Control System Toolbox™ Release Notes

How to Contact MathWorks

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

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

508-647-7000 (Phone)

508-647-7001 (Fax)

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

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

Control System Toolbox™ Release Notes

Contents

R2014a

R2013b

R2013a

R2012b

R2012a

R2011b

R2011a

R2010b

R2010a

R2009b

R2009a

R2008b

R2008a

R2007b

R2007a

R2006b

R2006a

R14SP3

No New Features or Changes

R14SP2

R2014a

Version: 9.7

New Features: Yes

Bug Fixes: Yes

Redesigned PID Tuner app for improved PID tuning workflow

The redesigned PID Tuner streamlines workflows for interactively tuning PID controllers for reference tracking and disturbance rejection.

To access the PID Tuner, use the `pidtool` command. For example, to tune a PI controller for an LTI model, `G`:

```
pidtool(G,'PI')
```

For more information about the PID Tuner, see "Designing PID Controllers with the PID Tuner".

PID controller tuning using system identification to model the plant from measured input-output data in the PID Tuner app (with System Identification Toolbox)

If you have System Identification Toolbox™ software, you can use PID Tuner to fit a linear model to the measured SISO response data from your system and tune a PID controller for the resulting model. For example, if you want to design a PID controller for a manufacturing process, you can start with response data from a bump test on your system.

PID Tuner uses system identification to estimate an LTI model from the response data. You can interactively adjust the identified parameters to obtain an LTI model with a response that fits your response data. PID Tuner automatically tunes a PID controller for the estimated model. You can then interactively adjust the performance of the tuned control system, and save the estimated plant and tuned controller.

For an example, see "Interactively Estimate Plant Parameters from Response Data".

`freqsep` function for decomposing a linear system into fast dynamics and slow dynamics

Use the new `freqsep` command for separating numeric LTI models into fast and slow components. `freqsep` allows you to specify the cutoff frequency about which the model is decomposed. The slow component contains poles with natural frequency below the cutoff frequency. The fast component contains poles at or above the cutoff.

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

`damp` command display includes time constant information

Compatibility Considerations: Yes

When you call the `damp` command with no output arguments, the display now includes the time constant for each pole. The time constant is calculated as follows:

$$\tau = \frac{1}{\omega_n \zeta}.$$

ω_n is the natural frequency of the pole, and ζ is its damping ratio.

Compatibility Considerations

For a discrete-time system with unspecified sample time (`Ts = -1`), `damp` now calculates the natural frequency and damping ratio by assuming `Ts = 1`. Previously, the software returned `[]` for the natural frequency and damping ratio of such systems.

`damp` returns outputs in order of increasing natural frequency. Therefore, this change can result in reordered poles for systems with unspecified sample times.

For more information on the outputs, see the `damp` reference page.

R2013b

Version: 9.6

New Features: Yes

Bug Fixes: Yes

`SamplingGrid` property for tracking dependence of array of sampled models on variable values

In Control System Toolbox™, you can derive arrays of numeric or generalized LTI models by sampling one or more independent variables. The new `SamplingGrid` property of LTI models tracks the variable values associated with each model in such an array.

Set this property to a structure whose fields are the names of the sampling variables and contain the sampled variable values associated with each model. All sampling variables should be numeric and scalar valued, and all arrays of sampled values should match the dimensions of the model array.

For example, suppose you create a 11-by-1 array of linear models, `sysarr`, by taking snapshots of a linear time-varying system at times `t = 0:10`. The following code stores the time samples with the linear models.

```
sys.SamplingGrid = struct('time',0:10)
```

For an additional examples, see:

- Array With Variations in Two Parameters
- Sample a Tunable (Parametric) Model for Parameter Studies

Option to retain unconnected states when interconnecting models using `connect` command

By default, the `connect` command discards states that do not contribute to the dynamics in the path between the inputs and outputs of the interconnected system. You can now optionally retain such unconnected states. This option can be useful, for example, when you want to compute the interconnected system response from known initial state values of the components.

To instruct `connect` to retain unconnected states, use the new `connectOptions` command with the existing `connect` command.

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

`connect` command always returns state-space or frequency response data model

Compatibility Considerations: Yes

The `connect` command now always returns a state-space model, such as an `ss`, `genss`, or `uss` model, unless one or more of the input models is a frequency response data model. In that case, `connect` returns a frequency response data model, such as an `frd` or `genfrd` model.

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

Compatibility Considerations

In previous releases, `connect` returned a `tf` or `zpk` model when all input models were `tf` or `zpk` models. Therefore, `connect` might now return state-space models in cases where it previously returned `tf` or `zpk` models.

`updateSystem` command for updating dynamic system data in a response plot

The new `updateSystem` command replaces the system data used to compute a response plot with data derived from a different dynamic system, and updates the plot. `updateSystem` is useful, for example, to cause a plot in a GUI to update in response to interactive input.

For more information, see:

- `updateSystem` reference page
- Build GUI With Interactive Plot Updates

`getLoopID` renamed to `getSwitches`

Compatibility Considerations: Yes

The `getLoopID` function is now called `getSwitches` to more clearly reflect the purpose of the function. Using `getLoopID` does not generate an error in this release, but the function may be removed in a future release.

Compatibility Considerations

If you have scripts or functions that use `getLoopID`, consider replacing those calls with `getSwitches`.

`LoopID` property of `loopswitch` renamed to `Location`

Compatibility Considerations: Yes

The `LoopID` property of the `loopswitch` model component is now called `Location` to more clearly reflect the purpose of the property. Using `LoopID` does not generate an error in this release, but the name may be removed in a future release.

