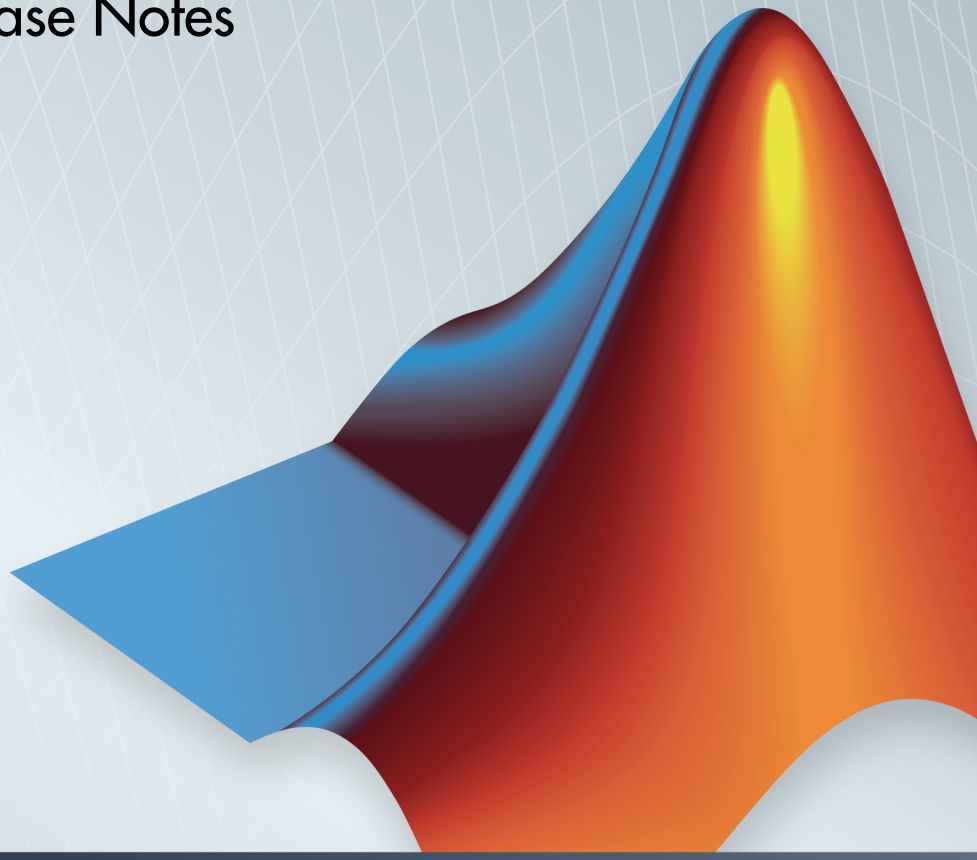
MATLAB® Release Notes



MATLAB®

MathWorks®

How to Contact MathWorks

Latest news: www.mathworks.com

Sales and services: www.mathworks.com/sales_and_services

User community: www.mathworks.com/matlabcentral

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

Phone: 508-647-7000

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

MATLAB® Release Notes

Contents

R2014b

R2014a

R2013b

R2013a

R2012b

R2012a

R2011a

R2010bSP2

Bug Fixes

R2010bSP1

Bug Fixes

R2010a

R2009bSP1

Bug Fixes

R2009b

R2008b

R2008a

R2007b

R2007a

R2006b

R2006a

R14SP3

R14SP2

R2014b

Desktop

Git and Subversion source control system integration through Current Folder browser, including syncing from Web-hosted repositories such as those on GitHub

The MATLAB® Current Folder browser includes a column with source code status for files that are contained in Git™ and Subversion® repositories. You can retrieve repositories and access source control functionality through the context menus. For more information, see “Source Control Integration”.

Packaging of custom MATLAB toolboxes into a single, installable file

You can package your toolbox as a single installer file (`.mltbx`) that contains all of the code, data, apps, documentation, and examples necessary to use your toolbox. Share your toolbox with others by uploading the installer file to the File Exchange or by sending it as an email attachment. For more information, see “Create and Share Toolboxes”.

Dialog box for managing custom MATLAB toolboxes

You can use the Manage Custom Toolboxes dialog box to view details about the installed custom toolboxes or to uninstall the toolboxes. To access the Manage Custom Toolboxes dialog box, go to the **Resources** section of the **Home** tab and select  **Manage Custom Toolboxes** from the **Add-Ons** menu.

Preference for controlling the initial working folder, with the option to start in the folder from your previous MATLAB session

For information about setting this option, see “General Preferences”.

Compatibility Considerations

On Windows® platforms, do not use the **Start in** field in the MATLAB Properties window. Instead, use the **Specify the full path to a folder** option in the Preferences panel.

Copying and pasting variables in the Workspace browser

You now can copy and paste the contents of workspace variables in the Workspace browser. Just right-click, select **Copy**, and then **Paste**. Alternatively, you can use the **Ctrl+C** and **Ctrl+V** keyboard shortcuts. Previously, this action opened the Import Tool for importing the variable name as a string.

Self-paced eLearning available from within MATLAB

MATLAB Academy is the entry point to self-paced eLearning from within MATLAB. Through MATLAB Academy, you can participate in interactive training courses to get you started with MATLAB.

To access MATLAB Academy from within MATLAB, go to the **Resources** section of the **Home** tab and select  **MATLAB Academy** from the **Help** menu. Alternatively, you can access MATLAB Academy from the "Getting Started with MATLAB" documentation or online at https://matlabacademy.mathworks.com/R2014b. You must have a MathWorks® account associated with an active license.

New startup switch to opt out of automatically switching to software OpenGL

If MATLAB detects it is running with a graphics driver that has known issues, it automatically switches to software OpenGL®. You can opt out of this behavior by calling MATLAB with the `-nosoftwareopengl` startup option. See `matlab (Windows)` or `matlab (UNIX)`.

To re-enable automatic switching to software OpenGL, call MATLAB with the `-softwareopengl` startup option.

These startup options are not available on Mac platforms.

Color settings preferences in Comparison Tool

You can change and save your color preferences for the MATLAB Comparison Tool. Apply your color preferences when comparing text files, MAT-files, variables, model files, zip files, or folders. For details, see "Change Color Preferences".

Automatic file saving when you click away from the Editor

If you are editing a file in the MATLAB Editor that you have saved at least once, new changes are automatically saved if you click away from the Editor.

This setting is on by default, and is accessible through MATLAB Editor/Debugger Preferences. On the **Home** tab, in the **Environment** section, click **Preferences**, and select **Editor/Debugger**. Disable the setting under **Automatic file changes** by clearing the **Save changes upon clicking away from a file** check box.

Language and Programming

`datetime`, `duration`, and `calendarDuration` arrays for efficient computation, comparison, and formatted display of dates and times

`datetime`, `duration` and `calendarDuration` are new data types for representing dates and times. You can manipulate arrays of these types in the same way that you work with numeric arrays. For example, you can add, subtract, sort, compare, concatenate, and plot date and time values.

The new data types support the following features:

- Functions for both calendar computations and fixed-length time computations on absolute dates and times, and elapsed times
- `TimeZone` property for setting and converting `datetime` time zone, accounting for daylight saving time
- Customizable default display format for dates
- Nanosecond precision for absolute times
- Import of data as `datetime` arrays using the Import Tool and the `readtable` and `textscan` functions

For more information, see “Dates and Time”.

Compatibility Considerations

When importing data from spreadsheets or text files using the Import Tool, there is no longer an option to convert dates to numeric serial date numbers. Instead, import the data into a table or as column vectors, and specify the `datetime` data type for each column of dates.

Suggested corrections for syntax errors in the Command Window

MATLAB suggests corrections for function names mistyped in the Command Window. This functionality now includes suggestions for:

- Missing closing parentheses or brackets such as `)`, `]`, and `}` in function calls and arrays

- Common assignment operators such as `++`, `+=`, and `-=`, and mathematical operator idioms such as `2(x+1)`
- Python® argument syntax and dictionary syntax

Suggested MathWorks products for undefined functions

MATLAB suggests which MathWorks product is required for an undefined function. This information includes links to documentation and product information. For example, if you do not have the Aerospace Toolbox,

```
angle2quat
```

```
To use 'angle2quat', you might need:
  angle2quat - Aerospace Toolbox
```

Prior to R2014b, MATLAB displayed an 'Undefined function' error.

Create multiple search indexes for help files you create

You can create multiple search indexes for help files you create: one for when you use MATLAB R2014a or earlier, and another for when you use MATLAB 2014b. MATLAB automatically uses the appropriate index for searching your documentation database. For details, see `builddocsearchdb`

py package for using Python functions and objects in MATLAB, and an engine interface for calling MATLAB from Python

For information about calling Python functions in MATLAB, see "Call Python Libraries".

For information about calling MATLAB from Python, see "MATLAB Engine for Python".

`matlab.wsdl.createWSDLClient` function for accessing SOAP-based Web services

For information about using `matlab.wsdl.createWSDLClient`, see "Access Services That Use WSDL Documents".

You must download supported versions of the Oracle® Java® JDK™ and the Apache™ CXF programs. For information, see "Set Up WSDL Tools".

RPC-encoded WSDL documents are not supported. For these documents, use `createClassFromWsdl`.

The following WSDL documents are not supported:

- Some documents with messages containing multiple parts.
- Some documents with schemas containing anonymous complex types.
- Documents that the Apache CXF program cannot compile into complete code.

Compatibility Considerations

`createClassFromWsdl`, `createSoapMessage`, `callSoapService`, and `parseSoapResponse` will be removed in a future release. Use `matlab.wsdl.createWSDLClient` instead.

Graphics objects in MEX-files use object handles instead of numeric handles

Graphics objects use object handles of various types instead of the numeric handles used in previous releases.

Compatibility Considerations

Do not write MEX-file code using the `mexGet` or `mexSet` functions, which assume a handle to a graphics object is a numeric value. Use the `mxGetProperty` or `mxSetProperty` functions instead. For more information, see “Upgrade MEX-Files to Use Graphics Objects”.

Workflow improvements when editing `classdef` files, including immediate impact on existing and new workspace variables

This feature eliminates the need to clear existing objects or call `clear classes` when changing the definition of the object's class. This feature also provides the ability to create objects based on the new definition without generating a warning that the class has changed. MATLAB updates existing objects according to the new class definition. For more information, see “Automatic Updates for Modified Classes”.

MATLAB errors attempting to define listener for nonobservable property

In previous releases, you could define a listener for a nonobservable property. MATLAB silently ignored the listener and did not call the listener callback. With release MATLAB 2014b, attempting to create a listener using `event.proplistener` or `addlistener` for a nonobservable property causes an error.

Compatibility Considerations

If your code used meta-data or the `properties` function to get a list of all properties defined by a class, and then assigns listeners to all properties, you should change your code. Rewrite the code to determine if a given property is observable before assigning the listener. For example, this code determines which properties of `MyClass` are `SetObservable` or `GetObservable`.

```
mc = ?MyClass;
p = findobj(mc.PropertyList,'SetObservable',1,'-or','GetObservable',1)
```

`p` contains the `meta.property` objects for the properties of `MyClass` that are `SetObservable` or `GetObservable`.

Script-based tests in unit testing framework

The MATLAB unit testing framework now provides script-based writing, execution, and verification of tests as an alternative to writing function-based or class-based tests. For more information, see “Write Script-Based Unit Tests”.

Plugin to report code coverage in unit testing framework

The `matlab.unittest` testing framework provides a plugin that produces a code coverage report for MATLAB source code. For more information, see the `matlab.unittest.plugins.CodeCoveragePlugin` documentation.

Control logging and verbosity in unit testing framework

Test authors can control the verbosity of logged messages in their tests. To register messages at different verbosity levels, call the `log` method with a `TestCase` or `Fixture`. To control the verbosity of the output, use a plugin such as

`LoggingPlugin` or `TestRunProgressPlugin`, or a test runner constructed with the `TestRunner.withTextOutput` method.

Constraint for scalar values in unit testing framework

You can use the `matlab.unittest.constraints.IsScalar` constraint class to test that an actual value is a scalar.

Test suites from packaged functions and scripts

The unit testing framework recognizes function-based and script-based tests that are contained within a user-defined package. For more information, see `TestSuite.fromPackage` and `runtests`.

Failure of unit tests using a relative tolerance when the expected value is infinite and the actual value is finite

If a test qualification applies a relative tolerance and the expected value is infinite, the test fails if the actual value is finite. For example,

```
tc = matlab.unittest.TestCase.forInteractiveUse;
tc.verifyEqual(1,Inf,'RelTol',1e-12);
```

```
Interactive verification failed.

---------------------
Framework Diagnostic:
---------------------
verifyEqual failed.
--> The values are not equal using "isequaln".
--> The error was not within relative tolerance.
--> Failure table:
            Index    Actual    Expected    Error    RelativeError    RelativeTolerance
            _____    ______    ________    _____    _____________    _________________

            1        1         Inf         -Inf     NaN              1e-12

Actual double:
         1
Expected double:
       Inf
```

Compatibility Considerations

Change any instances of test qualifications in which you compare an infinite expected value to a finite actual value using a relative tolerance, and expect the test to pass.

`rmdir` treatment of asterisk as literal character on Linux and Mac

On Linux® and Mac platforms, the `rmdir` function treats the asterisk character (*) as a literal character if a file or folder named * exists in the current folder. If such a file or folder does not exist, then `rmdir` treats the asterisk as a wildcard character. On Windows, `rmdir` always treats * as a wildcard character.

Previously, `rmdir` treated * as a wildcard character on all platforms.

Compatibility Considerations

Change code that expects to remove all files and folders using a wildcard character, on Linux and Mac.

Functionality being removed or changed

Functionality	Result	Use This Instead	Compatibility Considerations
`createClassFromWsdl`	Warns	`matlab.wsdl.createWSDLClient`	Replace all instances of `createClassFromWsdl` with `matlab.wsdl.createWSDLClient`. RPC-encoded WSDL documents are not supported.

`createSoapMessage`
`callSoapService`
`parseSoapResponse`
	Still runs	`matlab.wsdl.createWSDLClient`	Replace all instances of `createClassFromWsdl` with `matlab.wsdl.createWSDLClient`.

`matlab.unittest.TestCase` constructor	Still runs	`matlab.unittest.TestCase.forInteractiveUse`	Replace all instances of creating a `TestCase`

Functionality	Result	Use This Instead	Compatibility Considerations
			object using the default constructor. Use the `TestCase.forInteractiveUse` static method for interactive, command line use. When tests are run in the unit testing framework, instances are created by the test runner.
`matlab.unittest.plugins.TestSuiteProgress-Plugin`	Still runs	`matlab.unittest.plugins.TestRunProgressPlugin`	Replace all instances of `TestSuiteProgressPlugin`. To construct a plugin with the same level of detail, use `TestRunProgressPlugin.withVerbosity(2)`.
`mexGet` `mexSet`	Errors	`mxGetProperty` `mxSetProperty`	For MEX-files, replace all instances of `mexGet` and `mexSet` with `mxGetProperty` or `mxSetProperty` in the C/C++ and Fortran Matrix Library.

Mathematics

`histcounts` function for binning numeric data

The `histcounts` function sorts data into bins with data-dependent bin picking and options for bin control and normalization.

`triangulation` functions `nearestNeighbor` and `pointLocation` for identifying the closest vertex and enclosing triangle or tetrahedron for specified point

The `nearestNeighbor` function identifies the closest vertex to a query point, and the `pointLocation` function identifies the enclosing triangle or tetrahedron for a query point.

Option for interpolating to `'next'` and `'previous'` neighbors with the `interp1` function and `griddedInterpolant` class

The `interp1` function and `griddedInterpolant` class now have 1-D support for interpolating to `'next'` and `'previous'` neighbors.

The computation time and memory requirements for `'next'` and `'previous'` are the same as `'nearest'`.

Option for rounding numbers to a specified number of decimal or significant digits using the `round` function

The `round` function can now round to any specified number of decimal or significant digits.

For example, you can round `pi` to 2 decimal digits:

```
round(pi,2)
ans =

    3.1400
```

Or, you can round `pi` to 2 significant digits:

```
round(pi,2,'significant')
ans =

    3.1000
```

`boundary` function and `alphaShape` class for creating a conforming boundary around a discrete set of points

The `boundary` function and `alphaShape` class create enveloping boundaries, polygons, or polyhedra around a discrete set of 2-D or 3-D points, including options for tightening or loosening the boundaries around the points.

`cummin` and `cummax` functions for computing cumulative minimum and maximum of an array

The `cummax` and `cummin` functions compute the cumulative maximum and minimum values in an array. For example,

```
A = magic(4)

A =

    16     2     3    13
     5    11    10     8
     9     7     6    12
     4    14    15     1

B = cummin(A)

B =

    16     2     3    13
     5     2     3     8
     5     2     3     8
     4     2     3     1

C = cummax(A)

C =

    16     2     3    13
    16    11    10    13
```

```
    16     11     10     13
    16     14     15     13
```

Reverse accumulation option for the `cumsum`, `cummin`, `cummax`, and `cumprod` functions

The `'reverse'` option for `cumsum`, `cumprod`, `cummin`, and `cummax` reverses the direction of cumulation, working from `end` to `1` in the active dimension. This option allows quick directional calculations without requiring a flip or reflection of the input array.

Median and mode calculations of categorical data

The `median` and `mode` functions now support categorical arrays as inputs.

Functionality being removed or changed

Functionality	Result	Use Instead	Compatibility Considerations
`bitshift(A,k,N)`	Errors	`bitshift(A,k, assumedtype)`	Replace all instances of `bitshift(A,k,N)` with `bitshift(A,k,assumedtype)`.
`bitcmp(A,N)`	Errors	`bitcmp(A, assumedtype)`	Replace all instances of `bitcmp(A,N)` with `bitcmp(A,assumedtype)`.
`histc` function	Still Runs	`histcounts` function	Replace all instances of `histc` with `histcounts`

Data Import and Export

Faster data import from text files using Import Tool, and `readtable` and `textscan` functions

The performance of the Import Tool, the `readtable` function, and the `textscan` function improves for text files.

Import of data as `categorical` and `datetime` arrays using the `readtable` and `textscan` functions

The `readtable` and `textscan` functions can read data from text files a categorical or datetime arrays. Use the `%C` conversion specifier to read text as a category name. Use the `%D` conversion specifier to read text as a datetime value.

Data import from text files and collections of text files that do not fit into memory with `datastore`

The `datastore` function creates a datastore for reading collections of data that are too large to fit in memory. For example, `TabularTextDatastore` works with collections of text files.

`VideoReader` performance improvements and ability to start reading from a specified time in the video

The `VideoReader.hasFrame` and `VideoReader.readFrame` methods allow for improved performance when reading video files. The methods check for and read the next available frame in the video. You can set the `CurrentTime` property to begin reading from a specified time in the video file. This simplifies seeking into variable frame-rate video.

Compatibility Considerations

When you first create the `VideoReader` object, `VideoReader` no longer automatically populates the `NumberOfFrames` property . This behavior greatly improves performance, especially for large files. The `NumberOfFrames` property will be removed in a future release, and is now hidden when displaying the properties of a `VideoReader`

object. When you use the `get` function to query a `VideoReader` object, the output structure array now has a `CurrentTime` field and no longer has a `Type` field or a `NumberOfFrames` field. You cannot query the `NumberOfFrames` property if you have invoked the `readFrame` or `hasFrame` methods, or if you have explicitly set the `CurrentTime` property.

If `VideoReader` cannot determine the duration of the video file, then the `Duration` property is empty (`[]`). Previously the `Duration` property was set to zero.

`tcpclient` function for reading and writing data from network connected devices and servers using socket-based connections

TCP, or Transmission Control Protocol, is a highly used networking protocol. The MATLAB TCP/IP client support uses raw socket communication and lets you connect to remote hosts in MATLAB for reading and writing data. For example, you could use TCP/IP to acquire data from a remote weather station, and then plot the data.

You can create a TCP/IP connection to a server or hardware and perform read/write operations. Use the `tcpclient` function to create the connection, and the `write` and `read` functions for synchronously reading and writing data.

`webread` function for importing online data including JSON, CSV, and image data

The `webread` function reads content from RESTful Web services. You can use `webread` to import image, text, and JSON data from Web services into MATLAB. You also can use `websave` to write data imported from a Web service to file.

Reading non-ASCII encoded files with `readtable` function

The `readtable` function can read text files with non-ASCII character encoding schemes. Use the optional `'FileEncoding'` name-value pair argument to specify the encoding.

Writing quoted strings with `writetable` function

The `writetable` function can write quoted strings to text files. Use the optional `'QuoteStrings'` name-value pair argument to enclose MATLAB strings in double quotation marks when writing to a file.

`hdftool` functionality will not be removed

The R2013a Release Notes originally stated that `hdftool` would be removed in a future release. As of R2014b, there are no plans to remove this functionality. MATLAB no longer issues a warning when you call `hdftool`.

Functionality being removed or changed

Functionality	What Happens When You Use This Functionality?	Use This Instead	Compatibility Considerations
`movie2avi`	Still Runs	`VideoWriter`	Remove all instances of `movie2avi`. Write to AVI files with `VideoWriter`.
`avifile`	Errors	`VideoWriter`	Replace all instances of `avifile` with `VideoWriter`.
`aviread`	Errors	`VideoReader`	Replace all instances of `aviread` with `VideoReader`.
`exifread`	Errors	`imfinfo`	Replace all instances of `exifread` with `imfinfo`.
`urlread`	Still runs	`webread`	Use `webread` only for reading content with HTTP GET requests. To send data using HTTP POST, continue to use `urlread`.
`urlwrite`	Still runs	`websave`	Replace all instances of `urlwrite` with `websave`.
`read` method of `VideoReader` class	Still runs	`readFrame` method of `VideoReader` class	Replace all instances of `read` with `readFrame`.
`NumberOfFrames` property of `VideoReader` class	Still runs	none	Remove all instances of `NumberOfFrames`. When reading a video file, use the `CurrentTime` property to specify where reading should begin.

Functionality	What Happens When You Use This Functionality?	Use This Instead	Compatibility Considerations
`hdfgd`	Still runs	`matlab.io.hdfeos.gd`	Replace all instances of `hdfgd` with the corresponding function in the `matlab.io.hdfeos.gd` package.
`hdfsd`	Still runs	`matlab.io.hdf4.sd`	Replace all instances of `hdfsd` with the corresponding function in the `matlab.io.hdf4.sd` package.
`hdfsw`	Still runs	`matlab.io.hdfeos.sw`	Replace all instances of `hdfsw` with the corresponding function in the `matlab.io.hdfeos.sw` package.

Hardware Support

Documentation installation with hardware support package

Starting in R2014b, each hardware support package installs with its own documentation. See “Supported Hardware” for more information on support packages.

Support package for Android sensors

The MATLAB Support Package for Android™ Sensors allows you to collect sensor data from your mobile Android device and log it in MATLAB. You can then process the sensor data in MATLAB in a variety of ways, including creating plots. You can collect data from the following sensors:

- Acceleration
- Angular Velocity
- Orientation
- Magnetic Field
- Position

The Android Sensors product requires the following:

- MATLAB Mobile™ must be installed on your Android phone. You can acquire this app though the Google Play™ Store.
- MATLAB is required for creating the connection to the app and logging sensor data.
- Download and install the MATLAB Support Package for Android Sensors.

The support package includes the command-line interface in MATLAB and the ability to activate the Android Sensors mobile app, which is a separate tab in MATLAB Mobile. You must create a `mobiledev` object in MATLAB to connect to the sensors on the phone, and collect the data. The connection between your computer running MATLAB and the phone is done via Wi-Fi or your cellular network. To get started, see “MATLAB Support Package for Android Sensors”.

Support package for Arduino hardware

The MATLAB Support Package for Arduino® Hardware allows you to send and receive data on Arduino devices. You can then process this data in MATLAB in a variety of ways,

including creating plots. You can collect data from the following sensors connected to your Arduino hardware:

- I2C
- SPI
- Servo Motors

The support package includes the command-line interface in MATLAB. For more information see “MATLAB Support Package for Arduino Hardware”.

Support package for LEGO MINDSTORMS EV3 hardware

You can use MATLAB commands to connect to an EV3 brick and perform the following operations:

- Exchange data with sensors that are connected to the EV3 brick.
- Control EV3 motors.
- Interact with the LCD, status light, speaker, and buttons on the EV3 brick.
- Exchange data with the EV3 brick over USB, Wi-Fi, or Bluetooth.

See “Install Support for LEGO MINDSTORMS EV3 Hardware”.

For more information, see “MATLAB Support Package for LEGO MINDSTORMS EV3 Hardware”.

Graphics

Major update of MATLAB graphics system

New MATLAB Graphics System

Starting in R2014b, the MATLAB graphics system is built on an improved infrastructure with a new look and includes many new features for improved charts and customizations. For more information on the new features and for compatibility considerations, see the remaining graphics release notes.

New look of MATLAB graphics with improved clarity and aesthetics

MATLAB graphics has a new look with many enhancements and new features, including:

- A new default colormap called `parula`, a lighter figure background color, and new colors when plotting multiple lines to emphasize plotted data.
- Anti-aliased fonts and lines for smoother text and graphics. For more information, see the `GraphicsSmoothing` property for figures and the `FontSmoothing` property for axes and text objects.
- New axes properties for setting the grid line colors and for controlling the title and axis label font sizes. For more information, see the `GridColor`, `TitleFontSizeMultiplier`, and `LabelFontSizeMultiplier` properties.
- Filled contour plot enhancements. For example, line breaks around the contour labels make the labels easier to read. Additionally, there are no longer outlines around the data limits, so they are easier to distinguish from contour lines.

Compatibility Considerations

Most of the new look changes are visual differences and should not affect your code. For troubleshooting topics related to the new look, see “Why Are Plot Lines Different Colors?” and “How Do I Make the Graph Title Smaller?”.

Improved infrastructure based on MATLAB objects

Starting in R2014b, the MATLAB graphics system is built on an improved infrastructure. Graphics objects now behave like other MATLAB objects:

- Graphics objects use object handles of various types instead of the numeric handles used in previous releases.
- Graphics objects support dot notation for getting and setting properties. Property names are case-sensitive when using dot notation.

Compatibility Considerations

Most code written for numeric handles still works with object handles. However, you should not perform operations that assume or require handles to be numeric values. For strategies to update existing code to work with object handles, see "Graphics Handles Are Now Objects, Not Doubles".

Rotatable axis tick labels

New axes properties support rotated axis tick labels. For more information, see the `XTickLabelRotation`, `YTickLabelRotation`, and `ZTickLabelRotation` property descriptions.

Automatic update of `datetime` and `duration` tick labels with `plot` function

The `plot` function now supports the `datetime` and `duration` data types. Axis ticks and labels automatically update with the zoom, pan, and resize operations.

`histogram` function for plotting histograms

The `histogram` function plots histograms with data-dependent bin picking and options for bin control, normalization, and visualization.

Unlike the `hist` function, when you specify an output argument with the `histogram` function, it returns a histogram object that you can use to modify the properties of the histogram. For example, you can change the number of bins, use a different binning algorithm, or normalize the histogram in several ways. For more information, see Using histogram Objects.

`animatedline` function for creating line animations

The `animatedline` function optimizes line animations by accumulating and plotting data from a streaming data source.

Display of multilingual text and symbols

Axis tick labels support TeX and LaTeX markup for special characters, such as superscripts, subscripts, and Greek letters. By default, the axes interprets tick label characters using TeX markup. For more information, see the `TickLabelInterpreter` property of the axes. You also can use Unicode characters in axis tick labels, as well as in user interface objects.

Support for multiple colormaps in single figure

Figures support using a unique colormap for each axes in the figure. Specify an axes colormap by passing the axes handle to the `colormap` function.

Pie charts of `categorical` data with automatic slice labels

Pie charts support `categorical` data and automatically label the slices.

Image conversion functions `rgb2gray` and `im2double`, no longer requiring Image Processing Toolbox

MATLAB now includes the `rgb2gray` and `im2double` functions, which previously required the Image Processing Toolbox™. Use `rgb2gray` to convert RGB images or colormaps to grayscale. Use `im2double` to convert an image to double-precision values.

`imshow` function for displaying images from matrices or files, no longer requiring Image Processing Toolbox

MATLAB now includes the `imshow` function, which previously required the Image Processing Toolbox. Use `imshow` to display images from matrices or files.

`savefig` option that creates more compact files

Saving figures in `.fig` files with `savefig` creates more compact files using the `'compact'` option. Use this option with `.fig` files to be opened only in MATLAB version R2014b or later.

Compatibility considerations for graphics changes

The graphics changes introduced in R2014b support most of the functionality from previous releases, although there are some differences. Some of the changes that you are most likely to encounter are listed here.

- Plotting multiple lines with `hold on` uses the next color in the color order instead of starting from the first color with each plotting command.
- The `Clipping` property clips plotted objects to the six sides of the axes box defined by the axis limits. In previous releases, the `Clipping` property clips objects to the smallest 2-D rectangle that encloses the axes.
- If you add new data to a graph after using an `axis` command, such as `axis tight`, then MATLAB automatically updates the limits to incorporate the new data. In previous releases, MATLAB does not update the axis limits based on changes in the data.
- Colorbars and legends are no longer axes objects. They are new types of objects with their own sets of supported properties.
- The `Children` property for charting objects, legends, and colorbars no longer contains handles to underlying objects.
- `copyobj` no longer copies callbacks and application data. If you have existing code that copies object callback properties and object application data, use the `copyobj` function with the `'legacy'` option. This behavior is consistent with versions of `copyobj` before MATLAB release R2014b.

For troubleshooting topics related to these changes, see "Graphics Changes in R2014b". For a list of removed properties and syntaxes, see the tables under "Properties and syntaxes being removed or changed" on page 1-24 and "Save and print functionality being removed or changed" on page 1-29.

Some graphics features might not work or might be unreliable because of outdated graphics drivers. For the best results with graphics, upgrade to the latest graphics drivers provided by your graphics hardware manufacturer. For more information, see "System Requirements for Graphics".

Properties and syntaxes being removed or changed

Functionality	Result	Use Instead	Compatibility Considerations
`hold all` syntax	Still runs	`hold on`	Replace all instances of `hold all` with `hold on`.

Functionality	Result	Use Instead	Compatibility Considerations
`hist` function	Still runs	`histogram` function	Use `histogram` instead of `hist`.
`colorbar('peer',ax)` syntax	Still runs	`colorbar(ax`	Replace all instances of `colorbar('peer',ax)` with `colorbar(ax)`.
Using `0` to access the root object	Still runs	`groot` command	Use the `groot` command to access the root object instead of `0`.
`EraseMode` property for all graphics objects	Warns	Not applicable	Remove all instances of the `EraseMode` property. For more information, see "How Do I Replace the EraseMode Property?".
`noanimate` function	Warns	Not applicable	Remove all instances of the `noanimate` function. For more information, see "How Do I Replace the EraseMode Property?".
`HitTestArea` property for group, transform, and chart objects	Errors	The `PickablePar` property	Remove all instances of the `HitTestArea` property. To control the area of an object that captures mouse clicks, use the `PickableParts` property.
Using the vertical slash character (|) in a string to separate text for tick labels or text objects	Does not separate text	Use a cell array of strings, a padded string matrix, or numeric vectors	Replace instances of using the vertical slash character (|) to define multiple strings with a cell array of strings, a padded string matrix, or numeric vectors.

Functionality	Result	Use Instead	Compatibility Considerations
Passing complex inputs to charting functions that do not support complex data	Warns or errors	Real valued input arguments	Charting functions that do not support complex data will warn or error if part of the data is ignored. Pass real valued inputs to these charting functions.
`'symbol'` option for `FontName` property	Warns	Use TeX or LaTex markup	Remove all instances of setting the `FontName` property to `'symbol'`. Use TeX or LaTeX markup instead. See the text `Interpreter` property for more information.
`'oblique'` option for `FontAngle` property	Uses italic instead	The `'italic'` option	Replace all instances of setting the `FontAngle` property to `'oblique'` with `'italic'`.
Using the `line` function with the dot value (`'.'`) for `LineStyle` property	Errors	The `':'` option for dotted lines	Replace all instances of setting the `LineStyle` property to `'.'` with `':'`.
`DrawMode` axes property	Warns	The `SortMethod` axes property	Remove all instances of the `DrawMode` property. Use the `SortMethod` property instead.
`NormalMode` property for surface and patch objects	Warns	The `VertexNorma` or `FaceNormals` property	Remove all instances of the `NormalMode` property. Use the `VertexNormalsMode` or `FaceNormalsMode` property instead.
`lighting phong` syntax	Uses gouraud lighting instead	`lighting gouraud`	Replace all instances of `lighting phong` with `lighting gouraud`.

Functionality	Result	Use Instead	Compatibility Considerations
`'phong'` value for the `EdgeLighting` and `FaceLighting` properties for surface and patch objects	Uses gouraud lighting instead	`'gouraud'` option	Replace all instances of setting the `EdgeColor` or `FaceColor` property to `'phong'` with the `'gouraud'` value instead.
Using the `Children` property for the axes to get the text objects for the title and axis labels	Does not return these objects	`Title`, `XLabel`, `YLabel`, and `ZLabel` axes properties	The text objects used for the title and axis labels are no longer children of axes. Use the `Title`, `XLabel`, `YLabel`, and `ZLabel` properties for the axes to get the text objects instead.
Passing text, image, rectangle, and annotation objects as an input argument to the `legend` function	Errors	Not applicable	Remove all instances of passing these objects to the `legend` function. Text, image, rectangle, and annotation objects are not included in the legend.
`neverselect`, `autoselect`, `advise`, `verbose` and `quiet` inputs for the `opengl` function	Warns	Not applicable	Remove all instances of using these inputs with the `opengl` function.
`'zbuffer'` option for the figure `Renderer` property	Uses OpenGL renderer instead	The `'opengl'` or `'painters'` options	Replace all instances of setting the `Renderer` property to `'zbuffer'` with `'opengl'` or `'painters'`. The default renderer is `'opengl'`.
ResizeFcn figure property	Still runs	`SizeChanged` figure property	Use of the ResizeFcn figure property is not recommended. Use the `SizeChangedFcn` figure property instead.

Functionality	Result	Use Instead	Compatibility Considerations
`FixedColors`, `MinColorMap`, `WVisual`, `WVisualMode`, `XVisual`, and `XVisualMode` figure properties	Errors	Not applicable	Remove all instances of figure properties that are no longer supported. Use supported figure properties.
`Selected` and SelectionHighlight figure properties	No effect	Not applicable	Remove all instances of the `Selected` and SelectionHighlight figure properties.
`'fullcrosshair'` option for the figure Pointer property	Warns	Not applicable	Remove all instances of setting the figure Pointer property to `'fullcrosshair'`.
The root properties `CommandWindowSize`, `Diary`, `DiaryFile`, `Echo`, `Format`, `FormatSpacing`, `Language`, `More`, `RecursionLimit`, `BeingDeleted`, `ButtonDownFcn`, `UIContextMenu`, `Clipping`, `CreateFcn`, `DeleteFcn`, `BusyAction`, `Interruptible`, `HitTest`, `Selected`, `SelectionHighlight`, and `Visible`	Errors	Not applicable	Remove all instances of root properties that are no longer supported. Use supported root properties. For a list, see Root Properties.
Specifying an output argument for the `triplot` function	Returns single chart line object	Not applicable	Specifying an output argument for the `triplot` function returns a single chart line. Remove instances of code that rely on `triplot` returning multiple chart lines.

Functionality	Result	Use Instead	Compatibility Considerations
Specifying a single output argument for the `voronoi` function	Returns a vector of two chart line objects	Not applicable	Specifying a single output argument the `voronoi` function returns a vector of two chart lines. Remove instances of code that rely on `voronoi` returning more than two chart lines.

Save and print functionality being removed or changed

Functionality	Result	Use Instead	Compatibility Considerations
`pagesetupdlg` function	Warns	`printprevie`	Replace all instances of `pagesetupdlg` with `printpreview`.
`-crossplatform` and `-setup` options for `printdlg` function	Warns	Not applicable	Remove all instances of using the `-crossplatform` and `-setup` options with `printdlg`.
Using `getframe` function to capture content outside of the figure window	Warns	Specify area within figure window	Specify an area within a figure window when using `getframe` function.
`-dsetup` print option	Errors	Not applicable	Remove all instances of the `-dsetup` print option.
`'all'` option for `hgsave` and `hgload` functions	Errors	Not applicable	Remove all instances of using `hgsave` and `hgload` functions with the `'all'` option.
Setting the `PaperSize` figure property when the `PaperUnits` figure property is set to `'normalized'`	Errors	Not applicable	Remove instances of setting the `PaperSize` property when the `PaperUnits` property is set to `'normalized'`.
`mmat` option for `saveas` function	Errors	`savefig` function	Remove all instances of using the `mmat` option with `saveas`. To save a figure to

Functionality	Result	Use Instead	Compatibility Considerations
			a file, use `hgsave` with the `.fig` extension.
`printdmfile` function and `print -dmfile` syntax	Errors	`savefig` function or the figure menu	Remove all instances of `printdmfile` and `print -dmfile`. To save a figure to a file, use the `hgsave` function with the `.fig` extension, or use **File** > **Generate Code** on the figure menu instead.
`-dill` graphics format print option	Errors	`-depsc` or `-dsvg` print options for vector graphics formats	Remove all instances of using the `-dill` print option. For vector graphics formats, use `-depsc` or `-dsvg` option instead.
`-adobecset` print option	Errors	Not applicable	Remove all instances of the `-adobecset` print option.
The printer driver options `-dbj10e`, `-dbj200`, `-dbjc600`, `-dbjc800`, `-depson`, `-deps9high`, `-depsonc`, `-ddnj650c`, `-ddjet500`, `-dcdjmono`, `-dcdjcolor`, `-dcdj500`, `-dcdj550`, `-ddeskjet`, `-dlaserjet`, `-dljetplus`, `-dljet2p`, `-dljet3`, `-dljet4`, `-dpxlmono`, `-dpaintjet`, `-dpjxl`, `-dpjetxl`, `-dpjxl300`	Errors	Not applicable	Remove all instances of using these print options. To use a particular printer, install the printer driver.
The `-dps`, `-dpsc`, `-deps`, and `-depsc` print options generate PostScript® Level 3 output instead of PostScript Level 1	PostScript Level 3 output	Not applicable	PostScript Level 1 output is no longer supported.

GUI Building

uitab and uitabgroup components for creating user interfaces with tabbed panels

The `uitab` function creates a tabbed panel in which you can group related components together with a clickable tab label. The `uitabgroup` function creates the container that manages user selection of uitabs.

Changes introduced with new graphics system

The new MATLAB graphics system introduces these changes to the GUI building tools:

- All GUI components behave like MATLAB objects.
- Improvements to the stacking behavior of components for more consistent behavior.
- Improvements to component positioning for more consistency among different components and different platforms.

See “Graphics Changes in R2014b” for an overview of all the major changes.

Compatibility Considerations

- In some cases, the conditions and timing of the ResizeFcn callback execution have changed. See “Why Has the Behavior of ResizeFcn Changed?” for more information.
- The front-to-back order, or stacking order, of overlapping components might display differently in some GUIs. See “Why Are Some Components Missing or Partially Obscured?” for more information.
- GUI objects no longer support the use of `handle.listener` to create event listeners. See “Why Does handle.listener Return an Error?” for more information.
- The layout of UI components might change if you specify units in your code. Set the Units property before the Position property to ensure that MATLAB interprets the Position property values in those units. If you specify the Units property *after* the Position property, MATLAB interprets the Position values in the current units and converts the values to the equivalent values in the units you specified.
- Figure annotations and child objects of uipanels cannot cross uipanel boundaries. Previous versions of MATLAB allow annotations and child objects to extend into

(or out of) the uipanel boundaries. Now, these items clip at the uipanel boundary. Annotations and child objects of uibuttongroups behave the same way when they encounter a uibuttongroup boundary.

- The `waitforbuttonpress` function is now figure-specific. The figure that is current when you call the `waitforbuttonpress` function is the only area in which users can press a key or click a mouse button to resume program execution. If you need to support multiple open figures, then use the `gcf` function to determine the current figure when you call `waitforbuttonpress`.

Functionality being removed or changed

Functionality	Result	Use Instead	Compatibility Considerations
`selectmoveresize`	No effect	`plotedit`	Replace all instances of `selectmoveresize` with `plotedit`.
The Selected and SelectionHighlight properties of `figure`, `uicontrol`, `uitable`, `uipanel`, `uibuttongroup`, `uipushtool`, and `uitoggletool`.	No effect	Not applicable	Remove all instances that use the Selected and SelectionHighlight properties. See “Properties and syntaxes being removed or changed” for more information about all the changes to figure properties.
The ResizeFcn property of `figure`, `uipanel`, and `uibuttongroup`.	Still runs	SizeChangedF property	Use of the ResizeFcn property is not recommended. Use the SizeChangedFcn property instead. See “Properties and syntaxes being removed or changed” for more information about all the changes to figure properties.
The `'fullcrosshair'` option for the figure Pointer property	Warning	Not applicable	Remove all instances of setting the figure Pointer property to `'fullcrosshair'`.

Functionality	Result	Use Instead	Compatibility Considerations
			See “Properties and syntaxes being removed or changed” for more information about all the changes to figure properties.
The SelectionChangeFcn property of `uibuttongroup`	Still runs	SelectionChar property	Use of the SelectionChangeFcn property is not recommended. Use the SelectionChangedFcn property instead.
The Clipping property of `uicontrol`, `uitable`, and graphics `root`	Error	Not applicable	Remove all instances that use the Clipping property of a uicontrol, uitable, and graphics root. See “Properties and syntaxes being removed or changed” for more information about all the changes to root properties.
The `'none'` option for the uipanel and uibuttongroup BackgroundColor property.	Error	Not applicable	Remove all instances of setting the uipanel or uibuttongroup BackgroundColor property to `'none'`.

Performance and Big Data

Big data analysis on your desktop that can scale to Hadoop with `mapreduce`

The `mapreduce` function enables analysis of data sets that do not fit in your computer's memory. It is used to process large data sets on your desktop, and can also be extended to run on Hadoop® to process big data. MapReduce is a powerful technique for applying data processing methods to very large data sets, from simple statistics to complex machine learning algorithms.

For more information, including a selection of examples, see "MapReduce".

The functionality of `mapreduce` extends beyond MATLAB with the following products:

- Access relational databases using Database Toolbox™
- Increased performance on desktops with Parallel Computing Toolbox™
- Scaling up to Hadoop using MATLAB Distributed Computing Server™
- Create deployable archives or standalone applications that run against Hadoop using MATLAB Compiler™

Improved performance for sorting categorical data with `sort`

The performance of the `sort` function improves for large categorical array inputs.

`typecast` function performance improvements with long vectors

The performance of the `typecast` function improves for long input vectors.

R2014a

Desktop

Pop-up Command History for recalling, viewing, filtering, and searching recently used commands in the Command Window

Pop-up Command History (2 min, 41 sec)

The Command History now displays in response to the up arrow (↑) in the Command Window, by default. Previously, the Command History window occupied a designated space on the MATLAB desktop. The following are additional features enhancing the Command History.

- Colored marks on the left side of the Command History indicate commands that generate errors. These marks are of the same color as error messages in the Command Window.
- You can perform case-insensitive searches in the Command History, as well as search for partial matches anywhere in a command.
- You can select multiple commands using **Shift** + ↑ and rerun them at once. In addition, brackets display on the left side of the Command History to indicate commands processed as a group. Select the bracket to rerun the group.
- You can filter commands to display only the results that match your search.
- The Command History can display the command execution time.

For more information, see Command History.

Compatibility Considerations

To view the Command History window in a docked location as in previous releases of MATLAB, click  and then select **Dock**.

There is no longer any performance issue that justifies saving every *n*th command to the command history file. Therefore, this option no longer appears in the Command History Preferences. Instead, specify the total number of commands to save. The default is 25,000 commands, but you can specify a value up to 1,000,000. For more information, see Command History Preferences.

The history file is now named `History.xml`. Previously, it was named `history.m`. The first time you access an existing `history.m` file, MATLAB automatically converts it to `History.xml`, and you do not need to take further action.

Merge option in MATLAB Comparison Tool for resolving differences between text files

When comparing text files in the MATLAB Comparison Tool you can now merge changes from one file to the other. Merging changes can be useful when resolving conflicts between different versions of files.

For details, see Comparing Text Files.

Saving workspace variables and their values to a MATLAB script

MATLAB now provides the ability to save workspace variables to a MATLAB script. Once the script is saved, you can regenerate the workspace variables by running the script. Click **Save Workspace** on the MATLAB desktop and select **MATLAB script (*.m)** in the **Save as type** menu. Alternately, use matlab.io.saveVariablesToScript to perform this operation from the command line.

Variables that cannot be saved to a script are saved to a MAT-file with the same name as that of the script.

Korean and Chinese localization available on Windows and Mac platforms

If the Language setting on your computer is Korean or Chinese, then MATLAB now has a localized user interface and documentation.

Korean and Chinese documentation is also available on the Web. For details, see Korean and Chinese Documentation.

MathWorks file properties displayed for .SLX, .SLXP, and .MLAPPINSTALL files in file browsers and search engines on Windows and Mac systems

Users can find and organize MathWorks files that are based on the Open Packaging Conventions format (OPC) directly through the Windows and Mac operating systems. MathWorks OPC files are files with `.slx`, `.slxp`, or `.mlappinstall` file extensions.

Use the `Tags` property to add custom searchable text to the file.

Language and Programming

Suggested corrections for mistyped, user-defined functions in the Command Window

MATLAB suggests corrections for function names mistyped in the command window. This functionality now includes suggestions for custom, or user-defined, functions on the MATLAB path.

Streamlined MEX compiler setup and improved troubleshooting

`mex -setup` is no longer necessary in most situations

Setting up `mex` compilers has been simplified. For most users, there is no longer any need to run mex `-setup`. `mex` automatically locates and uses a supported installed compiler.

`mex` reports selected compiler and success status

`mex`, by default, identifies the compiler it is using at build time and reports success. To suppress this feature, use the new `-silent` option. `mex` still reports errors and warnings, even when you specify `-silent`.

`mex` maintains different settings for C and C++

`mex` maintains different settings for C and C++ compilers.

Compiling for MATLAB engine applications is different

To build an engine application or a standalone application to read MAT-files, use the `mex` command with the new `-client engine` option. For more information, see What You Need to Build Engine Applications or What You Need to Build Custom Applications.

`mex` uses standard quoting and no escape characters

If you specify *`varname=varvalue`* parameters when you invoke `mex`, you no longer need to use double-quotes (") on Windows, or escape the `$` character on Linux. To redefine a variable, use MATLAB-style single quotes (') and no escape characters.

`mex -setup` takes a language setting

`mex -setup` selects a compiler for a given language, `lang`. If `lang` is not specified, `mex -setup` searches for C compilers.

Compatibility Considerations

- Do not use the `-f` option to build engine and MAT-file applications. Use the `-client engine` option instead.
- The format of the `mex` configuration files has changed. If there is a `.bat` or `.sh` options file in the current or `prefdir` folder, MATLAB displays a warning. In a future release, these warnings will become errors.
- Informational messages from `mex` have changed. The `mex` command displays the selected compiler and success status. To suppress these messages, use the `-silent` option.
- The output of the `mex -setup` command has changed. Use `mex -setup` to change the default compiler for a given language. For more information, see Changing Default Compiler.
- The `mex -setup` syntax has changed. You can specify a language. If none is specified, `mex -setup` uses `C`.

Multidimensional array support for `flipud`, `fliplr`, and `rot90` functions

The flipud, fliplr, and rot90 functions now support arrays with any number of dimensions. These functions operate independently on the planes formed by the first and second dimensions.

Option for `circshift` to operate on a specified dimension

The syntax `circshift(A,K,dim)` allows you to shift the elements of array `A` by `K` positions along dimension `dim`. For more information, see the reference page for circshift.

Compatibility Considerations

The behavior of `Y = circshift(A,K)` for scalar `K` will change in a future release. Currently, the function acts on the first dimension of `A` by default. In a future release, the function will operate on the first dimension of `A` whose size does not equal 1 by default. To retain current behavior, use `circshift(A,[K 0])`.

Changes to empty string matching with `validatestring`

validatestring now validates empty strings.

```
V = validatestring('',{'Cantor','','Koch'})

V =

     ''
```

In versions of MATLAB prior to R2014a, the call to `validatestring` results in an error.

Compatibility Considerations

Change code that relies expects an error when testing the validity of an empty string.

`matlab.lang.makeValidName` and `matlab.lang.makeUniqueStrings` functions for constructing unique MATLAB identifiers

The matlab.lang.makeValidName function returns valid MATLAB identifiers from input strings. The output names are not guaranteed to be unique. For example,

```
S = {'Item#','Item#','Price/Unit','1st order','Contact'};
N = matlab.lang.makeValidName(S)

N =

    'Item_'    'Item_'    'Price_Unit'    'x1stOrder'    'Contact'
```

The matlab.lang.makeUniqueStrings function returns unique strings from input strings. For example,

```
S = {'coffee' 'tea' 'coffee' 'water' 'coffee'};
U = matlab.lang.makeUniqueStrings(S)

U =

    'coffee'    'tea'    'coffee_1'    'water'    'coffee_2'
```

Use these two new functions together to ensure that output strings are valid and unique MATLAB identifiers. For example,

```
S = {'Item#','Item#','Price/Unit','1st order','Contact'};
N = matlab.lang.makeValidName(S);
U = matlab.lang.makeUniqueStrings(N,{},namelengthmax)

U =
```

```
    'Item_'    'Item__1'    'Price_Unit'    'x1stOrder'    'Contact'
```

`details` function displays details about arrays

For more information, see details.

Changes to passing empty object to `isprop`

Calls to isprop with empty objects now return an empty logical array instead of `false`.

Compatibility Considerations

Change code that relies on `isprop` retuning `false` for empty objects.

Behavior change of `fullfile` function output

In R2012b and earlier, the fullfile function returns a full file specification that includes repeated file separators and relative path symbols if the input file parts include them. For example, repeated file separators include `\\` and `\\\`. Relative path symbols include the dot (.) and double-dot (..) notation.

As of R2013a, `fullfile` collapses inner repeated file separators and relative directories indicated by the dot and double-dot symbols, and does not display those symbols in the output full file specification. `fullfile` maintains repeated file separators only if they appear at the beginning of the full file specification. `fullfile` maintains dot and double-dot symbols only if they appear at either end of the full file specification.

For example, the commands,

```
f = fullfile('C:\foo\folder1', '..\folder2')
g = fullfile('C:\foo\folder1', '\\\folder3\\')
h = fullfile('\\folder4\.')
```

return:

```
f =
C:\foo\folder2

g =
```

```
C:\foo\folder1\folder3\

h =
\\folder4\.
```

In R2012b and earlier, the same commands returned:

```
f =
C:\foo\folder1\..\folder2

g =
C:\foo\folder1\\\folder3\\

h =
\\folder4\.
```

Compatibility Considerations

Change code that relies on locating exact strings containing repeated file separators or dot and double-dot symbols in a full file path.

Support array-creation functions in your class

Class authors can add support to their classes for the array creation functions (`ones`, `zeros`, `rand`, `eye`, `NaN`, `inf`, `true`, `false`, `cast`, `rand`, `randn`, and `randi`). This support includes the class name and prototype object syntaxes. For more information, see Class Support for Array-Creation Functions.

Custom plugins in unit testing framework

The `matlab.unittest` testing framework provides an interface class, matlab.unittest.plugins.TestRunnerPlugin, to create custom plugins and extend the `TestRunner`. For more information, see Write Plugins to Extend TestRunner and Create Custom Plugin.

Test parameterization and selection in unit testing framework

Test authors can write tests that are parameterized to combine and execute over lists of data. For more information, see Create Basic Parameterized Test and Create Advanced Parameterized Test.

The matlab.unittest.TestSuite.selectIf method, combined with classes in the matlab.unittest.selectors package, allows for the improved selection of tests included in the test suite.

`matlab.unittest` plugin for Test Anything Protocol (TAP) output

The `matlab.unittest` testing framework provides a plugin that produces a Test Anything Protocol (TAP) stream. This plugin allows integration of MATLAB unit test results into third-party systems that recognize the Test Anything Protocol such as continuous integration systems. For more information, see the matlab.unittest.plugins.TAPPlugin documentation.

Output stream direction for `matlab.unittest` plugins

The `matlab.unittest` testing framework provides a means to redirect text output from several plugins to standard output (ToStandardOutput) or to a file (ToFile). Output stream direction is supported for the DiagnosticsValidationPlugin, FailureDiagnosticsPlugin, TAPPlugin, and TestSuiteProgressPlugin plugins.

Additionally, the OutputStream class provides an interface for test authors to create custom output streams.

Comparator for MATLAB objects in unit testing framework

The PublicPropertyComparator can be used with the IsEqualTo constraint to compare public properties of MATLAB objects recursively.

Changes to compiler support for building MEX-files

MATLAB supports Visual C++® 2013 compilers for building MEX-files on Microsoft® Windows 32- and 64-bit platforms.

Compatibility Considerations

To ensure continued support for building your MEX-files, consider upgrading to another supported compiler. For an up-to-date list of supported compilers, see the Supported and Compatible Compilers website.

Changes to External Programming Language Interfaces documentation

External Programming Language Interfaces documentation is divided into two new categories in Advanced Software Development.

- Calling External Functions—How to use functionality from other languages, such as Java, .NET, and C, in MATLAB.
- MATLAB API for Other Languages—How to interact with MATLAB and MATLAB data types from other language applications. How to write and call MEX-functions.

Functionality being removed or changed

Functionality	What Happens When You Use This Functionality?	Use This Instead	Compatibility Considerations
`circshift(A,K)` for scalar `K`.	Still runs	`circshift(A, [K 0])` to retain current behavior	In a future release, the behavior of `circshift(A,K)`, where `K` is a scalar, will change. The function will operate on the first array dimension whose size does not equal 1 by default. Currently, the function operates on the first array dimension by default. To retain current behavior, use `circshift(A,[K 0])`.
`flipdim(A,dim)`	Still runs	flip	Replace all instances of `flipdim(A,dim)` with `flip(A)` or `flip(A,dim)`.

Functionality	What Happens When You Use This Functionality?	Use This Instead	Compatibility Considerations
`bitmax`	Warns	flintmax	Replace all instances of `bitmax` with `flintmax`.
`tstool`	Errors	timeseries, tscollection, openvar, or plot	Replace all instances of `tstool` with timeseries, tscollection, openvar, or plot. To create a time series object, use timeseries. To create a time series collection with one or more `timeseries` objects, use tscollection. To open a time series object or collection in the Variables editor, use openvar. To plot a time series object, use plot.

`genvarname`	Still runs	matlab.lang.makeVa and matlab.lang.makeUn	Replace all instances of `genvarname` with matlab.lang.makeValidName and matlab.lang.makeUniqueStrings. For example,
			
			`S = {'my.Name','my_Name','my_N`
			`N = matlab.lang.makeValidName(`
			`U = matlab.lang.makeUniqueStri`

Functionality	What Happens When You Use This Functionality?	Use This Instead	Compatibility Considerations
			`{},namelengthmax);`
`depdir`	Warns	matlab.codetools.requ	Replace all instances of `depdir` with matlab.codetools.requiredFilesAnd However, `matlab.codetools.requiredFi` returns the full path of a required file, including the file name.
`depfun`	Warns	matlab.codetools.requ	Replace all instances of `depfun` with matlab.codetools.requiredFilesAnd However, `matlab.codetools.requiredFi` does not identify opaque classes, such as Java or COM classes. There is no replacement for this functionality.
`createCopy` method of `inputParser` class	Errors	copy	Replace all instances of `createCopy` with copy.

`mex -f` *`filepath`*`\`*`mexopts`*`.`
 	
`mex -f` *`filepath`*`/`*`mexopts`*`.`
	Warns	For building engine and MAT-file applications, use `mex -client engine`.
		
		Replace custom `mex` options files with mex command-line options.	

Functionality	What Happens When You Use This Functionality?	Use This Instead	Compatibility Considerations
`mexIsGlobal` in the C/C++ and Fortran MEX API Library	Always returns `false`	mxIsFromGlobalWS in the C/C++ Matrix Library API and Fortran Matrix Library API	Replace all instances of `mexIsGlobal` with mxIsFromGlobalWS in MEX-files.

Mathematics

`isdiag`, `isbanded`, `issymmetric`, `ishermitian`, `istril`, `istriu`, and `bandwidth` functions for testing matrix structure

The following functions test various aspects of matrix structure and are useful in simplifying numerical algorithms.

- ishermitian determines if a matrix is Hermitian or skew-Hermitian.
- issymmetric determines if a matrix is symmetric or skew-symmetric.
- istriu determines if a matrix is upper-triangular.
- istril determines if a matrix is lower-triangular.
- isdiag determines if a matrix is diagonal.
- bandwidth returns the upper and lower bandwidth of a matrix.
- isbanded determines if a matrix is within the specified upper and lower bandwidths.

`sylvester` function for solving the Sylvester equation

The sylvester function solves the Sylvester equation, $AX + XB = C$, for X.

Option for `eig` function for computing left eigenvectors

The eig function can now calculate the left eigenvectors of a matrix A explicitly.

For example:

```
A = magic(3);
[V,D,W] = eig(A)
```

```
V =

   -0.5774   -0.8131   -0.3416
   -0.5774    0.4714   -0.4714
   -0.5774    0.3416    0.8131


D =

   15.0000         0         0
```

```
         0    4.8990         0
         0         0   -4.8990


W =

   -0.5774   -0.7416   -0.0749
   -0.5774    0.6667   -0.6667
   -0.5774    0.0749    0.7416
```

This produces a full matrix, `W`, whose columns are the left eigenvectors of `A` such that `W'*A = D*W'`.

Option for `rand`, `randi`, and `randn` functions for creating arrays of random numbers that match data type of an existing variable

The functions rand, randi, and randn can now return output that matches the data type of an existing variable.

For example:

```
A = int16(32);
r = randi(A,4,4,'like',A);
class(r)
```

```
ans =

int16
```

Integer type support for `mean`

The mean function now supports inputs of any integer data type.

`complex` function with one complex input

The complex function now supports one complex input. If `X` is complex, then `z = complex(X)` is identical to `X`. In previous releases, `complex` returned an error.

Change to `ind2sub` and `sub2ind` functions with nondouble inputs

The output behavior of the `ind2sub` and `sub2ind` functions has changed. The new output of these functions always has class `double` regardless of the class of the input.

Compatibility Considerations

In previous releases, the output class of these functions was dependent on the input class. To obtain nondouble output, cast the output into the required class, such as `int8(sub2ind(size,i,j))`.

Data Import and Export

Webcam support for previewing and acquiring live images and video

Webcam Support (2 min, 54 sec)

You can use the MATLAB Webcam support to bring live images from any USB Video Class (UVC) Webcam into MATLAB. This includes Webcams that might be built into laptops or other devices, as well as Webcams that plug into your computer via a USB port.

With simple MATLAB functions you can detect your connected Webcams, acquire single snapshots from a Webcam, and optionally set up a loop of acquired images. The `webcamlist` function allows you to detect the connected Webcams. The `webcam` function creates the Webcam object that is used to acquire images. And the `snapshot` function returns a single image from the camera. You can also preview your image and set properties for the image.

The Webcam support is available only through the Hardware Support Packages. You must download and install the necessary files using the Support Package Installer. To open the Support Package Installer, type `supportPackageInstaller` in MATLAB. Then on the **Select support package to install** screen, select the `USB Webcams` from the list. For more information on installing this support package, see Installing the Webcam Support Package.

The MATLAB Webcam support can be used on the following platforms:

- Microsoft Windows 32-bit and 64-bit
- Mac OS X 64-bit
- Linux

For more information about using the Webcam feature, see

- Connecting to Webcams — how to use the `webcamlist` function to detect your cameras
- Acquiring Images from Webcams — how to acquire live images from your camera into MATLAB
- Setting Properties for Webcam Acquisition — how to set object- or device-specific properties for the acquisition

Raspberry Pi hardware support for controlling devices such as motors and actuators, and for capturing live data from sensors and cameras directly from MATLAB

You can use MATLAB commands to connect to a Raspberry Pi™ board over a network and perform the following operations:

- Exchange data with sensors and actuators that are connected to the GPIO, serial port, I2C, and SPI interfaces
- Record video and take photographs using the Camera Board
- Issue Linux shell commands
- Transfer files to or from your host computer
- Control the on-board LED

To install or update this support package, perform the steps described in Install Support for Raspberry Pi Hardware.

For more information, see Raspberry Pi Hardware.

`readtable` improvements for reading spreadsheet and text files

- The readtable function now automatically recognizes `.xlsb`, `.xlsm`, `.xltm`, `.xltx`, and `.ods` as file extensions for spreadsheet files. You no longer need to specify the `'FileType','spreadsheet'` name-value pair argument when reading files with these extensions.
- The `readtable` function can now read from text files without a file extension. Previously, `readtable` searched for a file with a `.txt` extension.
- On systems with Excel® for Windows, the `readtable` function can read spreadsheet files in basic mode, using the `'Basic'` name-value pair argument. Basic mode is the default for systems without Excel for Windows.

Compatibility Considerations

If you specify a file name with no extension, for example, `foo`, the `readtable` function reads `foo` as a text file, if it exists. If `foo` does not exist, then `readtable` searches for and reads from `foo.txt`. In R2013b, `readtable` reads only from the file named `foo.txt`.

To read from a file with a `.txt` extension when an identically named file without an extension also exists, specify both the file name and extension in the call to `readtable`.

Functionality being removed or changed

Functionality	What Happens When You Use This Functionality?	Use This Instead	Compatibility Considerations
`aufinfo`	Warns	audioinfo	Replace all instances of `aufinfo` with `audioinfo`.
`auread`	Warns	audioread	Replace all instances of `auread` with `audioread`.
`auwrite`	Warns		Write audio files using audiowrite.
`wavfinfo`	Warns	audioinfo	Replace all instances of `wavfinfo` with `audioinfo`.
`wavread`	Warns	audioread	Replace all instances of `wavread` with `audioread`.
`wavwrite`	Warns	audiowrite	Replace all instances of `wavwrite` with `audiowrite`.
`wavplay`	Errors	audioplayer and play	Replace all existing instances of `wavplay` with `audioplayer` and `play`.
`wavrecord`	Errors	audiorecorder and record	Replace all existing instances of `wavrecord` with `audiorecorder` and `record`.
`mmreader`	Errors	VideoReader	Replace all instances of `mmreader` with `VideoReader`.

GUI Building

Panel Display in GUIDE Layout Area

While designing a GUI with GUIDE, if you place a panel on top of controls, the layout remains as you specify in the layout area. However, in the running GUI, the panel displays under the controls.

Compatibility Considerations

Previously, if you placed a panel on top of controls in the layout area, the panel automatically moved under the controls in the design area and appeared under the controls in the running GUI. Now, the panel remains as you place it in the layout area, but displays under the controls in the running GUI. The best practice is to place the panel in the layout area first, and then place the controls in the panel. This way, the design area matches the appearance of the running GUI. For existing GUIs, you can right-click the panel in the design area, and then select **Send to Back**.

Functionality being removed or changed

Functionality	What Happens When You Use This Functionality?	Use This Instead	Compatibility Considerations
HitTest, Selected, and SelectionHighlight properties for these components: • figure • uicontrol • uipanel • uibuttongroup • uipushtool • uitable • uitoggletool • uitoolbar	Still runs	Not applicable	Use of the HitTest, Selected, and SelectionHighlight properties is not recommended for the listed components. These properties might be removed from these components in a future release.

Performance

`conv2` function performance improvements with three inputs

The performance of the conv2 function improves when using the `conv2(h1,h2,A)` syntax.

`filter` function performance improvements for FIR and IIR

The performance of the filter function, `filter(b,a,X)`, improves in the following cases:

- where `a` is a scalar and `X` is a sufficiently long vector (FIR filter)
- where `a` is a vector and `X` is a vector, matrix, or multidimensional array (IIR filter)

R2013b

Language and Programming

`table` data container for managing, sorting, and filtering mixed-type tabular data

Tables and Categorical Arrays (6 min, 9 sec)

`table` is a new data type to collect mixed-type data and metadata properties, such as variable names, row names, descriptions, and variable units, in a single container. Tables are suitable for column-oriented or tabular data that is often stored as columns in a text file or in a spreadsheet. For example, you can use a table to store experimental data, with rows representing different observations and columns representing different measured variables.

Tables consist of rows and column-oriented variables. Each variable (column) in a table can have a different data type and a different size with the restriction that each variable must have the same number of rows. For example,

```
T =

                Gender    Age    Smoker     BloodPressure
                ------    ---    ------    ---------------
    Smith       'M'       38     true      124          93
    Johnson     'M'       43     false     109          77
    Williams    'F'       38     false     125          83
    Jones       'F'       40     false     117          75
    Brown       'F'       49     false     122          80
```

For more information, see Tables.

`categorical` array for ordered and unordered categorical data

Tables and Categorical Arrays (6 min, 9 sec)

`categorical` is a data type to store data with values from a finite set of discrete categories. A categorical array provides efficient storage and convenient manipulation of nonnumeric data, while also maintaining meaningful names for the values. Ordinal categorical arrays are a type of categorical array whose categories have a mathematical order. For example,

```
myCategorical =
```

```
     medium      large      small      small      medium

categories(myCategorical)

ans =

    'small'
    'medium'
    'large'
```

You can use categorical arrays in a table to select groups of rows. For more information, see Categorical Arrays.

`timeit` function for robust time estimates of function execution

The timeit function measures the time required to run a function. It provides a more robustly computed time estimate than `tic/toc`.

`localfunctions` function for getting handles to all local functions in a file

The localfunctions function returns a cell array of function handles to all local functions in the current file.

Functions for writing, executing, and verifying tests using the `matlab.unittest` testing framework without creating custom classes

As an alternative to writing object-oriented tests, the MATLAB xUnit-style testing framework now provides function-based writing, execution, and verification of tests. For more information, see Unit Testing Framework. For an example of function–based test writing, see Write Simple Test Case Using Functions.

`matlab.mixin.CustomDisplay` utility class to write custom display methods

Use the matlab.mixin.CustomDisplay class to customize object display for your MATLAB class.

`flip` function, a faster and more memory efficient alternative to `flipdim` for flipping arrays and vectors

The flip function provides a faster and more memory efficient alternative to `flipdim` for flipping arrays and vectors.

Partial name matching in `inputParser`

The `inputParser` now includes the `PartialMatching` property, which allows partial matching of parameter names. This property is `true` by default. For more information, see the inputParser reference page.

Compatibility Considerations

Parsing schemes requiring exact parameter name matching should set the `inputParser PartialMatching` property to `false`.

Additional `validateattributes` options for checking array values

The `validateattributes` function now checks the following additional attributes of numeric or logical input arrays.

Attribute	Description
`'decreasing'`	Each column element is less than the previous element and no element is `NaN`.
`'increasing'`	Each column element is greater than the previous element and no element is `NaN`.
`'nondecreasing'`	Each column element is greater than or equal to the previous element and no element is `NaN`.
`'nonincreasing'`	Each column element is less than or equal to the previous element and no element is `NaN`.

For more information, see the validateattributes reference page.

Conversion changes of out-of-range numbers passed to Java methods that take integers

MATLAB R2013b changes the way it converts out-of-range values for the following Java integer types:

- Signed 32-bit integer (Java `int`, `short` or `byte` parameters)
- Signed 64-bit integer (Java `long` parameters)

For a description of the conversion, see Converting Numbers to Integer Arguments.

Compatibility Considerations

If your MATLAB code can pass out-of-range values to Java methods with integer type arguments, the results might change. For example, the following value, `val`, is out-of-range for the argument to `java.lang.Integer`.

```
val = uint32(2^31);
java.lang.Integer(uint32(val))
```

In previous versions of MATLAB, the result is:

```
ans =

-2147483648
```

As of MATLAB R2013b, the result is:

```
ans =

-1
```

Additional properties for `mex.getCompilerConfigurations` function

The mex.getCompilerConfigurations function returns the following new properties:

- `ShortName` — Character string used to identify options file for the compiler
- `MexOpt` — Name and full path to options file
- `Priority` — The priority of this compiler

Changes to compiler support for building MEX-files

MATLAB no longer supports the following compilers on Microsoft Windows platforms:

- Intel® C++ Version 11
- Intel Visual Fortran Version 11
- Open Watcom C/C++

Compatibility Considerations

To ensure continued support for building your MEX-files, consider upgrading to another supported compiler. For an up-to-date list of supported compilers, see the Supported and Compatible Compilers website.

Changes to time alignment for time series objects

The `IsTimeFirst` property for time series objects is now read only. The property value is `false` for 3-D and higher dimensional data. Otherwise, the value is `true`.

- `true` — The first dimension of the data array is aligned with the time vector. For example, `ts = timeseries(rand(3,3),1:3);`
- `false` — The last dimension of the data array is aligned with the time vector. For example: `ts = timeseries(rand(3,4,5),1:5);`

Consequently, the alignment of the `Time` property with the first or last data dimension of the `Data` property is based on the number of dimensions of the data. When the data contains three or more dimensions, the length of the `Time` property matches the size of the last data dimension. Otherwise, the length of the `Time` property matches the size of the first data dimension.

Compatibility Considerations

You receive an error when creating a time series object with 3-D or higher dimensional data and with time alignment along the first dimension.

To preserve the property value of `true` for `IsTimeFirst` with 3-D or higher dimensional data, reshape the data array to two dimensions, such that the last data dimension and the time dimension are compatible. For example, replace `timeseries(ones(10,4,2),1:10)` with `timeseries(ones(10,8),1:10)`.

To preserve the number of dimensions of a 3-D or higher dimensional data array, use permute to reorder the data array, such that the size of the last data dimension aligns with the time vector.

Furthermore, you receive a warning when loading a time series object from a prior release with 3-D or higher dimensional data and with time alignment along the first dimension. In this case, MATLAB preserves the time alignment with the first dimension and reshapes the data to 2-D.

New fixture and plugin features for `matlab.unittest` testing framework

The `matlab.unittest` testing framework now provides four customized fixtures to ease the creation of setup and teardown code. You can use these fixtures to change the current working folder, add a folder to the MATLAB path, suppress the display of warnings, and create a temporary folder. For more information, see matlab.unittest.fixtures.

To share these fixtures across test classes, use the new `SharedTestFixtures` class attribute of TestCase. The getSharedTestFixtures method of TestCase provides access to the shared fixtures. You also can use fixtures within a test function by calling the applyFixture method of TestCase.

To pause execution of a test and enter debug mode upon a failure or uncaught error, you can add the new plugin, StopOnFailuresPlugin to the test runner. For more information, see matlab.unittest.plugins.

Conversion of error and warning message identifiers

For R2013b, error and warning message identifiers have changed in MATLAB.

Compatibility Considerations

If you have scripts or functions that use message identifiers that changed, you must update the code to use the new identifiers. Typically, message identifiers are used to turn off specific warning messages, or in code that uses a try/catch statement and performs an action based on a specific error identifier.

For example, the `MATLAB:InputParser:MustBeChar` identifier has changed to display either `MATLAB:InputParser:MustBeChar` or `MATLAB:InputParser:ParamMustBeChar`. If your code checks for `MATLAB:InputParser:MustBeChar`, you might need to update it to check for

`MATLAB:InputParser:ParamMustBeChar` instead. For a mapping of the new identifiers to the original identifiers, see Technical Support solution `1-ERAFNC`.

Functionality being removed or changed

Functionality	What Happens When You Use This Functionality?	Use This Instead	Compatibility Considerations
`addParamValue` method of inputParser class	Still Runs	`addParameter`	Replace all instances of `addParamValue` with `addParameter`.
`tstool`	Warns	timeseries, tscollection, openvar, or plot	Replace all instances of `tstool` with timeseries, tscollection, openvar, or plot. To create a time series object, use timeseries. To create a time series collection with one or more `timeseries` objects, use tscollection. To open a time series object or collection in the Variables editor, use openvar. To plot a time series object, use plot.
Time series object with 3-D or higher dimensional data and time aligned	Errors	Time series object with 2-D data and time aligned with the first dimension or a time series	For more information about updating your code and assessing the impact of

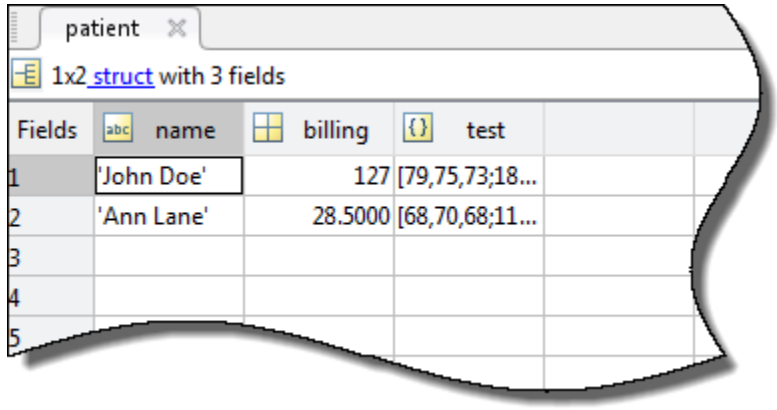
Functionality	What Happens When You Use This Functionality?	Use This Instead	Compatibility Considerations
with the first dimension		object with 3-D or higher dimensional data and time aligned with the last dimension	this change, see "Changes to time alignment for time series objects" on page 3-6.
`cat(0,A,B)`	Errors	`cat(1,A,B)`	In previous releases, the cat function silently changed `dim=0` to `dim=1`. Now, `cat` returns an error when `dim=0`. Replace all instances of `cat(0,A,B)` to `cat(1,A,B)`.
`cat(z,A,B)`, where `z` is a complex number	Errors	`cat(real(z),A,B)`	In previous releases, the `cat` function silently used the real part of `dim`. Now, cat returns an error when `dim` is complex. Replace all instances of `cat(z,A,B)` to `cat(real(z),A,B)`.
`mexIsGlobal` in the C/C++ and Fortran MEX API Library	Still Runs	mxIsFromGlobalWS in the C/C++ Matrix Library and Fortran Matrix Library	Replace all instances of `mexIsGlobal` with mxIsFromGlobalWS in MEX-files.

Desktop

Improved viewing and editing of one-dimensional structure arrays in the Variables editor

One-dimensional (n-by-1 or 1-by-n) structure arrays in the Variables editor now display field contents arranged in columns, in a single pane. You can reorder fields by dragging a column.

The new display allows you to work with one-dimensional structure arrays as you would with other workspace variables. For example, you can edit field values in-place, create new variables from a selection, and plot selected data using the options on the **Plots** tab.



For more information about editing structure arrays using the Variables editor, see Edit Table and Structure Array Data Interactively

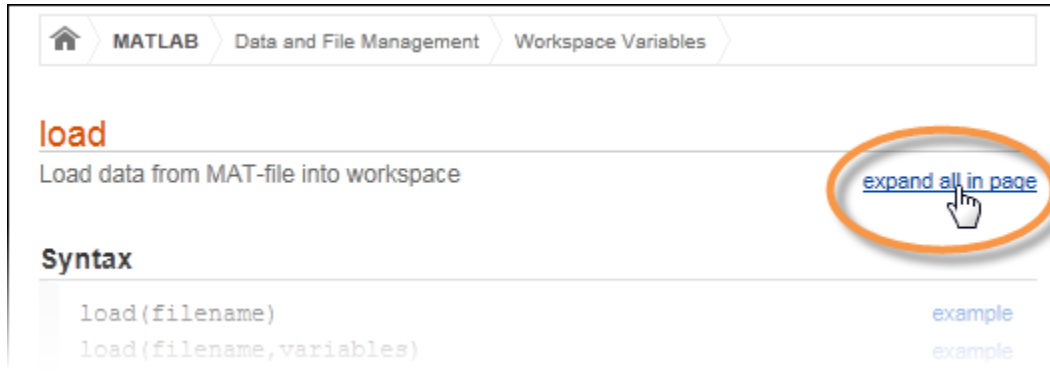
Improved management of a large number of open files, figures, and documentation pages

Files, docked figures, and documents each display in separate tabs that you can rearrange. For example, each file displays on a tab in the MATLAB Editor.

A drop-down list provides quick access to tabs that do not fit in the window when a large number of files, figures, or documents are open. When viewing multiple documents in a tiled layout, you can drag tabs to create new tiles or move documents between existing tiles. The tabs replace the document bar that was shared across tiles in earlier versions of MATLAB.

Expand all option for opening collapsed sections in documentation pages for printing and in-page searching

Some documentation pages include sections that are collapsed by default, such as input argument descriptions or examples. In-page searches do not find terms in those sections unless you first expand the sections. In R2013b, you can easily expand all sections on a page by clicking **expand all in page**, which is located at the top right of pages with collapsed sections. Then, you can search for terms anywhere on the page, or print the page in its entirety.



Java integration updated to version 7, providing access to new Java features and bug fixes

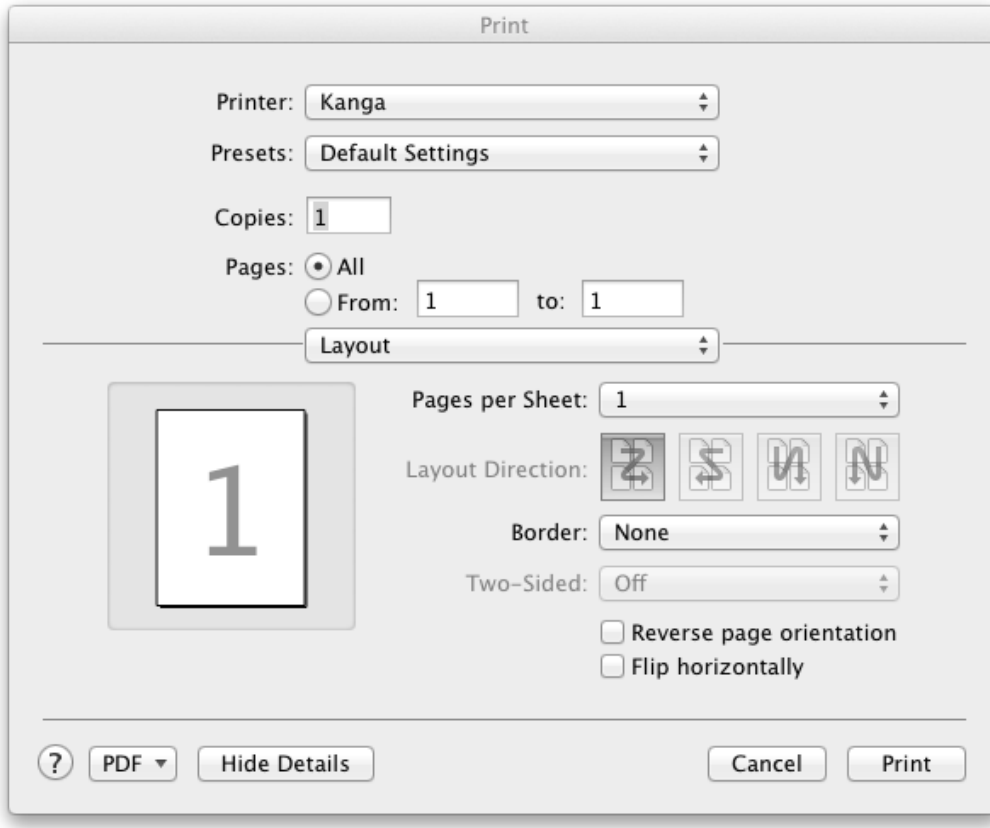
As of R2013b, support for the Oracle Java Runtime Environment (JRE™) has been updated to Java 7 Update 11 on all platforms.

Bundling of Java on Mac, removing dependency on Apple supplied Java runtime

The Mac version of MATLAB is no longer dependent on the Apple-provided JRE.

Enhanced print options on Mac operating systems

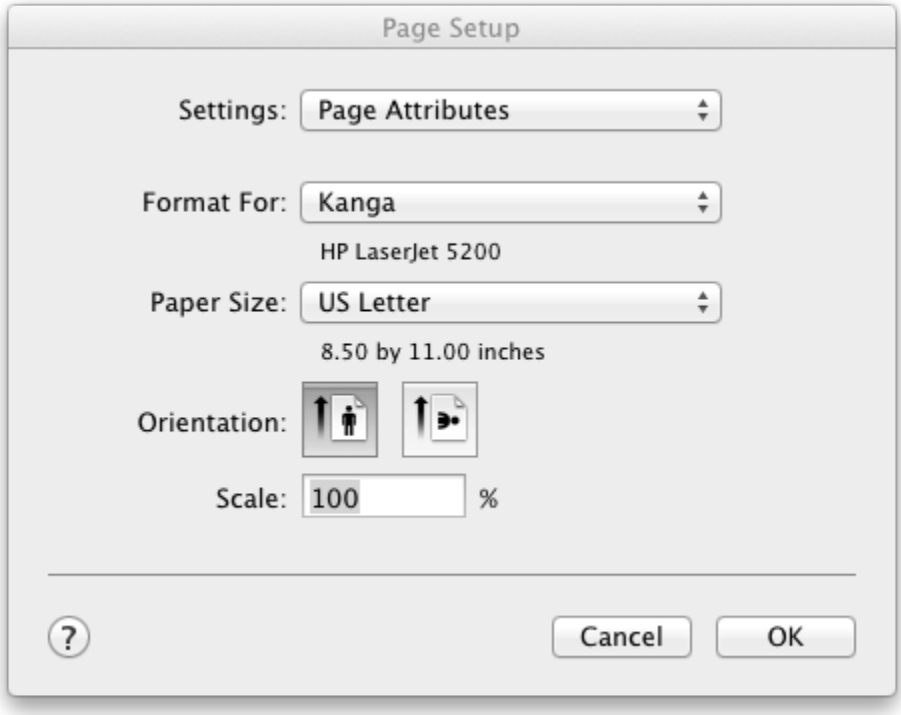
On Macintosh systems, the Print dialog box now provides more print options. You can access all print options provided by the native Macintosh print dialog. For example, you can select the number of pages to print per sheet, add a border or watermark, and print to a PDF file. To access the Print options, press **Command+P** from the Command Window or the MATLAB Editor.



