System Identification Toolbox™ Release Notes

How to Contact The 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.

System Identification Toolbox™ Release Notes

Contents

Summary by Version 1

Version 7.3.1 (R2009b) System Identification Toolbox Software 5

Version 7.3 (R2009a) System Identification Toolbox Software 6

Version 7.2.1 (R2008b) System Identification Toolbox Software 8

Version 7.2 (R2008a) System Identification Toolbox Software 9

Version 7.1 (R2007b) System Identification Toolbox Software 16

Version 7.0 (R2007a) System Identification Toolbox Software 17

Version 6.2 (R2006b) System Identification Toolbox Software 20

Version 6.1.3 (R2006a) System Identification Toolbox Software 22

Version 6.1.2 (R14SP3) System Identification Toolbox Software 24

Version 6.1.1 (R14SP2) System Identification Toolbox Software 25

Version 6.0 (R13SP2) System Identification Toolbox Software 26

Compatibility Summary for System Identification Toolbox Software **30**

Summary by Version

This table provides quick access to what's new in each version. For clarification, see "Using Release Notes" on page 2.

Version (Release)	New Features and Changes	Version Compatibility Considerations	Fixed Bugs and Known Problems	Related Documentation at Web Site
Latest Version V7.3.1 (R2009b)	No	No	Bug Reports	Printable Release Notes: PDF Current product documentation
V7.3 (R2009a)	Yes Details	No	Bug Reports	No
V7.2.1 (R2008b)	No	Yes Summary	Bug Reports	No
V7.2 (R2008a)	Yes Details	Yes Summary	Bug Reports	No
V7.1 (R2007b)	Yes Details	No	Bug Reports	No
V7.0 (R2007a)	Yes Details	No	Bug Reports	No
V6.2 (R2006b)	Yes Details	No	Bug Reports	No
V6.1.3 (R2006a)	Yes Details	Yes Summary	Bug Reports	No
V6.1.2 (R14SP3)	No	No	Bug Reports	No
V6.1.1 (R14SP2)	No	No	Fixed bugs	No
V6.0 (R13SP2)	Yes Details	Yes Summary	No bug fixes	V6.0 product documentation

Using Release Notes

Use release notes when upgrading to a newer version to learn about:

- New features
- Changes
- Potential impact on your existing files and practices

Review the release notes for other MathWorks™ products required for this product (for example, MATLAB® or Simulink®). Determine if enhancements, bugs, or compatibility considerations in other products impact you.

If you are upgrading from a software version other than the most recent one, review the current release notes and all interim versions. For example, when you upgrade from V1.0 to V1.2, review the release notes for V1.1 and V1.2.

What Is in the Release Notes

New Features and Changes

- New functionality
- Changes to existing functionality

Version Compatibility Considerations

When a new feature or change introduces a reported incompatibility between versions, the **Compatibility Considerations** subsection explains the impact.

Compatibility issues reported after the product release appear under Bug Reports at The MathWorks™ Web site. Bug fixes can sometimes result in incompatibilities, so review the fixed bugs in Bug Reports for any compatibility impact.

Fixed Bugs and Known Problems

The MathWorks offers a user-searchable Bug Reports database so you can view Bug Reports. The development team updates this database at release

time and as more information becomes available. Bug Reports include provisions for any known workarounds or file replacements. Information is available for bugs existing in or fixed in Release 14SP2 or later. Information is not available for all bugs in earlier releases.

Access Bug Reports using your MathWorks Account.

About Functions and Properties Being Removed

This section lists functions or properties removed or in the process of being removed. Functions and properties typically go through several stages across multiple releases before being completely removed. This provides time for you to make adjustments to your code.

- Announcement — The Release Notes announce the planned removal, but there are no functional changes; the function runs as it did before.
- Warning — When you run the function, it displays a warning message indicating it will be removed in a future release; otherwise the function runs as it did before.
- Error — When you run the function, it produces an error. The error message indicates the function was removed and suggests a replacement function, if one is available.
- Removal — When you run the function, it fails. The error message is the standard message when MATLAB does not recognize an entry.

Functions and properties might be in a stage for one or more releases before moving to another stage. Functions and properties are listed in the Functions and Properties Being Removed section only when they enter a new stage and their behavior changes. For example, if a function displayed a warning in the previous release and errors in this release, it appears on the list. If it continues to display a warning, it does not appear on the list because there was no change between the releases.