Compatibility Considerations

If you have scripts or functions that use the `LoopID` property, consider updating your code to use `Location` instead.

R2013a

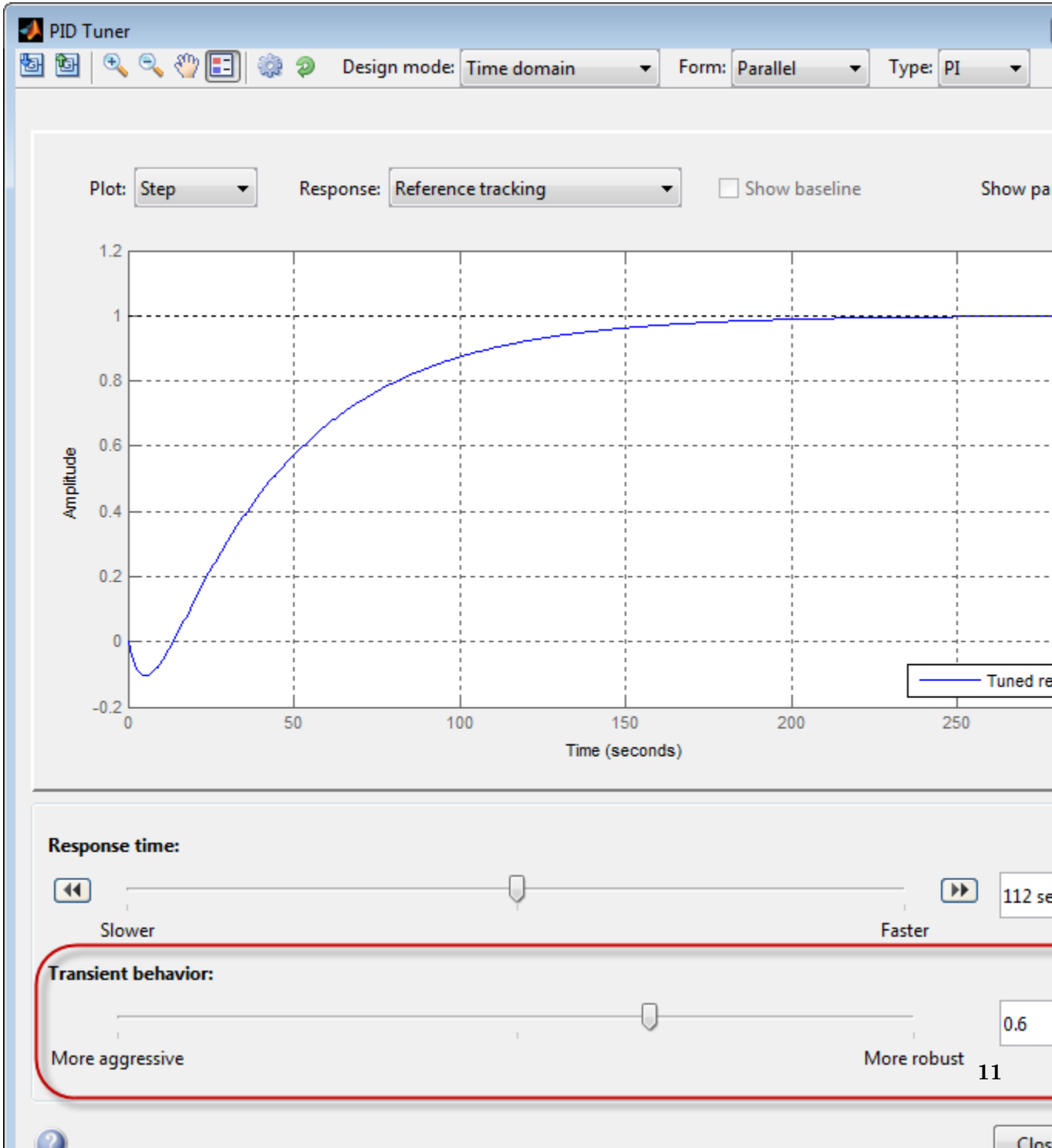
Version: 9.5

New Features: Yes

Bug Fixes: Yes

Transient behavior slider for PID Tuner, increasing 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.


PID Tuner
Design mode: Time domain
Form: Parallel
Type: PI
Plot: Step
Response: Reference tracking
Show baseline
Show pa
1.2
1
0.8
0.6
0.4
0.2
0
-0.2
Amplitude
0
50
100
150
200
250
Time (seconds)
Tuned re
Response time:
Slower
Faster
112 se
Transient behavior:
More aggressive
More robust
0.6

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.

R2012b

Version: 9.4

New Features: Yes

Bug Fixes: Yes

ltiblock.pid2 and loopswitch objects for tuning two-degree-of-freedom PID controllers and marking loop opening sites for open-loop requirements

New Control Design Blocks allow you to specify more control structures and more types of constraints for fixed-structure control system tuning in MATLAB®:

- `ltiblock.pid2` — Tunable two-degree-of-freedom PID controller
- `loopswitch` — Control Design Block for specifying feedback loop opening locations in a tunable `genss` model of a control system

You can use these Control Design Blocks to build control systems for tuning with Robust Control Toolbox™ tuning commands such as `systune` and `looptune`. For more information, see the `ltiblock.pid2` and `loopswitch` reference pages.