Compatibility Considerations

Page Setup options, such as paper size and orientation, are no longer accessible from the Print dialog box. To access Page Setup options, select the **Editor** tab. In the **File** section,

click **Print** ▾ and then select **Page Setup**. Alternatively, from the Command Window or Editor, use the keyboard shortcut, **Command+Shift+P**. The Page Setup dialog box opens. In the **Settings** menu, ensure that **Page Attributes** is selected.



To access the options previously available from the Page Setup dialog box (such as Layout, Header, and Fonts options), select **MATLAB** in the **Settings** menu.

Option for following documentation links to uninstalled products

By default, the Help browser displays only the documentation associated with your installed products. In previous releases, if you used this default and clicked a link to documentation in an uninstalled product, the Help system displayed a “Page Not Found” error. Now, the Help system asks if you want to view the page from the MathWorks Web site. If so, the page opens in the Help browser.

For information on changing the default settings, see Help Preferences.

Preferences dialog box improvements for easier navigation

Options in the left pane of the Preferences dialog box are arranged by product, and alphabetized within each product for simpler navigation. This change does not affect code that calls preferences. To access the Preferences dialog box, click **Preferences** on the **Home** tab.

Auto-adjust capability in Variables editor

You now can auto-adjust the column width for elements of numeric, cell, structure, and table arrays in the Variables editor. Point to the border to the right of a column heading, until a double-headed arrow appears. Then, double-click to auto-adjust the width of that column.

MATLAB support added to Windows 7 Default Programs control panel

You can manage MATLAB file associations on Microsoft Windows 7 systems using the **Set your default programs** option in the **Default Programs** control panel.

MATLAB no longer supports selective installation of individual file associations.

For more information, see Associate Files with MATLAB on Windows Platforms.

Japanese localization available on Mac platforms

If your Mac Language setting is Japanese, then MATLAB, Simulink®, and other localized MathWorks products now have a Japanese user interface and documentation.

Mathematics

Functionality being removed or changed

Functionality	What Happens When You Use This Functionality	Use This Instead	Compatibility Considerations
`interp1(X, Y, Xq, [], ...)` where `[]` specifies the default interpolation method	Still Runs	`interp1(X, Y, Xq, 'linear', ...)`	Replace all instances of `interp1(X, Y, Xq, [], ...)` with `interp1(X, Y, Xq, 'linear', ...)`.

Passing mixed-orientation vectors to `interp2`:
`Vq = interp2(x, y, V, xq, yq)`
Specifically, if one or both of the following are true:
• One of `x` and `y` is a row vector and the other is a column vector.
• One of `xq` and `yq` is a row vector and the other is a column vector.
	Still Runs	Construct the full grid with `meshgrid` first. Alternatively, use `griddedInterpolant` if you have a large data set.	Modify all instances that pass mixed-orientation vectors to `interp2`. You can modify your code in one of two ways:
			• Call `meshgrid` to construct the full grid first.
			`[X,Y] = meshgrid(x, y);`
			`[Xq,Yq] = meshgrid(xq, yq);`
			`Vq = interp2(X, Y, V, Xq, Yq);`
			• Pass the vectors to `griddedInterpolant` inside a cell array.
			`F = griddedInterpolant({x, y}, V.');`
			`Vq = (F({xq, yq})).'`

Functionality	What Happens When You Use This Functionality	Use This Instead	Compatibility Considerations

Passing mixed-orientation vectors to `interp3`:
 	
`interp3(x, y, z, V, xq, yq, zq)`
Specifically, if one or both of the following are true:
• `x`, `y`, and `z` are a combination of row and column vectors.
• `xq`, `yq`, and `zq` are a combination of row and column vectors.
	Still Runs	Construct the full grid with `meshgrid` first. Alternatively, use `griddedInterpolant` if you have a large data set.	Modify all instances that pass mixed-orientation vectors to `interp3`. You can modify your code in one of two ways:
			• Call `meshgrid` to construct the full grid first.
			`[X,Y,Z] = meshgrid(x, y, z);`
			`[Xq,Yq,Zq] = meshgrid(xq, yq, zq);`
			`Vq = interp3(X, Y, Z, V, Xq, Yq, Zq);`
			• Pass the vectors to `griddedInterpolant` inside a cell array.
			`V = permute(V, [2 1 3]);`
			`F = griddedInterpolant({x, y, z}, V);`
			`Vq = F({xq, yq, zq});`
			`Vq = permute(Vq, [2 1 3]);`

Functionality	What Happens When You Use This Functionality	Use This Instead	Compatibility Considerations

Passing mixed-orientation vectors to `interpn`:
 	
`interpn(x1,x2,...,xV, x1q,x2q,...,xnq)`
 	
Specifically, if one or both of the following are true:
 	
• `x1,x2,...,xn` are a combination of row and column vectors.
• `x1q,x2q,...,xnq` are a combination of row and column vectors.
	Still Runs	Construct the full grid with `ndgrid` first. Alternatively, use `griddedInterpolant` if you have a large data set.	Modify all instances that pass mixed-orientation vectors to `interpn`. You can modify your code in one of two ways:
			
			• Call `ndgrid` to construct the full grid first.
			`[X1,X2,...,Xn] = ndgrid(x1, x2, ..., xn);`
			`[X1q,X2q,...,Xnq] = ndgrid(x1q, x2q, ..., xnq);`
			`Vq = interpn(X1, X2, ..., Xn, V, X1q, X2q, ..., Xnq);`
			• Pass the vectors to `griddedInterpolant` inside a cell array.
			`F= griddedInterpolant({x1, x2, ..., xn}, V);`
			`Vq= F({x1q, x2q, ..., xnq});`

`interp1(..., 'cubic')`	Warns	`interp1(..., 'pchip')`	Replace all instances of `interp1(..., 'cubic')` with `interp1(..., 'pchip')`

Functionality	What Happens When You Use This Functionality	Use This Instead	Compatibility Considerations

Passing the `'pp'` flag to `interp1`. For example:
`pp = interp1(x, v, 'linear', 'pp');`
`vq = ppval(pp, xq);`
	Warns	Use griddedInterpolant to create an interpolating function that is efficient to evaluate in a repeated manner.	Replace all instances of `interp1(..., 'pp')` with `griddedInterpolant`. For example:
			`F = griddedInterpolant(x, v, 'linear');`
			`vq = F(xq);`

`bitshift(A,k,N)`	Warns	`bitshift(A, k, assumedtype)`	Replace all instances of `bitshift(A,k,N)` with `bitshift(A,k,assumedtype)`.
`bitcmp(A,N)`	Warns	`bitcmp(A, assumedtype)`	Replace all instances of `bitcmp(A,N)` with `bitcmp(A,assumedtype)`.
`repmat(A,r1,r2)`, where `r1` and `r2` are row vectors	Warns	`repmat(A,[r1 r2])`	Replace all instances of `repmat(A,r1,r2)` with `repmat(A,[r1 r2])`.
`repmat(A,empt)`, where `empt` is an empty array	Warns	`repmat(A,1)`	Replace all instances of `repmat(A,empt)` with `repmat(A,1)`.
`repmat(A,empt1,empt` where `empt1` and `empt2` are empty arrays	Warns	`repmat(A,1)`	Replace all instances of `repmat(A,empt1,empt2)` with `repmat(A,1)`.
`repmat(A,n,empt)`, where `empt` is an empty array	Warns	`repmat(A,[n 1])`	Replace all instances of `repmat(A,n,empt)` with `repmat(A,[n 1])`.

Functionality	What Happens When You Use This Functionality	Use This Instead	Compatibility Considerations
`repmat(A,empt,n)`, where `empt` is an empty array	Errors	`repmat(A,[n 1])`	Replace all instances of `repmat(A,empt,n)` with `repmat(A,[n 1])`.
`repmat(A,col)`, where `col` is a column vector	Errors	`repmat(A,col.')`	Replace all instances of `repmat(A,col)` with `repmat(A,col.')`.
`repmat(A,B)`, where `B` is a matrix	Errors	`bvec = B(1:length(B)); repmat(A,bvec)`	Replace all instances of `repmat(A,B)` with the following code: `bvec = B(1:length(B)); repmat(A,bvec)`.
`repmat(A,B,C)`, where `B` and `C` are matrices	Errors	`BC = [B C]; bcvec = BC(1:length(BC)); repmat(A,bcvec)`	Replace all instances of `repmat(A,B,C)` with the following code: `BC = [B C]; bcvec = BC(1:length(BC)); repmat(A,bcvec)`.
Noninteger-valued size inputs for zeros, ones, eye, Inf, NaN, true, false, rand, randi, randn, and cell	Errors	Integer valued size inputs	Replace all instances of noninteger-valued size inputs with integer-valued size inputs for `zeros`, `ones`, `eye`, `Inf`, `NaN`, `true`, `false`, `rand`, `randi`, `randn`, and `cell`. You can use floor for this conversion.
`mimofr`	Errors	Not Applicable	Remove all instances of `mimofr` from your existing code.

Graphics

Mac support for copying figures in vector formats to other applications

MATLAB for Mac now supports copying figures to other applications in a high-resolution PDF format. If the figure contains uicontrols, then MATLAB uses a TIFF format instead.

savefig for saving figures to FIG-files

savefig saves one or more figures to a FIG-file.

OpenGL workarounds

If MATLAB crashes because of problems with your current version of OpenGL, then use one of the following workarounds:

- Start MATLAB with the `-softwareopengl` startup flag to use the software version of OpenGL.
- Execute the `opengl neverselect` command in your `startup.m` file to prevent MATLAB from selecting OpenGL as the renderer.

Functionality being removed or changed

Functionality	What Happens When You Use This Functionality?	Use This Instead	Compatibility Considerations
Numeric specifiers to set legend location	Still runs	Supported string specifiers	Remove all instances of using numeric specifiers to set the legend location
`get(0,'CommandWindowSize')`	Still Runs	`matlab.deskt`	Replace all instances of `get(0,'CommandWindowSize')` with `matlab.desktop.commandwindow.si` The new command is compatible with MATLAB R2013a and later releases.

Functionality	What Happens When You Use This Functionality?	Use This Instead	Compatibility Considerations
`printdlg('-setup',fig)`	Warns	Use the operating system printer management utilities to set up a new printer	Remove all instances of using `printdlg` with the `-setup` option

GUI Building

Custom icons for MATLAB apps you create

You can now package a custom icon in an app that you create. When selecting an icon, click **Use your own icon...** .

A second dialog box opens. Click **Select icon** to browse for a custom icon. The dialog box also displays three icon sizes. MATLAB automatically scales your icon for use in the Install dialog, App gallery, and quick access toolbar. For more information, see Package Apps.

Performance

`repmat` with numeric, char, and logical types

The performance of the repmat function improves when the input array contains numeric, char, or logical values.

Linear algebra functions on computers with new AMD processors

The performance of the linear algebra functions improves on computers with AMD® processors supporting the Intel AVX instruction set.

Data Import and Export

`fprintf` function prints Unicode characters to the screen

The fprintf function now displays Unicode® characters on the screen. Previously, text was displayed with the user default character encoding, which is dependent on the user locale.

Compatibility Considerations

When displaying non-ASCII text on the screen, the `fprintf` function might return a different output value for the number of displayed bytes, compared to previous versions of MATLAB. This output value is now equal to the number of characters displayed on the screen.

There are no changes to the behavior of `fprintf` when writing text to a file, or when printing ASCII text to the screen.

Changes to default encoding for `sendmail` function

When calling the sendmail function on systems using a Japanese locale, the default character encoding is now UTF-8. Previously, the default encoding was SJIS.

Compatibility Considerations

This change affects only systems that use a Japanese locale. To use the SJIS character encoding, call:

```
setpref('Internet','E_mail_Charset','SJIS');
```

Functionality being removed or changed

Functionality	What Happens When You Use This Functionality?	Use This Instead	Compatibility Considerations
`hdftool`	Warns		Read data from HDF files using hdfread or the low-

Functionality	What Happens When You Use This Functionality?	Use This Instead	Compatibility Considerations
			level functions in the `matlab.io.hdf4.sd`, `matlab.io.hdfeos.gd`, and `matlab.io.hdfeos.sw` packages.

R2013a

Desktop

Option to add separators between controls on the quick access toolbar

You can now insert and move separators bars on the quick access toolbar. This helps to organize icons into groups. For information on how to customize and organize icons on the quick access toolbar, see Access Frequently Used Features.

Additional icon choices, auto-scaled thumbnails, and text-formatting options for customizing descriptions of MATLAB apps

When you package an app by clicking the **Package App** tab on the **Apps** tab, MATLAB opens the Package app dialog box. In the **Describe your app** section of the dialog box, you can:

- Click the app icon  , and then select a new icon from the display to represent your app in the Apps gallery.

- Use formatting options when writing the app description.

  Options include: bold, italic, monospace, and hyperlinked text; bulleted and numbered lists; images.

In addition, MATLAB automatically scales the screenshot you provide when packaging. This screenshot appears in the tooltip MATLAB displays when hovering over an app icon in the App gallery. The screenshot is scaled to fit a 300 x 300 area and maintains its original aspect ratio.

For more information, see Package Apps.

Left-aligned table of contents for navigating in the Help browser and online Documentation Center

The documentation now includes a table of contents that can remain open and that does not overlap the content. View the table of contents by clicking the Contents icon on the left edge of the window.

Search term highlighting and content expansion in the Help browser

In the Help browser, search terms now appear highlighted when you view pages linked from the search results. If the search term occurs within a collapsed section on the page, the browser automatically expands the content.

For example, the reference page for the `publish` function includes the term *bmp* in collapsed content. When you search the documentation for *bmp*, and then select the `publish` reference page, the Help browser expands the relevant section on the page and highlights all occurrences of the term.

To clear highlights, press the **Esc** key.

Context menu items in Help and Web browsers for zooming, page navigation, and saving

Both the Help browser and the MATLAB Web browser now provide easy access to commonly used features in their context menus. Right-click within the browser to see the available options, which include zooming to change the font size, searching within a page, navigating back or forward, or saving the HTML to a file.

Note: These context menu items are not available on Macintosh systems.

Removal of Handle Graphics support under `-nojvm` startup option

When you start MATLAB with the `-nojvm` startup option, Handle Graphics® functionality is not supported. (The `-nojvm` option is available for UNIX® platforms; see background information on the matlab (UNIX) reference page.)

Some aspects of MATLAB and related products use Handle Graphics in a way that might not be obvious. This includes anything that is based on or works using figures in MATLAB. Here is a summary of the affected features:

- Creating figures and performing plotting tasks, such as using the `plot`, `axes`, `getframe`, and `gcf` functions.
- Printing figures and using related functions such as `print`, `hgexport`, and `saveas`.
- Creating GUIs in MATLAB using GUI-building functions such as `warndlg`.
- Using Simulink scopes and printing Simulink models.

If you use the `-nojvm` startup option and use Handle Graphics functionality, MATLAB produces an error.

Compatibility Considerations

To avoid the error, start MATLAB without the `-nojvm` startup option:

- If you have been using the `-nojvm` startup option to work in a command line environment or because you do not use the MATLAB desktop, use the `-nodesktop` startup option instead.
- If you have been using the `-nojvm` startup option because of memory or performance benefits, look for other ways to gain those improvements when you start MATLAB without the `-nojvm` option. See topics in Performance and Memory.
- If you want to continue to use the `-nojvm` startup option, remove the code that is now producing the warnings.

`-noFigureWindows` startup option suppresses figures on Linux and Mac platforms

When you start MATLAB with the `-noFigureWindows` startup option, MATLAB disables the display of figure windows.

No default keyboard shortcut for overwrite mode

There is no longer a default keyboard shortcut for Overwrite mode, and **OVR** no longer appears at the bottom right corner of the MATLAB status bar. Previously, the **Insert** key by default toggled between Insert and Overwrite modes in the Command Window and in the MATLAB Editor.

Compatibility Considerations

To enable Overwrite mode, assign a keyboard shortcut to this action. On the **Home** tab, in the **Environment** section, click **Preferences > Keyboard > Shortcuts**. Search or scroll to find the action name, **Toggle Insert/Overwrite Mode**, and assign a keyboard shortcut. Use this shortcut to toggle between Insert and Overwrite modes.

Support requests using prerelease versions

If you have Internet access, you can submit a technical support request to MathWorks from MATLAB by selecting **Help > Request Support**. Previously, this feature was only available in the final version of each release. Beginning with R2013a, this feature is also available in the prerelease version.

Language and Programming

`matlab.unittest` package, an xUnit-style testing framework for the MATLAB language that allows writing and running unit tests, and analyzing test results

New MATLAB Unit Testing Framework (9 min, 24 sec)

For information about the `matlab.unittest` package, see Unit Testing Framework.

`strsplit` and `strjoin` functions for splitting and joining strings

The strsplit function splits a string into a cell array of strings. The strjoin function joins strings in a cell array into a single string.

Additional `validateattributes` options for checking array size and shape

The `validateattributes` function now checks the following attributes of input arrays.

Attribute	Description
`'3d'`	Array with three or fewer dimensions
`'ndims', N`	N-dimensional array
`'square'`	Square matrix
`'diag'`	Diagonal matrix

For more information, see the validateattributes reference page.

Help text for enumerations and events

The `help` and `doc` commands now display help text that you define for enumerations and events in custom classes. For more information, see Create Help for Classes.

Removal of support for `.jar` documentation files

The Help system no longer extracts documentation from `.jar` files. All custom documentation must be in uncompressed HTML files.

For more information, see Display Custom Documentation.

Changes to Microsoft .NET Framework support

As of R2013a, the MATLAB .NET interface requires the Microsoft .NET Framework Version 4.0 and above. The interface continues to support assemblies built on Framework 2.0 and above.

Compatibility Considerations

You must have the .NET Framework Version 4.0 or above installed on your system. If you enter any `NET.` or `System.` commands on a machine without Framework 4.0, MATLAB displays a warning. To determine if your system has a supported framework, use the NET.IsNETSupported function.

Changes to compiler support for building MEX-files

MATLAB supports these new compilers for building MEX-files.

Microsoft Windows platforms:

- Visual C++ 2012
- Intel C++ XE 2013
- Intel Visual Fortran XE 2013

Mac OS X platforms:

- Apple Xcode 4.2 and higher with Clang

MATLAB no longer supports the following compiler on Mac OS X platforms:

- Xcode with gcc

Compatibility Considerations

To ensure continued support for building your MEX-files, consider upgrading to another supported compiler. For an up-to-date list of supported compilers, see the Supported and Compatible Compilers website.

Changes to subclasses of built-in classes

Subclasses of built-in classes no longer inherit the following methods:

- `ones`
- `zeros`
- `true`
- `false`
- `cast`

As a result of this change, you cannot call these methods with an object of a built-in subclass. For example, given a class, `mySubclassOfDouble`, that derives from `double`, the following attempt to call `cast` results in an error:

```
a = mySubclassOfDouble(1);   % constructor
b = a.cast('int8');          % ok in R2012b but errors in R2013a
```

However, you can call the corresponding built-in functions with objects of the subclass. Use the function syntax instead of the method dot notation syntax:

```
b = cast(a,'int8');
```

Compatibility Considerations

Change code that attempts to call these methods to call the built-in function instead. Use function syntax instead of dot notation, as described in the previous section.

Conversion of error and warning message identifiers

For R2013a, error and warning message identifiers have changed in MATLAB.

Compatibility Considerations

If you have scripts or functions that use message identifiers that changed, you must update the code to use the new identifiers. Typically, message identifiers are used to turn off specific warning messages, or in code that uses a try/catch statement and performs an action based on a specific error identifier.

For example, the `MATLAB:addtodate:Nargin` identifier has changed to `MATLAB:addtodatemx:Nargin`. If your code checks for `MATLAB:addtodate:Nargin`, you must update it to check for `MATLAB:addtodatemx:Nargin` instead. For a mapping of the new identifiers to the original identifiers, see Technical Support solution `1-ERAFNC`.

No strict-match requirements for month formats when converting date strings

When reading date strings, the date functions datenum, datevec, and datestr now treat months expressed as full names, three-letter abbreviations, or numbers in date string inputs interchangeably. In previous releases, the commands fail if a date string input does not match the format input. Using the `datestr` function to write date strings is not affected.

For example, consider the cell array of date strings, where each date string indicates the month in a different format:

```
mydates = {'31-October-2012','31-Oct-2012','31-10-2012'};
datevec(mydates,'dd-mm-yyyy')
```

```
ans =

        2012          10          31           0           0           0
        2012          10          31           0           0           0
        2012          10          31           0           0           0
```

Replacing the date string format input, `'dd-mm-yyyy'`, with `'dd-mmm-yyyy'` or `'dd-mmmm-yyyy'` returns the same output. Previously, the command failed, because the first two date strings include a full month name and a month abbreviation, which are not strict matches to the `mm` month format specified.

Compatibility Considerations

The date functions `datenum`, `datevec`, and `datestr` do not error when month formats are inconsistent across multiple date strings in a cell array of strings. When reading a cell array of date strings, you no longer can rely on using a format that includes a field for month name or abbreviation to catch date strings that specify a month number. You should ensure that all date strings in a cell array have the same format to avoid the possibility of confusing day and month numbers. For example,

```
mydates = {'11-Oct-2012','11-10-2012'};
datevec(mydates,'dd-mmm-yyyy')

ans =

        2012          10          11           0           0           0
        2012          10          11           0           0           0
```

In R2013a, both dates are silently interpreted as October 11, 2012 although you might have intended `'11-10-2012'` to be in month-day-year format, representing November 10, 2012.

Date functions error on out-of-range quarter values

Valid values for quarter formats are Q1 through Q4 when converting date strings using the datenum or datevec functions. Values outside this range throw an error. Previously, the date functions accepted values outside the range, such as Q5, but returned incorrect results.

Compatibility Considerations

To prevent errors when parsing quarter values, remove values outside the valid range (Q1 through Q4) from your code.

String representations of large integers using exponential notation

The num2str function converts any large floating-point number that loses precision due to hardware limitations to a string representation that uses exponential notation. Such floating-point numbers are those greater than flintmax. Previously, `num2str` returned a string representation in decimal notation.

This change in behavior affects only the syntax `str = num2str(x)`. The following example compares the output of `num2str` in R2013a to R2012b.

R2013a	R2012b

`num2str(30e+25)`
 	
`ans =`
`3e+26`
	`num2str(30e+25)`
	
	`ans =`
	`300000000000000000000000000`

Compatibility Considerations

To obtain a string representation of large integers that uses decimal notation, call `int2str(x)` instead of `num2str(x)`.

Do not use `classpath.txt` file to modify Java static path

As of R2012b, the file `classpath.txt` is no longer used to add custom paths to the static Java class path. A new MathWorks product installation rewrites the static class path (found in the `classpath.txt` file).

Compatibility Considerations

To modify the Java class path, create `javaclasspath.txt` or `javalibrarypath.txt` files. For more information, see The Static Path and Locating Native Method Libraries.

Functionality being removed or changed

Functionality	What Happens When You Use This Functionality?	Use This Instead	Compatibility Considerations
`createCopy` method of `inputParser` class	Warns	`copy`	Replace all instances of `createCopy` with `copy`.
`hostid`	Warns		See Solution 1–171PI on the MathWorks website for instructions on obtaining your hostid.

Mathematics

`scatteredInterpolant` and `griddedInterpolant` support for extrapolation

- scatteredInterpolant is a new class for interpolating scattered data, and it returns extrapolated values by default when you evaluate at query points outside the convex hull.
- griddedInterpolant now returns extrapolated values by default when you evaluate at query points outside the domain of your sample grid.

Compatibility Considerations

`scatteredInterpolant` supports extrapolation by default. This behavior is different from that of `TriScatteredInterp`, which returns `NaN` when you evaluate at query points outside the convex hull using the `'linear'` or `'natural'` methods. To preserve the `TriScatteredInterp` behavior, set the `ExtrapolationMethod` property to `'none'`. See the `scatteredInterpolant` reference page for more information.

`griddedInterpolant` previously returned `NaN` values when you queried points outside the domain of the sample points using the `'linear'`, `'cubic'` or `'nearest'` interpolation methods. Now all interpolation methods support extrapolation by default. To preserve the old behavior, set the `ExtrapolationMethod` property to `'none'` when the interpolation method is `'linear'`, `'cubic'` or `'nearest'`. See the `griddedInterpolant` reference page for more information.

Syntax for `ones`, `zeros`, and other functions for creating arrays that match attributes of an existing variable

The functions ones, zeros, eye, Inf, NaN, true, false, and cast now can return an output that matches the data type, sparsity, and complexity (real or complex) of a variable `p`.

For example:

```
p = uint8([1 2]);
X = ones(2,3,'like',p);
class(X)
```

```
ans =

uint8
```

Integer type support for `prod`, `cumsum`, `cumprod`, `median`, `mode`, and number theory functions

The following functions now support inputs of any integer data type: median, mode, prod, cumprod, cumsum, isprime, primes, factor, gcd, lcm, perms, factorial, nextpow2, and nchoosek.

`flintmax` function for largest consecutive integer in floating-point format

Largest consecutive integer in floating-point format. flintmax returns the largest consecutive integer in IEEE® double precision, which is `2^53`. Above this value, double-precision format does not have integer precision, and not all integers can be represented exactly. `flintmax('single')` returns the largest consecutive integer in IEEE single precision, which is `single(2^24)`.

Scale option for `airy` function

`scale` is an optional flag you can pass to airy to scale the resulting Airy function. See the `airy` reference page for more information.

`scatteredInterpolant` class that replaces `TriScatteredInterp`

`TriScatteredInterp` will be removed in a future release. Use the new scatteredInterpolant class instead. The `scatteredInterpolant` class performs interpolation on 2-D and 3-D scattered data with support for extrapolation outside the convex hull of the sample points. `scatteredInterpolant` also supports queries in grid vector format to conserve memory.

Compatibility Considerations

- `scatteredInterpolant` supports extrapolation by default. This behavior is different from that of `TriScatteredInterp`, which returns `NaN` when you evaluate

at query points outside the convex hull using the `'linear'` or `'natural'` methods. To preserve the `TriScatteredInterp` behavior, set the `ExtrapolationMethod` property to `'none'`. See the `scatteredInterpolant` reference page for more information.

- `scatteredInterpolant` does not accept input of type `DelaunayTri`. Use nearestNeighbor or pointLocation to interpolate using a specific Delaunay triangulation. See Interpolation Using a Specific Delaunay Triangulation for more information.
- The `scatteredInterpolant` `Points` property is the array of sample point coordinates. The corresponding `TriScatteredInterp` property is called `X`. Use the `Points` property to refer to the sample points when you update your code.

`triangulation` class to replace `TriRep`

`TriRep` will be removed in a future release. Use the new triangulation class instead. The `triangulation` class represents 2-D and 3-D triangulations using a similar form and syntax as `TriRep`.

Compatibility Considerations

The property names and some method names have changed. The tables below map the current `TriRep` properties and methods to the `triangulation` replacements. Each replacement property has the same shape and holds the same values as the corresponding `TriRep` property. Similarly, each replacement method has the same syntax and returns the same result as the corresponding `TriRep` method.

Replacements for TriRep Properties

TriRep Property	Replacement triangulation Property	Name Change
`X`	`Points`	#
`Triangulation`	`ConnectivityList`	#

Replacements for TriRep Methods

TriRep Method	Replacement triangulation Method	Name Change
`baryToCart`	`barycentricToCartesian`	#

TriRep Method	Replacement triangulation Method	Name Change
cartToBary	cartesianToBarycentric	#
circumcenters	circumcenter	#
edgeAttachments	edgeAttachments	
edges	edges	
faceNormals	faceNormal	#
featureEdges	featureEdges	
freeBoundary	freeBoundary	
incenters	incenter	#
isEdge	isConnected	#
neighbors	neighbors	
size	size	
vertexAttachments	vertexAttachments	

`delaunayTriangulation` class to replace `DelaunayTri`

`DelaunayTri` will be removed in a future release. Use the new delaunayTriangulation class instead. `delaunayTriangulation` represents 2-D and 3-D Delaunay triangulations using a similar form and syntax as `DelaunayTri`.

Compatibility Considerations

Some of the property and method names have changed. The tables below map the current `DelaunayTri` properties and methods to the `delaunayTriangulation` replacements. Each replacement property has the same shape and holds the same values as the corresponding `DelaunayTri` property. Similarly, each replacement method has the same syntax and returns the same result as the corresponding `DelaunayTri` method.

Replacements for DelaunayTri Properties

DelaunayTri Property	Replacement delaunayTriangulation Property	Name Change
Constraints	Constraints	
X	Points	#

DelaunayTri Property	Replacement delaunayTriangulation Property	Name Change
`Triangulation`	`ConnectivityList`	#

Replacements for DelaunayTri Methods

DelaunayTri Method	Replacement delaunayTriangulation Method	Name Change
`convexHull`	`convexHull`	
`inOutStatus`	`isInterior`	#
`nearestNeighbor`	`nearestNeighbor`	
`pointLocation`	`pointLocation`	
`voronoiDiagram`	`voronoiDiagram`	

Set functions behavior change

The behavior of unique, union, intersect, setdiff, setxor, and ismember has changed.

- If there are repeated elements in the input arrays, the functions `unique`, `union`, `intersect`, `setdiff`, and `setxor` return the index to the first occurrence of the repeated elements (or rows).
- The optional output array, `locb`, returned by `ismember`, contains the lowest absolute indices in `B` for elements (or rows) that are also in `A`.
- All index vectors returned by `unique`, `union`, `intersect`, `setdiff`, or `setxor` are column vectors.
- The shape of the output, `C`, from `intersect`, `setxor`, and `union`, when the `'rows'` and `'legacy'` flags are not specified, is as follows. `C` is column vector unless both `A` and `B` are row vectors, in which case `C` is a row vector.
- The shape of the output, `C`, from `setdiff`, when the `'rows'` and `'legacy'` flags are not specified, is as follows. `C` is a row vector if `A` is a row vector. Otherwise, `C` is a column vector.
- `unique`, `union`, `intersect`, `setdiff`, `setxor`, and `ismember` support objects.
- The input arrays passed to `union`, `intersect`, `setdiff`, `setxor`, and `ismember` must be of the same class with the following exceptions:
  - Logical, char, and all numeric classes can combine with double arrays.

  - Cell arrays of strings can combine with char arrays.
- `ismember` treats trailing white space in cell arrays of strings as distinct characters. For example, `'word'` is different from `'word '`. If the `'legacy'` flag is specified, `ismember` ignores trailing white space and treats `'word'` the same as `'word '`.

Compatibility Considerations

If the changes adversely affect your code, you can specify `'legacy'` to preserve the behavior from R2012b and prior releases. For example:

```
[C,IA,IC] = unique([9 9 1])
C =

     1     9


IA =

     3
     1


IC =

     2
     2
     1
[C2,IA2,IC2] = unique([9 9 1],'legacy')
C2 =

     1     9


IA2 =

     3     2


IC2 =

     2     2     1
```

Functionality being removed or changed

Functionality	What Happens When You Use This Functionality	Use This Instead	Compatibility Considerations
`quad`	Still runs	integral	Replace all instances of `quad` with `integral`. For example, replace `q = quad('x.*cos(x)',0,1)` with `q = integral(@(x)x.*cos(x),0,1)`. If `quad` uses an absolute error tolerance, `tol`, replace all instances of the tolerance argument with the `'AbsTol'` or `'RelTol'` name-value pair arguments. For example, replace `q = quad(fun,a,b,tol)` with `q = integral(fun, a, b, 'AbsTol', tol, 'RelTol', tol)`.
`quadl`	Still runs	integral	Replace all instances of `quadl` with `integral`. For example, replace `q = quadl('x.*cos(x)',0,1)` with `q = integral(@(x)x.*cos(x,0,1)`. If `quadl` uses an absolute error tolerance, `tol`, replace all instances of the tolerance argument with the `'AbsTol'` or

Functionality	What Happens When You Use This Functionality	Use This Instead	Compatibility Considerations
			`'RelTol'` name-value pair arguments. For example, replace `q = quadl(fun,a,b,tol)` with `q = integral(fun, a, b, 'AbsTol', tol, 'RelTol', tol)`.
`quadv`	Still runs	integral with the `'ArrayValued',true` name-value pair argument	Replace all instances of `quadv` with `integral` using the `'ArrayValued',true` name-value pair argument.
`dblquad`	Still runs	integral2	Replace all instances of `dblquad` with `integral2`. If there are discontinuities in the interior of the region of integration, replace `dblquad` with `integral2` using the `'method','iterated'` name-value pair argument.
`triplequad`	Still runs	integral3	Replace all instances of `triplequad` with `integral3`. If there are discontinuities in the interior of the region of integration, replace `triplequad` with `integral3` using the `'method','iterated'` name-value pair argument.

Functionality	What Happens When You Use This Functionality	Use This Instead	Compatibility Considerations
`bitmax`	Still runs	flintmax	Replace all instances of `bitmax` with `flintmax`.
`TriRep`	Still runs	triangulation	Replace all instances of `TriRep` with `triangulation`. Most of the `triangulation` properties and methods are the same as those in `TriRep`, but there are a few exceptions. See "`triangulation` class to replace `TriRep`" on page 4-14 for details.
`DelaunayTri`	Still runs	delaunayTriangulation	Replace all instances of `DelaunayTri` with `delaunayTriangulation`. Most of the `delaunayTriangulation` properties and methods are the same as those in `DelaunayTri`, but there are a few exceptions. See "`delaunayTriangulation` class to replace `DelaunayTri`" on page 4-15 for details.

Functionality	What Happens When You Use This Functionality	Use This Instead	Compatibility Considerations
`TriScatteredInterp`	Still runs	scatteredInterpolant	Replace all instances of `TriScatteredInterp` with `scatteredInterpolant`. See "`scatteredInterpolant` class that replaces `TriScatteredInterp`" on page 4-13 for compatibility details.
`inline`	Still runs	Anonymous function	Replace all instances of `inline` with an anonymous function. See Anonymous Functions for more information. You can use symvar and func2str to convert your existing code.
rand or randn with the `'seed'`, `'state'` or `'twister'` inputs.	Still runs	rng	For more information about updating your code and assessing the impact of this change, see Updating Your Random Number Generator Syntax.
`permute(A,order)` where `order` contains noninteger or complex values.	Warns	Real, positive integer values for `order`	Replace all instances of noninteger or complex valued inputs with real, positive integer values. You can use round or real for this conversion.

Functionality	What Happens When You Use This Functionality	Use This Instead	Compatibility Considerations
Noninteger valued size inputs for zeros, ones, eye, Inf, NaN, true, false, rand, randi, randn, and cell	Warns	Integer valued size inputs	Replace all instances of noninteger valued size inputs with integer valued size inputs for `zeros`, `ones`, `eye`, `Inf`, `NaN`, `true`, `false`, `rand`, `randi`, `randn`, and `cell`. You can use floor for this conversion.
`cholinc(X,'inf')`	Errors	None	Remove all instances of `cholinc(X,'inf')` from your code.
All other syntaxes of `cholinc` except `cholinc(X,'inf')`	Errors	ichol	Replace these instances of `cholinc` with `ichol`.
`luinc`	Errors	ilu	Replace all instances of `luinc` with `ilu`.
`sparse(A)` `sparse(i,j,s,...)` where `A` and `s` are of type `char`.	Errors	`sparse(double(A))` `sparse(i,j,double(s),..`	Convert all `char` inputs to `double` before calling `sparse`.
`RandnAlg` property of RandStream class	Errors	`NormalTransform` property of `RandStream` class	Replace all existing instances of `RandnAlg` with `NormalTransform`.
`setDefaultStream` method of `RandStream` class	Errors	`setGlobalStream` method of `RandStream` class	Replace all existing instances of `RandStream.setDefaultStream` with RandStream.setGlobalStream.

Functionality	What Happens When You Use This Functionality	Use This Instead	Compatibility Considerations
`getDefaultStream` method of `RandStream` class	Errors	`getGlobalStream` method of `RandStream` class	Replace all existing instances of `RandStream.getDefaultStream` with RandStream.getGlobalStream.

Graphics

`gobjects` function for preallocating graphics handle array

The gobjects function creates an array of graphics handles. Use this function instead of the `ones` or `zeros` function when preallocating an array to store graphics handles.

Functionality being removed or changed

Functionality	What Happens When You Use This Functionality?	Use This Instead	Compatibility Considerations
Figure `DoubleBuffer` property	No effect	No replacement needed	On older computer systems, was used to produced flash-free graph animations.
Use of `ginput` when MATLAB is started with the `–noFigureWindow` or `–nodisplay` flag	Errors	Not applicable	Do not use `ginput` when you start MATLAB with the `–noFigureWindows` or `–nodisplay` flags.
Use of graphics format extensions `ai` or `ill` with `saveas`	Warns	No replacement	Remove all instances of `saveas` with the `ai` or `ill` extensions.
Use of `all` option with `hgsave` or `hgload`	Warns	No replacement	Remove all instances of passing `all` to `hgsave` or `hgload`.
Use of `getframe` with second input argument that specifies an area not contained in figure window	Warns	No replacement	If using a second input argument with `getframe`, then specify an area fully contained in figure window.
Use of `-dsetup` option with `print` to display the Windows Print Setup dialog	Warns	No replacement	Remove all instances of `-dsetup`. For the current figure, click **File > Print** in the Figure window to display the Print Setup dialog.

Data Import and Export

Reading and writing indexed and grayscale AVI files with `VideoReader` and `VideoWriter` objects

The VideoReader.read method now returns an array of grayscale data when reading grayscale AVI files. The `read` method also reads AVI files with indexed video and a colormap using the new `'native'` input argument.

The VideoWriter.writeVideo method now writes grayscale AVI files, and AVI files with indexed video and a colormap.

Compatibility Considerations

A `VideoReader` object created from an AVI file with indexed video and a grayscale colormap now has a `VideoFormat` property with the value `'Grayscale'`. By default, using the `VideoReader.read` method to read this object returns an `m`-by-`n`-by-1-by-`F` array, where `m` is the image frame height, `n` is the image frame width, and `F` is the number of frames read. In R2012b and earlier releases, the value of `VideoFormat` was `'RGB24'` and `read` by default returned an `m`-by-`n`-by-3-by-`F` array.

Writing MPEG-4 H.264 files on Mac with `VideoWriter` object

VideoWriter now writes MPEG-4 H.264 files with the extension `.mp4` or `.m4v` on systems with Mac OS X 10.7 or later.

`Tiff` object improvements for reading and writing RGB-class TIFF images

The Tiff.readRGBAImage method returns RGBA data from an entire RGB-compatible image. The Tiff.readRGBAStrip and Tiff.readRGBATile methods return RGBA data from a single strip or a single tile in an RGB-compatible image, respectively.

The `JPEGColorMode` property for `Tiff` objects controls YCbCr/RGB conversion when writing YCbCr images. For more information, see the Tiff reference page.

Importing non-ASCII encoded files with `textscan` function

The textscan function now reads text files with non-ASCII encodings. For a complete list of supported encoding schemes and the syntax for specifying the encoding, see the fopen reference page.

Multichannel JP2 support in `imread` function

By default, when reading JP2 files, the imread function now returns all image channels in the order in which they exist in the codestream. In R2012b and earlier releases, `imread` read up to 3 channels only.

Compatibility Considerations

The default syntax of `imread` no longer reorders the channels of a JP2 image where the channels are out of order. To reorder the channels, call `imread` with the `'V79Compatible'` name-value pair argument.

Previous behavior change of `xlsread` function output

In R2011b and earlier, the xlsread function returned columns of empty strings, `''`, in the text data output, where there were leading columns of numeric data preceding text data. As of R2012a, on systems with Excel for Windows, these columns of empty strings are trimmed from the text data output.

Compatibility Considerations

This change in behavior affects file reading on systems with Excel for Windows, when calling `xlsread` without the `'basic'` argument. For example, given a file, `myfile.xls` , where Sheet1 contains the following data:

```
1 1 1 A A
2 2 2 B B
3 3 3 C C
4 4 4 D D
5 5 5 E E
```

Calling `[num,txt] = xlsread('myfile.xls','Sheet1')` on systems with Excel for Windows returns:

```
num =
     1     1     1
     2     2     2
     3     3     3
     4     4     4
     5     5     5


txt =
    'A'    'A'
    'B'    'B'
    'C'    'C'
    'D'    'D'
    'E'    'E'
```

In R2011b and earlier, the command returned:

```
num =

     1     1     1
     2     2     2
     3     3     3
     4     4     4
     5     5     5


txt =

    ''    ''    ''    'A'    'A'
    ''    ''    ''    'B'    'B'
    ''    ''    ''    'C'    'C'
    ''    ''    ''    'D'    'D'
    ''    ''    ''    'E'    'E'
```

When reading XLS files, you can use the `'basic'` flag to reproduce the pre-R2012a output:

```
[num,txt] = xlsread('filename.xls',Sheet,'','basic');
```

Authentication, user name, and password inputs for `urlread` and `urlwrite` functions

The urlread and urlwrite functions accept the following optional name-value pair arguments.

Attribute	Value
`Authentication`	HTTP authentication mechanism. Currently, only basic authentication is supported.
`Username`	User identifier.
`Password`	User authentication string.

Additional audio and video file reading capabilities using Import Wizard and `importdata` function

The Import Wizard and the importdata function now import all audio file formats supported by the audioread function, including WAV, FLAC, and OGG files on all platforms, and MP3 and MPEG-4 AAC files on Windows 7 (or later), Macintosh, and Linux platforms.

The Import Wizard and the `importdata` function now import all video file formats supported by the VideoReader class.

`sound` function nonblocking

The sound function now returns immediately. In R2012b and earlier releases, `sound` blocked until all audio played back.

Compatibility Considerations

To play audio with blocking, use `playblocking(audioplayer(y,Fs))` instead of `sound(y,Fs)`.

Functionality being removed or changed

Functionality	What Happens When You Use This Functionality?	Use This Instead	Compatibility Considerations
`aufinfo`	Still runs	audioinfo	Replace all instances of `aufinfo` with `audioinfo`.

Functionality	What Happens When You Use This Functionality?	Use This Instead	Compatibility Considerations
`auread`	Still runs	audioread	Replace all instances of `auread` with `audioread`.
`auwrite`	Still runs		Write audio files using audiowrite.
`wavfinfo`	Still runs	audioinfo	Replace all instances of `wavfinfo` with `audioinfo`.
`wavread`	Still runs	audioread	Replace all instances of `wavread` with `audioread`.
`wavwrite`	Still runs	audiowrite	Replace all instances of `wavwrite` with `audiowrite`.
`cdfwrite`	Still runs		Write Common Data Format (CDF) files using the cdflib low-level functions.
`hdftool`	Still runs		Read data from HDF files using hdfread or the low-level functions in the `matlab.io.hdf4.sd`, `matlab.io.hdfeos.gd`, and `matlab.io.hdfeos.sw` packages.

Performance

`fft` function performance improvements on computers with new Intel and AMD processors

The fft function performance improves on computers with Intel and AMD processors supporting the AVX instruction set.

`permute` function performance improvements for 3-D and higher dimensional arrays

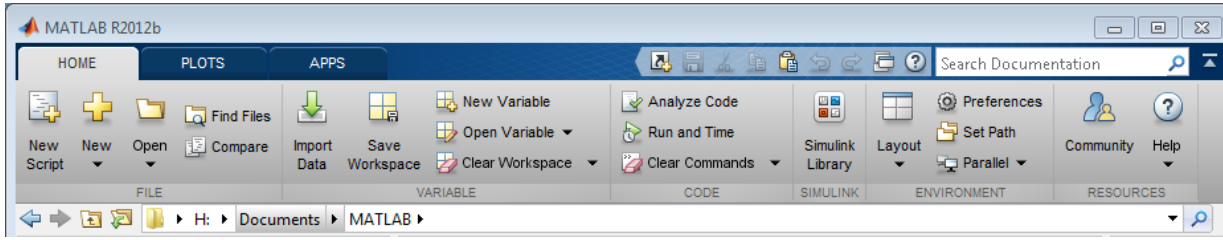
The permute function performance improves for 3-D and higher dimensional arrays.

R2012b

Desktop

Toolstrip that replaces menus and toolbars in MATLAB Desktop

New MATLAB Desktop (5 min, 3 sec)



- The Toolstrip contains components that were previously available in menus, toolbars, and the **Start** button. Tabs such as **Home**, **Plots**, **Apps**, **Editor**, and **Variable**, group functionality to support common tasks.
- The **Apps** tab contains a gallery of apps from the MATLAB family of products. You also can install your own apps, which appear in the apps gallery.
- The quick access toolbar contains frequently used options such as cut, copy, paste. The quick access toolbar is customizable; you can add items from tabs or from command shortcuts you create. You can also change the position of the toolbar on the Desktop.

- The current folder toolbar enables you to control the current working directory. The location and the controls available are customizable.

- The **Search Documentation** box enables you to search the documentation for functions and other topics of interest.

Apps gallery that presents apps from the MATLAB product family

The MATLAB desktop contains a gallery of apps from the MATLAB family of products. You can also install your own apps, which appear in the apps gallery.

- To access the apps gallery, on the desktop Toolstrip, click the **Apps** tab, and then, on the far right of the **Apps** section, click the arrow ▾.
- To find and install apps written by others, click the **Apps** tab, and in the **File** section, click **Get More Apps**.

Single-file application packaging as a MATLAB App Installer file for inclusion in the apps gallery

Packaging and Installing MATLAB Apps (2 min, 58 sec)

You can package your GUI as a single installer file that contains all of the code and data needed to run your app, as well as a description of your app. Share your app with others by uploading the installer file to File Exchange or sending it as an email attachment. When your app installs, it appears alongside apps from the MathWorks family of products in the apps gallery. For more information, see Package Apps.

Redesigned Help with improved browsing, searching, and filtering

Redesigned Help (5 min, 43 sec)

Searching and navigation

The R2012b release includes a redesigned documentation system with streamlined product pages, detailed search suggestions, and enhanced filtering of search results.

Each product's documentation is now organized by categories of functionality, such as Graphics or Simulation, rather than by information type, such as Function Reference or User's Guide. Categories include links to related reference pages, examples, and conceptual topics, allowing you to browse from one to another.

Most features of the previous Help browser are available in the new browser, although some functionality has a different appearance or location. For example:

- The table of contents for each product is collapsed by default to maximize the space available for reading topics. Expand the table of contents from any page in the Help browser by clicking the Table of Contents button,  .
- Links to alphabetical lists of reference pages, such as functions or blocks, appear at the bottom of each product page.
- Tasks previously performed using menu items in the Help browser, such as setting preferences or viewing page locations, are now accessible via context (right-click) menus, keyboard shortcuts, or toolbar buttons.

  Note: On Macintosh systems, there are no context menu items for **Evaluate Selection** or **Get Page Address**. You can copy selected code to the Command Window for evaluation by pressing **Shift+F7**. However, in this release, there is no way to identify the page address.

- Demos are now labeled *Examples*, and are accessible from the top of each product page. When you select video examples, they open in your system Web browser.

- All release notes for a product appear on the same page, allowing you to track changes across several releases from a single location.

The new Help browser is used exclusively for the documentation of MathWorks products. As a result:

- The `web` command opens all pages that are not part of the documentation in the MATLAB Web browser, even when you specify the `-helpbrowser` option. Because this is the default behavior of the `web` command, the `-helpbrowser` option has been removed from the documentation.
- When the `doc` command does not find a documentation page, but does find associated help text, it displays the text in the MATLAB Web browser rather than the Help browser.

- Custom documentation and documentation for downloadable or third-party products displays in a separate browser. To access this browser, open the Help browser and navigate to the documentation home page. Then, at the bottom of the page, click **Supplemental Software**.

Font size in Help browser and Web browser

The Help browser and Web browser now allow you to zoom in and out to adjust the font size by pressing **Ctrl** and **+** or **–**.

In-product access to online documentation

From within the product, you can access either the installed documentation or the documentation on the Web. By default, your Help preference is set to view the installed documentation. If you change the preference to view the Web documentation, then you can choose to view documentation for products that you do not have installed.

Searches using the doc command

In previous releases, if you passed a term to the `doc` command that did not correspond to the name of a MathWorks reference page or to a file with help text, the `doc` command issued an error. Now, the `doc` command searches the documentation for the term, and displays the search results in the Help browser.

Improved rendering in Help browser and Web browser

The new Help browser uses a different rendering engine than the previous Help browser. This engine provides improved legibility, particularly on Microsoft Windows 64-bit systems.

The MATLAB Web browser also uses this rendering engine on all platforms. In previous releases, the Web browser used this engine only on Microsoft Windows systems.

This rendering engine is not fully supported on Red Hat® 5 operating systems, so not all Help features are available on those systems. For example:

- Search suggestions are available from the search bar in the desktop, but not within the Help browser.
- Some components of the search results page are not available, such as filtering.
- The Help browser can display installed product documentation, but not the documentation on the Web.

Compatibility Considerations

- The Help system now requires that custom documentation files reside outside the `matlabroot` folder (but on the MATLAB search path). To view any installed custom documentation, open the Help browser and navigate to the documentation home page. Then, at the bottom of the page, click **Supplemental Software**.
- The `demo` command no longer supports categories of MATLAB or Simulink examples.
- For some products (primarily third-party toolboxes), the `demo` command requires different input values to specify the product type or name. The `demo` command now identifies each product using its `info.xml` file rather than its `demos.xml` file. For most MathWorks products, the product information is the same in both files.
- In a future release, MathWorks will remove support for `demos.m` files, an old way to include demos in the Help browser. For the recommended method, see Display Custom Examples.

Viewing of multiple documentation pages simultaneously with tabbed browsing

The Help browser now allows you to open multiple tabs so that you can view more than one documentation topic at a time. Open a new tab by clicking the New Tab button,  , or by right-clicking a link.

Suggested corrections for mistyped functions and variables in the Command Window

Command Window Suggestions for Mistyped Functions and Variables (2 min, 1 sec)

MATLAB can show suggestions in the Command Window for misspelled functions and variable names. If you enter an undefined function or variable name, MATLAB displays:

```
Did you mean:
```

followed by a suggested command at the command line. You can press **Enter** to execute that command, or **Esc** to delete the suggestion.

Full-screen view mode on Mac operating systems

MATLAB can operate in full-screen mode on Mac OS X 10.7 Lion.

Changes to -nojvm startup option on Mac

When you start MATLAB on a Mac with the `-nojvm` startup option, Handle Graphics functionality is no longer supported. MATLAB will produce an error. Previous to this release, calls to Handle Graphics functionality would work but generate a warning. (For more information about the `-nojvm` option, see the matlab (UNIX) reference page.)

Tabs in MATLAB Web browser

The MATLAB Web browser now allows you to open multiple tabs so that you can view multiple reports or Web sites simultaneously. Open a new tab by clicking the New Tab button,  .

Direct access to run configurations from the Run button

Now you can type commands directly using the **Run** options on the **Editor** tab.

In previous versions, you accessed run configurations by clicking the **Run** button down arrow and selecting **Edit Run Configurations for *`filename`***.

Compatibility Considerations

MATLAB R2012b does not automatically copy run configurations from versions prior to R2012b. To transfer run configurations from a release prior to the MATLAB R2012b installation:

1. Open `prefdir\run_configurations.m`. To identify your preference directory, type `prefdir` in the Command Window.

Run configuration entries appear similar to this example:

```
%% @name foo
%  @associatedFile C:\Documents\MATLAB\work\foo.m
%  @mostRecentlyActioned true
%  @uniqueId 695644b0:1355f0bb372:-7fb0

% Modify expression to add input arguments.
% Example:
%   a = [1 2 3; 4 5 6];
%   foo(a);
foo(1,2)
```

2. Copy the MATLAB command in the run configuration. In the example configuration, the command is `foo(1,2)`.
3. Open the MATLAB file associated with the run configuration that you want to keep. In the example configuration, the file is `C:\Documents\MATLAB\work\foo.m`.
4. Click  and then the field containing `type code to run`.
5. Paste the MATLAB command.

If the MATLAB expression spans multiple lines, create a script, and copy and paste the code into the script. You can then run the script directly using  .

Multiple vector creation from single selection in Variables editor

The Variables editor now allows you to create one or more new row or column vectors from a single selection within an existing variable. The names of the new variables are based on the name of the existing variable. Previously, the Variables editor could only create a new array from a single selection, and new variables were called `unnamed`, `unnamed1, …, unnamedN`.

Language and Programming

`Abstract` attribute for declaring MATLAB classes as abstract

Declare MATLAB classes as abstract by setting the class `Abstract` attribute. See Defining Abstract Classes for more information.

Diagnostic message improvements when attempting to create an instance of an abstract class

Attempting to instantiate an abstract class returns a list of abstract members and their defining classes in the error diagnostic message. The meta.abstractDetails function finds abstract members defined or inherited by abstract classes.

Handle and dynamicprops do not support the empty static method

For R2012b, the `handle` and `dynamicprops` classes no longer support the empty static method. This change makes these classes consistent with all MATLAB abstract classes.

Compatibility Considerations

MATLAB software issues an error for calls to the `empty` static method from the `handle` and `dynamicprops` classes:

```
% Will return errors
handle.empty
dynamicprops.empty
```

See Creating Empty Arrays for information on using `empty`.

Switch uses eq to compare enumerations

`Switch` statements use the enumeration `eq` method to compare the `switch` expression with the `case` expression (see switch). In previous releases, `switch` used a special comparison that required the result of both expressions to belong to the same enumeration class. The result of the comparison is determined by the result of:

```
switch_experssion == case_expression
```

The new behavior is consistent with that of other conditional statements, such as `if` statements.

Compatibility Considerations

Code that uses enumerations in `switch` statements can behave differently in release R2012b than in a previous release.

Cannot specify property attributes multiple times

Classes can define a property attribute only once in a properties block. Defining the same attribute more than once in a properties block causes an error when the class is instantiated.

Compatibility Considerations

Classes that define the same property attribute multiple times in a properties block will error when attempting to instantiate the class. Previously, MATLAB evaluated property attribute specifications in right to left order.

Discontinued compiler support for building MEX-files

MATLAB no longer supports the following compilers on Linux 32-bit platforms:

- GNU® gcc Version 4.4.x
- GNU gfortran 4.3.x

Compatibility Considerations

To ensure continued support for building your MEX-files, consider upgrading to another supported compiler. For an up-to-date list of supported compilers, see the Supported and Compatible Compilers website.

Jagged array support for .NET

- To create a .NET jagged array, use the NET.createArray.

- To convert a .NET jagged array to a MATLAB array, see How MATLAB Handles Jagged Arrays.
- To convert a MATLAB array to a .NET jagged array, see Pass MATLAB Arrays as Jagged Arrays.

MATLAB does not support:

- Autoconversion of `char` or cell arrays to jagged array arguments. For more information, see Call .NET Method with System.String Jagged Array Arguments.
- Autoconversion of MATLAB arrays to multidimensional jagged array arguments. For more information, see Call .NET Method with Multidimensional Jagged Array Arguments.

Java exceptions accessible to MATLAB code

If you call a Java method from MATLAB, and that code throws an exception, use the matlab.exception.JavaException class to handle the exception in MATLAB.

Ability to add jar files to static Java class path

To use third-party Java libraries in MATLAB, you can control the Java class path and native library path by creating `javaclasspath.txt` and `javalibrarypath.txt` files. For more information, see The Static Path and Locating Native Method Libraries.

Preservation of string functions for backwards compatibility

The R2010a Release Notes originally stated that the `isstr`, `setstr`, `str2mat`, `strread`, `strvcat`, and `textread` functions would be removed in a future release. As of R2012b, there are no plans to remove these functions. However, use of these functions is not recommended.

Function	Recommended Modification
`isstr`	Replace all existing instances of `isstr` with ischar.
`setstr`	Replace all existing instances of `setstr` with char.
`str2mat`	Replace all existing instances of `str2mat` with char.

Function	Recommended Modification

`strread`	Replace all existing instances of `strread` with textscan. For example, replace `[a,b,c] = strread(...)` with
	`C = textscan(...)`
	`[a,b,c] = deal(C{:})`
	Unlike `strread`, the `textscan` function converts numeric values to the specified data type, allowing preservation of integer types.

`strvcat`	Replace all existing instances of `strvcat` with char. Unlike `strvcat`, the `char` function does *not* ignore empty strings.
`textread`	Replace all existing instances of `textread` with textscan, similar to `strread`. Open and close files with `fopen` and `fclose`.

Conversion of Error and Warning Message Identifiers

For R2012b, error and warning message identifiers have changed in MATLAB.

Compatibility Considerations

If you have scripts or functions that use message identifiers that changed, you must update the code to use the new identifiers. Typically, message identifiers are used to turn off specific warning messages, or in code that uses a try/catch statement and performs an action based on a specific error identifier.

For example, the `MATLAB:uitable:InvalidParent` identifier has changed to `MATLAB:uitable:ParentMustBeFigureOrUIContainer`. If your code checks for `MATLAB:uitable:InvalidParent`, you must update it to check for `MATLAB:uitable:ParentMustBeFigureOrUIContainer` instead. For a mapping of the new identifiers to the original identifiers, see Technical Support solution `1-ERAFNC`.

Mathematics

Performance improvements and multithreading for `airy`, `psi`, and Bessel functions

The following functions show improved performance: besselh, besseli, besselj, besselk, bessely, airy, and psi.

`ddensd` function that solves delay differential equations of neutral type with state-dependent delays

The ddensd function solves neutral delay differential equations that involve both the solution and its first derivative evaluated in the past:
$y'(t) = f(t, y(t), y(dy_1),...,y(dy_j), y'(dyp_1),...,y'(dyp_k))$

The delays, *dy* and *dyp*, can be time-dependent, state-dependent, or both.

Signed integer support for bit-wise operations

The bitand, bitor, bitxor, bitcmp, bitshift, bitget, and bitset, functions now support signed integers.

An additional input argument, `assumedtype`, is an optional string argument you can pass to any of these functions. Use `assumedtype` to indicate the assumed integer class for input values of type `double`. For example, `bitor(14,240,'uint8')` treats `14` and `240` as unsigned 8-bit integers even though they are passed as integers of type `double`.

`assumedtype` can be one of the following strings: `'int8'`, `'uint8'`, `'int16'`, `'uint16'`, `'int32'`, `'uint32'`, `'int64'`, or `'uint64'`. The default value of `assumedtype` is `'uint64'` for input values of type `double`. If the input values belong to an integer class, `assumedtype` defaults to the class of the input values.

`atan2d` function that calculates four-quadrant inverse tangent with result in degrees

atan2d calculates the four-quadrant inverse tangent and returns angles in degrees that lie in the closed interval [–180,180].

Complex number support for trigonometry degree functions

The sind, cosd, tand, asind, acosd, atand, cscd, cotd, secd, acscd, asecd, and acotd functions now support complex values. For example, `sind(10+i)` returns `0.1737 + 0.0172i`. Likewise, `asind(2)` now returns the complex angle `90.0000 -75.4561i`.

Functionality being removed or changed

Functionality	What Happens When You Use This Functionality	Use This Instead	Compatibility Considerations
`bitshift(A,k,N)`	Still Runs	`bitshift(A,k,assumedtype`	Replace all instances of `bitshift(A,k,N)` with `bitshift(A,k,assumedtype)`.
`bitcmp(A,N)`	Still Runs	`bitcmp(A,assumedtype)`	Replace all instances of `bitcmp(A,N)` with `bitcmp(A,assumedtype)`.
`bitshift(A,k)` where A is of type`double`.	Still runs	Not Applicable	`bitshift` now interprets `double` input as `uint64` instead of 53-bit signed integer by default. For example, `bitshift(1,54)` now returns `1.8014e+16` instead of `0`.
`mimofr`	Warns	Not Applicable	Remove all instances of `mimofr` from your existing code.
Second output argument for besselh, besseli, besselj, besselk, bessely, and airy. For example, `[J,ierr] = besselj(nu,Z)`.	Errors	Syntax that returns only the solution vector. For example, `J = besselj(nu,Z)`.	Replace all instances that return two output arguments with the syntax that returns only the solution vector.
Passing mixed-orientation input vectors to `besselh`,	Errors	First construct the inputs with `ndgrid` or `meshgrid`. Alternatively, you can pass	Modify all instances that pass mixed-orientation vectors. You can modify

Functionality	What Happens When You Use This Functionality	Use This Instead	Compatibility Considerations

`besseli`, `besselj`, `besselk`, and `bessely`. For example, passing a row vector followed by a column vector:
`J= besselj(rowNu, col`
or passing a column vector followed by a row vector:
`J= besselj(colNu, row`
		a function handle and the mixed-orientation vectors to `bsxfun`.	your code in one of two ways:
			• Call `meshgrid` or `ndgrid` to construct the full grid first.
			`[nu,Z]= meshgrid(rowNu, ColZ);`
			`J= besselj(nu, Z);`
			• Pass a function handle and the mixed-orientation vectors to `bsxfun`. For example, if your existing code passes a row vector followed by a column vector, make the following change: `J= bsxfun(@besselj, rowNu, colZ);`or, if your code passes a column vector followed by a row vector, make the following change: `J= bsxfun(@besselj, colNu', rowZ.');`

Functionality	What Happens When You Use This Functionality	Use This Instead	Compatibility Considerations

`Y = psi(k0:k1,X)`	Errors	`Y = psi(k,X)` where `k` is a scalar specifying the `k`th derivative of ψ at the elements of `X`.	Replace all instances of `Y = psi(k0:k1,X)` with `Y = psi(k,X)`, where `k` is a scalar. To modify your code, loop through the values `k0:k1`. For example:
			`for k=k0:k1`
			`  Y(:,k) = psi(k,X);`
			`end`
			In the future, `size(Y)` will be `size(X)`. Modify any code that depends on `size(Y)`.

Passing empty and nonscalar input to `besselh`, `besseli`, `besselj`, `besselk`, `bessely`, and `airy`. For example,
`J = besselj([],(1:3))`
or
`J = besselj((1:3),[])`
	Errors	`J = besselj(nu,[])` or `J = besselj([],Z)` where `nu` and `Z` are scalars.	Modify all instances that pass combinations of empty arrays with nonscalar input. The inputs must be the same size or one must be a scalar.

Data Import and Export

Data import from delimited and fixed-width text files using Import Tool

The Import Tool now allows you to preview data in delimited and fixed-width text files and select ranges of data to import. You also can define rules for handling nonnumeric values, and import data as column vectors, a matrix, a cell array, or a dataset array, in a single step. This tool opens instead of the Import Wizard for text files.

For more information, watch this video.

Single-step import of numbers, text, and dates as column vectors from a spreadsheet with Import Tool

The Import Tool now allows you to import columns of numeric data, text, and dates from spreadsheets as multiple column vectors in one step. Previously, this functionality was available for numeric data only.

`audioread` and `audioinfo` functions for reading MP3, MPEG-4 AAC, WAVE, and other audio files

The audioread and audioinfo functions read and provide information about MP3, MPEG-4 AAC, WAVE, OGG, and FLAC files on all platforms.

`audioread` is a drop-in replacement for the most common forms of `wavread` and `auread`. In most cases where you use `wavread` or `auread`, you can rename instances of these functions to `audioread`.

`audiowrite` function for writing MPEG-4 AAC, WAVE, and other audio files

The audiowrite function writes data to WAVE, OGG, and FLAC files on all platforms. `audiowrite` writes data to MPEG-4 AAC files on Windows and Mac platforms.

`audiowrite` is a drop-in replacement for the most common forms of `wavwrite`. In most cases where you use `wavwrite`, you can rename instances of this function to `audiowrite`.

Reading and writing of BigTIFF image files larger than 4 GB

MATLAB now supports BigTIFF image files larger than 4 GB. You can use imread or the `Tiff` object.

Reading of XLSM, XLTX, and XLTM files on all platforms with `xlsread` function

The xlsread function now reads data from XLSM, XLTX, and XLTM files on all platforms. Previously, this functionality was available only on Microsoft Windows systems with Excel software.

`xlsread` function now supporting named ranges on all platforms

The xlsread function now supports named ranges as inputs, on all platforms. For example, you can call

```
num = xlsread(filename,sheet,xlRange);
```

where `xlRange` refers to a named range of data in your Excel workbook. Previously, this functionality was available only on Microsoft Windows systems with Excel software.

Multiple delimiter support in `textscan` function

The textscan function recognizes multiple different delimiters within a single file. `textscan` also can treat multiple repeated delimiter characters as a single delimiter.

Timeout, user agent, and character encoding inputs for `urlread` and `urlwrite` functions

The urlread and urlwrite functions accept the following optional attribute-value pair arguments.

Attribute	Value
`Timeout`	Timeout duration
`UserAgent`	Client user agent identification

Attribute	Value
`Charset`	Character encoding, determined from the headers of the file

For example, to download Web content from a the MATLAB Central File Exchange while specifying a timeout duration of 5 seconds, type

```
urlread('http://www.mathworks.com/matlabcentral/fileexchange','Timeout',5);
```

Functionality being removed or changed

Functionality	What Happens When You Use This Functionality?	Use This Instead	Compatibility Considerations
`avifile`	Warns	VideoWriter	Replace all instances of avifile with `VideoWriter`.
`wavplay`	Warns	audioplayer and play	Replace all existing instances of `wavplay` with `audioplayer` and `play`.
`wavrecord`	Warns	audiorecorder and record	Replace all existing instances of `wavrecord` with `audiorecorder` and `record`.
`wk1finfo`	Errors		Remove all instances of `wk1finfo`. Get information about Excel spreadsheets with xlsfinfo.
`wk1read`	Errors		Remove all instances of `wk1read`. Read Excel spreadsheets with xlsread.
`wk1write`	Errors		Remove all instances of `wk1write`. Write to Excel spreadsheets with xlswrite.

R2012a

Desktop Tools and Development Environment

Transpose and Sort Variables in the Variable Editor

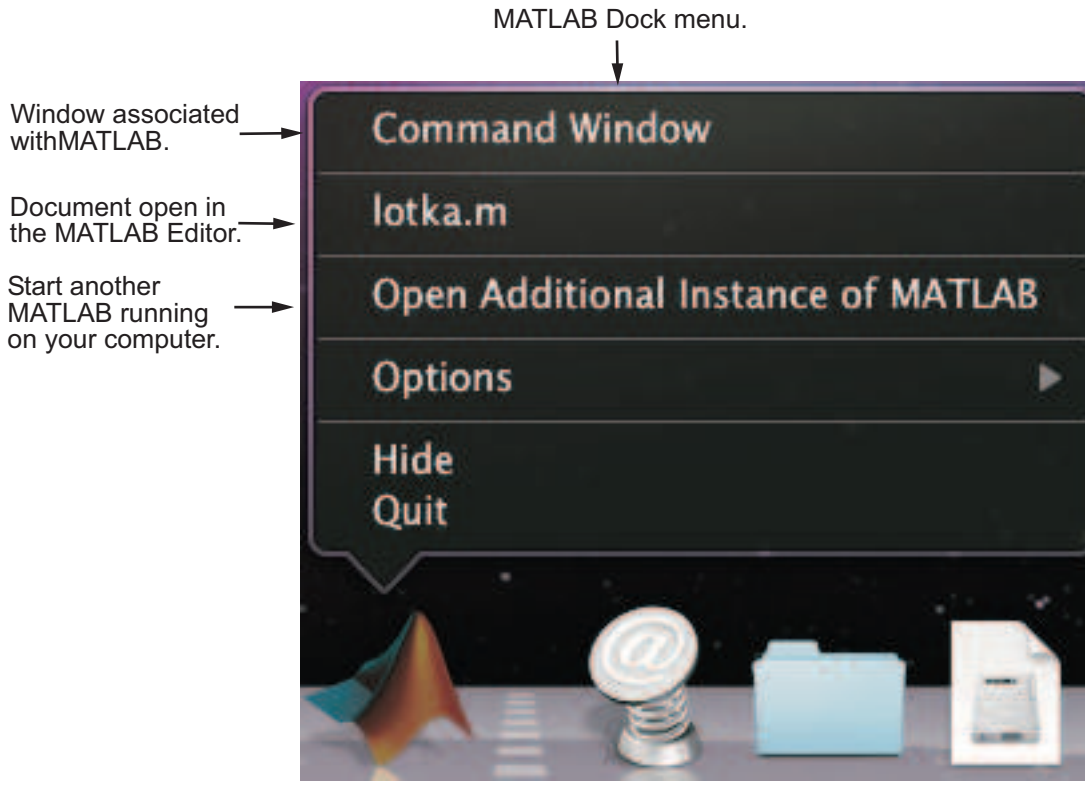
To transpose or sort variables, use the right-click menu options in the Variable Editor. You can sort variables based on single or multiple column selections. The columns you select can be noncontiguous and should always include all the rows. Also, you can create variables from noncontiguous data selections. For more information see the New Variable Editor Features in Release 2012a demo.

MATLAB Dock Menu on Mac Includes New Capabilities

Previously, on a Mac, the Dock menu associated with the MATLAB icon included only the default options, such as **Hide/Show** and **Quit**. Now, consistent with the behavior of most Mac applications, the MATLAB Dock menu lists open windows and documents associated with the running MATLAB application. By choosing on any of the entries in the Dock menu, you can bring that window or document to the front. To view the Dock menu, right-click the MATLAB icon in the Dock or **Ctrl+**click the icon, if you have a one-button mouse.

In addition, the MATLAB Dock menu includes a new option, named **Open Additional Instance of MATLAB**, that is a convenient way to start another instance of MATLAB running on your computer.

The following figure shows the MATLAB Dock menu with a document open in the MATLAB Editor.



Improved Rendering in MATLAB Web Browser

The MATLAB Web browser uses a new HTML rendering engine on Microsoft Windows systems. This new engine provides better rendering, particularly on 64-bit systems. This update impacts the display of HTML from these sources:

- `web` commands
- Published MATLAB file output
- Folder reports from the Current Folder browser
- Simulink Model Advisor

Technical Support Requests Use Proxy Settings

You can submit Technical Support requests from MATLAB by selecting **Help > Submit a MathWorks Support Request**. In previous releases, if you were connected to the Internet via a proxy server, you had to save the request form to a text file, and then email it to MathWorks. Starting with R2012a, you can submit the request directly from the form.

Published Code Can Display Syntax Highlighted Sample Code

When you publish MATLAB code, you can have sample code appear with syntax highlighting. Sample code is code that appears within comments. Previously, only uncommented code appeared with syntax highlighting.

For details, see Syntax Highlighted Sample Code. For a video demo, see Syntax Highlighting for Commented MATLAB Code in Published Scripts in Release 2012a.

Compatibility Considerations

Previously, to publish preformatted text (without syntax highlighting), MATLAB required two or more spaces between the comment character (%) and the first text character in a code cell comment block. Now, to publish such text, there must be two, four, or more spaces in that location. Three spaces results in syntax highlighted sample code.

The `publish` Function Accepts Name-Value Pairs

You can customize the output from the `publish` function using Name-Value pair arguments. Previously, to customize the output from this function, you had to use a structure. You now can use either a structure or Name-Value pair arguments. For details, see publish.

Internationalization

Displaying Non-7-Bit ASCII Characters on Mac OS X Platforms

On Apple Mac OS X platforms, you might see garbled text if you try to use non-7-bit ASCII characters. For example, if you create a variable in the MATLAB command

window containing these characters, you can type the characters, but if you display the contents of the variable, the characters are corrupted.

Or if you share MATLAB text files with a user on a computer with different locale settings, they might see corrupted text. The characters are rendered in that user's locale, not the language it was written in.

MATLAB text files include scripts and user-defined functions and classes. MATLAB displays characters in a text file using an *encoding scheme*, which is defined in the MathWorks locale database. This encoding scheme supports characters for the language specified by the user locale setting. If you try to use non-7-bit ASCII characters in a text file, you might have to use a different encoding scheme. UTF-8 is often used to handle multilingual characters. Beginning in R2011b, to handle non-7-bit ASCII characters, you can change the default encoding scheme to UTF-8 by switching the MathWorks locale database. Note that changing the default encoding scheme might cause characters other than 7-bit ASCII characters in existing text files to be garbled.

See How the MATLAB Process Uses Locale Settings for more information.

How to Change MATLAB's Default Encoding to UTF-8

If you have text files containing non-7-bit ASCII characters, you must convert the encoding in the files before changing the default encoding on your computer. For instructions, see "How to Convert Text File Encoding to UTF-8" on page 6-6.

You can change the default encoding scheme on Mac OS X platforms by using the UTF-8 locale database found in the *`matlabroot`*`/bin` folder.

To change the default locale database, type:

```
mldir = fullfile(matlabroot,'bin');
copyfile(fullfile(mldir,'lcdata.xml'),...
  fullfile(mldir,'lcdata_default.xml'));
copyfile(fullfile(mldir,'lcdata_utf8.xml'),...
  fullfile(mldir,'lcdata.xml'));
```

Alternatively, you can change the default locale database by setting an environment variable on your computer.

1. Set the environment variable, `MWLOCALE_LCDATA_FILENAME`, with the UTF-8 version of the locale database name, `lcdata_utf8.xml`, in the environment file, `environment.plist`. Please see this article from the Mac OS X Developer Library,

Technical Q&A QA1067 “Setting environment variables for user processes” at http://developer.apple.com/library/mac/#qa/qa1067/_index.html.

2. Close MATLAB if it is currently running.
3. Start MATLAB by double-clicking the MATLAB icon in the Applications folder.

How to Convert Text File Encoding to UTF-8

Before converting a file, copy the file to a new directory.

Use the Mac OS X TextEdit application to convert the file encoding. For example:

1. Open a MATLAB text file with TextEdit.
2. Select **File** -> **Save as...**
3. Change the file name.
4. Change **Plain Text Encoding:** to `Unicode (UTF-8)`.
5. Save the file.

Alternatively, the following MATLAB function, `convert_file_encoding`, creates a new text file with different encoding.

```
function convert_file_encoding(infile,outfile,from_encoding,to_encoding)

if strcmp(infile(end-2:end),'mdl');
    isMDL = 1;
else
    isMDL = 0;
end


fpIn = fopen(infile,'r','n',from_encoding);
fpOut = fopen(outfile,'w','n',to_encoding);

while feof(fpIn) == 0
    lineIn = fgets(fpIn);
    if isMDL && strncmp('SavedCharacterEncoding',strtrim(lineIn),22)
        lineIn = regexprep(lineIn,from_encoding,to_encoding);
    end

    fwrite(fpOut,lineIn,'char');
end

fclose(fpIn);
```

```
fclose(fpOut);
end
```

To use this function, you need to know the current encoding, `from_encoding`.

- For MATLAB R2010b or later:

  ```
  ret = feature('locale');
  from_encoding = ret.encoding;
  ```

- For MATLAB R2008a through R2010a:

  ```
  ret = feature('locale');
  [t,r] = strtok(ret.ctype,'.');
  from_encoding = r(2:end);
  ```

For example, a file, `myFile.m`, was created with MATLAB encoding set to `ISO-8859-1`. To convert the file to `UTF-8`, type:

```
convert_file_encoding('myFile.m','myFileUTF8.m','ISO-8859-1','UTF-8')
```

Compatibility Considerations

In a future release, the UTF-8 version of the MathWorks locale database will be the default on Mac OS X systems. Any text file created in MATLAB that is encoded in the user default encoding and contains characters other than 7-bit ASCII characters must be converted to the UTF-8 encoding scheme. Otherwise, MATLAB and other MathWorks products might not be able to properly load the files.

You must convert the file encoding before changing the default locale database.

Do not convert text files to UTF-8 if you share them with users on Windows platforms.

Mathematics

New Integral Functions

The new functions, integral, integral2, and integral3 perform numerical integration with additional support for nonrectangular and unbounded regions of integration. They are the recommended functions for performing quadrature.

Performance Enhancements

The following functions show improved performance:

- Arithmetic and similar basic math functions for double, single, and integer data types.
- The grid-based interpolation functions, interp2, interp3, and interpn.
- Generating random values using either `rng('combRecursive')` or `RandStream('mrg32k3a')`.

griddata Supports 3-D Data and Natural Neighbor Interpolation

griddata is now the recommended function for interpolating 2-D and 3-D scattered data. `griddata` also has a new `method` option, `'natural'`, for specifying natural neighbor interpolation.

TriScatteredInterp Accepts Complex Values

You can now use TriScatteredInterp to interpolate complex scattered data in a single pass. For example, `F = TriScatteredInterp(X,V)` now accepts a complex vector `V`.

Set Functions Provide Option to Return Sets in Original Order

You now can specify the ordering of the output array returned by the functions unique, union, intersect, setdiff, and setxor. The new argument, `setOrder`, is one of two strings, `'stable'` or `'sorted'`. Specify `'stable'` if you want the elements in the output array to be in the same order as in the input array. For example:

`C = unique([9 2 2],'stable')` returns `C = [9 2]`.

Specify `'sorted'` if you want the elements in the output array to be in sorted order. For example:

`C = unique([9 2 2],'sorted')` returns `C = [2 9]`.

Set Functions Changing Behavior in a Future Release

In a future release, the behavior of unique, union, intersect, setdiff, setxor, and ismember will change.

- If there are repeated elements in the input arrays, the functions `unique`, `union`, `intersect`, `setdiff`, and `setxor` will return the index to the first occurrence of the repeated elements (or rows).
- The optional output array, `locb`, returned by `ismember`, will contain the lowest absolute indices in `B` for elements (or rows) that are also in `A`.
- All index vectors returned by `unique`, `union`, `intersect`, `setdiff`, or `setxor` will be column vectors.
- `unique` and `union` will support objects with methods `sort` (`sortrows` for the `'rows'` option), and `ne`. This includes heterogeneous arrays derived from the same root class.
- `intersect`, `setdiff`, `setxor`, and `ismember` will support objects with methods `sort` (`sortrows` for the `'rows'` option), `ne`, and `eq`. This includes heterogeneous arrays derived from the same root class.
- The input arrays passed to `union`, `intersect`, `setdiff`, `setxor`, and `ismember` must be of the same class with the following exceptions:
  - Logical, char, and all numeric classes can combine with double arrays.
  - Cell arrays of strings can combine with char arrays.

Note: The set functions already conform to this behavior when you call them with the new `setOrder` argument.

Compatibility Considerations

The new behavior change is introduced for adoption in R2012a. If you want to assess the impact this change might have on your existing code, specify `'R2012a'` as the final

input argument to `unique`, `union`, `intersect`, `setdiff`, `setxor`, and `ismember`. For example:

```
[C,IA,IC] = unique([9 9 1],'R2012a')

C =

     1     9


IA =

     3
     1


IC =

     2
     2
     1
```

If the changes adversely affect your code, you can specify `'legacy'` to preserve the current behavior. For example:

```
[C,IA,IC] = unique([9 9 1],'legacy')

C =

     1     9


IA =

     3     2


IC =

     2     2     1
```

Interpolation and Computational Geometry Functionality Being Removed or Changed

Functionality	What Happens When You Use This Functionality	Use This Instead	Compatibility Considerations
`interp1q`	Still Runs	interp1	Replace all instances of `interp1q` with `interp1`.
`interp1(..., 'cubic')`	Still Runs	`interp1(..., 'pchip')`	Replace all instances of `interp1(..., 'cubic')` with `interp1(..., 'pchip')`.

Passing nonuniformly spaced points or grid to:
 	
`interp1(..., 'v5cubic`
`interp2(..., 'cubic')`
`interp3(..., 'cubic')`
`interpn(..., 'cubic')`
	Warns	`interp1(..., 'spline')`
		`interp2(..., 'spline')`
		`interp3(..., 'spline')`
		`interpn(..., 'spline')`
			In previous releases, interp1, interp2, interp3, and interpn silently changed the `'v5cubic'` and `'cubic'` methods to `'spline'` when the sample data was not uniformly spaced. Now, a warning is issued if the uniformity conditions are not honored. To avoid the warning message, change any instances that call for`'v5cubic'` or `'cubic'` to the `'spline'` method.

Passing the `'pp'` flag to `interp1`. For example:
`pp = interp1(x, v, 'linear', 'pp');`
`vq = ppval(pp, xq);`
	Still Runs	Use griddedInterpolant to create an interpolating function that is efficient to evaluate in a repeated manner.	Replace all instances of `interp1(..., 'pp')` with `griddedInterpolant`. For example:
			`F = griddedInterpolant(x, v, 'linear');`
			`vq = F(xq);`

Functionality	What Happens When You Use This Functionality	Use This Instead	Compatibility Considerations
`interp1(X, Y, Xq, [], ...)` where `[]` specifies the default interpolation method	Still Runs	`interp1(X, Y, Xq, 'linear', ...)`	Replace all instances of `interp1(X, Y, Xq, [], ...)` with `interp1(X, Y, Xq, 'linear', ...)`.

`interp1(x,V,..)` where `x` is a vector and `V` is an array representing multiple value sets. For example:
`x = (1:5)';`
`V = [x, 2*x, 3*x];`
`xq = (1:0.5:5)';`
`Vq = interp1(x, V, xq`
	Still Runs	Use `griddedInterpolant` and loop over each value set, updating and evaluating during each iteration.	Replace all instances of `interp1(x,V,...)`, where `x` is a vector and `V` is an array representing multiple value sets.
			Loop over `V`, calling `griddedInterpolant` on each value set. The following example code shows how to set up and interpolate over three value sets in `V` :
			`x = (1:5)';`
			`V = [x, 2*x, 3*x];`
			`xq = (1:0.5:5)';`
			`F = griddedInterpolant(x, V(:,1));`
			`for n=1:3`
			`  F.Values = V(:, n);`
			`  Vq(:, n) = F(xq);`
			`end`
			The columns of `Vq` are the interpolation results for the corresponding columns of `V`.

