxes that allow you to update a Simulink model with tunable parameter values from a generalized LTI model.

`writeBlockValue(ST,M)` updates the current values of the tunable parameters in an `sITunable` interface, `ST`, to match their values in a generalized model, `M`. The command also updates the corresponding parameters in the Simulink model associated with `ST`.

---

`writeBlockValue(ST,M,BlockID)` updates only the parameters in the specified blocks.

These syntaxes are useful when you use Robust Control Toolbox tuning commands such as `systemtune`, `looptune`, or `hinfstruct` to tune system responses that you extract from the `sITunable` interface. For example, suppose you extract a particular closed-loop response from an `sITunable` interface using `getIOTransfer`. You then use `hinfstruct` to tune this closed-loop response. To validate the tuned parameters in the full non-linear Simulink model, use `writeBlockValue` to write the tuned parameter values back into the model.

For more information, see the `sITunable.writeBlockValue` reference page.

## Format of `BlockData` structure identical for snapshot and operating point linearization

When you specify the linearization of a block as a function, the input argument to that function is the structure `BlockData`, whose fields are given in the `linearize` reference page. The field `BlockData.Inputs` is now a data structure array, regardless of whether you are linearizing the model at a snapshot time or at a specified operating point. The data structure `BlockData.Inputs` has the following fields:

- **BlockName** — Contains the name of the block whose output connects to the input of the block whose linearization you specifying. For example, if you are specifying the linearization of a block called `Dynamics`, and the second input of `Dynamics` is driven by a signal from a block called `Torque`, then `BlockData.Inputs(2).BlockName` is the full block path name of `Torque`.
- **PortIndex** — Identifies which output port of `BlockName` corresponds to the input of the block whose linearization you are specifying. For example, if the third output from `Torque` drives the second input of `Dynamics`, then `BlockData.Inputs(2).PortIndex = 3`.
- **Values** — The value of the signal specified by `BlockName` and `PortIndex`. If this signal is a vector-valued signal, `Values` is a vector of corresponding dimension.

Previously, `BlockData.Inputs` had this format only for snapshot linearization. For linearization at a specified operating point, `BlockData.Inputs` was a numeric array of input values.

## Compatibility Considerations

If you have scripts or functions that assume `BlockData.Inputs` is a numeric array, update your code to reflect the new structure of `BlockData.Inputs`.

## Linear Analysis Blocks and Model Verification Blocks save operating points with computed linear systems

The data logging capability of the Linear Analysis Blocks or Model Verification Blocks now includes an option to save the operating points corresponding to the computed linear systems. When you check **Save data to workspace** and **Save operating points for each linearization** in the Logging tab of the block dialog box, a field named `operatingPoints` is added to the data structure containing the logged data. This field stores the operating point corresponding to each logged linear system in the data structure.