Commands for obtaining open-loop responses, closed-loop responses, and current values of tunable components from control system models

New commands allow you to compute open-loop and closed-loop responses from a Generalized LTI model representing a control system.

- `getLoopTransfer` — Compute point-to-point open-loop response of a Generalized LTI model of a control system, at a loop-opening site defined by a `loopswitch` block. The new command `getLoopID` returns a list of such loop-opening sites.
- `getIOTransfer` — Extract the closed-loop response from a specified input to a specified output of a control system.

These commands are particularly useful for validating the response functions of control systems tuned using Robust Control Toolbox tuning commands such as `systune`.

Additionally, the new `showTunable` command displays the current value of tunable components in a generalized LTI model of a control system. This

command is useful for querying tuned parameter values of control systems tuned using Robust Control Toolbox tuning commands such as `systune`.

For more information, see the reference pages for these new commands and the following topics:

- Generalized Models
- Models with Tunable Coefficients

Option for elementwise operation of model query commands on model arrays

Compatibility Considerations: Yes

The new `'elem'` flag causes elementwise operation on model arrays of the model query commands:

- `hasInternalDelay`
- `hasdelay`
- `isstatic`
- `isreal`
- `isfinite`
- `isproper`
- `isstable`

For example, for an array, `sysarray`, of dynamic system models,

```
B = hasdelay(sysarray,'elem');
```

returns a logical array. `B` of the same size as `sysarray` indicating whether the corresponding model in `sysarray` contains a time delay. Without the `'elem'` flag,

```
B = hasdelay(sysarray);
```

returns a scalar logical value that is equal to `1` if any entry in `sysarray` contains a time delay.

Compatibility Considerations

`isfinite` and `isstable` now return a scalar logical value when invoked without the `'elem'` flag. Previously, `isfinite` and `isstable` returned a logical array by default.

If you have scripts or functions that use `isfinite(sysarray)` or `isstable(sysarray)`, replace those calls with `isfinite(sysarray,'elem')` or `isstable(sysarray,'elem')` to perform an elementwise query and obtain a logical array.

R2012a

Version: 9.3

New Features: Yes

Bug Fixes: No

Frequency Analysis Commands for Calculating Peak Gain and Finding Gain-Crossover Frequencies

Control System Toolbox software includes two new frequency analysis commands:

- `getPeakGain` — Peak gain of frequency response of a dynamic system model
- `getGainCrossover` — Frequencies at which system gain crosses a specified gain level

For more information, see the `getPeakGain` and `getGainCrossover` reference pages.

These functions use the SLICOT library of numerical algorithms. For more information about the SLICOT library, see http://slicot.org.

Specify Target Crossover Frequency as Input to pidtune

A new syntax for `pidtune` lets you specify a target crossover frequency directly as an input argument. For example, the following command designs a PI controller, `C`, for a plant model `sys`. The command also specifies a target value `wc` for the 0 dB gain crossover frequency of the open-loop response `L = sys*C`.

```
C = pidtune(sys,'pi',wc);
```

Previously, you had to use `pidtuneOptions` to specify a target crossover frequency.

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

Rescaled Impulse Response and Impulse-Invariant Time Domain Conversion

Compatibility Considerations: Yes

For discrete-time dynamic system models, the input signal applied by `impulse` is now a unit area pulse of length `Ts` and height `1/Ts`. `Ts` is the

sampling time of the discrete-time system. Previously, `impulse` applied a pulse of length `Ts` and unit height.

Compatibility Considerations

Results of this change include:

- The amplitude of the impulse response calculated by `impulse` and `impulseplot` is scaled by `1/Ts` relative to previous versions.
- Discretization using the impulse-invariant (`'impulse'`) method of `c2d` returns a model that is scaled by `Ts` compared to previous releases. This scaling ensures a close match between the frequency responses of the continuous-time model and the impulse-invariant discretization as `Ts` approaches zero (for strictly proper models). In previous releases, the frequency responses differed by a factor of `Ts`.

First-Order Hold Method for d2c

The `d2c` command now supports the first-order hold (FOH) method for converting a discrete-time dynamic system model to continuous time. The FOH method converts by performing linear interpolation of the inputs, assuming the control inputs are piecewise linear over the sampling period. For more information about using this method, see the `d2c` reference page and Continuous-Discrete Conversion Methods.

tzero Computes Invariant Zeros and Transmission Zeros

The `tzero` command computes the invariant zeros of SISO and MIMO dynamic system models. For minimal realizations, `tzero` computes transmission zeros. `tzero` also returns the normal rank of the transfer function of the system. For more information, see the `tzero` reference page.

Models Created With System Identification Toolbox Can Be Used Directly With Control System Toolbox Functions

Identified linear models that you create using System Identification Toolbox software can now be used directly with Control System Toolbox analysis and compensator design commands. In prior releases, doing so required conversion to Control System Toolbox LTI model types.

Identified linear models include `idfrd`, `idss`, `idproc`, `idtf`, `idgrey` and `idpoly` models.

Identified linear models can be used directly with:

- Any Control System Toolbox or Robust Control Toolbox functions that operate on dynamic systems, including:
  - Response plots — `nichols`, `margin`, and `rlocus`
  - Model simplification — `pade`, `balred` and `minreal`
  - System interconnections — `series`, `parallel`, `feedback` and `connect`

  For a complete list of these functions, enter:

  ```
  methods('DynamicSystem')
  ```

- Analysis and design tools such as `ltiview`, `sisotool` and `pidtool`.
- The LTI System block in Simulink® models.