Functionality	What Happens When You Use This Functionality	Use This Instead	Compatibility Considerations

Passing mixed orientation vectors to `interp2`:
`Vq = interp2(x, y, V, xq, yq)`
Specifically, if one or both of the following are true:
• One of `x` and `y` is a row vector and the other is a column vector.
• One of `xq` and `yq` is a row vector and the other is a column vector.
	Still Runs	Construct the full grid with `meshgrid` first. Alternatively, use `griddedInterpolant` if you have a large data set.	Modify all instances that pass mixed orientation vectors to `interp2`. You can modify your code in one of two ways:
			• Call `meshgrid` to construct the full grid first.
			`[X,Y] = meshgrid(x, y);`
			`[Xq,Yq] = meshgrid(xq, yq);`
			`Vq = interp2(X, Y, V, Xq, Yq);`
			• Pass the vectors to `griddedInterpolant` inside a cell array.
			`F = griddedInterpolant({x, y}, V.');`
			`Vq = (F({xq, yq})).'`

Functionality	What Happens When You Use This Functionality	Use This Instead	Compatibility Considerations

Passing mixed orientation vectors to `interp3`: `interp3(x, y, z, V, xq, yq, zq)` Specifically, if one or both of the following are true:
• `x`, `y`, and `z` are a combination of row and column vectors.
• `xq`, `yq`, and `zq` are a combination of row and column vectors.
	Still Runs	Construct the full grid with `meshgrid` first. Alternatively, use `griddedInterpolant` if you have a large data set.	Modify all instances that pass mixed orientation vectors to `interp3`. You can modify your code in one of two ways:
			• Call `meshgrid` to construct the full grid first.
			`[X,Y,Z] = meshgrid(x, y, z);`
			`[Xq,Yq,Zq] = meshgrid(xq, yq, zq);`
			`Vq = interp3(X, Y, Z, V, Xq, Yq, Zq);`
			• Pass the vectors to `griddedInterpolant` inside a cell array.
			`V = permute(V, [2 1 3]);`
			`F = griddedInterpolant({x, y, z}, V);`
			`Vq = F({xq, yq, zq});`
			`Vq = permute(Vq, [2 1 3]);`

Functionality	What Happens When You Use This Functionality	Use This Instead	Compatibility Considerations

Passing mixed orientation vectors to `interpn`: `interpn(x1,x2,...,xV, x1q,x2q,...,xnq)` Specifically, if one or both of the following are true:
• `x1,x2,...,xn` are a combination of row and column vectors.
• `x1q,x2q,...,xnq` are a combination of row and column vectors.
	Still Runs	Construct the full grid with `ndgrid` first. Alternatively, use `griddedInterpolant` if you have a large data set.	Modify all instances that pass mixed orientation vectors to `interpn`. You can modify your code in one of two ways:
			• Call `ndgrid` to construct the full grid first.
			`[X1,X2,...,Xn]= ndgrid(x1,x2,...,xn); [X1q,X2q,...,Xnq]= ndgrid(x1q,x2q,...,xnq); Vq= interpn(X1,X2,...,Xn, V, X1q,X2q,...,Xnq);`
			• Pass the vectors to `griddedInterpolant` inside a cell array.
			`F= griddedInterpolant({x1, x2,...,xn}, V); Vq= F({x1q, x2q,...,xnq});`

Functionality	What Happens When You Use This Functionality	Use This Instead	Compatibility Considerations

`interp1(..., *METHOD)`
`interp2(..., *METHOD)`
`interp3(..., *METHOD)`
`interpn(..., *METHOD)`
	Still Runs
	
	Errors for invalid input data
		`interp1(..., METHOD)`
		`interp2(..., METHOD)`
		`interp3(..., METHOD)`
		`interpn(..., METHOD)`
			In previous releases, `interp1`,`interp2`,`interp3`, and `interpn` provided a `*METHOD` option that bypassed error checking on the assumption of valid data. These checks are no longer bypassed. Use the `griddedInterpolant` class to perform repeated interpolation queries on the same data set without penalty of repeated error checks.

`vq= griddata(x,y,v,xq,y` where `xq` is a row vector and `yq` is a column vector.	Still Runs	`vq= griddata(x,y,v,Xq,Yq)` where `Xq` and `Yq` are the output arrays returned by meshgrid.	Modify all instances that pass mixed orientation vectors to griddata. To specify a grid of query points, construct a full grid with `meshgrid` before calling `griddata`.
`griddata3`	Errors	`griddata`	Replace all existing instances of `griddata3` with `griddata`.
`delaunay3`	Errors	delaunay	Replace all existing instances of `delaunay3` with `delaunay`.
`tsearch`	Errors	DelaunayTri/pointLocation	Replace all existing instances of `tsearch` with DelaunayTri/pointLocation.

Functionality	What Happens When You Use This Functionality	Use This Instead	Compatibility Considerations
`dsearch`	Errors	DelaunayTri/ nearestNeighbor	Replace all existing instances of `dsearch` with DelaunayTri/ nearestNeighbor.
`convhull(..., OPTIONS)`	Errors	Omit the `OPTIONS` argument when you call `convhull`.	Remove the `OPTIONS` argument from all instances that pass it to `convhull`.
`delaunay(..., OPTIONS)`	Errors	Omit the `OPTIONS` argument when you call `delaunay`.	Remove the `OPTIONS` argument from all instances that pass it to `delaunay`.
`griddata(..., OPTIONS)`	Errors	Omit the `OPTIONS` argument when you call `griddata`.	Remove the `OPTIONS` argument from all instances that pass it to `griddata`.
`voronoi(..., OPTIONS)`	Errors	Omit the `OPTIONS` argument when you call `voronoi`.	Remove the `OPTIONS` argument from all instances that pass it to `voronoi`.

Other Functionality Being Removed or Changed

Functionality	What Happens When You Use This Functionality	Use This Instead	Compatibility Considerations
`isequalwithequalnan`	Still Runs	isequaln	Replace all instances of `isequalwithequalnans` with `isequaln`.

Functionality	What Happens When You Use This Functionality	Use This Instead	Compatibility Considerations
`mimofr`	Still Runs	Not Applicable	Remove all instances of `mimofr` from your existing code.
`RandnAlg` property of RandStream class	Warns	`NormalTransform` property of `RandStream` class	Replace all existing instances of `RandnAlg` with `NormalTransform`.
`setDefaultStream` method of `RandStream` class	Warns	`setGlobalStream` method of `RandStream` class	Replace all existing instances of `RandStream.setDefaultStream` with RandStream.setGlobalStream.
`getDefaultStream` method of `RandStream` class	Warns	`getGlobalStream` method of `RandStream` class	Replace all existing instances of `RandStream.getDefaultStream` with RandStream.getGlobalStream.
`atan2(y,x)` for complex `y` and `x`.	Errors	`atan2(real(y),real(x))` when `y` and `x` are complex.	Replace all existing instances of `atan2(y,x)` with `atan2(real(y),real(x))` where `y` and `x` are complex.
Second output argument for besselh, besseli, besselj, besselk, bessely, and airy. For example, `[J,ierr] = besselj(nu,Z)`.	Warns	Syntax that returns only the solution vector. For example, `J = besselj(nu,Z)`.	Replace all instances that return two output arguments with the syntax that returns only the solution vector.

Functionality	What Happens When You Use This Functionality	Use This Instead	Compatibility Considerations

Passing mixed orientation input vectors to `besselh`, `besseli`, `besselj`, `besselk`, and `bessely`. For example, passing a row vector followed by a column vector:
`J= besselj(rowNu, col`
or passing a column vector followed by a row vector:
`J= besselj(colNu, row`
	Warns	Construct the inputs with `ndgrid` or `meshgrid` first. Alternatively, you can pass a function handle and the mixed orientation vectors to `bsxfun`.	Modify all instances that pass mixed orientation vectors. You can modify your code in one of two ways:
			• Call `meshgrid` or `ndgrid` to construct the full grid first.
			`[nu,Z]= meshgrid(rowNu, ColZ);`
			`J= besselj(nu, Z);`
			• Pass a function handle and the mixed orientation vectors to `bsxfun`. For example, if your existing code passes a row vector followed by a column vector, make the following change:
			`J= bsxfun(@besselj, rowNu, colZ);`or,
			if your code passes a column vector followed by a row vector, make the following change:
			`J= bsxfun(@besselj, colNu', rowZ.');`

Functionality	What Happens When You Use This Functionality	Use This Instead	Compatibility Considerations

`Y = psi(k0:k1,X)`	Warns	`Y = psi(k,X)` where `k` is a scalar specifying the `k`th derivative of ψ at the elements of `X`.	Replace all instances of `Y = psi(k0:k1,X)` with `Y = psi(k,X)`, where `k` is a scalar. To modify your code, loop through the values `k0:k1`. For example:
			`for k=k0:k1`
			`  Y(:,k) = psi(k,X);`
			`end`
			In the future, `size(Y)` will be `size(X)`. Modify any code that depends on `size(Y)`.

Passing empty and nonscalar input to `besselh`, `besseli`, `besselj`, `besselk`, `bessely`, and `airy`. For example,
`J = besselj([],(1:3))`
or
`J = besselj((1:3),[])`
	Warns	`J = besselj(nu,[])` or `J = besselj([],Z)` where `nu` and `Z` are scalars.	Modify all instances that pass combinations of empty arrays with nonscalar input. The inputs must be the same size or one must be a scalar.

Data Analysis

Programming

xlsread Reads XLSX Files on All Platforms

The `xlsread` function now reads data from XLSX files on all platforms, including support for specifying the range and worksheet number. Previously, this functionality was available only on Microsoft Windows systems with Excel software.

VideoWriter Supports MPEG-4 Files on Windows 7 Systems

`VideoWriter` now creates MPEG-4 files with the extension `.mp4` or `.m4v` on Windows 7 systems. For more information, see the VideoWriter reference page.

audioplayer Supports Overlapping Playback

In R2011a, changes that allowed `audioplayer` to support device selection also disabled the ability to play overlapping audio samples. With R2012a, overlapping playback is available again.

For example, play a second sample before the first one completes, and hear audio from both:

```
chirpData = load('chirp.mat');
chirpObj = audioplayer(chirpData.y,chirpData.Fs);

gongData = load('gong.mat');
gongObj = audioplayer(gongData.y,gongData.Fs);

play(chirpObj);
play(gongObj);
```

importdata Returns Different Output for Some Text Files

Previously, the `importdata` function recognized comments in text files and did not include them in the output. In addition, `importdata` parsed files that contained numeric values and comma, space, or tab delimiters into a numeric array, even when you specified a delimiter that did not match the contents of the file.

In this release, `importdata` honors the specified delimiter, and includes comments in the output. For example, consider a hypothetical file named `test.txt` with this space-delimited data:

```
% Comment in file
1 2 3
4 5 6
```

In previous releases,

```
t = importdata('test.txt',',')
```

returned a numeric array:

```
t =
     1     2     3
     4     5     6
```

In this release, the same code returns a cell array:

```
t =
    '% Comment in file'
    '1 2 3'
    '4 5 6'
```

If you remove the incorrect comma delimiter,

```
t = importdata('test.txt')
```

`importdata` returns a struct array:

```
t =
        data: [2x3 double]
    textdata: {'% Comment in file'}
```

Exponents Print with Two Digits

In previous releases, functions that printed floating-point values with exponents used three digits for the exponent on Windows systems, but two digits on any other system. Now, these functions use two digits for the exponent on all systems.

For example,

```
str = sprintf('%e',pi)
```

always returns

```
str =
3.141593e+00
```

Conversion of Error and Warning Message Identifiers

For R2012a, error and warning message identifiers have changed in MATLAB.

Compatibility Considerations

If you have scripts or functions that use message identifiers that changed, you must update the code to use the new identifiers. Typically, message identifiers are used to turn off specific warning messages, or in code that uses a try/catch statement and performs an action based on a specific error identifier.

For example, the `MATLAB:eigs:NonPosIntSize` identifier has changed to `MATLAB:eigs:RoundNonIntSize`. If your code checks for `MATLAB:eigs:NonPosIntSize`, you must update it to check for `MATLAB:eigs:RoundNonIntSize` instead. For a mapping of the new identifiers to the original identifiers, see Technical Support solution `1-ERAFNC`.

Specify a List of Allowed Subclasses in the Class Definition

You can control which classes are allowed to subclass a class that you define by specifying a list classes in the new class attribute, `AllowedSubclasses`. See Controlling Allowed Subclasses for more information.

Specify Which Classes Can Access Class Members

You can control access to specific class members (properties, methods, and events) by specifying a list of classes to which you want to grant access to these members. See Controlling Access to Class Members for more information.

Compatibility Considerations

In previous releases, you could assign a value to the following class member access attributes as a value returned by a function:

- Properties – `Access`, `GetAccess`, and `SetAccess`
- Methods – `Access`
- Events – `ListenAccess` and `NotifyAccess`

With this release, you can specify values for these attributes only explicitly as either the appropriate character string or a `meta.class` object returned by the ? operator, or a cell array of character strings and/or `meta.class` objects. See Specify Access to Class Members for more information.

Method Declared as Abstract and Private Now Errors

In previously releases, declaring a method as both `Abstract` and `Access = private` did not cause an error. With this release, declaring a method both `Abstract` and `private` causes an error.

Compatibility Considerations

If any of your class definitions declare a method as both `Abstract` and `private`, you need to declare the method either as `Abstract`, which make the class abstract and requires subclasses to implement the method with the declared syntax, or declare the method `Access = private`, which means that only methods within the defining class can call the `private` method.

Declaring a method as both `Abstract` and `private` is not supported.

New Capabilities for Writing Image Data to FITS Files

MATLAB now includes a function named fitswrite that you can use to write image data to Flexible Image Transport System (FITS) files. You can also display the metadata for all Header-Data Units (HDUs) in a FITS file using the fitsdisp function. (MATLAB already supports reading FITS files using the fitsread function.)

In addition, MATLAB now includes a new package, called `matlab.io.fits`, containing dozens of functions that provide access to the routines in the CFITSIO library. Using these low-level functions, you can create FITS files, read data from FITS files and write data to the files, using the CFITSIO library, version 3.27.

Access Data on Remote Servers Using the OPeNDAP Protocol

You can read data stored on remote servers using the OPeNDAP protocol capability of the following MATLAB NetCDF functions:

- ncread

- ncreadatt
- ncdisp
- ncinfo
- netcdf.open

To use these functions to access data on remote servers using OPeNDAP protocol, specify the URL instead of a file name.

Upgrades to Scientific File Format Libraries

The following table lists upgrades to scientific file format libraries used by MATLAB.

Library	Version
NetCDF	4.1.3 (upgraded from 4.1.2)

Ability to Read NetCDF Files Using HDF4 Functions Removed

In past releases, for certain NetCDF files on Linux and Macintosh systems, you could read the files using the HDF4 functions. This capability has been removed. To read NetCDF files, use the high-level NetCDF functions or the low-level NetCDF package.

Functionality Being Removed or Changed

Functionality	What Happens When You Use This Functionality?	Use This Instead	Compatibility Considerations
`avifile`	Still runs	VideoWriter	Replace all instances of avifile with `VideoWriter`.
`fileparts` return argument 4 (file version)	Errors — beginning in MATLAB Version 7.13 (R2011b)	No alternative; file versions are unsupported	Call fileparts with three return arguments: `[path_name, file_name, file_extension]`
`helpbrowser`	Still runs	doc	Replace all instances of `helpbrowser` with `doc`.

Functionality	What Happens When You Use This Functionality?	Use This Instead	Compatibility Considerations
`helpdesk`	Still runs	doc	Replace all instances of `helpdesk` with `doc`.
`helpwin`	Still runs	doc	Replace all instances of `helpwin` with `doc`.
`info`	Warns		Remove all instances of `info`. Find information about MathWorks at `www.mathworks.com/company/aboutus/contact_us`.
`mmreader`	Warns, creates a `VideoReader` object	VideoReader	Replace all instances of `mmreader` with `VideoReader`.
`mmreader.isPlatformSupported` and `VideoReader.isPlatformSuppor`	Errors		For a platform-specific list of supported video file formats, use getFileFormats.
`support`	Warns		Remove all instances of `support`. Find the MathWorks technical support page at `www.mathworks.com/support`.
`whatsnew`	Warns		Remove all instances of `whatsnew`.

Graphics and 3–D Visualization

Creating Graphical User Interfaces (GUIs)

External Interfaces/API

Changes to Compiler Support

New Compiler Support

MATLAB Version 7.14 (R2012a) supports these new compilers for building MEX-files on Linux 64- and 32-bit platforms:

- GNU gcc Version 4.4.6

MATLAB supports these new compilers for building MEX-files on Windows 64- and 32-bit platforms:

- Intel Visual Fortran Composer XE 2011 SP1 (12.1)

Discontinued Compiler Support

MATLAB no longer supports the following compilers:

Windows 64-Bit Platforms

- Microsoft Visual Studio® 2010 (10.0) Express

Windows 32-Bit Platforms

- Microsoft Visual Studio 2010 (10.0) Express
- Visual C++ 6.0

Linux (64- and 32-Bit) Platforms

- GNU gcc Version 4.3.x

Compatibility Considerations

To ensure continued support for building your MEX-files, consider upgrading to another supported compiler. For an up-to-date list of supported compilers, see the Supported and Compatible Compilers Web page.

mxAssert and mxAssertS Functions Throw MATLAB Exception

The mxAssert and mxAssertS functions now throw the MATLAB exception `MATLAB:mex:assertion` and terminate the MEX-file instead of aborting MATLAB.

Version Support for COM ProgID Values

You can specify version-specific Programmatic Identifiers (ProgID) for MATLAB Version 7.14 (R2012a) and later.

- `MATLAB.Desktop.Application`
- `MATLAB.Application`
- `MATLAB.Application.Single`, to specify a dedicated server.

R2011b

Desktop Tools and Development Environment

Command Window

Error Messages Reformatted for Improved Readability and Navigation

Error messages in the Command Window no longer appear with question marks or arrows. In addition, error messages for each function now provide a link to that function's documentation.

The following images show examples of the differences in presentation:

- Built-in functions:
  - MATLAB Version 7.13 (R2011b)

```
>> eig
Error using eig
Not enough input arguments.
```

  - MATLAB Version 7.12 (R2011a) and earlier

```
>> eig
??? Error using ==> eig
Not enough input arguments.
```

- Functions that are not built in:
  - MATLAB Version 7.13 (R2011b)

```
>> num2str
Error using num2str (line 40)
Not enough input arguments.
```

  - MATLAB Version 7.12 (R2011a) and earlier

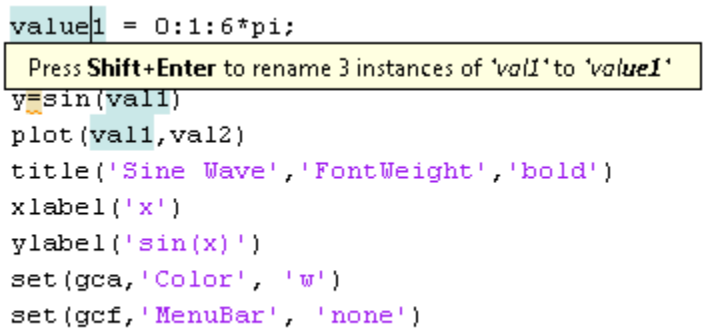
```
>> num2str
??? Error using ==> num2str at 42
Numeric array is unspecified
```

Editor

New Desktop features and changes introduced in this version are:

Automatically Renaming All Variables and Functions in File

Under certain conditions, as you rename a variable or function, a tooltip opens. The tooltip indicates that MATLAB can rename all instances of the function or variable in the current file. This is helpful in preventing errors that result if you change names manually and overlook or mistype one or more instances of the name. For a video overview, watch this video demo



For details, including the circumstances under which MATLAB prompts you, see Automatically Rename All Functions or Variables in a File.

Internationalization

Displaying Multilingual Characters on Mac OS X Platforms

On Apple Mac OS X platforms, you might see garbled text if you try to use multilingual characters. Suppose you create a variable in the MATLAB command window containing multilingual characters. You can type the multilingual characters, but if you display the contents of the variable, the characters are corrupted.

MATLAB text files include scripts and user-defined functions and classes. MATLAB displays characters in a text file using an *encoding scheme*, which is defined in the MathWorks locale database. This encoding scheme supports characters for the language specified by the user locale setting. If you try to use multilingual characters in a text

file, you may have to use a different encoding scheme. UTF-8 is often used to handle multilingual characters. In R2011b, to handle multilingual characters, you can change the default encoding scheme to UTF-8 by switching the MathWorks locale database. Note that changing the default encoding scheme might cause characters other than 7-bit ASCII characters in existing text files to be garbled.

See How the MATLAB Process Uses Locale Settings for more information.

How to Change MATLAB's Default Encoding to UTF-8

You can change the default encoding scheme on Mac OS X platforms by using the UTF-8 locale database found in the *`matlabroot`*`/bin` folder. If you have text files containing characters other than 7-bit ASCII characters, you must convert the encoding before changing the default encoding. For instructions, see “How to Convert Text File Encoding to UTF-8” on page 7-4.

To change the default locale database, type:

```
mldir = fullfile(matlabroot,'bin');
copyfile(fullfile(mldir,'lcdata.xml'),...
  fullfile(mldir,'lcdata_default.xml'));
copyfile(fullfile(mldir,'lcdata_utf8.xml'),...
  fullfile(mldir,'lcdata.xml'));
```

How to Convert Text File Encoding to UTF-8

Before converting a file, copy the file to a new directory.

Use the Mac OS X TextEdit application to convert the file encoding. For example:

1. Open a MATLAB text file with TextEdit.
2. Select **File ->Save as...**
3. Change the file name.
4. Change **Plain Text Encoding:** to `Unicode (UTF-8)`.
5. Save the file.

Alternatively, the following MATLAB code creates a new text file with the encoding set to `UTF-8`:

```
function convertencoding(infile, outfile)

fpIn = fopen(infile, 'r', 'n');
```

```
fpOut = fopen(outfile, 'w', 'n', 'UTF-8');

while feof(fpIn) == 0
  lineIn = fgets(fpIn);
  fwrite(fpOut, lineIn, 'char');
end

fclose(fpIn);
fclose(fpOut);
end
```

Compatibility Considerations

In a future release, the UTF-8 version of the MathWorks locale database will be the default on Mac OS X systems. Any text file created in MATLAB that is encoded in the user default encoding and contains characters other than 7-bit ASCII characters must be converted to the UTF-8 encoding scheme. Otherwise, MATLAB and other MathWorks products might not be able to properly load the files.

You must convert the file encoding before changing the default locale database.

Mathematics

New Functionality for Grid-Based Interpolation

MATLAB includes a new class, griddedInterpolant, for grid-based interpolation. `griddedInterpolant` works with grids in ndgrid format and complements the functionality provided by interpn to provide improved performance and memory efficiency.

Performance Enhancements

The following functions show improved performance:

- Some integer math functions
- Some linear algebra functions, including chol and 3-output form of qr
- Trigonometric functions

Permutation Option for randperm

The randperm function and the RandStream.randperm method now have the option of returning random permutations of `k` integers selected at random from the integers 1 through `n`.

Return Permutation Information in Vector for qr

If you use the three-output form of qr in noneconomy mode, you can now specify `'vector'` as an input argument. The permutation information will return a vector instead of a matrix.

Changes to meshgrid and ndgrid

The functions meshgrid and ndgrid have changed behavior under certain calling scenarios:

- When more than one input argument is passed to `meshgrid`, the dimensionality of the output arrays is deduced from the number of inputs. Previously the dimensionality was deduced from the number of outputs. For example, `[x,y] =`

`meshgrid(1:3,1:4,1:5)` will now assume a missing third output argument and return 3-D output arrays `x` and `y`. Previously, the third input was ignored and the function returned 2-D arrays for `x` and `y`.

- When a single argument is passed to `ndgrid`, the dimensionality of the output arrays is dictated by the number of output arguments. `ndgrid` has been revised to maintain consistency in the case of a single input and output. `ndgrid` will now degenerate naturally to support 1-D outputs. For example, `x = ndgrid(1:5)` returns a `5`-by-`1` output vector `x`. Previously, a `5`-by-`5` array was returned.

Functionality Being Removed or Changed

Functionality	What Happens When You Use This Functionality	Use This Instead	Compatibility Considerations
`bessel`	Errors	besselj	Replace all instances of `bessel` with besselj.
ODE solver syntax *`solver`*`('vdp',t0,tfi` where *`solver`* is one of the ODE solvers `ode45`, `ode23`, `ode113`, `ode15s`, `ode23s`, `ode23t`, or `ode23tb`	Errors	*`solver`*`('vdp',[t0 tfinal],y0)` or *`solver`*`(@vdp,[t0 tfinal],y0)`	Replace all instances of the syntax `t0,tfinal` with `[t0 tfinal]`.
Values `on` and `off` for `Mass` property of odeset and odeget	Errors	Set the `Mass` property to a constant matrix or a function that evaluates the variable mass matrix.	Update your code to use a constant mass matrix or a function that evaluates the variable mass matrix. For more information, check the supported settings of `Mass` and `MassStateDependency` in the odeset properties.
`MassConstant` property of odeset and odeget	Errors	For a constant mass matrix, set the `Mass` property to that matrix. For a time-dependent mass matrix, set the `Mass`	Update your code to use a constant mass matrix or a function that evaluates the variable mass matrix.

Functionality	What Happens When You Use This Functionality	Use This Instead	Compatibility Considerations
		property to a function that evaluates the matrix and set the `MassStateDependency` property to `none`.	For more information, check the supported settings of `Mass` and `MassStateDependency` in the odeset properties.

Data Analysis

Programming

Load and Save Parts of Variables in V7.3 MAT-Files

The new `matfile` function creates a `matlab.io.MatFile` object that can efficiently load or save to parts of variables in Version 7.3 MAT-Files. Loading part of a variable requires less memory than loading the entire contents of that variable.

For example, these commands create a MAT-file and add data to part of variable `X`:

```
new = matfile('newfile.mat','Writable',true);
new.X(6:10,6:10) = magic(5);
```

This command loads part of the data into variable `partOfX`:

```
partOfX = new.X(1:8,1:8);
```

Note: Syntax such as `size(new.X)` or `new.X(end,end)` temporarily loads the entire contents of variable `X` into memory. To determine the size of a variable without loading, use the `size` method for `matlab.io.MatFile` objects:

```
sizeOfX = size(new,'X');
```

For more information, see the matfile reference page or watch this video demo.

Nonmatching Function Name Warning is Now an Error

In previous releases of MATLAB, entering a function name that did not match any known function name in the given letter case resulted in a warning. After displaying this warning message, MATLAB attempted find a function that would match if letter case were ignored. If such a match were found, MATLAB ran this function instead.

In MATLAB R2011b, the MATLAB software generates an error if it can't find an exact-case match. The message displayed by the error may suggest the correct spelling for the function that was entered. The message identifier for this error reads as follows:

```
'MATLAB:dispatcher:InexactCaseMatch'
```

Compatibility Considerations

If you have a program that calls another program with nonmatching letter case, the call to this function now throws an error.

New narginchk Function Replaces nargchk

narginchk is a new function that replaces and adds to the functionality of nargchk.

The `narginchk` function differs from `nargchk` in the following ways:

- `narginchk` accepts only 2 inputs: the minimum and maximum allowable number of arguments one can pass to the currently running function.
- `narginchk` returns no outputs. Instead, it throws an error if the number of arguments passed to the current function is outside the allowable range

Conversion of Error and Warning Message Identifiers

For R2011b, error and warning message identifiers have changed in MATLAB.

Compatibility Considerations

If you have code that uses message identifiers that changed, you must update the code to use the new identifiers. Typically, message identifiers are used to turn off specific warning messages, or in code that uses a try/catch statement and performs an action based on a specific error identifier.

For example, the `MATLAB:eigs:NonPosIntSize` identifier has changed to `MATLAB:eigs:RoundNonIntSize`. If your code checks for `MATLAB:eigs:NonPosIntSize`, you must update it to check for `MATLAB:eigs:RoundNonIntSize` instead. For a mapping of the new identifiers to the original identifiers, see Technical Support solution `1-ERAFNC`.

New Spreadsheet Import Tool

The new Spreadsheet Import Tool allows you to select ranges of data and define rules for handling nonnumeric values. This tool opens instead of the Import Wizard for spreadsheets and comma-separated value (CSV) files. For more information, watch this video demo.

Two Functions Added to netCDF Low-Level Package To Aid Performance

The NetCDF low-level package now includes two functions that you can use to improve performance:

- `netcdf.getChunkCache` – Return default chunk cache settings.
- `netcdf.setChunkCache` – Set the default chunk cache settings.

Improved Performance for TIFF Input and Output

Input and output performance for TIFF files has been improved, when using `imread` and `imwrite`.

Upgrades to Scientific File Format Libraries

The following table lists upgrades to scientific file format libraries used by MATLAB.

Library	Version Upgrade
NetCDF	4.0.1 to 4.1.2
HDF5	1.8.3 to 1.8.6
HDF4	4.2r4 to 4.2.5
HDF-EOS	2.16 to 2.17
TIFF	3.7.1 to 3.9.5

VideoReader Supports MPEG-4 and MOV on Windows 7 Systems

`VideoReader` now imports MPEG-4 and Apple QuickTime movies, and any other formats supported by Microsoft Media Foundation, on Windows 7 systems. For more information, see the VideoReader reference page.

MATLAB Warns if Listener Defined for Nonobservable Property

If you attempt to create a `PreSet` or `PostSet` listener for a property that is not declared as `SetObservable`, MATLAB issues a `MATLAB:class:nonSetObservableProp` warning.

If you attempt to create a `PreGet` or `PostGet` listener for a property that is not declared as `GetObservable`, MATLAB issues a `MATLAB:class:nonGetObservableProp` warning.

In previous releases, MATLAB did not warn in either case, but did not call the listener callback function when a set or get event occurred. In a future release, these warnings will become errors.

Compatibility Considerations

If you use class metadata or the properties function to get a list of properties for which you create listeners, add warning/error handling for these conditions. A better approach is to use findobj to get the list of observable properties for any given class and assign listeners only for those properties.

MATLAB Warns if Property Is Not Member of Class When Defining Listener

When calling the event.proplistener constructor, or calling the addlistener method with an array of meta.property objects, warning messages alert you if a property is not a member of the correct class.

- **Nonscalar input objects**: if any one of the properties is not a valid member of the class of the object array, MATLAB issues the `MATLAB:class:PropNotMember` warning .
- **Scalar input object**: if any of the properties are not valid dynamic or class properties, MATLAB issues the `MATLAB:class:DynPropNotMember` or `MATLAB:class:PropNotMember` warning.

In previous releases, MATLAB did not warn in either case. In a future release, these warnings will become errors.

Built-In disp Works with Empty Objects

It is now possible to call the built-in disp function with empty objects. For example, define the `TestClass` as:

```
classdef TestClass
   properties
      PropertyA = 1;
```

```
   end
end
```

Then call the built-in version of `disp`:

```
>> builtin('disp',TestClass.empty)
  0x0 empty TestClass

  Properties:
    PropertyA

  Methods
```

In previous releases, calling the built-in `disp` function did not display anything with empty objects.

MATLAB Warns if Class Defines Property as Dependent and Constant

MATLAB now issues a warning if a class defines properties as both `Dependent` and `Constant`. In a future release, this warning will become an error.

Support for Switch/Case Statements with Objects

You can construct switch statement blocks using objects as both the switch expression and the case expression. The object's class must define or inherit an `eq` (`==`) method. See Using Switch/Case Statements with Objects in Functions Used with Objects for more information.

Functionality Being Removed or Changed

Functionality	What Happens When You Use This Functionality?	Use This Instead	Compatibility Considerations
nargchk	Still runs	narginchk	Accepts only `min` and `max` inputs, returns no output, generates error.
nargoutchk	Still runs	nargoutchk (see new syntax)	Accepts only `min` and `max` inputs, returns no output, generates error.

Graphics and 3-D Visualization

Creating Graphical User Interfaces (GUIs)

External Interfaces/API

Changes to Compiler Support

New Compiler Support

MATLAB Version 7.13 (R2011b) supports these new compilers for building MEX-files on Windows 64- and 32-bit platforms:

- Microsoft Windows Software Development Kit for Windows 7 and .NET Framework 4, Version 7.1
- Intel C++ Composer XE 2011
- Intel Visual Fortran Composer XE 2011

MATLAB Version 7.13 (R2011b) supports the following new compiler for building MEX-files on Mac OS X 64-bit platforms:

- Apple Xcode 4.0.0 with gcc 4.2.x

Compiler Support To Be Phased Out

Support for the following compilers on Windows 64- and 32-bit platforms will be discontinued in a future release, at which time new versions will be supported. For an up-to-date list of supported compilers, see the Supported and Compatible Compilers Web page.

- Intel C++ Version 11.1
- Intel Fortran Version 11.1

New Object Support for mxGetProperty and mxSetProperty Functions

You can read and modify properties of Java and COM objects using the mxGetProperty and mxSetProperty functions.

MEX Header File Does Not Define C++ Type char16_t

The `matrix.h` header file, used by the `mex.h` header file for building C/C++ MEX-files, no longer defines the C++ type `char16_t`.

Compatibility Considerations

If your MEX-file uses the `char16_t` type, you must include the appropriate header file for your compiler in order to build the MEX-file. For example, the header file for Microsoft Visual Studio Version 10.0 is `yvals.h`.

New Support for Features in Microsoft .NET Framework

Support for Cell Arrays

Creating Cell Arrays from .NET Stings and Objects

Use the MATLAB cell function to convert .NET `System.String` and `System.Object` arrays to MATLAB cell arrays. For more information, see Converting .NET Arrays to Cell Arrays.

Passing Cell Arrays to .NET Methods

If an input argument to a .NET property or method is an array of `System.Object` or `System.String`, you can pass a cell array. MATLAB automatically converts a cell array into the appropriate .NET array. For more information, see Pass Cell Arrays.

Support for Auto-Conversion of Multidimensional Arrays

You can pass MATLAB arrays directly to .NET without explicitly converting them into .NET arrays. MATLAB also converts an array of a lower dimensionality to a higher dimensionality .NET array. For more information, see Pass Arrays.

Compatibility Considerations

In most cases, you no longer need to use the NET.convertArray function. Review your MATLAB code to update the use of this function. If you continue to use `NET.convertArray`, note the following change of behavior for empty matrices:

- If you call `NET.convertArray` with an N-D empty matrix and try to convert it to 1-D .NET array, MATLAB throws the following error:

  ```
  Source and destination array dimension sizes do not match.
  ```

- Different results for:

  ```
  NET.convertArray(rand(0,1,1))
  ```

  As of R2011a, MATLAB creates a `System.Double[,]`. In R2011b, MATLAB creates a `System.Double[]`.

COM Automation Server Error Message Formatting

Although some MATLAB Command Window messages have been reformatted (see "Error Messages Reformatted for Improved Readability and Navigation" on page 7-2), error messages from the Execute function can still be identified with leading `???` characters.

R2011a

Desktop Tools and Development Environment

Desktop

New features and changes introduced in Version 7.12 (R2011a) are:

MATLAB Menus Display at the Top of the Apple Mac Screen

Previously when running on Apple Mac, menus displayed at the top of the MATLAB desktop. In addition, if a tool was undocked from the desktop, the menu displayed at the top of that tool. Now, to be consistent with the behavior of most Mac applications, MATLAB menus display at the top of the Mac screen.

Help Browser

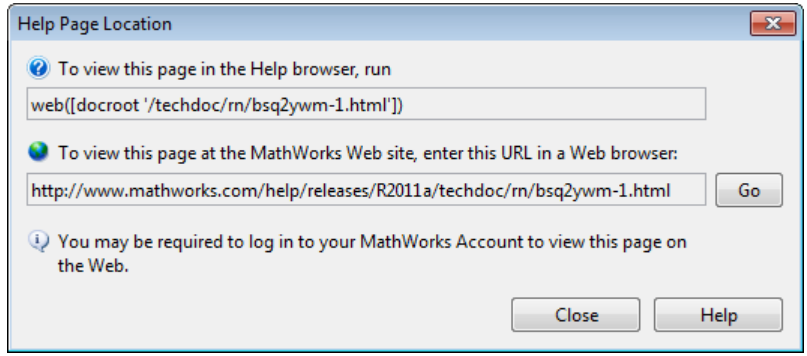
New Location and Archived Content for Product Documentation

The new MathWorks Web site location for product documentation is http://www.mathworks.com/help/. Accessing the old URL, http://www.mathworks.com/access/helpdesk/help/helpdesk.html, redirects you the new location.

In addition to all current product documentation, the new location provides archived product documentation from all releases going back to R13sp2 (including Japanese, when available). To go to the archive, click the View documentation for other releases link. The archived release documentation is available only at the MathWorks Web site, and not via the Help browser.

To view some documentation you must log on to `www.mathworks.com` with your MathWorks user name and password. For example, you need to log in to view current PDF files and documentation from prior releases.

The **View > Page Location** dialog box in the Help browser now links to documentation at the MathWorks Web site that accompanies the release you are currently running. For example, when you run R2011a, the dialog box looks like this.



As of R2011a, the Help Browser **View > Page Location** dialog box (shown above) is available on Japanese language systems.

Submit Support Requests Directly from MATLAB

As of this release, if you need assistance from MathWorks product support, you can request it directly from the MATLAB desktop. Selecting the **Submit a MathWorks Support Request** option from the **Help** menu opens a login dialog for you to provide the e-mail address and password for your MathWorks account. After you log in, a form displays for you to create a service request. Select the product for which you want help, explain the issue you are having, give the request a meaningful title, describe the issue you are having, provide reproduction steps, and attach files that enable Technical Support to follow the steps.

If you do not have a MathWorks account, the login form enables you to create one. MathWorks accounts are free and without obligation. You can also use the account login form to obtain a forgotten MathWorks password. You can log out before submitting your request or remain logged in.

For an example of submitting a support request, see this video demo. For further details about requesting service requests from MATLAB, see Contact Technical Support.

Managing Files

New features and changes introduced in Version 7.11 (R2011a) are:

- “Renaming Files and Folders in the Current Folder Browser Now Reflected in the Editor” on page 8-4

Renaming Files and Folders in the Current Folder Browser Now Reflected in the Editor

If a file is open in the Editor and you:

- Rename that file in the Current Folder browser, then the file name updates in the Editor
- Rename the folder containing that file in the Current Folder Browser, then MATLAB updates the file specification for all open Editor documents in the renamed folder
- Delete that file from the Current Folder browser, then MATLAB deletes the file from disk and renames the Editor document to `Untitled*`

  If more than one unnamed document is open in the Editor, then MATLAB renames the deleted document `Untitledn*`, where `n` is an integer.

However, modifying file and folder names from either of the following does not update names for open documents in the Editor:

- The operating system file manager — such as Windows Explorer
- The MATLAB Command window — using movefile, for instance

Options Names Changed for Locating and Opening Files and Folders Outside the MATLAB Desktop

To locate a file or folder in Windows Explorer or Apple Mac Finder, you previously used the Current Folder browser context-menu option, **Locate on Disk**. Now the option is one of the following:

- On Microsoft Windows systems—**Show in Explorer**
- On Macintosh systems—**Show in Finder**

In addition, from the Current Folder browser, you can open the current folder in Explorer or Finder. Right-click in white space, and then select **Open Current Folder in Explorer** or **Open Current Folder in Finder**.

Comparison and Merging of MAT-file Variables

When comparing MAT-files, you now can view details of differences between variables to see which fields of a structure are different and to view differences in individual elements of an array. You can merge changes between files by copying variables from one file to another. For details, see Comparing MAT-Files.

Filter Results in Folder Comparisons

You now can define filters to exclude unimportant differences when comparing folders. For example, you can exclude backup files or files created by a revision control system. Filters can save time when reviewing differences, especially when comparing many subfolders. For details, see Comparing Folders and Zip Files.

Showing Differences Only in Text Comparisons

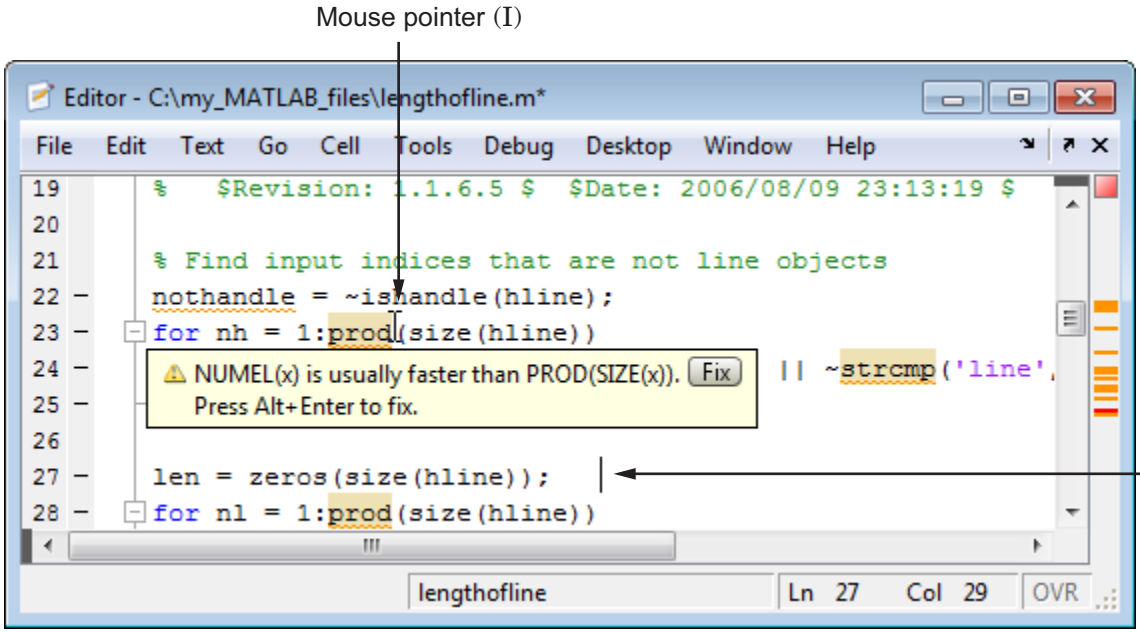
When comparing text files, you now can specify whether to show only differences. Use the new toolbar button in the Comparison Tool to hide sections of the report that do not include any differences. It can be useful to hide unmodified lines in large text comparison reports. For details, see Comparing Text Files.

Editing and Debugging MATLAB Code

New Desktop features and changes introduced in Version 7.11 (R2011a) are:

Change to Tooltip and Behavior for M-Lint Messages

Previously, when you hovered the *mouse pointer* over code in the Editor that has an autofixable M-Lint warning message, a tooltip opened that included the phrase, **Press Alt + Enter to fix**, as shown in this image.



Now, this phrase is removed from the M-Lint tooltip. You can still use the keyboard shortcut, **Alt+Enter**, to apply the autofix, but your *cursor* must be within the code to which the autofix will be applied.

Compatibility Considerations

When an M-Lint tooltip is open and you want to apply the autofix for that warning, click the **Fix** button. To apply the same autofix using a keyboard shortcut, move the cursor within the code that generates the warning, and then click **Alt+Enter**.

Changed Default Preference for Deleting Autosaved Files

Previously, the **Automatically delete autosave files** preference for the Editor was disabled by default. Now, it is enabled by default. However, as usual, MATLAB retains settings you set explicitly in a previous release (unless you delete `matlab.prf`.)

For more information, see Autosaving Files and Where MATLAB Stores Preferences.

Shared Scope Color Preferences Apply to Persistent and Global Variables

Previously, the preference for **Variables with shared scope** was labeled **Nonlocal variables** and it affected the color of variables within nested functions only. Now, the Editor also applies this color preference to variables you declare as global or persistent.

This feature makes it easier to find reuses of a variable, which sometimes indicate variable scoping problems. For details, including information on how to disable or change the colors that the Editor uses for variables with shared scope, see Avoid Variable and Function Scoping Problems.

Publishing MATLAB Code

New Desktop features and changes introduced in Version 7.11 (R2011a) are:

Change to Menu Option for Including Blocks of LaTeX Code

Previously, if you wanted to include blocks of LaTeX code within your MATLAB code intended for publishing, you selected **Cell > Insert Text Markup > LaTeX Equation**. Now you select **Cell > Insert Text Markup > LaTeX Display Math**. The name of the option was changed to better distinguish it from a new publishing option that enables you to include inline LaTeX equations in your published code.

For details see LaTeX Display Math.

Menu Option to Include Inline LaTeX Math in Published MATLAB Code

The Cell menu now provides an option for including inline LaTeX equations. Position the cursor within the comment line where you want the LaTeX equation to appear, and then select **Cell > Insert Text Markup > LaTeX Inline Math**.

For details see Inline LaTeX Math Equations.

MATLAB Notebook Menu Labels

Menu labels for MATLAB Notebook are changed. To ensure your menu options are up to date, and match those presented in the documentation, run the following command:

```
notebook -setup
```

MathWorks recommends you run this command whenever you install a new version of MATLAB.

Mathematics

New Function rng

The new rng function controls the random number generator used by rand, randi, and randn. For a demonstration, see this instructional video.

rng is the recommended alternative to former rand and randn input syntaxes `'seed'`, `'state'` and `'twister'`. rng is a simpler alternative to the `RandStream` class. To use rng, see the documentation on Updating Your Random Number Generator Syntax.

New Function ichol

The new ichol function performs incomplete Cholesky factorization and is the recommended alternative to `cholinc`.

New Option for gammainc

The new options `gammainc(X,A,'scaledlower')` and `gammainc(X,A,'scaledupper')` now return scaled versions of the incomplete gamma function gammainc.

Performance Enhancement

- Matrix transpose
- Element-wise single precision functions
- Sparse matrix indexed assignment
- Many linear algebra functions
- Convolution for long vectors and large matrices with `conv` and `conv2`

Changes to qr

In R2010a and R2010b, the upper triangular output `R` from the full qr function was guaranteed to have real and nonnegative diagonal elements. In this release, the behavior reverts to that of R2009b and prior releases. That is, the diagonal of `R` may contain complex and negative elements, and will affect the unitary output `Q` correspondingly.

Compatibility Considerations

Since the QR factorization is not unique, these different results are still correct. Ensure that your code does not depend on the values of the elements of the factors `Q` and `R`.

Functionality Being Removed

Function Name	What Happens When You Use This Function	Use This Function Instead	Compatibility Considerations
`bessel`	Errors	besselj	Replace all instances of `bessel` with besselj.
`intwarning`	Errors	None	Remove all instances of `intwarning` from your code.
`luinc`	Warns	ilu	Replace instances of `luinc` with ilu.
`cholinc(X,'inf')`	Warns	None	Remove all instances of `cholinc(X,'inf')` from your code.
All other syntaxes of `cholinc` except `cholinc(X,'inf')`	Warns	ichol	Replace these instances of `cholinc` with ichol.
`RandnAlg` property of RandStream class	Still runs	`NormalTransform` property of RandStream class	Replace all existing instances of `RandnAlg` with `NormalTransform`.
`setDefaultStream` method of `RandStream` class	Still runs	`setGlobalStream` method of `RandStream` class	Replace all existing instances of `RandStream.setDefaultStream` with RandStream.setGlobalStream.
`getDefaultStream` method of `RandStream` class	Still runs	`getGlobalStream` method of `RandStream` class	Replace all existing instances of `RandStream.getDefaultStream` with RandStream.getGlobalStream.

Function Name	What Happens When You Use This Function	Use This Function Instead	Compatibility Considerations
rand or randn with the `'seed'`, `'state'` or `'twister'` inputs	Still runs	rng	See Updating Your Random Number Generator Syntax in the MATLAB Mathematics documentation.

Programming

Regenerate P-code Files Built Before Version 7.5

To enable the MATLAB software to take advantage of significant performance improvements planned for a future release, MathWorks recommends that you begin to regenerate any P-code files you use that were generated prior to MATLAB 7.5 (release R2007b). P-code files generated in releases earlier than R2007b will not run in this future release with the new performance features enabled.

Compatibility Considerations

Using MATLAB 7.5 or later, rebuild any P-code files that you expect to need in the future.

VideoWriter Supports Motion JPEG 2000 Files

`VideoWriter` now creates Motion JPEG 2000 (`.mj2`) files, which support logging data with these features:

- Multi-byte precision, such as 10, 12, or 16 bits
- Signed data values
- Lossless compression

For more information, see the VideoWriter reference page.

audioplayer and audiorecorder Support Device Selection on All Platforms

`audioplayer` and `audiorecorder` now allow you to specify the input or output device on all supported platforms. In previous releases, you could only specify devices on Microsoft Windows systems.

Compatibility Considerations

- `audioplayer`, `audiorecorder`, and `audiodevinfo` now use different technology for enumerating system input and output devices. If your existing code specifies a

device ID (other than the default, `-1`), check whether you need to change the ID. To determine your device IDs, call `audiodevinfo`.

- `audioplayer` does not support overlapping playback. For example, in this code, the second call to `play` returns an error:

```
chirpData = load('chirp.mat');
chirpObj = audioplayer(chirpData.y, chirpData.Fs);

gongData = load('gong.mat');
gongObj = audioplayer(gongData.y, gongData.Fs);

play(chirpObj);
play(gongObj);
```

New Class Forms the Basis for Heterogeneous Hierarchies

The `matlab.mixin.Heterogeneous` class enables you to form heterogeneous arrays containing instances of classes derived from this class. You can create heterogeneous hierarchies of both handle and value classes.

New Class Provides the Basis for Customizable Handle Object Copy Method

The `matlab.mixin.Copyable` class enables you to define handle classes that inherit a copy method whose behavior you can modify in subclasses.

MATLAB Meta-Classes Can Now Form Heterogeneous Arrays

All MATLAB meta-classes are now defined in a heterogeneous hierarchy, which enables the formation of heterogeneous arrays of meta-class objects. This capability enables functions like `findobj` and `findprop` to return heterogeneous arrays containing instances of meta-classes of different specific types. The hidden class `meta.MetaData` forms the root of the heterogeneous hierarchy.

New High-Level NetCDF Functions

MATLAB now includes several new functions that provide a high-level interface to NetCDF files. These functions let you read and write to NetCDF files, without having to use the programming paradigm required by the low-level functions in the netCDF

package. Using these functions, you can create a new NetCDF file based on the schema of an existing file, convert files between NetCDF formats, and create a new NetCDF file by merging together two existing NetCDF files.

- nccreate — Create variable in NetCDF file
- ncdisp — Display contents of netCDF file
- ncinfo — Return information about netCDF file
- ncread — Read data and attributes from netCDF file
- ncreadatt — Read global attribute or attribute associated with variable from netCDF file.
- ncwrite — Write data to netCDF file
- ncwriteatt — Write attribute to netCDF file
- ncwriteschema — Add netCDF schema definitions to a NetCDF file

New High-Level HDF5 Functions

MATLAB now includes several new high-level functions for working with HDF5 files. These functions let you read and write to HDF5 files, without having to use the programming paradigm required by the low-level functions in the HDF5 package.

- h5create — Create HDF5 data set
- h5disp — Display contents of HDF5 file
- h5info — Return information about HDF5 file
- h5read — Read data from HDF5 data set.
- h5readatt — Read attribute from HDF5 group or data set
- h5write — Write to HDF5 data set
- h5writeatt — Write HDF5 attribute to group or data set

Compatibility Considerations

The new high-level HDF5 functions have the following compatibility considerations with the existing high-level HDF5 functions.

- hdf5read is not recommended. Use h5read instead.
- hdf5write is not recommended. Use h5write instead.

- hdf5info is not recommended. Use h5info instead.

Two New Functions Added to CDFLIB Package

The MATLAB interface to the CDF library now includes the following new functions.

- cdflib.setFileBackward — Sets the backward compatibility mode. If the backward mode is set to `on`, files created can be read by clients using version 2.7 of the library
- cdflib.getFileBackward — Return the current backward compatibility mode setting.

HDF4 Functions Grouped into Packages

The MATLAB HDF4 low-level functions and HDF EOS functions are now grouped into three new packages:

- `matlab.io.hdf4.sd` — access to more than 50 functions in the HDF library SD interface
- `matlab.io.hdfeos.gd` — access to more than 50 functions in the HDF-EOS library grid interface
- `matlab.io.hdfeos.sw` — access to more than 50 functions in the HDF-EOS library swath interface

For example, MATLAB previously included one function, `hdfsd`, that you used to call all the routines in the HDF library SD interface. Now, the new package `matlab.io.hdf4.sd` contains many individual functions that correspond to routines in the HDF SD Interface C library.

FITSREAD Function Now Supports Data Subsetting

The MATLAB fitsread function now supports data subsetting by using several new options: `PixelRegion`, `TableColumns`, and `TableRows`.

Unrecognized Name Warning Changed to Error

In earlier releases of MATLAB, a statement in which an unrecognized (e.g., misspelled) name immediately follows a function name and dot (`.`) results in a warning instead of an error. Here is the format this type of statement:

```
functionname.unrecognizedname
```

The following example shows such a statement case and part of the resulting warning:

```
simulink.badname

Warning: Direct access of structure fields returned by a
function call (e.g., call to simulink) is not allowed.  ...
```

In addition to being misleading, handling this case with merely a warning also allows the function (`simulink`, in this case) to execute, regardless of the fact that the name to the right of the dot is invalid.

In MATLAB version 7.12 (R2011a), this type of statement throws an error. The text of the error message is:

```
simulink.badname

??? Undefined variable "simulink" or function "simulink.badname".
```

Compatibility Considerations

Any such statements in your code that have only generated a warning message in past releases will now throw an error. Any such error that is not caught and handled appropriately will terminate the function in which it occurs. MathWorks recommends that you replace the unrecognized name to the right of the dot with the correct name.

This change also affects the output of the lasterror function if you attempt to access any field value (`message`, `identifier`, or `stack`) directly. Earlier versions of MATLAB generate a warning:

```
lasterror.message
Warning: Direct access of structure fields returned by a
function call (e.g., call to lasterror) is not allowed.  ...
```

This and future versions throw an error:

```
lasterror.message
??? Undefined variable "lasterror" or function "lasterror.message".
```

Regular Expressions Support Zero-Length Matching

Regular expressions in MATLAB now support successful zero-length matching. See the documentation on Empty Match mode in the description of Command Options for the regexp function.

Growing Arrays Is Faster

This release improves the performance of growing an array in the trailing dimension if that array has not been preallocated.

Error Checking Improved

MATLAB provides more effective error checking and returns new error messages in the following cases.

Nonstatic Method

A reference to a class method using the class name is valid only in cases where the method is static. Therefore, code of the form:

```
ClassName.ordinaryMethod
```

Where `ordinaryMethod` is not a static method of the class `ClassName`, previously returned the error: `MATLAB:class:InvalidStaticMethod`.

This code now returns the error:
`MATLAB:subscripting:classHasNoPropertyOrMethod`

Nonexistent Method Name

A reference to a nonexistent method name:

```
obj.badName
```

Where `badName` is not a method defined by the class of `obj`, previously returned the error: `MATLAB:noSuchMethodOrField`.

This code now returns the error:
`MATLAB:subscripting:classHasNoPropertyOrMethod`.

Compatibility Considerations

Code that checks for the specific errors previously returned must be updated to check for the new errors.

Functions and Function Elements Being Removed

Function or Function Element Name	What Happens When You Use the Function or Element?	Use This Instead	Compatibility Considerations
`audioplayer(BufferLength)`	No effect (no-op)		Remove all instances of `BufferLength`.
`audioplayer(NumberOfBuffers)`	No effect (no-op)		Remove all instances of `NumberOfBuffers`.
hdf5read	Still runs	h5read	Replace with preferred function.
hdf5write	Still runs	h5write	Replace with preferred function.
hdf5info	Still runs	h5info	Replace with referred function.

Graphics and 3-D Visualization

Plot Catalog with a New Look, More Plots, and Diagnostics

The Plot Catalog GUI now offers a wider variety of plotting options. Its appearance and operation closely resemble the Plot Selector, with short descriptions of each graph type and a search box. It also categorizes graphs by type and by toolbox, and lets you designate personal Favorites (the top menu category). Access the Plot Catalog from the Plotting Tools Figure Palette, the bottom of the Plot Selector menu in the Workspace Browser, the Variable Editor, and various context menus. For a demonstration, see this instructional video.

This table lists the differences and similarities between the Plot Catalog and the Plot Selector.

	Plot Catalog	Plot Selector	Both
Window Characteristics	Opens in its own window, which persists until you close it	Opens in a pop-up window that closes after it loses focus	Show icons for plot types with descriptions and Favorites.
Help for Plot Types	Displays partial help listings from reference pages in a resizeable pane within its window	Opens a popup help window with reference information when you hover	Provide the same help content and a **More Help** hyperlink to the Help Browser.
Plot Creation and Validation	Provides a field in which you can type variable names or expressions to plot	Plots variables you select in the Workspace browser or Variable Editor	Validate input variables and display diagnoses of incorrect or insufficient inputs.
Plot Destination	Current figure or new figure	Current figure	Create figure, if none exists

Previously, the Plot Catalog did not validate input data. If you provided incorrect inputs for a plotting function, it attempted to use them, resulting in an incorrect plot or errors. Now, the Plot Catalog provides the same diagnostics as the Plot Selector. When you type workspace variable names into the Variables box on top, the tool validates variable types,

sizes, and ordering. If validation fails, the Plot Selector provides a diagnostic message in its Help pane and does not let you run the plotting function.

In addition to MATLAB plots, the Plot Catalog offers the same set of choices as the Plot Selector. The choices include most types of plots from the following toolboxes (if installed):

- Control System Toolbox™
- Curve Fitting Toolbox™
- DSP System Toolbox™
- Financial Toolbox™
- Image Processing Toolbox
- Mapping Toolbox™
- Signal Processing Toolbox™
- Statistics Toolbox™
- System Identification Toolbox™

For more information about the Plot Catalog, see Selecting a Graph from the Plot Catalog. For information on the Plot Selector, see “Enhanced Plot Selector Simplifies Data Display” on page 14-24.

Creating Graphical User Interfaces (GUIs)

Do not Repopulate Menus on the Mac from Inside Their Callbacks

In R2011a, figures display their menus on the Mac screen menubar instead of across the top of figure windows. Prior to R2011a, GUIs could create dynamic menus using callbacks that completely deleted, and then repopulated the contents of menus. But on a Mac, running a GUI with a menu whose callback changes all items in this manner can result in the display of a blank menu (no items). The unexpected behavior only happens when a `uimenu` callback deletes all submenus (child uimenu components) and repopulates the menu with a new set of items at the time the menu opens.

In R2011a, on the Mac platform only, `uimenu` callbacks are no longer able to replace all submenus during menu selection. Note that repopulating some (not all) menu items does not create this issue. However, it is not good programming practice to remove and insert menu items within a menu callback routine.

For more information, see “MATLAB Menus Display at the Top of the Apple Mac Screen” on page 8-2.

Compatibility Considerations

If you have a uimenu callback that *repopulates all of its menu items* when you open that menu, you must change that code if you want it to work on a Mac. For example, your code might be able to remove and install submenus from outside of the callback that handles the menu. The callback can also rename, disable, hide, and show submenus instead of deleting them and creating new ones. The Mac is the only platform impacted by this incompatibility. Menus on Microsoft Windows, Unix, and Linux continue to behave as they did in previous releases.

Functions and Function Elements Being Removed

Function or Function Element Name	What Happens When You Use the Function or Element?	Use This Instead	Compatibility Considerations
`pagesetupdlg`	Warns	`printpreview`	printpreview provides all functionality and more, except for some unit choices. Update any GUIs that call `pagesetupdlg` to do page setup.

External Interfaces/API

Changes to Compiler Support

New Compiler Support

MATLAB Version 7.12 (R2011a) supports these new compilers for building MEX-files:

Linux (64- and 32-Bit) Platforms

- GNU gfortran 4.3.x

Apple Mac 64-Bit Platforms

- Apple Xcode 3.2 with gcc 4.2.x

Compiler Support To Be Phased Out

Support for the following compilers will be discontinued in a future release, at which time new versions will be supported. For an up-to-date list of supported compilers, see the Supported and Compatible Compilers Web page.

Microsoft Windows 32-Bit Platforms

- Microsoft Visual Studio 2005 SP1
- Visual C++ 6.0

Windows 64-Bit Platforms

- Microsoft Visual Studio 2005 SP1

Discontinued Compiler Support

MATLAB no longer supports the following compilers:

Windows (64- and 32-Bit) Platforms

- Microsoft Visual Studio 2008 (9.0) Express
- Intel Visual Fortran Version 10.1

Linux (64- and 32-Bit) Platforms

- GNU g95 0.90

Mac 64-Bit Platforms

- Xcode 3.1 with gcc 4.0.1

Compatibility Considerations

To ensure continued support for building your MEX-files, consider upgrading to another supported compiler. For an up-to-date list of supported compilers, see the Supported and Compatible Compilers Web page.

Changes to Shared Library Compiler Support

In MATLAB Version 7.12 (R2011a), you can use the loadlibrary command with any supported compiler. For an up-to-date list of supported compilers, see the Supported and Compatible Compilers Web page.

Compatibility Considerations

You must run `mex -setup` before using the `loadlibrary` command. For information about selecting a compiler, see Selecting a Compiler on Windows Platforms or Selecting a Compiler on UNIX Platforms.

New Support for Features in Microsoft .NET Framework

Support for .NET System.Enum Objects

MATLAB exposes .NET enumerations as native .NET classes:

- Support for non-`Int32` underlying types
- Support for bit-wise bitand, bitnot, bitor, and bitxor operators
- Support for invocation of `System.Enum` methods, such as the `HasFlag` method in Framework Version 4.0
- Support for comparison and binary operators `eq`, `ne`, `ge`, `gt`, `le`, and `lt` on enumeration types

See .NET Enumerations in MATLAB.

Compatibility Considerations

MATLAB displays an error when you use the enumeration command to return arrays of .NET enumeration objects. To read enumeration members into MATLAB arrays, see Refer to a .NET Enumeration Member.

MATLAB enumerations no longer inherit from the MATLAB `int32` class.

You cannot create arrays of .NET enumeration objects. For example, if you type:

```
a = [EnumTest.Colors.Red EnumTest.Colors.Blue]
```

MATLAB displays:

```
??? Array formation and indexing are not allowed on .NET objects.
```

To combine members of an enumeration into a MATLAB variable, see Combining Enumerations into a Single MATLAB Variable.

Support for Asynchronous .NET Delegate Callback Handling

You can use delegates to call a synchronous method asynchronously. See Calling a .NET Method Asynchronously.

R2010bSP2

Version: 7.11.2

Bug Fixes

R2010bSP1

Version: 7.11.1

Bug Fixes

R2010b

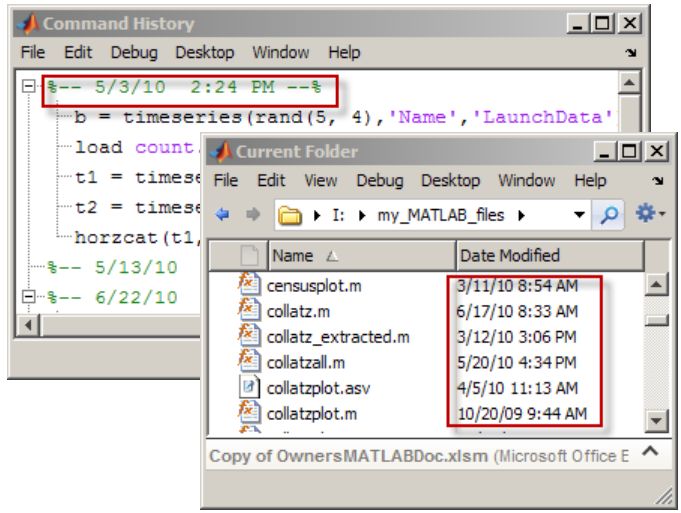
Desktop Tools and Development Environment

Desktop

New desktop features and changes introduced in Version 7.11 (R2010b) are:

Ability to Customize the Date Format

You can now customize the date format that the Current Folder browser and the Command History window use to display dates, such as the dates indicated here:



Previously, the date format in both of these tools was M/d/yy (for instance, 6/4/90 for June 4, 1990) and you could not change it. Now, both tools use the operating system short date format. For details, see Customizing the Column Display.

Keyboard Shortcuts Preferences Integrated with File Exchange

MATLAB now integrates keyboard shortcuts preferences with File Exchange. You can access File Exchange directly from the Keyboard Shortcuts Preferences panel and find:

- MATLAB keyboard shortcuts available in Version 7.9 (R2009a) and earlier releases
- Keyboard shortcuts sets created by other MATLAB users

For details, see Download Keyboard Shortcut Settings Files from File Exchange .

MATLAB Provides Enumeration Template

MATLAB provides a template for classes that define enumerations. Enumeration classes enable you to define a fixed set of names representing a particular type of value. Choose **File > New > Enumeration**. For more information, see Enumerations.

Help Browser

New Language Preference for Help Browser

If your system displays documentation in Japanese, you can use the **Language** panel in the Help Preferences dialog to instruct the Help browser to display documentation in English instead of Japanese. You can toggle between languages with the preference panel, enabling you to revert to Japanese at any time. The option changes the language used in the Help Browser and for GUI context-sensitive help, but it does *not* affect the language appearing in menus or elsewhere in products.

For example, the `help` command still provides help in Japanese after you switch the Help browser to English. The **Language** preference is available only when the system locale is Japanese and the Japanese documentation is installed. If the documentation for a product is not translated, the Help Browser displays the English documentation for it no matter how you set the preference.

Accessing Product Documentation in Japanese

MathWorks usually provides Japanese translations of product documentation about 2 months after new versions of products first ship. When you install the new version of a product, the *previous version* of translated documentation is installed in most cases.

To read the most current documentation for your release, switch the Help browser to English. After the translations of the latest documentation are available, you can download and install them in your products. At that time, you can find a link to

download the latest translated documentation on the Japanese documentation start page.

Managing Files

New features and changes introduced in Version 7.11 (R2010b) are:

Ability to View Zip File Contents in Current Folder Browser

Using the Current Folder browser, you can now view the files in a zip file without having to extract them. This feature enables you to:

- Confirm the contents of a newly created zip file
- Selectively open items from a zip file
- Add or remove files from the zip file

For details, see Creating and Managing Zip File Archives or watch the Zip File Browsing video demo.

Details Panel of Current Folder Provides Preview of Graphic Files

Now, when you select a `JPEG`, `JPG`, `BMP`, `WBMP`, `PNG`, or `GIF` image in the Current Folder browser, the Details panel displays a thumbnail of the image and lists its width and height in pixels. For more information, see Viewing File Details Without Opening Files or watch the File Preview Enhancements video demo.

Current Folder Browser Indicates Whether File Is Modified in Editor

Now, if you modify a file in the Editor, but have not yet saved the changes, the Current Folder browser indicates the file state. An asterisk (*) appears next to the name of such a file in the Current Folder browser. In addition, when you select such a file in the Current Folder browser, the detail panel reflects the modified file, not the file saved on disk. This feature is useful when you are creating zip files and want to be sure that all files included

in the archive are saved and up-to-date. For details see, Viewing File Details Without Opening Files or watch the File Preview Enhancements video demo.

Compare Zip Files and Folders

You can now compare any combinations of zip files, folders, and Simulink Manifests with the Comparison Tool. Right-click files or folders in the Current Folder browser and choose **Compare Selected Files/Folders**, or **Compare Against > Choose**.

For details, see Comparing Folders and Zip Files.

Enhanced Comparison Tool

The Comparison Tool now provides the following capabilities:

- Select what type of comparison to run from a list of possible comparison options, e.g., text, binary or XML comparison.
- Enhanced MAT-file comparisons now include size, data type and change summary.
- New option to ignore whitespace changes in text comparisons.

For details, see Comparing Files and Folders.

Editing and Debugging MATLAB Code

New Desktop features and changes introduced in Version 7.11 (R2010b) are:

Ability to Save File to Backup Without Closing That File

You now can save a file that is open in the Editor to a backup file. Select **File > Save Backup**, and then specify a backup file name. The file that is active in the Editor when you perform the save operation remains active. If changes to the active file prove problematic, you can use the backup file to return to a known state or to compare with the active file. This comparison can help you determine where errors exist.

For information on other save options, see Save Files.

Enhanced Comment Wrapping

The enhanced ability to wrap MATLAB comments includes:

- Wrapping an entire block of comments by selecting **File > Wrap Comments**.

  There is no need to select the block of comments first. For details see Wrap Comments Manually.
- Specifying where you want column counting to begin.

  For example, if you indent comments, you can specify that you want the maximum width of comments to be 75 columns from the start of a comment, rather than from the start of a line. To set **Comment formatting** preferences, select **File > Preferences > Editor/Debugger > Language**.

Variable and Function Highlighting

The Editor now presents variables that are not local variables in a teal blue color, by default. Also by default, when you click a function or local variable, the Editor highlights it and all other references to it in a sky blue color. As demonstrated in the Variable and Subfunction Highlighting video demo, this feature makes it easier to:

- Find unintentional reuses of a variable within a nested function

  For details, see Avoid Variable and Function Scoping Problems.
- Navigate through a file from function to function reference or variable to variable reference

  For details, see Navigate to a Specific Location.
- Search a file for a particular function or variable name

  For details, see Find and Replace Functions or Variables in the Current File.

To disable or change the colors that the Editor uses for variable and function highlighting, see Use Automatic Function and Variable Highlighting.

Options for Setting Current Folder and Search Path Available from Editor Context Menu

When multiple documents are open and docked in the Editor, you can right-click the document tab, and then from the context menu choose:

- **Change Current Folder to** *folder-name*
- **Add** *folder-name* **to Search Path** or **Remove** *folder-name* **from Search Path**, respectively
- **Locate on Disk**

  This option locates the document in your operating system file browser. It is not available on Linux platforms.
- **Copy Full Path to Clipboard**

Open As Text Option

You can use the **File > Open as Text** option to open a file in the Editor as a text file, even if the file type is associated with another application or tool. This feature is useful, for example, if you import a tab-delimited data file (`.dat`) into the workspace, and then you find you want to add a data point. For details, see Open Existing Files.

Mathematics

64-Bit Integer Arithmetic

Core MATLAB arithmetic functions now support `int64` and `uint64` classes natively. Functions added are plus (+), minus (–), uminus (–), times (.*), rdivide (./), ldivide (.\), power (.^), rem, mod, bitcmp, any, all, sum, diff, colon (:), sign, accumarray, and bsxfun.

New Utility Functions: isrow, iscolumn, ismatrix

New functions isrow, iscolumn, and ismatrix provide basic information about inputs.

Output Option for Point Distances in DelaunayTri/nearestNeighbor Method

The two-output form of DelaunayTri/nearestNeighbor returns the corresponding Euclidean distances between the query points and their nearest neighbors.

Changes to convhull and delaunay Functions

The functions convhull and delaunay now support 3-D input in either multiple vector or multicolumn matrix format. In addition, the `simplify` option for convhull provides the option of removing vertices that do not contribute to the area or volume of the convex hull.

Performance Enhancements

- Three-Output Form of svd

  The three-output form of svd displays significantly enhanced performance.

- Sparse Column Assignment

  The assignment of elements into multiple columns of a MATLAB sparse matrix displays significantly enhanced performance.

- Trigonometric Functions

All degree-based trigonometric functions (sind, cosd, tand, cotd, secd, cscd) and their inverses (asind, acosd, atand, acotd, asecd, acscd) display significantly enhanced performance.

Functions Being Removed

Function Name	What Happens When You Use This Function	Use This Function Instead	Compatibility Considerations
`bessel`	Warns	besselj	Replace all instances of `bessel` with besselj.
`erfcore`	Errors	erf, erfc, erfcx, erfinv, or erfcinv	See the function reference pages for the individual functions.
`intwarning`	Errors	None	Remove all instances of `intwarning` from your code.

optimset Errors for Optimization Toolbox Options

If you do not have an Optimization Toolbox™ license, and you set an optimset option for a solver that is only available in Optimization Toolbox, optimset errors. Previously, optimset would warn, not error, and ignore the option.

Compatibility Considerations

Change your code to set only those options that apply to your solver: fminbnd, fminsearch, fzero, or lsqnonneg.

atan Warning Being Removed

When you use `atan(1i)` and `atan(-1i)` you no longer get a warning.

Compatibility Considerations

Remove instances of the warning ID `MATLAB:atan:singularity` from your code.

Data Analysis

Arrays of Time Series Objects Supported

MATLAB now enables you to create arrays of timeseries objects. In Version 7.10 (R2010a) and before, `timeseries` objects behaved as arrays, but some array behavior was overridden.

Compatibility Considerations

Note: It is likely that this change is important to you *only* if you write code that uses `timeseries` objects.

In Version 7.11 (R2010b), all of the overridden behaviors for the functions listed in the table that follows are removed. The behavior of these functions on `timeseries` objects is the built-in behavior for arrays of objects. Think of a `timeseries` object as a single object that you can concatenate into an array of objects. Do not think of each `timeseries` itself as an array.

Functions Being Modified

Function or Function Element Name	What Happens When You Use the Function or Element?	Use This Instead	Compatibility Considerations
horzcat	No error or warning	Not applicable	You can now use `horzcat` with timeseries objects.

`isempty (timeseries)`	No error or warning	`timeseries.Length==0`	Replace code such as:
			`isempty(ts)`
			with:
			`ts.Length == 0`

`length (timeseries)`	No error or warning	`timeseries.Length` property	Replace code such as:
			`length(ts)`
			with:
			`ts.Length`

Function or Function Element Name	What Happens When You Use the Function or Element?	Use This Instead	Compatibility Considerations
`size (timeseries)`	No error or warning	`[timeseries.Length 1]`	Replace code such as: `size(ts)` with: `[ts.Length 1]`
subsref	No error or warning	getsamples	Replace code such as: `t3 = t1(1:3);` with: `getsamples(t1,1:3);`
`vertcat (timeseries)`	May return `Index exceeds matrix dimension` error	append	Replace code such as: `t3 = [t1;t2];` with `t3 = append(t1,t2);`

Example of timeseries Object Concatenation

Suppose you concatenate the following two timeseries objects:

```
t1 = timeseries(ones(5,1),0:4);
t2 = timeseries(zeros(5,1),5:9);
```

In R2010a, the following syntax created a single `timeseries` object, `t3`, with 10 samples spanning the time range 0-9. The first 5 samples derived from `t1` and the last 5 samples derived from `t2`. The size of the `timeseries`, `t3`, was `[10 1]` and its length was 10, because `t3` has 10 samples.

```
t3 = [t1;t2]
t3 = vertcat(t1,t2)
```

Now, the same syntax creates a 2x1 array comprising the two `timeseries` objects `t1` and `t2`. Its size is `[2 1]`, and its length is 2 because the array has two rows, each a single `timeseries`.

Programming

arrayfun Accepts Array of Objects

The arrayfun function now accepts an array of objects as an input. The output can also be a scalar object, as long as the `UniformOutput` flag is set to `false`.

Comparing Object Arrays that Contain NaNs

In previous versions of MATLAB, calling isequalwithequalnans to compare identical arrays of objects that contain one or more NaN (Not a Number) values incorrectly returned `false`. Similarly, calling isequal to compare identical arrays of objects that contain NaNs incorrectly returned `true`.

In MATLAB version 7.11, these functions return the expected values (`true` for `isequalwithequalnans`, and `false` for `isequal`) when used to compare object arrays.

Compatibility Considerations

Program code that compares object arrays that may contain NaNs should be examined to ensure correct behavior.

Functions isa and islogical Now Consistent for Objects

In earlier releases of MATLAB, the following statements returned different results for objects with a base class of `logical`:

```
isa(obj, 'logical')
islogical(obj)
```

In MATLAB version 7.11, both of these statements return `true` if the class of `obj` is derived from `logical`, and `false` otherwise.

Compatibility Considerations

If you have program code that calls either isa or islogical on objects with a base class of logical, it is advisable to verify that this new behavior does not adversely affect the execution of the program.

New Enumeration Classes

MATLAB provides support for classes that define enumerations. Enumeration classes enable you to define a fixed set of names representing a particular type of value. See Enumerations for more information.

Use this link to watch a video that introduces enumerations: Play demo

New Functionality for Writing Video Files

The new `VideoWriter` function allows you to create AVI files on all platforms. As an improvement over the `avifile` function, `VideoWriter` can create files larger than 2 GB. For more information, watch this video demo or see the VideoWriter reference page.

mmreader Renamed VideoReader

For consistency with the new `VideoWriter` class, the `mmreader` class is now called `VideoReader`. A new function, `VideoReader`, replaces the `mmreader` function. `VideoReader` and `mmreader` return identical `VideoReader` objects for the same input file.

Compatibility Considerations

Replace all instances of `mmreader` with `VideoReader`. The two functions use identical syntax. When requesting help or documentation for methods from the command line, or calling a static method such as `getFileFormats`, use the new `VideoReader` class name (such as `doc VideoReader/read`). You do not need to change any calls to the `read`, `get`, or `set` methods.

Previously, the R2010a Release Notes recommended replacing instances of `aviinfo` with `mmreader` and `get`, and instances of `aviread` with `mmreader` and `read`. Instead, replace `aviinfo` with `VideoReader` and `get`, and replace `aviread` with `VideoReader` and `read`.

New HDF4 Functions

The MATLAB interface to the grid functions in the HDF-EOS C library, `hdfgd`, now includes three new syntaxes. These syntaxes convert grid coordinates to longitude and latitude values.

- `ij2ll`
- `ll2ij`
- `rs2ll`

To view the help for these functions, use the `help` command at the MATLAB prompt, as in the following example:

```
help hdfgd
```

New HDF5 Low-Level Functions

The MATLAB interfaces to the HDF5 C library now include the following new functions.

- `H5A.open_by_idx`
- `H5A.open_by_name`
- `H5D.get_access_plist`
- `H5I.is_valid`
- `H5L.copy`
- `H5L.get_name_by_idx`
- `H5P.set_chunk_cache`
- `H5P.get_chunk_cache`
- `H5P.set_copy_object`
- `H5P.get_copy_object`

To view the help for these functions, use the `help` command at the MATLAB prompt, as in the following example:

```
help H5P.get_copy_object
```

New netCDF Functions

The MATLAB interface to the netCDF C library now includes the following new functions.

- `netcdf.inqFormat`
- `netcdf.inqUnlimDims`
- `netcdf.defGrp`
- `netcdf.inqNcid`

- `netcdf.inqGrps`
- `netcdf.inqVarIDs`
- `netcdf.inqDimIDs`
- `netcdf.inqGrpName`
- `netcdf.inqGrpNameFull`
- `netcdf.inqGrpParent`
- `netcdf.defVarChunking`
- `netcdf.defVarDeflate`
- `netcdf.defVarFill`
- `netcdf.defVarFletcher32`
- `netcdf.inqVarChunking`
- `netcdf.inqVarDeflate`
- `netcdf.inqVarFill`
- `netcdf.inqVarFletcher32`

To view the help for these functions, use the `help` command at the MATLAB prompt, as in the following example:

```
help netcdf.inqVarFletcher32
```

Upgrades to Scientific File Format Libraries

The following table lists upgrades to scientific file format libraries used by MATLAB.

Library	Version Upgrade
netCDF	3.6.2 to 4.0.1

New Examples in Command Line Help

The MATLAB command line help for the HDF5, netCDF, and CDF low-level functions now includes hundreds of new examples.

imread and imwrite Can Now Handle N-channel J2C JPEG 2000 Files

You can now use the `imread` and `imwrite` functions with n-channel J2C JPEG 2000 files.

J2C files are raw JPEG2000 codestreams that usually have the file extension `.j2c` or `.j2k`. J2C files don't contain color space, color palette, capture resolution, display resolution and vender specific information.

In previous releases, if you use `imread` to read a 4-channel J2C file, you receive a warning that some channels may be ignored and `imread` returns a 3-channel image. Now, `imread` returns a 4-channel image and does not issue a warning.

If you try to write a 4-channel J2C file in previous releases, `imwrite` issues an error. In this release, `imwrite` creates the 4-channel J2C file.

csvread and csvwrite Will Not Be Removed

The R2010a Release Notes originally stated that `csvread` and `csvwrite` would be removed in a future release. As of R2010b, there are no plans to remove these functions.

sprintf and fprintf Print Null Characters in Strings

In previous releases, calls to `sprintf` or `fprintf` that printed strings using the `%s` qualifier prematurely terminated strings that contained null characters. For example, this code

```
sprintf('%s', ['foo' 0 'bar'])
```

returned

```
ans =
foo
```

The same code now returns

```
ans =
foo bar
```

Compatibility Considerations

If you do not want to print the entire contents of strings that contain null characters, test the string for these characters (for example, using the `isstrprop` function).

Functions and Function Elements Being Removed

Function or Function Element Name	What Happens When You Use the Function or Element?	Use This Instead	Compatibility Considerations
fileparts return argument 4 (file version)	Warns, returns empty fourth argument	No alternative; file versions are unsupported	Call `fileparts` with three return arguments: `[path_name file_name file_extension]`
`mmreader`	Still runs, creates a `VideoReader` object	VideoReader	Replace all existing instances of `mmreader` with `VideoReader`.
`wavplay`	Still runs	audioplayer and play	Replace all existing instances of `wavplay` with `audioplayer` and `play`.
`wavrecord`	Still runs	audiorecorder and record	Replace all existing instances of `wavrecord` with `audiorecorder` and `record`.
`wk1finfo`	Warns		Remove all instances of `wk1finfo`. Get information about Excel spreadsheets with xlsfinfo.
`wk1read`	Warns		Remove all instances of `wk1read`. Read Excel spreadsheets with xlsread.
`wk1write`	Warns		Remove all instances of `wk1write`. Write to Excel spreadsheets with xlswrite.

MATLAB Did Not Pass struct to loadobj When Property Was Deleted

If you save an object having a property value equal to the property's default value, and then that property is removed from the class definition, MATLAB did not detect the fact that the property was missing. In this case, MATLAB did not build a `struct` to pass to loadobj. MATLAB now returns a `struct` in this case.

Compatibility Considerations

If you rely on the following situation *not* returning a `struct` to `loadobj`, you need to update your code:

- You saved an object using the save command
- This object had a property that was set to the default value defined by the class at the time of saving
- he property set to its default value at save time is subsequently removed from the class definition
- You load the property and expect the load operation to return an object (it should return a `struct` because the class definition has changed in a way the load cannot create the object).
- Your `loadobj` method is not prepared to use the returned `struct` to create an object

You must implement a `loadobj` method to recreate the object using the current class definition.

See Saving and Loading Objects for more information.

Graphics and 3-D Visualization

Print –dmfile and printdmfile Issue Deprecation Warnings

The –`dmfile` `print` command option and the `printdmfile` function now issue a deprecation warning. Both option and function will be removed in a subsequent release. The –`dmfile` option and the `printdmfile` function were used by GUIDE.

Compatibility Considerations

Remove any use of `print –dmfile` or `printdmfile` from your code.

The saveas 'mmat' option Issues a Deprecation Warning

The `saveas` function now issues a deprecation warning if you use the `'mmat'` format option. This option will be removed in a subsequent release. This option was used with GUIDE.