# R2013a

**Version: 3.7**

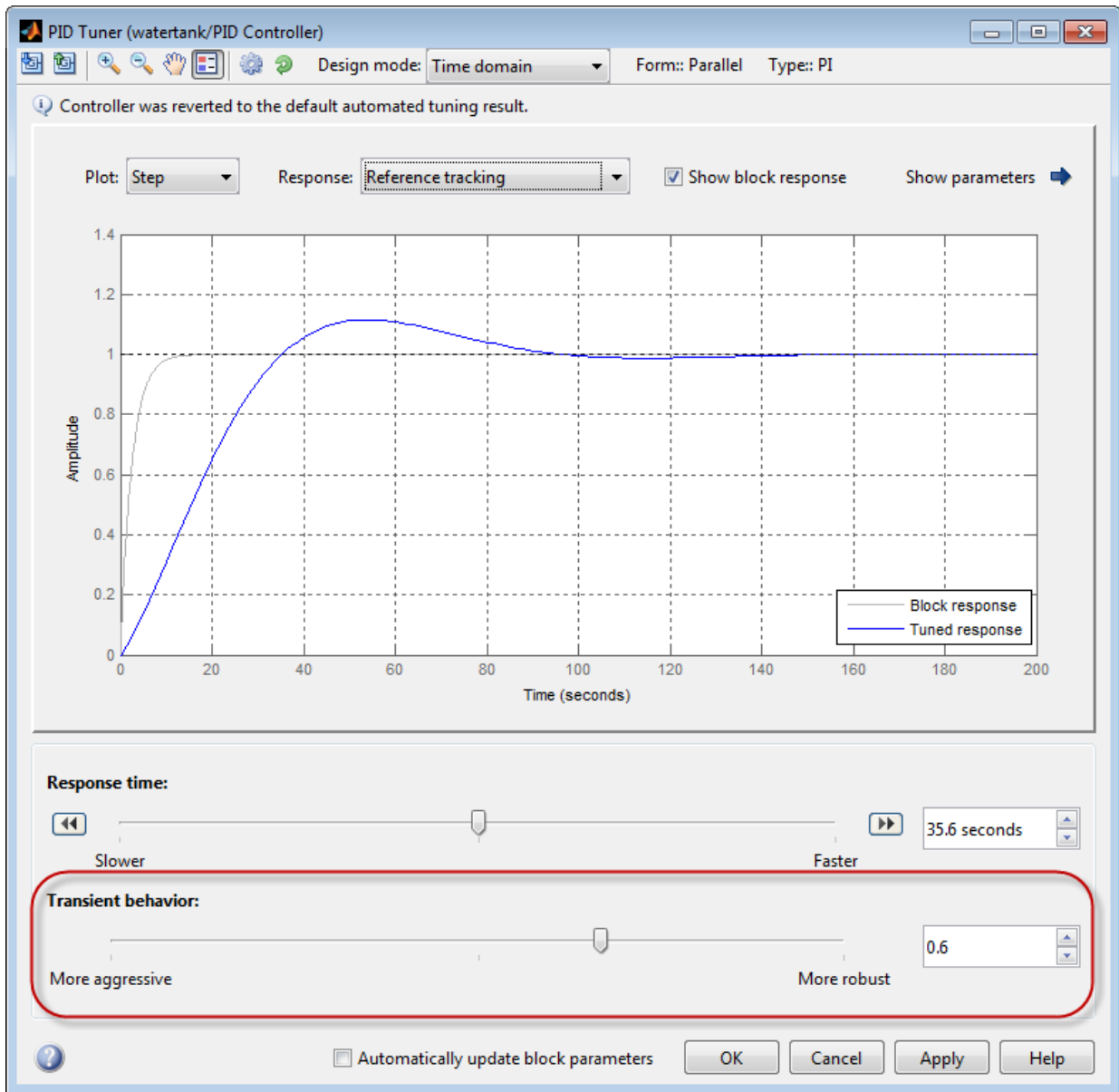
**New Features**

**Bug Fixes**

**Compatibility Considerations**

## **Transient behavior slider added to PID Tuner for increased control over reference tracking and disturbance rejection performance**

The PID Tuner now has a **Transient behavior** slider for emphasizing either reference tracking or disturbance rejection. When you open the PID Tuner, the tool starts in the **Time domain** design mode, displaying a step plot of the reference tracking response. The new **Transient behavior** slider is beneath the **Response time** slider.



You can use the **Transient behavior** slider when:

- The tuned system's disturbance rejection response is too sluggish for your requirements. In this case, try moving the **Transient behavior** slider to the left to make the controller more aggressive at disturbance rejection.
- The tuned system's reference tracking response has too much overshoot for your requirements. In this case, try moving the **Transient behavior** slider to the right to increase controller robustness and reduce overshoot.

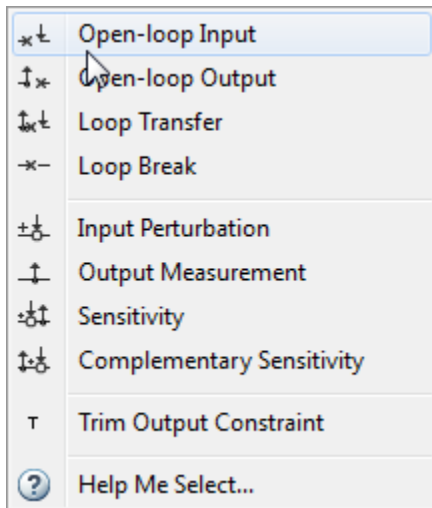
In Frequency domain design mode, the PID Tuner has **Bandwidth** and **Phase margin** sliders. These sliders are the frequency-domain equivalents of the **Response time** and **Transient behavior** sliders, respectively.

For an example illustrating the balance between reference tracking and disturbance rejection, see Tune PID Controller to Balance Tracking and Disturbance Rejection Performance.

## Linear analysis points redesigned to clarify I/O types and loop openings

Linear analysis points (linearization inputs, outputs and loop-opening locations), that you use to specify the portion of the Simulink model to linearize, have been redesigned. The redesign clarifies what the I/O point means and helps you select the right type of point to compute the desired response.

The analysis points that you select in the **Linear Analysis Points** submenu or the Linear Analysis Tool have been renamed. The points also have updated icons that correspond to markers that appear on your model to indicate the linear analysis point type. The corresponding strings for the **type** argument in the `linio` command have also been renamed.



You can also specify inputs or outputs as open loop while specifying the I/O type. For example:

- In the **Linear Analysis Points** submenu or the Linear Analysis Tool, select **Open-loop Output**.
- At the command line, type `io(2)=linio('magball/Magnetic Ball Plant',1,'openoutput')`.

Thus, when performing tasks such as plant linearization or open-loop linearization, you do not need to specify the opening separately.

The following table maps the renamed analysis points to the point types in previous releases:

Command Line		Linear Analysis Points Submenu or Linear Analysis Tool	
Previous Releases	R2013a	Previous Releases	R2013a
<code>type = 'in' and openloop = 'on'</code>	<code>type = 'openinput'</code>	<b>Input Point and Open Loop</b>	<b>Open-loop Input</b>
<code>type = 'in' and openloop = 'off'</code>	<code>type = 'input'</code>	<b>Input Point</b>	<b>Input Perturbation</b>

Command Line		Linear Analysis Points Submenu or Linear Analysis Tool	
Previous Releases	R2013a	Previous Releases	R2013a
type = 'out' and openloop = 'on'	type = 'openoutput'	<b>Output Point and Open Loop</b>	<b>Open-loop Output</b>
type = 'out' and openloop = 'off'	type = 'output'	<b>Output Point</b>	<b>Output Measurement</b>
type = 'inout' and openloop = 'off'	type = 'sensitivity'	<b>Input-Output</b>	<b>Sensitivity</b>
type = 'outin' and openloop = 'on'	type = 'looptransfer'	<b>Output-Input and Open Loop</b>	<b>Loop Transfer</b>
type = 'outin' and openloop = 'off'	type = 'compsensitivity'	<b>Output-Input</b>	<b>Complementary Sensitivity</b>
type = 'none' and openloop = 'on'	type = 'loopbreak'	<b>Open Loop</b>	<b>Loop Break</b>

Use the context-sensitive help, or see the `linio` reference page to assist you in selecting an analysis point. For examples on how to specify linear analysis points graphically, see `Specify Portion of Model to Linearize in Simulink Model` and `Specify Portion of Model to Linearize in Linear Analysis Tool`.