Not all functions and properties go through all stages. For example, a function's impending removal might not be announced, but instead, the first notification might be that the function displays a warning.

The Release Notes include actions you can take to mitigate the effects of function or property removal, such as adapting your code to use a replacement function.

Version 7.3.1 (R2009b) System Identification Toolbox Software

This table summarizes what's new in Version 7.3.1 (R2009b):

New Features and Changes	Version Compatibility Considerations	Fixed Bugs and Known Problems	Related Documentation at Web Site
No	No	Bug Reports	Printable Release Notes: PDF Current product documentation

Version 7.3 (R2009a) System Identification Toolbox Software

This table summarizes what's new in Version 7.3 (R2009a):

New Features and Changes	Version Compatibility Considerations	Fixed Bugs and Known Problems	Related Documentation at Web Site
Yes Details below	No	Bug Reports	Printable Release Notes: PDF Current product documentation

New feature introduced in this version:

- "Enhanced Handling of Offsets and Trends in Signals" on page 6
- "Ability to Get Regressor Values in Nonlinear ARX Models" on page 7

Enhanced Handling of Offsets and Trends in Signals

This version of the product includes new and expanded functionality for handling offsets and trends in signals. This data processing operation is necessary for estimating more accurate linear models because linear models cannot capture arbitrary differences between the input and output signal levels.

The previous version of the product let you remove mean values or linear trends from steady-state signals using the GUI and the `detrend` function. For transient signals, you had to remove offsets and trends using matrix manipulation.

The GUI functionality for removing means and linear trends from signals is unchanged. However, you can now do the following at the command line:

- Save the values of means or linear trends removed during detrending using a new `detrend` output argument. You can use this saved trend information to detrend other data sets. You can also restore subtracted trends to the

output simulated by a linear model that was estimated from detrended data.

For example, this syntax computes and removes mean values from the data, and saves these values to the output variable `T`: `[data_d,T]=detrend(data)`. `T` is an object with properties that store offset and slope information for input and output signals.

- Remove any offset or linear trend from the data using a new `detrend` input argument. This is useful for removing arbitrary nonzero offsets from transient data or applying previously saved trend information to any data set.

  For example, this syntax removes an offset or trend specified by `T`: `data_d = detrend(data,T)`.

- Add an arbitrary offset or linear trend to data signals. This is useful when you want to simulate the response of a linear model about a nonzero equilibrium input-output level and this model was estimated from detrended data.

  For example, this syntax adds trend information to a simulated model output `y_sim`, which is an `iddata` object: `y = retrend(y_sim,T)`. `T` specifies the offset and slope information for inputs and outputs.

For more information, see "Handling Offsets and Trends in Data".

Ability to Get Regressor Values in Nonlinear ARX Models

The `getreg` command can now return the numerical values of regressors in nonlinear ARX models and provides an intermediate output of nonlinear ARX models.

This advanced functionality converts input and output values to regressors, and passes the regressor values to the `evaluate` command to compute the model response. This incremental step lets you gain insight into the propagation of information through the nonlinear ARX model.

For more information, see the `getreg` reference page. To learn more about the nonlinear ARX model structure, see "Nonlinear Black-Box Model Identification".

Version 7.2.1 (R2008b) System Identification Toolbox Software

This table summarizes what's new in Version 7.2.1 (R2008b):

New Features and Changes	Version Compatibility Considerations	Fixed Bugs and Known Problems	Related Documentation at Web Site
No	Yes Summary	Bug Reports	No

Functions and Properties Being Removed

For more information about the process of removing functions, see "About Functions and Properties Being Removed" in "What Is in the Release Notes" on page 2.

Function or Property Name	What Happens When You Use Function or Property?	Use This Instead	Compatibility Considerations
`model.Algorithm.Trace`	Still runs	`model.Algorithm.Display`	Using `model.Algorithm.Trace` results in a warning.