Functionality Being Removed or Changed

Compatibility Considerations: Yes

Functionality	What Happens When You Use This Functionality?	Use This Instead	Compatibility Considerations
`impulse(sys)` and `impulseplot(sys)`, for discrete-time `sys`	Still works.	N/A	Amplitude of response is scaled by `1/Ts` compared to previous versions. `Ts` is sampling time of `sys`.
`c2d(sys,Ts,'impulse')`	Still works.	N/A	Resulting discretized model is scaled by `Ts` compared to previous releases.

`[y,t] = impulse(sys,Tfinal)`
`[y,t] = step(sys,Tfinal)`
`[y,t,x] = initial(sys,Tfinal)`
	For discrete-time `sys` with undefined sample time (`Ts=-1`), `Tfinal` is interpreted as the number of sampling periods to simulate.	N/A	Expect the number of simulation data points to be `Tfinal + 1` instead of `Tfinal`.

R2011b

Version: 9.2

New Features: Yes

Bug Fixes: No

Formula-Based Specification of Summing Junctions and Vector Signal Naming for sumblk and connect

You can now use formula strings to specify the behavior of summing junctions with `sumblk`. For example, to create a summing junction, `S`, that takes the difference between signals `r` and `y` to produce signal `e`, enter the following command:

```
S = sumblk('e = r-y');
```

Additionally, both `sumblk` and `connect` now support vector-based signal naming for interconnecting multi-input, multi-output (MIMO) models. For more information, see the `sumblk` and `connect` reference pages.

Commands for Interacting with Control Design Blocks in Generalized LTI Models

The following new commands allow you to examine and set the values of Control Design Blocks in Generalized LTI Models:

- `getValue` — Get nominal value of Generalized Model (replaces `getNominal`)
- `setValue` — Modify value of Control Design Block
- `getBlockValue` — Get nominal value of Control Design Block in Generalized Model
- `setBlockValue` — Set value of Control Design Block in Generalized Model
- `showBlockValue` — Display nominal values of Control Design Blocks in Generalized Model

For more information about these commands, see the reference pages for each command.

Functionality Being Removed or Changed

Compatibility Considerations: Yes

Functionality	What Happens When You Use This Functionality?	Use This Instead	Compatibility Considerations
`delay2z`	Errors	`absorbDelay`	Replace `delay2z` with `absorbDelay`.
`getNominal`	Errors	`getValue`	Replace `getNominal` with `getValue`.
`Scale` and `Info` properties of `realp` parameter	Errors	None	None
`sumblk('a','b','c','+-')`	Still works	`sumblk('a=b-c')`	Use new formula-based syntax for `sumblk`.

R2011a

Version: 9.1

New Features: Yes

Bug Fixes: No

New Model Objects for Representing Tunable Parameters and Systems with Tunable Components

Control System Toolbox includes new model objects that you can use to represent systems with tunable components. You can use these models for parameter studies or controller synthesis using `hinfstruct` (requires Robust Control Toolbox). The new model types include:

- Control Design Blocks—Parametric components that are the building blocks for constructing tunable models of control systems. Control Design Blocks include:
  - `realp`—Tunable real parameter
  - `ltiblock.gain`—Tunable static gain block
  - `ltiblock.tf`—Fixed-order SISO transfer function with tunable coefficients
  - `ltiblock.ss`—Fixed-order state-space model with tunable coefficients
  - `ltiblock.pid`—One-degree-of-freedom PID controller with tunable coefficients
- Generalized Matrices—Matrices that include parametric (tunable) values. Generalized matrices are `genmat` models.
- Generalized and Uncertain LTI Models—Models representing systems that have both fixed and tunable coefficients. Generalized LTI models include:
  - `genss`—Generalized state-space model
  - `genfrd`—Generalized frequency response data model

  These models arise from interconnections between numeric LTI models (such as `tf` , `ss`, or `frd`) and Control Design Blocks. You can also create `genss` models by using the `tf` or `ss` commands with one or more `realp` or `genmat` inputs.

This release also adds new functions for working with generalized models:

- `getNominal`—Nominal value of generalized model
- `replaceBlock`—Replace Control Design Blocks in generalized model
- `nblocks`—Number of blocks in generalized model

- `isParametric` — Determine if model has tunable blocks
- `getLFTModel`—Decompose generalized model

For more information about the new model types and about modeling systems that contain tunable coefficients, see the following in the *Control System Toolbox User's Guide*:

- Types of Model Objects
- Models with Tunable Coefficients

New Time and Frequency Units for Models and Response Plots

All linear model objects now have a `TimeUnit` property for specifying unit of the time variable, time delays in continuous-time models, and sampling time in discrete-time models. The default time units is seconds. You can specify the time units, for example, as hours. See Specify Model Time Units for examples.

Frequency-response data (`frd` and `genfrd`) models also have a new `FrequencyUnit` property for specifying units of the frequency vector. The default frequency units is `rad/TimeUnit`, where `TimeUnit` is the system time units. You can specify the units, for example as KHz, independently of the system time units. See Specify Frequency Units of Frequency-Response Data Model for examples. If your code uses the `Units` property of frequency-response data models, it continues to work as before.

See the model reference pages for available time and frequency units options.

Changing the `TimeUnit` and `FrequencyUnit` properties changes the overall system behavior. If you want to simply change the time and frequency units without modifying system behavior, use `chgTimeUnit` and `chgFreqUnit`, respectively.