Compatibility Considerations

Remove any use of the `mmat` option with `saveas` from your code.

The movie Function is No Longer a Built-in Function

The movie function is no longer a built-in function. Therefore, you cannot call `movie` from the builtin function. `movie` syntax remains the same.

Compatibility Considerations

Remove any use of the `builtin` function to call the `movie` function from your code.

Creating Graphical User Interfaces (GUIs)

Functions and Function Elements Being Removed

The `uitabgroup` and `uitab` GUI components are undocumented features that provide tabbed panels to GUIs that you create programmatically. The current method of calling these functions and some of their properties will change in a future release. MathWorks has never supported their use. However, if your MATLAB code includes these functions, read the following information:

Function or Function Element Name	What Happens When You Use the Function or Element?	Use This Instead	Compatibility Considerations
`uitabgroup('v0',...)` `uitab('v0',...)`	Warns	`uitabgroup(...)` `uitab(...)`	Using the `'v0'` argument generates a warning, but the object is created.
`uitabgroup` property `SelectionChangeFcn`	Not recommended	`SelectionChange Callback`	Event data for the callback is still a struct with fields `OldValue` and `NewValue` but those values are now `uitab` handles, not `uitab` indexes.
`uitabgroup` property `SelectedIndex`	Not recommended	`SelectedTab`	Both properties still exist, but replace instances of `SelectedIndex` (an integer tab index) with `SelectedTab` (a handle to a `uitab` object).
`uitabgroup` property `BackgroundColor`	Not recommended	no replacement	`uitabgroup` now uses the default `uicontrol` background color.
`uitabgroup` event `SelectionChanged`	Errors	`SelectionChange` event	Event name changed to be consistent with other components. Event data is still a struct with fields `OldValue` and `NewValue` but those values are now `uitab` handles, not `uitab` indexes.

You should update any code that you maintain which uses the undocumented `uitabgroup` and `uitab` functions to conform to the new standards, as described in the uitab and uitabgroup Migration Document. Existing code for these functions is unlikely to work in R2010b going forward.

External Interfaces/API

Changes to Compiler Support

New Compiler Support

MATLAB Version 7.11 (R2010b) supports these new compilers for building MEX-files:

Microsoft Windows (64- and 32-Bit) Platforms

- Microsoft Visual Studio 2010 (10.0) Professional
- Microsoft Visual Studio 2010 (10.0) Express

Linux (64- and 32-Bit) Platforms

- GNU gcc Version 4.3.x

Compiler Support to Be Phased Out

Support for the following compilers will be discontinued in a future release, at which time new versions will be supported. For an up-to-date list of supported compilers, see the Supported and Compatible Compilers Web page.

Windows (64- and 32-Bit) Platforms

- Intel Visual Fortran Version 10.1
- Microsoft Visual Studio 2005 SP1

Discontinued Compiler Support

MATLAB no longer supports the following compilers:

Windows (64- and 32-Bit) Platforms

- Intel C++ Version 9.1

Linux (64- and 32-Bit) Platforms

- GNU gcc Version 4.2.3

Apple Macintosh (32-Bit) Platforms

- Apple Xcode 3.1 (gcc / g++ Version 4.0.1)
- GNU gfortran Version 4.2.2

Compatibility Considerations

To ensure continued support for building your MEX-files, consider upgrading to another supported compiler. For an up-to-date list of supported compilers, see the Supported and Compatible Compilers Web page.

-largeArrayDims Option to MEX Will Become Default in Next Release of MATLAB

In the next release of MATLAB, the mex command will change to use the large-array-handling API by default. This means the `-largeArrayDims` option will become the default.

You do not need to make changes to build and run MEX-files with MATLAB Version 7.11 (R2010b).

Compatibility Considerations

Source for MEX-files built on 64-bit platforms must be updated in order to successfully build and run with the next release of MATLAB. Review your source MEX-files and `mex` build scripts. For information about migrating your MEX-files to use the large-array-handling API, see Upgrading MEX-Files to Use 64-Bit API.

MEX Function -argcheck Option Removed

Function Option Name	What Happens When you use the Option?	Compatibility Considerations
mex `-argcheck` option	Errors	Remove all existing instances of `-argcheck` in build scripts.

The `-argcheck` option was a strategy for identifying memory corruption caused by a MEX-file. When you understand the likely causes of memory corruption, you can take preventative measures in your code or use your debugger to identify the source of errors.

If a binary MEX-file causes a segmentation violation or assertion, it means the MEX-file attempted to access protected, read-only, or unallocated memory.

These types of programming errors are sometimes difficult to track down. Segmentation violations do not always occur at the same point as the logical errors that cause them. If a program writes data to an unintended section of memory, an error might not occur until the program reads and interprets the corrupted data. Consequently, a segmentation violation can occur after the MEX-file finishes executing.

One cause of memory corruption is to pass a null pointer to a function. To check for this condition, add code in your MEX-file to check for invalid arguments to MEX Library and MX Matrix Library API functions.

To troubleshoot problems of this nature, run MATLAB within a debugging environment. For more information, see Debugging C/C++ Language MEX-Files or Debugging Fortran Source MEX-Files.

MEX Function -inline Option Removed

Function Option Name	What Happens When you use the Option?	Compatibility Considerations
mex `-inline` option	Errors	Remove all existing instances of `-inline` in build scripts.

New Support for Features in Microsoft .NET Framework

MATLAB supports:

- Using .NET Delegates in MATLAB.
- Accessing Microsoft Office Applications with .NET.
- Using `System.Nullable`, described in How MATLAB Handles System.Nullable.
- Calling constructors or NET.invokeGenericMethod with `ref` or `out` type arguments.
- Calling methods with default arguments (`System.Reflection.Missing.Value` is supported.)

New COM Data Type Support

Support for the following COM data type has been added. See Handling COM Data in MATLAB Software for a description of supported data types.

- `VT_I8` — `signed int64`
- `VT_UI8` — `unsigned int64`
- Unsigned integer

R2010a

Desktop Tools and Development Environment

Desktop New Features Video for R2010a

For an overview of the major new features in the MATLAB Desktop Tools and Development Environment area, watch this video demo.

Startup and Shutdown

AUTOMOUNT_MAP Environment Variable No Longer Used by MATLAB

On UNIX systems, MATLAB no longer uses the `AUTOMOUNT_MAP` environment variable, the path prefix map for automounting. Previously, MATLAB set the value of the variable in `.matlab7.rc`, but you could override this setting using the `AUTOMOUNT_MAP` environment variable.

Desktop

New Desktop features and changes introduced in Version 7.10 (R2010a) are:

Enhancements for Managing Keyboard Shortcuts

Keyboard shortcuts now provide the following:

Ability to List a Set of Keyboard Shortcuts in a Text Editor

You can now list all of the keyboard shortcuts for a given set in a text editor. For details, see List All Keyboard Shortcuts in a Set.

Ability to Remove Keyboard Shortcuts Sets

You can now remove keyboard shortcut sets that you created or imported. For details, see Delete a Set of Keyboard Shortcuts.

Ability to Compare Sets of Keyboard Shortcuts

You can now compare one set of keyboard shortcuts to another. This enables you, for example, to see what is different between the current Windows default keyboard shortcuts and those that were the defaults in MATLAB Version 7.8 (R2009a) and earlier. For details, see Compare Sets of Keyboard Shortcuts.

Method for Accessing M-Lint Preferences and the M-Lint Report

The **M-Lint Preferences** is now renamed the **Code Analyzer Preference**. Therefore, to set preferences for analyzing MATLAB code, select **File > Preferences > Code Analyzer**. Similarly, the M-Lint Report is now renamed the Code Analyzer Report. To access this report, click the Actions button  in the Current Folder browser, and then select **Reports > Code Analyzer Report**. The names of the `mlint` and `mlintrpt` functions are unchanged.

Preference for Java Heap Memory

A new preference setting allows you to adjust the amount of memory that MATLAB allocates for storing Java objects. Insufficient memory for these objects results in `OutOfMemory` errors. For more information, see Java Heap Memory Preferences.

Compatibility Considerations

Technical Solution 1-18I2C specifies a procedure for creating a text file named `java.opts` to set the Java heap size. Do not use both the new MATLAB preference and a `java.opts` file to adjust the heap size.

Help Browser

For a demonstration of previous enhancements to the Help browser, watch the Help Browser Enhancements video.

Improved Instructions and Examples for Adding Help and Demos to the Help Browser

The MATLAB documentation now includes more details about how to display your own HTML help and demos. In particular, the documentation clarifies procedures for setting up files and folders and provides three new XML file templates that you can inspect, copy, and modify.

The documentation also adds a new complete working example, called the Upslope Area Toolbox. Documentation for the toolbox includes a getting started guide, user guide, function reference pages, and release notes. The documentation set is generated entirely from MATLAB script files (also provided), using the `publish` command. The example folder includes all program files for the toolbox. However, certain routines require Image Processing Toolbox for full functionality.

You can also find the Upslope Area Toolbox on the MATLAB Central File Exchange. For more information about techniques and algorithms the toolbox uses, see the set of articles about it on this MATLAB Central blog: http://blogs.mathworks.com/steve/category/upslope-area.

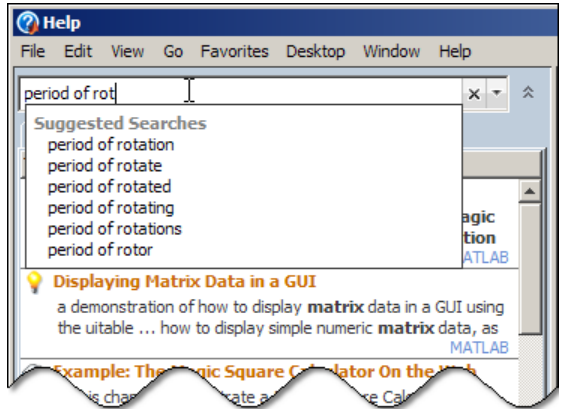
Find the updated documentation in Create Help and Demos. For details, XML templates, and examples, see these subsections:

- Add Documentation to the Help Browser
- Add Demos to the Help Browser

You can view the Upslope Toolbox documentation example in the Help browser now by clicking here, which places its folder on the search path.

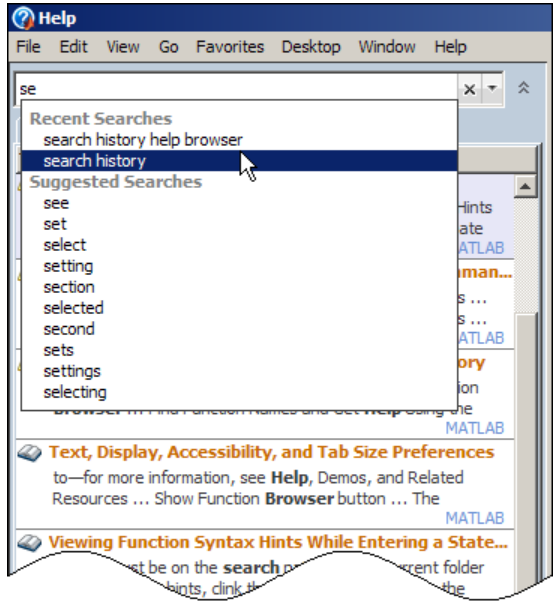
New Search Hints

The Help Browser now provides suggestions for search terms as you type in the search field. These hints save you time and can suggest words and phrases you might not have thought of but that can be useful in making a search. The hints display as a drop-down menu that changes its contents as you type. The following illustration shows the appearance of the menu.



Search History Persists Between Sessions

The Help browser can now recall searches you have made across MATLAB sessions. Your search history and search hints display as separate lists in a drop-down pane as you type in the search field. Select **Show Search History** from the pop-up menu to the right of the search field, or press the down arrow key in the search field when it is empty. The following illustration shows search history and search hints as they appear when you enter search terms.



Hide Search Results Previews

You can now choose not to see the one-line descriptions of search results called *previews*. To toggle all previews off, right-click in the Search Results pane of the Help Navigator and choose **Hide Previews**. The search results listing is smaller so you can scan it quickly when you hide previews. To see previews again, right-click in the Search Results pane, and choose **Show Previews**.

docopt Function Removed

This release obsoletes the `docopt` function, which specified which system Web browser to use on UNIX platforms. This function has been removed. You can instead specify a default system browser in your preferences. For more information, see “New System Browser Preference Instead of docopt.m for MATLAB on UNIX Platforms” on page 15-2.

Compatibility Considerations

If you call `doctopt` in your startup file or some other program file, remove the call to avoid receiving an error.

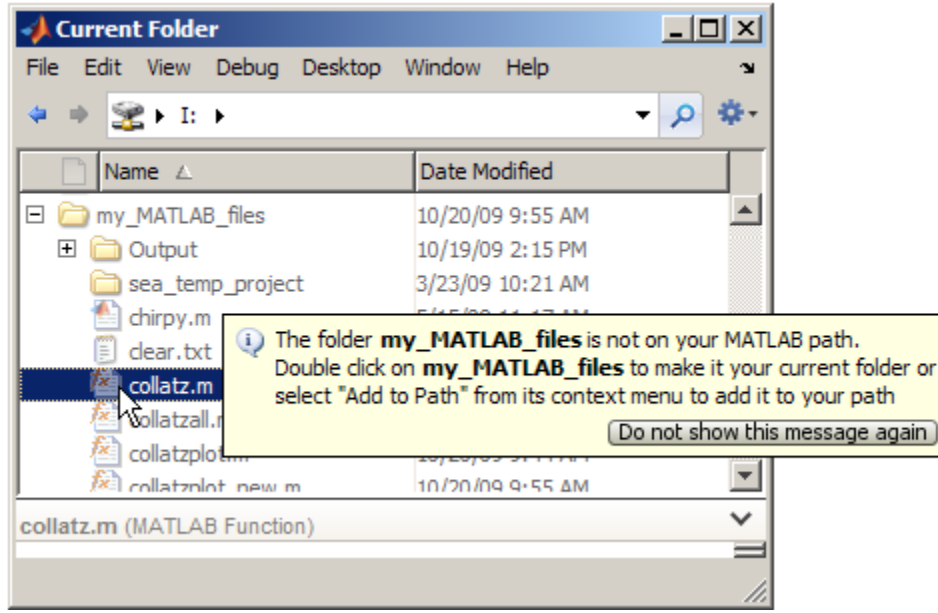
Managing Files

Create and Expand Zipped Archives from Current Folder Browser

Effective this release, the capability to create and expand, and archive `.zip` files has been added to the Current Folder browser. Select files and folders in the Current Folder browser, then right click, and choose **Create Zip File** to create an archive. The archive appears in the current folder. You can expand an archive in the same way. For details, see Creating and Managing Zip File Archives.

Visual Aids for Identifying Files Inaccessible to MATLAB

This release changes the default behavior of the Current Folder browser. The Current Folder browser now dims files that are inaccessible to MATLAB to indicate their unavailability. A tooltip provides an explanation.



You can also customize or disable this feature. For more information, see Preferences for the Current Folder Browser and Viewing Files and Folders on the Search Path.

Ability to Remove Folders and Subfolders from the Path Using the Current Folder Browser

You can now remove folders from the MATLAB path as follows:

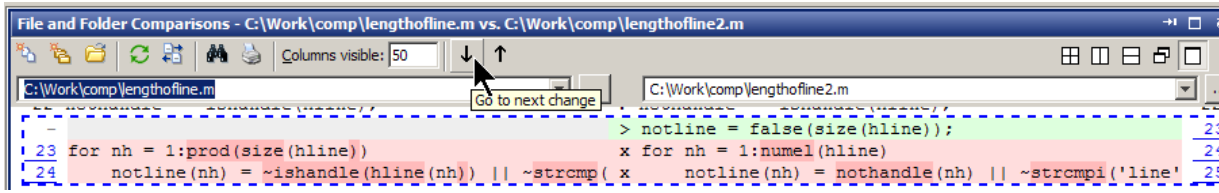
1. In the Current Folder browser, right-click the folder you want to remove from the path.
2. From the context menu, select **Remove from Path**, and then select **Selected Folders** or **Selected Folder and Subfolders**.

If the default preferences for the Current Folder **Path indication** option are set, the folders you removed now appear dimmed in the Current Folder browser. For more information, see Preferences for the Current Folder Browser and Removing Folders from the Search Path.

Enhancements for File and Folder Comparisons

This release enhances the File and Folder Comparisons tool to improve the usability of comparisons between text files and folders.

File comparisons now highlight changes within lines and provide new toolbar buttons for stepping through differences, as the following figure shows.



Folder comparisons now provides results that are sortable by name, type, size, timestamp, or change summary. Click column headers to sort the table. Click **compare** links to explore differences between files and folders. The following example shows results sorted by **Type**.

Comparing folder curvefitting vs. folder curvefitting2

Left file list	Contents of folder C:\Work\comp\curvefitting
Right file list	Contents of folder C:\Work\comp\curvefitting2

Click on a column header to sort the table

		In left list (folder curvefitting)		In right list (folder curvefitting2)		
Type ▾	**File Name**	**Size (bytes)**	**Last Modified Date**	**Size (bytes)**	**Last Modified Date**	**Change Summary**
ASV file	lengthofline.asv (open)	*(not in this list)*		2403	2009-11-16 15:24:45	*added*
HTML file	manifest_report.html (open)	2410	2008-11-27 14:24:50	*(not in this list)*		*removed*
MAT file	splinetool.mat (open)	1720	2001-08-20 18:14:12	*(not in this list)*		*removed*
M file	csapidem.m (open)	*(not in this list)*		15038	2009-01-08 13:56:28	*added*
M file	lengthofline.m (open: left| right)	2405	2009-11-12 16:56:52	2408	2009-11-16 15:25:14	contents changed (compare)
TXT file	kdm.txt (open)	*(not in this list)*		1367	2008-10-03 11:13:37	*added*
XML file	report.xml (open)	*(not in this list)*		4868	2008-09-11 17:53:32	*added*
folder	curvefit	-	2009-11-16 15:18:38	-	2009-11-16 15:20:08	contents changed (compare)
folder	cftoolgui	-	2009-11-16 15:18:38	-	2009-11-16 15:20:04	contents changed (compare)
folder	demosearch	-	2009-11-16 15:19:13	-	2009-11-16 15:19:44	*identical*

Compare in a new window

When you compare subfolders with many files, now the tool continues analysis in the background, reporting progress. You can now skip items or cancel as you review this analysis.

For more information, see Comparing Files and Folders.

Editing and Debugging MATLAB Code

New features and changes introduced in Version 7.10 (R2010a) are:

Tab Completion for Local Variables and Functions

The MATLAB Editor now supports tab completion for local variable and subfunction names within MATLAB program files. For more information, see Complete Names in the Editor Using the Tab Key.

Toolbar Buttons for Stepping Through Code Cells Without Evaluating Code

This release provides two New Cell Mode toolbar buttons, Next Cell  and Previous Cell  . You can use these buttons to step through cells in the Editor without evaluating the code within the cells. By default, these buttons do not appear on the Cell Mode toolbar.

For more information, see Navigate Among Code Cells in a File.

Mathematics

New Multithreading Capability

MATLAB's new multithreading capability now includes:

- fft for long vectors
- The two-input form of conv2
- Integer conversion and arithmetic

Performance Improvements

MATLAB features significant performance improvements for:

- Sparse matrix indexed assignment and deletion
- bvp4c and bvp5c for sparse problems
- sortrows
- mrdivide (/)
- convn
- histc

Changes To qr

The factorization routine in `qr` produces an upper triangular matrix, `R`. Now this matrix always contains real and nonnegative diagonal elements. In previous releases, the diagonal of `R` could contain complex and negative elements.

Compatibility Considerations

`R` now contains only real, nonnegative diagonal elements. The `QR` factorization is not unique, so the answer is still correct.

Change in Indexing for Sparse Matrix Input

Now subscripted reference into a sparse matrix always returns a sparse matrix. In previous versions of MATLAB, using a double scalar to index into a sparse matrix resulted in full scalar output.

Compatibility Considerations

Scalars extricated from sparse matrices are no longer full. If you previously used the output with a function that does not support sparse input, you now need to update your code. For a list of the functions that do not support sparse matrices, see Functions That Do Not Support Sparse Matrices in the MATLAB Mathematics documentation.

Improved Error Checking for Sparse Functions

Functions that erroneously accepted N-D arrays and returned reshaped sparse 2-D outputs no longer accept full N-D inputs and now return an error. The functions affected are:

- spones, spfun, sprand, sprandn, and sprandsym
- cat, horzcat, and vertcat

Also, binary functions that formerly accepted both full N-D inputs along with sparse 2-D inputs and warned about the reshape, now return an error. For example, `ones(2,2,2).*sparse(3)` formerly returned a sparse 2-by-4 matrix along with a warning about the reshape. This code now returns an error.

Computational Geometry Functions Being Changed

The computational geometry functions delaunay, convhull, griddata, voronoi, delaunay3, griddata3 no longer use the `options` arguments.

Compatibility Considerations

Use of the `options` arguments to delaunay, convhull, griddata, and voronoi now throws a warning.

Computational Geometry Functions Being Removed

The functions `griddata3`, `delaunay3`, `dsearch`, `tsearch` will be removed in a future release. Use of these functions now throws a warning.

Function Name	Use This Method Instead
`griddata3`	TriScatteredInterp

Function Name	Use This Method Instead
`dsearch`	DelaunayTri/nearestNeighbor
`tsearch`	DelaunayTri/pointLocation
`delaunay3`	DelaunayTri

Compatibility Considerations

Update your code to use the DelaunayTri and TriScatteredInterp computational geometry classes .

lsqnonneg No Longer Uses Optional Starting Point Input

The lsqnonneg solver no longer uses the optional input `x0` as a starting point.

Compatibility Considerations

If you give a starting point `x0`, MATLAB issues a warning. Also, `lsqnonneg` ignores `x0`, and instead uses a starting point of a vector of zeros.

Function erfcore Removed

Support for the `erfcore` function has been removed. Use of `erfcore` now results in the following error message:

```
ERFCORE will be removed in a future release.
Use ERF, ERFC, ERFCX, ERFINV, or ERFCINV instead.
```

Compatibility Considerations

Replace `erfcore` with erf, erfc, erfcx, erfinv, or erfcinv. See the function reference pages for the individual error functions.

Integer Warning Messages Removed

These warning messages for integer math and conversion have been removed:

- `MATLAB:intConvertNaN`

- `MATLAB:intConvertNonIntVal`
- `MATLAB:intConvertOverflow`
- `MATLAB:intMathOverflow`

Compatibility Considerations

The warning messages for integer math and conversion are no longer available. Remove these warning IDs from your code.

Function intwarning Being Removed

The `intwarning` function will be removed in a future release. Use of `intwarning` now throws a warning:

```
Warning: All four integer warnings are removed.
INTWARNING will be removed in a future release.
```

Compatibility Considerations

The warnings previously thrown when you used `intwarning` on are now removed. Remove all instances of `intwarning` from your code.

atan Warning Being Removed

The warning thrown when you use `atan(i)` and `atan(-i)` will be removed. Currently, the warning message is:

```
Warning: Singularity in ATAN. This warning will be
removed in a future release.Consider using DBSTOP
IF NANINF when debugging.
```

Compatibility Considerations

Remove instances of the warning ID `MATLAB:atan:singularity` from your code.

nextpow2 Returns Output the Same Size As Input

In previous releases, `nextpow2` with a vector input `v` returned `nextpow2(length(v))`. Now `nextpow2(v)` returns a vector the same size as the input.

Compatibility Considerations

If you require the old behavior, replace instances of `nextpow2` with `nextpow2(length(v))`.

Math Libraries Not Available to Build MEX-Files with Compaq Visual Fortran

MATLAB no longer provides the `libdflapack.dll` and `libdfblas.dll` math libraries for building Fortran MEX-files with a Compaq® Visual Fortran compiler.

Compatibility Considerations

If you distribute MEX-files that call BLAS or LAPACK functions built on a Compaq Visual Fortran compiler, you must also distribute the `libdflapack.dll` and `libdfblas.dll` library files.

If MATLAB displays errors when loading a MEX-file and the Dependency Walker indicates missing `libdflapack.dll` and/or `libdfblas.dll` libraries, contact the MEX-file vendor for copies of the libraries. The third-party product, Dependency Walker, is available from the Web site `http://www.dependencywalker.com/`.

Data Analysis

Operations on Timeseries Objects Sometimes Warn About the isTimeFirst Property

In a future release, the timeseries Boolean property `isTimeFirst` will behave differently. If the value is `true`, this property indicates that the time vector is the *first* dimension of a time series. If the value is `false`, the time vector is the *last* dimension of a time series. In this release, you receive a warning that certain settings of `isTimeFirst` will not be valid in a future release. The warning indicates that for 3-D or N-D data, time must be the *last* dimension, while 2-D time series data can have time as the first dimension.

Compatibility Considerations

You can receive a warning when:

- Constructing a time series object with three or more dimensions without specifying a time vector. In a future release, MATLAB will calculate a value of the `isTimeFirst` property that can differ from the value it currently calculates.
- Changing the `data` of a time series to data of a different dimensionality. In a future release MATLAB will calculate a value of the `isTimeFirst` property that differs from the value it currently calculates.
- Directly setting the `isTimeFirst` property for a time series to a value that will be invalid in a future release.

The warning does not affect how you can use time series data in this release. It is intended to inform you that the behavior of `isTimeFirst` will change in a future release. If you receive a warning about `isTimeFirst`, you can use the `timeseries` method `migrateistimefirst` to correct the problem. For information type

```
help timeseries/migrateistimefirst
```

in the Command Window.

Time Series Time Vectors Can Now Contain Duplicate Sample Times

Starting with this release, a vector containing sample times can have duplicate times in contiguous positions. Previously, time vectors needed to increase monotonically and

duplicated time values generated errors. This condition is now relaxed, such that time vectors must be *nondecreasing*. Interpolation of time series can produce duplicated times if the input to the interpolation method contained duplicate time samples. For more information, see the timeseries reference documentation.

Programming

Subscripting Into Function Return Values

If you have a function, such as the following, that returns a struct array:

```
function structOut = getStruct
structOut = struct('fieldA', 5, 'fieldB', 10);
```

it is no longer valid to access fields of the structure by directly dot indexing the function's return value, as shown here:

```
getStruct.fieldA
```

Instead, you should first assign the returned structure to a variable, and then dot index the variable:

```
s = getStruct;
s.fieldA
ans =
     5
```

This type of field access has never been allowed within function code, and this change causes scripts and command line statements to be in agreement with function rules.

Compatibility Considerations

If you have scripts that employ this type of dot indexing, they will now throw an error. Replace this unsupported style of dot indexing with the style shown above.

New Constructor for Map Containers

The following command constructs an empty `containers.Map` object that uses a key type of `kType`, and a value type of `vType`:

```
M = containers.Map('KeyType', kType, 'ValueType', vType)
```

See the containers.Map function reference page for more information.

Function Handle Access to Private and Protected Methods

When creating a function handle inside a method of a class, the function is resolved using the permissions of that method. When MATLAB invokes the function handle,

it does so using the permissions of the class. This gives MATLAB the same access as the locations where the function handle was created, including access to private and protected methods accessible to that class.

See Obtaining Permissions from Class Methods for more information.

Listing Video File Formats Supported by mmreader

The new `mmreader.getFileFormats` method returns a list of the formats that `mmreader` supports. The list of formats is static for Windows and UNIX systems, but dynamic on Macintosh systems. For more information, see the getFileFormats reference page.

unzip Preserves Write Attribute of Files

In previous releases, for files archived using WinZip® software, the `unzip` function set the file write attribute of extracted files to read only. Starting with R2010a, `unzip` preserves the original attribute.

New Package Provides Access to low-level CDF API Routines

MATLAB has included high-level routines for accessing Common Data Format (CDF) files: `cdfread`, `cdfwrite`, and `cdfinfo`. However, the CDF API is so large, these functions cannot satisfy every need. To solve this, MATLAB now includes a new package, called `CDFlib`, that provides access to dozens of the functions in the CDF API. Using these low-level functions, you can create CDF files and populate them with variables and attributes using the CDF library, version 3.3.0.

Upgrades to Scientific File Format Libraries

The following table lists upgrades to several scientific file format libraries used by MATLAB.

Library	Version Upgrade
CDF	3.2.1 to 3.3.0
HDF5	1.8.1 to 1.8.3
HDF4	4.2.r1 to 4.2.r4

Library	Version Upgrade
HDF-EOS2	2.8 to 2.16
PNG	1.2.8 to 1.2.39

Tiff Class Enhancements

The Tiff class includes the following enhancements.

Tiff Class Now Excludes Reading OJPEG Format Image Data

You can no longer read YCbCr OJPEG ("old-style" JPEG compression) TIFF images. The Tiff object was not intended to work with OJPEG data but some TIFF files contain data in this format. Reading this data could cause LibTIFF to terminate.

Compatibility Considerations

To read OJPEG image data, use the `imread` function. Note that the `imread` function transforms the image data into the RGB colorspace.

Additional Reading and Writing Capabilities

The Tiff class includes the following additional capabilities:

- Writing image data that use 16-bit colormaps (palettes)
- Reading and writing `LogL` and `LogLUV` High Dynamic Range (HDR) images. To write a `LogL` or `LogLUV` image, you must use the new `SGILogDataFmt` property.

MATLAB Adds Support for Creating JPEG 2000 Files

You can now use the `imwrite` function to create a JPEG 2000 file and write data to the file. The `imwrite` function supports format-specific parameters that let you specify the mode, compression ratio, and other characteristics.

Sealed No Longer Listed as meta.property Class Property

While class properties do not have a `Sealed` attribute, the `meta.property` class listed `Sealed` as a property. `Sealed` is no longer listed as a `meta.property` class property.

Functions and Function Elements Being Removed

Function or Function Element Name	What Happens When You Use the Function or Element?	Use This Instead	Compatibility Considerations
`aviread`	Warns	`mmreader` and `read`	Replace all existing instances of `aviread` with `mmreader` and `read`.
`aviinfo`	Warns	`mmreader` and `get`	Replace all existing instances of `aviinfo` with `mmreader` and `get`.
`exifread`	Warns	imfinfo	Replace all existing instances of `exifread` with `imfinfo`
`fileparts`	Still runs, *`versn`* output is always empty (`''`)		Remove all references to a fourth output (*`versn`*) from fileparts.
`intwarning`	Warns	warning	If you use `intwarning`, you can implement the same warnings using the warning function.
`isstr`	Still runs	ischar	Replace all existing instances of `isstr` with `ischar`.
`mmreader.isPlatformSupported`	Warns, always returns `true`		For a platform-specific list of supported video file formats, use getFileFormats.
`setstr`	Still runs	char	Replace all existing instances of `setstr` with `char`.

Function or Function Element Name	What Happens When You Use the Function or Element?	Use This Instead	Compatibility Considerations
`str2mat`	Still runs	char	Replace all existing instances of `str2mat` with `char`.

`strread`	Still runs	textscan	Replace all existing instances of `strread` with `textscan`. For example, replace `[a,b,c] = strread(...)` with
			`C = textscan(...)`
			`[a,b,c] = deal(C{:})`
			Unlike `strread`, the `textscan` function converts numeric values to the specified data type, allowing preservation of integer types.

`strvcat`	Still runs	char	Replace all existing instances of `strvcat` with `char`. Unlike `strvcat`, the `char` function does *not* ignore empty strings.
`textread`	Still runs	textscan	Replace all existing instances of `textread` with `textscan`, similar to `strread`. Open and close files with `fopen` and `fclose`.
`wk1finfo`	Still runs		Remove all instances of `wk1finfo`. Get information about Excel spreadsheets with xlsfinfo.

Function or Function Element Name	What Happens When You Use the Function or Element?	Use This Instead	Compatibility Considerations
`wk1read`	Still runs		Remove all instances of `wk1read`. Read Excel spreadsheets with xlsread.
`wk1write`	Still runs		Remove all instances of `wk1write`. Write to Excel spreadsheets with xlswrite.

Graphics and 3-D Visualization

Plot Selector Supports Additional Toolboxes

The Plot Selector, which was upgraded in Version 7.9 (R2009b), now also generates plots for display functions in the following toolboxes:

- Curve Fitting Toolbox
- DSP System Toolbox
- Image Processing Toolbox
- Signal Processing Toolbox

The Plot Selector continues to support display functions for Control System Toolbox, Financial Toolbox, and Statistics Toolbox. You must have installed a toolbox to display data using its plotting functions. For more information about the Plot Selector, see “Enhanced Plot Selector Simplifies Data Display” on page 14-24.

External Interfaces/API

Changes to Compiler Support

New Compiler Support

MATLAB Version 7.10 (R2010a) supports these new compilers for building MEX-files:

Windows (32-Bit) Platforms

- Intel C++ Version 11.1
- Intel Visual Fortran Compiler Professional Edition for Windows Version 11.1, installed with Microsoft Visual Studio 2008 Shell. This product is bundled.
- Intel Visual Fortran Compiler Professional Edition for Windows Version 11.1 and Microsoft Visual Studio 2008 SP1 Professional Edition. Separate products.
- Open Watcom Version 1.8

Windows (64-Bit) Platforms

- Intel C++ Version 11.1
- Intel Visual Fortran Compiler Professional Edition for Windows Version 11.1, installed with Microsoft Visual Studio 2008 Shell. This product is bundled.
- Intel Visual Fortran Compiler Professional Edition for Windows Version 11.1 and Microsoft Visual Studio 2008 SP1 Professional Edition. Separate products.

Compiler Support to Be Phased Out

Support for the following compilers will be discontinued in a future release, at which time new versions will be supported. For an up-to-date list of supported compilers, see the Supported and Compatible Compilers Web page.

Windows (32-Bit) Platforms

- Intel Visual Fortran Version 10.1
- Intel C++ Version 9.1

Windows (64-Bit) Platforms

- Intel Visual Fortran Version 10.1
- Intel C++ Version 9.1

Discontinued Compiler Support

MATLAB no longer supports the following compilers:

Windows (32-Bit) Platforms

- Open Watcom Version 1.7
- Microsoft Visual Studio .NET 2003 Version 7.1

Solaris SPARC (64-Bit) Platforms

- Sun™ Studio 11 cc / CC Version 5.8
- Sun Studio 11 f90 Version 8.2

Compatibility Considerations

To ensure continued support for building your MEX-files, consider upgrading to another supported compiler. For an up-to-date list of supported compilers, see the Supported and Compatible Compilers Web page.

Changes to Libraries on Linux with Debian Systems

If you run MATLAB Version 7.10 (R2010a) on a Linux with Debian® system, you must use Debian 5, as indicated on the Platforms & Requirements Web page.

MEX-files created with MATLAB Version 7.2 (R2006a) or earlier depend on the `libstdc++.so.5` library. Debian 5 does not include this library. MATLAB running on Linux platforms with Debian 5 cannot load these pre-R2006a MEX-files.

Compatibility Considerations

You cannot run MEX-files created with MATLAB R2006a or earlier on Linux with Debian 5. Recompile these files on a system with Debian 5.

Math Libraries Not Available to Build MEX-Files with Compaq Visual Fortran

MATLAB no longer provides the `libdflapack.dll` and `libdfblas.dll` math libraries for building Fortran MEX-files with a Compaq Visual Fortran compiler.

Compatibility Considerations

If you distribute MEX-files that call BLAS or LAPACK functions built on a Compaq Visual Fortran compiler, you must also distribute the `libdflapack.dll` and `libdfblas.dll` library files.

If MATLAB displays errors when loading a MEX-file and the Dependency Walker indicates missing `libdflapack.dll` and/or `libdfblas.dll` libraries, contact the MEX-file vendor for copies of the libraries. The third-party product, Dependency Walker, is available from the Web site `http://www.dependencywalker.com/`.

Cannot Create MEX-Files with DLL File Extension

On Windows 32-bit platforms, support for MEX-files with a `.dll` file extension is being phased out. Use the `.mexw32` extension instead.

MATLAB Version 7.10 will continue to execute `.dll` MEX-files, but future versions of MATLAB will not support the `.dll` extension. You can no longer use the mex function `-output` switch to create MEX-files with a `.dll` extension. If you enter a command such as:

```
mex myfile.c -output newfile.dll
```

MATLAB creates the MEX-file `newfile.mexw32` and displays a warning message.

For more information about MEX-files with `.dll` extensions, see Running MEX-Files with .DLL File Extensions on Windows 32-bit Platforms.

Compatibility Considerations

Recompile MEX-files with `.dll` file extensions. Update MATLAB scripts or `makefiles` that explicitly specify `.dll` extensions for compiled MEX-files.

If you use MEX-files with a `.dll` extension from a third-party source, contact that vendor to get a recompiled version, referring to these release notes.

-largeArrayDims Option to MEX Will Become Default in Next Release of MATLAB

In the next release of MATLAB, the default mex command will change to use the large-array-handling API. This means the `-largeArrayDims` option will become the default.

You do not need to make changes to build and run MEX-files with MATLAB Version 7.10 (R2010a).

Compatibility Considerations

Source for MEX-files built on 64-bit platforms must be updated in order to successfully build and run with the next release of MATLAB. Review your source MEX-files and `mex` build scripts. For information about migrating your MEX-files to use the large-array-handling API, Upgrading MEX-Files to Use 64-Bit API.

R2009bSP1

Version: 7.9.1

Bug Fixes

New features and changes introduced in this version are as follows:

The version number of the MATLAB Compiler Runtime (MCR) in release R2009bSP1 is different from the MCR version number in release R2009b. For details about MCR version numbers and corresponding MATLAB releases, see http://www.mathworks.com/support/solutions/en/data/1-4GSNCF/?solution=1-4GSNCF.

R2009b

Desktop Tools and Development Environment

Startup and Shutdown

Changes to -nodisplay and -noFigureWindows Startup Options

When you start MATLAB using `-nodisplay` (UNIX) or `-noFigureWindows` (Microsoft Windows) startup options, running a built-in GUI (predefined dialog boxes) generates this warning:

```
This functionality is no longer supported under
the –nodisplay and -noFigureWindows startup options.
```

If the GUI is modal, MATLAB execution suspends, and you need to type **Ctrl+C** for the Command Window prompt to reappear.

This change affects predefined dialog boxes, such as `dialog`, `msgbox`, `printpreview`, `uigetfile`, `uisetcolor`, `waitfor`, `waitbar`, and more. For a list of predefined dialog boxes, see Predefined Dialog Boxes. For more information on startup options, see the matlab (UNIX) and matlab (Windows) reference pages, respectively.

In a future release, instead of a warning, MATLAB will generate an error when you use such a dialog box under the `-nodisplay` or `-noFigureWindows` options.

Compatibility Considerations

To avoid generating the warning, start MATLAB without the `-nodisplay` or `-noFigureWindows` startup options. To avoid warnings while continuing to use these startup options, remove the code that is now producing the warnings.

Changes to Memory Manager Startup Options

Version 7.9 removes support for the `-memmgr` and `-check_malloc` command line arguments and the `MATLAB_MEM_MGR` environment variable.

Compatibility Considerations

MATLAB ignores any memory manager options.

Desktop

New features and changes introduced in Version 7.9 (R2009b) are:

Ability to Customize Keyboard Shortcuts

MATLAB supports customizing keyboard shortcuts for desktop tools. Press combinations of keyboard keys to perform a desktop action, instead of using the mouse to select items from menus. For example, press **Ctrl+N** to open a blank file in the Editor.

The ability to customize keyboard shortcuts enables you to:

- Modify keyboard shortcuts across desktop tools on an action-by-action basis.
- Modify keyboard shortcuts to match other applications you use or to match your personal preferences.
- Share your keyboard shortcuts with others, or use shortcuts created by someone else.
- Create keyboard shortcuts for some actions that currently have no shortcut.
- Use your preferred keyboard shortcuts when you use MATLAB on a different system than you typically use.

To customize keyboard shortcuts, choose **File > Preferences > Keyboard > Shortcuts**.

MATLAB does not support customizing keyboard shortcuts for Figure Windows, dialog boxes, or toolboxes. For details, see Customize Keyboard Shortcuts. For a video demo, see Customizable Keyboard Shortcuts.

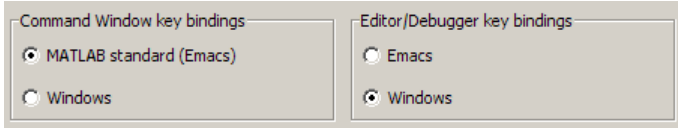
Compatibility Considerations

- The method for specifying keyboard shortcuts differs from previous versions.

  In previous versions, you chose **File > Preferences > Keyboard**, and then chose a set of key bindings. MATLAB no longer limits your choices to either platform-specific settings or Emacs settings.
- Some default keyboard shortcuts differ from the previous defaults.

  Defaults changed in Version 7.9 (R2009b) to be more consistent across the MATLAB desktop. Previously, you could specify keyboard shortcut preferences for the

Command Window and Editor/Debugger only. For instance, on Windows, default keyboard shortcut preferences appeared as follows:



To restore keyboard shortcuts that were the default before Version 7.9 (R2009b):

1. Choose **File > Preferences > Keyboard > Shortcuts**.
2. From the **Active settings** drop-down menu, choose **R2009a Windows Default Set**, **R2009a Macintosh Default Set**, or **R2009a UNIX Default Set**, depending on the platform on which MATLAB is installed.

Ability to Set Fonts Preferences for Extended M-Lint Messages and Function Browser

MATLAB now supports setting the font size and type for extended M-Lint messages and the function browser. In previous releases, changes to **HTML Proportional Text** affected only the display in the MATLAB Web browser, the Profiler, and Help topics. To set font preferences:

1. Choose **File > Preferences > Fonts > Custom**.
2. Under **Desktop tools**, select **HTML Proportional Text**, and then specify the font size and type you want to use.

For more information, see Fonts.

Save Files from MATLAB Web Browser

To save a file being displayed in the MATLAB Web Browser, select **File > Save As**.

Help Browser

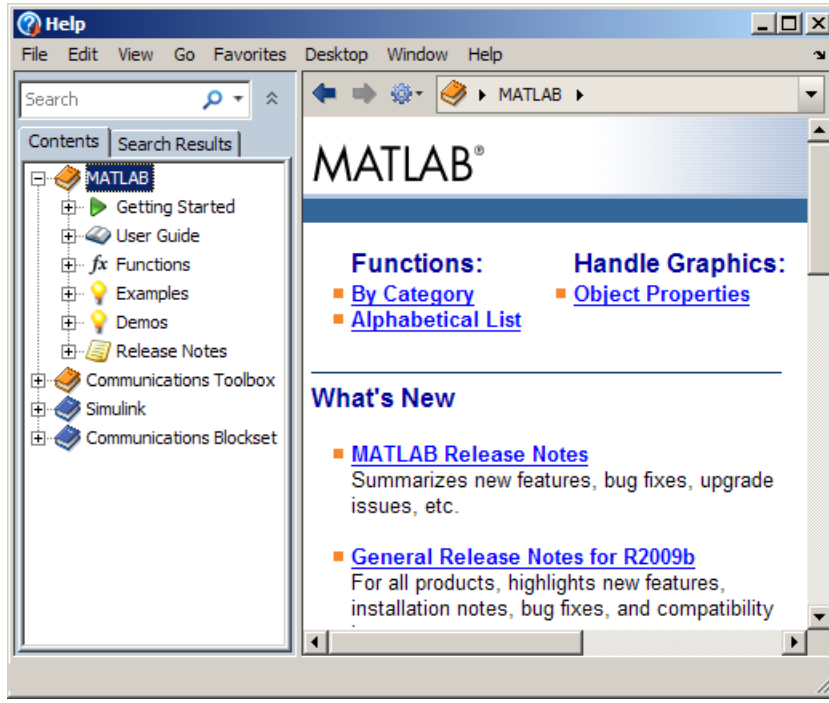
For a demonstration of enhancements to the Help browser, watch the Help Browser Enhancements video.

Improved Contents Listing

When you expand a product in the **Contents** pane, you now can view the following:

- Function and block names with brief descriptions — Expand **Functions** or **Blocks**, and then expand a category to see the list and descriptions. Select a function or block to view its reference page.

  If you provide your own HTML help files for use in the Help browser, you now can include **Functions** and **Blocks** entries for your toolbox.
- Demos — Expand **Demos**, and then select a demo from the list to view or run it.



Enhanced Presentation of Search Results

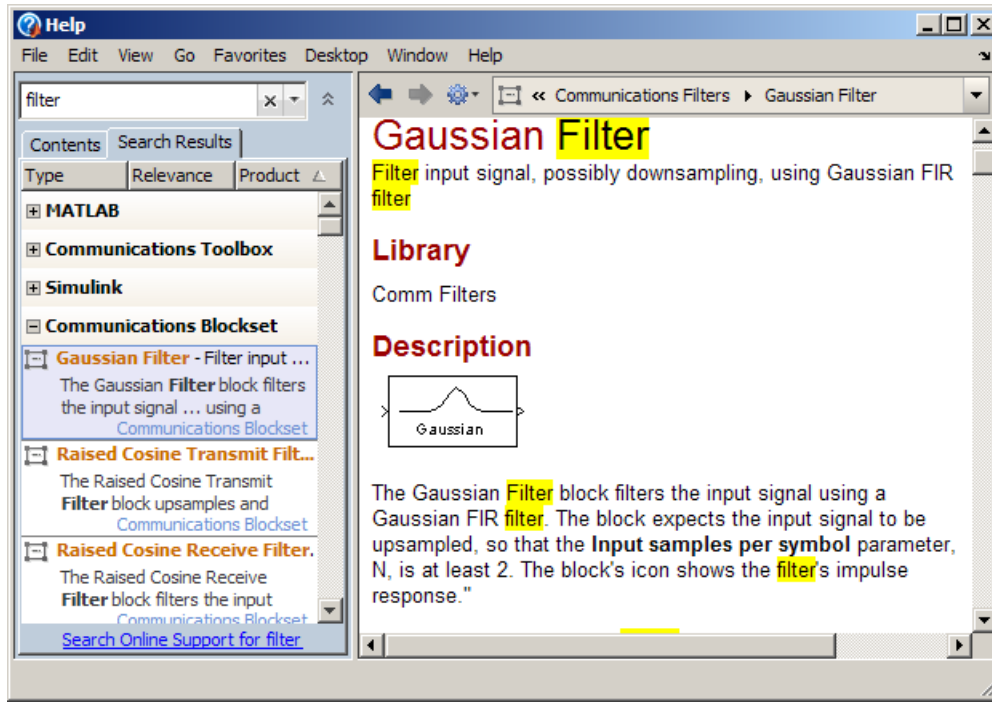
Each search result includes:

- A preview of where the search words were found within the page.
- An icon representing the type of document, such as a reference page or demo.

Use new sorting and grouping features to arrange results:

- Sort results by **Relevance**, in addition to sorting by **Type** and **Product**, which were available in previous versions.
- After sorting by **Type** or **Product**, you now can collapse and expand results for each type or product. To expand or collapse all groups, right-click in the results pane and select the option you want from the context menu.

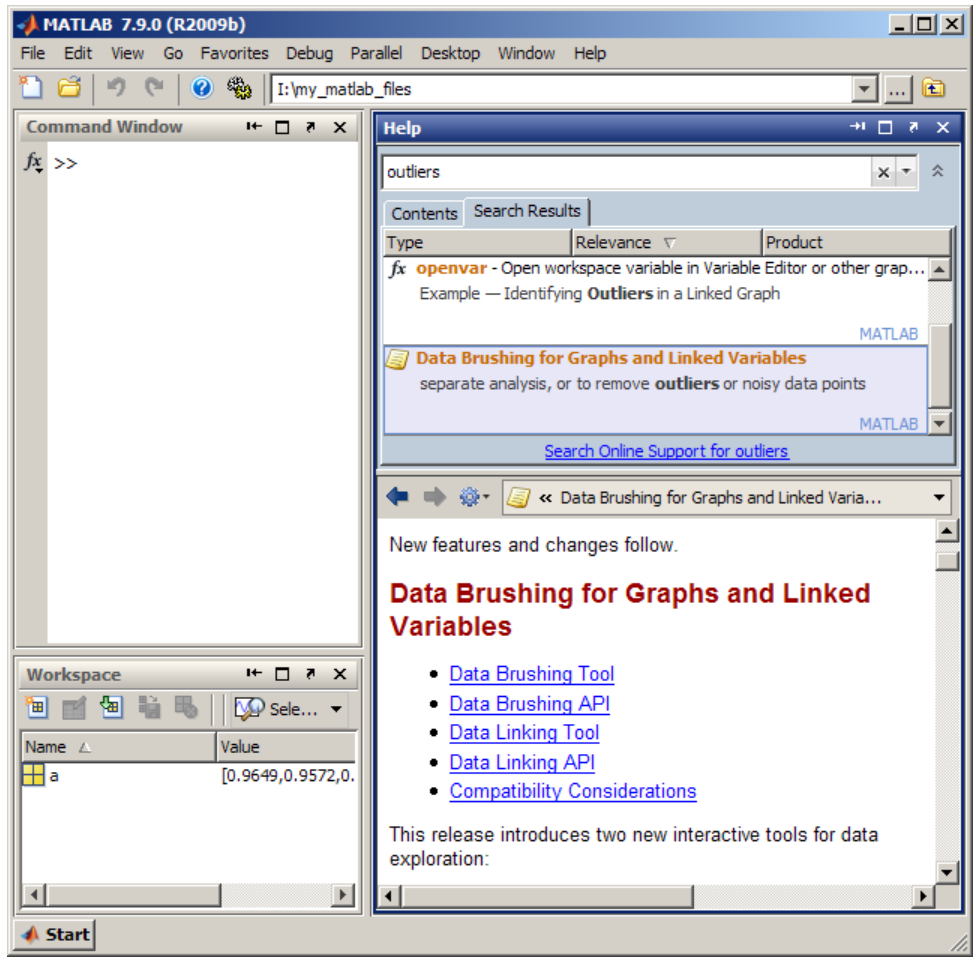
The following example of **Search Results** in the Help browser illustrates grouping, previews, and the block reference page icon.



Viewing Pages

- To see where the current page is located within the documentation, use the navigation bar at the top of the display pane. To go to another topic in the documentation, select an entry from the navigation bar.

- Use the new Actions button  on the display pane toolbar to access features such as Refresh, which clears the search results highlighting for the current page.
- Get links to reference pages for overloaded functions. When you run `doc foo`, if `foo` is an overloaded function, a message appears at the top of the display pane that provides links to the other `foo` reference pages.
- To use the Help browser alongside other tools, dock it in the desktop. When docked in a narrow area of the desktop, the pane for **Contents** and **Search Results** moves above the display pane.



Compatibility Considerations

The **Index** and **Demos** tabs are no longer in the Help browser:

- To find terms that were in the **Index**, use the search feature instead.
- To access demos for a product, go to the **Contents** pane and expand the **Demos** entry for the product.

The Actions button on the toolbar provides access to features for the displayed page. Previously, some of these features were available on individual toolbar buttons. To add individual toolbar buttons, right-click the toolbar and select **Customize**.

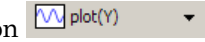
If you provide your own files for use in the Help browser:

- Remove the `helpindex.xml` entry from the `info.xml` file for your toolbox. MATLAB no longer supports the **Index**.
- Demos you add now appear in the **Contents** pane under `Other Demos`, which is after the entries for all MathWorks products.

For more information about the new Help browser features, see Help and Product Information.

Workspace and Variable Editor

Improved Workspace Plotting Tool

The Plot Selector button on 

the Workspace Browser and Variable Editor toolbars has a new look and added capabilities, options, and help. The tool now displays larger icons, includes many more graphing functions, summarizes each function, and displays pop-up windows with syntax descriptions (function hints). You can customize the tool by rearranging and categorizing functions, and by creating a list of "favorites".

For further details, see this video demo and the release note "Enhanced Plot Selector Simplifies Data Display" on page 14-24.

Managing Files

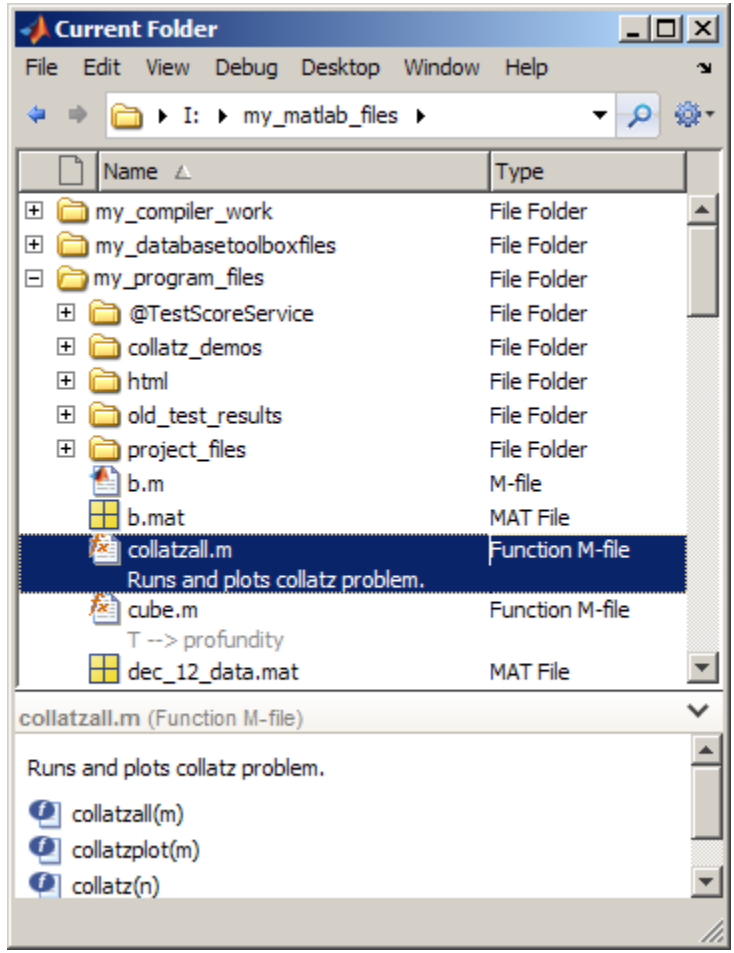
Enhanced Current Folder (Directory) Browser

The Current Directory browser is now the Current Folder browser.

For an overview of the enhancements, watch this video, Current Folder Browser Enhancements, or review the following summary:

- Use the expandable tree structure to view the contents of the current folder. Double-click a subfolder to make it the current folder.
- Display the file type in a column.
- Distinguish types of MATLAB program files using new icons:
  -  class
  -  function
  -  script
- View file descriptions below file names. To show or hide descriptions, select **View** > **Show** > **Description**.
- Display hidden files a new way:
  - On Windows platforms, MATLAB follows your Windows preference for hidden files.
  - On other platforms, MATLAB uses the new setting in Current Folder preferences.
- Find files and folders within the current folder and its subfolders:
  - To access the search feature, click the Search button  on the address bar, and then enter the string you want to find.
  - To clear the results and display all files and folders in the current folder, click the Clear button  .

The following illustration shows the new Current Folder browser, including the tree view, new icons for program files, and the search button.



Compatibility Considerations

- With the new tree structure, you now see all files, including those in subfolders. You might not be able to run the files you see in the subfolders. To run a file from a subfolder, the subfolder must be on the search path or the subfolder must become the current folder. In previous versions, you could only see files in the current folder, so you could run every file you could see.

- You now search for files by clicking the search button in the address bar. In previous versions, the search feature, referred to as the filter field, appeared if your preference was set to display it.
- File descriptions now appear below the file name. In previous versions, they displayed in a column.

For more information about the new features, see Managing Files in MATLAB.

File Exchange Desktop Tool — Find and Get Files Created by Other Users

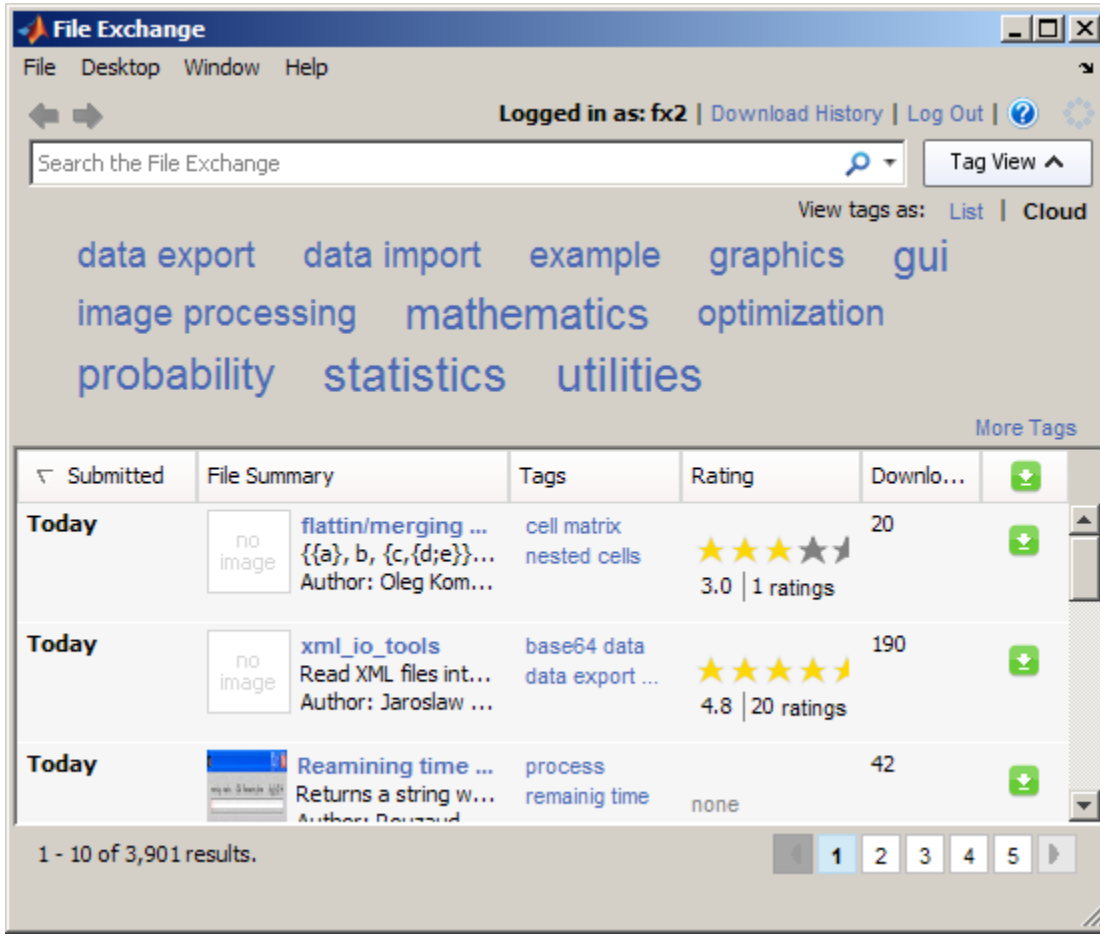
Watch this video, MATLAB File Exchange Access, or review the following summary of what you can do with the new File Exchange desktop tool:

- Access user-created files from the File Exchange repository. The files are at MATLAB Central, the community area of the MathWorks Web site.
- Use the files in your own work to save time and get new ideas.
- Work with the file repository from within the MATLAB desktop.
- Use features like those found in the Web site interface to the File Exchange repository.

The File Exchange desktop tool offers these main features:

- Find files by searching and by selecting tags (keywords associated with files).
- Sort results. For example, show the most highly rated or the most recent files first.
- View details about a file.
- Download files to use in MATLAB.
- Provide feedback about files.

To open the File Exchange tool, select **Desktop > File Exchange**.



If you have questions while you work, access the File Exchange FAQ by clicking the Help button . For full documentation, see File Exchange — Finding and Getting Files Created by Other Users.

Editing and Debugging MATLAB Code Files

New features and changes introduced in Version 7.9 (R2009b) are:

- "Syntax Highlighting for VHDL and Verilog Code" on page 14-13

- “File and Folders Comparison Tool Enhanced” on page 14-13

Syntax Highlighting for VHDL and Verilog Code

The MATLAB Editor now supports syntax highlighting for VHDL and Verilog code. For details see, Highlight Syntax to Help Ensure Correct Entries in the Editor.

File and Folders Comparison Tool Enhanced

When you use the File and Folders Comparisons tool to compare folders, it now includes the following information about each file in each folder:

- The file size, in bytes
- The date the file was last modified

In addition, the File and Folders Comparisons tool enables you to reload the information by clicking the refresh button or selecting **File** > **Refresh**. For details see, Comparing Files and Folders.

Publishing MATLAB Code Files to PDF Output Format

MATLAB now supports PDF as the output format for publishing MATLAB code files. For instructions on how to publish to `.pdf` files, watch the Publishing Features video demo. For details, see Steps for Publishing sine_wave_f.m to PDF.

Internationalization

How MATLAB Reads Customized Locale Settings on Macintosh OS X Platforms

If you use the **Customize** option to modify the locale setting, MATLAB ignores the customized portion. For example, if you select the euro currency symbol for a locale set to the United Kingdom, the locale name is:

```
en_GB@currency=EUR
```

MATLAB uses the locale name:

```
en_GB
```

Mathematics

Computational Geometry Functions Being Changed

2- and 3-D computational geometry functions (`delaunay`, `convhull`, `griddata`, `voronoi`, `delaunay3`, `griddata3`) no longer use QHULL or the QHULL options arguments. The N-D functions `gridatan`, `delaunayn`, `convhulln`, and `voronoin` still use QHULL.

Compatibility Considerations

The QHULL options arguments to `delaunay`, `convhull`, `griddata`, and `voronoi` are no longer required and are currently ignored. Support for these options will be removed in a future release.

Computational Geometry Functions Being Removed

A future release will remove the `griddata3`, `dsearch`, `tsearch`, and `delaunay3` functions. See the table below for alternatives.

Function Name	Use This Method Instead
`griddata3`	TriScatteredInterp
`dsearch`	DelaunayTri/nearestNeighbor
`tsearch`	DelaunayTri/pointLocation
`delaunay3`	DelaunayTri

Compatibility Considerations

Update your old code to use the new computational geometry classes DelaunayTri and TriScatteredInterp.

New Sparse Matrix Functionality In qr and mldivide Functions

mldivide supports complex rectangular sparse input matrices. qr supports complex sparse input matrices. `qr` for sparse matrix input now supports a third output argument that contains a fill-reducing permutation.

Support for Large-Sized Dimensions In fft

MATLAB functions `fft`, `fft2`, and `fftn` (and their inverses `ifft`, `ifft2`, and `ifftn`) can now handle input arrays with a size in 1 dimension greater than $2^{31}-1$ on 64-bit platforms.

Performance Improvement For Large Data Sets

- MATLAB includes improved sparse matrix performance for indexing, basic math, binary and relational operators, and exponential functions.
- There are significant performance improvements to `conv2`.

erfcore Being Removed

Use of `erfcore` will produce a warning:

```
ERFCORE will be removed in a future release.
Use ERF, ERFC, ERFCX, ERFINV, or ERFCINV instead.
```

Compatibility Considerations

Replace all instances of `erfcore` with `erf`, `erfc`, `erfcx`, `erfinv`, or `erfcinv`.

New Multithreading Capability

Many MATLAB functions are now multithreaded:

- sort
- bsxfun
- mldivide for sparse rectangular matrix input
- qr for sparse matrix input
- filter for matrices and higher-dimensional arrays
- gamma, gammaln
- erf, erfc, erfcx, erfinv, erfcinv

New Test Matrices in gallery Function

The gallery suite of test matrices includes new options `integerdata`, `normaldata`, and `uniformdata`.

Data Analysis

Improved Plot Selector Makes Graphic Data Exploration Easier

The Plot Selector button on the Workspace Browser and Variable Editor toolbars has a new look and added capabilities, options, and help. Now the Plot Selector button 

changes its appearance to reflect the variable or variables you select. Now it suggests a plot type and indicates its calling argument sequence. You can also make the Plot Selector list the types of graphs you use most often at the top of its drop-down menu. The tool provides descriptions and hints to help you discover data graphing functions and learn to use them more effectively.

For further details, see this video demo and the release note “Enhanced Plot Selector Simplifies Data Display” on page 14-24.

Programming

Ignore Selected Arguments on Function Calls

This version of MATLAB introduces a new usage for the tilde (~) operator. As in earlier releases, tilde signifies a logical NOT. In the 9b release, you can also use the tilde operator to specify unused outputs in a function call, or unused inputs in a function definition. See Ignore Function Inputs in the Programming Fundamentals documentation for more information on this feature.

Replacing Output Variables with Tilde

This feature enables you to replace this type of function call:

```
[val1, ignoreThisOutput, val3] = myTestFun;
```

with the following:

```
[val1, ~, val3] = myTestFun;
```

MATLAB ignores any values returned by a function that have a corresponding tilde in the output list. This new syntax can help you avoid confusion in your program code and unnecessary clutter in your workspace. It also avoids wasting memory to store unused outputs returned from the called function.

Replacing Input Arguments with Tilde

You can also use the tilde operator to specify unused inputs when you are creating a function. You can replace this type of function definition:

```
function myTestFun(arg1, ignoreThisInput, arg3)
```

with the following

```
function myTestFun(arg1, ~, arg)
```

Whenever this function is called, MATLAB ignores any inputs passed to the function that have a tilde in the corresponding position in the input argument list. You are likely to find the tilde operator most useful when writing callback functions and subclass methods that must match a predefined function interface.

Internal Packages Make Reserved Functions Easy to Identify

The MathWorks® reserves the use of packages named `internal` for utility functions used by internal MATLAB code. Functions that belong to an `internal` package are intended for The MathWorks use only. Using functions that belong to an `internal` package is strongly discouraged. These functions are not guaranteed to work in a consistent manner from one release to the next. In fact, any of these functions and classes may be removed from the MATLAB software in any subsequent release without notice and without documentation in the product release notes.

See Internal Utility Functions in the Programming Fundamentals documentation for more information.

Use of lasterror, lasterr, rethrow(errStruct) Not Recommended

In version 7.5, The MathWorks introduced a new class called MException for use in error handling. Prior to this change, the lasterr and, more recently, lasterror functions kept track of only the one most recently thrown error. The state of any errors thrown before that was overwritten by newer errors. Also, this error state was globally accessible, which means that it could be unintentionally modified or destroyed either during an interactive session or by another function.

When using the newer, object-based error mechanism, every error generated by a MATLAB function stores information about what caused the error in a separate `MException` object. This enables MATLAB to store information about multiple errors, adds the capacity to store new fields of information including links to related errors, and also resolves the problem of being globally accessible.

The MathWorks now discourages the use of the `lasterr` and `lasterror` functions because the information they return is global and thus has the potential to be corrupted. You should replace the use of these two functions with the new style of try-catch statement that captures an `MException` object to represent the error. When entering commands at the MATLAB command line, you can also use the static method Mexception.last to retrieve information on the most recent error. For more information on this method of error handling, see The MException Class.

This change also affects which form of the `rethrow` function to use when reissuing an exception. The original rethrow function, in use since before version 7.5, accepts only a structure as input. Because this structure is typically obtained using the lasterror function, this form of `rethrow` should be avoided. The later form of

rethrow(MException), introduced in version 7.5, accepts an `MException` object as input, and is the recommended form.

Compatibility Considerations

It is strongly recommended that you discontinue the use of `lasterror` and `lasterr`, and that you replace the use of these functions in your program code with the `MException`-based style of error handling described above.

Use of maxNumCompThreads No Longer Recommended

Invoking the function `maxNumCompThreads` now returns the following warning:

```
Warning: maxNumCompThreads will be removed in a future release.
Please remove any instances of this function from your code.
```

`maxNumCompThreads` continues to operate the same in this release of MATLAB. However, the function will be discontinued in a future release of the product.

Compatibility Considerations

It is recommended that you discontinue the use of `maxNumCompThreads`. However, if you do have any program code that invokes this function, you should make sure that such programs are not affected by this new warning.

Excel Worksheet Selection in the Import Wizard

In previous releases, the Import Wizard imported only the first populated worksheet in an Excel file. The Import Wizard now allows you to specify the worksheet to import. To start the Import Wizard, use one of the following methods:

- Select **File** > **Import Data**.
- Call `uiimport`.
- Double-click a file name in the Current Folder browser.

Motion JPEG 2000 Files Supported by mmreader

`mmreader` now imports Motion JPEG 2000 (`.mj2`) files on Windows, Macintosh, and Linux platforms.

Compatibility Considerations

Because `mmreader` imports Motion JPEG 2000 files, the function `mmreader.isPlatformSupported` always returns `true` on Windows, Macintosh, and Linux platforms. For a list of file formats that `mmreader` supports, and the requirements to read these formats on each platform, see the `mmreader` reference page.

Minimum Sample Rate for audioplayer

If you specify a sample rate less than 80 samples per second, audioplayer generates an error with the following ID:

```
MATLAB:audioplayer:positivesamplerate
```

Compatibility Considerations

In previous releases, the error ID on Windows platforms for low sample rates was:

```
MATLAB:audioplayer:negativesamplerate
```

Change any references to this identifier to:

```
MATLAB:audioplayer:positivesamplerate
```

Documentation Changes: File I/O and Data Import and Export

The function category “File I/O” is renamed “Data Import and Export.” It appears immediately below the category “Desktop Tools and Development Environment.”

In the MATLAB User Guide, Data Import and Export content previously appeared in the Programming Fundamentals documentation. The User Guide now includes a stand-alone topic for Data Import and Export.

Object Array Property Indexing

Object array indexing operations on properties now return an error for improper array references. Before MATLAB 7.9, an expression such as:

```
obj.Prop(n) % for non-scalar obj is invalid if Prop is a property
```

did not return an error. MATLAB 7.9, you can reference or assign properties from scalar objects only by entering:

```
obj(int).Prop(n)
```

where `int` is a positive integer.

Equality of Objects Using isequal Now Ignores Numeric Class

Before MATLAB 7.9, an expression such as:

```
isequal(a,b)
```

returned false in cases where `a` and `b` are objects of the same class that have properties set to numeric values which are mathematically equivalent, but of different classes (for example, `double` and `single`),

The behavior of `isequal` in MATLAB 7.9 is consistent with the documented behavior (see isequal). `isequal` does not consider class when comparing numeric values.

Class Defining Private/Abstract Property Now Errors

MATLAB 7.9 returns an error when it loads a class that defines a property as both `Abstract` and `Private`.

Subclasses of Built-in Classes and numel

Before MATLAB 7.9, the numel function did not return the same results for built-in classes and subclasses of built-in classes. In MATLAB 7.9, the behavior of `numel` is consistent with built-in classes and subclasses of built-in classes. See Understanding size and numel for a discussion of the current behavior.

Array Expansion with Indexed Assignment

Before MATLAB 7.9, initializing a handle object array by specifying the last element in the array to create it:

```
obj(10) = MyClass;
```

resulted in the same handle being assigned from the second to the penultimate elements of the array. With MATLAB 7.9, all elements in the handle object array have unique handles. See Creating Object Arrays for more information on the current behavior.

New Tiff Object Enables Writing of Tiled Data and Broader Metadata Support

Using the new `Tiff` object, you can now write portions of a TIFF file and update the values of individual metadata tags in an image. MATLAB has long offered the capability of reading and writing TIFF files, using the `imread` and `imwrite` functions. However, if you wanted to update any part of the image, you had to write the entire image to the file. Similarly, if you just wanted to update a tag, you had to write the entire image. By providing access to functions in the LibTIFF library, the Tiff object offers more flexibility in creating and editing TIFF images. You can write data to specific tiles in an image and update individual tags in a file, without having to write the entire image.

Ambiguity Error Now Reported

Due to a bug, versions of MATLAB prior to Version 7.9 did not report an error in functions such as the following, even though it cannot be determined in the last line whether `U` is a variable or a function:

```
function r = testMfile1(A)
A = 1;
eval('U = 1;');
r = A(logical(end), U(end));
```

However, if you replace the last line with

```
r = A(U(end));
```

MATLAB reports an ambiguity error and has done so since Version 7.0. Starting in Version 7.9, MATLAB reports an ambiguity error for either statement.

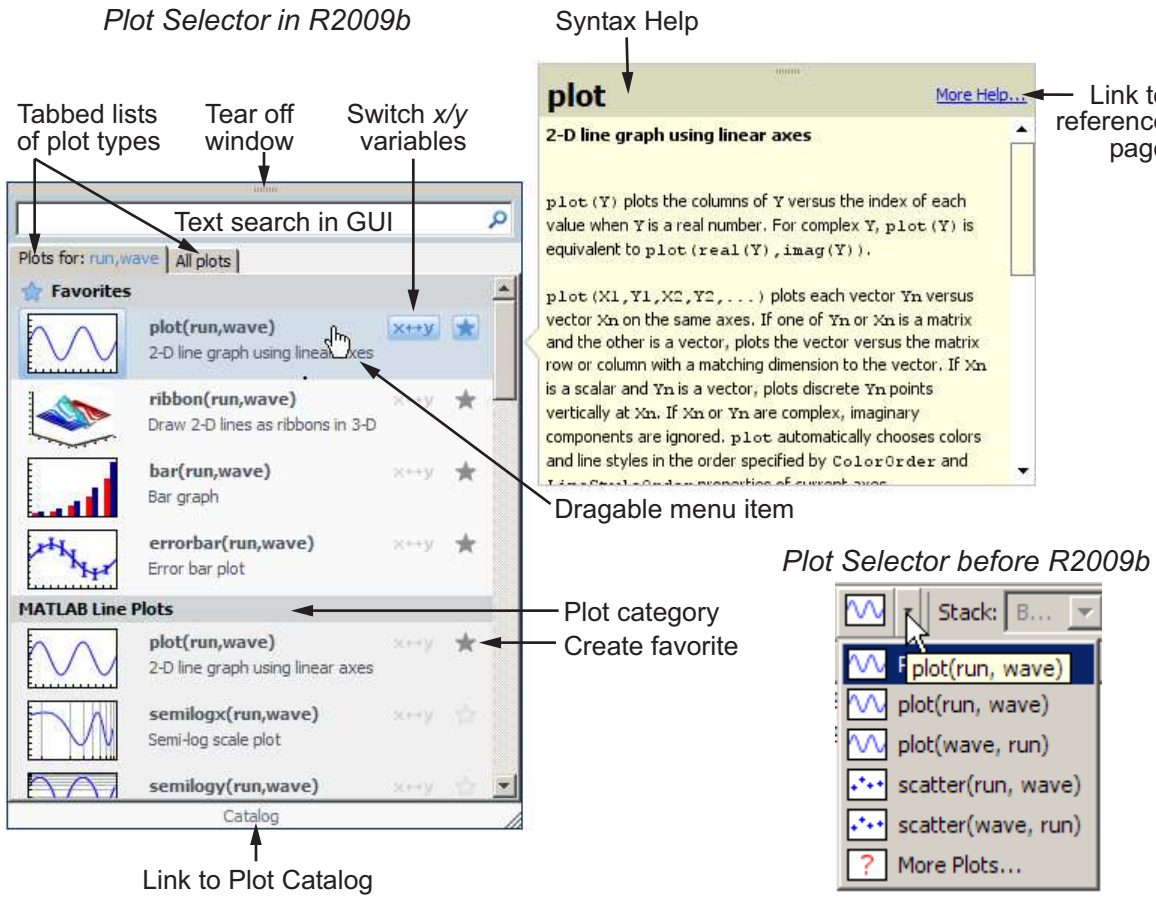
Compatibility Considerations

In certain circumstances, it is possible that you will see ambiguity errors generated in code that did not report this error in previous releases. If you do get such an error, examine your code and resolve the ambiguous statements.

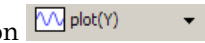
Graphics and 3-D Visualization

Enhanced Plot Selector Simplifies Data Display

The Plot Selector workspace tool creates graphs of workspace variables. As this video demo shows, the tool lets you access more types of data display functions and provides help about each one. It also categorizes display functions and lets you organize them within its drop-down menu. The following illustration shows the old and new versions of the Plot Selector and calls out new features.



Use the Plot Selector tool to instantly generate graphs of workspace variables. You can choose from more than 40 two-dimensional, three-dimensional, and volumetric MATLAB data display types. It also constructs graphs using Control System Toolbox, Financial Toolbox, and Statistics Toolbox functions if your installation includes those products.

The Plot Selector button 

displays a graph icon and the names of selected variables in the Workspace browser or the Variable Editor. Until you select one or more variables, the button reads **Select to plot** 

. The graphing function for one or two data vectors still defaults to `plot`, but the tool now gives you additional options and information, including the following:

- The grouping of menu items into categories of graph types, such as **Stem and Stair Plots**, **3-D Surface Plots**, and **Analytic Plots**. You can rearrange categories and items within a category by dragging them.
- A **Favorites** category at the top of the menu, where you can collect the types of graphs you use most often. Your collection of favorites persists across MATLAB sessions.
- A star-shaped button  on menu items that adds a graph type to or removes one from your Favorites collection.
- Two tabs 

for toggling the scope of the menu:

  - **Plots for <variable names>** — The set of graphs you can generate with the currently selected variables
  - **All plots** — A master list of graphing functions available on your system

  On the **All plots** tab, graphing functions that are not compatible with the selected variables display in gray and do not plot. Incompatible plot types do not appear on the first-tab menu.
- A button 

to interchange the axes positions of two selected variables
- Pop-up descriptions of function syntax when your mouse pointer lingers over a menu item
- Direct access to your Favorites from context menus in the Workspace Browser and the Variable Editor

For more information, see Creating Plots from the Workspace Browser.

Compatibility Considerations

- In previous versions, the Plot Selector automatically assigned variables named `t` and `time` to the *x*-axis. Now, the variable you select first always plots on the *x*- axis.

  Use the switch variable input order button  to interchange the *x* and *y* variables before plotting them.
- In previous versions, clicking **Plot Selector > More Plots** opened the Plot Catalog tool. Now, you click **Catalog** at the bottom of the Plot Selector GUI to open the Plot Catalog tool. The Plot Catalog has not changed.

Certain Print Options and Devices Now Warn When Used

The following print command option and device now throw warnings when used and will be removed in a future release:

Option/ Device	Warning ID	Warning Text
`'-adobecset'` print option	`'MATLAB:print: adobecset:DeprecatedOption'`	The `-adobecset` option will be removed in a future release.
`'-dill'` print device (native support for Adobe Illustrator)	`'MATLAB:print: Illustrator:DeprecatedDevice'`	The `'-dill'` print device will be removed in a future release. Use Encapsulated PostScript (`'-depsc'`) instead.

If you wish to prevent these warnings from displaying, use the warning command.

- To disable the warning generated by using the `–adobecset` print option:

  ```
  warning('off', 'MATLAB:print:adobecset:DeprecatedOption');
  ```
- To disable the warning generated by using the `–dill` print device:

  ```
  warning('off', 'MATLAB:print:Illustrator:DeprecatedDevice');
  ```

Compatibility Considerations

If you have developed code that uses the previous option or device, in the future you must remove such calls or replace them with other code.

The view Function No Longer Supports 4-by-4 Transformation Matrices as Input

The view function. no longer supports 4-by-4 transformation matrices as input arguments. Previously, you could create a 4-by-4 transformation matrix and use it with `view`.

Compatibility Considerations

If you have code that calls `view(T)`, explore using `view([az el])` instead.

Creating Graphical User Interfaces (GUIs)

Expanded Documentation on Techniques for Programmatic GUI Design

The section Lay Out a Programmatic GUI of the Creating Graphic User Interfaces documentation has been expanded. The updated section, formerly called "Aligning Components," now has the title Compose and Code GUIs with Interactive Tools.

The renamed section describes techniques, functions, and predefined dialog boxes you can use to accelerate programmatic construction of GUIs (that is, those you create without using GUIDE).

The section includes examples of setting component positions, colors, and font properties interactively. Also included is an example function that generates a `set` statement for a specified property of any GUI component. By running this function and pasting its output into your GUI code file, you can save time and avoid making typographical errors.

Previous Change to How UI Components Set the Figure SelectionType Property

The MATLAB `SelectionType` figure property had an undocumented change in V. 7.6 (R2008a). That change modified how the property is set in response to single-clicking and double-clicking objects, particularly uicontrols, with and without key modifiers.

For information about this change, including potential incompatibilities, see "Changes to How uicontrols Set Figure SelectionType" on page 17-42. The table in that release note specifies the `SelectionType` setting resulting from clicking a UI component in both enabled and disabled states.

External Interfaces/API

Changes to Compiler Support

Support for Apple Macintosh (64-bit) Platforms

MATLAB Version 7.9 (R2009b) supports these new compilers for building MEX-files:

- Apple Xcode 3.1 (gcc / g++ Version 4.0.1)
- GNU gfortran Version 4.3

Compiler Support to Be Phased Out

MATLAB Version 7.9 (R2009b) supports the following compilers but will not support them in a future version.

Windows (32-bit) Platforms

- Intel Visual Fortran Version 10.1
- Intel C/C++ Version 9.1
- Microsoft Visual Studio .NET Version 7.1

Windows (64-bit) Platforms

- Intel Visual Fortran Version 10.1
- Intel C/C++ Version 9.1

Solaris SPARC (64-bit) Platforms

- Sun Studio 11 cc / CC Version 5.8
- Sun Studio 11 f90 Version 8.2

Discontinued Support for Intel Visual Fortran Version 9.1

MATLAB no longer supports the Intel Visual Fortran Version 9.1 compiler on the following platforms:

- Windows 32-bit
- Windows 64-bit

Compatibility Considerations

To ensure continued support for building your Fortran programs, consider upgrading to another supported compiler. For an up-to-date list of supported compilers, see the Supported and Compatible Compilers Web page.

Run-Time Libraries Required for Applications Built with Microsoft Visual Studio 2008 Compiler

You must provide the Visual C++ run-time libraries if you distribute any of the following:

- MEX-file
- Engine application
- MAT-file application built with the Visual Studio 2008 compiler

You need these files to run applications developed with Visual C++ on a computer that does not have Visual C++ 2008 installed. For information on locating the Microsoft Visual C++ 2008 Redistributable Package (x86), containing `vcredist_x86.exe` and `vcredist_x64.exe`, consult your Microsoft documentation.

Changes to Building MEX-Files

INLINE Option to MEX Function Deprecated

The use of the `-inline` switch has been deprecated and will not be available in future versions of the mex function.