## Linear Analysis Blocks and Model Verification Blocks save data in single object with Simulink model logging output

The data logging option in the Linear Analysis Blocks or Model Verification Blocks saves linear systems computed by the blocks. As of R2013a, when you configure your Simulink model to save simulation output in a single object, the logging output of these blocks is saved as a field in the simulation output object. Previously, these blocks saved logging output as a separate variable in the MATLAB workspace, regardless of the model configuration settings.

For more information about data logging in Simulink, see `Export Simulation Data` and the `Simulink.SimulationOutput` reference page.

---

## Compatibility Considerations

If you have a Simulink model which you have configured to save simulation output in a single object, and you use data logging option in a Linear Analysis Block or Model Verification Block, the resulting logging output is no longer a separate variable in the MATLAB workspace. If you use the logging output in any scripts, update those scripts to use the `Simulink.SimulationOutput` object instead of a separate variable.





# R2012b

**Version: 3.6**

**New Features**

**Bug Fixes**

**Compatibility Considerations**

## **MATLAB code generation from Linear Analysis Tool for batch estimation of model frequency responses**

You can now generate MATLAB code for frequency response estimation from the Linear Analysis Tool. You can generate either a MATLAB script or a MATLAB function. Generated MATLAB scripts are useful when you want to programmatically reproduce a result you obtained interactively. A generated MATLAB function allows you to perform multiple estimations with systematic variations in estimation parameters such as operating point (batch estimation).

For more information, see [Generate MATLAB Code for Repeated or Batch Frequency Response Estimation](#).

## **Operating point calculation (trimming) from multiple specifications with only one model compilation**

The `findop` command now can find operating points for multiple operating point specifications with a single model compilation. This feature allows you to find multiple trimmed operating points without the overhead of compiling the model for each trimming computation. For an example, see [Batch Compute Operating Points with Single Model Compilation](#).

For more information about operating point calculation and operating point specifications, see the `findop` and `operspec` reference pages.

## **Export and import operating point specifications in Linear Analysis Tool**

When you modify an operating point specification in the Linear Analysis Tool, you can now export the specification to the MATLAB workspace. Exported specifications are saved as operating point specifications objects (see `operspec`). Exporting specifications can be useful when you expect to perform multiple trimming operations using the same or a very similar set of specifications. Additionally, you can export interactively-edited operating point specifications when you want to use the `findop` command to perform multiple trimming operations with a single compilation of the model (see “Operating point calculation (trimming) from multiple specifications with only one model compilation” on page 10-2).

You can also import saved operating point specifications to the Linear Analysis Tool and use them to interactively compute trimmed operating points. Importing a specification

---

can be useful when you want to trim a model to a specification that is similar to one you previously saved. In that case, you can import the specification to the Linear Analysis Tool and interactively change it. You can then export the modified specification, or compute a trimmed operating from it.

For more information, see [Import and Export Specifications For Operating Point Search](#).

For more information about operating point specifications, see the [operspec](#) and [findop](#) reference pages.

## **MATLAB script or function generation from Linear Analysis Tool for repeated or batch linearization**

When you generate MATLAB code for linearization from the Linear Analysis Tool, you now have a choice of generating a script that uses the current linearization parameters, or a function that takes parameter values as input. Previously, you could only generate a MATLAB function with no input parameters.

Generated MATLAB scripts are useful when you want to programmatically reproduce a result you obtained interactively. A generated MATLAB function allows you to perform multiple linearizations with systematic variations in linearization parameters such as operating point (batch linearization).

For more information, see [Generate MATLAB Code for Repeated or Batch Linearization](#).

## **Print plots to MATLAB figure in Linear Analysis Tool**

You can now export a plot from the Linear Analysis Tool to a MATLAB figure window.

For more information, see [Print Plot to MATLAB Figure in Linear Analysis Tool](#).

## **Commands for setting and querying rate conversion methods in tunable blocks**

New commands allow you to query and specify the rate conversion method that the `slTunable` interface uses for converting the sampling time of the parametrization of tunable blocks in a Simulink model.

- `slTunable.getBlockRateConversion` — Query rate conversion method of tunable block

- `sITunable.setBlockRateConversion` — Set rate conversion method of tunable block

These commands are useful, for example, when you use the `sITunable` interface to tune a mixed-rate Simulink model, such as a model having a discrete-time controller and a continuous-time plant. The `sITunable` interface automatically converts the sampling times of tunable blocks where necessary. Using `sITunable.setBlockRateConversion`, you can control the conversion method that the `sITunable` interface uses.