The time and frequency units of the model appear on the response plots by default. For multiple systems, the units of the first system are used. You can change the units of the time and frequency axes:

- Graphically, using the following editors:

  - Toolbox Preferences Editor
  - LTI Viewer Preferences Editor
  - Graphical Tuning Window Preferences Editor
  - Property Editor of individual plots
- Programmatically, by setting the following properties of plots:
  - `TimeUnits` for time-domain plots using `timeoptions`
  - `FreqUnits` for frequency-domain plots using, for example, `bodeoptions`

Discrete-Time PID Controller Objects Have Stable Derivative Filter Pole

Compatibility Considerations: Yes

New requirements for creating `pid` and `pidstd` controller objects ensure that the derivative filter pole is always stable.

- For a discrete-time `pid` controller with a derivative filter (`Tf` $\neq$ 0) and `Dformula` set to `'ForwardEuler'`, the sampling time `Ts` must be less than `2*Tf`.
- For a discrete-time `pidstd` controller with a derivative filter (`N` $\neq$ `Inf`) and `Dformula` set to `'ForwardEuler'`, the sampling time `Ts` must be less than `2*Td/N`.
- The `Trapezoidal` value for `DFormula` is not available for a discrete-time `pid` or `pidstd` controller with no derivative filter (`Tf = 0` or `N = Inf`).

Compatibility Considerations

On loading `pid` or `pidstd` controllers saved under previous versions, the software changes certain properties of controllers that do not have stable derivative filter poles.

- For a discrete-time `pid` controller with a derivative filter (`Tf` $\neq$ 0), `Dformula` set to `'ForwardEuler'`, and sampling time `Ts` $\geq$ `2*Tf`, the derivative filter time is reset to `Tf = Ts`.

- For a discrete-time `pidstd` controller with a derivative filter (`N`≠ `Inf`), `Dformula` set to `'ForwardEuler'`, the sampling time `Ts` ≥ `2*Td/N`, the derivative filter constant is reset to `N = Td/Ts`.
- For a discrete-time `pid` or `pidstd` controller with no derivative filter and `DFormula = 'Trapezoidal'`, the derivative filter integrator formula is reset to `DFormula = 'ForwardEuler'`.

The software issues a warning when it changes any of these values. If you receive such a warning, validate your controller to ensure that the new values achieve the desired performance.

New Variable q^–1 for Expressing Discrete-Time Transfer Functions

You can now express discrete-time `tf` and `zpk` models in terms of the inverse shift operator `q^-1`. The variable `q^-1` is equivalent to `z^-1`.

Note This new definition is consistent with the System Identification Toolbox definition of `q^-1`.

Use the new variable by setting the `Variable` property of a `tf` or `zpk` model to `q^-1`. For example, entering:

```
H = tf([1 2 3],[5 6 7],0.1,'Variable','q^-1')
```

creates the following discrete-time transfer function:

```
Transfer function:
1 + 2 q^-1 + 3 q^-2
-------------------
5 + 6 q^-1 + 7 q^-2

Sampling time (seconds): 0.1
```

When you set `Variable` to `q^-1`, `tf` interprets the numerator and denominator vectors as ascending powers of `q^-1`.

For more information, see the `tf` and `zpk` reference pages.

R2010b

Version: 9.0

New Features: Yes

Bug Fixes: No

New Commands and GUI for Modeling and Tuning PID Controllers

This release introduces specialized tools for modeling and designing PID controllers.

PID Controller Design with the New PID Tuner GUI

The new PID Tuner GUI lets you interactively tune a PID controller for your required response characteristics. Using the GUI, you can adjust and analyze your controller's performance with response plots, such as reference tracking, load disturbance rejection, and controller effort, in both time and frequency domains.

The PID Tuner supports all types of SISO plant models, including:

- Continuous- or discrete-time plant models
- Stable, unstable, or integrating plant models
- Plant models that include I/O time delays or internal time delay

For more information about using PID Tuner, see:

- Designing PID Controllers in the *Control System Toolbox Getting Started Guide*
- The new demo Designing PID for Disturbance Rejection with PID Tuner

PID Controller Design with the New pidtune Command

The new `pidtune` command lets you tune PID controller gains at the command line.

`pidtune` automatically tunes the PID gains to balance performance (response time) and robustness (stability margins). You can specify your own response time and phase margin targets using the new `pidtuneOptions` command.

`pidtune` supports all types of SISO plant models, including:

- Continuous- or discrete-time plant models.

- Stable, unstable, or integrating plant models.
- Plant models that include I/O time delays or internal time delays.
- Arrays of plant models. If `sys` is an array, `pidtune` designs a separate controller for each plant in the array.

For additional information, see:

- The `pidtune` and `pidtuneOptions` reference pages
- The new Control System Toolbox demo Designing Cascade Control System with PI Controllers

Modeling PID Controllers in Parallel Form or Standard Form

The new LTI model objects `pid` and `pidstd` are specialized for modeling PID controllers.

With `pid` and `pidstd` you can model a PID controller directly with the PID parameters, expressed in parallel (`pid`) or standard (`pidstd`) form. The `pid` and `pidstd` commands can also convert to PID form any type of LTI object that represents a PID controller.

Previously, to model a PID controller, you had to derive the controller's equivalent transfer function (or other model), and could not directly store the PID parameters.

For additional information, see the `pid` and `pidstd` reference pages

Improved PID Tuning Options in SISO Design Tool

This release includes improvements to the PID Tuning options in the Automated Tuning pane of SISO Design Tool.