MEX Function No Longer Automatically Includes mexversion.c When Building MEX-Files

The mex function no longer automatically compiles and links its own copy of `mexversion.c` when generating MEX-files.

New Features for Interface to Microsoft .NET Framework

This release includes several changes that affect your use of the Microsoft .NET framework:

- The NET.addAssembly function returns information about the assembly in the NET.Assembly class. You can also add an assembly using an instance of the `System.Reflection.AssemblyName` class.

- You can access elements of a .NET array using MATLAB one-based indexing, as described in Accessing .NET Array Elements in MATLAB.
- To call a generic method, use the NET.invokeGenericMethod function.
- You can change a static property or field name using the NET.setStaticProperty function.
- You can use overloaded operators, such as + and *. For a complete list of supported operators, see How MATLAB Represents .NET Operators.

R2009a

Desktop Tools and Development Environment

Desktop New Features Video

For an overview of the major new features in the MATLAB Desktop Tools and Development Environment area, watch this video demo. Another way to access the demo is by selecting the **Demos** tab in the Help browser, and then selecting **MATLAB > New Features in Version 7.8**.

Startup and Shutdown

The `matlab` command to start MATLAB now supports the `-singleCompThread` option on all platforms. When you specify this option, you limit MATLAB to a single computational thread. By default, if you do not specify this option, MATLAB makes use of the multithreading capabilities of the computer on which it is running.

Desktop

New features and changes introduced in Version 7.8 (R2009a) are:

New System Browser Preference Instead of docopt.m for MATLAB on UNIX Platforms

On UNIX platforms (except Apple Macintosh), MATLAB now uses the Mozilla® Firefox® browser by default to display documents or Web sites in a system browser. If you want MATLAB to use a different system browser, use the new **System Web browser** setting in Web preferences to specify it. In addition, the `web` function with the `-browser` option now determines the system browser to use from the preference. For more information, click the **Help** button in the Web preference pane, or see Web Preferences.

Compatibility Considerations

In previous versions, the default system browser on UNIX platforms was Netscape Navigator®; it is now Firefox. If you do not have Firefox on your system, when MATLAB

tries to use a system browser, it produces a warning. To correct the problem, use Web preferences to specify a system browser that is installed.

In previous versions, if you wanted to use a different browser, you specified it in the `docopt.m` file. Starting in R2009a, MATLAB ignores the browser specified in `docopt.m`. If you have code that relies on `docopt.m`, your code still runs, but it produces a warning. Remove the calls from your code. In future versions, the code will not run and will produce an error.

If you have your own `docopt.m` file, delete it and either use the new default, Firefox, or specify a different system browser using Web preferences.

Test Proxy Settings for Accessing the Internet from MATLAB

If you want to access the Internet from MATLAB, and your network uses a firewall or another means of protection that restricts Internet access, you now can ensure your proxy server settings are working correctly. Click the new **Test Connection** button in Web preferences, and MATLAB will use the proxy settings you specified in Web preferences to attempt to access the Internet. For more information about the proxy server features, click the **Help** button in the Web preferences pane, or see Web Preferences.

Running Functions — Command Window and History

Tab Completion for Class Directories and File Names

The Command Window and Editor now support tab completion for class directories. In addition, the Command Window supports tab completion for class file names. For details, see Complete Names in the Command Window Using the Tab Key.

Help and Related Resources

New features and changes introduced in Version 7.8 (R2009a) are:

Help Browser No Longer Reopens at Startup

When you start MATLAB, the Help browser no longer automatically opens if you had it open when you last quit MATLAB.

Compatibility Considerations

In previous versions, the Help browser opened at MATLAB startup if it had been open when you last quit MATLAB. If you want the Help browser to automatically open at startup, use a startup option. For example, you can include a helpbrowser statement in your `startup.m` file. For more information, see Startup Options.

If you include a statement in a `finish.m` file that closes the Help browser automatically whenever you quit MATLAB, you now can remove that statement because MATLAB now performs the action by default.

docsearch Accepts Multiple Words Without Parentheses

The docsearch function, which you can use to search the documentation, now accepts multiple words as input, without requiring the function form of the syntax. For example, in previous versions, you used `docsearch('word1 word2')`, but now you can use `docsearch word1 word2`. With the new form of the syntax, you can use all options for `docsearch`, such as wildcards (for example, `docsearch wor*)` and exact phrases (for example, `docsearch "word1 word2")`.

View Your Platform (32-bit or 64-bit) and Architecture in the About Dialog Box

To find out if you are currently running a 32-bit or 64-bit version of MATLAB, select **Help** > **About MATLAB**, and view the value in the resulting About MATLAB dialog box. You might need to know the version if you want to take advantage of the benefits of 64-bit MATLAB, or if you want to use files that depend on the version, such as MEX-files.

The About MATLAB dialog box also shows the architecture value, for example, `(win32)` or `(win64)`. You use this value for the `arch` option of the `mex` function.

For more information, see Information About your Installation.

Workspace, Search Path, and File Operations

Enhancements to Current Directory Browser

These are the enhancements to the Current Directory browser:

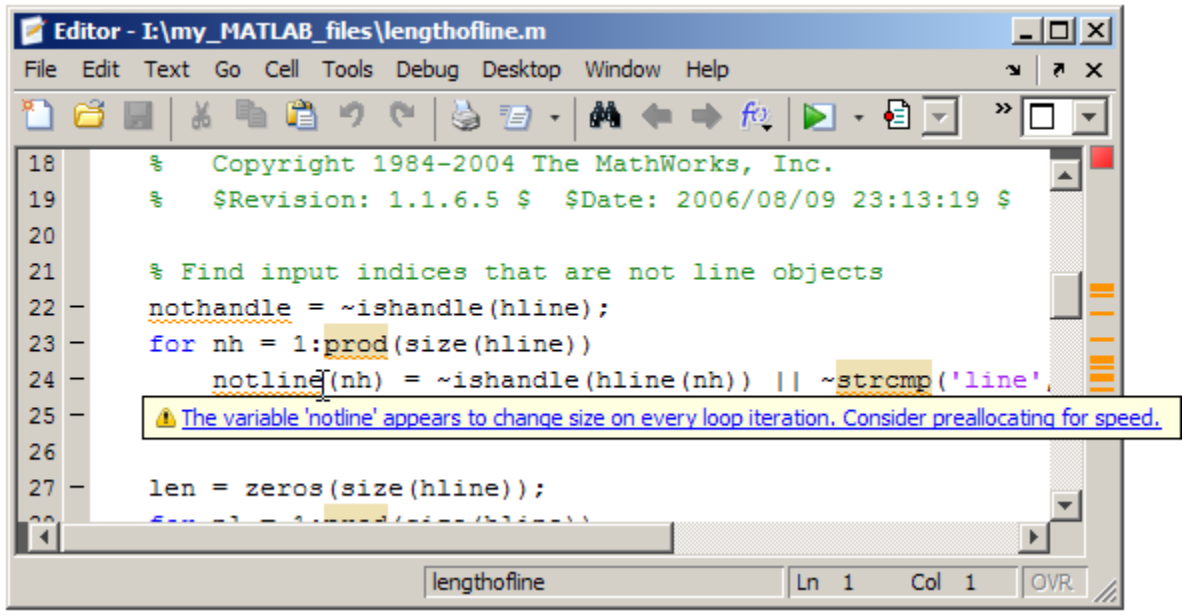
- You can use the new **View** menu to choose the columns to display, to specify the sort order for a column, and to apply grouping for any attribute. In the previous version, you performed these actions using the column header and its context menus, which you can still do (but only on Microsoft Windows platforms). For more information, see Sorting and Grouping Files and Folders.
- A new **Description** column displays the brief description for files and directories in the current directory. To show the column, use **View > Choose Columns**. Descriptions include the first help line in a MATLAB program file, and a model file's description, which is useful because you do not need to start the Simulink software to view it. The description is the same one that appears with the details for a selected file. For more information, see Viewing Help for a MATLAB Program File.
- If you try to rename a MATLAB program file in the Current Directory browser to an invalid name, such as `*myfile.m`, a warning appears, notifying you about the name problem. If you want to run the ile, change the file name to a valid one. For more information about what a MATLAB file name requires to be valid so you can run it, see Naming Functions.
- When you right-click a FIG-file in the Current Directory browser, there is a new option to open the figure in GUIDE.

Editing and Debugging MATLAB Code

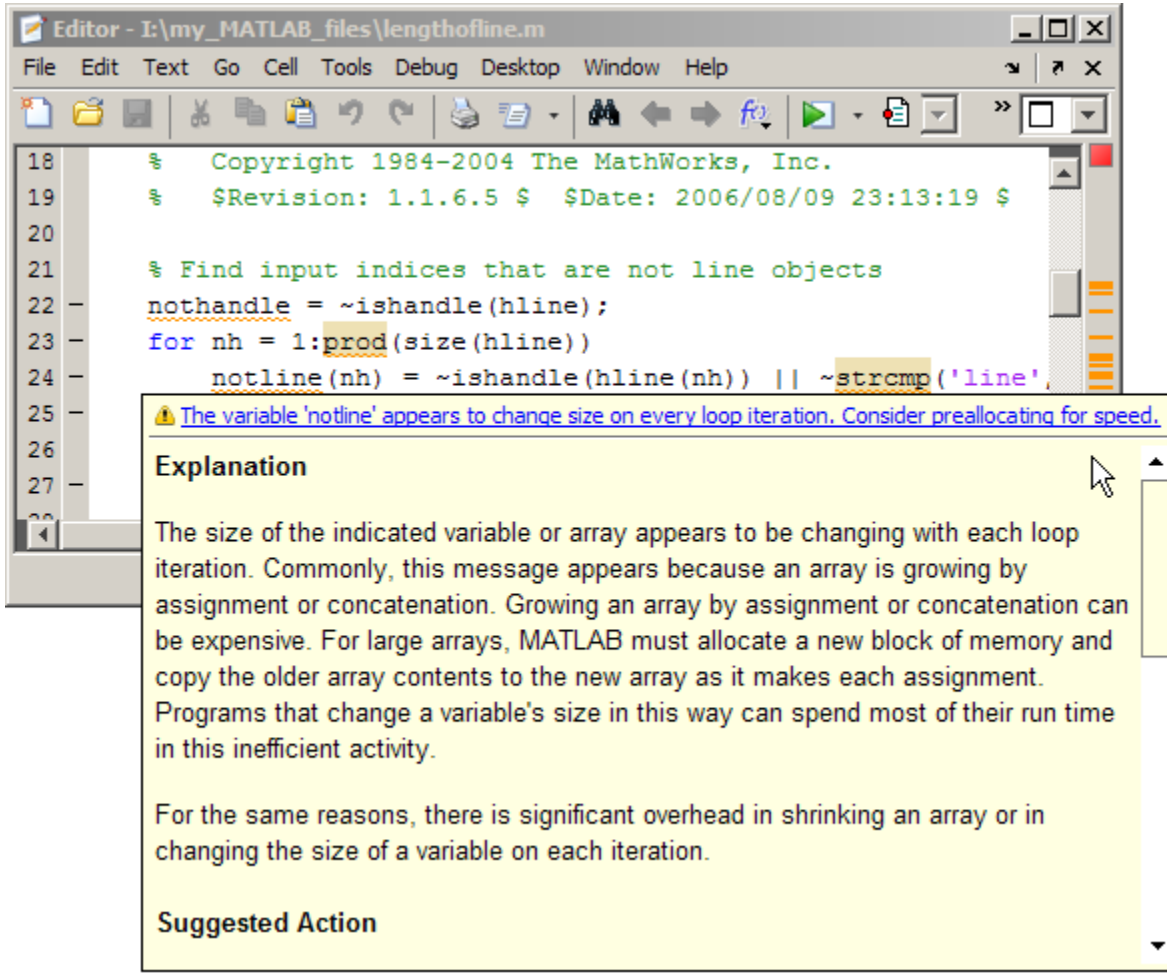
New features and changes introduced in Version 7.8 (R2009a) are:

Many M-Lint Messages Now Extend to Provide an Explanation and Suggested Action

When you create a file with integrated M-Lint warning and error messages enabled, you can get additional information about many of the messages. When you hover the pointer over an M-Lint indicator that has an extended message, the message appears as a link:



When you click the link, the message window expands to display an explanation and suggested action.



When you click a link within the extended message, the Help browser opens to provide more information.

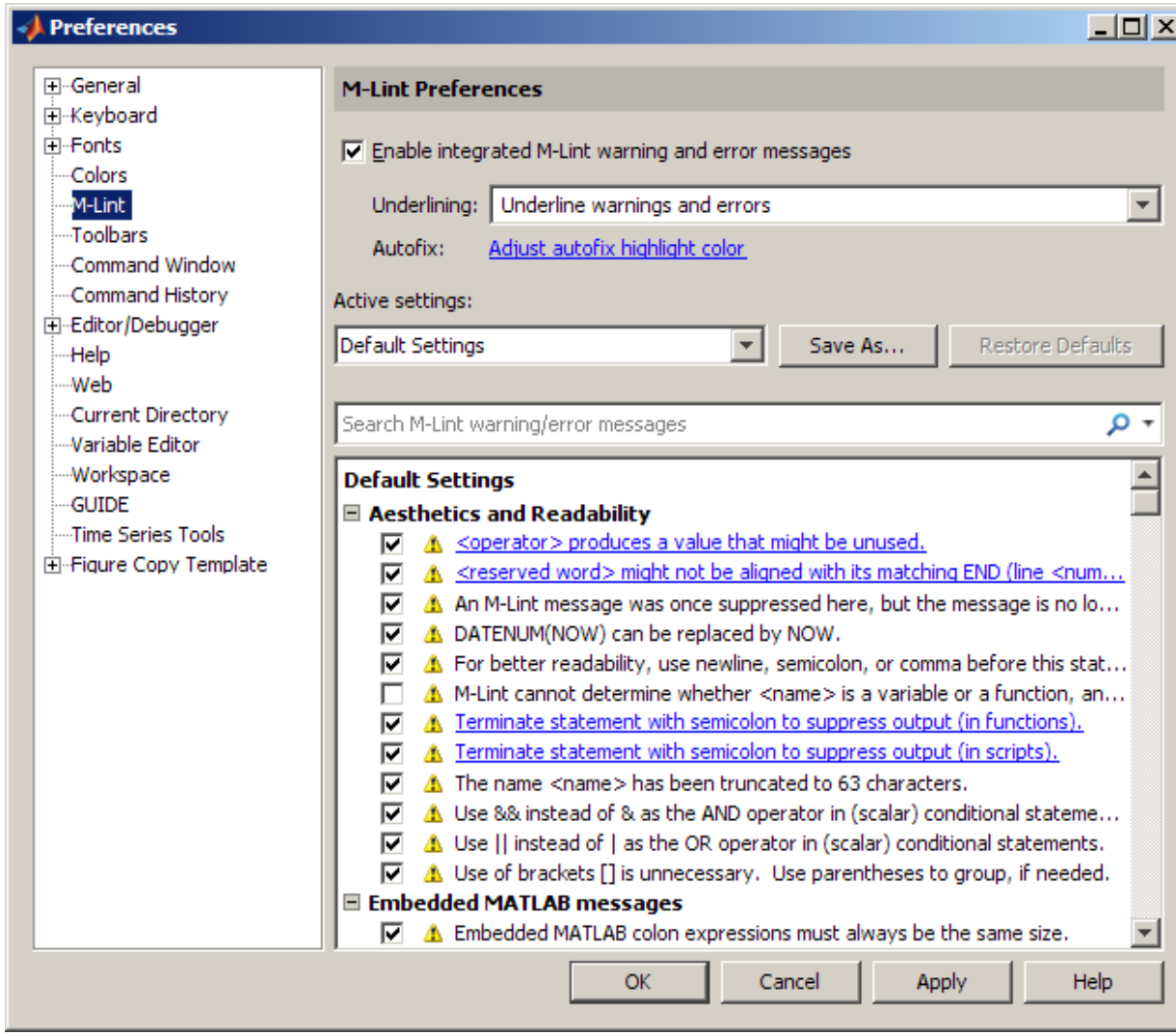
M-Lint Messages Now Searchable in Preferences

You can now filter the list of M-Lint messages in the preferences panel (**File** > **Preferences** > **M-Lint**), to find a message of interest.

For example, you can search for a message:

- Containing a string
- Corresponding to a particular message ID
- Within a given category
- With a setting different from the default

Filtering the list of messages can help you see, for example, why certain messages are suppressed, and which messages are disabled. It can be helpful when you want to see the explanation and suggested action for a message, as described in "Many M-Lint Messages Now Extend to Provide an Explanation and Suggested Action" on page 15-5.



Block Indenting Option No Longer Provided

The **Block Indent** option is no longer available. Previously, this option was available for MATLAB, Java, and C/C++ programming languages, when you selected **File** > **Preferences** > **Editor/Debugger** > **Language**.

Note: Do not confuse the **Block Indent** option with the **Smart indenting** option, which is still provided.

Compatibility Considerations

To attain the effect of block indenting, you can use the **No indent** option and indent lines manually using the **Tab** and **space** keys.

Integrated Text Editor Option Removed from Editor/Debugger Preferences Panel

The MATLAB Editor no longer supports EmacsLink. In previous releases, you could choose **File > Preferences > Editor/Debugger**, and then (if you correctly registered EmacsLink with MATLAB) you could select **Integrated text editor**. This option is no longer available.

New Navigation Aids in File and Directory Comparisons Tool

The File and Directory Comparisons Tool now provides links to help you quickly navigate to the areas of differences. This is useful for large files that have too many lines to fit on the screen. At the top of the page is a link to the first difference. In addition, each set of differences has an up arrow that you click to go to the previous set of differences, and a down arrow you click to go to the next set of differences. For details, see Stepping Through Differences.

Wrap Around Option for Find and Replace Now On By Default

When the **Find Next** operation for searching files in the Editor reaches the end of a file, it automatically wraps around to search from the beginning of the file for the text you specified. In previous releases, the **Wrap around** option was off by default. This meant that if you were in the middle of a file and the text you were searching for appeared before your current position, **Find Next** would not find that text.

Because it is more convenient to have this option on, it is now selected by default. To access this option, select **Edit > Find and Replace**.

For more information, see Find and Replace Text in Files.

Tuning and Managing MATLAB Code Files

Profile Summary Report Includes Information on Excluded Profiling Overhead

The bottom of the Profile Summary Report now indicates the amount of time spent in profiling overhead, when possible. For details, see Profile Summary Report.

Publishing MATLAB Code Files

New features and changes introduced in Version 7.8 (R2009a) are:

New Options for Capturing Figures in Published Documents

Two new options are available to specify how you want figures captured for published documents: `entireGUIWindow` and `entireFigureWindow`. The `entireGUIWindow` option is appropriate for most publishing purposes and is now the default. The `entireFigureWindow` option is appropriate when you want to capture all the details, including the title bar and other window decorations in your published document. Use this option, for example, if you are creating a tutorial on using MATLAB software. In the GUI, you can select these options from the **Figure capture method** setting in the Edit M-file Configuration dialog box. In the `publish` function, you can specify these options with `figureSnapMethod`.

For more information, see Figure capture method and the publish reference page.

Dynamic Links in Published Documents

When you publish a document to HTML, you can include dynamic links. Dynamic links are links to files on the MATLAB path. They are called dynamic links because MATLAB evaluates them when the reader of your document clicks on one. To use a dynamic link, the reader must open the HTML file in the MATLAB Web browser.

For more information, see Dynamic Hyperlinks.

Mathematics

Upgrade to Computational Geometry

MATLAB includes a new object-oriented suite of computational geometry tools, together with a new underlying library called CGAL. The new library provides improved robustness, performance, and memory efficiency. The new tools are presented in three classes:

- New class TriRep provides topological and geometric queries for triangulations in 2-D and 3-D space.
- New class DelaunayTri provides increased functionality for Delaunay triangulation including topological and geometric queries, incremental modification, and edge constraints.
- New class TriScatteredInterp provides fast robust scattered data interpolation and a new natural-neighbor interpolation technique.

Incomplete Inverse Gamma Function gammaincinv and Incomplete Inverse Beta Function betaincinv

MATLAB has new functions for incomplete inverse gamma and incomplete inverse beta functions. These functions, gammaincinv and betaincinv, provide the Statistics Toolbox functionality of the inverse incomplete gamma and beta functions to MATLAB.

Krylov Subspace Methods bicgstabl and tfqmr

MATLAB has new iterative methods for solving systems of linear equations. bicgstabl provides the stabilized biconjugate gradients method. tfqmr provides a transpose-free implementation of the quasi-minimal residual method.

New Function quad2d

The quad2d function provides additional quadrature functionality for nonrectangular areas of integration.

Changes To conv

The conv function now accepts the `shape` parameter as input.

Changes To conv2 and convn

Use of the conv2 and convn functions with one empty input now returns the matrix of the correct size as described by the `shape` input.

Compatibility Considerations

Use of the conv2 and convn functions with one empty input no longer returns an empty matrix. Instead, it returns the matrix of size specified by the `shape` input. For information on how to use the `shape` input, see the reference pages for the conv and conv2 functions.

nextpow2 Changing to Element-By-Element Calculation in a Future Release

Compatibility Considerations

nextpow2 with a nonscalar produces a warning:

```
Warning: NEXTPOW2(X) where X is non-scalar will change
behavior in future versions: it will operate on each element
of X. To retain current behavior, use NEXTPOW2(LENGTH(X)) instead.
```

This behavior will change in a future release. Replace instances of nextpow2 with nonscalar input `X` to `nextpow2(length(X))` to maintain the current behavior.

Function finite Being Removed

The `finite` function is obsolete. Use of the finite function now causes an error in MATLAB.

Compatibility Considerations

Replace all instances of `finite` with isfinite.

New Multithreading Capability in MATLAB Functions

The MATLAB functions for Fourier transforms fft, fft2, and fftn, and their inverses ifft, ifft2, and ifftn are now multithreaded. In addition, the MATLAB functions prod, sum, max, and min are multithreaded.

64-bit Support in LAPACK and BLAS

MATLAB supports 64-bit integers for matrix dimensions in LAPACK, and BLAS. Linear algebra operations can now handle matrices of dimensions greater than $2^{31}-1$.

Compatibility Considerations

`MEX` files that call BLAS or LAPACK need to be updated. All integer variables passed into BLAS or LAPACK need to be of type `mwSignedIndex`. `MEX` files compiled in previous versions of MATLAB that call BLAS or LAPACK could lead to undefined behaviors. If you have existing code that generates `MEX` files that pass variables to BLAS or LAPACK you need to update the code to use the proper data types and recompile.

Upgrade to ACML 4.1.0

AMD Core Math Library (ACML) is upgraded to version 4.1.0.

Data Analysis

Programming Fundamentals

Setting the Number of Threads Removed from Preferences Panel

The capability to adjust the number of computational threads in a single MATLAB session is no longer available as of this release. This change removes from the MATLAB preferences panel the ability to set the maximum number of computational threads. The primary reason for this change is that products that MATLAB is dependent upon have the ability to spawn threads from the currently-executing thread. This makes it infeasible to monitor and/or limit the number of computational threads at any given time in a MATLAB process.

MATLAB versions 7.8 and later require that you decide between threading and no threading at the time you launch your MATLAB session. Multithreading is enabled by default. To disable this feature, start MATLAB using the new `singleCompThread` option.

Compatibility Considerations

If you currently use the preferences panel to enable or disable multithreading or to adjust the number of computational threads, you need to be aware that this capability is no longer available. See the Startup Optionssection in the Desktop Tools and Development Environment documentation to find out how to enable or disable multithreading when launching a new MATLAB session.

Timer Objects Saved in New Format

The format in which MATLAB saves Timer objects has changed in MATLAB version 7.8. Any Timer objects that you create and save while running MATLAB 7.8 cannot be loaded into an earlier version of MATLAB.

Compatibility Considerations

If you need to use a Timer object that you have constructed using MATLAB 7.8, you will have to reconstruct and save the object in an earlier version of MATLAB.

mmreader Supports Linux Platforms

The `mmreader` object now supports Linux platforms. For more information about using `mmreader`, see the `mmreader` reference page.

Support of Microsoft Excel 2007 File Formats

If you have installed Excel 2007 (or Excel 2003 with the Compatibility Pack) on your Windows system, the xlswrite function exports data to XLSX, XLSB, and XLSM formats.

To write data to an Excel file, specify the name and extension of the output file in the call to `xlswrite`. If the file already exists, `xlswrite` writes data in the existent file format. If the file does not exist, `xlswrite` creates a new file, using the format that corresponds to the file extension you specify. If you do not specify a file extension, `xlswrite` applies the XLS extension, and writes a new file in the XLS format.

The xlsread function imports any file format recognized by your version of Excel, including XLS, XLSX, XLSB, XLSM, and HTML-based formats. The importdata function imports XLS, XLSX, XLSB, and XLSM formats. The Import Wizard imports XLS and XLSX formats.

Note: Large files in XLSX format might load very slowly. For better import and export performance, Microsoft recommends that you use the XLSB format.

Anonymous Functions Support str2func

The `str2func` which, prior to this release, converted a function name to a function handle, now also converts an anonymous function definition to a function handle. See the function reference page for str2func for more information.

```
N = 5;   NthPower = str2func(['@(x)x.^', num2str(N)]);
NthPower(8)
ans =
     32768
```

size and range Implemented for validateattributes

The validateattributes function now enables you to check the size and range of the input value. The following commands validate the size and range of the values of `x` and `y` respectively:

```
x = rand(4,2,6);
y = uint8(50:10:200);

validateattributes(x, {'numeric'}, {'size', [4,2,6]});
validateattributes(y, {'uint8'}, {'>=', 50, '<=', 200})
```

isempty Supported for Map Objects

MATLAB now supports the isempty function for `containers.Map` objects. This example creates a 0-by-1 Map object and runs `isempty` on it:

```
mapObj = containers.Map;
size(mapObj)
ans =
     0     1

isempty(mapObj)
ans =
     1
```

Bug Fix for Misinterpreted Variables

The example below describes a bug in the MATLAB software that has been fixed in version 7.8. The bug was caused by misinterpretation of an unassigned variable. For certain names, MATLAB mistakenly interpreted the variable as a stem in dot indexing.

The following example illustrates the problem. The example is a bit artificial (which is why the bug went undiscovered for so long).

Suppose you have a function that is intended to perform two levels of dot-indexing:

```
function y = dotindextwice_A(j)
     y = j.lang.String;
```

Calling this function with no arguments results in an error in all versions of MATLAB, as it should.

Now modify the function slightly so that the input variable is named `java` and run it on a version of MATLAB prior to Version 7.8:

```
function y = dotindextwice_B(java)
y = java.lang.String;
```

Now when you run the function without arguments, MATLAB misinterprets the word `java`, treating it as if it were the stem of a Java class name:

```
x = dotindextwice_B;    % Deliberately called with no arguments
```

Prior to Version 7.8, this function did not throw an error. In fact, it returned an instance of the `java.lang.String` class:

```
class(x)
ans =
    java.lang.String
```

This violates the rule that variables in functions are supposed to hide all other uses of the name. Beginning in Version 7.8, calling function `dotindextwice_B` without arguments results in an error, just as calling `dotindextwice_A` does.

MATLAB Upgrades Support for HDF5 to Version 1.8.1

The R2009a release of MATLAB uses version 1.8.1 of the HDF5 library. The HDF Group has deprecated two of the HDF5 library functions, `H5Pget_cache` and `H5Pset_cache`. Their M-file counterparts, `H5P.get_config` and `H5P.set_config`, may not work as they did in prior releases of MATLAB. To replace these deprecated functions in your code, consider these four new HDF5 functions: `H5P.get_mdc_config`, `H5P.set_mdc_config`, `H5F.get_mdc_config`, and `H5F.set_mdc_config`.

Compatibility Considerations

If your code uses `H5P.get_cache` or `H5P.get_cache`, your program will produce a warning message.

Indirect Calls to Superclass Constructors Now Errors

You can no longer call an indirect superclass constructor from a subclass constructor. For example, suppose class B is derived from class A, and class C is derived from class B. The

constructor for class C should not call the constructor for class A to initialize properties. The call to initialize class A properties should be made from class B.

Compatibility Considerations

If you define classes in which subclass constructors call indirect superclass constructors, MATLAB now issues an error when you attempt to create an instance of the subclass. Call Only Direct Superclass from Constructor for information on how to correctly code subsclass constructors.

Graphics and 3-D Visualization

Functions Previously Only Available in the Image Processing Toolbox Now Available in MATLAB

The following functions have been added to MATLAB from the Image Processing Toolbox:

- `rgb2ind`: Convert RGB image to indexed image.
- `dither`: Convert image using dithering.
- `cmunique`: Eliminate unneeded colors in colormap of indexed image.
- `cmpermute`: Rearrange colors in colormap.
- `imapprox`: Approximate indexed image by one with fewer colors.

Compatibility Considerations

- The `dither` and `imapprox` functions no longer display their output as an image via a call to `imshow` when called with no output arguments. Instead, the first output argument appears in the Command Window if no semicolon ends the line.
- Function `rgb2ind` errors when called with syntax `rgb2ind(rgb)`.
- Function `imapprox` errors when called with syntax `imapprox(x,map)`.

Creating Graphical User Interfaces (GUIs)

New Programmatic GUI Doc Example

A new example in the documentation teaches you how to code a GUI that manages multiple lists, such as to-do and shopping lists, contact information, music or video catalogs, or any set of itemizations. Among other things, it illustrates how to write MATLAB code to:

- Share callbacks among uicontrols.
- Obtain component handles from their tags.
- Create a new version of an existing GUI.
- Import and export list data.
- Edit, add, delete, and reorder list items.
- Save a list in the GUI itself.
- Concurrently run multiple GUIs containing different data that call back to the same function.

This example does not use GUIDE. To read about and run it, see A GUI That Manages List Data in the documentation for Creating Graphical User Interfaces.

GUIDE Help Menu Enhanced

The Help menu in GUIDE now links to more topics in the documentation than previously. It also links to a set of video tutorials on the MATLAB Central Web site. For details, see Getting Help in GUIDE in the Opening GUIDE section of the Creating Graphical User Interfaces documentation.

External Interfaces/API

New Interface to Microsoft .NET Framework

The interface to .NET allows you to bring .NET assemblies into the MATLAB environment, to construct objects from classes contained in the assemblies, and to call methods on these objects. For complete documentation of this feature, see Using .NET Libraries from MATLAB. For an overview of the .NET interface, watch this video demo.

Expanded Platform Support Added for MATLAB Serial Port

MATLAB Serial Port is now supported on the following platforms:

- Microsoft Windows 64-bit
- Apple Macintosh OS X
- Macintosh OS X 64-bit
- Linux
- Linux 64-bit
- Sun Solaris™ 64-bit

Changes To Compiler Support

New Compiler Support

MATLAB Version 7.8 (R2009a) supports these new compilers for building MEX-files:

Microsoft Windows (64- and 32-bit) Platforms

- Microsoft Visual Studio 2008 SP1

Linux (64- and 32-bit) Platforms

- gcc Version 4.2.3

Apple Macintosh (32-bit) Platforms

- GNU gfortran Version 4.2.2

Compiler Support To Be Phased Out

The following compilers are supported in Version 7.8 (R2009a), but will not be supported in a future version of MATLAB:

Windows (32-bit) Platforms

- Intel Visual Fortran Version 9.1
- Intel Visual Fortran Version 10.1
- Intel C/C++ Version 9.1
- Microsoft Visual Studio .NET Version 7.1

Windows (64-bit) Platforms

- Intel Visual Fortran Version 9.1
- Intel Visual Fortran Version 10.1
- Intel C/C++ Version 9.1

Solaris SPARC (64-bit) Platforms

- Sun Studio 11 cc / CC Version 5.8
- Sun Studio 11 f90 Version 8.2

Discontinued Compiler Support

MATLAB no longer supports the following compilers:

Windows (32-bit) Platforms

- Open Watcom Version 1.3
- Microsoft Visual Studio 2005 Express Edition

Windows (64-bit) Platforms

- Microsoft Platform SDK

Apple Macintosh (32-bit) Platforms

- g95 Version 0.90

Compatibility Considerations

To ensure continued support for building your C/C++ programs, consider upgrading to another supported compiler. For an up-to-date list of supported compilers, see the Supported and Compatible Compilers Web page.

Do Not Use mxFree to Destroy mxArrays

It is improper to call `mxFree` on an `mxArray`. Previously, to remedy misleading statements in older documentation, under limited circumstances, MATLAB issued a warning in code that made this error. MATLAB no longer issues the warning.

Compatibility Considerations

The correct function to use to release memory for an `mxArray` is mxDestroyArray. Calling `mxFree` on an `mxArray` could cause memory corruption, which might result in a segmentation violation.

Cannot Build MEX-Files Using MATLAB Version 5 API

MATLAB does not support the `-V5` option to the mex function.

Compatibility Considerations

You are no longer able to build a MEX-file using the MATLAB Version 5 API. If you use any of the functions shown in the "Obsolete Functions: MX Array Manipulation" on page 21-31 table, you must replace them with functions from the Replacement column, if available. These obsolete functions were deprecated when MATLAB Version 6 was released over 5 years ago.

MEX-Files Calling BLAS or LAPACK Functions Must Be Updated On 64-Bit Platforms

You must update any MEX-file that calls functions in the BLAS or LAPACK math packages on 64-bit platforms. The change occurs as a result of updated support, described in "64-bit Support in LAPACK and BLAS" on page 15-14. Existing MEX-files generated in previous versions of MATLAB will result in undefined behavior (likely crashes), if run in R2009a. The previous versions pass 32-bit integer arguments, but the math routines now read and write to 64 bits of memory. The results you see depend on what is stored in the subsequent 32 bits of memory.

Compatibility Considerations

On 64-bit platforms, you must use 64-bit integers for all input and output variables when calling LAPACK and BLAS routines in C and Fortran source MEX-files. Use the mwSignedIndex type for platform-independent code.

Object .o Files Saved on Macintosh Systems for Debugging

MATLAB saves object `.o` files when compiling MEX-files on Apple Mac OS Version 10.5 systems so that you can use source-level debugging.

Run-Time Libraries Required for Applications Built with Microsoft Visual Studio 2008 Compiler

If you distribute a MEX-file, an engine application, or a MAT-file application built with the Visual Studio 2008 compiler, you must provide the Visual C++ run-time libraries. These files are required to run applications developed with Visual C++ on a computer that does not have Visual C++ 2008 installed. For information on locating the Microsoft Visual C++ 2008 Redistributable Package (x86), containing `vcredist_x86.exe` and `vcredist_x64.exe`, consult your Microsoft documentation.

New Features for Shared Library Interface

- It is now possible to use the `char**` return value and to increment the resulting pointer to retrieve all values. See Passing an Array of Strings.
- Added support for accessing values exported by a library.
- All fully and partly sized arrays should now work.

New Java Thread Safety Functions

Use the following new functions to work with Sun Java objects on the Event Dispatch Thread (EDT).

Function	Description
javaObjectEDT	Construct Java object on EDT
javaMethodEDT	Call Java method from EDT

Improved Robustness of Web Services Functions

The underlying technology used in the createClassFromWsdl and parseSoapResponse functions was modified to better ensure support for WSDL and SOAP standards.

Compatibility Considerations

There was no intended change to functionality or results of the createClassFromWsdl and parseSoapResponse functions, which MathWorks verified through testing. There are many variations among WSDL files and Web services and they cannot all be tested. Therefore, it is possible that your results using the `createClassFromWsdl` and `parseSoapResponse` functions in this version could differ from a previous version.

Ensure that your results using `createClassFromWsdl` and `parseSoapResponse` functions are as expected.

R2008b

Desktop Tools and Development Environment

Desktop New Features Video

For an overview of the major new features in the MATLAB Desktop Tools and Development Environment area, watch this video demo. Another way to access this and other video demos is to select the **Demos** tab in the Help browser, and then select **MATLAB > New Features in Version 7.7**.

Startup and Shutdown

New features and changes introduced in Version 7.7 (R2008b) are:

Macintosh Startup and Root Directory Enhancements and Changes

MATLAB for Apple Macintosh platforms is now installed like many other applications for Macintosh platforms, as a Macintosh `.app` bundle. This has resulted in some enhancements and changes to starting and using MATLAB on Macintosh platforms.

Startup Location

To start MATLAB on the Macintosh platform, double-click the `MATLAB_R2008b` icon in the `Applications` folder. This differs from R2008a (V7.6), where you started MATLAB by double-clicking the `MATLAB 7.6` icon in the `Applications/MATLAB_R2008a` folder; there was an additional folder level in R2008a.

Starting MATLAB from a File

You can now start MATLAB by double-clicking a file with a file extension that has been associated with MATLAB, such as a file with a `.m` extension. Similarly, you can drag a file onto the `MATLAB_R2008b` icon in the `Applications` folder or in the dock. These actions start MATLAB and open the file.

Contents of MATLAB Root Directory

When you use file browser GUIs to navigate in the MATLAB root directory, `/Applications/MATLAB_R2008b`, (known as the *`matlabroot`* directory), you cannot directly view or access its contents. For example, when you select **File > Open** and navigate to `Applications/MATLAB_R2008b`, no contents display. To access the contents, press **Command+Shift+G**, and enter the path to the MATLAB root directory in the resulting Go To Folder dialog box, for example, `/Applications/MATLAB_R2008b.app`.

Similarly, when you select `MATLAB_R2008b` in the `Applications` folder using the Finder, you do not see the contents. To access the contents, right-click (or **Ctrl**+click) `MATLAB_R2008b`, and from the context menu, select **Show Package Contents**. For more information, see Navigating Within the MATLAB Root Folder on Macintosh Platforms.

Startup Options and the Start MATLAB Settings Dialog Box

You can no longer set startup options using the Start MATLAB Settings dialog box and you can no longer start MATLAB from any `.smat` files you saved using Start MATLAB Settings. The dialog box is now used only for diagnostics and only appears if MATLAB experiences a problem during startup.

To set the startup directory in MATLAB, use the userpath function. To instruct MATLAB to run a specified statement upon startup, start MATLAB using the `matlab` command from a shell, as you would to start MATLAB on any UNIX platform. For more information, see Startup Options.

Starting MATLAB in a Shell

In R2008b (V7.7), to start MATLAB from a shell, enter the path to the executable, `/Applications/MATLAB_R2008b.app/bin/matlab`. This differs from R2008a (V7.6), in which the path was `/Applications/MATLAB_R2008a/bin/matlab`.

Compatibility Considerations

The compatibility considerations are described along with the above changes.

Updated Version of JVM Software

MATLAB is now using Sun Microsystems™ JVM™ Version 6 Update 4 software on all platforms, except the Apple Macintosh platform. If you specify a version of Java software to use with MATLAB, this change might impact your work.

Specifying Address Space Protection During Startup on Windows Platforms

When you start MATLAB on Microsoft Windows 32-bit platforms, you can set a startup option to help ensure the largest available contiguous block of memory after startup, which is useful if you run memory-intensive operations, such as processing large data sets. You can use the new `-shield` startup option to specify different levels of protection of the address space during the MATLAB startup process. This can also help resolve problems if MATLAB fails to start. For more information, see the matlab (Windows) reference page.

Changes to -nojvm Startup Option

When you start MATLAB with the `-nojvm` startup option, Handle Graphics functionality will no longer be supported. (The `-nojvm` option is available for UNIX platforms; see background information at the matlab (UNIX) reference page.)

Some aspects of MATLAB and related products use Handle Graphics in a way that might not be obvious. This includes anything that is based on or works using figures in MATLAB. Here is a summary of the affected features:

- Creating figures and performing plotting tasks, such as using the `plot`, `axes`, `getframe`, and `gcf` functions.
- Printing figures and using related functions such as `print`, `hgexport`, and `saveas`.
- Creating GUIs in MATLAB using GUI-building functions such as `warndlg`.
- Using Simulink scopes and printing Simulink models.

In MATLAB Version 7.7 (R2008b), if you use the `-nojvm` startup option and use Handle Graphics functionality, MATLAB produces this warning:

```
This functionality is no longer supported under the -nojvm startup
option.
```

In a future release, instead of a warning, MATLAB will produce an error when you start with the `-nojvm` option and use Handle Graphics functionality.

Compatibility Considerations

To avoid the warning, start MATLAB without the `-nojvm` startup option:

- If you have been using the `-nojvm` startup option to work in a command line environment or because you do not use the MATLAB desktop, use the `-nodesktop` startup option instead.

- If you have been using the `-nojvm` startup option because of memory or performance benefits, look for other ways to gain those improvements when you start MATLAB without the `-nojvm` option. See the Performance and Memory Usage topics in the MATLAB Programming Fundamentals documentation.
- If you want to continue to use the `-nojvm` startup option, remove the code that is now producing the warnings.

Changes to matlab Memory Manager Startup Options

The `matlab` command line arguments `-memmgr` and `-check_malloc` are deprecated and will be removed in a future release. The environment variable `MATLAB_MEM_MGR` is also deprecated and will be removed. For information about these options, see matlab (Windows) or matlab (UNIX).

Compatibility Considerations

If you use these options, MATLAB generates a warning message.

Desktop

New features and changes introduced in Version 7.7 (R2008b) are:

New Default Layout for Desktop

There is a new desktop layout when you select **Desktop > Desktop Layout > Default**. It includes the same components as the previous default desktop layout, however, they are arranged differently. If you prefer the previous default layout, arrange your desktop in that way and save the layout. You then can reuse the saved layout at any time.

Closing Document Windows Using Middle Mouse Button

If you have multiple documents open, you can now close a document by clicking the middle mouse button when the pointer is in the document's button on the document bar.

Preferences Opens to Last Pane Used

When you open the Preferences dialog box, it displays the last preference pane you viewed in the current session. In prior versions, the Preferences dialog box displayed the pane associated with the tool from which you accessed it.

Changes to Desktop Text Font

You now can specify that the desktop text font use the system default font. To do this, select **File > Preferences > Fonts**. Then, for **Desktop text font**, select **Use system font**. For more information, click the **Help** button in the Fonts Preferences dialog box.

The default settings for the desktop text font and the HTML Proportional Text font have changed. This only affects existing users who choose to use a new preferences file (`matlab.prf`) in R2008b.

Provide Authentication Settings for Proxy Server when Accessing the Internet from MATLAB

If you want to access the Internet from MATLAB and your network uses a firewall or another means of protection that restricts Internet access and requires you to provide a username and password, use the new proxy server authentication settings in Web preferences to specify the values. For more information, click the **Help** button in the Web preferences pane, or see Web Preferences.

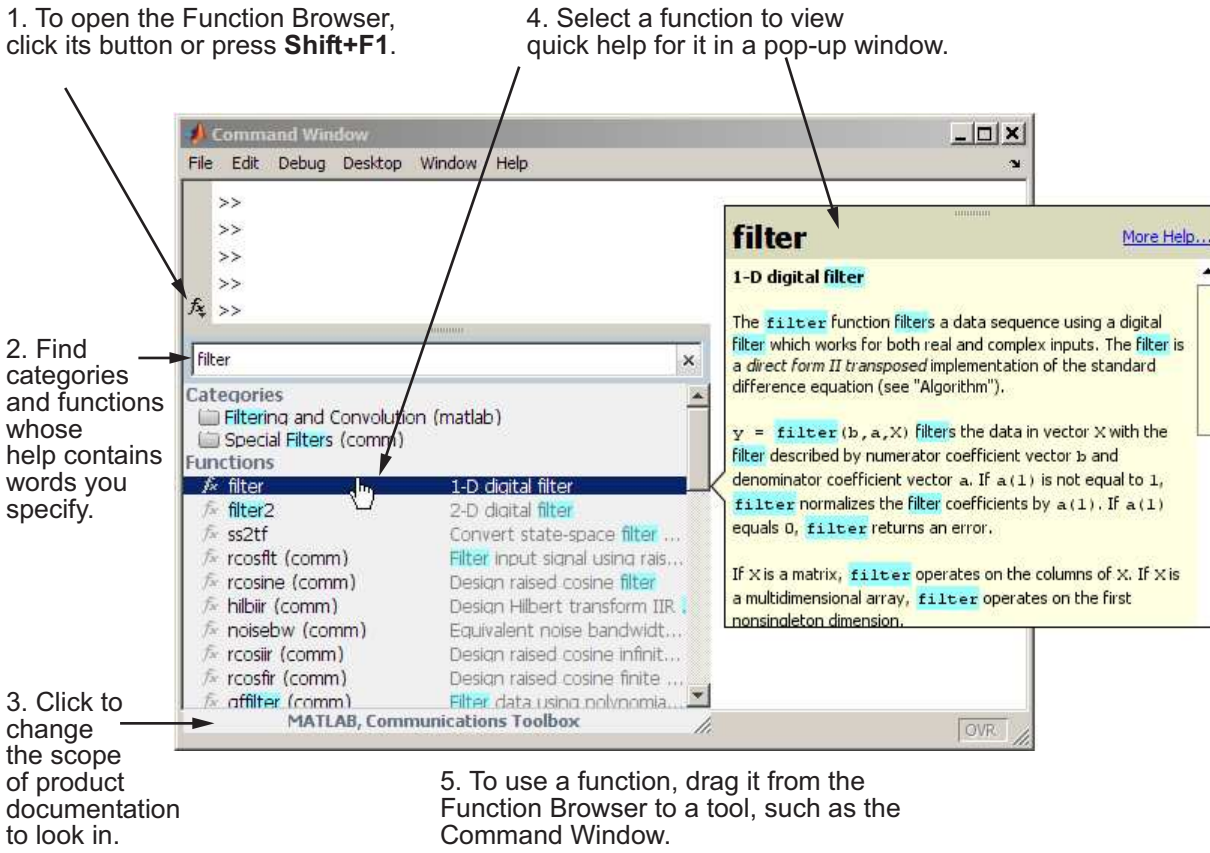
Running Functions — Command Window and History

Find Function Names and Get Help Using the New Function Browser

While you work, you can find the names of functions and get help for them using the new Function Browser. The Function Browser is useful if you want to find a function whose name you cannot remember, determine if there are functions that do what you want, or view the reference page for a function. The Function Browser uses a subset of the information found in the function reference pages in the Help browser for quick access while you work.

To access the Function Browser, press **Shift+F1**, or if you are in the Command Window or Editor, click the Function Browser button  . The Function Browser opens. You can

specify the products to look in, browse categories of functions, and search for words that appear in function reference pages. For more information, see Find Functions Using the Function Browser.



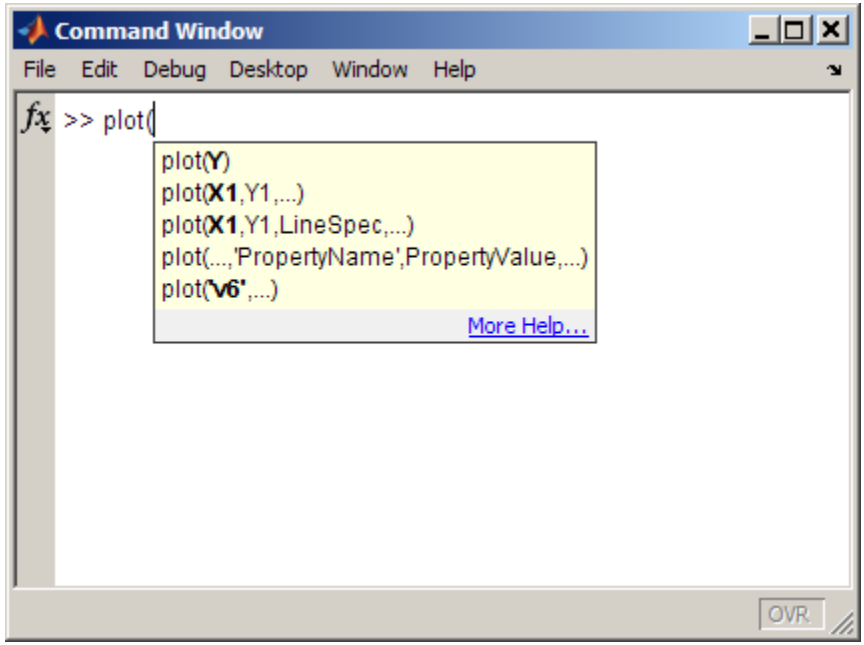
View Syntax Hints While Entering Statements

While you enter statements in the Command Window and Editor, you can view the allowable input argument syntax for a function in a pop-up window. This feature is called *function hints*. To use function hints:

1. Type the function name, followed by the left parenthesis, `(`, and pause. The syntax for input arguments automatically displays in a pop-up window near the statement you are entering.

2. The arguments you can enter appear in bold. Enter your first argument and type a comma (,) after it. The syntax options in the pop-up window then change, based on the argument you just entered.
3. Continue entering arguments, using the hints as needed. You can dismiss the function hints pop-up window at any time by pressing **Esc**. When you type the closing parenthesis,), or when there are no more arguments to enter, the pop-up window automatically closes.

The following illustration shows function hints for the `plot` function.



To turn off the function hints feature so the pop-up menu does *not* automatically display the syntax reminders when you type an opening parenthesis, use Keyboard preferences; for **Function Hints**, clear the check box that enables them.

For more information, see View Function Syntax Hints While Entering a Statement.

Help and Related Resources

- "New Help Features — Function Browser and Function Hints" on page 16-9

New Help Features — Function Browser and Function Hints

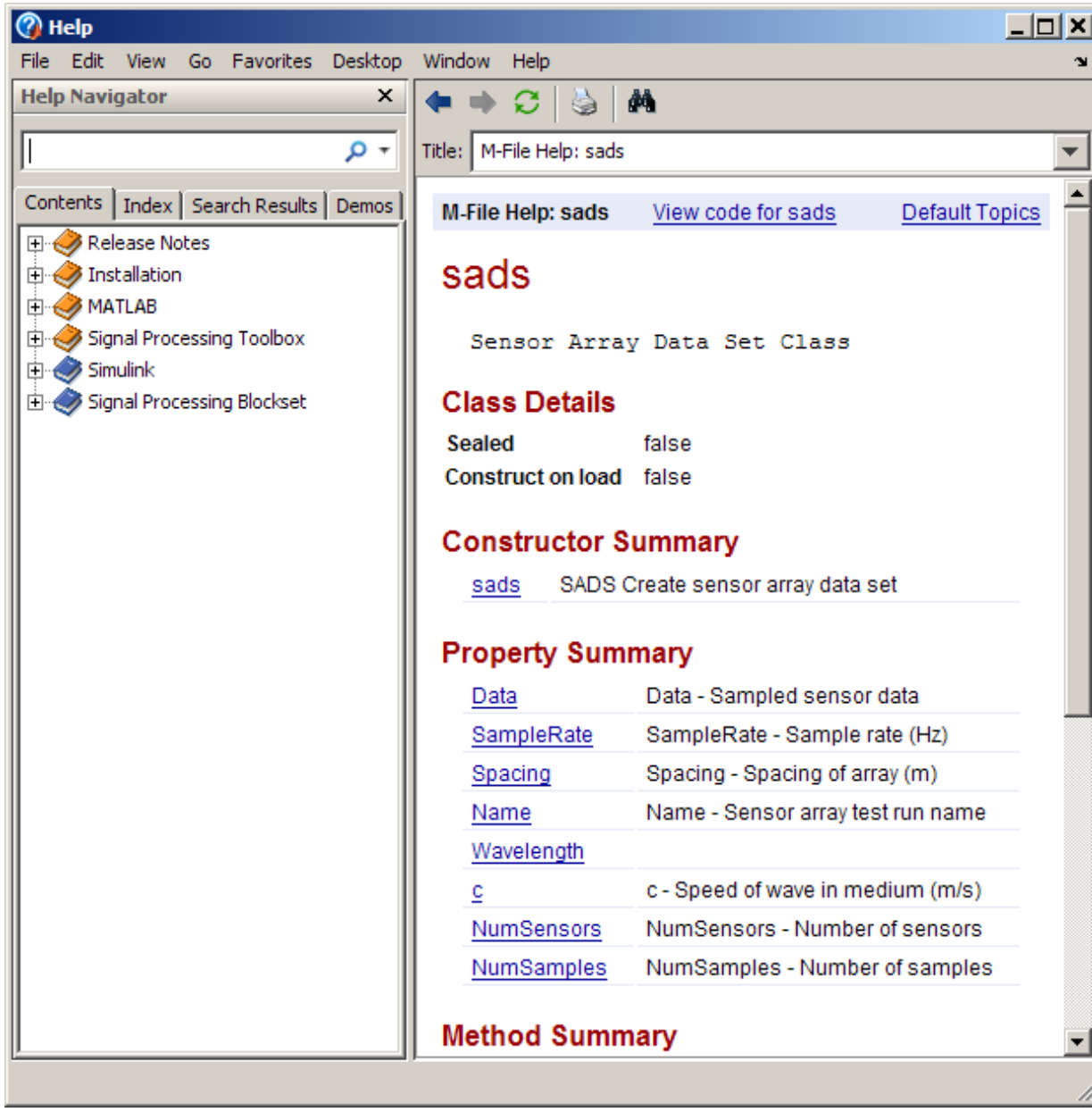
There are two new features that provide help while you work:

- Use the new Function Browser to find function names while you work. It is most useful in the Command Window and Editor, but you can access it from any tool. For more information, see "Find Function Names and Get Help Using the New Function Browser" on page 16-6.
- Use function hints to help you complete syntax for statements in the Command Window or Editor. For more information, see "View Syntax Hints While Entering Statements" on page 16-7.

Viewing an HTML Version of Help for Classes You Create

If you create your own class definition files in MATLAB and you provide help in the `classdef` file for the class, properties, and methods, you can conveniently view the help in the Help browser by running `doc` *`classname`*. For more information, see Help for Classes You Create. The following example shows class information in the Help browser for the user-created class, `sads`, displayed by running

```
doc sads
```


Help
File Edit View Go Favorites Desktop Window Help
Help Navigator
Contents Index Search Results Demos
Release Notes
Installation
MATLAB
Signal Processing Toolbox
Simulink
Signal Processing Blockset
Title: M-File Help: sads
M-File Help: sads View code for sads Default Topics
sads
Sensor Array Data Set Class
Class Details
Sealed false
Construct on load false
Constructor Summary
sads SADS Create sensor array data set
Property Summary
Data Data - Sampled sensor data
SampleRate SampleRate - Sample rate (Hz)
Spacing Spacing - Spacing of array (m)
Name Name - Sensor array test run name
Wavelength
c c - Speed of wave in medium (m/s)
NumSensors NumSensors - Number of sensors
NumSamples NumSamples - Number of samples
Method Summary

Finding Text In Small Help Windows

The Find dialog box is now available from the small help windows used for the Help on Selection feature and for context-sensitive help. The find feature is useful when you want to search for a specific word or phrase within one of these help windows. To access the Find dialog box:

- In a window used for the Help on Selection feature, press **Ctrl+F** on Windows and UNIX platforms or **Cmd+F** on Macintosh platforms. You can also click the **Find text**  button
- In a context-sensitive help window, press **Ctrl+F** on Windows and UNIX platforms, and **Cmd+F** on Macintosh platforms

Changes to Search Field in Help Browser

You can quickly access the Product Filter from the search field by selecting the down arrow at the right side of the search field and from it, selecting **Filter by Product**. The Help pane of the Preferences dialog box opens. For more information about the product filter, click the **Help** button in the dialog box.

As you enter a term in the search field, a history of terms you previously entered in the current session appears. To view the full history, select the down arrow at the right side of the search field and from it, select **Show Search History**. You can select an item in the search history to rerun the search.

To execute a search, enter the search terms and press **Enter**. There is no longer a Go button.

Workspace, Search Path, and File Operations

New features and changes introduced in Version 7.7 (R2008b) are:

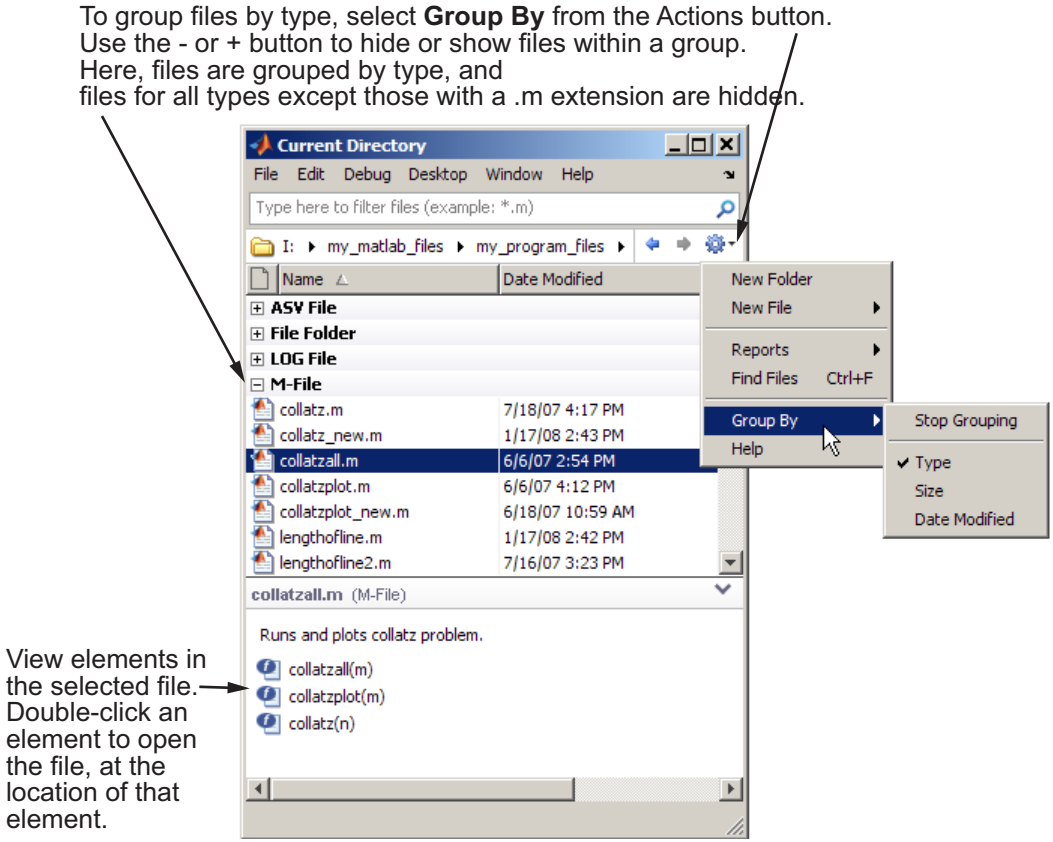
- "Current Directory Browser Enhanced, Including New Navigation and Grouping Features" on page 16-12
- "Structure Results of dir for Nonexistent Files Now Include Empty Matrices" on page 16-15
- "Workspace Browser Toolbar Is Now Configurable" on page 16-16
- "Semicolon (;) Path Separator Character Now Used on UNIX Platforms" on page 16-16

Current Directory Browser Enhanced, Including New Navigation and Grouping Features

The Current Directory browser includes new ways to navigate and to view the directory contents. There are also new ways to find files, including a new filter field.

- Use the new address bar to view the current directory and to view and navigate to subdirectories within the current directory path.
- Access common features from the Actions button  on the toolbar.
- Add and remove buttons from the toolbar by right-clicking the toolbar and selecting **Customize**.
- List only files whose names contain a specified string by using the new filter field. To show the filter field, use **File > Preferences > Current Directory**.
- In the **Details** pane at the bottom of the tool, view a list of elements in the selected file, such as subfunctions in a MATLAB program file. Double-click an element to open the file at the location of the element.
- For a MAT-file, drag selected variables from the **Details** pane to the Workspace browser to load them into MATLAB. Similarly, save workspace variables to a MAT-file by dragging them from the Workspace browser to the Current Directory browser.
- Group items in the current directory, that is, view related items together. From the Actions button, select **Group By** and select the attribute you want to group by.

Some of the major changes are highlighted in the following illustrations.



For more information, see Managing Files in MATLAB.

Compatibility Considerations

The following aspects of using the Current Directory browser are different in Version 7.7 (R2008b) than in the previous version. Following is a table of what changed and the way to perform the same action in Version 7.7.

Change	What To Do Instead In Version 7.7 (R2008b)
The toolbar no longer includes the buttons it previously included.	You can now specify which buttons to include on the toolbar. You can include the toolbar buttons included in previous versions, or use these alternatives:

Change	What To Do Instead In Version 7.7 (R2008b)
	• Go up one level button — Instead, select a directory one level up by using the address bar. • Find files button — Instead, select **Find Files** from the Actions button on the toolbar, or use **Ctrl+F**. • New folder button — Instead, select **New Folder** from the Actions button on the toolbar. • Directory reports — Instead, select **Reports** from the Actions button on the toolbar.
The string display of the full path for the current directory has been replaced by the address bar view.	You can still access the string display, which can be useful if you want to copy it or edit it directly. To access the string, click the blank area to the right of the last subdirectory in the address bar. To return to the address bar view, press the **Escape** key.
You do not select attributes (columns) to display using Current Directory Preferences (**Browser display options**).	Right-click any column header to select or clear the columns to display, such as file size. You can only do this on Microsoft Windows platforms.
The file description no longer appears in a column.	View the description for a selected file in the detail pane.
The full help for a MATLAB program file no longer appears in the detail pane.	Press **F1** to view the reference page for the selected file.
You do not elect to show details using Current Directory Preferences (**Browser display options**).	Use the arrow on the right side of the detail pane separator to show or hide the details.

Change	What To Do Instead In Version 7.7 (R2008b)
The **File Filter** options, accessible from the context menu and the **View** menu, have been removed.	Use the new filter field (use Preferences for the Current Directory browser to show the filter field), or sort by file type (click the document icon header), or group by file type using the new grouping feature (available from the Actions button).
The **View** menu was removed.	Use the new filter field (previously described) to filter the view. For Directory Reports, select **Reports** from the Actions button on the toolbar.
When you look for items in the current directory by typing the initial letters in the name, a pop-up window no longer appears showing the letters that you typed.	As before, the first entry whose name begins with the letters you typed is selected. If you want to see what you type when finding items in the current directory, instead use the filter field.

Structure Results of dir for Nonexistent Files Now Include Empty Matrices

When you run `dir` with an output argument and the results include a nonexistent file or a file that `dir` cannot query for some other reason, `dir` now returns empty matrices for the `date`, `bytes`, and `datenum` fields. The most common occurrence is on UNIX platforms when `dir` queries a file that is a symbolic link and the symbolic link points to a nonexistent target. A nonexistent target is when a target has been moved, removed, or renamed. For example, if `my_file` in `my_dir` is a symbolic link to another file that has been deleted, then running

```
r = dir('my_dir')
```

includes this result for `my_file`:

```
r(n) =
  name: 'my_file'
  date: ''
  bytes: []
  isdir: 0
  datenum: []
```

where *n* is the index for `my_file`, found by searching `r` by the `name` field.

With empty matrices returned, code you write to use the results of `dir` can be simpler and more robust. For more information, see the reference page for the dir function.

Compatibility Considerations

In previous versions, `dir` returned inappropriate values for files that could not be queried:

```
r(n) =
 name: 'my_file'
 date: '18-Jan-2038 22:14:07'
 bytes: 0
 isdir: 0
 datenum: 7.4438e+05
```

If you have existing files that rely on the results of `dir` for a file that could not be queried, your code might not produce the result it used to. If you write new code that depends on the new results for `dir` and run it in a previous version of MATLAB, it might not produce the results you expect. In either case, there will probably not be a warning or error.

Workspace Browser Toolbar Is Now Configurable

Rearrange, add, or remove buttons and other controls from the Workspace Browser toolbar using **File > Preferences > Toolbars**. Alternatively, right-click the toolbar and select **Customize** from the context menu. By default, the Print button is not on the toolbar.

For more information on customizing the toolbar, see Toolbar Customization.

Semicolon (;) Path Separator Character Now Used on UNIX Platforms

On UNIX platforms, MATLAB now uses a semicolon (;) as the character that separates directories in the search path file, `pathdef.m`, which is the character used on Windows platforms. This is beneficial if you work with the contents of the `pathdef.m` file programmatically on both Windows and UNIX platforms. In versions prior to Version 7.7 (R2008b), MATLAB used the colon (:) character as the path separator on UNIX platforms.

Compatibility Considerations

MATLAB will continue to accept the colon (:) character as a valid path separator on UNIX platforms, so any existing code you have that relies on it will still work.

You will experience a problem if you have any directories that contain a semicolon in their name. You will need to rename the directories to include them on the search path.

Editing and Debugging MATLAB Code

New features and changes introduced in Version 7.7 (R2008b) are:

Create New Function and Class Files Using Templates

When you create new functions and class definition files (`classdef`), you can start with a template that reminds you to include standard information. Select **File > New > Function M-File**, or **File > New > Class M-File**. A new file containing template information opens in the Editor.

View Syntax Hints, Find Function Names, and Get Quick Help

While you enter statements in the Editor or Command Window, you can display the syntax for a function in a temporary pop–up window. For more information, see "View Syntax Hints While Entering Statements" on page 16-7.

You also can find the names of and get help for functions using the new Function Browser—for more information, see "Find Function Names and Get Help Using the New Function Browser" on page 16-6.

Set Color and Width of Right-Hand Text Limit

In the Editor, where you can enable a vertical line to indicate a right-hand text limit, you now can set the width and color of the line. Note that the default color for the line is now gray, instead of light red.

Set preferences for the line by selecting **File > Preferences > Editor/Debugger > Display**, which opens the Preferences dialog box.

Click **Help** in the Preferences dialog box for more information.

Set Cursor to First Nonwhite Character on Line

When you press the **Home** key, the cursor goes to the first nonwhite character on the current line. When you press the **Home** key twice, the cursor goes to the beginning of the line.

Compatibility Considerations

In versions prior to Version 7.7 (R2008b), the **Home** key always moved the cursor to the first column of the current line.

Suppress a Specific M-Lint Message Throughout a File

You can now suppress a specific M-Lint message throughout a file by right-clicking on an M-Lint indicator that elicits the message, and then from the context menu, selecting `Suppress All Instances in this File`.

For details, see Suppress All Instances of a Message in the Current File in the MATLAB Desktop Tools and Development Environment documentation. For information on manually inserting the string that tells M-Lint to suppress the message throughout the file, see the documentation for the mlint function.

New M-Lint Message for Suppressed Messages

An M-Lint message now appears if you previously suppressed a message using the `%#ok` directive, and that message no longer appears. This can result if the message is already suppressed using preferences (**File > Preferences > M-Lint**), and the code has changed such that the message is no longer generated, or if the rules M-Lint follows for generating the message have changed. The new message is:

```
An M-Lint message was suppressed here, but the message no longer
```

```
appears. Use the context menu to fix.
```

For an example, see the *Suppressing All Messages on a Line with mlint* example in the mlint function documentation.

View M-Lint Message in ToolTip Using the Keyboard

To open an M-Lint Message ToolTip using the keyboard, place the cursor over the marked code and press **Ctrl + M**. This feature is offered in addition to the identical behavior available when you use the mouse pointer to hover over code that is marked by M-Lint. For an example of viewing an M-Lint message in a ToolTip, see Check Code for Errors and Warnings in the MATLAB Desktop Tools and Development Environment documentation.

Apply M-Lint Autofix Using the Keyboard

To fix a problem marked by M-Lint as having an automatic fix available, place the cursor over the marked code, and then press **Alt + Enter**. This feature is offered in addition to the identical behavior available when you use the context menu. For an example of using the M-Lint autofix feature, see Check Code for Errors and Warnings in the MATLAB Desktop Tools and Development Environment documentation.

Code Fold Single Program, Multiple Data (spmd) Blocks

By default, the Editor now supports code folding for single program, multiple data (spmd) blocks. For more information on code folding, see Code Folding — Expand and Collapse Code Constructs in the MATLAB Desktop Tools and Development Environment documentation.

File and Directory Comparisons Tool: Highlight Changes

The File and Directory Comparisons Tool now uses shades of colors to mark differences in the contents of two directories being compared. Light colors indicate files that differ and dark colors indicate subdirectories that differ. For details and an example, see Comparing Folders and Zip Files in the MATLAB Desktop Tools and Development Environment documentation.

Compatibility Considerations

In releases prior to Version 7.7 (R2008b), the same color intensity highlighted differences in both files and directories.

Block Indenting Will Not Be Included in Next Version

The **Block Indent** option will no longer be provided, starting in the next version of MATLAB. Currently, this option is available for M, Java, and C/C++ programming languages, when you select **File** > **Preferences** > **Editor/Debugger** > **Language**. To attain the effect of block indenting, you can use the **No indent** option and indent lines manually using the **Tab** and **space** keys.

If you have concerns about the pending removal of the **Block Indent** option, please contact Technical Support at http://www.mathworks.com/support/contact_us.

Accessing Contents of MATLAB Root Directory on Macintosh Platforms

Starting in MATLAB 7.7 (R2008b), on Macintosh platforms, you cannot use file browser GUIs to directly access contents of the MATLAB root directory. For example, when you use **File** > **Open** from the Editor, you cannot directly select a file located within `matlabroot`. For more information, see “Contents of MATLAB Root Directory” on page 16-3.

Tuning and Managing MATLAB Code Files

Access Directory Reports

You now access Directory Reports by navigating to the directory containing the MATLAB program files for which you want to produce reports. Then, on the Current Directory browser toolbar, click the **Actions** down arrow  and select the type of report you want to run for all of the MATLAB program files in the current directory. Note that these reports are now referred to as Reports, rather than Directory Reports.

Compatibility Considerations

In versions prior to Version 7.7 (R2008b), you navigated to the directory containing the MATLAB program files for which you wanted to produce reports. Then, you clicked the **Directory Reports** down arrow  on the Current Directory browser toolbar.

Publishing MATLAB Code Files

New features and changes introduced in Version 7.7 (R2008b) are:

Include Figure Window Details in Published Documents

In versions prior to Version 7.7 (R2008b), when you published a file that included a figure window, only the graph or figure was included in the published document. Using the `figureSnapMethod` option, you can now specify that you want the window details included in the published document. For details, see the publish reference page.

Inline Math Supported in Published Documents

In versions prior to Version 7.7 (R2008b), you could publish LaTeX code in a published document as a code block, separate from the rest of your comments, if any. Now, you can publish LaTeX math symbols inline with the rest of your comments. For details, see Inline LaTeX Math Equations.

Publish Setting: Cascading Style Sheet Is Now XSL File

In versions prior to Version 7.7 (R2008b), there was a **Cascading Style Sheet** setting on the Publish Configurations dialog box (which you access by clicking the Publish down-arrow button ). This setting is now called **XSL File**, to more accurately reflect the setting.

Compatibility Considerations

The **Cascading Style Sheet** setting on the Publish Configurations dialog box is now **XSL File**.

Mathematics

Upgrade to Random Number Generator

- The randn function uses a new longer period random number algorithm as its default.
- The new function randi returns random integers from a uniform discrete distribution.
- The @RandStream class allows you to construct a random number stream object and set its properties. For more information, see Random Numbers in the MATLAB Mathematics documentation.

Compatibility Considerations

The randn function now produces different results than in previous releases. Because the values returned by randn are intended to be random, this change should not affect most code.

rand and randn now draw from the same random number stream. In prior releases, rand and randn had separate independent underlying random number streams. Since rand and randn now access the same stream, using randn will affect subsequent values produced by rand and vice-versa. See Creating and Controlling a Random Number Stream in the MATLAB Mathematics documentation for more information.

Multipoint Boundary-Value Problems with `bvp5c`

The solver bvp5c will take multipoint boundary-value problems.

Upgrades to `lsqnonneg`

lsqnonneg now runs more efficiently. It accepts sparse matrices as inputs and maintains sparsity throughout its internal iterations.

Functions and Properties Being Removed

Function or Property Name	What Happens When You Use Function or Property?	Use This Instead	Compatibility Considerations
`betacore`	Fails	betainc	Replace all existing instances of `betacore` with betainc.
`colmmd`	Fails	colamd	Replace all existing instances of `colmmd` with colamd.
`symmmd`	Fails	symamd	Replace all existing instances of `symmmd` with symamd.
`flops`	Fails	None	Remove all instances of `flops`. With the incorporation of LAPACK in MATLAB Version 6, counting floating-point operations is no longer practical.

Upgrade to Intel Math Kernel Libraries

For Windows, Intel Mac, and Linux platforms, MATLAB software supports the Intel Math Kernel Library (MKL) version 10.0.3.

Data Analysis

Specialized Data Tips for the hist Function

When you create a histogram display using hist and place data tips in the plot in data cursor mode, they now snap to the top center of the bin you attach them to. The data tip contents have changed as well, and now consist of:

- Number of observations falling into the selected bin
- The x-value of the bin's center
- The lower and upper x-values for the bin

For details, see Using Data Cursors with Histograms in the MATLAB Graphics documentation.

Compatibility Considerations

Data tips for histograms no longer display the *x* and *y* coordinates of their locations and no longer snap to the four corners of the bins, as they do for `bar` plots. The data tips are also larger, to accommodate the extra information.

Programming Fundamentals

Fast Key Lookup Provided with New Map Data Structure

This release introduces a new MATLAB class called a Map. An object of the Map class is an array of any MATLAB data type that supports lookup table functionality. Unlike most arrays in MATLAB that only allow access to the elements by means of integer indices, indices for Map containers can be nearly any scalar numeric value or a character string.

The following example creates a Map object that is an array of strings. It is very much like any other string array except that with each *value* of this array there is also a lookup *key* associated with it. This particular Map contains the names of capital cities in the United States. These are the values of the Map object. Associated with each capital city value is the US state that it resides in. These are the keys of the Map object. You look up values in the Map using key indices. Call the containers.Map constructor to create an array of six capital cities indexed by six US states. (The capital of Alaska has purposely been entered incorrectly):

```
US_Capitals = containers.Map( ...
{'Arizona', 'Nebraska', 'Nevada', ...       % 6 States
 'New York', 'Georgia', 'Alaska'}, ...
{'Phoenix', 'Lincoln', 'Carson City', ...   % 6 Capitals
 'Albany', 'Atlanta', 'Fairbanks'});
```

Show the capitals of three of the states by looking them up in the Map using the string indices `'Nevada'`, `'Alaska'` and `'Georgia'`:

```
values(US_Capitals, {'Nevada', 'Alaska', 'Georgia'})
ans =
    'Carson City'    'Fairbanks'    'Atlanta'
```

Correct the capital city of Alaska by overwriting the entry at string index `'Alaska'`:

```
US_Capitals('Alaska') = 'Juneau';

US_Capitals('Alaska')
ans =
   Juneau
```

The term `containers.Map` refers to a `Map` class that is part of a MATLAB package called `containers`. For more information, see Map Containers in the Programming Fundamentals documentation.

Tic and Toc Support Multiple Consecutive Timings

The `tic` and `toc` timing functions now support multiple consecutive timings. Call `tic` with an output `t0` to save the current time as the starting time for some operation. When the operation completes, call `toc` with the same `t0` as its input and MATLAB displays the time between that particular tic and toc.

In the following example, MATLAB measures the time used by each function call and, at the same time, measures the time required for the overall operation:

```
t0 = tic;
   t1 = tic;   W = myfun1(A,B);        toc(t1)
   t2 = tic;   [X,Y] = myfun2(C,W);    toc(t2)
   t3 = tic;   Z = myfun3(A,C,Y);      toc(t3)
toc(t0)
```

You can still call `tic` and `toc` without any arguments. In this case, `toc` just measures the time since the most recent `tic`.

See the function reference page for tic or `toc` for more information.

New Options for MException getReport

The MException `getReport` method has several new options available in this release. You select these options when you call `getReport`. They give you more control over the content and format of the information displayed or returned by `getReport`.

See the function reference page for getReport for more information.

what Function Returns Package Information

The `what` function now includes package information in its output display and a `package` field in the structure array that it returns.

List the packages used in the MathWorks Communications System Toolbox™:

```
s = what('comm');

s.packages
ans =
```

```
    'crc'
    'commdevice'
    'commsrc'
    'commgui'
    'commscope'
    'commutils'
```

You can also call `what` on a specific package name to see what types of directories and files are in the package directory.

See the function reference page for what for more information.

addtodate Accepts Hours, Minutes, Seconds, Milliseconds

In previous releases, the `addtodate` function supported modifying a date number by a specified number of years, months, or days. In this release, you can also modify the date by a specified number of hours, minutes, seconds, or milliseconds.

Add 2 hours, 45 minutes, and 17 seconds to the current time:

```
d1 = now;
datestr(d1)
ans =
   12-Jun-2008 16:15:38

d2 = addtodate(d1, 2, 'hour');
d2 = addtodate(d2, 45, 'minute');
d2 = addtodate(d2, 17, 'second');
d2 = addtodate(d2, 3000, 'millisecond');

datestr(d2)
ans =
   12-Jun-2008 19:00:58
```

See the function reference page for addtodate for more information.

Querying Options Added to pause

There are new syntaxes for the `pause` function:

- To see whether pausing is enabled or not, use one of the following commands:

```
pause query
state = pause('query')
```

- To return the previous pause state when enabling or disabling pausing, use one of the following:

```
oldstate = pause('on')
oldstate = pause('off')
```

See the function reference page for pause for more information.

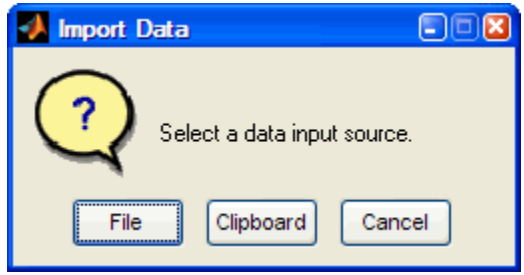
File Selection Restriction in Import Wizard

This release introduces a change in how you select the source to import using the Import Wizard. In previous releases, you could select and view the contents of any number of files within the wizard before choosing the one to import. As of this release, if you need to import from a different source (a specific file or the system clipboard) than the one you had originally selected, you must exit and restart the Import Wizard. This change gives the Import Wizard increased flexibility in handling different types of files, removes redundancy in the import process, and also removes a potential source of unexpected behavior from the product.

Compatibility Considerations

The user interface to the Import Wizard is very much the same as in previous versions of MATLAB. However, you should take note of the following changes:

- The panel named **Select Data Source**, that appears at the top of the Import Wizard preview dialog box in earlier releases, is no longer available. As before, you select the source file to import, or the clipboard, at the time you activate the Import Wizard. This applies whether you use **File > Import Data** from the MATLAB Command Window menu or the uiimport function at the command prompt. To make a new file or clipboard selection, click the **Cancel** button in any of the dialog boxes, and then restart the Wizard.
- Calling `uiimport` without a file name displays the following new dialog box. Choosing **File** opens the Import Data dialog box. Selecting **Clipboard** opens the wizard with the contents of the clipboard in the preview panel.



- To import from the clipboard using the MATLAB menus, use **Edit > Paste to Workspace** instead of using **File > Import Data** , and then change the source to **Clipboard**. This method is preferred because it is more direct. The capability of switching from file to clipboard within the wizard is no longer supported.

Function Handle Array Warning Is Now An Error

The only way to make an array of function handles is to use a cell array. Attempting to create any other type of array of function handles is invalid. For the past several releases, MATLAB has issued a warning if you should attempt to put function handles into any type of array other than a cell array. As of this release, MATLAB throws an error instead of a warning when this is attempted.

Replace

```
A = [@sin @cos @tan];
??? Error using ==> horzcat
Nonscalar arrays of function handles are not allowed; use
cell arrays instead.
```

with

```
A = {@sin @cos @tan};
```

Compatibility Considerations

If any of your program code attempts to create a regular array of function handles, this code will now generate an error.

Two Types of issorted Warnings Are Now Errors

In previous versions, the issorted function generated a warning for the following two cases:

- `issorted(x)`, where `x` is a complex integer, and
- `issorted(x,'rows')`, where `x` is an ND array

Using issorted on a Complex Integer

In this case, a statement such as the following

```
issorted(int8(complex(1,2)))
```

now issues the error message

```
??? Error using ==> issorted
ISSORTED on complex inputs with integer class is obsolete.
Please use ISSORTED(DOUBLE(X)) or ISSORTED(SINGLE(X)) instead.
```

Using issorted on an N-D Array

In this case, a statement such as

```
issorted(ones(3,3,3),'rows')
```

now issues the error message

```
??? Error using ==> issorted
X must be a 2-D matrix.
```

Compatibility Considerations

If any of your program code attempts to use either of these types of statements, this code will now generate an error.

Possible Conflict with New Keyword: SPMD

This release introduces a new keyword, `spmd`, that, although used solely by the Parallel Computing Toolbox (PCT), may cause conflicts with MATLAB users as well. See spmd Construct in the PCT release notes for more information on this keyword.

Compatibility Considerations

Because `spmd` is a new keyword, it will conflict with any user-defined functions or variables of the same name. If you have any code with functions or variables named `spmd`, you must rename them.

Do Not Create MEX-Files with DLL File Extensions

In the future, on 32-bit Windows systems, MATLAB will not support MEX-files with a `.dll` file extension. See release note "Do Not Use DLL File Extensions for MEX-Files" on page 16-38 for information on how this might affect you.

isequal Is Now Called Explicitly for Contained Objects

When you pass two MATLAB objects to isequal, MATLAB dispatches to the `isequal` method of the dominant object (see Object Precedence in Expressions Using Operators). If the dominant object does not overload `isequal`, then MATLAB uses the built-in version.

Compatibility Considerations

What is different with this release, is that MATLAB now calls `isequal` explicitly for each contained object. This means that, if any contained object overloads `isequal`, MATLAB calls the overloaded version for that object.

Previously, MATLAB compared contained objects using the built-in `isequal` functionality without regard to any special behavior programmed into overloaded `isequal` methods. The effect of this change is that, for objects that contain other objects and those contained objects overload `isequal`, the overloaded behavior establishes the basis with which MATLAB determines equality for those contained objects.

Indexed Assignment with Objects of the Form p(:) = o Now Consistent with MATLAB Language

The behavior of indexed assignment with MATLAB objects is consistent with the behavior of all MATLAB intrinsic types and V5 MATLAB objects. For example, attempting the following assignment, where `d` does not previously exist, gives an error:

```
>> d(:) = 5;
```

```
???  In an assignment  A(:) = B, the number of elements in A and B
must be the same.
```

MATLAB objects behave in the same way:

```
ts_obj = timeseries;
t(:) = ts_obj;
???  In an assignment  A(:) = B, the number of elements in A and B
must be the same.
```

Compatibility Considerations

In MATLAB Version 7.6 Release 2008a, indexed assignment of the form `p(:) = object` did not result in an error.

fopen No Longer Supports VAXD, VAXG, and Cray Machine Formats

Calls to fopen with any of the following values for machine format return an error:

- `'vaxd'` or `'d'`
- `'vaxg'` or `'g'`
- `'cray'` or `'c'`

Compatibility Considerations

In previous releases, calls to `fopen` with machine format values associated with VAXD, VAXG, and Cray did not result in an error.