Version 7.2 (R2008a) System Identification Toolbox Software

This table summarizes what's new in Version 7.2 (R2008a):

New Features and Changes	Version Compatibility Considerations	Fixed Bugs and Known Problems	Related Documentation at Web Site
Yes Details below	Yes Summary	Bug Reports	No

New features introduced in this version are:

- "Simulating Nonlinear Black-Box Models in Simulink Software" on page 9
- "Linearizing Nonlinear Black-Box Models at User-Specified Operating Points" on page 10
- "Estimating Multiple-Output Models Using Weighted Sum of Least Squares Minimization Criterion" on page 11
- "Improved Handling of Initial States for Linear and Nonlinear Models" on page 12
- "Improved Algorithm Options for Linear Models" on page 13
- "New Block Reference Pages" on page 14
- "Functions and Properties Being Removed" on page 14

Simulating Nonlinear Black-Box Models in Simulink Software

You can now simulate nonlinear ARX and Hammerstein-Wiener models in Simulink using the nonlinear ARX and the Hammerstein-Wiener model blocks in the System Identification Toolbox™ block library. This is useful in the following situations:

- Representing dynamics of a physical component in a Simulink model using a data-based nonlinear model

- Replacing a complex Simulink subsystem with a simpler data-based nonlinear model

---

Note Nonlinear ARX Model and Hammerstein-Wiener Model blocks read variables from the MATLAB (base) workspace or model workspace. When the MATLAB workspace and model workspace contain a variable with the same name and this variable is referenced by a Simulink block, the variable in the model workspace takes precedence.

If you have installed Real-Time Workshop® software, you can generate code from models containing nonlinear ARX and the Hammerstein-Wiener model blocks. However, you cannot generate code when:

- Hammerstein-Wiener models use the `customnet` estimator for input or output nonlinearity.
- Nonlinear ARX models use custom regressors or use the `customnet` or `neuralnet` nonlinearity estimator.

You can access the new System Identification Toolbox blocks from the Simulink Library Browser. For more information about these blocks, see the IDNLARX Model (nonlinear ARX model) and the IDNLHW Model (Hammerstein-Wiener model) block reference pages.

Linearizing Nonlinear Black-Box Models at User-Specified Operating Points

You can now use the `linearize` command to linearize nonlinear black-box models, including monlinear ARX and Hammerstein-Wiener models, at specified operating points. Linearization produces a first-order Taylor series approximation of the system about an operating point. An *operating point* is defined by the set of constant input and state values for the model.

If you do not know the operating point, you can use the `findop` command to compute it from specifications, such as steady-state requirements or values of these quantities at a given time instant from the simulation of the model.

For nonlinear ARX models, if all of the steady-state input and output values are known, you can map these values to the model state values using the `data2state` command.

`linearize` replaces `lintan` and removes the restriction for linearizing models containing custom regressors or specific nonlinearity estimators, such as `neuralnet` and `treepartition`.

If you have installed Simulink® Control Design™ software, you can linearize nonlinear ARX and Hammerstein-Wiener models in Simulink after importing them into Simulink.

For more information, see:

- "Linear Approximation of Nonlinear Black-Box Models" about computing operating points and linearizing models
- "Simulating Model Output" about importing nonlinear black-box models into Simulink

Estimating Multiple-Output Models Using Weighted Sum of Least Squares Minimization Criterion

You can now specify a custom weighted trace criterion for minimization when estimating linear and nonlinear black-box models for multiple-output systems. This feature is useful for controlling the relative importance of output channels during the estimation process.

The `Algorithm` property of linear and nonlinear models now provides the `Criterion` field for choosing the minimization criterion. This new field can have the following values:

- `det` — (Default) Specify this option to minimize the determinant of the prediction error covariance. This choice leads to maximum likelihood estimates of model parameters. It implicitly uses the inverse of estimated noise variance as the weighting function. This option was already available in previous releases.
- `trace` — Specify this option to define your own weighing function that controls the relative weights of output signals during the estimation. This criterion minimizes the weighted sum of least square prediction errors. You

can specify the relative weighting of prediction errors for each output using the new `Weighting` field of the `Algorithm` property. By default, `Weighting` is an identity matrix, which means that all outputs are weighed equally. Set `Weighting` to a positive semidefinite symmetric matrix.

For more information about these new Algorithm fields for linear estimation, see the `Algorithm Properties` reference page. For more information about Algorithm fields for nonlinear estimation, see the `idnlarx` and `idnlhw` reference pages.

Note If you are estimating a single-output model, `det` and `trace` values of the `Criterion` field produce the same estimation results.