In addition to the Robust Response Time tuning algorithm, SISO Design Tool offers a collection of classical design formulas, including the following:

- Approximate *M*-Constrained Integral Gain Optimization (MIGO) Frequency Response

- Approximate MIGO Step Response
- Chien-Hrones-Reswick
- Skogestad Internal Model Control (IMC)
- Ziegler-Nichols Frequency Response
- Ziegler-Nichols Step Response

For information about using SISO Design Tool, see SISO Design Tool in the *Control System Toolbox User's Guide*. For specific information about the automatic PID Tuning options in SISO Design Tool, see PID Tuning in the *Control System Toolbox User's Guide*.

Ability to Analyze a Controller Design for Multiple Models Simultaneously in SISO Design Tool

You can now analyze a controller design for multiple models simultaneously using the SISO Design Tool. This feature helps you analyze whether the controller satisfies design requirements on a system whose exact dynamics are not known and may vary.

System dynamics can vary because of parameter variations or different operating conditions. You represent variations in system dynamics of the plant (`G`), sensor (`H`), or both in a feedback structure using arrays of LTI models. Then, design a controller for a nominal model in the array and analyze that the controller satisfies the design requirements on the remaining models using the design and analysis plots. For more information, see:

- Control Design Analysis of Multiple Models in the Control System Toolbox documentation.
- Compensator Design for a Set of Plant Models demo.
- Reference Tracking of a DC Motor with Parameter Variations demo in Simulink Control Design™ software.

Change in Output of repsys Command

Compatibility Considerations: Yes

The output of the `repsys` command when called with a single dimension argument has changed.

In prior versions, the output of `repsys(sys,N)` was the same as that of `append(sys,...,sys)`.

Now, `repsys(sys,N)` returns the same result as `repsys(sys,[N N])`.

The results of other syntaxes for `repsys` have not changed.

See the `repsys` and `append` reference pages for more information.

Compatibility Considerations

Code that depends upon the previous result of `repsys(sys,N)` no longer returns that result. To obtain the previous result, replace `repsys(sys,N)` with `sys*eye(N)`.

R2010a

Version: 8.5

New Features: Yes

Bug Fixes: No

Enhanced c2d Command to Approximate Fractional Time Delays in Tustin and Matched Discretization Methods

The `c2d` command can now approximate fractional time delays when discretizing linear models with the `tustin` or `matched` methods. The new `c2dOptions` command lets you specify an optional Thiran all-pass filter. The Thiran filter approximates fractional delays for improved phase matching between continuous and discretized models. Previously, `c2d` rounded fractional time delays to the nearest multiple of the sampling time when using the `tustin` or `matched` methods. For more information, see the `c2d` and `c2dOptions` reference pages and Continuous-Discrete Conversion Methods in the *Control System Toolbox User Guide*.

New Commands for Specifying Options for Continuous-Discrete Conversions

Compatibility Considerations: Yes

New commands `c2dOptions`, `d2dOptions`, and `d2cOptions` make it easier to specify options for

- Discretization using `c2d`
- Resampling using `d2d`.
- Conversion from discrete to continuous time using `d2c`.

Compatibility Considerations

This release deprecates the `prewarp` method for `c2d`, `d2d`, and `d2c`. Instead, use `c2dOptions`, `d2dOptions`, or `d2cOptions` to specify the `tustin` method and a prewarp frequency. For more information, see Continuous-Discrete Conversion Methods and the `c2d`, `d2d`, and `d2c` reference pages.

New FDEL Command to Remove Specified Data from Frequency Response Data (FRD) Models

You can now remove selected data from `frd` models using the new `fdel` command. For example, use `fdel` to:

- Remove spurious or unneeded data from `frd` models you create from measured frequency response data.
- Remove data at intersecting frequencies from `frd` models before merging them into a single `frd` model with `fcat`, which can only merge `frd` models containing no common frequencies.

For more information, see `fdel` reference page.

R2009b

Version: 8.4

New Features: Yes

Bug Fixes: No

Ability to Design Compensators for New Types of Plants

In the SISO Design Tool, you can now design compensators for plants models that:

- Contain time delays

  Previously, you had to approximate delays before designing compensators.

- You specify as frequency-response data (FRD)

For more information on designing compensators using the SISO Design Tool, see SISO Design Tool.

New Automated PID Tuning Method

You can now tune compensators using a new automated PID tuning algorithm called Robust Response Time, which is available in the SISO Design Tool. 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 Automated Tuning.

R2009a

Version: 8.3

New Features: Yes

Bug Fixes: No

Variable q Now Defined as the Forward Shift Operator z

Compatibility Considerations: Yes

The variable q is now defined in the standard way as the forward shift operator z. Previously, q was defined as z^{-1}.

Note This new definition is consistent with the System Identification Toolbox definition of q.

Compatibility Considerations

If you use the q variable, you may receive different results than in previous releases when you:

- Create a transfer function
- Modify the `num` or `den` properties of an existing transfer function

The resulting transfer function differs from previous releases when both the

- `Variable` property is set to q
- `num` and `den` properties have different lengths

For example, the following code:

```
H = tf([1,2],[1 3 8],0.1,'Variable','q')
```

now returns the transfer function

$$\frac{q+2}{q^2+3q+8} \equiv \frac{z+2}{z^2+3z+8}$$

Previously, the code returned the transfer function

$$\frac{1+2q}{1+3q+8q^2} \equiv \frac{1+2z^{-1}}{1+3z^{-1}+8z^{-2}} \equiv \frac{z^2+2z}{z^2+3z+8}$$

The two transfer functions have different numerators.