To read files in VAXD and VAXG formats, consider the workaround available on the MATLAB Central File Exchange, file ID #22675.

Graphics and 3-D Visualization

Certain Printer Formats and Drivers Now Warn When Used

Going forward, MathWorks is planning to leverage existing operating system (OS) support for printer drivers and devices. As a result, the ability to specify certain print devices using the `print -d` command, and certain graphics formats using the `print -d` command and/or the `saveas` command, will be removed in a future release.

Graphic Format Drivers		
`-pkm`	`-pkmraw`	`-tifflzw`

Printer Driver	`print` Command Option String
Canon® BubbleJet BJ10e	`-dbj10e`
Canon BubbleJet BJ200 color	`-dbj200`
Canon Color BubbleJet BJC-70/ BJC-600/BJC-4000	`-dbjc600`
Canon Color BubbleJet BJC-800	`-dbjc800`
Epson® and compatible 9- or 24-pin dot matrix print drivers	`-depson`
Epson and compatible 9-pin with interleaved lines (triple resolution)	`-deps9high`
Epson LQ-2550 and compatible; color (not supported on HP®-700)	`-depsonc`
Fujitsu® 3400/2400/1200	`-depsonc`
HP DesignJet 650C color (not supported on Windows OS)	`-ddnj650c`
HP DeskJet 500	`-ddjet500`
HP DeskJet 500C (creates black and white output)	`-dcdjmono`
HP DeskJet 500C (with 24 bit/pixel color and high-quality Floyd-Steinberg color dithering) (not supported on Windows OS)	`-dcdjcolor`

Printer Driver	`print` Command Option String
HP DeskJet 500C/540C color (not supported on Windows OS)	`-dcdj500`
HP Deskjet 550C color (not supported on Windows OS)	`-dcdj550`
HP DeskJet and **DeskJet Plus**	`-ddeskjet`
HP LaserJet	`-dlaserjet`
HP LaserJet+	`-dljetplus`
HP LaserJet IIP	`-dljet2p`
HP LaserJet III	`-dljet3`
HP LaserJet 4.5L and 5P	`-dljet4`
HP LaserJet 5 and 6	`-dpxlmono`
HP PaintJet color	`-dpaintjet`
HP PaintJet XL color	`-dpjxl`
HP PaintJet XL color	`-dpjetxl`
HP PaintJet XL300 color (not supported on Windows OS)	`-dpjxl300`
HP-GL® for HP 7475A and other compatible plotters. (Renderer cannot be set to Z-buffer.)	`-dhpgl`
IBM® 9-pin Proprinter	`-dibmpro`

Compatibility Considerations

The following Web site provides more detailed information on the file types that will issue warnings and how to suppress the warnings if desired, as well as a form to provide feedback to MathWorks about your use of these options:

http://www.mathworks.com/support/contact_us/dev/obsoleteprintdevices.html

Handle Graphics Not Supported Under -nojvm Startup Option

If you start MATLAB with `matlab -nojvm` (which disables Java) you will receive a warning when you attempt to create or load figures, open GUIs, print or capture figures using `getframe`.

Compatibility Considerations

For information, see “Changes to -nojvm Startup Option” on page 16-4 in the Desktop Tools and Development Environment release notes.

Creating Graphical User Interfaces (GUIs)

Undocumented Functions Removed

The following set of deprecated functions, all of which were previously undocumented, have been removed. Alternatives to most of them exist, which are described.

- `axlimdlg` — No alternative
- `cbedit` — No alternative
- `clruprop` — Use `rmappdata` instead
- `ctlpanel` — No alternative
- `edtext` — Set text object's `Editing` property
- `extent` — Get text object's `Extent` property
- `getuprop` — Use `getappdata` instead
- `hidegui` — Set figure's `HandleVisibility` property
- `hthelp` — Use `web` instead
- `layout` — No alternative
- `matq2ws` — Combine `save` + `load` or `uisave` + `uiload`
- `matqdlg` — Combine `save` + `load` or `uisave` + `uiload`
- `matqparse` — Combine `save` + `load` or `uisave` + `uiload`
- `matqueue` — Combine `save` + `load` or `uisave` + `uiload`
- `menubar` — The string `'none'`
- `menuedit` — No alternative
- `pagedlg` — Use `pagesetupdlg` instead
- `setuprop` — Use `setappdata` instead
- `umtoggle` — Set uimenu's `Checked` property
- `wizard` — Use `guide` instead
- `ws2matq` — Combine `save` + `load` or `uisave` + `uiload`

Compatibility Considerations

If you have developed MATLAB code that uses any of the above undocumented functions, you must remove such calls or replace them with other code, as suggested within the above bullets.

Handle Graphics Not Supported Under -nojvm Startup Option

If you start MATLAB with `matlab -nojvm` (which disables Java) you will receive a warning when you attempt to create or load figures, open GUIs, print, or capture figures using `getframe`. For more information, see “Changes to -nojvm Startup Option” on page 16-4 in the Desktop Tools and Development Environment release notes.

New Menu Options to Hide or Show GUIDE Toolbar and Status Bar

Two new menu options in GUIDE let you hide/show the GUIDE toolbar and status bar. By default both are visible, but you can deselect **Show Toolbar**, **Show Status Bar**, or both from the **View** menu to make the layout area larger.

Compatibility Considerations

The GUIDE Preferences option **Show Toolbar** is no longer available. Use **Show Toolbar** from the GUIDE **View** menu instead.

GUIDE Status Bar Now Shows Tag Property of Selected Object

The GUIDE Layout Editor now shows the `Tag` property of any object you select, in the left corner of the status bar that runs along its bottom. This display saves you from having to open the Property Inspector to read `Tag` names. Information in the status bar is read-only; you still need to use the Property Inspector to change a component's `Tag`.

Four New Major GUI Examples

The Creating Graphical User Interface documentation has four new extensive examples of building GUIs with GUIDE and programmatically. All of them feature uitables, a feature introduced in R2008a, and all the GUIs plot data. The new examples are:

- A Working GUI with Many Components (GUIDE)
- GUI for Animating a 3-D View (GUIDE) (GUIDE)
- GUI to Interactively Explore Data in a Table (GUIDE) (GUIDE)
- GUI that Displays and Graphs Tabular Data (programmatic)

This release includes FIG- and code files for all these examples. The documentation discusses many of their callbacks and includes hyperlinks to code in the code files.

External Interfaces/API

Do Not Use DLL File Extensions for MEX-Files

In the future, on 32-bit Microsoft Windows systems, MATLAB will not support MEX-files with a `.dll` file extension. In MATLAB Version 7.7 (R2008b), if you run a MEX-file with a `.dll` file extension, MATLAB displays a warning.

Additionally, if you use the mex function with the `-output` switch to create a MEX-file with a `.dll` extension, MATLAB displays a warning. If you use the `-output` switch to name a MEX-file, you do not need to provide a file extension. MATLAB automatically appends the appropriate extension. For example, the following command creates a MEX-file named `newtest.mexw32`:

```
mex mytest.c -output newtest
```

Compatibility Considerations

You must recompile MEX-files with a `.mexw32` file extension. This is the default of the `mex` command.

MEX-Files Must Be Recompiled When -largeArrayDims Becomes Default MEX Option

In a future version of MATLAB, the default mex command will change to use the large-array-handling API. This means the `-largeArrayDims` option will be the default. For information about migrating your MEX-files to use the large-array-handling API, see the Technical Support solution 1-5C27B9.

Compatibility Considerations

In the near future you will be required to update your code to use the new API and to recompile your MEX-files. You should review your source MEX-files and `mex` build scripts.

New Compiler Support

MATLAB Version 7.7 (R2008b) supports these new compilers for building MEX-files:

Microsoft Windows 64-bit and 32-bit Platforms

- Microsoft Visual Studio 2008 Express Edition

Compiler Support to Be Phased Out

The following compilers are supported in Version 7.7 (R2008b), but will not be supported in a future version of MATLAB.

Windows (32-bit) platform

- Intel Visual Fortran Version 9.1
- Microsoft Visual Studio .NET Version 7.1
- Open Watcom Version 1.3

Windows (64-bit) platforms

- Intel Visual Fortran Version 9.1

Solaris SPARC (64-bit) platform

- Sun Studio 11 cc / CC Version 5.8
- Sun Studio 11 f90 Version 8.2

Use mxDestroyArray to Release Memory for mxArray

The documentation for the mxSetCell, mxSetField, and mxSetFieldByNumber functions in the MATLAB C and Fortran API incorrectly instructs customers to use mxFree to release memory for any `mxArray` returned by mxGetCell, mxGetField, or mxGetFieldByNumber.

Compatibility Considerations

The correct function to use to release memory for an `mxArray` is mxDestroyArray. Calling `mxFree` on an `mxArray` only frees the array header, but does not actually free the data itself and can result in a memory leak.

To help diagnose this problem, MATLAB issues a warning if calling `mxFree` on an `mxArray` could cause memory corruption. In future versions of MATLAB, this condition might result in a segmentation violation.

New Function Displays Information about MEX Compiler Configurations

The mex.getCompilerConfigurations function displays information about the selected compiler and associated switches and options that MATLAB uses to build a MEX-file. The selected compiler is the one you choose when you run the mex `-setup` command. For more information, see Building MEX-Files.

New Functions to Catch Errors in MEX-Files Replace mexSetTrapFlag

Two new MEX library functions have been added to the MATLAB C and Fortran API.

The mexCallMATLABWithTrap function, like `mexCallMATLAB`, lets you call MATLAB functions from within a MEX-file. In addition, `mexCallMATLABWithTrap` lets you catch, or trap, errors. Using this function for exception handling is more flexible that using `mexCallMATLAB` with the `mexSetTrapFlag` function.

Likewise, the mexEvalStringWithTrap function adds error handling to the `mexEvalString` function.

Compatibility Considerations

In the near future you will be required to update your MEX-files to remove use of the mexSetTrapFlag function.

"Duplicate dylib" Warning on Macintosh Systems

When compiling MEX-files on an Apple Mac OS Version 10.5 system you can ignore a warning about a duplicate library `libz.1.dylib`. The MEX-file builds properly and runs as expected. The warning message contains the following information:

```
ld: warning, duplicate dylib
```

Microsoft Visual Studio "X64 Compilers and Tools" Required for 64-bit Systems

If you use Microsoft Visual Studio with MATLAB on 64-bit systems, you must choose "X64 Compilers and Tools" when you install the following products:

- Visual Studio 2008 Express Edition

- Visual Studio 2008 Professional Edition
- Visual Studio 2005 Professional Edition

Run-Time Libraries Required for Applications Built with Microsoft Visual Studio 2008 Compiler

If you distribute a MEX-file, an engine application, or a MAT-file application built with the Visual Studio 2008 compiler, you must provide the Visual C++ run-time libraries. These files are required to run applications developed with Visual C++ on a computer that does not have Visual C++ 2008 installed. For information on locating the Microsoft Visual C++ 2008 Redistributable Package (x86), containing `vcredist_x86.exe` and `vcredist_x64.exe`, consult your Microsoft documentation.

Do Not Use get or set Function to Manage Properties of Java Objects

If you want to read or update a property of a Sun Java object created in MATLAB using the Java class constructor, do *not* use the MATLAB get or set functions on the property. For example, if you create a Java object called `javaObject` that has a property called `PropertyName`, the following commands might cause memory leaks and will be deprecated in a future version of MATLAB:

```
propertyValue = get(javaObject, 'PropertyName');
set(javaObject, 'PropertyName', newValue);
```

Compatibility Considerations

In future versions of MATLAB, using `get` or `set` on Java objects to manage the properties will generate an error. The correct commands to use are:

```
propertyValue = javaObject.getPropertyName;
javaObject.setPropertyName(newValue);
```

COM Objects Might Display Different Number of Supported Events

If you use events from COM servers implementing event interface versioning, COM objects created with MATLAB Version 7.4 (R2007a), Version 7.5 (R2007b), or Version 7.6 (R2008a) might have a different number of supported events than COM objects created with MATLAB Version 7.7 (R2008b) or any version prior to 7.4.

Compatibility Considerations

Calling the events (COM) function on affected types of COM server components returns a list of events for the latest [default] interface version which might be different from the list of events displayed by MATLAB Version 7.4, Version 7.5, or Version 7.6.

If an event in an older COM event interface version is no longer supported or renamed in the newer interface version, the registerevent function generates an error when such an event is used in an `.m` file.

R2008a

Desktop Tools and Development Environment

Desktop New Features Video

For an overview of the major new features in the MATLAB Desktop Tools and Development Environment area, watch this video demo. You can also access this and other video demos by selecting the **Demos** tab in the Help browser, and then selecting **MATLAB > New Features in Version 7.6**.

Startup and Shutdown

New features and changes introduced in Version 7.6 (R2008a) are:

Windows Platforms — Startup Changes, Including Use of My Documents/MATLAB or Documents/MATLAB Directory

On Microsoft Windows platforms, when MATLAB starts, it automatically adds the `My Documents/MATLAB` directory (or `Documents/MATLAB` on Windows Vista™) to the top of the MATLAB search path. This directory is known as the `userpath`. If you remove the `My Documents/MATLAB` (or `Documents/MATLAB`) directory from the search path and save the changes, either using the Set Path dialog box or using the `rmpath` and `savepath` functions, the MATLAB search path will not include the `userpath` directory at the next startup. On Windows platforms, the `userpath` is also the default startup directory.

Use the new function, `userpath`, to view the current value, clear the value so the directory is *not* on the search path and is *not* the startup directory, specify a different

value, or reset the value to the default. For more information, see Default Startup Directory (Folder) on Windows Platforms and the userpath reference page.

Compatibility Considerations

In previous versions, MATLAB automatically added the `My Documents/MATLAB` directory (or `Documents/MATLAB` on Windows Vista platforms) to the search path upon startup, even if you had removed it from the path and saved your changes during the previous session.

UNIX Platforms — Startup Changes

On UNIX platforms, when MATLAB starts, it automatically adds the *`userhome`*`/Documents/MATLAB` directory to the top of the MATLAB search path. This directory is known as the `userpath`. Use the new function, `userpath`, to view the current value, clear the value so the directory is *not* on the search path, specify a different value, or reset the value to the default. You can also specify that `userpath` be the MATLAB startup directory by setting the value for the environment variable `MATLAB_USE_USERWORK` to `1` prior to startup. For more information, see Startup Directory on UNIX Platforms and the userpath reference page.

Compatibility Considerations

In previous versions, no directories were added to the search path upon startup.

Macintosh Platforms — Startup Changes

On Apple Macintosh platforms, when MATLAB starts, it automatically adds the *`userhome`*`/Documents/MATLAB` directory to the top of the MATLAB search path. This directory is known as the `userpath`. Use the new function, `userpath`, to view the current value, clear the value so the directory is *not* on the search path, specify a different value, or reset the value to the default. If you start MATLAB from a shell, you can also specify that `userpath` be the MATLAB startup directory by setting the value for the environment variable `MATLAB_USE_USERWORK` to `1`.

For more information, see Startup Directory on <trademark Macintosh Platforms and the userpath reference page.

Compatibility Considerations

In previous versions, no directories were added to the search path upon startup.

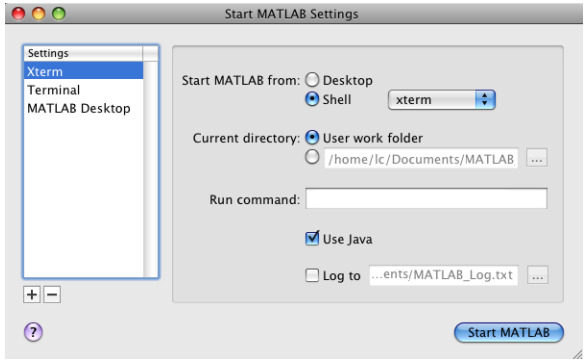
Macintosh Platforms — Define Startup Options Using New Dialog Box

On Apple Macintosh platforms, you can specify startup options using the new Start MATLAB Settings dialog box. The first time you start MATLAB Version 7.6 (R2008a), the Start MATLAB Settings dialog box opens automatically. It does not open automatically on subsequent startups; to open the dialog box, double-click `Start MATLAB Settings`, located in the same directory as the MATLAB application.

You can set these options and others:

- How you will interact with MATLAB (desktop or specified shell)
- The startup directory; the default is *`userhome`*`/Documents/MATLAB`
- Statement MATLAB runs upon startup

You can create and save multiple startup options files, each with different settings. For more information, see Startup Options.



Macintosh Platforms — Run Startup Diagnostics

Upon startup on Macintosh platforms, MATLAB automatically runs diagnostics to ensure your system has the required X11 and Sun Microsystems Java applications. If there are no problems, MATLAB starts as expected. If there is a problem, MATLAB displays a dialog box informing you of the problem and how to address it. You can run these diagnostics manually from the Start MATLAB Settings dialog box, using items in the **Diagnostics** menu.

Updated Version of JVM Software on Solaris Platform

MATLAB is now using Sun Microsystems JVM Version 6 on the Sun Microsystems Solaris platform.

Compatibility Considerations

If you use a specific version of Sun Microsystems Java with MATLAB on the Solaris platform, this change might impact your work.

Changes to Abnormal Termination Process

When MATLAB encounters a serious problem, such as a segmentation violation, a dialog box opens to notify you about the problem. From the dialog box, you can close MATLAB, or try to save your work in progress before closing.

If you try to save your work in progress, be aware that MATLAB is in an unreliable state and you should exit MATLAB as soon as you finish saving your work. The Command Window displays the message `Please exit and restart MATLAB` to the left of the prompt, which reminds you to discontinue use. For more information, see Abnormal Termination.

Compatibility Considerations

In previous versions, when MATLAB encountered a serious problem, an error message appeared in the Command Window that instructed you to close MATLAB.

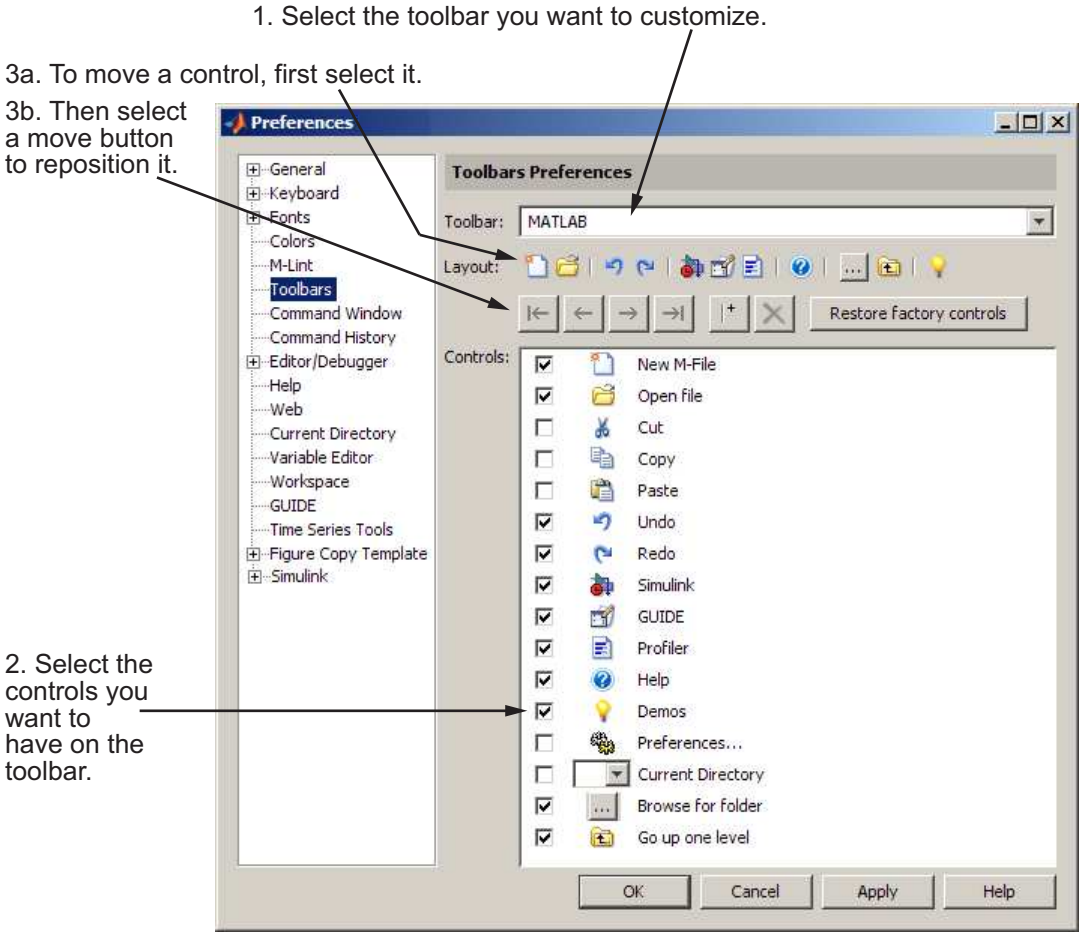
With multithreaded computation enabled, a platform-specific dialog box appeared, from which you immediately closed MATLAB.

Desktop

New features and changes introduced in Version 7.6 (R2008a) are:

Customize the MATLAB Desktop and Editor Toolbars

Rearrange, add, or remove buttons and other controls from the MATLAB desktop or Editor toolbars using **File > Preferences > Toolbars**. Alternatively, right-click a toolbar and select **Customize** from the context menu.



For more information, see Toolbar Customization.

There are new buttons you can add to the MATLAB desktop toolbar:

- Preferences, , which displays the Preferences dialog box, open to the pane last used
- Demos, , which displays the listing of Demos in the Help browser

You can also add Save All and Save As buttons to the Editor toolbar.

You can change the position of the toolbars within a tool, for example, putting both the Editor and Editor cell mode toolbars next to each other instead of stacked. To move a toolbar, grab the anchor for a toolbar, then drag the toolbar to the new location.

Clear a Browser with New Method

Clear an open browser in MATLAB with a new `close` method. For example, open a browser to display the MathWorks Web site by running `[stat,h1]=web('http://www.mathworks.com')`. Then `close(h1)` clears `mathworks.com` from that browser window. For more information, see the reference page for the web function.

Manage Your License

You can use new licensing features to perform license management activities, such as activating your license, deactivating your license, or updating your license. You can also visit the License Center at the MathWorks Web site to perform other license-related activities. Access the features by selecting **Help** > **Licensing**.

Check for Updates Feature Enhanced

When you select **Help** > **Check for Updates**, the dialog box now allows you to see the latest versions for all MathWorks products, or just those you install. You can also access release notes for each product from the dialog box. For more information, see Check for Software Updates.

Running Functions — Command Window and History

Command History Preference — Default Value Changed

The default value for the Command History preference, **Save after *n* commands**, is now 1. This allows you to more easily rebuild your state in MATLAB if MATLAB terminates abnormally, such as after a power failure. For more information about this preference, see Command History Preferences.

Compatibility Considerations

Previously, the default value for the **Save after *n* commands** preference was 5.

Help

Preferences for Help on Selection

When you click a function name in the Command Window or Editor, and then press **F1** or select **Help on Selection** from the context menu, the help appears in a pop-up window by default. Now, you can specify that the help appear in the Help browser rather than in a pop-up window via the new **Help on Selection** preference. For more information, see Help Preferences.

Slight Reordering of Products in Help Browser

In the Help browser, the order of some documentation names in the **Contents** pane and the Product Filter preference dialog box has changed slightly. Documentation now appears in this order:

- Release Notes (general)
- Installation
- MATLAB
- Toolboxes and other products based on MATLAB, arranged alphabetically
- Simulink
- Blocksets and other products based on Simulink, arranged alphabetically
- Link and target products, arranged alphabetically

Workspace, Search Path, and File Operations

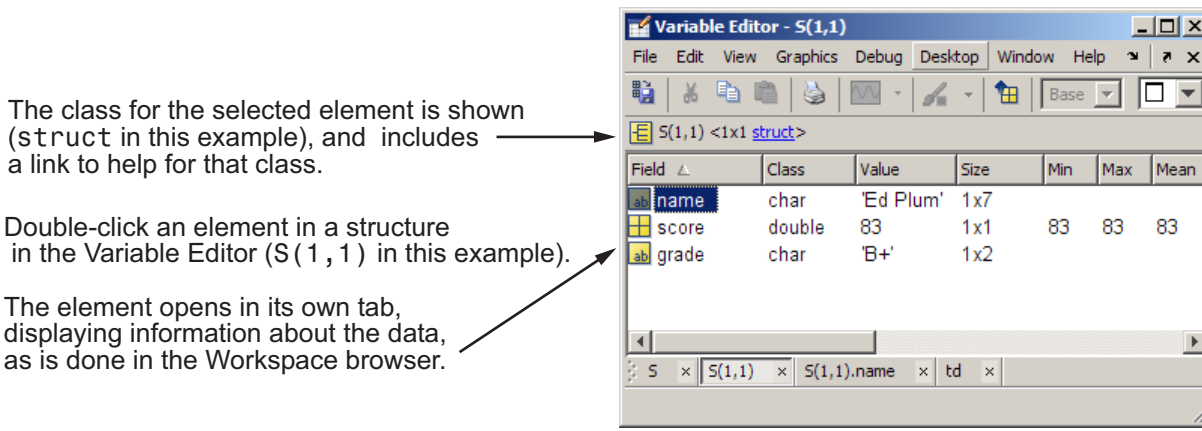
New features and changes introduced in Version 7.6 (R2008a) are:

- "Array Editor Renamed to Variable Editor; Offers Enhanced Support for Structures and Classes" on page 17-9
- "Search Path — Changes to User Portion" on page 17-10
- "New Context Menu Options in Current Directory Browser" on page 17-11
- "File and Directory Comparisons Tool" on page 17-11

Array Editor Renamed to Variable Editor; Offers Enhanced Support for Structures and Classes

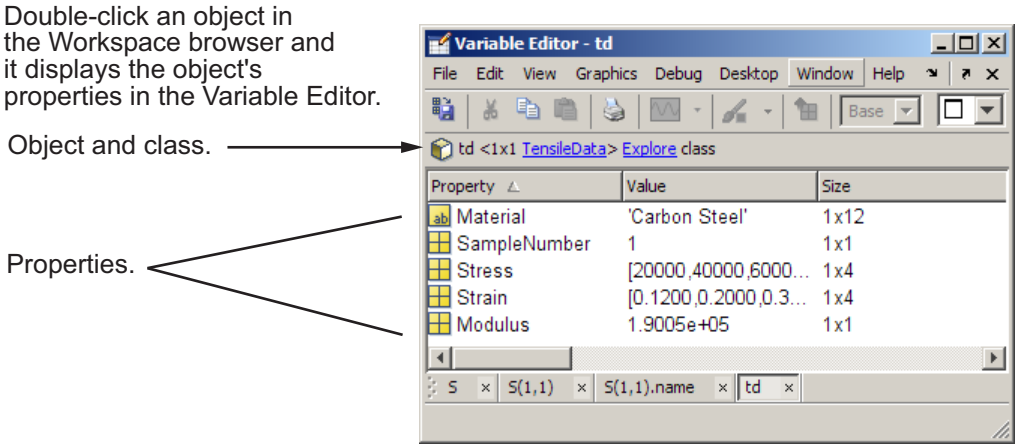
The Array Editor has been renamed to the Variable Editor, which better reflects its support for non-array data such as structures and properties.

The Variable Editor now reports, just below the toolbar, the class and size of the selected variable. For many classes, there is also a link to help for the class. To view a structure in the Variable Editor, double-click one of its elements. The resulting display is much like the display of that element in the Workspace browser, showing the Class, Value, Size, Min, Max, and other information. For more information about the Variable Editor, see Viewing and Editing Workspace Variables with the Variable Editor.



Use the data brushing button on the Variable Editor toolbar  to mark observations on graphs and then remove or save them to new variables. For more information, see “Data Brushing for Graphs and Linked Variables” on page 17-30.

The Variable Editor supports the new MATLAB class system. Double-click an object in the Workspace browser, and it opens in the Variable Editor. The Variable Editor displays the object, the class, and the properties of the object.



Compatibility Considerations

In previous versions, the Variable Editor was called the Array Editor. Starting in R2008a, MATLAB Version 7.6, the tool will be referred to as the Variable Editor.

Search Path — Changes to User Portion

By default, MATLAB adds a directory to the top of the search path upon startup, known as the `userpath` directory. By default, its value is `Documents/MATLAB` on Windows platforms, or `My Documents/MATLAB` on Windows Vista platforms. On UNIX and Macintosh platforms, the default directory is *`userhome`*`/Documents/MATLAB`. Use the new function, `userpath`, to specify a different directory, clear the `userpath` value, or reset it to the default value. On UNIX and Macintosh platforms, you can specify additional directories for MATLAB to add to the top of the search path upon startup using the `MATLABPATH` environment variable. When you remove the `userpath` portion of the search path, it clears the value for `userpath` and might impact the startup directory. For related information, see “Startup and Shutdown” on page 17-2. For details see Locations for Storing Your Files and the userpath reference page.

Compatibility Considerations

In previous versions, on Windows platforms, MATLAB added the default directory to the search path upon startup, even if you removed it and saved the changes to the path. On UNIX and Macintosh platforms, MATLAB did not add a directory to the search path on startup.

New Context Menu Options in Current Directory Browser

The context menu, which you access by right-clicking anywhere within the Current Directory browser, provides these three new options for creating M-files in the current directory:

- **New > Blank M-File**

  Creates an empty M-file
- **New > Function M-File**

  Creates an M-file with a template for writing an M-file function
- **New > Class M-File**

  Creates an M-file with a template for writing an M-file class definition

Compatibility Considerations

The **New > M-File** option is replaced by the **New > Function M-File** option, which has the same effect.

File and Directory Comparisons Tool

The File Comparisons tool is now called the File and Directory Comparisons Tool. In addition to enabling you to compare lines in two text files, it now enables you to:

- Compare variables in two MAT files
- Determine whether the contents of two binary files are the same
- Compare two directories to determine which file names are unique to each directory
- Compare two directories to determine if files with the same name in each directory have the same content

See Comparing Files and Folders for details.

Tuning and Managing M-Files

Profiling — Setting Intel Multi-Core Processors

If your system uses Intel multi-core chips, and you plan to profile using CPU time, set the number of active CPUs to 1 before you start profiling. See Intel Multi-Core Processors — Setting for Most Accurate Profiling on Windows Systems for details.

Editing and Debugging M-Files

New features and changes introduced in Version 7.6 (R2008a) are:

Stand-Alone Editor No Longer Provided

The MATLAB stand-alone Editor (`meditor.exe`) is no longer provided. Instead of the stand-alone Editor, you can use the MATLAB Editor.

Compatibility Considerations

Some users have preferred the stand-alone Editor to the MATLAB Editor because of slightly better startup performance and because it does not require a MATLAB software license. For those situations, you can use any text editor you have, such as UltraEdit or Emacs.

Run/Continue Button Now Two Separate Buttons

The Continue button is now separate from the Run button. Previously, the Run button served as both the Run and Continue button.

Compatibility Considerations

You now use the Run button to execute a run configuration and the Continue button to continue execution of an M-file after a breakpoint during debugging. See Run MATLAB Files in the Editor and Step Through a File for details.

Evaluate Entire File Button Off Toolbar by Default

The Evaluate Entire File button,  , is no longer on the Editor Cell Mode toolbar by default.

Compatibility Considerations

Previously, the Evaluate Entire File button was on the Editor Cell Mode toolbar by default. You can put the Evaluate Entire File button back on the toolbar by customizing it. See “Customize the MATLAB Desktop and Editor Toolbars” on page 17-5 for more information.

TLC and XML Syntax Highlighting Supported

You can specify preferences for TLC and XML syntax highlighting in the Editor/Debugger Language Preferences panel by selecting **File > Preferences > Editor > Language** and then, in the **Language** drop–down menu choosing **TLC** or **XML/HTML**.

Click **Help** in the Preferences dialog box for more information.

Code Folding Enhanced to Support More Language Constructs

You can enable code folding for all these programming constructs:

- Blocks of comments
- Cells used for rapid code iteration and publishing
- Class code
- Class enumeration blocks
- Class event blocks
- Class method blocks
- Class properties blocks
- For and parfor blocks
- Function and class help

- Function code
- If/else blocks
- Switch/case blocks
- Try/catch blocks
- While blocks

Prior to MATLAB Version 7.6 (R2008a), code folding was supported for function code and function help only.

See Code Folding — Expand and Collapse Code Constructs for details.

mlint Function Uses Preference Settings when Java Software is Available

When Sun Microsystems Java software is available, the `mlint` function honors the M-lint preferences that you specify using the M-Lint Preferences dialog box (**File** > **Preferences** > **M-Lint**).

In addition, the `'-config=settings.txt'` option to the `mlint` function enables you to override the current default M-Lint preferences settings with a settings file that you previously created and saved using the M-Lint Preferences dialog box. If you use the `'-config=settings.txt'` option, you must specify the full path to the file. If you prefer that m-lint ignore all M-Lint preferences and use the factory default settings instead, specify the `'-config=factory'` flag. See m-lint for details.

Compatibility Considerations

Previously, the `mlint` function always used the factory default settings, regardless of the M-Lint preferences that you set. To restore that behavior, use the `'-config=factory'` flag on the `mlint` function.

New M-Lint Warning Related to the MException Class

MATLAB Version 7.6 (R2008a) adds a new M-Lint warning related to the `MException` class. This warning, along with two M-Lint warnings added in MATLAB Version 7.5 (R2007b), intentionally make it difficult for you to completely ignore an error without receiving an M-Lint warning. It is a best practice to check error information even when an error is expected or frequent, so that you can rule out unexpected situations.

The three messages are as follows—the first is the one added in MATLAB Version 7.6 (R2008a):

- `LASTERR and LASTERROR are better replaced by an identifier on the CATCH block. See doc CATCH.`
- `The value assigned here to variable 'x' might never be used.`

  This message appears when the code contains a `catch` statement that is never used.
- `TRY statement without a CATCH.`

For example, suppose you want to read options from a file, `options.txt`, but it is acceptable if that file is not present. You might write the following code, expecting the `read_options` program to throw an error if the file is not present:

```
options = {};
try
    options = read_options( 'options.txt' );
end
```

The problem with the preceding code is that the file might be present, but its permissions may prevent the program from reading it. The program ignores the file, and potentially confuses the user, who knows the file is there. Better code for accomplishing the task is as follows, which assigns a structure value to the variable `err` if an error is thrown in the `try` block. The structure value contains information about the error that was thrown.

```
try
    options = read_options( 'options.txt' );
catch err
    if strcmp( err.identifier, 'Program:ReadOptions:NoOptionsFile' )
        options = {};
    else
        rethrow( err );
    end
end
```

Using this code, if a problem other than the "file is missing" error occurs, MATLAB reports the error to the user. For instance, MATLAB reports an error if the file format is incorrect, or if the file has the wrong permissions.

If you feel comfortable ignoring the errors completely, it is probably best to use the `try` statement with no `catch` statement, and suppress the M-Lint warnings that result. You can suppress the warnings through the M-Lint preferences or by placing the `%#ok` pragma at the end of the line that triggers the message. However, The MathWorks suggests that if you suppress an M-Lint message, you include a comment in your code indicating why you think it is appropriate to ignore the message.

For more information about the `MException` class, see the Error Handling section in the MATLAB Programming Fundamentals documentation.

dbstop and dbclear Functions — Option to Specify File Not on Path

The `–completenames` option to the `dbstop` and `dbclear` functions enables you to set and clear breakpoints, respectively, for M-files that are not on the search path in MATLAB. See dbstop and dbclear for details.

edit Function Can Create New File in Existing Subdirectory

The edit function now allows you to specify a file that is not in the current directory. If the file does not exist, the edit function creates it in the directory you specify. However, the directory, must exist; the edit function will not create a directory for you. See edit for details.

Nest Cells for Rapid Code Iteration; Includes Changes to Cell Highlighting

You can nest cells in an M-file, including within functions and control statements, such as `for` loops and `if-then` blocks. This gives you greater control over how a published document appears. This nesting ability also enables you to evaluate subsections of code on a finer grain. See "Nest Cells for Finer Control" on page 17-19 for details.

Compatibility Considerations

With the introduction of nested cells, cells definitions result in cell highlighting that looks different from previous releases. See Define Code Cells for details.

Publishing M-Files

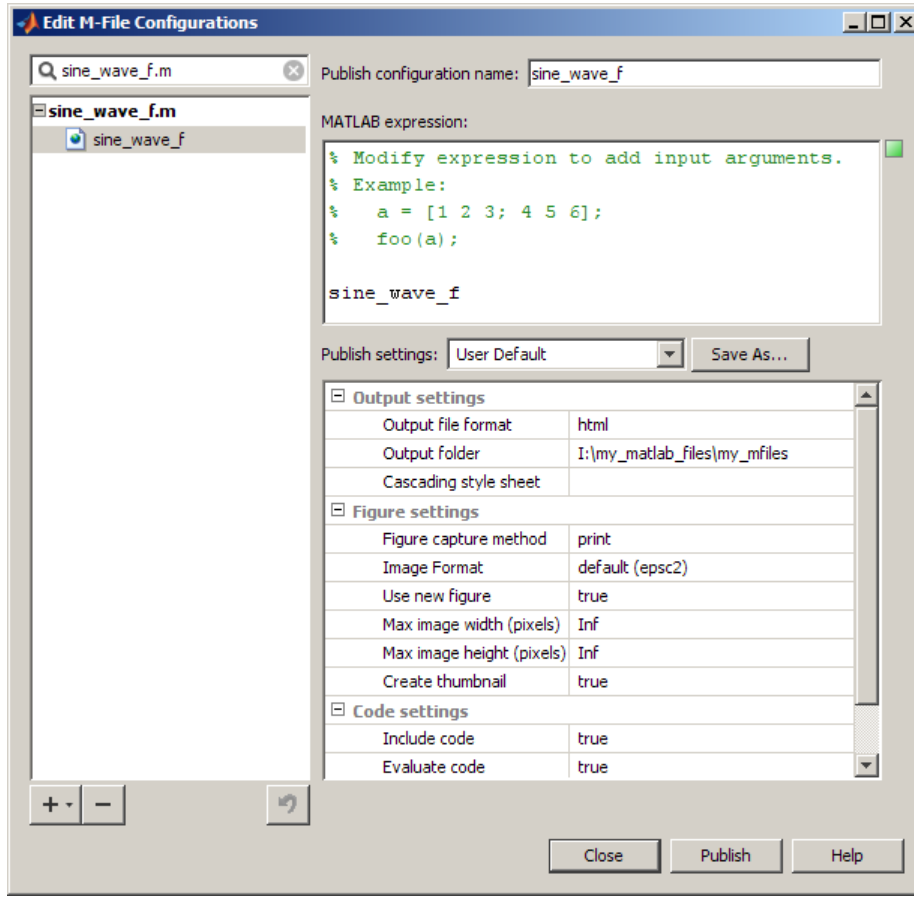
New features and changes introduced in Version 7.6 (R2008a) are:

Publish Functions and Scripts Using Publish Configurations; Includes Replacement of Publishing Preferences

In the Editor, you can now do the following when publishing M-file code:

- Specify code that you want MATLAB to evaluate before publishing the code, including input arguments for functions
- Specify publishing settings, such as an output directory and file format, that you can save and reuse as a group

To create a publish configuration, first open an M-file in the Editor. Then, select **File > Publish Configurations for *`filename`* > Edit Publish Configurations for *`filename`***. In the resulting Edit M-File Configurations dialog box, modify the **MATLAB expression**, specify **Publish settings**, and name the publish configuration. For more information, see Creating a Publish Configuration for a MATLAB File.



Compatibility Considerations

Previously, preferences for publishing and publishing images were available on the Preferences dialog box, which you accessed by selecting **File > Preferences > Editor/ Debugger** and then choosing the **Publishing** or **Publishing Images** node. Now you set these preferences when you create or update a publish configuration by using the **Publish settings** options on the Edit M-file Configurations dialog box.

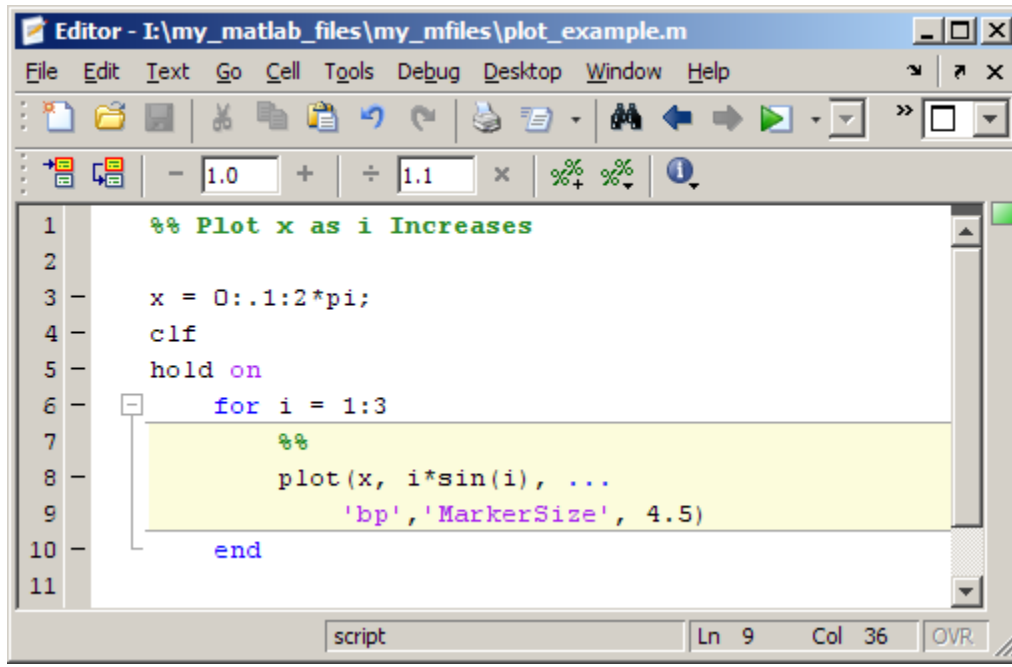
The Edit M-file Configurations dialog box continues to provide support for creating configurations that enable you to run an M-file. These are now called run configurations to differentiate them from publish configurations.

Nest Cells for Finer Control

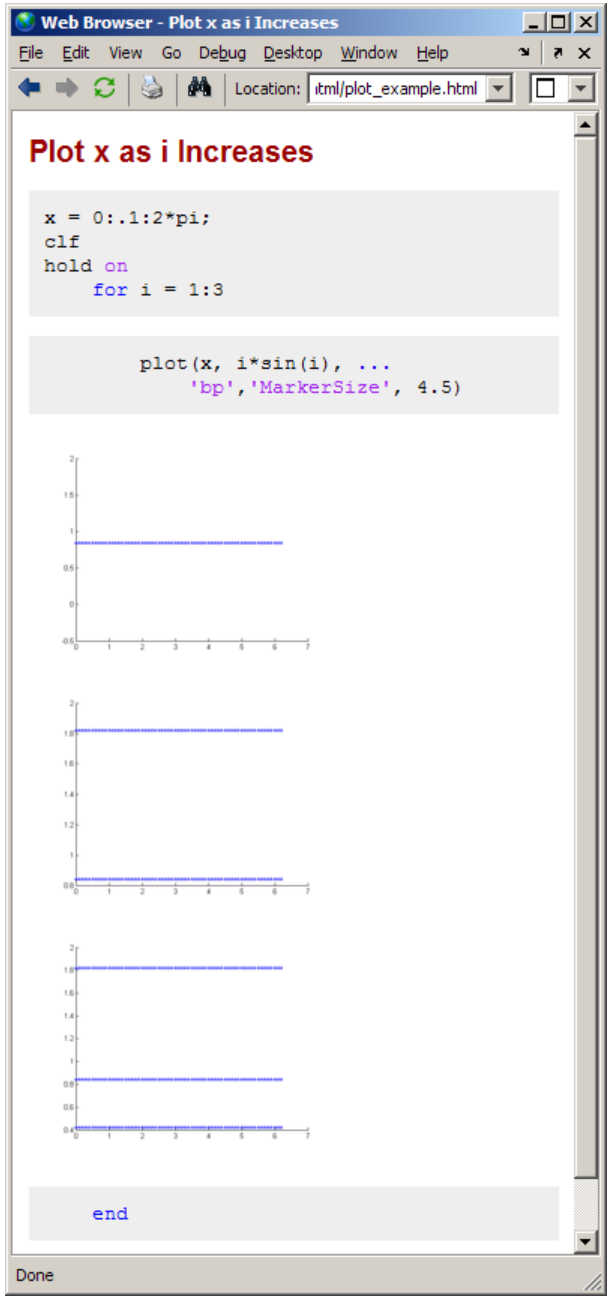
You can nest cells in an M-file, including within functions and control statements, such as `for` loops and `if-then` blocks. This gives you greater control over how a published document appears. This nesting ability also enables you to evaluate subsections of code on a finer grain when using rapid code iteration.

You can insert white space before the double percent (%%) characters that specify a cell break (which is also referred to as a cell divider). This helps to improve readability of the M-file when the cell break is within indented code. In prior releases, the %% characters had to be in the first column of the code.

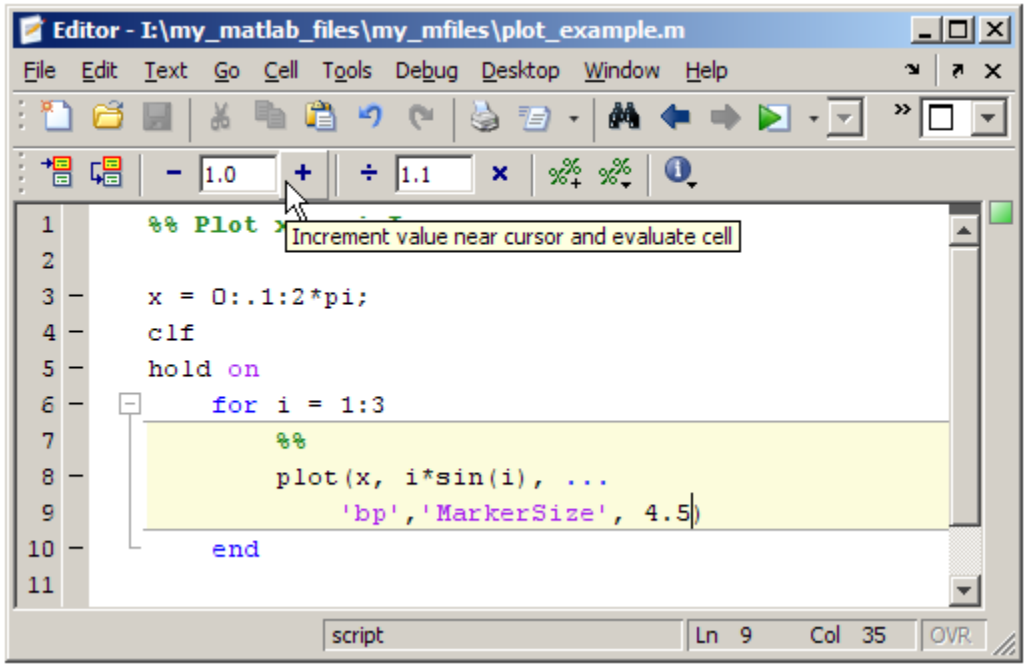
The following image shows a simple example of an M-file with nested cells.



If you publish the file to HTML (reducing the size of the images), the cell break nested within the `for` loop causes MATLAB to publish each iteration of the `for` loop as it evaluates the code in the loop:


Web Browser - Plot x as i Increases
File Edit View Go Debug Desktop Window Help
Location: itml/plot_example.html
Plot x as i Increases
x = 0:.1:2*pi;
clf
hold on
for i = 1:3
plot(x, i*sin(i), ...
'bp','MarkerSize', 4.5)
end
Done

Similarly, the cell break within the `for` loop enables you to run the M-file and experiment with the marker size value without the need to save the file between adjustments:



For more information see Mark Up MATLAB Code for Publishing and Evaluate Subsections of Files Using Code Cells.

Compatibility Considerations

In prior releases, the cell break characters (%%) had to be in the first column of the code for the Editor to recognize the characters as a cell break. This is no longer true, the Editor now recognizes these characters as a cell break regardless of the amount of white space that precedes them.

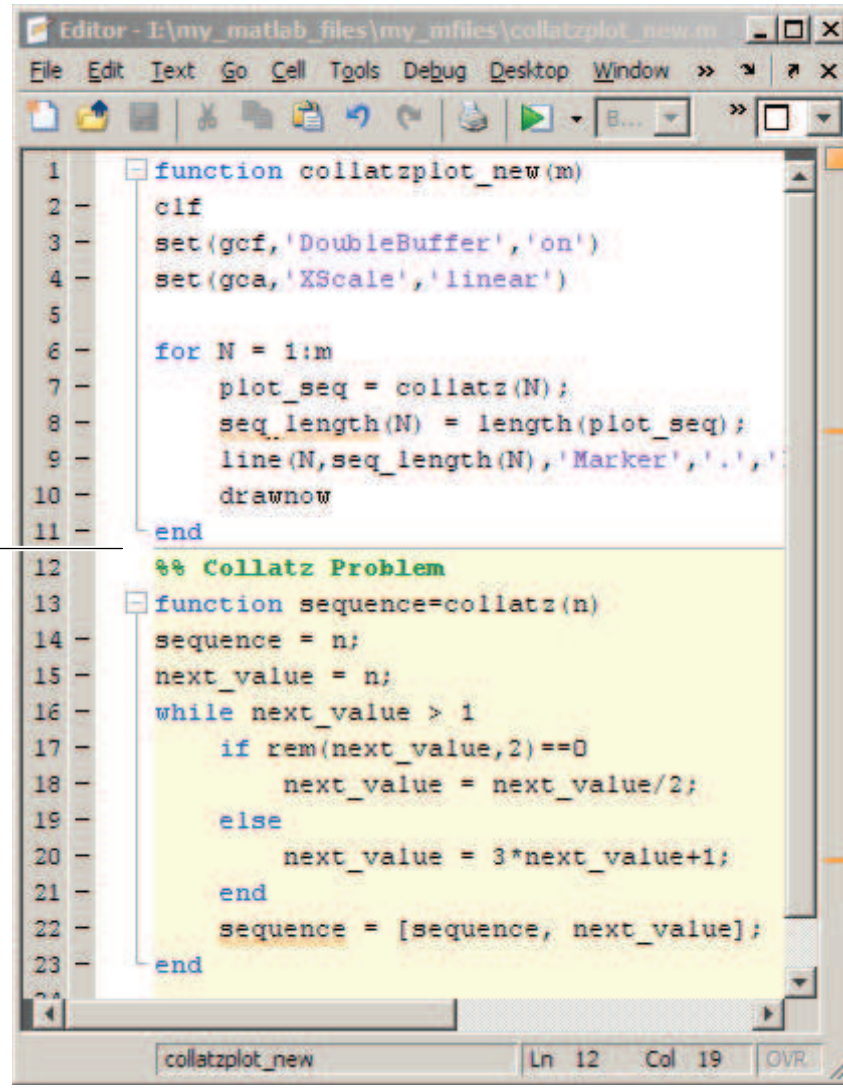
Furthermore, with the introduction of nested cells, inserting cell breaks in this release has different effects than in the previous release. In the previous release, if you inserted a cell break before a subfunction declaration, MATLAB created two cells; one above the

cell break and one below it. Now, if you insert a cell break, MATLAB also inserts implicit cell breaks.

In the example shown, it inserts two implicit cell breaks in the subfunction where you inserted the explicit cell break; one on the first line of the subfunction and one on the last line of the subfunction. This results in three cells: one containing the entire file, one containing the `collatzplot_new` function, and one containing just the Collatz Problem cell title.

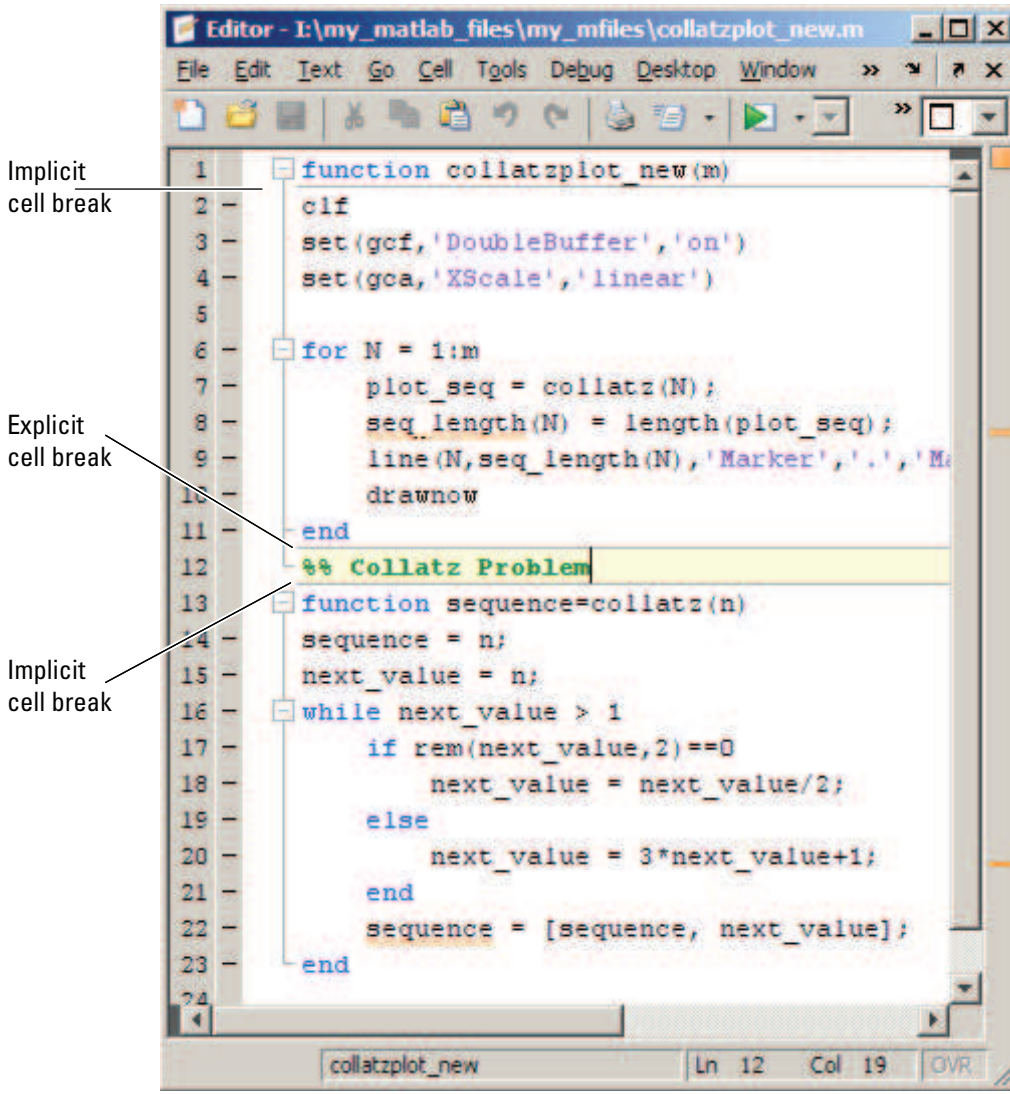
Nested cells are introduced to support function publishing.

Version 7.5 (R2007b)



Explicit cell break

Version 7.6 (R2008a)



See Nested Code Cells for details.

Publish Button Moved

The Publish button,  , is now located on the Editor toolbar.

Compatibility Considerations

Previously, the Publish button was located on the Editor Cell Mode toolbar. Now you find it on the Editor toolbar.

Publish Trademark Symbols

If the comments in your M-file include trademarked terms, you can format the comment to produce a trademark symbol (™) or registered trademark symbol (®) in the published output. See Trademark Symbols for details.

Specifying Code for MATLAB Software to Evaluate with the publish Function

Use the `codetoEvaluate` option to the `publish` function to specify code that you want MATLAB software to evaluate when it publishes an M-file. By default, this is the code in the M-file. However, if you want, you can use this option to specify additional code or alternative code for MATLAB to evaluate. For example, you might want MATLAB to evaluate code that calls the M-file that you are publishing.

stopOnError Option No Longer Available with publish Function

The `stopOnError` option is no longer available for the `publish` function. MATLAB software will always stop when an error occurs, unless you add code to handle the error.

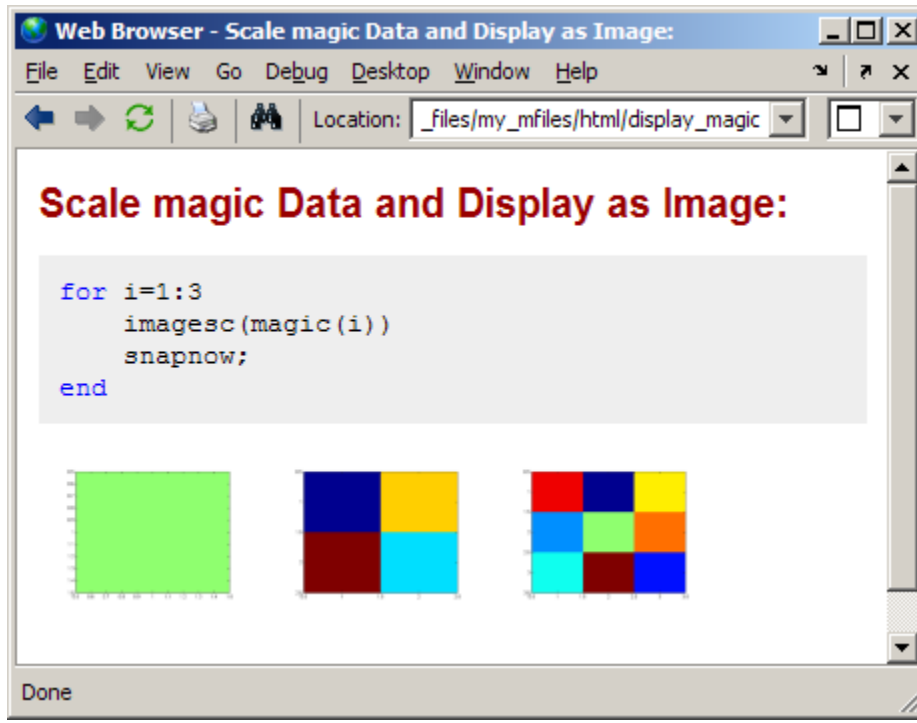
Compatibility Considerations

To have MATLAB continue processing code when an error occurs, handle the error using a `try-catch` statement. For more information, see The try-catch Statement.

Include Snapshot of M-file Output in Published Document

You can include snapshots of output that an M-file generates within a published document. Select **Cell > Insert Text Markup > Force Snapshot**. This menu option inserts the `snapnow` function into your M-file code. This is particularly useful when you have code that generates numerous images that you want to include in the published document. See Force a Snapshot of Output for details.

The following image, for example, shows a published document that uses this feature (with the size of the images reduced). Notice that the published images appear after the `for` loop that generates them.



Internationalization

Locale Information Added to MATLAB Documentation

Information about using locale in MATLAB can be found in Internationalization in the Desktop Tools and Development Environment documentation.

Changes to Locale Database

Windows Platform Changes

On Microsoft Windows systems, users can select the Pashto language with the Afghanistan country code. This locale setting is `ps_AF.1256`.

Macintosh OS X Platform Changes

On Apple Macintosh OS X systems, for users selecting the Chinese language and the China country code, the locale setting is `zh_CN.gb2312`. The previous setting was `zh_CN.GBK`.

Mathematics

Upgrade to BLAS Libraries

MATLAB software now uses new versions of the Basic Linear Algebra Subroutine (BLAS) libraries. For Windows, Intel Mac, and Linux platforms, MATLAB software supports the Intel Math Kernel Library (MKL) version 9.1. For the Solaris platform, MATLAB software uses the Sun Performance Library from Sun Studio 12.

Upgrade to LAPACK Library

MATLAB software now uses Version 3.1.1 of the Linear Algebra Package (LAPACK) library.

More Multithreaded Support For Elementwise Math Functions With Warnings

Multithreaded support has been added to elementwise math functions that may generate warnings: `rdivide`, `ldivide`, `log`, `log2`, and `rem`.

New Algorithms for ldl, logm, and funm Functions

The `ldl`, `logm`, and `funm` functions include new algorithms based on recent numerical methods research.

Functions and Properties Being Removed

Function or Property Name	What Happens When You Use Function or Property?	Use This Instead	Compatibility Considerations
`betacore`	Errors	betainc	Replace all existing instances of `betacore` with betainc.
`colmmd`	Errors	colamd	Replace all existing instances of `colmmd` with colamd.

Function or Property Name	What Happens When You Use Function or Property?	Use This Instead	Compatibility Considerations
`flops`	Errors	None	Remove all existing instances of `flops`. With the incorporation of LAPACK in MATLAB version 6, counting floating-point operations is no longer practical.
`symmmd`	Errors	symamd	Replace all existing instances of `symmd` with symamd.
`quad8`	Errors	quadl	Replace all existing instances of `quad8` with quadl.
`table1`	Errors	interp1 or interp1q	Replace all existing instances of `table1` with interp1 or interp1q.
`table2`	Errors	interp2	Replace all existing instances of `table2` with interp2.
`bessela`	Errors	besselj	Replace all existing instances of `bessela` with besselj.
beta using three input arguments	Errors	betainc using three input arguments	Replace all existing instances of `beta` using three input arguments with betainc using three input arguments.

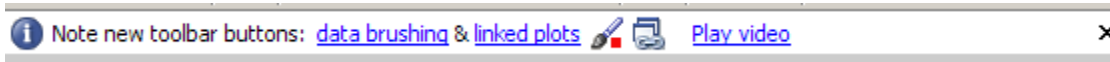
Data Analysis

Data Brushing for Graphs and Linked Variables

This release introduces two new interactive tools for data exploration:

- **Data brushing** — For marking observations on graphs, allowing you to remove or save them to new variables
- **Data linking** — For connecting graphs with data sources (workspace variables) to automatically and interactively update them

In addition, figure windows have a new banner, called the *linking and brushing message bar*. By default, when you plot data into a figure (i.e., add axes), an informational banner appears across top of the figure that looks like this.



In the figure's message bar, click the first two links to read about these new tools. Click the **Play video** link to open a nine-minute video tutorial about the tools in a browser window (the video also describes new GUI-building features.) To dismiss the banner, click the **X**. Once you do, the banner only reappears on subsequent plots if you select **Show linking and brushing message bar** in the MATLAB Preferences Confirmation Dialogs panel.

Note: The linking and brushing message bar can obscure a plot's title. Also, if you do not dismiss the message bar, it is visible in images of figures captured with getframe, but it does not print. See ??? for details.

Use data brushing when you want to isolate observations in a 2-D or 3-D graph for separate analysis, or to remove outliers or noisy data points. Data brushing can be

applied to most graphs (some plot types do not support brushing). Data brushing is an exclusive, persistent mode. That is, when using it, you cannot use other figure tools, but the results of brushing data persist when you select a different tool or no tool.

Use data linking to make plots dynamically respond to changes in the variables they plot. Data linking applies to most graphs with identifiable data sources and operates at the figure level. Data linking is not modal and persists until you toggle it off or the connection between a plot and its data sources is broken.

The two tools work smoothly together and with the Variable Editor to visually highlight brushed observations and the data values they represent:

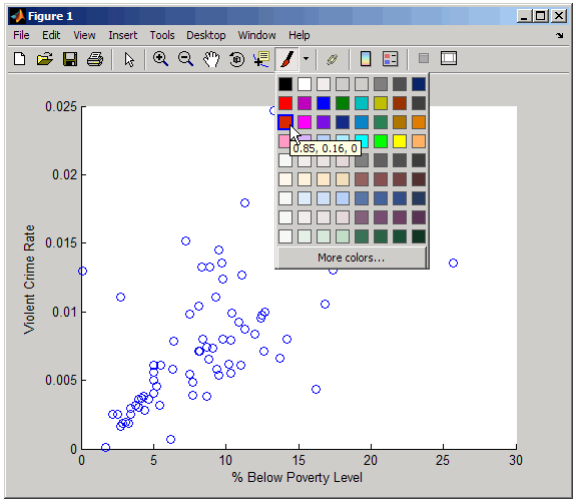
- Brushing data-linked observations on a graph highlights them on other graphs that display them.
- Brushing highlights values in the Variable Editor when a brushed variable is displayed there.
- Using the Brush tool in the Variable Editor highlights values you brush that appear in linked plots.
- Changing values of variables causes linked graphs displaying them to update with the changes.
- Clearing variables disconnects them from all linked figures displaying graphs of them

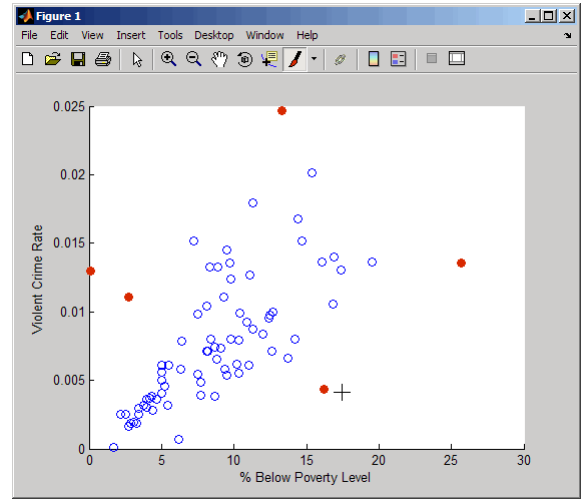
You can modify variables from the command line, the Variable Editor, or with M-files. When used within functions, data linking operates in the function's workspace, not the base workspace. This is also the case when debugging.

Data Brushing Tool

All figure windows that contain axes now include a data brushing tool (the Brush/Select data icon ) that lets you enter and exit brushing mode and select a color with which to brush observations. The tool draws selection rectangles (in 2-D plots) or prisms (in 3-D plots) and permits you to select discontiguous regions and negate previously brushed observations. Undo is also supported.

The Brush/Select data tool is a “split button” control with a brush icon on the left and a drop-down color palette on the right. When you depress the brush icon, you are in brushing mode; all data observations you select are highlighted with the current brush color. The figures below illustrate these operations.





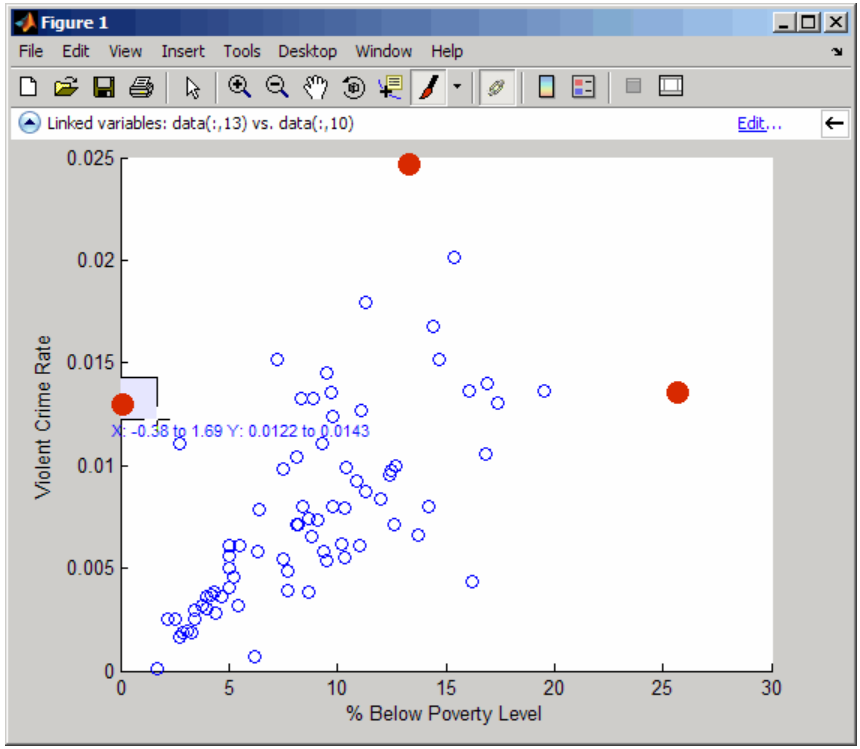
If you leave data brushing mode to zoom, pan, or edit the plot, all brushed observations remain highlighted. You can then reenter brushing mode and pick up where you left off. Brush marks are not preserved when you save a figure and reopen it from the FIG-file, however.

Data Brushing API

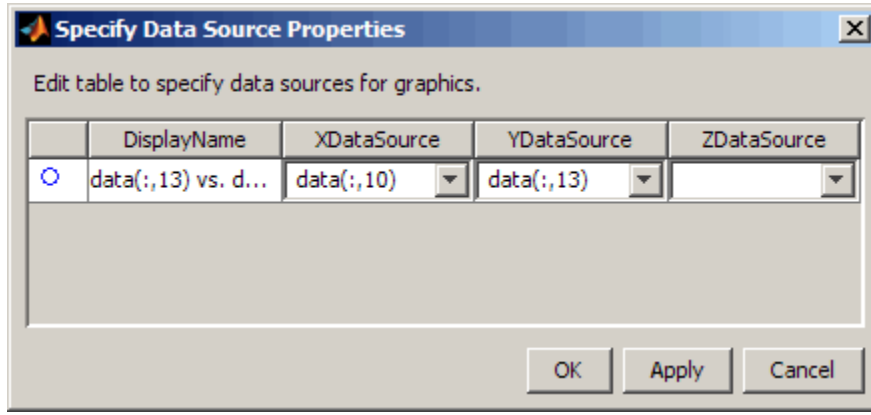
Use the brush function to turn brushing on and off, and to select a color for brushing graphs. You can change brush colors on the fly with either the API or with the Brush tool.

Data Linking Tool

All figure windows that contain axes now include a data linking tool (the Linked Plots button ) to toggle linked mode on and off (the default). When you toggle it on, an information bar appears underneath the lowest toolbar on the figure, as shown below. It displays what variables are linked to each series (data sources for *x*-, *y*-, and *z*- data in the graphs).



On the left side of the information bar is a drop-down menu  that displays the symbolism and identifies the data source for each series currently linked. On the right side is an **Edit** button that opens the Data Source Properties dialog box in which you can set display names and data sources. Usually it is possible to unambiguously determine what data sources a graph has, but sometimes you need to indicate what data source to use, for example, when you plot a subrange of a data array. The information bar explains that you need to do this as soon as you turn on data linking; then, you can open the Data Source Properties dialog box to identify your data source(s).



Data Linking API

Use the linkdata function to turn data linking on or off for the current figure or for a figure for which you supply a handle.

Compatibility Considerations

If you capture figure windows with the getframe function, their images will include the message bar if the figures being captured possess them. To prevent this from happening, click the **X** on the right side of the message bar to dismiss it before calling `getframe`. Subsequent figures will not display a message bar. If you want to restore the message bar at a later time, select **Show linking and brushing message bar** in the MATLAB Preferences Confirmation Dialogs panel.

If the figure has a title, its Linked Plots/Data Brushing message bar can obscure it. This is the case for figures at the default size. Remove the message bar if you want a title to display.

Programming

Multithreaded Computations Enabled

Multithreaded computations, introduced in R2007a, are now on by default.

Compatibility Considerations

To disable multithreaded computations, open the Preferences dialog, choose Multithreading, and then disable it explicitly.

Enhancements to Object-Oriented Programming Capabilities

Major enhancements to object oriented programming capabilities enables easier development and maintenance of large applications and data structures.

New features include:

- The new `classdef` keyword enables you to define properties, methods, and events in a class definition file. See User-Defined Classes.
- A new `handle` class with reference behavior enables you to create more sophisticated data structures, such as linked lists and to manage external resources, such as files. See Comparing Handle and Value Classes.
- Events and listeners enable you to monitoring object property changes and other actions. See Events — Sending and Responding to Messages.
- Packages enable scoping of classes and functions. See Create a Namespace with Packages.
- Meta-classes provide support for class introspection. See Information from Class Metadata.
- JIT/Accelerator support provides significantly improved performance over the previous object oriented-programming system.

For a full description of object-oriented features, see Object-Oriented Programming.

Packages for Classes and Functions

This release provides the capability to manage name space by placing classes and functions in packages.

Clear Variables with Exceptions

With the new clearvars function, you can specify which variables you do not want cleared from memory.

Information on the State of Memory

The new memory function provides memory usage information such as largest block available, allowing you to diagnose memory problems on Microsoft Windows platforms.

Compatibility Considerations

The memory function existed in previous versions of MATLAB, but its purpose has changed. Previously, `memory` provided help text on how to free additional memory space for your MATLAB application. The function now returns information on the current state of memory use and availability in your system.

Define Your Own Function Cleanup Tasks

With the new onCleanup function, you can specify one or more tasks for MATLAB to perform just before exiting the current function.

New Functions

Name	Description
clearvars	Clear variables from memory
memory	Display memory information
onCleanup	Cleanup tasks at function completion

Extended JIT Support

JIT/Accelerator support now extends to statements executed at the MATLAB Command Line and in cell mode in the MATLAB Editor. This provides improved performance in these environments.

Enhancements to Image Information and Writing Functions

The image information and writing functions have the following enhancements:

- imfinfo can now return Exif data for JPEG or TIFF format image files. Information specific to the digital camera can be found in the `'DigitalCamera'` field, while any global positioning system information can be found in the `'GPSInfo'` field.
- imwrite now supports the `'RowsPerStrip'` parameter that you can use to specify how many image rows to include in a strip when writing TIFF files. By default, `imwrite` limits the number of rows included in a strip so that the size of the strip does not exceed 8 KB. Now you can specify strips of larger size.

Compression of -v7.3 MAT-Files

You can store data items that are over 2 gigabytes in size in a MAT-file using the `-v7.3` option for the save function. This option was introduced in MATLAB R2006b. With MATLAB R2008a, `save` now compresses these MAT-files.

Changes to Programming Documentation

Some of the chapters in the MATLAB Programming documentation have been moved or renamed in this release. Also the title of the Programming documentation has been changed to Programming Fundamentals in order to differentiate this part of the MATLAB help from the new documentation on Developing MATLAB Classes.

Graphics and 3-D Visualization

New Figure Toolbar Buttons

Two new toolbar buttons and an information bar have been added in this release that control the new *Data Brushing* and *Data Linking* tools. Brushing and linking let you interactively explore and analyze data.

By default, now when you create axes or plot something into a blank figure, an informational banner appears across top of the figure with links to documentation for data brushing and linking capabilities. You can dismiss it by clicking the X button on right-hand side of the message bar. Once you dismiss it, subsequent figures will not display the banner unless you select **Show linking and brushing message bar** in the MATLAB Preferences Confirmation Dialogs panel.

For more information, see "Data Brushing for Graphs and Linked Variables" on page 17-30 in the Data Analysis release notes and Interactive Data Exploration in the Data Analysis documentation. Also view the video tutorial that describes these and other new features.

"v6" Plotting Option Update — Affected Functions

The Version 7.5 (R2007b) release note "The "v6" Option for Creating Plot Objects is Obsolete" on page 18-33 identified plotting functions that accept the `v6` option, which is now obsolete and will be removed in a future version of MATLAB. There is no change to the status of these functions in R2008a. However, the list of affected functions in the R2007b release note had errors and omissions. Below is the correct list of functions that support the option in their syntax and now warn when it is used:

- area
- bar
- barh
- colorbar
- contour
- contourf
- errorbar
- legend

- loglog
- mesh
- plot
- plot3
- quiver
- quiver3
- scatter
- scatter3
- semilogx
- semilogy
- stairs
- stem
- stem3
- subplot
- surf

Note that the updated list adds functions `plot3` and `quiver3`.

In the earlier release note, the following functions were incorrectly identified as accepting the `v6` option:

- meshc
- meshz
- surfc

These functions do not call `v6` code and are not affected by it becoming obsolete.

Compatibility Considerations

Specifying the `v6` flag to any plotting function now results in a warning that the option is being removed, but the option still functions. To generate a FIG-file for a plot created with the `v6` option, you still need to use the `-v6` option to the hgsave command in order to save it in a form that a previous version of MATLAB can read. Figures containing annotations (such as textboxes, arrows, ovals, and rectangles) that are saved this way open in previous versions, but the annotations do not display because different objects

are used to contain them in Version 7 than before. That is, the annotation objects have never been backward compatible.

Creating Graphical User Interfaces (GUIs)

New GUI Table Component

The new `uitable` component allows you to show data in a table. This component replaces the undocumented MATLAB `uitable` implementation. If you are using the old `uitable` component, please refer to the uitable Migration Document for help migrating to the supported `uitable` component.

Event Data Input to GUIDE Callbacks

Auto-generated callbacks of GUIDE GUIs can now access event data for Handle Graphics callbacks. The following Handle Graphics callbacks provide event data when triggered:

- `KeyPressFcn` in `uicontrol` and `figure`
- `KeyReleaseFcn` in `figure`
- `SelectionChangeFcn` in `uibuttonGroup`
- `WindowKeyPressFcn` in `figure`
- `WindowKeyReleaseFcn` in `figure`
- `WindowScrollWheelFcn` in `figure`
- `CellEditCallback` in `uitable`
- `CellSelectionCallback` in `uitable`

For example, the event data for keypress provides information on the key that is pressed. See the Callback Templates documentation for more information.

uigetfile and uiputfile Support of '.', '..', and '/'

Starting in R2007b, the `uigetfile` and `uiputfile` functions interpret `'.'`, `'..'`, and `'/'` the same way as does the `cd` command. `'.'` is interpreted as the current directory, `'..'` is the directory above the current directory, and `'/'` is the top level directory. When specifying a directory rather than a filename for either the `Filterspec` or `DefaultName` argument, you no longer need to end the string with a `'/'`. However, such strings ending with a `'/'` are interpreted as they were in previous releases.

hidegui Function Being Obsoleted

The hidegui function is being obsoleted and will be removed in a future version. Instead of this function use the `set` function to set the figure handle's `handlevisibility` property to `on` or `off`:

```
set(figurehandle, ‘handlevisibility’, 'on')
```

Changes to How uicontrols Set Figure SelectionType

SelectionType is a figure property that user interface components set when you click them. It is a read-only property that describes the gesture used when clicking the most recently selected object. For single mouse clicks, components no longer set the figure `SelectionType` property to anything but `'normal'` when they are enabled. By default, all components are enabled (their `Enable` property is `'on'`). Previously, enabled components responded to modifier keys (**Ctrl**, **Alt**, **Shift**, and **Cmd**) by setting `SelectionType` to `'alt'` or `'extend'`. Now all key modifiers set `SelectionType` to `'normal'` for UI components.

Disabled components (those with their `Enable` property set to `'off'`) set `SelectionType` to `'alt'` or `'extend'` in response to modifier keys.

When `Enable` is `'on'`, the control is active and your `Callback` responds to clicks. When `Enable` is `'off'`, the control is not active, and your `ButtonDownFcn` responds to clicks. You can provide components with a `ButtonDownFcn` to detect changes in `SelectionType`.

An additional change affects how *list box* `uicontrol` components respond to double-clicking. The second of two successive clicks sets the `SelectionType` property of an enabled list box to `'open'`. Other types of uicontrols set their `SelectionType` to `'normal'` after both the first and the second clicks unless key modifiers were used.

The following table describes the click responses for `uicontrol` components.

Enable Status	Mouse Button Pressed	Key Modifier Pressed	SelectionType on First Click	SelectionType on Second Click
on	left		`'normal'`	`'normal'` *
on	right		`'alt'`	`'open'`
on	left	**Ctrl**	`'normal'`	`'normal'` *

Enable Status	Mouse Button Pressed	Key Modifier Pressed	SelectionType on First Click	SelectionType on Second Click
on	right	**Ctrl**	'alt'	'open'
on	left	**Shift**	'normal'	'normal' *
on	right	**Shift**	'extend'	'open'
off	left		'normal'	'open'
off	right		'alt'	'open'
off	left	**Ctrl**	'alt'	'open'
off	right	**Ctrl**	'alt'	'open'
off	left	**Shift**	'extend'	'open'
off	right	**Shift**	'extend'	'open'

* Double-click result is 'open' for list boxes.

For a two-button mouse, pressing the left and right buttons together (without key modifiers) can set SelectionType to either 'normal' or 'extend'; the results are unpredictable, possibly depending on which button you actually depressed first.

uipushtool and uitoggletool toolbar buttons do not set SelectionType at all, no matter what keys or mouse buttons you press. When they are disabled, uipushtool and uitoggletool controls do not execute their ClickedCallback (or OnCallback/OffCallback), nor does using them trigger a WindowButtonDownFcn callback.

Compatibility Considerations

If your callback code depends on the value for SelectionType, be aware that for the left mouse button, pressing **Ctrl** no longer sets it to 'alt' and that **Shift**-clicking no longer sets it to 'extend'.

The SelectionType behavior for the right mouse button has not changed.

External Interfaces/API

Interface to Generic DLLs Supported on 64-bit Platforms

The ability to load a generic DLL on 64-bit platforms using loadlibrary is available in MATLAB Version 7.6 (R2008a).

Compatibility Considerations

You must install a C compiler and Perl to use this feature. For a list of supported compilers and how to install them, see Using loadlibrary on 64-Bit Platforms.

Changes to Compiler Support

The set of compilers that MATLAB supports has changed in MATLAB Version 7.6 (R2008a). For an up-to-date list of supported compilers, see the Supported and Compatible Compilers Web page.

- "New Compiler Support" on page 17-44
- "Discontinued Compiler Support" on page 17-45
- "Compiler Support to Be Phased Out" on page 17-45

New Compiler Support

MATLAB Version 7.6 (R2008a) supports new compilers for building MEX-files.