Improved Handling of Initial States for Linear and Nonlinear Models

The following are new options to handle initial states for nonlinear models:

- For nonlinear ARX models (`idnlarx`), you can now specify a numerical vector for initial states when using `sim` or `predict` by setting the `Init` argument. For example:

  ```
  predict(model,data,'init',[1;2;3;4])
  ```

  where the last argument is the state vector.

  For more information, see the `sim` and `predict` reference pages.

- For Hammerstein-Wiener models (`idnlhw`), you can now choose to estimate the initial states when using `predict` or `nlhw` by setting `INIT='e'`.

  For more information, see the `predict` and `nlhw` reference pages.

If you want to specify your own initial states, see the corresponding model reference pages for a definition of the states for each model type.

If you do not know the states, you can use the `findop` or the `findstates` command to compute the states. For more information about using these commands, see the `findop(idnlarx)`, `findop(idnlhw)`, `findstates(idnlarx)`, and `findstates(idnlhw)` reference pages.

To help you interpret the states of a nonlinear ARX model, you can use the `getDelayInfo` command. For more information, see the `getDelayInfo` reference page.

The `findstates` command is available for all linear and nonlinear models. Also see the `findstates(idmodel)` and `findstates(idnlgrey)` reference pages.

Improved Algorithm Options for Linear Models

The following improvements are available for the `Algorithm` property of linear models to align linear and nonlinear models (where appropriate) and improve robustness for default settings:

- The `SearchDirection` field (`model.Algorithm.SearchDirection`) has been renamed to `SeachMethod` (`model.Algorithm.SearchMethod`) to be consistent with the nonlinear models, where the corresponding field is `SeachMethod`.
- The new `lsqnonlin` option for specifying `SearchMethod` is available. `model.Algorithm.SearchMethod='lsqnonlin'` uses the `lsqnonlin` optimizer from the Optimization Toolbox™ software. You must have Optimization Toolbox software installed to use this option.
- The improved `gn` algorithm (subspace Gauss-Newton method) is available for specifying `SearchDirection`. The updated `gn` algorithm better handles the scale of the parameter Jacobians and is also consistent with the algorithm used for nonlinear model estimation.
- The default values for the `LimitError` field of the `Algorithm` property (`modelname.Algorithm.LimitError`) is changed to 0, which is consistent with the corresponding option for estimating nonlinear models. In previous releases, `LimitError` default value was 1.6, which robustified the estimation process against data outliers by associating a linear penalty for large errors, rather than a quadratic penalty. Now, there is no robustification by default (`LimitError=0`). You can estimate the model with the default setting and plot the prediction errors using `pe(data.model)`. If the resulting plot shows occasional large values, repeat the estimation with `model.Algorithm.LimitError` set to a value between 1 and 2.
- The `model.Algorithm.Advanced` property has a new tolerance field `GnPinvConst` corresponding to the `gn SearchMethod`. `GnPinvConst`

specifies that singular values of the Jacobian that are smaller than `GnPinvConst*max(size(J))*norm(J)*eps` are discarded when computing the search direction. You can assign a positive real value for this field. Default value is `1e4`.

- The default value of `model.Algorithm.Advanced.Zstability` property has been changed from `1.01` to `1+sqrt(eps)`. The new default reduces the possibility of a situation where the estimation algorithm does not converge (predictor becomes unstable) while still allowing enough flexibility to capture lightly damped modes.

For more information about Algorithm properties of linear models, see the `Algorithm Properties` reference page.

New Block Reference Pages

New documentation for System Identification Toolbox blocks is provided. For more information, see “Block Reference” in the System Identification Toolbox reference documentation.

Functions and Properties Being Removed

For more information about the process of removing functions, see "About Functions and Properties Being Removed" in “What Is in the Release Notes” on page 2.

Function or Property Name	What Happens When You Use Function or Property?	Use This Instead	Compatibility Considerations
`lintan`	Still runs	`linearize(idnlhw)` `linearize(idnlarx)`	See “Linearizing Nonlinear Black-Box Models at User-Specified Operating Points” on page 10.