R2008b

Version: 8.2

New Features: Yes

Bug Fixes: No

New Design Tools for Linear-Quadratic-Gaussian (LQG) Servo Controllers with Integral Action

Compatibility Considerations: Yes

You can now design a Linear-Quadratic-Gaussian (LQG) servo controller for set-point tracking using the new `lqi` and `lqgtrack` commands. This compensator ensures that the system output tracks the reference command and rejects process disturbances and measurement noise.

For more information on forming LQG servo controllers, see Linear-Quadratic-Gaussian (LQG) Design, the `lqi` reference page, and the `lqgtrack` reference page.

Current Flag Moved from lqgreg to kalman

The `'current'` flag was moved from the `lqgreg` function to the `kalman` function.

Compatibility Considerations

The following code:

```
kest = kalman(sys,Qn,Rn)
c = lqgreg(kest,k)
```

now returns the current regulator $u[n] = -K\hat{x}[n \mid n]$ instead of the delayed regulator $u[n] = -K\hat{x}[n \mid n-1]$.

To update your code to return the same results as in previous releases, use the following code with the added string `'delayed'` in the `kalman` command:

```
kest = kalman(sys,Qn,Rn,'delayed')
c = lqgreg(kest,k)
```

For information on using these functions with the current flag in the `kalman` function, see the `kalman` and `lqgreg` reference pages.

New Upsampling Method for Rate Conversion in Discrete-Time Models

You can now upsample a discrete-time system to an integer multiple of the original sampling rate without any distortion in the time or frequency domain using the `upsample` command.

For more information on upsampling, see the `upsample` reference page and Upsample a Discrete-Time System in the *Control System Toolbox User's Guide*.

New Scaling Tools to Enhance the Accuracy of Computations with State-Space Models

You can now scale state-space models to maximize accuracy over the frequency band of interest using the `prescale` command and associated GUI. Use this functionality when you cannot achieve good accuracy at all frequencies and some tradeoff is necessary. A warning alerts you when accuracy may be poor and using prescaling is recommended.

For more information on setting the frequency band for scaling state-space realizations, see Scaling State-Space Models and the `prescale` reference page.

New Command to Reorder the States of State-Space Models

You can now reorder the states of state-space models according to a specified permutation using the `xperm` command.

For more information on reordering states, see the `xperm` reference page.

Enhanced Support for Customizing Response Plots

You can now make the following changes to your Control System Toolbox response plots using the figure plotting tools:

- System name

- Line color
- Line style
- Line width
- Marker type

For more information on customizing the appearance of response plots using plot tools, see Customizing Response Plots Using Plot Tools in the *Control System Toolbox User's Guide*.

R2008a

Version: 8.1

New Features: Yes

Bug Fixes: No

Updated Error and Warning Message System

The Control System Toolbox 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: 8.0.1

New Features: Yes

Bug Fixes: No

Updated and Expanded Demos

The Control System Toolbox demos have been reformatted and expanded to include more examples and content. Demos in the following categories now have new and improved content:

- Getting Started with LTI Models
- Discretization and Sampling Rate Conversions
- How to Get Accurate Results

To open the Control System Toolbox demos, type

```
demo toolbox control
```

at the MATLAB prompt.

R2007a

Version: 8.0

New Features: Yes

Bug Fixes: No

Analysis of Time Delay Systems Now Fully Supported

Control System Toolbox software now lets you:

- Model, simulate, and analyze any interconnection of linear systems with delays, such as systems containing feedback loops with delays.
- Exactly analyze and simulate control systems with long delays. You can evaluate control strategies, such as Smith Predictor and PID control for first-order-plus-dead-time plants.
- Use new commands for modeling state-space models with delays including: `delayss`, `getDelayModel`, and `setDelayModel`.

For more information, see the section on Models with Time Delays in the Control System Toolbox documentation.

New and Updated Automated Tuning Methods

Control System Toolbox software now provides the following new and updated automated tuning methods:

- New Singular Frequency Based Tuning lets you design PID compensators for both stable and unstable plants.
- New H-infinity Loop Shaping lets you find compensators based on a desired open-loop bandwidth or loop shape. This feature requires Robust Control Toolbox software.
- Updated Internal Model Control (IMC) Tuning now supports unstable plants.

For more information, see the section on automated tuning in the Control System Toolbox documentation.

New Tustin and Prewarp Options for d2d Function

The `d2d` function now includes the following new options for the resampling method:

- `'tustin'`—Performs Bilinear (Tustin) approximation
- `'prewarp'`—Performs Tustin approximation with frequency prewarping

For more information, see the `d2d` reference pages.

R2006b

Version: 7.1

New Features: Yes

Bug Fixes: No

New Loop Configurations in the SISO Design Tool

Two new loop configurations are available from the SISO Design Tool. See Modifying Block Diagram Structure for more information.

New Design Requirements

The LTI Viewer now supports step response and upper/lower time bound design requirements. See Adding Design Requirements to the LTI Viewer for more information.

R2006a

Version: 7.0

New Features: Yes

Bug Fixes: No

SISO Design Tool

The SISO Design Tool now provides one-click automated tuning using systematic algorithms such as Ziegler-Nichols PID tuning, IMC design, and LQG design. In addition, you can calculate low-order approximations of the IMC/LQG compensators to keep the control system complexity low.

Compensator Optimization Is Now Supported

If you have installed Simulink Response Optimization™ software, you can now optimize the compensator parameters inside the SISO Design Tool GUI. You can specify time- and frequency-domain requirements on SISO Design Tool plots such as `bode` and `step`, and use numerical optimization algorithms to automatically tune your compensator to meet your requirements. See the Simulink Response Optimization documentation for more details.