Microsoft Windows (64-bit) platform

- Microsoft Visual Studio 2008
- Windows SDK for Vista
- Intel Visual Fortran 10.1

Windows (32-bit) platform

- Microsoft Visual Studio 2008
- Open Watcom Version 1.7
- Intel Visual Fortran 10.1

Sun Solaris SPARC (64-bit) platform

- Sun Studio 12 cc / CC Version 5.9

Macintosh (Intel-based 32-bit) platforms

- Apple Xcode 3.0 (gcc / g++ Version 4.0.1)

Discontinued Compiler Support

The following compilers are no longer supported.

Windows platforms

- Intel C++ Version 7.1
- Intel Visual Fortran Version 9.0
- Borland® C++Builder® 6 Version 5.6
- Borland C++Builder 5 Version 5.5
- Borland C++ Compiler Version 5.5
- Compaq Visual Fortran Version 6.1
- Compaq Visual Fortran Version 6.6

Compatibility Considerations

To ensure continued support for building your C/C++ programs, consider upgrading to another supported compiler. For an up-to-date list of supported compilers, see the Supported and Compatible Compilers Web page.

Compiler Support to Be Phased Out

The following compilers are supported in Version 7.6 (R2008a), but will not be supported in a future version of MATLAB.

Windows (32-bit) platform

- Open Watcom Version 1.3

Solaris SPARC (64-bit) platform

- Sun Studio 11 cc / CC Version 5.8

New Version of Perl on Windows Platforms

MATLAB Version 7.6 (R2008a) includes Perl on Windows Version 5.8.8.

Compatibility Considerations

Prior to this release, MATLAB contained Perl Version 5.005. Consult your Perl documentation for details on the changes between Perl versions.

Rebuild MEX-Files Created on Linux Platforms

MATLAB V7.6 (R2008a) on Linux platforms is built with a compiler that utilizes `glibc` Version 2.3.6.

Compatibility Considerations

To work with MATLAB V7.6 (R2008a), MEX-files compiled on a Linux platform must be rebuilt.

Use mxDestroyArray to Release Memory for mxArray

The documentation for the mxSetCell, mxSetField, and mxSetFieldByNumber functions in the MATLAB C and Fortran API incorrectly instructs customers to use mxFree to release memory for any `mxArray` returned by mxGetCell, mxGetField, or mxGetFieldByNumber.

Compatibility Considerations

The correct function to use is mxDestroyArray. Calling `mxFree` on an `mxArray` only frees the array header, but does not actually free the data itself and can result in a memory leak.

To help diagnose this problem, MATLAB issues a warning if calling `mxFree` on an `mxArray` could cause memory corruption. In future versions of MATLAB, this condition may result in a segmentation violation.

Do Not Use get or set Function to Manage Properties of Java Objects

If you want to read or update a property of a Sun Java object created in MATLAB using the Java class constructor, do not use the MATLAB get or set functions on the property. For example, if you create a Java object called `javaObject` that has a property called `PropertyName`, the following commands may cause memory leaks and will be deprecated in a future version of MATLAB:

```
propertyValue = get(javaObject, 'PropertyName');
set(javaObject, 'PropertyName', newValue);
```

Compatibility Considerations

The correct commands to use are:

```
propertyValue = javaObject.getPropertyName;
javaObject.setPropertyName(newValue);
```

In future versions of MATLAB, using `get` or `set` on Java objects to manage the properties will generate an error.

New mxArray Functions for Use with MATLAB Class Objects

You can read and modify properties of MATLAB class objects using the mxGetProperty and mxSetProperty functions.

mex.bat File Removed from matlabroot\bin\$ARCH

Beginning with MATLAB Version 7.3 (R2006b), the Windows script `mex.bat` is located in the directory *`matlabroot`*`\bin`. Copies of this file were also in the directory *`matlabroot`*`\bin\`*`$ARCH`*. In MATLAB Version 7.6 (R2008a), `mex.bat` is only located in *`matlabroot`*`\bin`.

Compatibility Considerations

If you did not make the updates described in “Location of mex.bat File Changed” on page 20-37, you may need to make changes now.

Run-time Libraries Required for Applications Built with Microsoft Visual Studio 2008 Compiler

If you distribute a MEX-file, an engine application, or a MAT-file application built with the Visual Studio 2008 compiler, you must provide the Visual C++ run-time libraries. These files are required to run applications developed with Visual C++ on a computer that does not have Visual C++ 2008 installed. For information on locating the Microsoft Visual C++ 2008 Redistributable Package (x86), containing `vcredist_x86.exe` and `vcredist_x64.exe`, consult your Microsoft documentation.

Environment Variables Required with Intel Visual Fortran 9.0

When you build a MEX-file, an engine application, or a MAT application using Intel Visual Fortran 9.0, MATLAB requires that you define an environment variable for the Windows platform you are using.

Windows (32-bit) platform

Define the environment variable `VS71COMNTOOLS`. The value of this environment variable is the path to the `Common7\Tools` directory of the Microsoft Visual Studio .NET 2002 or 2003 installation directory. (Intel Visual Fortran requires Visual Studio .NET 2002 or 2003 on 32-bit Windows platforms.) The Visual Studio .NET 2003 installation program commonly defines this environment variable. For example, you might set the environment variable as follows:

```
C:\Program Files\Microsoft Visual Studio .NET 2003\Common7\Tools
```

Windows x64 platform

Define the environment variable `MSSdk`. The value of this environment variable is the path to the installation directory for Microsoft Platform SDK for Windows Server® 2003. (Intel Visual Fortran requires Microsoft Platform SDK for Windows Server 2003 on Windows x64 platforms.) The Microsoft Platform SDK installation program does not commonly define this environment variable. For example, the environment variable might have the value

```
C:\Program Files\Microsoft Platform SDK
```

-largeArrayDims Option to MEX Will Become Default

In a future version of MATLAB, the default mex command will change to use the large-array-handling API. This means the `-largeArrayDims` option will be the default. For information about migrating your MEX-files to use the large-array-handling API, see the Technical Support solution 1-5C27B9.

Compatibility Considerations

In the near future you will be required to update your code to utilize the new API. You should review your source MEX-files and `mex` build scripts.

Changes to Dynamic Data Exchange (DDE) Documentation

In MATLAB Version 5.1, all development work for the Dynamic Data Exchange (DDE) server and client was stopped. MathWorks provides, instead, a MATLAB interface to COM technology that is documented in Using COM Objects from MATLAB .

Obsolete Functionality No Longer Documented

Documentation for the following functions no longer included in External Interfaces.

Obsolete Functions
`ddeadv`
`ddeexec`
`ddeinit`
`ddepoke`
`ddereq`
`ddeterm`
`ddeunadv`

The following syntax for `enableservice` no longer included in External Interfaces.

```
enableservice('DDEServer',enable)
```

Compatibility Considerations

If you must support this obsolete functionality, we suggest you print and keep a copy of the relevant MATLAB function reference pages from V7.5 (R2007b) or earlier.

R2007b

Version: 7.5

New Features

Bug Fixes

Compatibility Considerations

Desktop Tools and Development Environment

Startup and Shutdown

Windows Platforms Startup Changes

You can now change the MATLAB startup directory on Microsoft Windows platforms using the standard shortcut **Start in** field. The `My Documents\MATLAB` subfolder (or `Documents\MATLAB` on the Microsoft Windows Vista platform) is the default startup directory. Upon startup, MATLAB automatically creates a `My Documents\MATLAB` subfolder (or `Documents\MATLAB` on the Windows Vista platform) if it does not exist, and adds it to the top of the MATLAB search path. To change the startup directory:

1. Right-click the MATLAB shortcut icon and select **Properties** from the context menu. The MATLAB Properties dialog box opens to the **Shortcut** pane.
2. In the **Start in** field, specify the directory in which you want MATLAB to start, for example, `C:\My MATLAB Place`.

   You can specify the directory via a UNC path (that is, the path can begin with `\\`).

The **Target** field specifies the full path to the file to start MATLAB, `matlab.exe`, located in the `bin` folder (for example, `C:\Program Files\MATLAB\R2007b\bin\matlab.exe`). Use the `bin\matlab.exe` to start MATLAB instead of `matlab.bat` or `matlab.exe` located in a platform directory such as `bin\win32`. The `bin\matlab.exe` detects the Windows platform and ensures required run-time files are installed.

Compatibility Considerations

The **Target** field no longer contains the `-sd $documents` startup option. In MATLAB Version 7.4 (R2007a), the startup directory was specified via the `-sd` startup option in the **Target** field. You had to specify the directory via a mapped drive. Any value in the **Start in** field was ignored.

The file to start MATLAB, as specified in the **Target** field, was `matlab.bat`.

Change any scripts you use to start MATLAB to specify the full path to `bin\matlab.exe`. If you use `matlab.bat` in R2007b, MATLAB issues a warning message instructing you to use `matlab.exe` instead.

If scripts include the `-sd` startup option to specify the startup directory, that will be the startup directory, even if a directory is specified in the **Start in** field.

Desktop

New features and changes introduced in Version 7.5 (R2007b) are:

Minimizing Tools in the Desktop Now Supported on Macintosh Platforms

You can now minimize tools in the desktop on Apple Macintosh platforms. It was introduced for other platforms in a previous version.

Double-Click to Maximize or Restore Minimized Tools in Desktop

After you minimize a tool within the desktop, you can now:

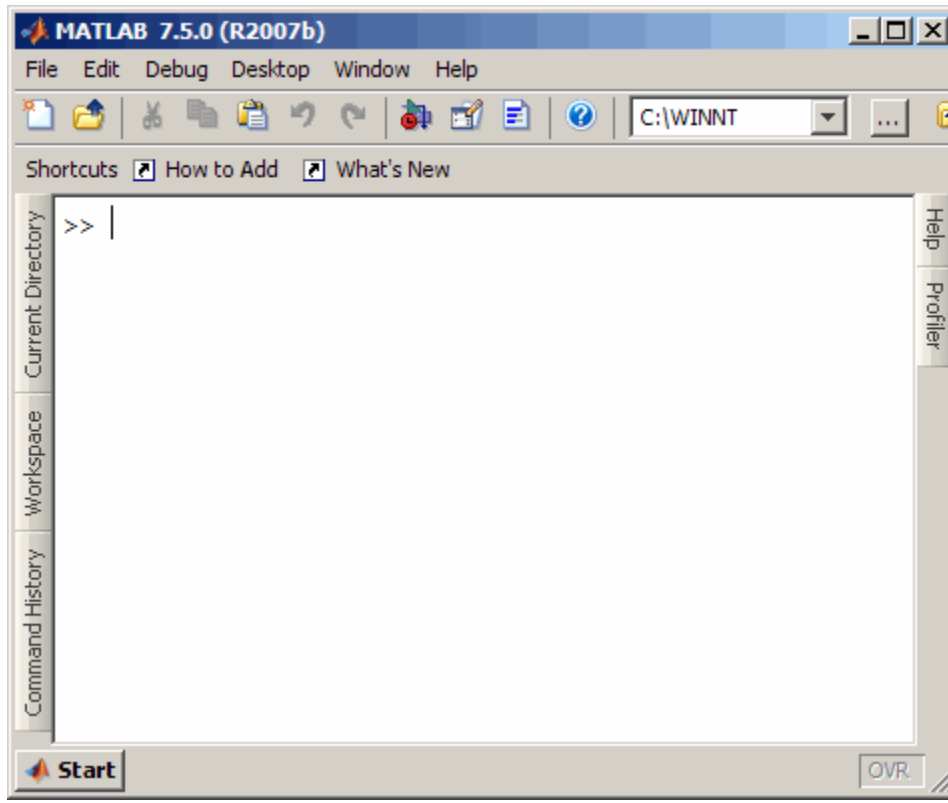
- Restore the tool to its former position by double-clicking the button.
- Drag a button to move its position—drag it to another edge of the desktop or to a new position within the edge where it's currently located.
- Restore the tool by dragging the button to a location within the desktop, or outside the desktop to undock the tool.

Similarly, you can double-click a tool's title bar to maximize the tool in the desktop; then double-click the title bar again to restore it to its former position. (The capability was introduced in R2007a, MATLAB Version 7.4).

For more information, see Open and Rearrange Desktop Tools and Documents.

New Desktop Layout — All but Command Window Minimized

Select **Desktop > Desktop Layout > All but Command Window Minimized** to arrange the desktop as shown here. The Command Window is open in the desktop, and all other desktop tools are open, but minimized.



Start Button Now Includes New Category for Links and Targets

In the Start button, there is a new category for Link and Target products. Select **Start > Links and Targets**, and then select one of the products. In previous versions, you accessed these products from the Toolboxes or Simulink software categories.

You also use this new category when running demos, or accessing **Demos** or **Contents** in the Help browser. For more information, see “Demos and Help Browser Contents Now Include New Category for Links and Targets” on page 18-6.

Start Button — View Source Files Renamed

To add your own toolboxes to the **Start** button, select **Start > Desktop Tools > View Start Button Configuration Files**. In previous versions, this menu item was **View Source Files**. There has been no change in functionality or features.

Changes to Look of Buttons in Desktop and Other Tools

Some icons on toolbar buttons have changed slightly. In addition, standard desktop icon image files are no longer provided in the *matlabroot*`/toolbox/matlab/icons` directory.

Compatibility Considerations

If your code relied on icon files in the *matlabroot*`/toolbox/matlab/icons` directory (for example, for adding your entries to the **Start** button or the Help browser), you might need to use other image files.

Antialiasing Option No Longer Necessary on Windows and Macintosh Platforms

MATLAB now follows the operating system's font settings on Microsoft and Macintosh platforms. This provides smooth fonts without the need for antialiasing within MATLAB.

Running Functions — Command Window and History

New features and changes introduced in Version 7.5 (R2007b) are:

Command History — Find Entry by Letter Now Looks in Collapsed Sessions

When you type letters in the Command History, it finds and selects the next entry that begins with the letters you typed. Now, if the entry is in a session that was collapsed, MATLAB automatically expands the session and selects the matching entry in it. In previous versions, MATLAB did not find matching entries in collapsed sessions.

If you do *not* want to find entries in collapsed sessions (the previous behavior), you can instead select **Edit > Find**, which finds text in the Command History, but not in

collapsed sessions. For more information, see Quick Search for Entries Beginning with Specified Letters or Numbers.

Pop-Up Help for a Function in the Command Window

For more information, see “Help on Selection Enhanced in Command Window and Editor” on page 18-8.

Help

New features and changes introduced in Version 7.5 (R2007b) are:

Minor Visual Changes to Help Browser

- To open the Help browser from a tool's **Help** menu, select **Product Help**. In previous versions you selected **Full Product Family Help**.
- When the Help browser first opens, it displays help for MATLAB. In previous versions, it displayed a Begin Here page. The information previously available on the Begin Here page has been incorporated into the MATLAB roadmap page.
- When you close and reopen the Help browser, it maintains the list of pages you previously viewed, but does not open to the page you last viewed. In previous versions, upon reopening, the Help browser displayed the page you last viewed.

Demos and Help Browser Contents Now Include New Category for Links and Targets

When you run the `demo` function or access **Demos** or **Contents** in the Help browser, there is a new category for Link and Target products.

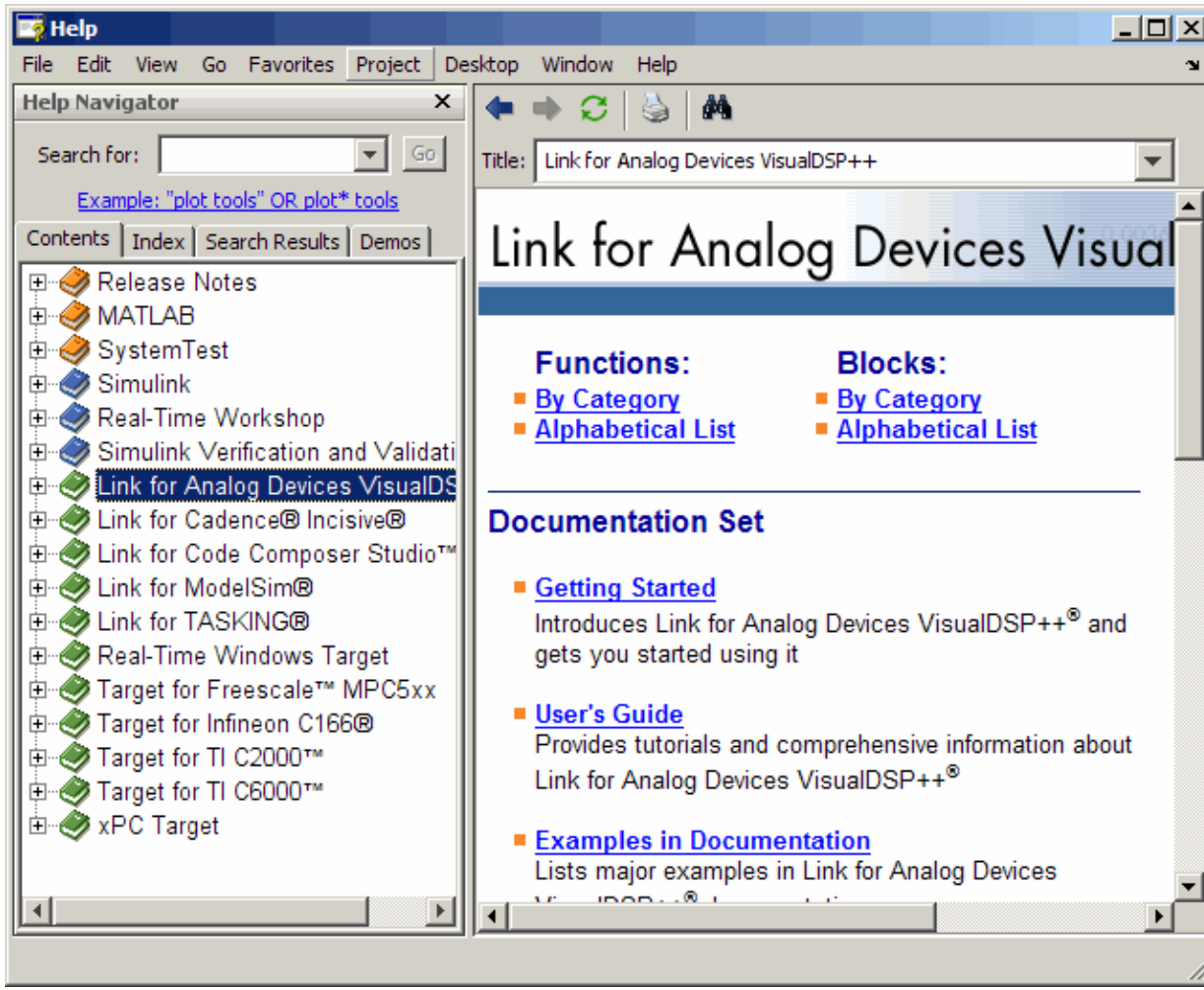
To use the `demo` function to access a demo that is now in the Links and Targets category, you specify the new subtopic 'links and targets', followed by the category. For example

```
demo('links and targets', 'link for modelsim')
```

displays the **Demos** pane, and expands the Link for ModelSim® demos listing.

In the Help **Demos**, Link and Target products appear together in their own category, and are identifiable by the new Links and Targets icon,  .

In the Help **Contents**, Link and Target products appear together after any installed Simulink and blockset products, and are identifiable by the new Links and Targets green book icon,  .



This new category is also used in the **Start** button in the MATLAB desktop—for more information, see “Start Button Now Includes New Category for Links and Targets” on page 18-4.

If you add help or demos to the Help browser for your own toolbox or list your own toolbox in the **Start** button and you want to take advantage of the new Links and Targets category, use the new type, `links_targets`, in the `info.xml` file for your toolbox.

Compatibility Considerations

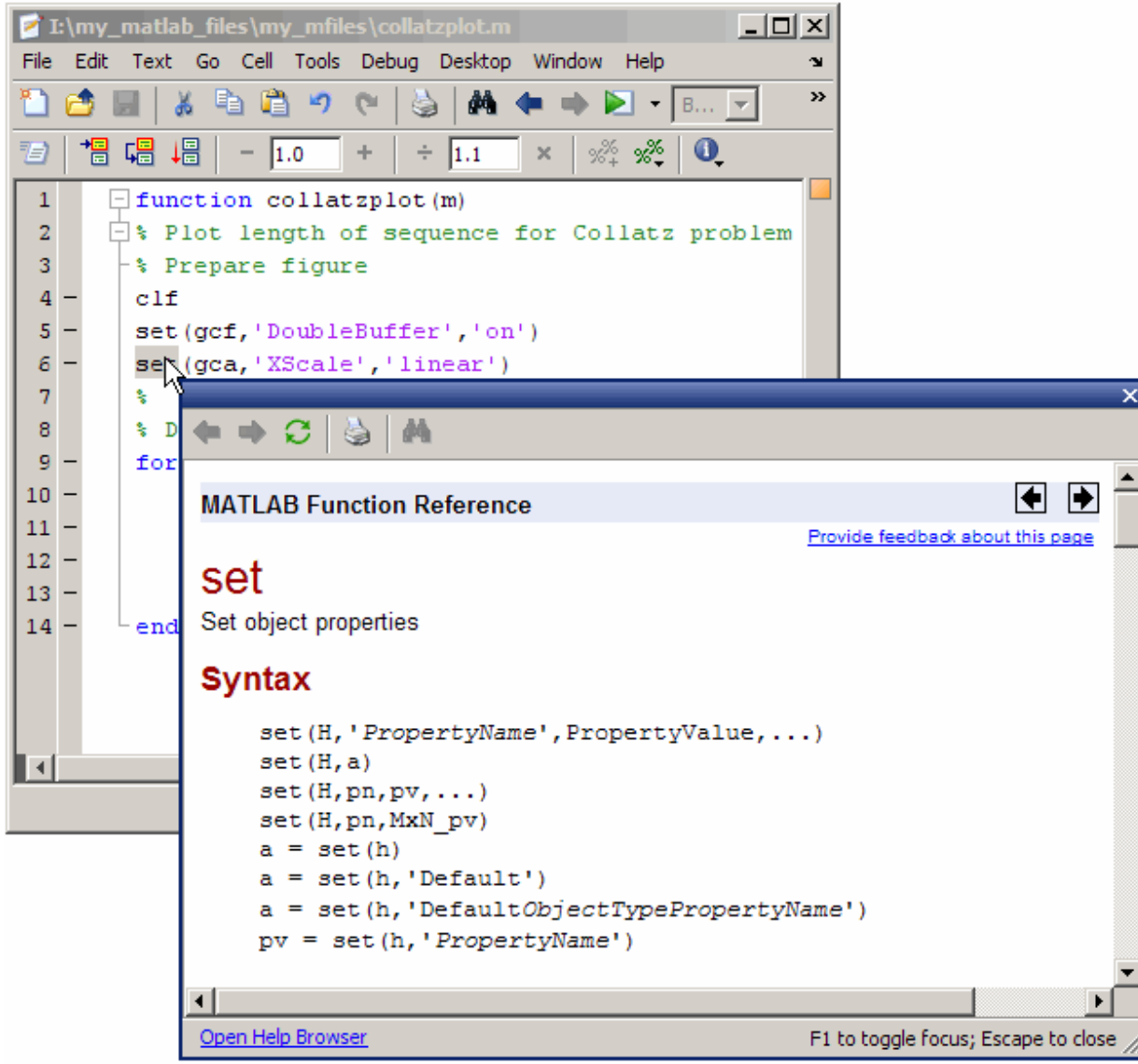
In previous versions, when you used the `demo` function to access a demo that is now in the Links and Targets category, you specified a different subtopic and category. If you have any code that relies on the `demo` function for accessing Links and Targets demos, you will need to replace the subtopic and category in the code.

In previous versions, you accessed the products in the Help browser **Demos** from the Toolbox or Simulink software categories.

In previous versions, you accessed the products in the Help browser **Contents** from within the list of toolbox products (orange book icon) or Simulink products (blue book icon).

Help on Selection Enhanced in Command Window and Editor

To get help for a function in the Command Window or the Editor, click the pointer in the function name and press **F1**. The reference page for that function appears in a small help window. To close the window, press **Escape**. You can also access the feature by choosing **Help on Selection** from the context menu. To change the window in which this help appears, select **File > Preferences > Help**, and adjust the option for **Help on Selection and More Help**.


I:\my_matlab_files\my_mfiles\collatzplot.m
File Edit Text Go Cell Tools Debug Desktop Window Help
function collatzplot(m)
% Plot length of sequence for Collatz problem
% Prepare figure
clf
set(gcf,'DoubleBuffer','on')
set(gca,'XScale','linear')
MATLAB Function Reference
Provide feedback about this page
set
Set object properties
Syntax
set(H,'PropertyName',PropertyValue,...)
set(H,a)
set(H,pn,pv,...)
set(H,pn,MxN_pv)
a = set(h)
a = set(h,'Default')
a = set(h,'DefaultObjectTypePropertyName')
pv = set(h,'PropertyName')
Open Help Browser
F1 to toggle focus; Escape to close

Compatibility Considerations

In the previous version, you could select the function name, right-click, and select **Help on Selection**. The documentation appeared in the Help browser. Now if you want to see the documentation for the function in the Help browser, first access the pop-up help, and then click the **Open Help Browser** link.

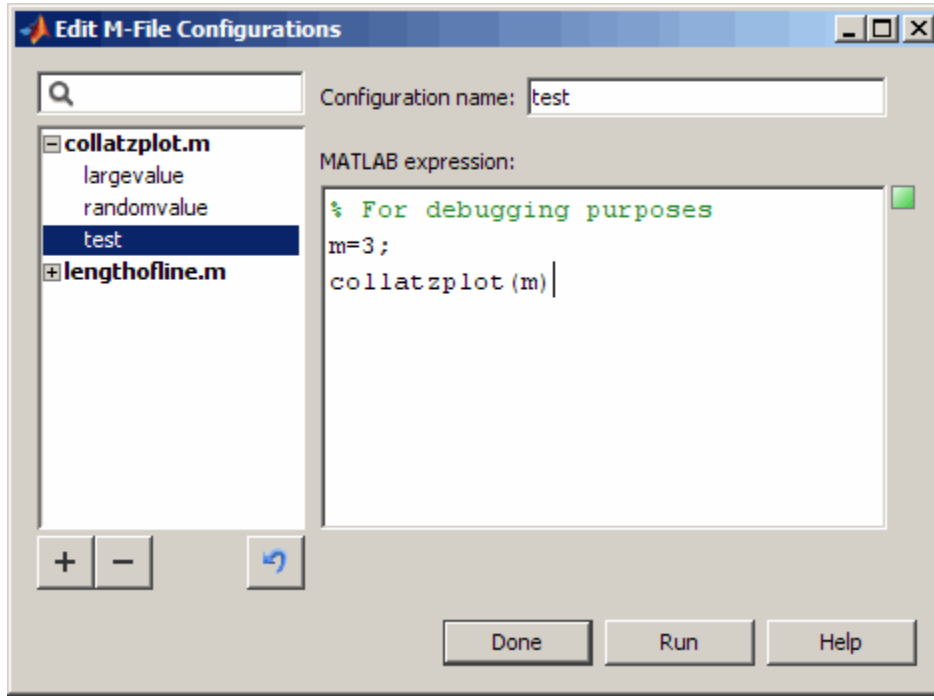
Editing and Debugging M-Files

New features and changes introduced in Version 7.5 (R2007b) are:

Run Your Function M-Files in the Editor/Debugger Using Configurations

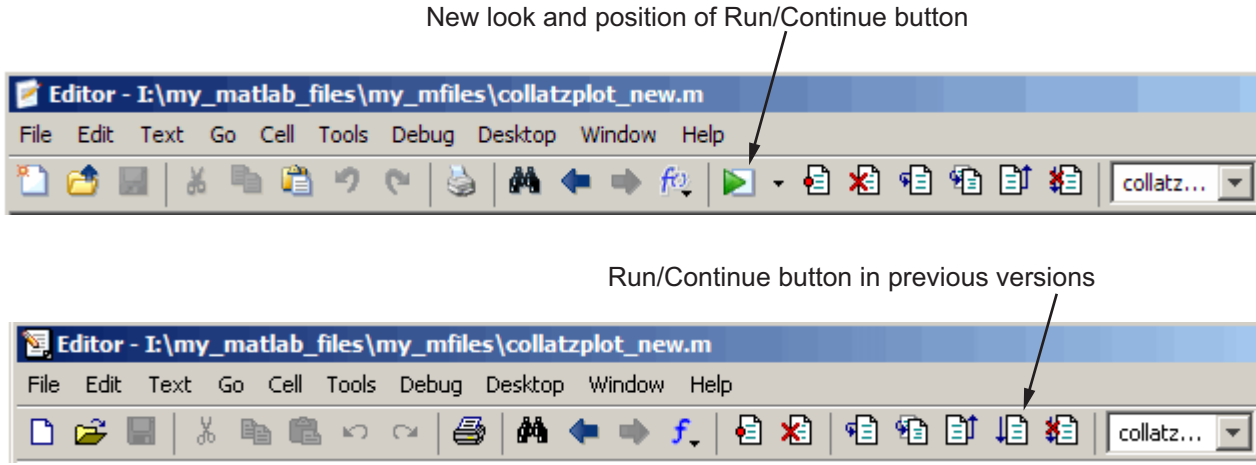
In the Editor/Debugger, you can provide values for a function's input arguments using a configuration, and then run that configuration to use the assigned values. Use a configuration as an alternative to running the function in the Command Window. You can associate multiple configurations with an M-file, each for different input values. MATLAB saves the configurations between sessions.

To create a configuration, first open an M-file in the Editor/Debugger. Then, from the down arrow on the Run button in the toolbar  select **Edit Configurations for *`filename`***. In the resulting Edit M-File Configurations dialog box, add statements and name the configuration. For more information, see Run Files with Input Arguments in the Editor.



Run/Continue Button Changes

The Run/Continue button has a new look and new location on the Editor/Debugger toolbar.



Compatibility Considerations

The button performs the same as it did in previous versions, but you need to access it in the new position.

Code Folding Feature for Collapsing and Expanding Code

To improve the readability of files containing several subfunctions, the Editor includes a code folding feature, which is enabled by default. Using this feature you can collapse and expand subfunctions and their associated help. The following figure shows the `collatzplot_new` function collapsed, such that only the function definition is displayed. The figure shows the `collatz` function expanded, revealing both the help code and the function code. If you collapse just the help code, only the H1 help line displays.

- To expand code that is collapsed, click the plus sign (+) to the left of the code you want to expand.
- To collapse code that is expanded, click the minus sign (-) to the left of the code you want to collapse.
- To expand or collapse all of the code in an M-file, place your cursor anywhere within the M-file, right-click, and then select **Code Folding > Expand All** or **Code Folding > Collapse All** from the context menu.

For more information, see Code Folding—Expanding and Collapsing M-File Constructs

Quick Help for a Function in the Editor

For more information, see “Help on Selection Enhanced in Command Window and Editor” on page 18-8.

Line Endings Removed in Files Provided with MATLAB Software for Windows Platforms; Impacts Viewing in Notepad Application

In previous versions, text files provided with MATLAB for Windows platforms included a carriage return and line feed at the end of each line. Starting in R2007b, the text files MATLAB provides do not include a carriage return and line feed at the end of each line.

File types affected are:

- `.asc`
- `.bat`
- `.c`
- `.cc`
- `.cdr`
- `.cpp`
- `.def`
- `.for`
- `gs.rights`
- `.h`
- `.ini`
- `.m`
- `.mdl`
- `.pl`
- `readme`
- `.tlc`
- `.tmf`
- `.txt`

There is no impact if you view the files in MATLAB and other common text editors, with the known exception of the Microsoft Notepad application.

Compatibility Considerations

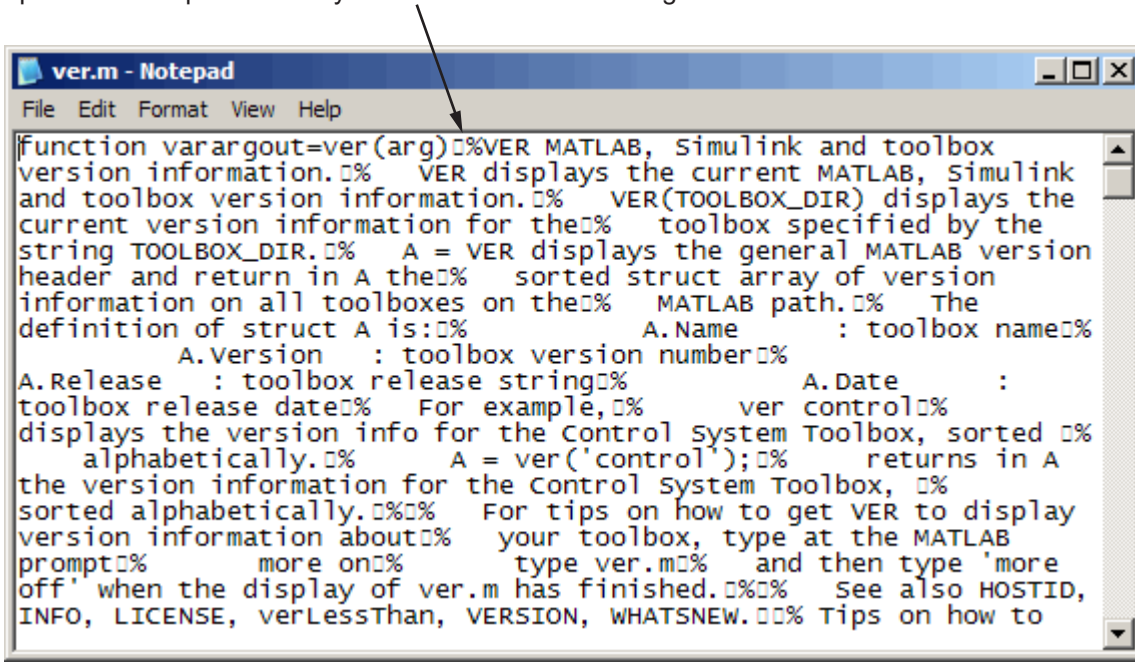
If you use the Notepad application to view files provided with MATLAB, you see carriage return and line feed symbols □% instead of line endings. This makes the files less readable in the Notepad application. Other text editors might display the symbols instead of line endings, but of the common text editors tested, none have been found that do so.

As an alternative to the Notepad application, use the Microsoft WordPad application, provided with Windows platforms, or another text editor to view the files.

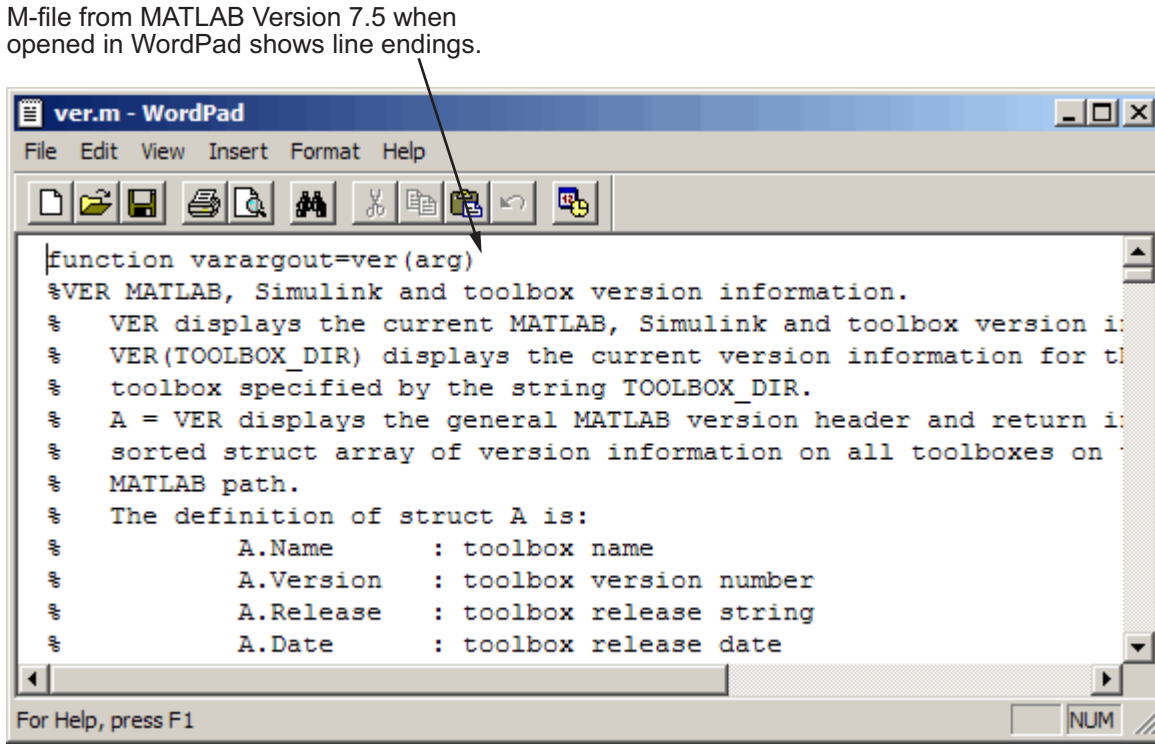
If your Windows file associations are set to associate any of the listed file types with Notepad, change the associations to use WordPad or another text editor.

The following illustration shows how the `ver` M-file included with MATLAB Version 7.5 looks when opened in the Notepad application.

M-file from MATLAB Version 7.5 when opened in Notepad shows symbols instead of line endings.



The following illustration shows how the `ver` M-file included with MATLAB Version 7.5 looks when opened in the WordPad application.



There are no problems with files you create or edit in the Notepad application, and then view or edit in MATLAB. The files have line endings in the MATLAB Editor, and continue to have line endings when you open them in the Notepad application.

(Microsoft product screen shot(s) reprinted with permission from Microsoft Corporation.)

Stand-Alone Editor Will Not Be Included in Next Version

The MATLAB stand-alone Editor (`meditor.exe`) will no longer be provided, starting in the next version of MATLAB. Instead of the stand-alone Editor, you can use the MATLAB Editor/Debugger. It provides all the features of the stand-alone Editor, plus the following:

- Tab completion
- Debugging M-files
- Evaluating selections
- Accessing source control features
- Docking the tool in the MATLAB desktop
- Using cell features for rapid code iteration or publishing

Compatibility Considerations

Some users have preferred the stand-alone Editor to the MATLAB Editor/Debugger because of slightly better startup performance and because it does not require a MATLAB license. For those situations, you can use any text editor you have, such as the UltraEdit® application from IDM Computer Solutions, or the GNU Emacs software.

Determine the McCabe (Cyclomatic) Complexity of an M-File

The `cyc` option to the mlint function enables you to determine the McCabe complexity (also referred to as the cyclomatic complexity) of an M-file. Higher McCabe complexity values indicate higher complexity, and there is some evidence to suggest that programs with higher complexity values are more likely to contain errors. Frequently, you can lower the complexity of a function by dividing it into smaller, simpler functions. In general, smaller complexity values indicate programs that are easier to understand and modify. Some people advocate splitting up programs that have a complexity rating over 10. See mlint for syntax and an example.

Publishing Results

New features and changes introduced in Version 7.5 (R2007b) are:

- "Notebook and Word for Office 2007" on page 18-16
- "Text Markup in Cells for Publishing" on page 18-17
- "Preference to Restrict Lines of Output" on page 18-17

Notebook and Word for Office 2007

Notebook now supports Microsoft Word for Office 2007. For details, see Creating a MATLAB Notebook to Publish to Microsoft Word.

Text Markup in Cells for Publishing

The following Editor/Debugger menu items are added to assist you in marking up cells in the Editor/Debugger for publishing. Access the menu items presented in the following list from **Cell > Insert Text Markup** . When you select the menu item, the Editor inserts code to assist you in adding the text markup for the specified item.

- **Document Title and Introduction**
- **Section Title with Cell Break**
- **Hyperlinked Text**
- **Image**
- **Numbered List**
- **HTML Markup**
- **LaTeX Markup**

The first two list items are provided instead of the **Cell Title** and **Descriptive Text** menu items that were offered in Version 7.4 (R2007a).

As an alternative to using the Cell menu items, you can manually insert code to mark up cells in your M-file for publishing. For details on the Cell menu items and the resulting code see Marking Up Text in Cells for Publishing

Preference to Restrict Lines of Output

You can now specify options to restrict the number of lines included in the output of a published M-file. To access this option from the Editor/Debugger, follow these steps:

1. Select **File > Preferences > Editor/Debugger > Publishing**
2. In the Editor/Debugger Publishing Preferences pane, select the **Evaluate code** and **Restrict output to** options.
3. Specify the maximum number of lines that you want to include in the output.

Mathematics

New Functions

Function	Description
quadgk	Numerically evaluates the integral, adaptive Gauss-Kronrod quadrature
bvp5c	Solves boundary value problems for ordinary differential equations, notably useful for small error tolerances
maxNumCompThreads	Gets and sets the maximum number of computational threads

finite Function Deprecated

In this release, the `finite` function displays a warning message that the function is now deprecated. Support for `finite` will be removed in a future release of MATLAB software.

Compatibility Considerations

It is recommended that you replace all calls to `finite` with the isfinite function.

dmperm Function Gives Coarse Decomposition

The dmperm function now provides two additional output arguments for the indices of the Dulmage-Mendelsohn coarse decomposition.

ldl Function Supports Real Sparse Symmetric Matrices

The ldl function now provides factorization and solving for an additional output argument, the scaling matrix, when the input matrix is real sparse and symmetric.

Upgrade to LAPACK Library

MATLAB software now uses Version 3.1 of the Linear Algebra Package (LAPACK) library.

Upgrade to BLAS Libraries

For AMD processors, MATLAB software now uses Version 3.6 of the AMD Core Math Library (ACML™) for the Basic Linear Algebra Subroutine (BLAS) libraries.

Library for LAPACK and BLAS Symbols Separated

The binder library, `libmwlapack.lib`, containing both LAPACK and BLAS symbols is now two separate library files: `libmwlapack.lib` for LAPACK symbols and `libmwblas.lib` for BLAS symbols.

Compatibility Considerations

If you previously linked to the `libmwlapack.lib` library to use the BLAS symbols, you will need to update your code to link to the `libmwblas.lib` library.

Colon Operations on Characters Return Character Type Data

Using a colon with characters to iterate a for-loop now returns data of type character. For example,

```
for x='a':'b',x,end
```

results in

```
x =
    a
x =
    b
```

Compatibility Considerations

Previously, colon operations with characters iterating a for-loop returned data of type double. In previous releases the above example returned:

```
for x='a':'b',x,end

x =
    97
x =
```

```
    98
```

Existing program code that relies on the colon operations of character arrays returning a double, needs to be updated to expect a character data type.

Matrix Generating Functions No Longer Accept Complex Inputs

Calling matrix generating functions, such as `ones`, `zeros`, `rand`, `randn`, `true`, and `false`, with a complex number as dimensions input now returns the error:

```
true([1 i])
??? Error using ==> true
Size vector must be a row vector with real elements.
```

Compatibility Considerations

In previous releases, if you supplied a complex number as a dimension input, MATLAB software returned:

```
true([1 i])
Warning: Size vector should be a row vector with integer elements.
         Complex inputs will cause an error in a future release.

ans =

   Empty matrix: 1-by-0
```

Existing program code that relies on entering complex numbers as dimension input to a matrix generating function should be modified.

Data Analysis

Programming

Increased Size for Large Arrays

On 64-bit platforms, MATLAB arrays are no longer limited to 2^{31} elements. The limit in MATLAB 7.5 is 2^{48}-1. For example, given sufficient memory, many numeric and low-level file I/O functions now support real double arrays greater than 16 GB.

Documentation for Multiprocessing in MATLAB

Documentation for "Multiprocessing in MATLAB" has moved from Desktop Tools and Development Environment to the "Improving Performance and Memory Usage" section of MATLAB Programming.

Setting Number of Threads Programmatically

In this release, MATLAB provides a way to set or retrieve the maximum number of computational threads from within an M-file program. With the maxNumCompThreads function, you can either set the maximum number of computational threads to a specific number, or indicate that you want the setting to be done automatically by MATLAB

New Internal Format for P-code

P-code files have a new internal format in MATLAB Version 7.5. The new P-code files are smaller and more secure than those built with MATLAB 7.4 and earlier, and provide a more robust solution to protect your intellectual property.

Any P-code files that were built using MATLAB 7.4 or earlier also work in 7.5. However, support for these older files will at some point be removed from MATLAB.

P-code files built with MATLAB 7.5 only work on 7.5 or later. They cannot be used with MATLAB 7.4 or earlier versions.

Compatibility Considerations

Rebuild any P-code files using MATLAB 7.5 that you expect to need in the future.

New Split String Functionality in regexp

Using the regular expressions function regexp, you can now split an input string into sections by specifying the new `'split'` option when calling `regexp`:

```
s1 = ['Use REGEXP to split ^this string into ' ...
      'several ^individual pieces'];

s2 = regexp(s1, '\^', 'split');

s2(:)
ans =
    'Use REGEXP to split '
    'this string into several '
    'individual pieces'
```

The `split` option returns those parts of the input string that are delimited by those substrings returned when using the `regexp 'match'` option.

Changes Related to Error Handling

New Error Handling Mechanism

MATLAB extends its error-handling capabilities with the new `MException` class to provide you with a more secure and extensible system for throwing and responding to errors. Read about this new feature in the Error Handling section of the MATLAB Programming documentation and in related function reference pages such as MException.

This feature *extends* the error-handling capabilities available in earlier releases, but does not replace that functionality. Your M-file programs will continue to function the same when using Version 7.5.

New Syntax for catch Function

As part of new error-handling mechanism, the catch function has a new, optional syntax as shown here in a try-catch statement:

```
try
   % Try to execute this block of code. Go to 'catch' on error
   - code that may error -
catch ME
```

```
   % Deal with the error captured in MException object, ME
   - code to handle the error -
end
```

`ME` is an object of the `MException` class. This command gives you access to the `MException` object that represents the error (i.e., exception) being caught.

Warning and Error Messages Now Wrap

In previous versions of MATLAB, warning and error messages that were longer than the width of your terminal screen extended beyond the visible portion of your screen. These messages now wrap onto succeeding lines so that the entire text of the warning or error is visible.

Change to Error Message from Anonymous Function

The error message and M-file line number that MATLAB displays when encountering an error in an anonymous function defined within a script file or at the MATLAB Command Line has changed in this release. In MATLAB Version 7.4, the error message included the line number (set to `1` for anonymous functions), and the stack information returned by the lasterror function also showed the line number as `1`. In MATLAB 7.5, the line number is not displayed in the error message and is set to `0` in the returned stack information.

For example, when you enter the following two lines at the command line, MATLAB generates an error from the anonymous function:

```
X = @() error('* Error *');
X()
```

This example shows the difference between MATLAB Versions 7.4 and 7.5:

```
e = lasterror;

e.message
ans =
Error using ==> @()error('Error') at 1    % V7.4 response
Error using ==> @()error('Error')         % V7.5 response
* Error *

e.stack
ans =
    file: ''
```

```
    name: '@()error('* Error *')'
    line: 1                                 % V7.4 response
    line: 0                                 % V7.5 response
```

Compatibility Considerations

If you have programs that rely on the line number returned in response to an error in an anonymous function, these programs may not work as expected. Remove dependencies on the returned error message and line number, or update your program code to use the new string and value.

New Message In Response to Ctrl+C

MATLAB now displays a more user-friendly message when you press **Ctrl+C**. The previous response to **Ctrl+C** was

```
Error in ==> testctrlc>waitawhile at 5
pause(100);

Error in ==> testctrlc at 2
waitawhile
```

In this and future releases, pressing **Ctrl+C** still halts program execution, but now displays the response

```
??? Operation terminated by user during ==> testctrlc>
   waitawhile at 5

In ==> testctrlc at 2
waitawhile
```

Compatibility Considerations

You only need to be aware that the change in the text of this message is intentional and does not signify any error on your part.

hdfread Errors Instead of Warns on I/O Failures

In previous releases, hdfread issued a warning when a requested I/O operation failed. In addition, `hdfread` created an empty variable in the workspace. In this release, `hdfread` now errors when a requested I/O operation fails and does not create an empty variable in the workspace.

Compatibility Considerations

If you call `hdfread` in a script to perform an I/O operation and that operation fails, your script will now terminate. Previously, because `hdfread` only warned when an I/O operation failed, your script would continue processing.

Results From tempname Are More Unique

The tempname function now produces a string such as

```
C:\Temp\tpe51f2ba3_9ad3_490f_8142_58359c98f4a5
```

when Java is present, or a string like

```
C:\Temp\tp346976948758473
```

when `nojvm` is selected. Underscores are included in the name so you can use the filename portion of it as a valid M-function name. If a string row vector is passed in as an argument, that string is used instead of tempdir as the root.

Compatibility Considerations

Because the new string generated by tempname is generally longer than the string constructed in earlier versions, there is a possibility of exceeding length restrictions, especially if your program code passes a string to the `tempname` function. If you consider this to be a potential problem, verify that the strings you pass to `tempname` will not result in an overly long string being returned.

MATLAB Includes New Input Argument Validation Functions

MATLAB now includes two new functions that validate the input arguments passed to functions. For example, you can use these functions to make sure that an input argument is numeric and nonempty. The following table lists these functions with a brief description.

Function	Description
validateattributes	Check validity of array
validatestring	Check validity of text string

Windows Current Working Directory Corrected

On Windows, you can define a current working directory, `cwd`, for each drive letter. For example, entering the command `cd D:\work` at the DOS prompt defines your `D` current working directory as `D:\work`. All references to `D:` are then relative to this directory.

The term . . .	Represents the directory . . .
`D:`	`D:\work`
`D:matlab`	`D:\work\matlab`
`D:\matlab`	`D:\matlab`

(Note the difference between `D:\` and `D:`, where the former is the drive `D`, and the latter is a user-defined working directory that may or may not be equal to `D:\`.)

Previous versions of MATLAB have been inconsistent in the way that volume-relative path specification is handled. For example, in MATLAB 7.4 and earlier, if `D:` were defined as the directory `D:\work`, the following commands on the left and right returned identical results:

```
dir D:                              dir D:\
dir D:matlab                        dir D:\work\matlab
fileparts('D:myfile.m')             fileparts('D:\myfile.m')
```

This has been fixed in MATLAB 7.5 so that the following are now equivalent:

```
dir D:                              dir D:\work
dir D:matlab                        dir D:\work\matlab
fileparts('D:myfile.m')             fileparts('D:\work\myfile.m')
```

Compatibility Considerations

Some MATLAB commands may fail or return unexpected results if you use Windows current working directories on MATLAB. You might have to update hard-coded paths if you have been relying on the incorrect behavior exhibited in earlier versions.

New Multimedia Functionality

A new `mmreader` video file reader object for Windows platforms supports formats such as AVI, MPEG, and WMV, and adds the ability to read additional video codecs that aviread does not support. For more information, see the `mmreader` and read reference pages.

Compressed AVI Video Files in Windows Vista and Windows XP x64

Because Windows Vista, Windows Vista 64-bit, and Windows XP x64 operating systems do not ship with the Indeo® 5 codec, which is the default codec used by MATLAB Audio-Video functions for file compression, the functions movie2avi and avifile now generate uncompressed AVI files on these platforms. Also, aviread on these platforms cannot read files that were compressed using the Indeo 5 codec.

Compatibility Considerations

If you have upgraded your Windows operating system to Windows Vista, Windows Vista 64-bit, or Windows XP x64, you need to install the Indeo 5 codec to read or create Indeo 5 compressed AVI files with aviread, avifile, or movie2avi. You can learn more about downloading the Indeo 5 codec from the Ligos Corporation Web site at `http://ligos.com/index.php/home/products/indeo/`

mmfileinfo Reads Files on MATLAB Path

The mmfileinfo function now reads files on the MATLAB path, not only those in the current directory. The `mmfileinfo` output struct contains a field called `Path`, which indicates the directory where the file exists.

Compatibility Considerations

The `Filename` field of the output struct from `mmfileinfo` now contains only the filename itself without any path information, while the path information is contained in the `Path` field. In previous releases, the `Filename` field contained both path and filename information.

Changes to imread Support of TIFF Format

The imread function includes several updates to its TIFF support:

- `imread` reads TIFF files that use JPEG, LZW, and Deflate compression.
- `imread` reads image data from TIFF files in any arbitrary samples-per-pixel and bits-per-sample combination.
- `imread` provides increased performance when reading large images, when used with the 'PixelRegion' parameter.

Removal of freeserial Function

The `freeserial` function is now obsolete. Use `fclose` to release the serial port.

Compatibility Considerations

If your program code still makes use of the `freeserial` function, replace each instance with the `fclose` function instead.

Graphics and 3-D Visualization

Datatips Are Now Saved to FIG-Files

When you save a figure, all datatips existing in it are saved along with other annotations. When you open the FIG-file, the datatips are displayed and can be manipulated or deleted in the same ways they could in the original figure.

Compatibility Considerations

If you open a FIG-file containing datatips while using a previous MATLAB version (V7.4 or earlier), no error results, but the datatips do not display.

New Options for Displaying Groups of Lines in Legends

You can now customize how legends for figures display groups of lines, such as contours. Previously, legends displayed groups of lines such as contourgroups with a glyph that represented the entire group; now users have the flexibility to designate a single legend entry, a legend entry for each child of the group, or no legend entries for the group.

By default, a legend entry for an hggroup now consists of the `DisplayName` of its first child and a glyph representing it (previously, no glyph appeared, only the `DisplayName`). This is what you now see after clicking the legend tool icon in the figure's toolbar. However, you can set the new `Annotation` property of hggroups to control how the group is represented in a legend. For details and examples of its use in customizing legends, see Controlling Legends in the Graphics documentation.

Drawnow Update Option Now Updates Uicontrols Only

The `drawnow` command can now selectively update the display of UI components. The `update` option enables you to update only uicontrol objects without allowing callbacks to execute or processing other events in the queue.

Annotation Textboxes Can Automatically Resize to Fit their Contents

In previous releases, textboxes had fixed sizes that users needed to adjust to fit the size of their contents. Now, if you create a textbox annotation using a GUI or the

`annotation` function, it can grow or shrink to just fit the text you type into it. This behavior is controlled by the Annotation Textbox object's `FitBoxToText` property, which can be `'on'` or `'off'`. When you create a textbox with the Annotation toolbar (using the  tool), this property is set to `'on'` if you create a textbox without dragging; however, if you drag to make the new textbox have a certain size, the property is initially `'off'`. When you create a textbox with the annotation function, for example,

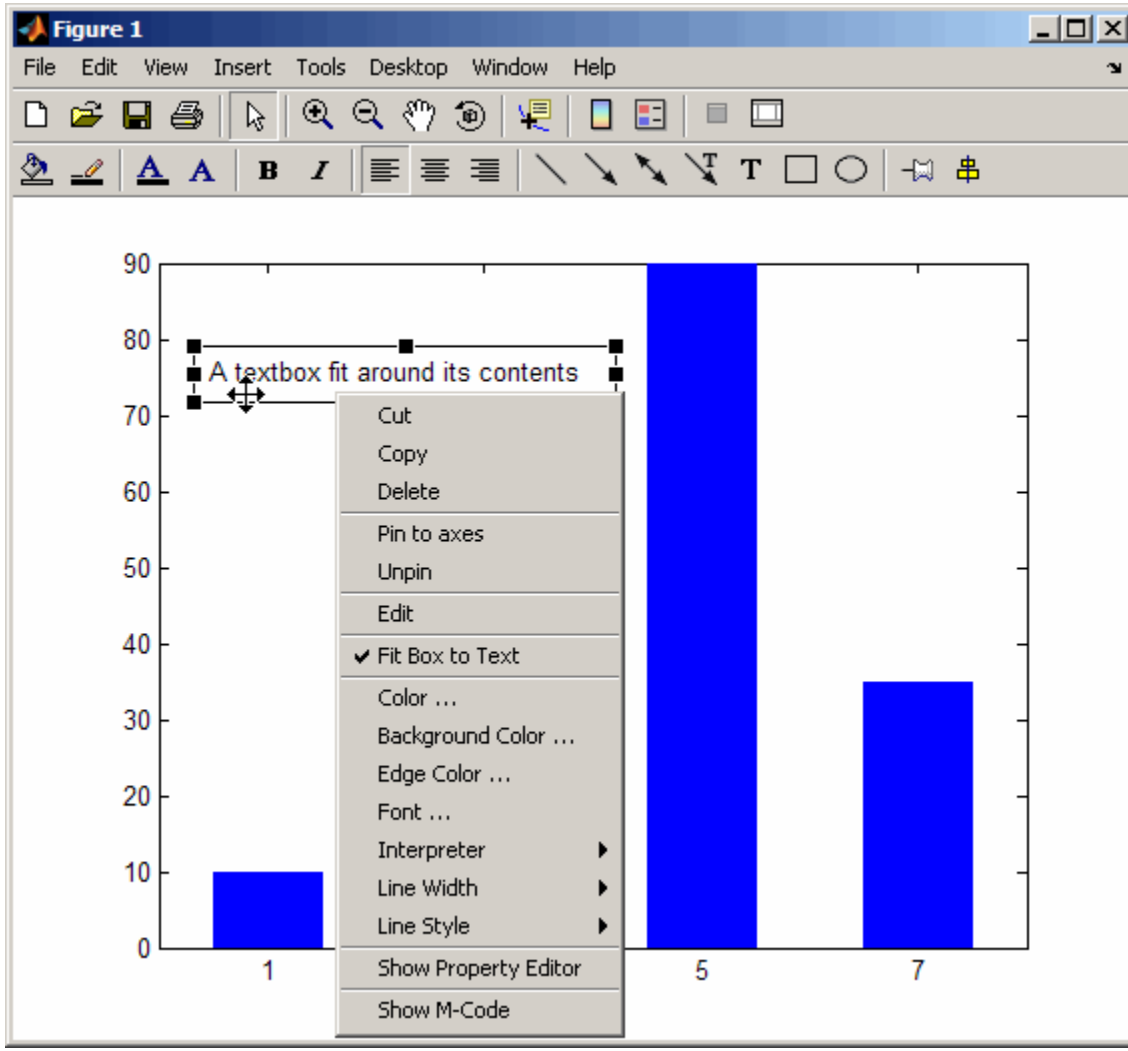
```
htb = annotation('textbox')
```

without specifying a position and size, the textbox is created with `FitBoxToText` set to `'on'`. If you specify a position vector in the command, for example,

```
htb = annotation('textbox', [.1 .8 .4 .1])
```

the textbox is created with `FitBoxToText` set to `'off'`.

Similarly, if you resize a textbox in plot edit mode or change the width or height of its `position` property directly, its `FitBoxToText` property is set to `'off'`. You can toggle this property with `set`, with the Property Inspector, or more conveniently, via the object's context menu, as the illustration below shows.



If you edit a textbox that has `FitBoxToText` set to `'on'`, the textbox resizes to accommodate the number of characters and lines in the text as you type. You can reposition the textbox without changing the `FitBoxToText` property, but as soon as you resize it, the property becomes (or remains) `'off'`.

Property Inspector Now Has Context-Sensitive Help

When you use the Property Inspector (the inspect command), you can now ask for a description of any property it shows. The descriptions come from the property reference page for the type of object being inspected (e.g., axes, lineseries, annotations, uicontrols, etc.). To get a description of a property, right-click its name in the left-hand column of the Property Inspector and select **What's This?** from the context menu that appears. A mini-help window opens to show the property's description from the object's property reference page. If you need an overview of all properties pertaining to particular kinds of objects and how these objects relate, scroll through the entries in the mini-help window or open the Handle Graphics Property Browser from the Help Browser.

The "v6" Option for Creating Plot Objects is Obsolete

Prior to MATLAB Version 7, handles returned by high-level plotting functions referenced graphic primitives, such as lines and patches. In Version 7, MATLAB began bundling the primitives into "series" objects, for example, lineseries, barseries, and contourgroups. At that time, the `v6` option was added to plotting functions to enable users of MATLAB Version 7.x to create FIG-files that previous versions can open. The `v6` option is now obsolete. It will be removed in a future MATLAB release. The following functions include the option in their syntax and now warn when it is used:

- area
- bar
- barh
- colorbar
- contour
- contourf
- errorbar
- legend
- loglog
- mesh
- meshc
- meshz
- plot
- quiver

- scatter
- scatter3
- semilogx
- semilogy
- stairs
- stem
- stem3
- subplot
- surf
- surfc

Compatibility Considerations

Specifying the `v6` flag to any plotting function now results in a warning that the option is being removed, but the option still functions. To generate a FIG-file for a plot created with the `v6` option, you still need to use the `-v6` option to the hgsave command in order to save it in a form that a previous version of MATLAB can read. Figures containing annotations (such as textboxes, arrows, ovals, and rectangles) that are saved this way open in previous versions, but the annotations do not display because different objects are used to contain them in Version 7 than before. That is, the annotation objects have never been backward compatible.

Creating Graphical User Interfaces (GUIs)

New Editors for Creating Custom Toolbars within GUIDE

In previous releases, adding toolbars to a GUI had to be done programmatically, by writing code for the `uitoolbar`, `uipushtool` and `uitoggletool` functions. GUIDE now has a Toolbar Editor and an associated Icon Editor that allow you to lay out a toolbar for a GUI and populate it with standard predefined tools for printing, saving, panning, zooming, annotating, etc. or with custom tools that you design yourself. You can draw or modify an icon for any tool on your toolbar with the Icon Editor or import one from an image file.

The Toolbar Editor

When you run GUIDE to create a new GUI or edit an existing one, click the Toolbar Editor icon in the Layout Editor Toolbar or choose **Toolbar Editor** from the **Tools** menu to open the Toolbar Editor. To add tools to the toolbar of your GUI, you drag standard or custom tool icons to the toolbar layout area at the top. You can modify a tool's properties using the Toolbar Editor and Property Inspector. You can also create and customize icons for tools using the new Icon Editor, described next. You control the behavior of a tool in your toolbar with its `ClickedCallback` (and for toggle tools, their `OnCallback` and `OffCallback`) in the GUI's associated M-file. If you use an existing tool, such as Print or Save, you do not need to modify its callback; it is predefined as `%default`.

The Icon Editor

The Icon Editor lets you customize the icons for existing tools or create entirely new icons. Icons are stored as `CData` for each tool, and can also be read from and saved to `.icn` files. The Icon Editor also includes a Color Editor, a palette you can use for picking predefined colors and defining new ones.

Coordinate Readouts in Layout Editor

The GUIDE Layout Editor now has two fields in its lower left corner that continuously provide numeric feedback for:

- **Current Point** — The cursor location within the layout window
- **Position** — The Position property of the currently selected object(s) in the layout window

Both measurements are given in pixels, regardless of the current `Units` settings for GUI objects. When multiple objects are selected, any elements of their `Position` properties that are not identical are read out as `MULTI`. The readouts help you to precisely size and align GUI components.

Documentation for Making GUIDE GUIs Interact

A new documentation section, Making Multiple GUIs Work Together, has been added to illustrate how multiple GUIDE GUIs can work together by sharing data to create a more complicated GUI. It first summarizes the techniques that GUIDE GUIs can use to share data with one another. It then steps through two examples, accompanied by their M-files and FIG-files, to show how to use those techniques for specific tasks.

Functions Being Removed

Function Name	What Happens When You Run the Function?	Use This Function Instead	Compatibility Considerations
`axlimdlg`	Errors	None	No replacement
`cbedit`	Errors	guide	Use the guide command instead
`clruprop`	Errors	rmappdata	Replace existing instances of `clruprop` with rmappdata
`ctlpanel`	Errors	guide	Use the guide command instead
`edtext`	Errors	Set the `editing` property on the text object	No replacement
`extent`	Errors	Get the `extent` property of the text object	No replacement
`getuprop`	Errors	getappdata	Replace existing instances of `getuprop` with getappdata

Function Name	What Happens When You Run the Function?	Use This Function Instead	Compatibility Considerations
`hthelp`	Errors	web	Replace existing instances of `hthelp` with web
`layout`	Errors	None	No replacement
`matq2ws`	Errors	None	No replacement
`matqdlg`	Errors	None	No replacement
`matqparse`	Errors	None	No replacement
`matqueue`	Errors	None	No replacement
`menubar`	Errors	Set the `menubar` property of the figure to `none`	No differences
`menuedit`	Errors	guide	Use the guide command instead
`pagedlg`	Errors	pagesetupdlg	Replace existing instances of `pagedlg` with pagesetupdlg
`setuprop`	Errors	setappdata	Replace existing instances of `setuprop` with setappdata
`umtoggle`	Errors	Set the `Checked` property of uimenu objects	No differences
`wizard`	Errors	guide	Use the guide command instead
`ws2matq`	Errors	None	No replacement

External Interfaces/API

Support for 64-bit mxArrays

MATLAB Version 7.5 (R2007b) supports 64-bit mxArrays. This change allows C/C++ and Fortran files built on 64-bit platforms to handle large data arrays.

In earlier versions of MATLAB, `mxArray`s are limited to 2^{31}-1 elements. In Version 7.5 (R2007b) your mxArray can have up to 2^{48}-1 elements.

The mex command option, `-largeArrayDims`, uses the large-array-handling mxArray API. Use mwSize to represents size values, such as array dimensions and number of elements. Use mwIndex to represent index values, such as indices into arrays.

Compatibility Considerations

MEX-files that built properly in previous versions of MATLAB continue to build in Version 7.5 (R2007b).

The default option for `mex` is `-compatibleArrayDims`. If you use this option, `mex` builds the files using the 32-bit array-handling API.

To work with 64-bit mxArrays, your C, C++ and Fortran source code must comply with the 64-bit array-handling API. To use the API to create C/C++ MEX-files, see Handling Large mxArrays . To use the API to create Fortran MEX-files, see Handling Large mxArrays .

Fortran MEX-Files Will Require mwSize and mwIndex

In a future version of MATLAB, the default mex command option will change to `-largeArrayDims`. Fortran MEX-files will be required to use the mwSize and mwIndex preprocessor macros

Compatibility Considerations

To make your Fortran MEX-file compatible with the `-largeArrayDims` option, and to create platform-independent code, you need to include the `fintrf.h` header file in your Fortran source files, and you need to name your source files with an uppercase `.F` file extension. For information on creating MEX-files, see the Gateway Routine in the Fortran MEX-Files topic.

Changes to the MATLAB Locale Setting

Retrieving and using the proper locale setting is a mandatory operation in creating and using applications for international audiences. In R2007b, MATLAB Version 7.5 standardizes the way it initializes the locale setting across platforms. As a result, on Microsoft Windows platforms, MEX-files that use C/C++ locale-dependent standard library functions should note that MATLAB now sets the locale using the `setlocale` function.

Because of changes to Microsoft Visual Studio, MEX-Files created on Windows with dMicrosoft Visual Studio 2003 will not have the same locale setting as MATLAB Version 7.5 (R2007b).

Note: C/C++ users must not change the locale setting using the `setlocale` function. This is true for all versions of MATLAB on all platforms.

Changes to MEX Error-Handling Functions mexErrMsgTxt and mexErrMsgIdAndTxt

In MATLAB Version 7.5 (R2007b). the mexErrMsgTxt and mexErrMsgIdAndTxt functions determine where the error occurred, and display the function name. In previous versions, these functions only display the error message.

For example, if an error occurs in the function `foo`, `mexErrMsgTxt` and `mexErrMsgIdAndTxt` display the following information before the error message:

```
??? Error using ==> foo
```

Rebuild MEX-Files Created with MATLAB Versions Earlier Than V7 (R14)

To work with MATLAB V7.5 (R2007b), MEX-files compiled on any platform with MATLAB versions earlier than V7 (R14) no longer load correctly and must be rebuilt.

Changes to Compiler Support

The set of compilers that MATLAB supports has changed in MATLAB Version 7.5 (R2007b). For an up-to-date list of supported compilers, see the Supported and Compatible Compilers Web page.

New Compiler Support

MATLAB V7.5 (R2007b) supports new compilers for building MEX-files.

Windows platforms

- Microsoft Visual Studio 2005 SP1

Sun Solaris SPARC (64-bit) platform

- gcc / g++ Version 4.1.2

Discontinued Compiler Support

The following compilers are no longer supported.

Windows platforms

- Microsoft Visual Studio 2005 without SP1

Solaris SPARC (64-bit) platform

- gcc / g++ Version 3.2.3

Compatibility Considerations

To ensure continued support for building your C/C++ programs, consider upgrading to another supported compiler. For an up-to-date list of supported compilers, see the Supported and Compatible Compilers Web page.

Compiler Support to Be Phased Out

The following compilers are supported in Version 7.5 (R2007b), but will not be supported in a future version of MATLAB.

Windows (32-bit) platform

- Intel Visual Fortran Version 9.0

- Compaq Visual Fortran Version 6.6
- Intel C++ Version 7.1
- Borland C++Builder 6 Version 5.6
- Borland C++Builder 5 Version 5.5
- Borland C++ Compiler Version 5.5
- Compaq Visual Fortran Version 6.1

Changes to Applications Built with Borland 5.5 or 5.6 C Compilers

MATLAB applications built with Borland Version 5.5 or 5.6 C compilers have changed in MATLAB Version 7.5 (R2007b). MATLAB applications that run under Windows are now implemented as console applications, not Windows applications.

Compatibility Considerations

If you have customized the build process based on one of the `bcc*engmatopts.bat` options files, you must edit this file making changes to the appropriate `LINKFLAGS` statement, as described in the `.bat` file comments.

Environment Variable Required for mex with Microsoft Platform SDK Compiler

When you build a MEX-file, an engine application, or a MAT application on a Windows 64-bit platform using the Microsoft Platform SDK compiler, MATLAB requires that you define the environment variable `MSSdk`. The value of this environment variable is the path to the installation directory for Microsoft Platform SDK. The Microsoft Platform SDK installation program does not commonly define this environment variable. For example, you might set the environment variable as follows:

```
C:\Program Files\Microsoft Platform SDK
```

Environment Variables Required for mex with Intel Visual Fortran 9.0

When you build a MEX-file, an engine application, or a MAT application using Intel Visual Fortran 9.0, MATLAB requires that you define an environment variable for the Windows platform you are using.

Windows (32-bit) platform

Define the environment variable `VS71COMNTOOLS`. The value of this environment variable is the path to the `Common7\Tools` directory of the Visual Studio .NET 2002 or 2003 installation directory. (Intel Visual Fortran requires Visual Studio .NET 2002 or 2003 on 32-bit Windows platforms.) The Visual Studio .NET 2003 installation program commonly defines this environment variable. For example, you might set the environment variable as follows:

```
C:\Program Files\Microsoft Visual Studio .NET 2003\Common7\Tools
```

Windows x64 platform

Define the environment variable `MSSdk`. The value of this environment variable is the path to the installation directory for Microsoft Platform SDK for Windows Server 2003. (Intel Visual Fortran requires Microsoft Platform SDK for Windows Server 2003 on Windows x64 platforms.) The Microsoft Platform SDK installation program does not commonly define this environment variable. For example, the environment variable might have the value

```
C:\Program Files\Microsoft Platform SDK
```

Changes to Handling ActiveX Methods

In MATLAB Version 7.4 (R2007a), a change was made to Microsoft ActiveX® methods which was not documented in the release notes. This change is described in “Changes to Handling Microsoft ActiveX Methods” on page 19-44 in the MATLAB Release Notes, Version 7.4 (R2007a).

Changes to Dynamic Data Exchange (DDE) Documentation

In MATLAB V5.1, all development work for the Dynamic Data Exchange (DDE) server and client was stopped. This functionality is no longer documented in V7.5 (R2007b). MathWorks provides, instead, a MATLAB interface to COM technology that is documented in Using COM Objects from MATLAB .

Obsolete Functionality No Longer Documented

Documentation for Dynamic Data Exchange (DDE) is no longer included in External Interfaces.

Compatibility Considerations

If you must support this obsolete functionality, we suggest you print a copy of the External Interfaces chapter “COM and DDE Support (Windows Only)” from MATLAB V7.4 (R2007a) or earlier.

Documentation for Obsolete Functions To Be Phased Out

Documentation for the following functions is included in the MATLAB Function Reference documentation for MATLAB V7.5 (R2007b), but will not be included in a future version of MATLAB.

Obsolete Functions
`ddeadv`
`ddeexec`
`ddeinit`
`ddepoke`
`ddereq`
`ddeterm`
`ddeunadv`

The following syntax for enableservice, included in the MATLAB Function Reference documentation for MATLAB V7.5 (R2007b), will be removed from the documentation.

```
enableservice('DDEServer',enable)
```

Compatibility Considerations

If you must support this obsolete functionality, we suggest you print and keep a copy of the relevant MATLAB function reference pages from V7.5 (R2007b) or earlier.

R2007a

Desktop Tools and Development Environment

Startup and Shutdown

New features and changes introduced in Version 7.4 (R2007a) are

Double-Clicking Associated File Type in Explorer Now Opens File in Existing Session of MATLAB Software

When you open a file from Microsoft Windows Explorer whose type is associated with the MATLAB application, the file opens in the appropriate tool in the existing session of MATLAB, if MATLAB is already running, or if not, starts MATLAB. This assumes you associated the file types with MATLAB by accepting the defaults while installing MATLAB, or by changing the associations. For example, in Explorer, double-click an M-file to open the file in the MATLAB Editor/Debugger, or double-click a MAT-file to open the Import Wizard and load the data into the workspace in MATLAB. For details, including how to change file associations, see Associating Files with MATLAB on Windows Platforms.

Compatibility Considerations

In previous versions, when you double-clicked a file in Explorer whose type was associated with MATLAB, the file opened in a new session of MATLAB. The change made in MATLAB Version 7.4 (R2007a) to open the file in an existing session was based on many user requests.

In previous versions, double-clicking a MAT-file associated with MATLAB in Explorer opened a new session ofMATLAB and loaded the data into the workspace. When clicked in the Apple Macintosh Finder, the data loaded into the existing session of MATLAB.

Now on Microsoft Windows and Macintosh platforms, the Import Wizard opens in the existing session of MATLAB, and you use it to load the data into the workspace.

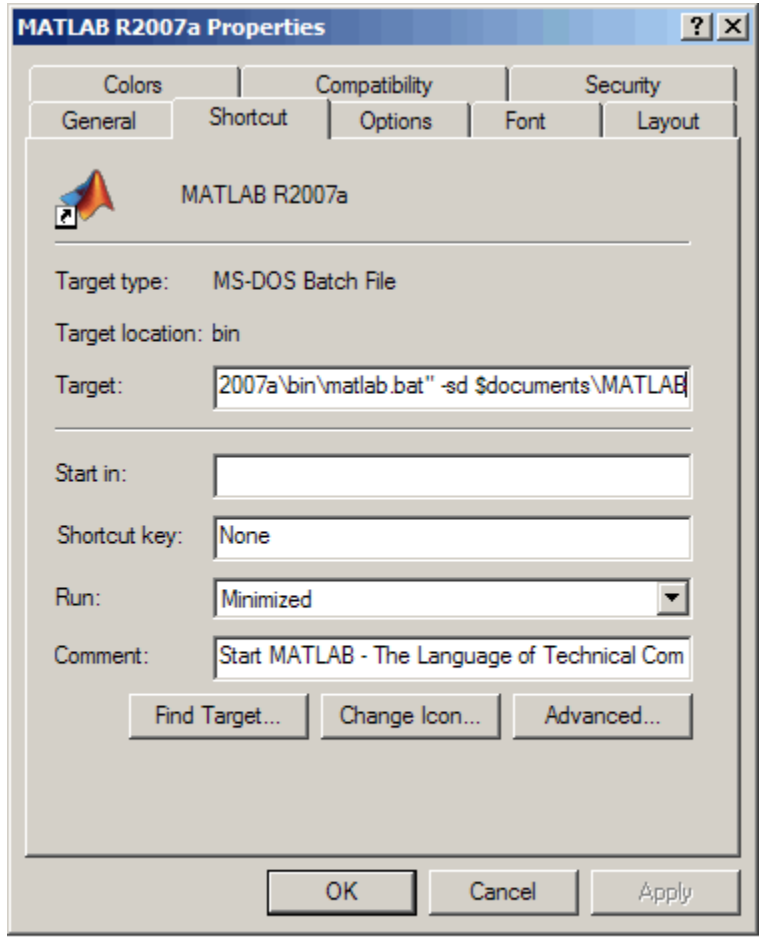
By default, in previous versions, double-clicking an M-file in Explorer opened it in the MATLAB stand-alone Editor. For more information about that change, see "Stand-Alone Editor No Longer Opens By Default; To Be Removed in a Future Version" on page 19-18.

Changes to Startup Directory (Folder) and Startup Options for MATLAB Application on Windows

The default startup directory for MATLAB on Microsoft Windows platforms is now `My Documents\MATLAB`, or `Documents\MATLAB` on the Microsoft Windows Vista platform. Upon startup, MATLAB automatically locates this `MATLAB` folder (creating it if it does not exist), and adds it to the top of the search path. This utilizes standard folders on Windows and Windows Vista platforms to provide a unique startup directory for each user. As such, the folder is part of theWindows (or Windows Vista) user's profile, and will be accessible when the user runs MATLAB from other machines, when the profile is set up to roam.

In the Properties dialog box of the shortcut icon for MATLAB, the **Target** field includes a new startup option `-sd` (for *s*tartup *d*irectory), followed by `$documents\MATLAB`, where MATLAB interprets `$documents` as the `My Documents` folder for the current user (or `Documents` for the Windows Vista platform). If the path for `$documents` is specified in the configuration of the Windows environment via UNC pathname, MATLAB automatically assigns it a mapped drive, creating one if necessary. The `-sd` startup option overrides anything in the **Start in** field, which is now empty by default.

For more information, see Changing the Startup Directory.



Compatibility Considerations

In previous releases, the default startup directory was `\Work`, located in the directory in which MATLAB was installed. For example, in R2006b, if you installed MATLAB in `C:\Program Files`, the default startup directory was `C:\Program Files\MATLAB\R2006b\Work`.

Windows Vista User Account Control (UAC) security features restrict access to `Program Files`. To accommodate this enhanced security model, the default startup directory in MATLAB has been moved outside of `Program Files`.

These are the differences between the startup directory for R2007a and previous releases:

- In previous releases, upon installation, the default startup directory was specified in the **Start in** field of the Properties dialog box for the shortcut icon for MATLAB. You could change the startup directory by replacing the pathname in that field. In R2007a, if you want to specify the startup directory using the **Start in** field, you must also remove from the **Target** field the `-sd` startup option and the pathname that follows it.
- The default startup directory is no longer specific to a release. When you upgrade from R2007a to a future release, files you created and saved in `My Documents\MATLAB` (or `Documents\MATLAB` on Vista) will automatically be in the startup directory for the new release. Previously, you had to move files you created and saved in the `Work` folder for a release, for example, `R2006a\Work`, to the default startup directory for the new release, for example, `R2006b\Work`.
- In previous releases, the default startup directory, `Work`, was shared by all users running that installation of MATLAB. The default startup directory in R2007a, `My Documents\MATLAB` (or `Documents\MATLAB` on Vista), provides each user with a unique startup directory.
- In previous releases, MATLAB added the default startup directory, `Work`, to the bottom of the search path. In R2007a, MATLAB adds the default startup directory, `My Documents\MATLAB` (or `Documents\MATLAB` on the Windows Vista platform), to the top of the search path. In R2007a, for consistency with previous releases, MATLAB adds `...\MATLAB\R2007a\Work` to the bottom of the search path. However, we encourage you to stop using the `Work` folder because support for it might be removed in a future release.

JVM for Windows Updated

MATLAB is now using Sun Microsystems Java (JVM) version 1.5.0_07 on Windows platforms. Java is supplied with MATLAB for Windows platforms, so this change requires no action on your part.

Compatibility Considerations

If you use a specific version of Java with MATLAB on Windows platforms, this change might impact you.

New Version of Java Access Bridge Software

MATLAB now installs Java Access Bridge 2.0, which is used by Freedom Scientific BLV Group JAWS® software for accessibility support.

Confirm Exit Preference to be Enabled by Default in Future Version

When you exit from MATLAB, MATLAB terminates without displaying a confirmation dialog box that asks if you are sure you want to quit. This is the default behavior but you can set a preference to display a confirmation dialog box. In a future version, the confirmation dialog box will appear by default when you quit MATLAB.

Compatibility Considerations

When the confirmation dialog appears by default, you will be able to disable it using preferences. If you have files that include exit statements, you might need to ensure the preference for the confirmation dialog box is disabled.