For more information, see:

- Tuning of a Digital Motion Control System
- `sITunable.getBlockRateConversion` and `sITunable.setBlockRateConversion` reference pages

## **“Ignore saturation when linearizing” checked by default in PID Controller and PID Controller (2DOF) blocks**

The default linearization behavior of the PID Controller and PID Controller (2DOF) blocks now forces linearization commands to ignore block output limits. Ignoring output limits allows you to linearize a model around an operating point even if that operating point causes the PID Controller block to exceed the output limits.

To cause linearization commands not to ignore block output limits, clear the **Ignore saturation when linearizing** checkbox in the block dialog box, PID Advanced tab.

## **showBlockValue renamed to showTunable**

The command `showBlockValue` is now called `showTunable`. Use `showTunable` to obtain the current value of block parametrizations from an `sITunable` interface.

## **Compatibility Considerations**

Replace instances of `showBlockValue` in your code with `showTunable`.

# R2012a

**Version: 3.5**

**New Features**

**Bug Fixes**

## Create Linearization Input/Output Sets in the Linear Analysis Tool

You can now interactively create I/O sets for linearization or frequency response estimation in the Linear Analysis Tool, without adding linearization points to your model. Previously, you had to modify your model to interactively create I/O sets.

For more information, see [Create Linearization I/O Sets In Linear Analysis Tool](#).

For more information about using the Linear Analysis Tool, see [Linearize at Model Operating Point and Estimating Frequency Response](#).

## Specify Feedback Sign for `getLoopTransfer` Without Specifying Loop Openings

An additional syntax for `slTunable.getLoopTransfer` allows you to specify the feedback sign without having to provide an `openings` argument. Use the syntax

```
L = getLoopTransfer(ST,location,sign)
```

to calculate the point-to-point open-loop transfer function at `location` of the Simulink model described by the `slTunable` interface `ST`. The software uses the sign convention specified by `sign` (+1 or -1) to compute `L`.

Previously, to specify a feedback sign, you had to use the syntax

```
L = getLoopTransfer(ST,location,openings,sign)
```

This syntax required you to specify an `openings` argument to specify `sign`. The new syntax allows you to specify `sign` while using the default loop openings.

For more information, see the `slTunable` and `slTunable.getLoopTransfer` reference pages.

# R2011b

**Version: 3.4**

**New Features**

**Bug Fixes**

**Compatibility Considerations**

## Redesigned Graphical Tool for Improved Linear Analysis Workflows

The new interactive Linear Analysis Tool streamlines workflows for linear analysis tasks such as finding steady-state operating points (trimming) and linearization.

To access the Linear Analysis Tool:

- 1 Open a Simulink model.
- 2 In the Simulink model, select **Tools > Control Design > Linear Analysis**.

For more information about using the Linear Analysis Tool for operating-point and linearization workflows, see:

- Steady-State Operating Points
- Linearization

## Interactive Frequency Response Estimation and Validation of Linearization Results

The new interactive Linear Analysis Tool provides a graphical interface for frequency response estimation.

To access the Linear Analysis Tool:

- 1 Open a Simulink model.
- 2 In the Simulink model, select **Tools > Control Design > Linear Analysis** to launch the Linear Analysis tool.
- 3 Click the Frequency Response Estimation tab to begin an estimation task.

For more information about using the Linear Analysis Tool for frequency response estimation, see Frequency Response Estimation.

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

If you have Simulink Design Optimization™ software, you can optimize the Simulink model to meet frequency-domain requirements specified in Model Verification blocks.



---

For example, you can optimize the linear system to meet Bode magnitude or gain and phase margin requirements. For more information, see [Design Optimization To Meet Frequency-Domain Requirements \(GUI\)](#).

You can also include time-domain requirements such as step response characteristics for optimization. For more information, see [Design Optimization to Meet Time- and Frequency-Domain Requirements](#).

## **Automatic Tuning of PID Controller Blocks in a Referenced Model**

You can now use the PID Tuner to tune a PID Controller block in a model that is referenced in one or more open models. When you launch the PID Tuner from a block within a model reference, the software prompts you to select which of the open models is the top-level model for analysis and tuning. Previously, the PID Tuner could only use the model containing the PID Controller block as the top-level model.

For more information, see [Tuning a PID Controller Within a Model Reference](#).

## **Control System Tuning for Simulink Models with `looptune` or `hinfstruct` Using `sITunable` Interface**

If you have Robust Control Toolbox software, you can use tuning commands, such as `sITunable.looptune` and `hinfstruct`, to tune control systems modeled in Simulink. The `sITunable` object provides an interface between your Simulink model and these commands.

Use `sITunable` to specify information about your control structure and parametrization. `sITunable` also automates tasks such as linearizing the Simulink model, parametrizing the tunable blocks of your system, and applying tuned parameter values to the model. After you create and configure an `sITunable` object for your control architecture, you can tune the control system using `sITunable.looptune` or `hinfstruct`.

For more information, see [Tuning Fixed Control Architectures in the Robust Control Toolbox documentation](#).

## **Change in Default Number of Samples in `frest.Chirp`**

The default number of samples (`NumSamples`) in a `frest.Chirp` input signal for frequency response estimation is now given by the formula:

$$\frac{4\pi}{Ts * \min(\text{FreqRange})}$$

`Ts` is the sampling time of the chirp signal. `FreqRange` is the vector of signal frequencies of the chirp signal.

This formula returns twice the value returned in previous releases.

For example, if you create a default chirp input signal with the command

```
input = frest.Chirp
```

the value of `input.NumSamples` is 10000, instead of the previous value of 5000.