Function or Property Name	What Happens When You Use Function or Property?	Use This Instead	Compatibility Considerations
`model.Algorithm.SearchDirection`	Still runs	`model.Algorithm.SearchMethod`	See "Improved Algorithm Options for Linear Models" on page 13.
`gns` option of `model.Algorithm.SearchDirection`	Still runs	`gn`	See "Improved Algorithm Options for Linear Models" on page 13.
`GnsPinvTol` of `model.Algorithm.Advanced`	Still runs	`GnPinvConst`	See "Improved Algorithm Options for Linear Models" on page 13.

Version 7.1 (R2007b) System Identification Toolbox Software

This table summarizes what's new in Version 7.1 (R2007b):

New Features and Changes	Version Compatibility Considerations	Fixed Bugs and Known Problems	Related Documentation at Web Site
Yes Details below	No	Bug Reports	No

New feature introduced in this version:

New Polynomial Nonlinearity Estimator for Hammerstein-Wiener Models

You can now estimate nonlinearities for Hammerstein-Wiener models using a single-variable polynomial at either the input or the output. This nonlinearity estimator is available at the command line.

For more information, see the `poly1d` reference pages. For more information about estimating Hammerstein-Wiener models, see "Identifying Hammerstein-Wiener Models" in the System Identification Toolbox documentation.

Version 7.0 (R2007a) System Identification Toolbox Software

This table summarizes what's new in Version 7.0 (R2007a):

New Features and Changes	Version Compatibility Considerations	Fixed Bugs and Known Problems	Related Documentation at Web Site
Yes Details below	No	Bug Reports	No

New features and changes introduced in this version are:

- "New Nonlinear Black-Box Modeling Options" on page 17
- "New Nonlinear Grey-Box Modeling Option" on page 18
- "New Getting Started Guide" on page 19
- "Revised and Expanded User's Guide" on page 19

New Nonlinear Black-Box Modeling Options

You can now estimate nonlinear discrete-time black-box models for both single-output and multiple-output time-domain data. The System Identification Toolbox product supports the following types of nonlinear black-box models:

- Hammerstein-Wiener
- Nonlinear ARX

To learn how to estimate nonlinear black-box models using the System Identification Tool GUI or commands in the MATLAB Command Window, see the System Identification Toolbox documentation.

Note You can estimate Hammerstein-Wiener black-box models from input-output data only. These models do not support time-series data, where there is no input.

New demos are available to help you explore nonlinear black-box functions. For more information, see the collection of demos in the Tutorials on Nonlinear ARX and Hammerstein-Wiener Model Identification category.

New Nonlinear Grey-Box Modeling Option

You can now estimate nonlinear discrete-time and continuous-time models for arbitrary nonlinear ordinary differential equations using single-output and multiple-output time-domain data, or time-series data (no measured inputs). Models that you can specify as a set of nonlinear ordinary differential equations (ODEs) are called *grey-box models*.

To learn how to estimate nonlinear grey-box models using the commands in the MATLAB Command Window, see System Identification Toolbox documentation.

Specify the ODE in an M-file or a MEX-file. The template file for writing the MEX-file, `IDNLGREY_MODEL_TEMPLATE.c`, is located in `matlab/toolbox/ident/nlident`.

To estimate the equation parameters, first construct an `idnlgrey` object to specify the ODE file and the parameters you want to estimate. Use `pem` to estimate the ODE parameters. For more information, see the `idnlgrey` and `pem` reference pages.

New demos are available to help you explore nonlinear grey-box functions. For more information, see the collection of demos in the Tutorials on Nonlinear Grey-Box Model Identification category.

Optimization Toolbox Search Method for Nonlinear Estimation Is Supported

If you have Optimization Toolbox software installed, you can specify the `lsqnonlin` search method for estimating black-box and grey-box nonlinear models in the MATLAB Command Window.

```
model.algorithm.searchmethod='lsqnonlin'
```

For more information, see the `idnlarx`, `idnlhw`, and `idnlgrey` reference pages.

New Getting Started Guide

The System Identification Toolbox product now provides a new Getting Started Guide. This guide introduces fundamental identification concepts and provides the following tutorials to help you get started quickly:

- "Tutorial – Identifying Linear Models Using the GUI" — Tutorial for using the System Identification Tool graphical user interface (GUI) to estimate linear black-box models for single-input and single-output (SISO) data.
- "Tutorial – Identifying Low-Order Transfer Functions (Process Models) Using the GUI" — Tutorial for using the System Identification Tool graphical user interface (GUI) to estimate low-order transfer functions to fit single-input and single-output (SISO) data.
- "Tutorial – Identifying Linear Models Using the Command Line" — Tutorial for estimating models using System Identification Toolbox objects and methods for multiple-input and single-output (MISO) data.