Improved Compensator Editor

The Compensator Editor used to edit the numerical values of poles and zeros has been upgraded to better handle common control components such as lead/lag and notch filters.

Multi-Loop Compensator Design Support

Many control systems involve multiple feedback loops, some of which are coupled and need joint tuning. The SISO Design Tool now lets you analyze and tune multi-loop configurations. You can focus on a specific loop by opening signals to remove the effects of other loops, gain insight into loop interactions, and jointly tune several SISO loops.

SISO Design Tool Fully Integrated with the Controls & Estimation Tools Manager

To improve workflow and better leverage other tools, such as Simulink Control Design software and Simulink Response Optimization software, the SISO Design Tool is now fully integrated with the Controls & Estimation Tools Manager (CETM). This provides a signal environment for the design and tuning of compensators.

When you open the SISO Design Tool, the CETM also opens with a SISO Design Task. Many SISO Design Tool features, such as importing models, changing loop configurations, etc., have been moved to the SISO Design Task in CETM. In addition, related tasks such as Simulink based Tuning and Compensator Optimization are seamlessly integrated with the SISO Design Task. See the *Control System Toolbox Getting Started Guide* for details on the new work flow.

LTI Viewer Enhancements

The LTI Viewer now lets you plot the response of a system to user-defined input signals (`lsim`) and initial conditions (`initial`). A new GUI lets you select input signals from a signal generator library, or import signal data from a variety of file formats.

LTI Objects

Descriptor and Improper State-Space Models Fully Supported

There is now full support for descriptor state-space models with a singular `E` matrix. This now lets you build state-space representations, such as PID, and manipulate improper models with the superior accuracy of state-space computations. In previous versions, only descriptor models with a nonsingular `E` matrix were supported.

New Commands to Calculate Time Response Metrics

The new `stepinfo` and `lsiminfo` commands compute time-domain performance metrics, such as rise time, settling time, and overshoot. You can use these commands to write scripts that automatically verify or optimize such performance requirements. Previously, these metrics were available only from response plots.

Simplified System Interconnections Using I/O Channel Names

The commands `connect`, `feedback`, `series`, `parallel`, and `lft` now let you connect systems by matching names of I/O channels. A helper function, `sumblk`, has also been added to simplify the specification of summing junctions. Altogether this considerably simplifies the task of deriving models

for complicated block diagrams. In previous releases, only index-based system connection was supported.

Changes in the Representation of I/O Delays in State-Space Models

The `ioDelay` property is deprecated from state-space models. Instead, these models have a new property called `InternalDelay` for logging all delays that cannot be pushed to the inputs or outputs. Driving this change is the switch to a representation of delays in terms of delayed differential equations rather than frequency response. See Models with Time Delays in the Control System Toolbox documentation for more details on internal delays, and `ss/getdelaymodel` for details on the new internal representation of state-space models with delays.

New Name Property for LTI Objects

This new property lets you attach a name (string) to a given LTI model. The specified name is reflected in response plots.

New Commands and Operations for LTI Objects

The new `exp` command simplifies the creations of continuous-time transfer functions with delays. For more information, type help `lti/exp` at the MATLAB prompt.

The `frd` object has the following new methods:

- `fcat` — Concatenates one or more FRD models along the frequency dimension (data merge).
- `fselect` — Selects frequency points or range in `frd` model.
- `fnorm` — Calculates pointwise peak gain of `frd` model.

The `.*` operation is supported for transfer functions and zero-pole-gain objects. This allows you to perform element-by-element multiplication of MIMO models.

Numerical Algorithms

There have been several major improvements in the Control System Toolbox numerical algorithms, many of which benefit the upgraded SISO Design Tool:

- New scaling algorithm that maximizes accuracy for badly scaled state-space models
- Performance improvement in time and frequency response computations through MEX-files
- More accurate computations of the zero-pole-gain and transfer function representations of a state-space model
- More accurate state-space representations of zero-pole-gain models
- Better handling of nonminimal modes in model reduction commands (`balred`, `balreal`)
- `canon` now computes a block modal form for `A` matrices that are not diagonizable or are nearly defective
- Exact phase computation for zero-pole-gain models in `bode` and `nichols`
- Accurate handling of improper models using the descriptor state-space representation

R14SP3

Version: 6.2.1

New Features: No

Bug Fixes: No

No New Features or Changes

R14SP2

Version: 6.2

New Features: Yes

Bug Fixes: No

Command-Line API for Customizing Plots

The Control System Toolbox software now provides a command-line API for customizing units, labels, limits, and other plot options. You can now change default plot options before generating a plot, or modify plot properties after creation.

For a detailed description of the commands, see the Control System Toolbox documentation.

Constraint Types for SISO Design

You can now create

- Single piecewise linear constraints for root-locus and Bode plots
- Gain/phase exclusion regions for Nichols plots

Design constraints are displayed as shaded regions.

Bode and Nichols Plots Have Additional Options

When editing Bode and Nichols plots, you can now

- Set the lower limit of the magnitude manually.
- Adjust the phase offsets by multiples of 360 degrees to facilitate comparing multiple responses.

Model-Approximation and Order-Reduction Commands

New commands have been added for model approximation and order reduction:

- `hsvd` computes and plots the Hankel singular values.

- `balred` computes low-order approximations using a numerically stable, balancing-free algorithm. You can perform multiple order reductions with a single command.