For more information, see the `frest.Chirp` reference page.

## Compatibility Considerations

If you have scripts that rely on the default `NumSamples` formula, modify your scripts to account for the new value.

# R2011a

**Version: 3.3**

**New Features**

**Bug Fixes**

## Ability to Select Individual Bus Elements as Linearization Input and Output Points

Instead of selecting an entire bus, you can now select individual elements of the bus signal as linearization inputs or outputs (I/Os). By selecting individual bus elements, you can:

- Obtain the linearization only for the channels of interest.
- Specify multiple I/Os, possibly as different types, in the same bus.

Select individual bus elements as I/Os when you want to:

- Linearize a Simulink model using the GUI or programmatically.
- Visualize time- and frequency-domain response during simulation using the Linear Analysis Plots blocks.
- Verify frequency-domain characteristics using the Model Verification blocks.

For more information, see [Select Individual Bus Elements as Linearization Points](#) and the [linio](#) reference page.

## Enhanced LINLFT Command Optionally Returns Linearization of Excluded Blocks

A new optional output argument to `linlft` returns the linearization of blocks you specify for exclusion from the linearized model.

For more information, see the [linlft](#) reference page.

## Access to Current Linearization of a Simulink Block for Specifying Custom Linearization

When you use a specification function to specify a custom linearization of a Simulink block, you can now access the current linearization in your function. The current linearization is stored in the new `BlockLinearization` field of the `BlockData` structure. The software automatically creates the `BlockData` structure and passes it to your block linearization specification function.

For more information and an example showing how to use the `BlockLinearization` field of `BlockData`, see [Augmenting the Linearization of a Block](#).

---

## **Enhanced PID Controller Blocks Display Compensator Formula in Block Dialog Box**

The PID Controller and PID Controller (2DOF) blocks now display the current compensator formula in the block dialog box. This display reflects the current settings for controller type, controller form, and time domain.



# R2010b

**Version: 3.2**

**New Features**

**Bug Fixes**

## New Blocks for Plotting and Verifying Linear System Characteristics of Simulink Models

This version of Simulink Control Design software provides new blocks for:

- “Plotting Linear System Characteristics of Simulink Models” on page 14-2.
- “Verifying Linear System Characteristics of Simulink Models” on page 14-2.

### Plotting Linear System Characteristics of Simulink Models

Simulink Control Design software provides six new blocks in the **Linear Analysis Plots** library. Use these blocks to plot the time- and frequency-domain characteristics of a linear system computed from a nonlinear Simulink model. The linear system is computed and plotted during simulation at:

- Simulation snapshot times. The default snapshot time is 0.
- Trigger-based simulation events.

You can also use these blocks to specify bounds on the linear system characteristics, and view the bounds on the plot.

Because these blocks are same as the Model Verification blocks except for the default settings of the bound parameters, you can optionally use the blocks to verify that the bounds are satisfied during simulation.

For more information, see:

- Visualize Bode Response of Simulink Model During Simulation
- Linear Analysis Plots block reference pages
- Plotting Linear System Characteristics of a Chemical Reactor demo

### Verifying Linear System Characteristics of Simulink Models

Simulink Control Design software provides six new blocks in the **Model Verification** library. Use these blocks to verify that the time- and frequency-domain characteristics of a linear system, computed from a nonlinear Simulink model, satisfy specified bounds during simulation. For example, you can verify whether the linearized behavior of your model satisfies upper and lower magnitude bounds on a Bode plot or gain and phase margins.



---

You can perform the verification during simulation at:

- Simulation snapshot times. The default snapshot time is 0.
- Trigger-based simulation events

Because these blocks are same as the Linear Analysis Plots blocks except for the default settings of the bound parameters, you can use the blocks to view the bounds on time- and frequency-domain plots.

You can also use these blocks with the Simulink Model Verification library blocks to design 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:

- Model Verification in the Simulink Control Design documentation.
- Model Verification block reference pages
- Verifying Frequency-Domain Characteristics of an Aircraft demo

## **New Tools for Identifying Time-Varying Source Blocks for Frequency Response Estimation**

When you are performing frequency response estimation, time-varying source blocks in the signal path can skew the results. Previously, to obtain accurate estimation, you manually identified source blocks and inserted open-loop points in your model to disable them.

The new `frest.findSources` command automatically detects time-varying source blocks in the signal path of the output linearization points in your Simulink model. Additionally, a new option to the `frestimate` command, `BlocksToHoldConstant`, allows you to disable these blocks during frequency response estimation.

For more information, see the `frest.findSources` and `frestimateOptions` reference pages.

There is also a new Model Advisor check for detecting time-varying source blocks in your model. For more information about using the Model Advisor, see Consulting the Model Advisor in the *Simulink User's Guide*.

## Tuning Tools Update Workspace Variables That Define Parameters of Tuned Blocks

When you use Simulink Control Design tools to tune a block with parameters defined as workspace variables, the software now updates the values of the workspace variables. This update occurs when you write the compensator design to your Simulink model. The block remains parametrized by the variables. The software can update variable parameters that represent numerical values or `Simulink.Parameter` objects, whether they reside in the base workspace or the model workspace.

Previously, tuning a block parametrized by workspace variables overwrote the block parameters with numerical values. Thus, such tuning did not change the value of the workspace variables.