Revised and Expanded User's Guide

The System Identification Toolbox documentation has been revised and expanded.

Version 6.2 (R2006b) System Identification Toolbox Software

This table summarizes what's new in Version 6.2 (R2006b):

New Features and Changes	Version Compatibility Considerations	Fixed Bugs and Known Problems	Related Documentation at Web Site
Yes Details below	No	Bug Reports	No

New feature introduced in this version:

MATLAB Compiler Support

The System Identification Toolbox product now supports the MATLAB® Compiler™ product.

You can use MATLAB Compiler to take M-files as input and generate redistributable, standalone applications that include System Identification Toolbox functionality, including the following:

- Creating data and model objects
- Preprocessing and manipulating data
- Simulating models
- Transforming models, including conversions between continuous and discrete time and model reduction
- Plotting transient and frequency response

To use these features, write an M-file that uses System Identification Toolbox commands. Use the MATLAB Compiler software to create a standalone application from the M-file. For more information, see the MATLAB Compiler documentation.

Standalone applications that include System Identification Toolbox functionality have the following limitations:

- No access to the System Identification library in the Simulink software (`slident`)
- No support for model estimation

Version 6.1.3 (R2006a) System Identification Toolbox Software

This table summarizes what's new in Version 6.1.3 (R2006a):

New Features and Changes	Version Compatibility Considerations	Fixed Bugs and Known Problems	Related Documentation at Web Site
Yes Details below	Yes Details below. See also Summary.	Bug Reports	No

New features and changes introduced in this version are:

- "balred Introduced for Model Reduction" on page 22
- "Search Direction for Minimizing Criteria Can Be Computed by Adaptive Gauss-Newton Method" on page 22
- "Maximum Number of Bisections Used by Line Search Is Increased" on page 23

balred Introduced for Model Reduction

Use `balred` to perform model reduction instead of `idmodred`.

Search Direction for Minimizing Criteria Can Be Computed by Adaptive Gauss-Newton Method

An adaptive Gauss-Newton method is now available for computing the direction of the line search for cost-function minimization. Use this method when you observe convergence problems in the estimation results, or as an alternative to the Levenberg-Marquard (`lm`) method.

The `gna` search method was suggested by Adrian Wills, Brett Ninness, and Stuart Gibson in their paper "On Gradient-Based Search for Multivariable System Estimates", presented at the IFAC World Congress in Prague in 2005. `gna` is an adaptive version of `gns` and uses a cutoff value for the singular values of the criterion Hessian, which is adjusted adaptively depending on the success of the line search.

Specify the gna method by setting the SearchDirection property to 'gna'. For example:

```
m = pem(data,model_structure,'se','gna')
```

The default initial value of gamma in the gna search is 10^-4. You can set a different value using the InitGnaTol property. For more information about SearchDirection, see the Algorithm Properties reference pages.

Maximum Number of Bisections Used by Line Search Is Increased

The default value for the MaxBisections property, which is the maximum number of bisections along the search direction used by line search, is increased from 10 to 25. This increases the number of attempts to find a lower criterion value along the search vector.

For more information about Search properties, see the Algorithm Properties reference page.

Functions and Properties Being Removed

For more information about the process of removing functions, see "About Functions and Properties Being Removed" in “What Is in the Release Notes” on page 2.

Function or Property Name	What Happens When You Use Function or Property?	Use This Instead	Compatibility Considerations
idmodred	Still runs	balred	See “balred Introduced for Model Reduction” on page 22.