## Enhanced PID Tuner Including New Response Plots

This release introduces several enhancements to the PID Tuner, including:

- New response plot options let you analyze system responses in either time domain (step plot) or frequency domain (Bode plot). Available responses are reference tracking, input and output disturbance rejection, controller effort, open loop, and plant.
- New options when the plant linearizes to zero at the operating point defined in the Simulink model.

When the plant linearizes to zero on launch of the PID Tuner, the PID Tuner provides an option to linearize at a different operating point. The PID Tuner also lets you import an LTI model of your plant, such as a model obtained by frequency response estimation. Previously, the PID Tuner did not launch when the plant linearized to zero.

For an example illustrating these options, see the new Simulink Control Design demo, *Designing PID Controller in Simulink with Estimated Frequency Response*.

For more information about using the PID Tuner, see Automatic PID Tuning in the *Simulink Control Design User's Guide*.

---

## **New Demo Illustrating Control Design for a Plant That Has Parameter Variations**

The Reference Tracking of a DC Motor with Parameter Variations demo illustrates compensator design for a plant that has parameter variations. The demo shows the following techniques:

- Performing batch linearization to obtain multiple linear models from a single Simulink model
- Using the new SISO Design Tool ability to analyze a control design for multiple models simultaneously



# R2010a

**Version: 3.1**

**New Features**

**Bug Fixes**

**Compatibility Considerations**

## New Parallel Computing Support For Frequency Response Estimation

If you have the Parallel Computing Toolbox™ software installed, you can use parallel computing to speed up frequency response estimation.

For more information, see *Speeding Up Estimation Using Parallel Computing* in the Simulink Control Design documentation.

## New Commands Support Recomputing Frequency Response Estimation Results at Specific Frequencies

If you use the `sinestream` input signal for estimation, you can now recompute your frequency response estimation for only those specific frequencies that do not reach steady state. Previously, you had to redo the entire estimation if a few frequencies needed recomputing.

For more information, see *Time Response Not at Steady State*.

## New `frest.simcompare` Output Argument Returns Simulation Output Data From Linear System

This release provides enhanced `frest.simCompare` functionality, which allows the return of simulated output data. If your linear model uses state-space representation, you can also return the state vector.

For more information, see the `frest.simCompare` reference page.

## New Options in Simulink Results Viewer GUI for Viewing Frequency Response Estimation Results

The Simulation Results Viewer GUI, which you display using `frest.simView`, now provides additional options for analyzing your frequency response estimation:

- You can now enable phase unwrapping in the frequency response plot. To unwrap phase, right-click the Bode plot, and select **Unwrap Phase**.
- You can also import frequency response estimation results into an open Simulation Results Viewer, which replaces the existing results. Previously, you had to open a new Viewer each time you viewed estimation results. To import results, select **File > Import** in the Simulation Results Viewer.

---

## New Option for Labeling Bus Signal I/O Names in the SISO Design Task

You can now configure the SISO Design Task to automatically label model I/Os using bus signal names. Model I/O labels describe available loops in the Graphical Tuning tab and label response plots in the Analysis Plots tab of the SISO Design Task GUI. Previously, labels for bus signals were based on the block and port path, or the Simulink signal name.

To use this option, select **Tools > Options** in the SISO Design Task GUI, and enable **Use bus signal names to label model I/O**.

## Existing Simulink Blocks Now Have Analytic Jacobians

The following Simulink blocks now have Analytic Jacobians, which are optimized for memory consumption in large-scale models:

- Switch
- Multiport Switch
- Math Function (Transpose and Hermitian Transpose)

For more information on each block, see the block reference pages.

## Change in Format of Time Series in `frestimate` Output

When you use the `simout` output of the `frestimate` command, you obtain a cell array of `Simulink.Timeseries` objects. Previously, the time dimension was always the first dimension of each `Simulink.Timeseries` object.

As of R2010a, the time dimension of each `Simulink.Timeseries` in `simout` is:

- The first dimension, if the time series is 2-D
- The last dimension, if the time series is 3-D or higher

## Compatibility Considerations

If you have scripts that run `frestimate` on a model having an output point on a vector or matrix signal and perform operations on the resulting 3-D or higher-dimensional `Simulink.Timeseries` object in `simout`, modify your scripts to reflect the new placement of the time dimension in the time series.

For additional information, see:

- `frestimate` reference page
- `Simulink.Timeseries` reference page



# R2009b

**Version: 3.0**

**New Features**

**Bug Fixes**

## **New GUI for Tuning New PID Controller Blocks**

You can automatically tune the new Simulink PID Controller blocks in the PID Tuner GUI R2009b introduces. You launch the PID Tuner directly from the PID block dialogs boxes. These blocks are available in the Continuous and Discrete Simulink libraries.

For more information on tuning PID Controller using the PID Tuner, see [Automatic PID Tuning](#).

## **New Automated PID Tuning Algorithm**

You can now tune compensators using a new automated PID tuning algorithm called Robust Response Time, which is available in the SISO Design Task. You specify the open-loop bandwidth and phase margin, and the software computes PID parameters to robustly stabilize your system.

For information on tuning compensators using automated tuning methods, see [Completing the Design](#).

## **Ability to Compute Frequency Response of Simulink Models**

You can now compute the frequency response of a Simulink model using simulation. You use new commands to easily create input signals and simultaneously simulate and estimate the frequency response without changing your model. You can also use this capability to validate the accuracy of exact linearizations.

## **Ability to Specify the Linearization of Simulink Blocks**

You can now specify the linearization of Simulink blocks, subsystems, and model references without having to replace any block in your model. You can specify the linearizations as LTI models or Robust Control Toolbox uncertain models.

## **Ability to Design Compensators for Plant Models With Time Delays**

You can now design compensators for plants with exact time delay representations. Previously, you had to specify a Padé approximation before designing compensators.

For more information on designing compensators for plants with time delays, see [Designing Compensators for Plants with Time Delays](#).

---

## **New Commands to More Efficiently Compute Multiple Linearizations**

You can now more efficiently compute multiple linearizations for large models when only a few blocks or model references change per linearization. You linearize the fixed portion of the model once using `linlft` and linearize the varying portion multiple times. Then, you combine the results using `linlftfold` to obtain linearizations equal to those you would receive if you linearized the entire model multiple times.

For more information, see the `linlft` and `linlftfold` reference pages.

## **Ability to Set Default Plot Type for Linear Analysis Results from GUI**

You can now set the default plot type for viewing linear analysis results computed in the Control and Estimation Tools Manager. This setting applies to all future Control and Estimation Tools Manager sessions.

To set this option in the Simulink Control Design Preferences, select **File > Preferences** in the Control and Estimation Tools Manager.



# R2009a

**Version: 2.5**

**New Features**

**Bug Fixes**

## **Ability to Generate MATLAB Code from the GUI for Creating Operating Points and Linearizing Models**

You can now generate MATLAB code reflecting the configuration in the GUI when creating operating points and linearizing models.

## **Ability to Tune Additional Blocks**

You can now tune the following blocks:

- Blocks that you discretized using the Simulink Model Discretizer
- Blocks in the Simulink Extras library that specify initial states or outputs

For information about how to tune these blocks, see [Selecting Blocks to Tune](#) in the Simulink Control Design documentation.

## **New Option for Labeling Bus Signal I/O Names in Linearization Results**

You can now compute linear models that show the bus signal names for linearization I/O points located on buses. You can select this option using the following:

- Linearization Options GUI
- `linoptions` command

# R2008b

**Version: 2.4**

**New Features**

**Bug Fixes**

## **New Upsampling Option for Rate Conversion When Linearizing Simulink Models**

Version 2.4 includes an upsampling rate conversion method for linearization. This method upsamples discrete-time LTI systems at any sampling rate that is an integer-value-times faster than the sampling rate of the original system. You can select the upsampling rate conversion method in the following ways:

- Using the linearization options GUI
- From the command line using `linoptions`

For more information on the upsampling rate conversion method, see the `linoptions` reference page.

## **Ability to Specify State Order of Linearized Models from the Command Line**

You can now specify the order of the states in your linearized model directly from the command line using the `linearize` command. Previously, you could only specify state order using the GUI.

For more information on specifying state order from the command line, see the `linearize` reference page.

## **Ability to Filter the Linearization Inspector to Show Blocks in the Linearization Path**

You can now filter the list of blocks in the Linearization Inspector to show only the blocks in the linearization path. This filtering makes it easier for you to find blocks in the linearization path that you want to inspect.

## **Ability to Disable the Calculation of Linearization Diagnostics and Inspector Data in the GUI**

You can now disable the calculation of the linearization inspector and diagnostics information when you linearize using the GUI. This capability allow you to choose when you want to calculate diagnostic information.



# R2008a

**Version: 2.3**

**New Features**

**Bug Fixes**

## **New Diagnostic Messages Help You Troubleshoot Linearization Results**

You can now view diagnostic messages for your linearized model that help you diagnose and troubleshoot linearization results. These messages identify blocks in your model that encounter the following block issues during linearization:

- Blocks that have been marked as not supported for linearization
- Blocks with linearization configuration warning messages
- Blocks without pre-programmed exact Jacobian that linearize using numerical approximation

## **Ability to Find Operating Points for Simscape Models**

You can now find operating points for models that include Simscape™ and SimHydraulics® blocks using the Simulink Control Design `findop` command.

For more information on finding operating point for Simscape Models, see the Simscape documentation.

## **Updated Error and Warning Message System**

The Simulink Control Design error and warning IDs and messages have been updated. If you use error and warning IDs in your code, you must update your code to reflect the new IDs.

# R2007b

**Version: 2.2**

**New Features**

**Bug Fixes**

## **Ability to Linearize Models with Model-Reference Blocks by Any Linearization Method**

You can now perform any type of linearization for models containing Model blocks that reference other Simulink models.

Previously, you could only perform numerical perturbation linearization for models containing Model blocks. Now, you can also perform block-by-block linearization for such models when you set the simulation mode of the Model blocks to Normal.

## **Ability to Design Compensators for Models Containing Model-Reference Blocks**

You can now design compensators for tunable blocks inside external models referenced by your model. Your model references such external models by using Model blocks. You can also update the tunable blocks in the external model with the new compensator designs.

To view and select tunable blocks for compensator design from the referenced model, set the simulation mode of the Model block to Normal.

For more information about the types of blocks that you can tune, see [Selecting Blocks to Tune](#) in the Simulink Control Design documentation.

## **Ability to Generate Linearized Models with Exact Time-Delay Representations**

You can now use the Simulink Control Design software to compute linearized models with exact time-delay representations. Time delays in the original nonlinear model can result from any of the following blocks:

- Transport Delay
- Variable Time Delay
- Variable Transport Delay
- Unit Delay
- Integer Delay

Previously, you could only achieve approximate linearizations of models with continuous time delays using a Padé approximation.