Version 6.1.2 (R14SP3) System Identification Toolbox Software

This table summarizes what's new in Version 6.1.2 (R14SP3):

New Features and Changes	Version Compatibility Considerations	Fixed Bugs and Known Problems	Related Documentation at Web Site
No	No	Bug Reports	No

Version 6.1.1 (R14SP2) System Identification Toolbox Software

This table summarizes what's new in Version 6.1.1 (R14SP2):

New Features and Changes	Version Compatibility Considerations	Fixed Bugs and Known Problems	Related Documentation at Web Site
No	No	Fixed bugs	No

Version 6.0 (R13SP2) System Identification Toolbox Software

This table summarizes what's new in Version 6.0 (R13SP2):

New Features and Changes	Version Compatibility Considerations	Fixed Bugs and Known Problems	Related Documentation at Web Site
Yes Details below	Yes Summary	No bug fixes	V6.0 product documentation

New features and changes introduced in this version are:

- "idproc Model Object Added" on page 26
- "Estimation and Validation in Frequency Domain Now Supported" on page 27
- "Continuous-Time Data Can Now Be Stored Using Frequency-Domain Objects" on page 27
- "Simulink Software Now Supports iddata and idmodel Objects" on page 28
- "advice About Data and Models Now Available" on page 28
- "Theta Models No Longer Supported" on page 28

idproc Model Object Added

A new model object, `idproc`, is used to represent simple continuous-time process models. This object is characterized by static gain, possible dead time, and dominating time constant(s). A new GUI that supports this object is available in the System Identification Toolbox GUI.

To learn more about this object, type `iddemopr` at the MATLAB prompt to run a demo.

You can also try the command

```
m = pem(data,'p1d')
```

Estimation and Validation in Frequency Domain Now Supported

You can now perform estimation and validation using frequency-domain data, such as the following:

- Inputs and outputs, entered as frequency-domain data in the `iddata` object
- Frequency-response data from a frequency analyzer

Both System Identification Toolbox functions and the graphical user interface (GUI) support this.

All estimation, simulation, and validation routines accept frequency-domain data and frequency-response data as inputs, similar to time-domain data. The frequency-response data must be packaged as an `frd` or `idfrd` object.

Use the `fft` and `ifft`functions to transform between the time and frequency domains. Use the `spafdr` function to estimate frequency responses using frequency-dependent resolution.

Type at the MATLAB prompt:

```
help iddata
```

or

```
idprops data
```

for complete descriptions. To access a demo, type `iddemofr`.

Continuous-Time Data Can Now Be Stored Using Frequency-Domain Objects

You can now store continuous-time data as a frequency-domain data object. Continuous-time Fourier-transformed data is now stored at a finite number of arbitrary frequencies, letting you estimate continuous-time models directly. For example, type at the MATLAB prompt:

```
help oe
```

Simulink Software Now Supports iddata and idmodel Objects

You can now import and simulate System Identification Toolbox `idmodel` models in the Simulink environment. You can also import `iddata` objects into Simulink.

The command `slident` opens a System Identification block library, which you can use to simulate `idmodel` models (with or without noise). This library also contains data sources and sinks for `iddata` objects.

advice About Data and Models Now Available

Use the new `advice` command to get helpful tips about the quality, problems, and options associated with an `iddata` or `idmodel` object.

For more information, type at the MATLAB prompt:

```
help iddata/advice
```

and

```
help idmodel/advice
```

Theta Models No Longer Supported

Theta models (matrices) are no longer supported.

Functions and Properties Being Removed

For more information about the process of removing functions, see "About Functions and Properties Being Removed" in “What Is in the Release Notes” on page 2.

Function or Property Name	What Happens When You Use Function or Property?	Use This Instead	Compatibility Considerations
`th`	Errors	None	See "Theta Models No Longer Supported" on page 28.
`th2par`	Still runs	None	See "Theta Models No Longer Supported" on page 28.
`th2ss`	Still runs	None	See "Theta Models No Longer Supported" on page 28.

Compatibility Summary for System Identification Toolbox Software

This table summarizes new features and changes that might cause incompatibilities when you upgrade from an earlier version, or when you use files on multiple versions. Details are provided with the description of the new feature or change.

Version (Release)	New Features and Changes with Version Compatibility Impact
Latest Version V7.3.1 (R2009b)	None
V7.3 (R2009a)	None
V7.2.1 (R2008b)	See "Functions and Properties Being Removed" on page 8.
V7.2 (R2008a)	See "Functions and Properties Being Removed" on page 14.
V7.1 (R2007b)	None
V7.0 (R2007a)	None
V6.2 (R2006b)	None
V6.1.3 (R2006a)	See "Functions and Properties Being Removed" on page 23.
V6.1.2 (R14SP3)	None
V6.1.1 (R14SP2)	None
V6.0 (R13SP2)	See "Functions and Properties Being Removed" on page 28.