---

## **Ability to Linearize Periodic Function-Call Subsystems**

You can now linearize periodic function-call subsystems with a constant sample time as discrete subsystems. You must set the sample time of the function-call trigger block to equal the sample time of the function-call generator block.

For more information on function-call subsystems, see [Function-Call Subsystems](#) in the Simulink documentation.



# R2007a

**Version: 2.1**

**New Features**

**Bug Fixes**

## Ability to Linearize Using an Operating Point Specified Directly in a Model

You can use the Simulink Control Design software to linearize around any operating point that you specify directly in a Simulink model. This capability allows you to make changes to a model and then perform a linearization around the newly specified operating point with the click of a single button. Previously, this same task required two steps: creating a new operating point and then linearizing around this operating point.

There are two ways to linearize around an operating point specified directly in the model:

- From the Control and Estimation Tools Manager
- Using `linearize`

## Ability to Capture Linearization Snapshots in GUI

You can use the Control and Estimation Tools Manager to linearize at snapshots of your model operating point at the following simulation points:

- Specified simulation times, such as when the simulation reaches a steady state solution
- Events during a specified simulation interval

As in prior releases, you can use `linearize` to perform linearization at snapshots in your model operating point, as described in `linearize` in the reference pages.

## Ability to Perform Control Design at Snapshots in GUI

You can use the Control and Estimation Tools Manager to perform control design at snapshots of your model operating point at the following simulation points:

- Specified simulation times, such as when the simulation reaches a steady state solution
- Events during a specified simulation interval

## Ability to Retrieve Stored Compensator Designs

When you design a compensator using the SISO design tool, you can store the current design and then continue making changes to this design. A new button called **Retrieve**



---

**Design** lets you retrieve the stored design at any time by undoing the design changes you made since you last stored the design. For more information on retrieving stored compensator designs, see *Storing and Retrieving Designs* in the Simulink Control Design documentation.



# R2006b

Version: 2.0.1

Bug Fixes



# R2006a

**Version: 2.0**

**New Features**

**Bug Fixes**

## Compensator Design in Simulink Is Now Supported

This release provides new tools to streamline the workflow for designing Single-Input Single-Output (SISO) control loops directly in Simulink. In previous releases, designing a compensator was a multistep process that involved several tools.

The new tools support any linearizable control architecture, such as single loops, multiple loops, and cascaded loops. With the new tools, you simply select the blocks you want to tune. Then, the Simulink Control Design software automatically analyzes your model to identify the relevant control loops and opens a preconfigured session of the SISO Design Tool (in the Control System Toolbox™ software). For more information, see “Designing Compensators” in the Simulink Control Design documentation.

Supported tunable SISO Simulink blocks include Gain, Transfer Function, Zero-Pole-Gain, State-Space, and PID blocks.

In the SISO Design Tool you can

- Graphically tune multiple SISO loops in a single GUI.
- Gain visual insight into loop interactions and coupling effects.
- Focus the analysis on a specific loop in a multiloop design by removing the effect of other feedback loops.
- Generate first-cut compensator designs using systematic design algorithms, such as Ziegler-Nichols PID tuning, IMC design, or LQG design.
- Optimize linear responses to meet time and frequency-based design constraints (requires the Simulink Response Optimization™ software).
- Directly tune Simulink block parameters, such as PID gains, zero-pole-gain representations, and masked blocks.
- Tune continuous or discrete control loops.
- Examine the closed-loop response of any portion of a model.

After a design is completed, you can write the tuned parameter values back to your model for verification with the full nonlinear system.

# R14SP3

**Version: 1.3**

**New Features**

**Bug Fixes**

## **Control and Estimation Tools Manager Enhanced**

You can copy and edit operating points within the Control and Estimation Tools Manager. .

You can initialize a model for simulation using operating points from within the Control and Estimation Tools Manager.

## **Support for Operating Point Search and Linearization of Models with Model Reference Blocks**

You can linearize and compute operating points for models that reference other models using the Model block. Linearization of model reference models must use the numerical perturbation linearization algorithm. This algorithm accepts state and input perturbation values in the form of an operating point object.



# R14SP2

**Version: 1.2**

**New Features**

**Bug Fixes**

## Context-Sensitive Help Added

Access context-sensitive help for the Linearization, Operating Point Search, and Linearization State Ordering Options window of the Control and Estimation Tools Manager. To access help on a field within the options window, right-click the option's label and select **What's this?** from the context menu. Help for the option will appear in this window.

## View Linearizations in the Control and Estimation Tools Manager

You can view state space, transfer function, and zero-pole gain representations of linearized models within the Control and Estimation Tools Manager without exporting to the workspace. These linearized models appear in the linearization summary pane of the Control and Estimation Tools Manager.

## Discretization Methods Added

You can select from three different discretization methods for linearization of multirate and hybrid models.

## List of Blocks with Preprogrammed Analytic Jacobians Added

You can view a complete list of blocks indicating which blocks have preprogrammed analytic Jacobians for use with the block-by-block analytic linearization algorithm.

## Block Name Readability Improved

You can use either truncated block names or full block names in the state space matrices of a linearized model, and within the LTI Viewer, to improve readability.