Desktop

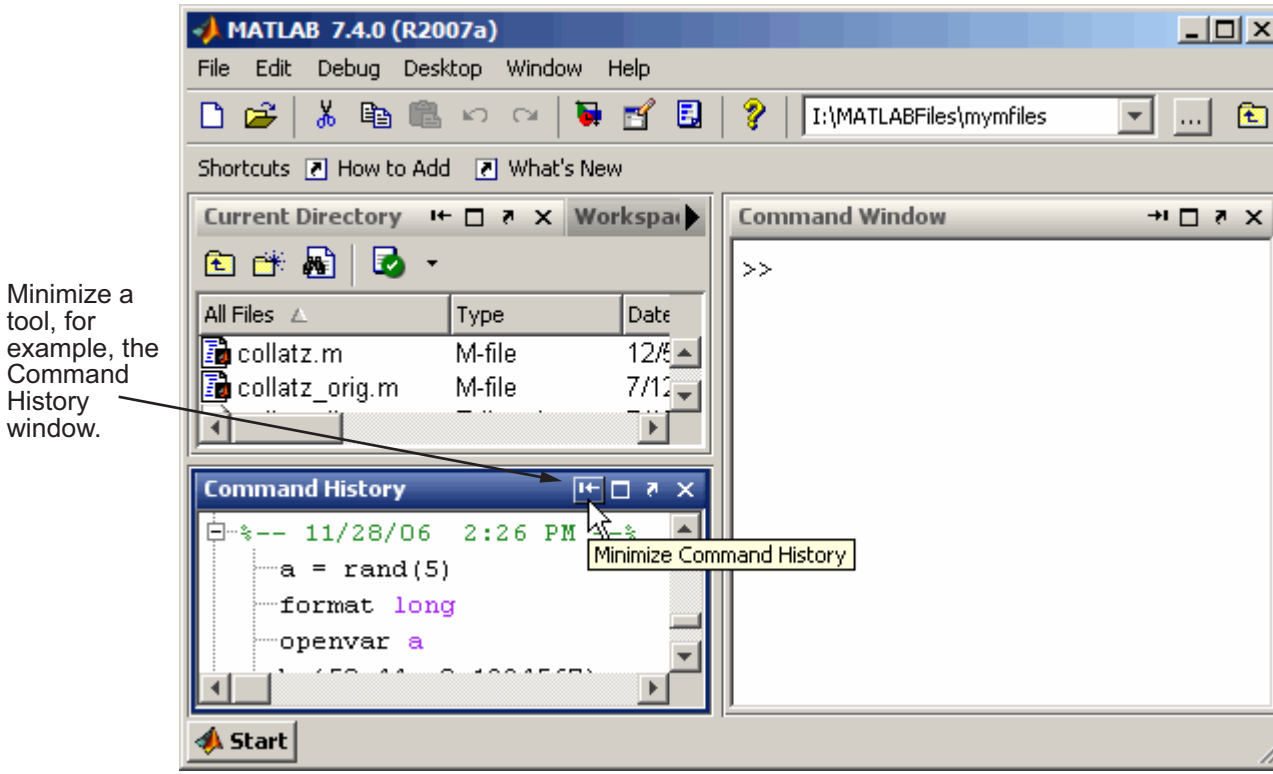
New features and changes introduced in Version 7.4 (R2007a) are

Minimize, Temporarily Display, and Restore Tools in the Desktop

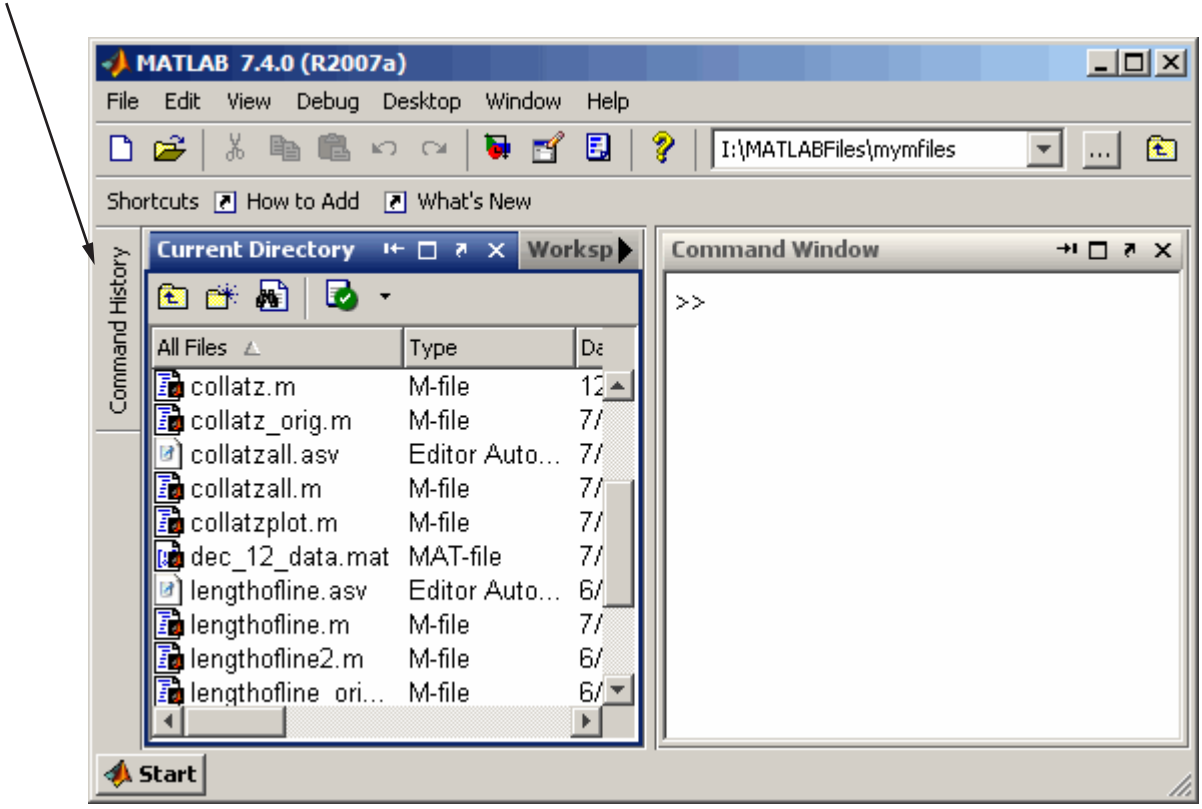
On Windows and UNIX platforms, you can minimize any tool in the desktop. Minimizing a tool creates a button for it along the specified edge. When you want to view the tool, hover over or click the button to temporarily show it in the desktop. Right-click the

button and select **Restore *toolname*** to return the tool to the desktop. Perform these tasks for the selected tool using items in the **Desktop** menu, equivalent mnemonics (for example **Alt+D, N** to minimize), or buttons on the tool's titlebar.

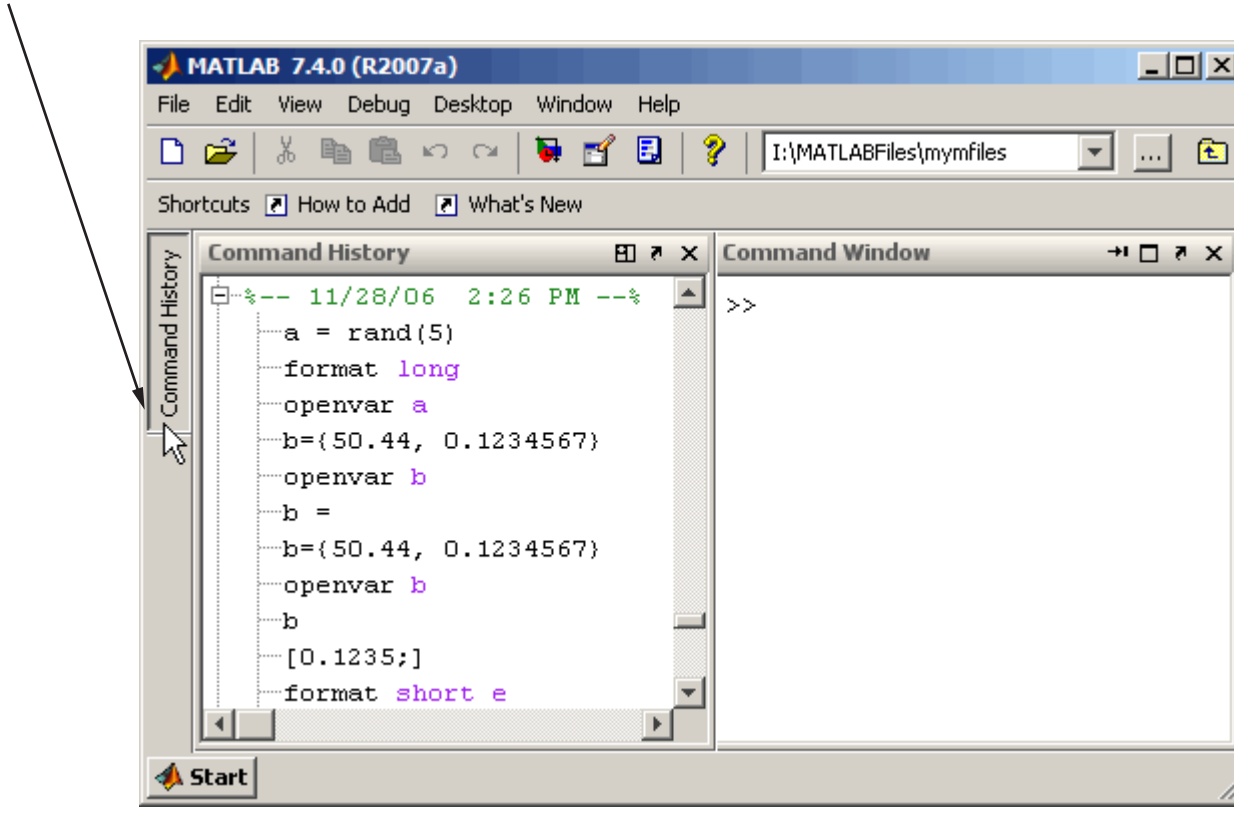
The following illustrations show how to use these features, using the example of the Command History window in the default desktop layout.



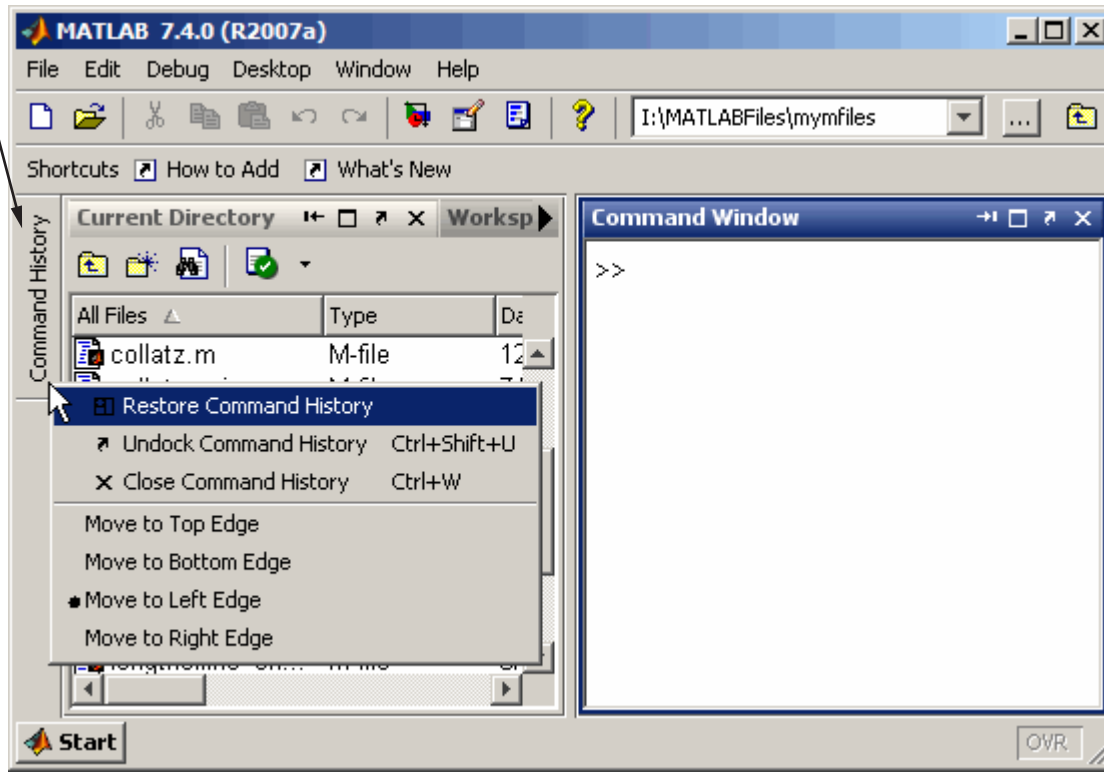
When minimized, a tool, such as the Command Window in this example, is represented by a button on the desktop border.



Hover over or click the button for a minimized tool to temporarily view the tool.
The tool is temporarily displayed until you select another tool.
Then the tool becomes minimized again.



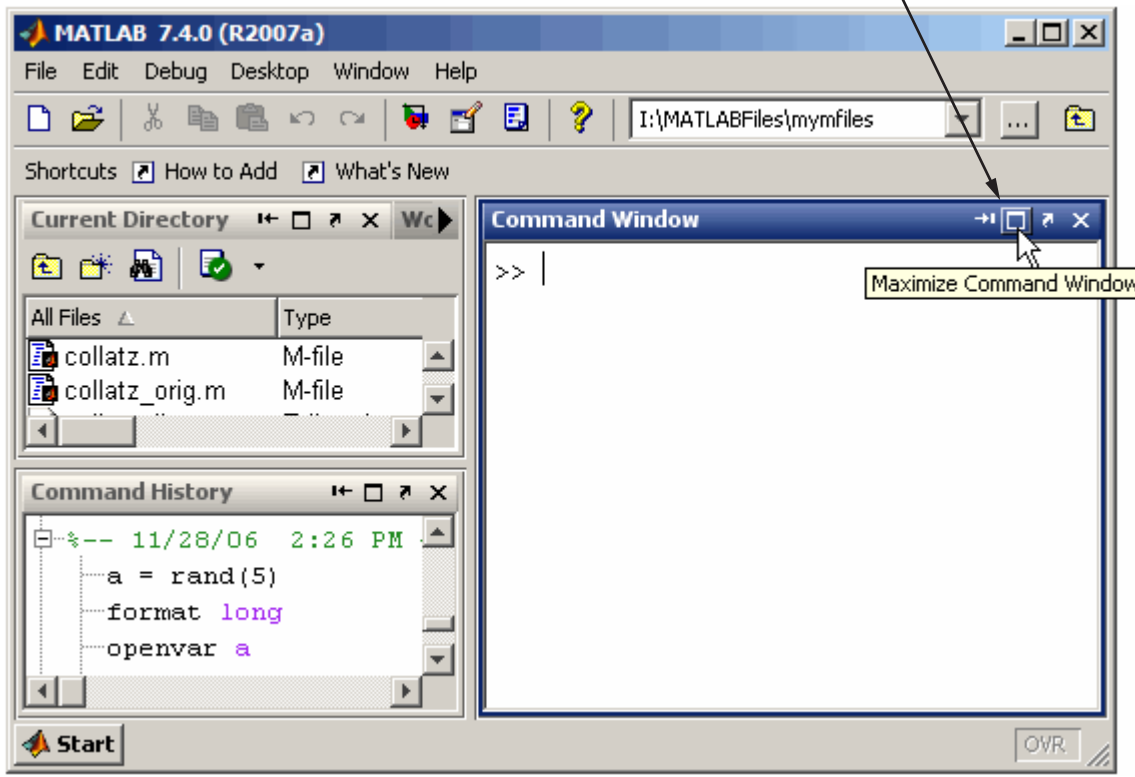
On the button for a minimized tool, right-click, and from the context menu, select **Restore**. The tool resumes the size and position it had in the desktop before it was minimized.

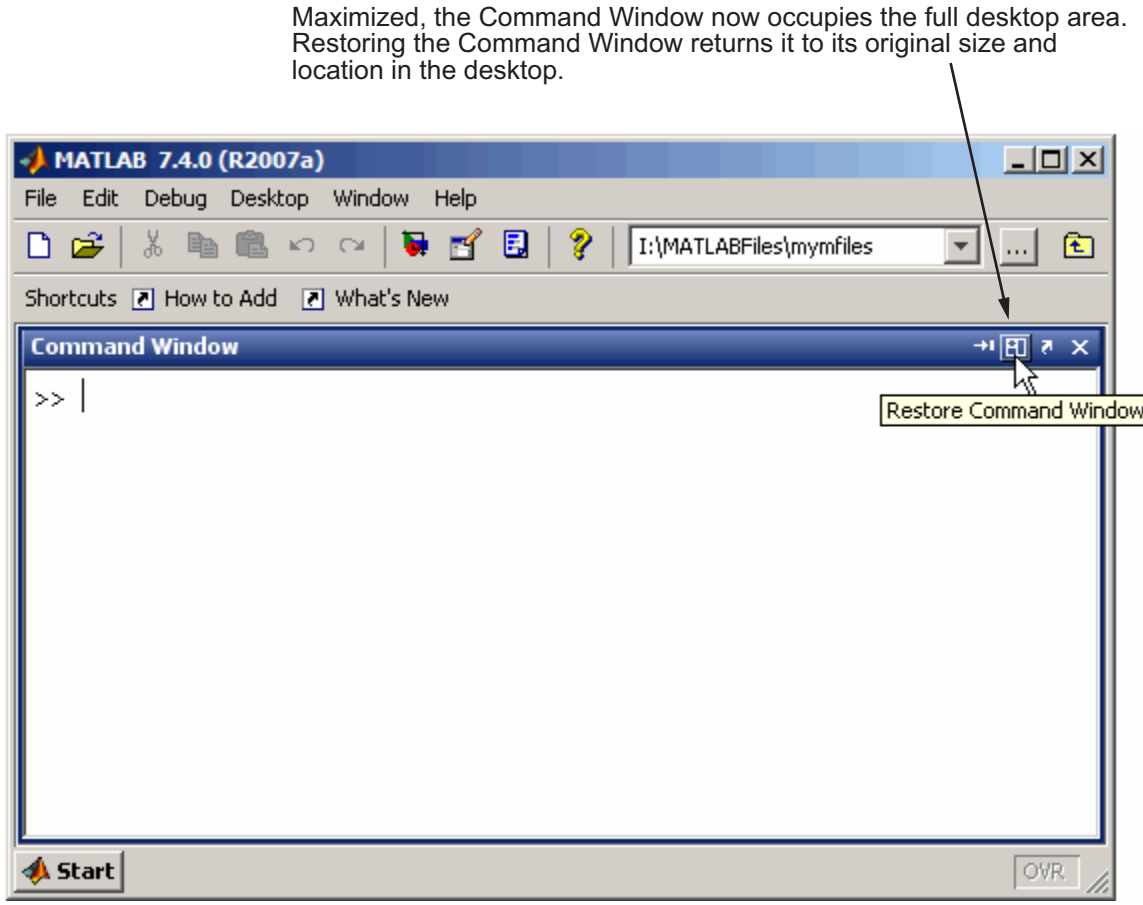


Maximize Tools in the Desktop

On Windows and UNIX platforms, you can maximize a tool so it occupies the entire desktop tool area in MATLAB, then restore it to return it to its previous location. Perform these tasks for the selected tool using items in the **Desktop** menu, equivalent mnemonics (for example **Alt+D, X** to maximize), or buttons on the tool's titlebar.

Default desktop layout.
Maximize a tool, for example, the Command Window
so it occupies the full MATLAB desktop area.

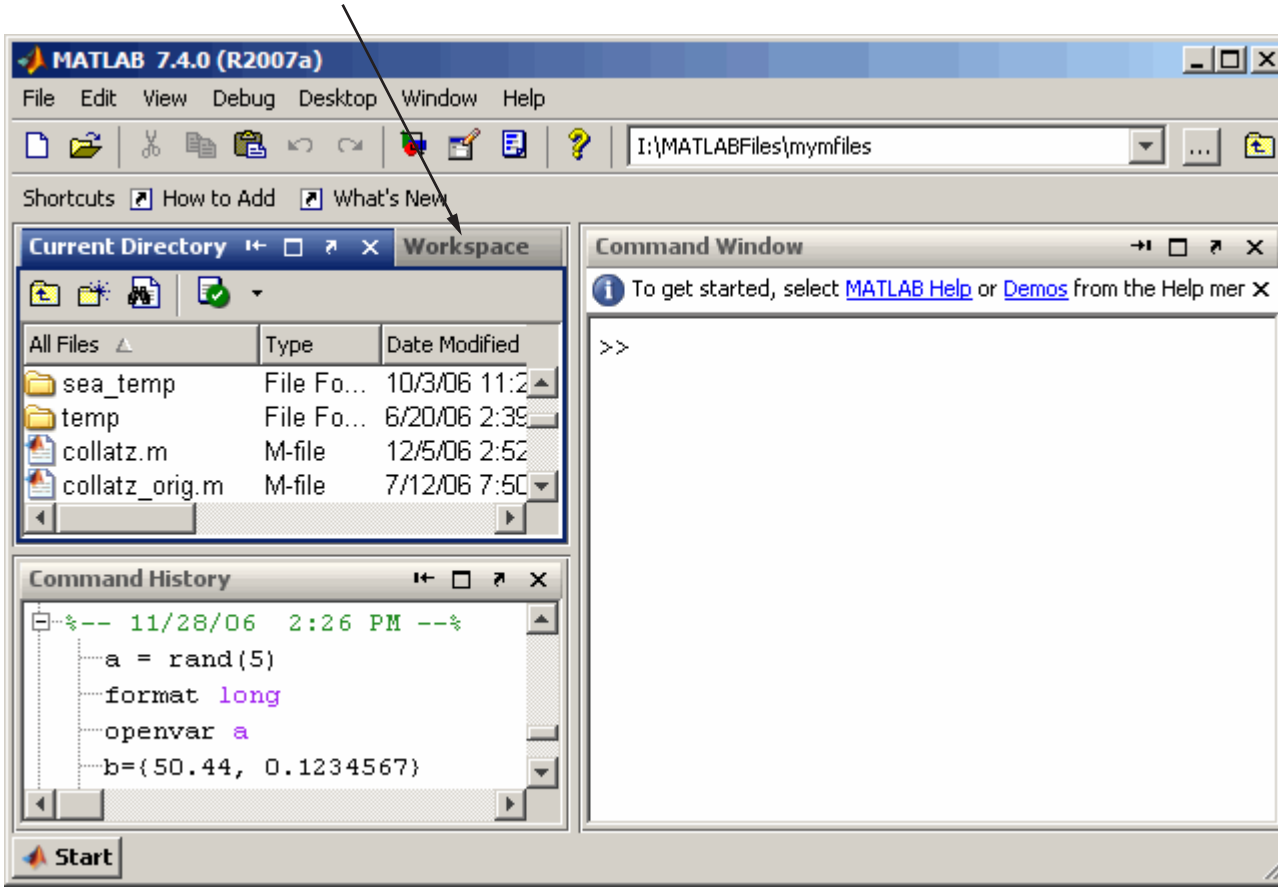




Tabs for Tools Replaced by Title Bars

In the desktop, when you tab tools together, that is, arrange them so they occupy the same position, the tools' title bars share the title bar area. To make a tool active, select its name in the title bar.

The Current Directory browser and Workspace browser are
"tabbed together".
The name of the tool in the title bar serves as the tab. To make a tool active, click its name in the title bar.
For example, click Workspace in the title bar to make the Workspace browser active.



Compatibility Considerations

In previous versions, tools tabbed together each had a tab at the bottom of the area they occupied. The tabs have been removed in favor of tools sharing the titlebar area.

Multithreaded Computation Support Added; Enable Via New Preference

If you run MATLAB on a multiple-CPU system (multiprocessor or multicore), use a new preference to enable multithreaded computation. This can increase performance in MATLAB for element-wise and BLAS library computations.

By default the preference is not set, so you must set it to enable multithreaded computation. With the preference enabled, MATLAB automatically specifies the recommended number of computational threads, although you can change that value. On AMD platforms running the Linux operating system, MATLAB supports multithreaded computation, but requires an extra step to change the default BLAS.

If you are using a multiple-CPU system, you can run the Multithreaded Computation demo to see the performance impact. For more information, see "Enabling Multithreaded Computation" in the MATLAB Desktop documentation.

Running Functions—Command Window and History

New features and changes introduced in Version 7.4 (R2007a) are

Startup Message Bar Replaces Startup Message in Command Window

When you start MATLAB, a getting started message bar appears at the top of the Command Window, that provides links to information that is helpful for new users. You can dismiss the message bar by using the close box in it. Use Command Window preferences to specify whether or not the message bar should appear.

Command History Searching Enhanced

You can now find entries in the Command History window by typing the first few letters of an entry; the previous entry that begins with those letters is selected. Then use up and down arrow keys to extend the find to the next and previous instances. Use the **Ctrl** key with an arrow key to select the current as well as next or previous instances. Use **Ctrl+A** to find and highlight all instances at once. The search does not look at entries in sessions that are collapsed. For more information about this search feature, see Quick Search for Entries Beginning with Specified Letters or Numbers.

Macintosh Platforms—Some Key Bindings in Command Window Changed

On Macintosh platforms, some key bindings were changed to make them more consistent with Mac OS XMac OS X standard behaviors.

Compatibility Considerations

Key bindings you have been used to in the Command Window might have changed.

Help

New features and changes introduced in Version 7.4 (R2007a) are

Demos Now Included in Search Results and Product Filtering

When you perform a search in the Help browser, the results now include code and text found in **Demos**. Also, the **Demos** listing in the Help Navigator pane now synchronizes with the demo currently open when you have the synchronization preference for the Help browser selected. For more information, see Search Syntax and Tips.

When you select **File > Preferences > Help** and enable the Product Filter, only demos for selected products are shown in the **Demos** pane, and searched via the **Search for** field.

Get URL of Displayed Page for Viewing on Web

When viewing a page in the Help browser, select **View > Page Locations**. A window appears providing the location of the current Help page you are viewing. The window provides the page location on both your local system and the MathWorks Web site. Use this feature to

- Send the URL to someone else who wants to view that information and might not have MATLAB or the same version of MATLAB, for example, a colleague or technical support.

- More easily see if this same documentation page has been updated for the latest product version.

Include Search Database for Your Own Help Files

If you create your own HTML help files for use with the MATLAB Help browser, the Help browser can now search the content of your files. Create a search database for your help files using the new builddocsearchdb function. The function creates a `helpsearch` directory that contains the search database files. For information on this process, see Making Your HTML Help Files Searchable.

Compatibility Considerations

In previous versions of MATLAB, some users created help search databases for their own help files via assistance from MathWorks Technical Support. If you created a help search database for use with R2006b, it might work in R2007a, but it is recommended that you recreate it for R2007a, even if no content has changed. If you created a help search database prior to R2006b, you must recreate it for R2007a.

Help search databases in prior releases were built from a help `jar` file rather than `html` files. The `builddocsearchdb` function only supports `html` files. However, it is expected to support `jar` files in a future release. If you used a `jar` file for a prior version and want to use a `jar` file in R2007a, build your R2007a help search database using the same technique you used in R2006b.

Video Tutorials Now Accessed Via Web Site

Previously video tutorials were installed when you installed MATLAB. Now the tutorials are on the MathWorks Web site.

Compatibility Considerations

You can still link to and play video tutorials from within the Help browser Demos listings or other links to them in the product and documentation, however this now requires an active Internet connection.

Workspace, Search Path, and File Operations

New features and changes introduced in Version 7.4 (R2007a) are described here.

Current Directory Browser Enhancements

- When you double-click a Windows shortcut in the Current Directory browser, it runs the shortcut.
- When you double-click a `prj` file in the Current Directory browser, it opens in the Deployment Tool.
- You can now find entries in the Current Directory browser by typing the first few letters of an entry; the entry that begins with those letters is selected.

Workspace Browser

When you double-click an object, it opens in the Property Inspector.

Array Editor Enhancements

You can now undo and redo the last operation you performed in the Array Editor. This applies to cut, paste, insert, delete, and clear contents features.

Search Path Changes

On Windows platforms, MATLAB now adds the default startup directory, `My Documents\MATLAB` (or `Documents\MATLAB` on Windows Vista), to the top of the search path upon startup.

Compatibility Considerations

In previous releases, MATLAB added the default startup directory, `Work`, to the bottom of the search path on Windows platforms. For example, in R2006b, it added `...\MATLAB\R2006b\Work` to the bottom of the search path. In R2007a, for consistency with previous releases, MATLAB adds `...\MATLAB\R2007a\Work` to the bottom of the search path. However, we encourage you to stop using the `Work` folder because support for it might be removed in a future release. For more information, see "Changes to Startup Directory (Folder) and Startup Options for MATLAB Application on Windows" on page 19-3.

Editing and Debugging M-Files

New features and changes introduced in Version 7.4 (R2007a) are

- "Stand-Alone Editor No Longer Opens By Default; To Be Removed in a Future Version" on page 19-18

Stand-Alone Editor No Longer Opens By Default; To Be Removed in a Future Version

Starting in this release, by default, double-clicking an M-file in Windows Explorer opens the file in the MATLAB Editor/Debugger rather than in the MATLAB stand-alone Editor. In a future version, the stand-alone Editor will not be provided with MATLAB.

Compatibility Considerations

The change to open M-files in the MATLAB Editor/Debugger rather than the stand-alone Editor was made based on many user requests. You can still use the MATLAB stand-alone Editor in this release by starting the application located at: *`matlabroot`*`\bin\win##\meditor.exe`.

Starting MATLAB by double-clicking an M-file requires a MATLAB license, while starting the stand-alone Editor does not require a MATLAB license.

If you want to associate M-files so that when you double-click them they open in the MATLAB stand-alone Editor rather than in the MATLAB Editor/Debugger, follow the instructions in Associating Files with MATLAB on Windows Platforms; rather than specifying the executable for MATLAB, `matlab.exe`, specify `meditor.exe`, which is in the same folder.

Instead of the stand-alone Editor, you can use the MATLAB Editor/Debugger. It provides all the features of the stand-alone Editor plus some not found in the stand-alone Editor, such as debugging and tab completion; for a full list, see "Stand-Alone Editor Will Not Be Included in Next Version" on page 18-15.

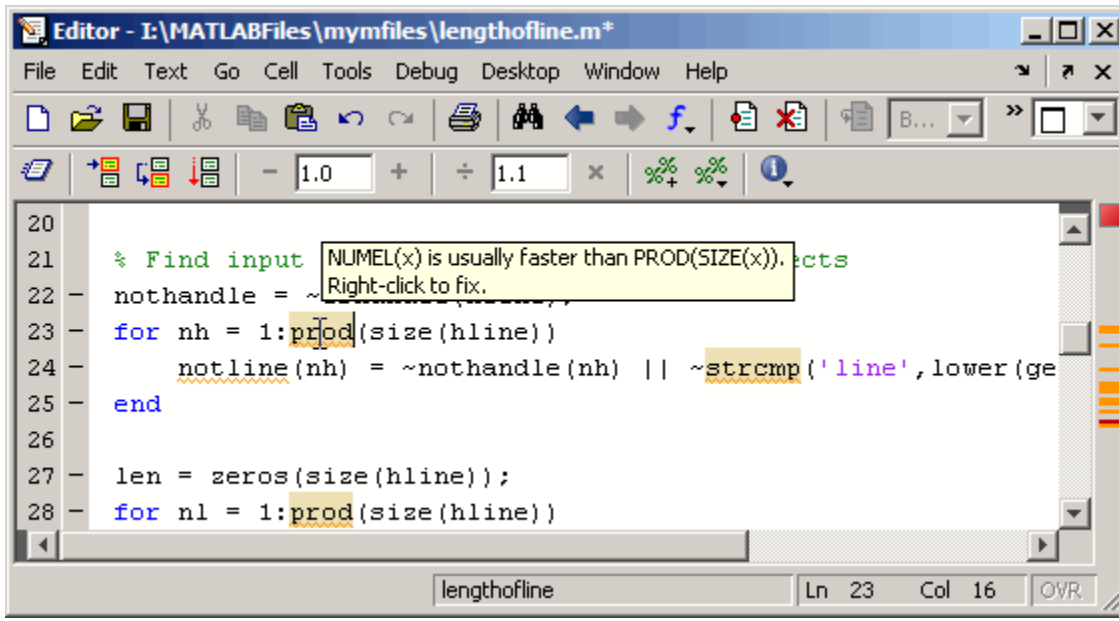
Some users have preferred the stand-alone Editor to the MATLAB Editor/Debugger because of slightly better startup performance and because it does not require a license for MATLAB. For those situations, you can use any text editor you have, such as UltraEdit or Emacs. If you have concerns about the pending removal of the stand-alone Editor, please contact us at editor-feedback@mathworks.com, so we can plan to minimize transition issues.

Delimiter Matching Extended to Include Language Keyword Pairs

You can now see the match to language keyword pairs using delimiter matching features that previously existed for parentheses and brackets. For example, when you type `end`, the Editor/Debugger highlights the matching `if`. To set delimiter matching preferences, select **File** > **Preferences** > **Keyboard** > **Delimiter Matching**; click **Help** for more information.

M-Lint Automatic Correction Feature

For some types of warnings or errors, M-Lint can apply an automatic fix to the code. Code that can be automatically corrected appears with a different background color. To perform the fix, right-click the code that is highlighted (for a single single-button mouse, use **Ctrl**+click); from the resulting context menu, select the auto-fix action.



For more information, see step 8 in the example at Automatically Check Code in the Editor — Code Analyzer.

M-Lint Detection of Missing End-of-Line Semicolons Enhanced

In previous versions, there was a single output message generated by a line with a missing terminating semicolon:

```
Terminate statement with semicolon to suppress output.
```

However, M-Lint suppressed displaying the message for files with three or more cells (a cell being indicated by a line with an initial %%). This was done based on the assumption that such files were demo programs, and therefore display of the output was intentional.

In MATLAB Version 7.4 (R2007a), M-Lint no longer makes a distinction for files with three or more cells. Instead, M-Lint displays one message for scripts and a different message for functions:

```
Terminate statement with semicolon to suppress output (in functions).
Terminate statement with semicolon to suppress output (in scripts).
```

The corresponding M-Lint tags to suppress display of such messages within a line are `%#ok<NOPRT>` for functions, and `%#ok<NOPTS>` for scripts.

By default, both of the messages are initially selected in Preferences.

Compatibility Considerations

In MATLAB Version 7.3 (R2006b), `%#ok<NOPRT>` was the only tag used to suppress the missing terminating semicolon. The tag will remain valid for function files, but will not be valid for script files. If you added any `%#ok<NOPRT>` tags in script files in R2006b, change those tags to the new tag `%#ok<NOPTS>`.

Macintosh Platforms—Some Key Bindings in Editor/Debugger Changed

On Macintosh platforms, some key bindings were changed to make them more consistent with standard behaviors for Mac OS X.

Compatibility Considerations

Key bindings you have been used to in the Editor/Debugger might have changed.

Other Changes in the Editor/Debugger

- There is a new confirmation preference for displaying a warning about exiting debug mode in order to save a file.

Tuning and Managing M-Files

There was a minor change in M-Lint behavior—for details, see “M-Lint Detection of Missing End-of-Line Semicolons Enhanced” on page 19-19.

Compatibility Considerations

See the compatibility considerations associated with "M-Lint Detection of Missing End-of-Line Semicolons Enhanced" on page 19-19.

Publishing Results

New features and changes introduced in Version 7.4 (R2007a) are

Publishing Function M-Files Now Supported

You can now publish an M-file function. When publishing an M-file function, you cannot evaluate the code. This feature effectively allows you to save M-file functions to output formats such as HTML or Microsoft Word documents, with formatting. To publish an M-file function, first clear the **Evaluate code** preference in **Editor/Debugger Publishing Preferences**, or run the publish function with the `evalCode` option set to `false`.

Specify Maximum Number of Output Lines in Published File

Using the publish function, you can specify the maximum number of lines in a published file. Set the new `maxOutputLines` field to a nonnegative value. The default value is `Inf`.

New Publishing Options

New publishing options provide increased flexibility and control over the appearance of the published document. The added options are for adding inline links, inline links with link text, graphics, and HTML markup. See Mark Up MATLAB Comments for Publishing for more information.

Mathematics

New Functions

Function	Description
bsxfun	Applies an element-by-element binary operation to two full arrays with singleton expansion enabled
ilu	Performs the sparse incomplete LU factorization

More Efficient Matrix Multiplication for Sparse Matrices

Matrix multiplication for `A'*b` now handles sparse matrices more efficiently.

rand Function Uses the Mersenne Twister Algorithm as Default

The rand function now uses the Mersenne Twister algorithm as default generator algorithm. This method generates double precision values in the closed interval [2^(-53), 1-2^(-53)], with a period of (2^19937-1)/2. Prior to this release, MATLAB used an algorithm known as the Subtract-with-Borrow (SWB) algorithm.

The `rand` function now produces different results than in previous releases. However, the results returned are still pseudorandom values drawn from a standard uniform distribution. Because the values returned by `rand` are intended to be random, this change should not affect your code.

Compatibility Considerations

There are several things to keep in mind regarding this change:

- If your code requires the exact values returned by `rand` in previous releases, you should use the statement

  ```
  rand('state',0);
  ```

  to reset `rand` to use the SWB algorithm. The new default algorithm's internal state at startup is equivalent to using the statement

```
rand('twister',5489);
```

Note that the `'state'` keyword corresponds specifically to the SWB algorithm; it cannot be used generally to refer to the internal state of the currently active algorithm.

- Existing code that uses the `'state'` keyword to reinitialize `rand` in a statement such as

  ```
  rand('state',sum(100*clock))
  ```

  causes `rand` to switch to using the SWB algorithm. You may want to use the `'twister'` keyword, which resets `rand` but does not switch algorithms.

- Existing code that uses the `'state'` keyword to save and restore the internal state of `rand` in statements such as

  ```
  savedState = rand('state');
   ...                              % code that uses rand
  rand('state',savedState);
  ```

  may no longer work as intended. Specifically, the first line of code saves the state of the SWB generator algorithm (see the `rand` documentation for details). If the default Mersenne Twister algorithm was the active one at that time, then using the saved state in the last line does not restore the `rand` internal state to its original conditions. Instead, it switches the `rand` algorithm to SWB. To save and restore the internal state of the Mersenne Twister algorithm, use the `'twister'` keyword.

Upgrade to BLAS Libraries

MATLAB now uses new versions of the Basic Linear Algebra Subroutine (BLAS) libraries. For Windows, Intel Mac, and Intel processors on Linux platforms, MATLAB supports the Intel Math Kernel Library (MKL) version 9.0. For AMD processors on Linux platforms, MATLAB uses the AMD Core Math Library (ACML) version 3.5. For the Solaris platform, MATLAB uses the Sun Performance Library from Sun Studio 11.

mode of Empty Array Now Returns NaN

The mode function, when operating on an empty array (`[]`), returns a 1-by-0 array in previous releases, while related functions mean, median, std, and var return `NaN` when given the same input. In this release, `mode` returns `NaN` for an empty array input.

Compatibility Considerations

Existing program code that relies on mode of an empty array to return an empty array should be modified.

Change to Syntax for Setting BLAS Library Version on Linux

If you change the BLAS library used by MATLAB on Linux platforms, MATLAB now loads libraries in the left-to-right order specified in the syntax. For example, to load the Intel MKL BLAS, from a system prompt, run

```
setenv BLAS_VERSION mkl.so:mklcompat.so
```

MATLAB loads `mkl.so` first, and then loads `mklcompat.so`.

This also applies if you edit `bin\$(ARCH)\blas.spec` directly.

Compatibility Considerations

This syntax differs from that used for Linux platforms in prior versions.

Data Analysis

Programming

New Functions

New Functions in this release are

Function	Description
assert	Generates an error when a specified condition is violated. Displays either the default error message or one that you have written.
ismac	Returns `true` if you are currently running one of the Mac OS X versions of MATLAB.
verLessThan	Compares the specified version number and toolbox name with the version of that same toolbox that is currently running. Use `verLessThan` in your functions when you want to write code that can run across multiple versions of MATLAB.

Parse Inputs with Consistently Implemented Mechanism

The new `inputParser` class provides a consistent mechanism for parsing and validating input arguments in the M-file functions you write. Using `inputParser` methods in the body of your function, you build a schema that represents each valid input argument to the function. In the schema, you can specify whether arguments are required or optional, and if they are to be passed as single values or as parameter-value pairs. The schema also provides a means of validating each incoming argument. The properties of the `inputParser` class give you additional information about arguments that are passed to the function.

For more information, see Parse Function Inputs in the MATLAB Programming documentation.

textscan Returns Like Values in Same Cell Array

In previous versions of MATLAB, the textscan function always returned each output value in a separate cell array, even if those values were of the same data type. In this release, if you call `textscan` with the new `CollectOutput` switch, MATLAB returns all consecutive data that is of the same type in the same cell array. This can save you the

extra task of sorting through the output and merging like data types together in your own code.

For more information, see the textscan function reference page.

Numbered Arguments for Formatted String Functions

Using numbered argument specification with MATLAB functions that employ format specifiers such as `%d` or `%s` (e.g., sprintf, error), you can pass the numeric and character string values that correspond to these format specifiers in a varying order. This can be useful when translating format strings in a different sentence structure.

In addition to identifying the values, you can also use numbered arguments to specify the field width and precision of the format specifiers. In the following example, the first format specifier in the sprintf command is `%1$*4$f` (quite different from the usual `%f` specifier). The `1$` tells MATLAB that the value that is to replace this format specifier is in the first argument following the format string; that is, 123.45678. The `*4$` indicates that the field width for this specifier is being passed in the fourth argument following the format string: 15.

In the second and third arguments (`%2$.*5$f` and `%3$*6$.*7$f`), the symbols `2$` and `3$` represent the values, `*5$` and `*7$` represent precisions, and `*6$` represents field width:

```
sprintf('%1$*4$f   %2$.*5$f   %3$*6$.*7$f', ...
123.45678, 16.42837, pi, 15, 3, 6, 4)

ans =
     123.456780   16.428   3.1416
```

For more information, see Formatting Strings in the MATLAB Programming documentation.

The dir Function Returns Additional datenum Field

The dir function now returns the date the file or directory was last modified in two formats: string and numeric. The numeric date value is not specific to any particular locale, and thus is compatible for international use:

```
fileinfo = dir('myfile.txt')
```

```
fileinfo =
       name: 'myfile.txt'
       date: '16-Mar-2006 13:34:01'
      bytes: 251
      isdir: 0
    datenum: 7.3275e+005
```

This new output is also returned when running dir on an FTP server.

Using whos -file on Objects with Overloaded size or class Methods

MATLAB is unable to determine the true size of an object stored in a MAT-file if the class of this object overloads the MATLAB size function. Likewise, MATLAB cannot determine the true class name of an object if it overloads the MATLAB class function. For these reasons, in previous versions of MATLAB, the command whos `-file` does not return or display object size and class accurately if the class of that object overloads these methods, but instead always returns `1x1` for the size and a default name for the class.

In this release, `whos -file` returns the empty matrix (`[]`) or displays a hyphen (`-`) for objects that overload `size`, and returns and displays `unknown` for objects that overload `class`.

Compatibility Considerations

If you use `whos -file` on objects in any of your programs, and if any of these objects overloads the `size` function, then you need to be aware that MATLAB now returns `[]` instead of a 2–element vector of ones in the `size` field of the output structure.

mat2str Returns Correct Output for Strings

The documentation for the mat2str function states that the `str` output of this function "is suitable for input to the eval function such that `eval(str)` produces the original matrix to within 15 digits of precision." The behavior of `mat2str` when given a character array as input, however, did not abide by this rule. This inconsistency has been fixed in this release.

In MATLAB Version 7.3, the eval command below generated an error.

```
s = mat2str('MATLAB')
s =
```

```
   'MATLAB'

eval(s)
ans =
   MATLAB
```

Compatibility Considerations

You might have to modify M-file programs that expect the previous behavior from mat2str.

Warning Generated by try-catch

To accommodate future changes in the MATLAB error-handling capabilities, there is a new restriction to the syntax of the try-catch block. When the first MATLAB statement that follows the `try` keyword consists of just a single term (e.g., `A` as opposed to `A+B`) occurring on the same line as the `try`, then that statement and the `try` keyword should be separated by a comma. For example, the line

```
try A
```

should be written as either

```
try, A
```

or on two lines as

```
try
   A
```

This affects only single-term statements. For example, the following statement continues to be valid:

```
try A+B
```

The same holds true for the `catch` keyword and a single-term statement following the keyword on the same line. A valid `try-catch` statement of this type should be composed as follows:

```
try, A, catch, B, end
```

If you omit the commas following try and/or catch, your code will continue to operate correctly. However, MATLAB will issue a warning:

```
try statements, catch statements, end

Warning: This try-catch syntax will continue to work in R2007a,
but may be illegal or may mean something different in future
releases of MATLAB.
```

As with previous releases, the recommended syntax for a `try-catch` block is as follows:

```
try
   try_statements
catch
   catch_statements
end
```

Note: Due to a bug in the R2007a release, the warning for a `catch` followed immediately by a single term is thrown as an error, even though the text of the message says that it is a warning.

Compatibility Considerations

Your M-file programs may generate the warning shown above if the correct syntax for `try` and `catch` is not used.

save -regexp Saves to Correct Filename

In previous releases, the following command mistook the `-regexp` argument (`[ab]` in the case shown below) to be the name of the file to save to:

```
save -regexp [ab]
```

In this release, the filename is correctly understood to be the default save filename, `matlab.mat`, and any arguments that follow the `-regexp` flag and precede the next flag are interpreted as patterns to be matched.

Functions Not Callable If in Directory Under Class Directory

M-files placed in a directory that is one or more levels beneath a MATLAB class directory are not callable from that same directory. A MATLAB class directory contains methods of a class and has a filename that begins with the @ character.

In the following example, function `myfun1` is not callable if the current working directory is set to `dir1`. The same holds true for `myfun2` if the current working directory is set to `dir2`.

```
\home\matlab\@myobj\dir1\myfun1.m
\home\matlab\@myobj\dir1\dir2\myfun2.m
```

This behavior existed in the R2006b release, as well as in this release.

Compatibility Considerations

You will no longer be able to call a file that is one or more levels beneath a MATLAB class directory if your current working directory is set to that same directory.

Improved Performance on Certain Platforms and Operations

As of release R2006b, MATLAB offers improved performance in the following areas:

- Improved performance on 64-bit Windows XP and Linux platforms. This is independent of the size of data set in use.
- Faster scalar indexing into cell arrays.
- Faster assignment of cell array data to variables.

Technique to Conserve Memory on Windows Vista

To conserve memory on machines running Windows Vista, you can reduce the amount of virtual memory space reserved by the operating system by using the command:

```
BCDEdit /set increaseuserva 3072
```

More documentation on this option can be found at the following URL: `http://msdn2.microsoft.com/en-us/library/aa906211.aspx`.

ispuma Function Deprecated

Because MATLAB no longer supports Mac OS X 10.1, the `ispuma` function always returns `false` in this release, and also displays a warning message that the function is now deprecated. Support for `ispuma` will be removed in a future release of MATLAB.

Compatibility Considerations

It is recommended that you remove calls to `ispuma` from your M-file functions.

Graphics and 3-D Visualization

Nudging Annotations with Arrow Keys

In plot edit mode, annotations such as textboxes, lines, arrows, doublearrows, and text now respond to pressing directional arrow keys. Each keypress will move the selected annotation(s) one or more pixels in the indicated direction. If you select multiple objects, they all move together in response to arrow key strokes. Normally, selected objects move one pixel with each press of an arrow key. If you have selected **Snap to Layout Grid** from the **Tools** menu, each keypress makes objects move to the next grid position.

Movies No Longer Play During Loading

The `movie` function no longer plays each frame one extra time. Previously, it would show frames as they were loaded into memory to speed up display. This behavior is no longer required and has been eliminated.

Compatibility Considerations

If you have code that calls `movie` in a loop a certain number of times and want that number to remain the same, you need to add one iteration to the loop.

Ghostscript Printing Software Upgraded

The Ghostscript software used by some print devices has been upgraded from an older version to Version 8.54. Starting in this release, MathWorks is no longer shipping a separate, standalone Ghostscript executable program.

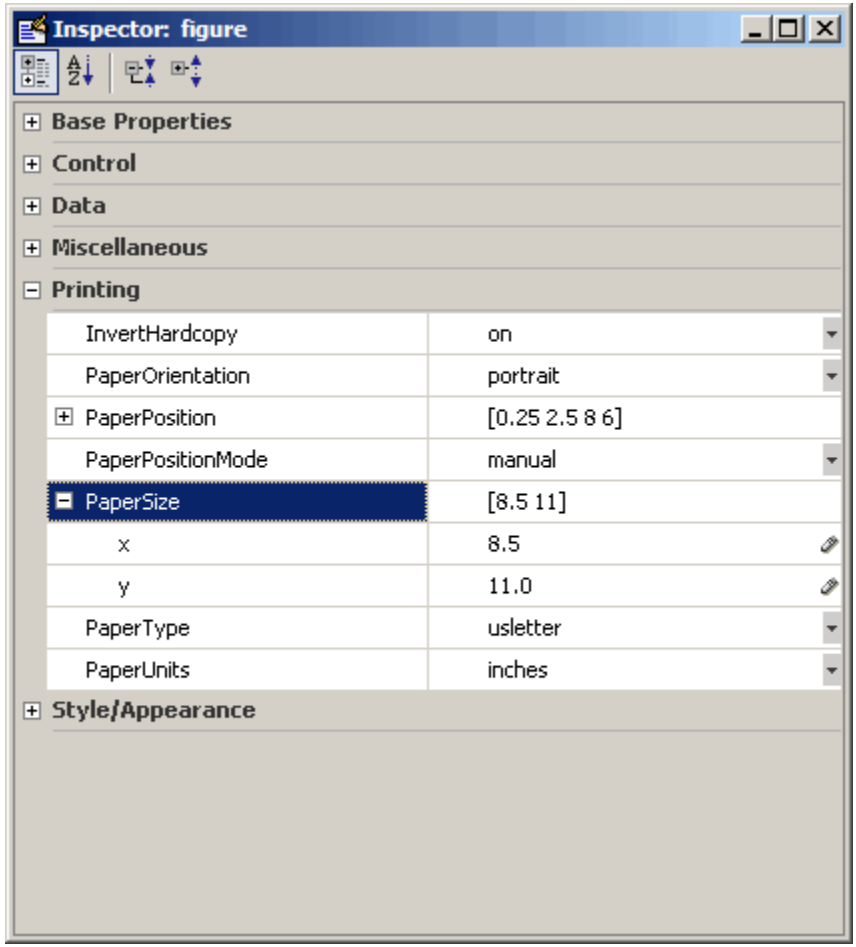
Compatibility Considerations

If you have written M-code based on undocumented functions that called the old version of Ghostscript, your code will no longer work.

As a consequence of this upgrade, support for printing on DEC LN03 printers (`print -dln03`) has been removed from MATLAB.

Property Inspector Now Categorizes Graphic Object Properties

The Property Inspector (accessed via the Property Editor, GUIDE, and the `inspect` function) now provides tree views of groups of graphic object properties as well as an alphabetical list of all properties. You can switch between views via the two buttons at the top of the window, as shown in the following picture of the tree view of the Printing category:



In addition, the type of object currently being inspected is now shown in the title bar of the Property Inspector window. Formerly it was shown in the gray area below the title bar, which now contains the view manipulation buttons.

To change view, click a button without the blue background (the collapse and expand buttons at right are disabled in alphabetic list view). When you change views, the selected property stays the same. To inspect properties of graphic objects in the tree view, click the + icon next to a category of interest. Do the same to open properties within a category that have multiple elements. Note that only Handle Graphics objects have categories; annotation and most other MATLAB objects can be displayed only in alphabetic list view.

The functionality of the Property Inspector has not changed; only its GUI has. As previously, if you select a set or array of objects and inspect them, the Property Inspector only displays those properties that all the objects share.

For details on using the Property Inspector, see Accessing Object Properties with the Property Inspector in the MATLAB Graphics documentation and the inspect reference page.

Compatibility Considerations

Text that you type or delete after clicking a text field that has a text widget icon (for example, `XTickLabel`), will not persist unless you press **Return** before clicking elsewhere. This mimics the behavior of clicking the text widget icon to open a text entry dialog box, typing into it, and accepting the result by clicking **OK**. This behavior differs from that which existed prior to R2006b (MATLAB V. 7.3).

Figure WindowScrollWheelFcn Property Programs Scrollwheel Events

The new figure property `WindowScrollWheelFcn` enables you to write a callback function that handles mouse wheel scrolling events while the figure has focus. See the documentation for the figure WindowScrollWheelFcn for more information.

Figure KeyReleaseFcn Property Programs Key Release Events

The new figure property `KeyReleaseFcn` enables you to write a callback function that handles key release events (analogous to `KeyPressFcn`). See the documentation for the figure KeyReleaseFcn for more information.

Creating Graphical User Interfaces (GUIs)

GUIDE No Longer Copies Callbacks When You Duplicate Components

In GUIDE, when you copy a component for which one or more of its callback properties are defined, the callback properties of the newly created component are now set to their default values, the ones you get when adding a component directly from the Component Palette. Previously, their values were copied from the corresponding callback properties of the original component. New callback subfunctions can be generated in the GUI M-file for the new component in the same way as for any component in the GUIDE layout.

Compatibility Considerations

If you have relied on the old behavior, in which the callbacks properties of the new component point to the callbacks of the original component, you will now have to explicitly change the callback properties, using the Property Inspector, to do that.

GUIDE M-File-Defined handles Structure Fields No Longer Become Permanent

The `handles` structure that is provided to every GUIDE generated callback subfunction in the GUI M-file is constructed at runtime every time the GUI runs. When the GUI is fully initialized, the `handles` structure contains only handles to all the components in the GUI and custom data added in any `CreatedFcn` callbacks and/or the `OpeningFcn`. Previously, fields that the M-file had added to the handles structure could sometimes become a permanent part of the structure.

Compatibility Considerations

If your GUI runs well but displays error messages about missing fields in the `handles` structure when starting up, you may have to update the code in the GUI M-file where those fields are accessed. You may need to initialize those `handles` fields properly in a `CreateFcn` callback and/or in the `OpeningFcn`.

UNIX: File Dialog 'Location' Property Is Obsolete

For UNIX platforms, the `Location` property of `uigetfile` and `uiputfile` is obsolete. Previously, for UNIX platforms only, you could use the `Location` property to specify the screen location at which the dialog box would originally be displayed.

Compatibility Considerations

Since the UNIX `uigetfile` and `uiputfile` `Location` property is now obsolete, MATLAB ignores any occurrences in existing code of the `Location` property

```
uigetfile(...,'Location',[X Y])
```

or the older use of `X` and `Y` arguments to specify location

```
uigetfile(...,X,Y)
```

A warning is displayed for either syntax, on all platforms.

Functions Becoming Obsolete

The following functions are either obsolete or grandfathered in MATLAB 7.4 (R2007a): `cshelp`, `figflag`, `getstatus`, `menulabel`, `popupstr`,`setstatus`, `hidegui`, `uigettoolbar`.

Compatibility Considerations

The functions shown in the following table will continue to work but their use may generate a warning message. As soon as possible, replace any occurrences you may have of these functions with the function(s) shown in the following table, if any, or with other suitable code.

Function	Replacement
`cshelp`, `figflag`, `getstatus`, `menulabel`, `popupstr`, `setstatus`	Grandfathered. There are no replacements. No enhancements will be made and only critical bugs will be fixed. No warnings will be displayed. These functions may become obsolete if replacements become available.

Function	Replacement
`hidegui`	Obsoleted. Now issues warnings. Set the figure `HandleVisibility` property instead.
`uigettoolbar`	Obsoleted. No warnings will be displayed until a replacement is available in a future release.

External Interfaces/API

New File Extensions for MEX-Files

MEX-Files in MATLAB for Apple Macintosh (Intel)

With the introduction of MATLAB for Macintosh (Intel), the MEX-files you build have the extension `.mexmaci`. The mexext command in MATLAB returns `mexmaci` for Macintosh (Intel).

Compatibility Considerations

MEX-files built using MATLAB for Macintosh PowerPC®, which have `.mexmac` extensions, cannot be used in MATLAB for Macintosh (Intel).

Note: All MEX-files on Macintosh platforms need to be recompiled for R2007a.

MEX-Files in MATLAB for 64-bit Sun Solaris SPARC

With the introduction of MATLAB for 64–bit Solaris SPARC®, you can now build 64-bit MEX-files for the Solaris platform. These MEX-files have the extension `.mexs64`. The mexext command in MATLAB returns `mexs64` for Solaris SPARC.

Compatibility Considerations

MEX-files built using MATLAB for Solaris 32, which have `.mexsol` extensions, cannot be used in MATLAB for Solaris SPARC.

Note: All MEX-files on Solaris 64 platforms need to be recompiled for R2007a.

MEX-Files Built in MATLAB R11 or Earlier Must Be Rebuilt

In order to work with MATLAB V7.4 (R2007a), MEX-files compiled on Microsoft Windows (32–bit) platforms with MATLAB R11 or earlier will no longer load correctly and must be recompiled.

Compatibility Considerations

Recompile these MEX-files with MATLAB R12 or later.

Changes to Compiler Support

The set of compilers that MATLAB supports has changed in MATLAB Version 7.4 (R2007a). For an up-to-date list of supported compilers, see the Supported and Compatible Compilers Web page.

New Compiler Support

MATLAB V7.4 (R2007a) supports new compilers for building MEX-files.

Windows (64-bit) platform

- Intel C++ version 9.1
- Intel Visual Fortran version 9.1

Windows (32-bit) platform

- Intel C++ version 9.1
- Intel Visual Fortran version 9.1
- Microsoft Visual C++ 2005 version 8.0 Express Edition

Note: If you use the `mex -f` *`matlabroot`*`/bin/`*`$ARCH`*`/mexopts/msvc80freeopts.bat` switch to build a MEX-file using Microsoft Visual C++ 2005 Express Edition, the environment variable `MSSdk` must be defined. The value of this environment variable is the path to the installation directory for Microsoft Platform SDK for Windows Server 2003. (Microsoft Visual C++ 2005 Express Edition requires Microsoft Platform SDK for Windows Server 2003.)

Macintosh PowerPC and Macintosh (Intel) platforms

- Apple Xcode 2.4.1 (gcc / g++ version 4.0.1)
- g95 version 0.90

Note: All MEX-files on Macintosh platforms need to be recompiled for R2007a.

Linux (64-bit) platform

- gcc / g++ version 4.1.1
- g95 version 0.90

Linux (32-bit) platform

- gcc / g++ version 4.1.1
- g95 version 0.90

Note: All Fortran MEX-files compiled on Linux platforms need to be recompiled for R2007a.

Solaris SPARC (64-bit) platform

- cc / CC version 5.8
- gcc / g++ version 3.2.3
- f90 version 8.2

Fortran Compatibility Considerations

In R2007a we have added support for a new Fortran compiler g95 on the Linux and Macintosh platforms. This compiler implements the Fortran 95 language standard. It replaces previously supported Fortran compilers which implemented a previous language standard.

This may cause incompatibilities in your Fortran source code for MEX-files.

Compatibility Considerations

Refer to the IBM XL Fortran V10.1 for Linux Language standards Web site `http://publib.boulder.ibm.com/infocenter/lnxpcomp/v8v101/index.jsp?topic=/com.ibm.xlf101l.doc/xlflr/languagestandards.htm` for information about incompatibilities between language standards.

Discontinued Compiler Support

The following compilers are no longer supported.

Linux (64–bit) platform

- gcc / g++ version 3.4.5
- g77 version 3.4.5

Linux (32–bit) platform

- gcc / g++ version 3.4.5
- g77 version 3.4.5

Macintosh PowerPC platform

- gcc / g++ version 3.3
- Absoft f77 / f90 version 8.2a

Compatibility Considerations

To ensure continued support for building your C/C++ programs, consider upgrading to another supported compiler. For an up-to-date list of supported compilers, see the Supported and Compatible Compilers Web page.

Compiler Support To Be Phased Out

The following compilers are supported in Version 7.4 (R2007a) but will not be supported in a future version of MATLAB.

Windows (32-bit) platform

- Intel C++ version 7.1
- Intel Visual Fortran 9.0
- Borland C++Builder 6 version 5.6

- Borland C++Builder 5 version 5.5
- Borland C++ Compiler version 5.5
- Compaq Visual Fortran version 6.6
- Compaq Visual Fortran version 6.1

Additional Linker Support for Intel Fortran

MATLAB V7.4 (R2007a) supports new linkers for building MEX-files with Intel Visual Fortran 9.0.

Windows (32-bit) platform

- Microsoft Visual Studio .NET 2003
- Microsoft Visual Studio 2005

Windows (64-bit) platform

- Microsoft Visual Studio 2005
- Microsoft Platform SDK for Windows Server 2003 (Build 1289 or later)

New COM Features

Programmatically Connect to Instances of a COM Automation Server

MATLAB can now programmatically connect to an instance of a COM Automation server using the new actxGetRunningServer function.

Improve the Custom Interface API

Changes to the actxserver function allow you to create a COM Automation server using a custom interface.

New COM Data Type Support

Additional COM data type support has been added. See Handling COM Data in MATLAB Software for a description of supported data types.

Enhanced Support for COM Interface Events

MATLAB users can take full advantage of event interfaces provided by COM Automation servers. This change is implemented in the events, eventlisteners, isevent, registerevent, unregisterevent, and unregisterallevents functions.

For an example of this feature, see Responding to Interface Events from an Automation Server.

Get the Status of a MATLAB Automation Server

Using the enableservice function you can learn the current state of a MATLAB Automation server. The function returns a logical value, where logical `1` (`true`) means MATLAB is an Automation server and logical `0` (`false`) means MATLAB is not an Automation server.

For example, if you type

```
enableservice('AutomationServer')
```

and MATLAB displays

```
ans =

     1
```

then MATLAB is currently an Automation server.

Changes to MATLAB Version-Specific ProgID

A programmatic identifier, or ProgID, is used to create a COM component. MATLAB's version-specific ProgID Matlab.Application.N.M now let's you specify both a major and minor version number.

For example, to specify MATLAB version 7.4, use the ProgID `Matlab.Application.7.4`.

Changes to Handling Microsoft ActiveX Methods

Beginning in MATLAB Version 7.4 (R2007a), an ActiveX method with the same name as a class is treated as a constructor and cannot be called in the same way as an ordinary method.

In the following example:

```
myApp = actxserver('Excel.Application');
op = invoke(myApp.Workbooks, 'open', 'MyFile.xls');
Sheets = myApp.ActiveWorkBook.Sheets;
target_sheet = get(Sheets, 'item','Sheet1');
```

```
invoke(target_sheet, 'Activate');
Activesheet = myApp.Activesheet;
cellname = 'B2';
Range = Activesheet.cells.Range(cellname,cellname);
```

the term `Range` is both a function on the MATLAB path and a constructor of the class `Range`. MATLAB tries to execute the function `range`, which generates the error:

```
??? Error using ==> range
Too many input arguments.

Error in ==> MyScript at 8
Range = Activesheet.cells.Range(cellname,cellname);
```

To get the property value, use the get function.

For example:

```
Range = get(Activesheet.cells, 'Range', cellname, cellname);
```

R2006b

Desktop Tools and Development Environment

Startup and Shutdown

New features and changes introduced in Version 7.3 (R2006b) are described here.

Associate Files from MATLAB Program with Windows Operating System Using New Utility

You can run a utility from the Help browser to associate `.m`, `.mat`, `.fig`, `.p`, and `.mdl` files with the MATLAB program in the Microsoft Windows operating system. After running the utility, you will be able to start MATLAB by double-clicking any of those file types in Windows Explorer. You can still use Windows Explorer Folder Options to perform these file associations, but the utility makes the task more convenient. For details, see Associating Files with MATLAB on Windows Platforms.

Redirect Output on UNIX Now Sends Errors to Shell

When you start MATLAB on platforms running The Open Group UNIX operating system using the `-nodesktop` startup option, and you redirect output, for example to a file, MATLAB now sends any errors to the shell, while normal output goes to the redirect target. This change was made so that redirection withMATLAB follows standard behavior for the UNIX operating system.

For example:

```
matlab -nodesktop -"r magic(3), magi(5)" > test.txt
```

starts MATLAB in `nodesktop` mode, runs the statement `magic(3)` and writes the output to `test.txt`. When MATLAB runs `magi(5)`, execution fails and MATLAB displays the error message in the shell.

Compatibility Considerations

In previous versions, MATLAB redirected both output and error messages, so in the above example, MATLAB wrote the output of `magic(3)` as well as the error message from `magi(5)` to `test.txt`.

If you have shell scripts that use > to redirect output, and you rely on errors appearing in the output target, you need to modify the `matlab` startup statements in those scripts.

To achieve the former behavior, that is, to redirect both output and errors to the specified target, use that specific redirect syntax for your shell. For example, in tcsh, use `>&`, as in

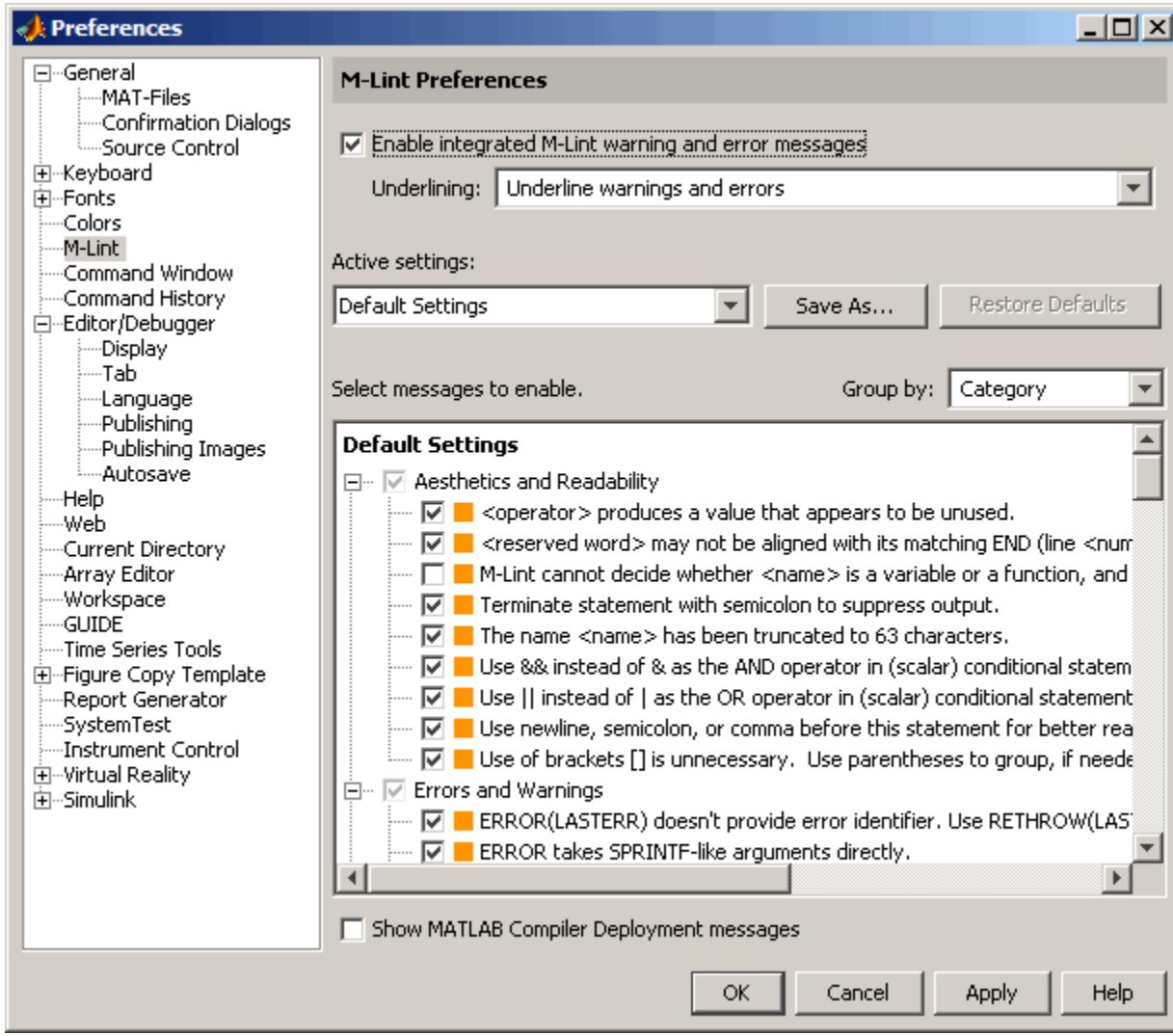
```
matlab -nodesktop -"r magic(3), magi(5)" >& test.txt
```

Desktop

New features and changes introduced in Version 7.3 (R2006b) are described here.

M-Lint Preferences Added and Now Appear on M-Lint Panel

M-Lint preferences now appear on a new M-Lint panel. There are new M-Lint preferences to disable specific messages or categories of messages, as well as to save the settings for use in a later session. In addition, you can choose to show or hide messages for the MATLAB Compiler product if the MATLAB Compiler product is installed. These new M-Lint preferences apply to the M-Lint code analyzer operating automatically in the Editor/Debugger as well as to the M-Lint Code Check Report you run via the Current Directory browser Directory Reports or **Tools > M-Lint > Show M-Lint Report**.



Compatibility Considerations

M-Lint preferences were previously accessed via the Editor/Debugger Language settings for M.

Close All Documents and Close Selected Documents Feature Added

When you have multiple documents open within a tool, such as M-files in the Editor/Debugger, select **File > Close** to readily close selected files in that tool. Alternatively, right-click the document bar to close all open documents or all open documents except the selected document in that tool.

Accessibility Documentation Included

Accessibility features and assistive technologies are now part of the Desktop documentation. In prior versions, they were documented in the general Release Notes.

New Look and Feel on Linux and Solaris Platforms

MATLAB has a new look and feel on the Linux operating system from Linus Torvalds and the Sun Microsystems Solaris operating system that affects windows, decorations, color schemes, and responsiveness of the interface in MATLAB.

Compatibility Considerations

All of the changes are cosmetic, except for file dialog boxes, like Open file. The new file dialog boxes are more conventional and easier to use than in previous versions, and there is no loss in functionality.

Invalid info.xml File on Path Now Generates an Error

When MATLAB finds an `info.xml` file on the search path or in the current directory, it assumes the file is intended to add information to the **Start** button or the Help browser, and automatically validates the file against the supported schema. If there is an invalid construct in the `info.xml` file, MATLAB displays an error in the Command Window. The error is typically of the form

```
XML-file failed validation against schema located in
...
XML-file name: full path to...\info.xml
```

and might appear when you start MATLAB, press the **Start** button, or in other situations. For more information about the error and how to eliminate it, see Add Your Own Toolboxes to the Start Button.

Compatibility Considerations

In previous versions, MATLAB displayed a warning when it encountered an invalid `info.xml` file.

Help

New features and changes introduced in Version 7.3 (R2006b) are described here.

Exact Phrase and Wildcard Searching Added; Change to Search Database

These improvements were made to the Help browser search feature. Note that they are not supported on Japanese systems.

- Search Field Always Shown — The **Search for** field is now always in view when the Help browser is open. The list of pages found appears in the **Search Results** tab.
- Exact Phrase Searches for More Relevant Results — Find an exact phrase by typing quotation marks around the search term. For example, `"plot tools"` finds only pages that include `plot tools` together, but does not find pages that include `plot` in one part of the page and `tools` in another part of the page. You can specify more than one exact phrase in a search term, such as `"plot tools" "figure pallette"`.
- Wildcards (`*`) in Search Terms for Variations of a Word (Partial Word Search) — Use the wildcard character (`*`) in place of letters or digits in your search terms. For example, `plot*` finds various forms of the word `plot`, such as `plot`, `plots`, and `plotting`, while `p*t` find those variations as well as variations of `print` and `part`, among others. You can use multiple wildcards in a word or search term.
- Boolean Operator Evaluation Order Changed — Boolean `NOT` operators in search terms are now evaluated first, followed by `OR`s, and then `AND`s. In prior versions, Boolean operators were evaluated in left to right order.

Compatibility Considerations

The search enhancements were facilitated by changing to a new type of database. As a result, any existing Help search databases for non-Mathworks products will not work in R2006b and beyond. The documentation still displays in the Help browser, but it is not included in searches.

If you use Help browser documentation for non-MathWorks products with a pre-R2006b search database, a message will display in the Help browser to notify you that the

documentation will not be included in searches. If you want search to work for that documentation, contact the product provider to request an R2006b-compatible search database.

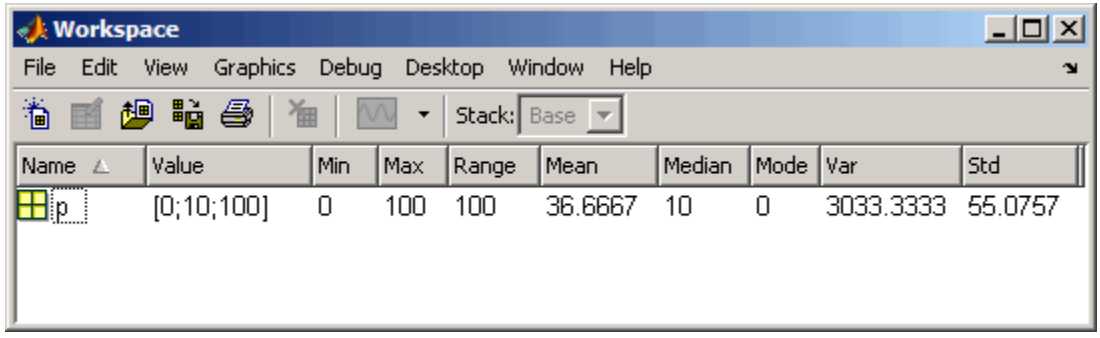
If you provide documentation for the Help browser and want the documentation to be included in the searches, you need to update the `helpsearch.db` entry in the `info.xml` file to the `helpsearch` directory, and prepare an R2006b-compatible help search database. This process is much easier than in the past. For instructions, contact Technical Support.

Workspace, Search Path, and File Operations

New features and changes introduced in Version 7.3 (R2006b) are described here.

Statistical Results in Workspace Browser

The Workspace browser includes new columns that automatically display results of common statistical calculations: Minimum, Maximum, Range, Mean, Median, Mode, Variance, and Standard Deviation. Use **View > Choose Columns** to specify the columns to show.



See also **File > Preferences > Workspace** for associated preferences.

Open Larger Arrays in Array Editor

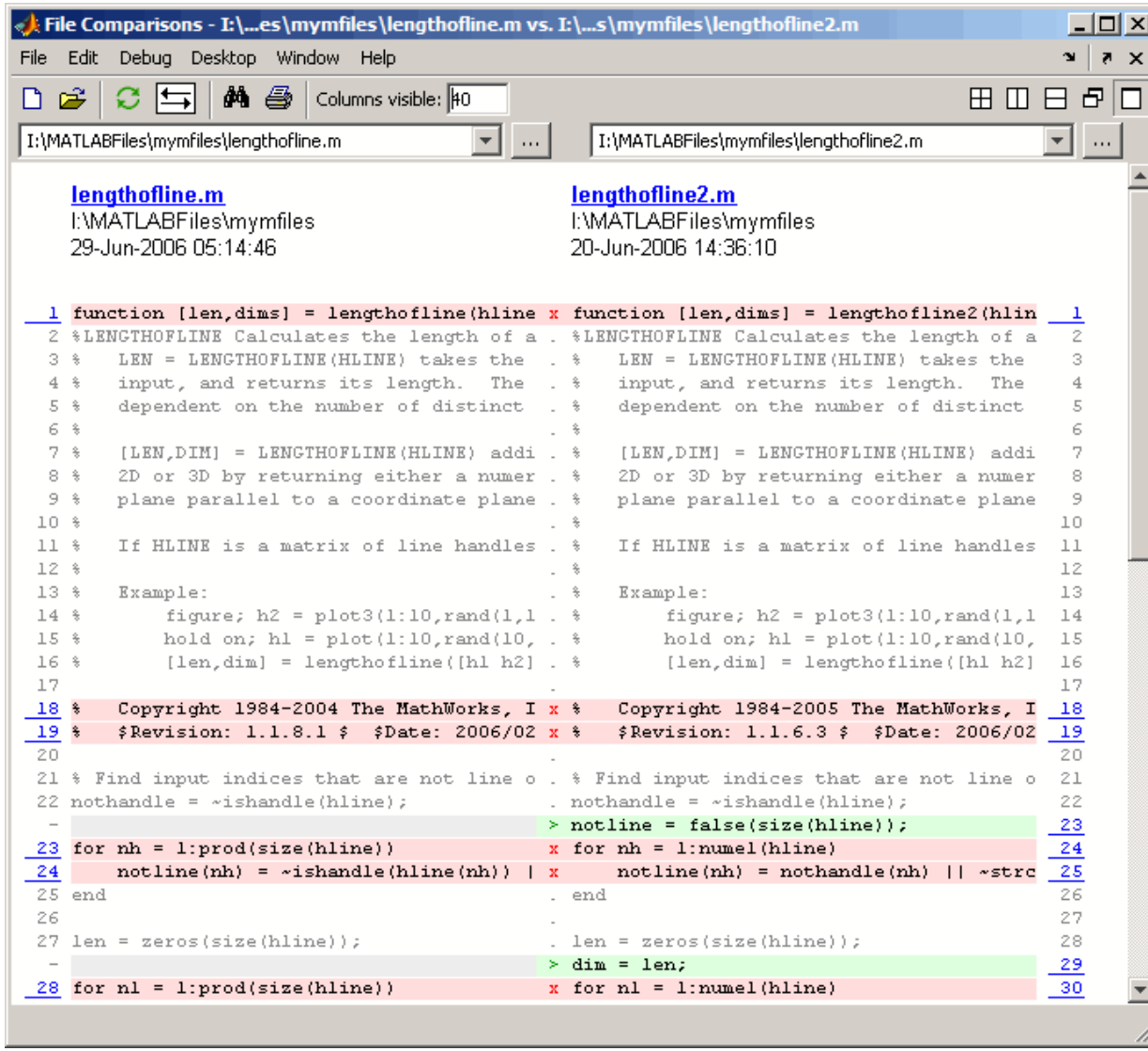
You can open larger arrays in the Array Editor. In previous versions, large arrays would fail to open in the Array Editor.

Editing and Debugging M-Files

New features and changes introduced in Version 7.3 (R2006b) are

File Comparisons Tool Added

Use the new File Comparisons tool to highlight the differences between two files. Open a file, select **Tools > Compare Against**, and then browse to select the second file or drag it into the tool from the Current Directory browser or Windows Explorer. The two files appear aligned in a window with highlights and indicators for any lines that differ, as shown in this example. For more information, see Comparing Files and Folders.



Use elements in the toolbar to exchange the left-right positions of the two files and to specify the number of columns shown.

You can also run the File Comparisons tool from the desktop by selecting **Desktop > File Comparisons**. Drag a file from the Current Directory browser or Windows Explorer into the tool. Then drag the second file into the tool.

M-Lint Enhancements for Suppressing Messages, and Showing Messages for the MATLAB Compiler Product

These are the new features and changes to M-Lint:

General Enhancements—Japanese Messages, and Line Number in Indicator Bar

- On Japanese systems, messages now appear in Japanese.
- The messages that appear at the indicator bar now includes the line number.

Suppressing M-Lint Messages

The M-Lint automatic code analyzer in the Editor/Debugger now provides a few ways for you to suppress specific messages and categories of messages:

- Ignore Specific Instance — Right-click at the underline for an M-Lint message, and from the context menu, suppress just that instance. M-Lint adds `%#ok` and a new message ID tag to the end of the line, which is the syntax in MATLAB for suppressing an M-Lint message.
- Disable All Instances — Right-click at the underline for an M-Lint message, and from the context menu, disable all instances of that message in all files. This changes the preference setting for that message.
- Use Preferences to Disable All Instances — Select **File > Preferences > M-Lint**, and disable specific messages or categories of messages, which applies to all instances in all files. You can save the settings to an M-Lint preference file. From the Editor/Debugger, you can select the M-Lint preference file from the **Tools > M-Lint** menu.

The M-Lint Code Check report also uses the preferences and suppresses the display of specific messages and categories of messages, per the preference settings.

Deployment Messages in M-Lint

M-Lint now displays deployment messages for the MATLAB Compiler product, such as `MCC does not permit the CD function`, which appear if the MATLAB Compiler

product is installed and you select the M-Lint preference to **Show MATLAB Compiler deployment messages**. You can also disable or enable all deployment messages via the Editor/Debugger **Tools > M-Lint** menu. You can suppress specific messages using the same methods as for other M-Lint messages.

Compatibility Considerations

M-Lint preferences were previously accessed via the Editor/Debugger Language settings for M. See also "mlint Message IDs Changed and %#ok Syntax Enhanced" on page 20-12

Restore Breakpoints Using New dbstop Feature

You can save the status of breakpoints to a MAT-file using `s=dbstatus` and restore the breakpoint status at a later time by loading the MAT-file using the new `dbstop(s)` option. For example, set breakpoints in `collatz`. Run `s=dbstatus` to assign the breakpoint status to `s`; use the `'completenames'` option to save absolute pathnames and breakpoint function nesting sequence. Save `s` to a MAT-file by running, for example, `save mydebugsession s`. At a later time, run `load mydebugsession` to restore `s`, and then run `dbstop(s)` to restore the breakpoints.

End Debugging Completely Using New dbquit('all') Option

If you debug multiple files at once, you can exit debug mode for all files by running `dbquit('all')`. If you debug `file1` and step into `file2`, running `dbquit` terminates debugging for both files. However, if you debug `file3` and also debug `file4`, running `dbquit` terminates debugging for `file4`, but `file3` remains in debug mode until you run `dbquit` again. The new `dbquit('all')` option ends debugging for both `file3` and `file4`, and as `dbquit` does, ends debugging for `file1` and `file2`.

Tuning and Managing M-Files

New features and changes introduced in Version 7.3 (R2006b) are described here.

File Comparison Directory Report Removed; Replaced by File Comparison Tool

The File Comparison Report, one of the directory reports available in the Current Directory browser, was removed.

Compatibility Considerations

The new File Comparisons tool replaces the File Comparison Report. The tool provides the same functionality as the report did, and adds new features. For details, see "File Comparisons Tool Added" on page 20-8.

M-Lint Code Check Report Enhancements and Changes

You can indicate specific messages or categories of messages you want M-Lint to report. For details, see "M-Lint Enhancements for Suppressing Messages, and Showing Messages for the MATLAB Compiler Product" on page 20-10.

Compatibility Considerations

In previous versions, the M-Lint Code Check report showed all messages except those suppressed via a `%#ok` at the end of a line. Now, the M-Lint Code Check report will show only the messages that are enabled via M-Lint preferences.

mlint Message IDs Changed and %#ok Syntax Enhanced

The mlint function with the `-id` option returns message IDs using a new form. For example, when you run

```
mlint('filename.m', '-id')
```

MATLAB Version 7.2 (R2006a) returns

```
L 22 (C 1-9) 2:AssignmentNotUsed : The value ...
```

whereas MATLAB Version 7.3 (R2006b) returns

```
L 22 (C 1-9): NASGU: The value ...
```

The `%#ok` syntax used in M-files to suppress the display of M-Lint messages for a line has been enhanced to support multiple messages per line. For example, given this line in an M-file,

```
data{nd} = getfield(flds,fdata{nd});
```

two M-Lint messages result:

```
34: 'data' might be growing inside a loop; ...
```

```
   consider preallocating for speed.

34: Use dynamic fieldnames with structures instead of GETFIELD...
   Type 'doc struct' for more information.
```

In MATLAB Version 7.2 (R2006a), M-Lint messages could only be suppressed for the entire line:

```
data{nd} = getfield(flds,fdata{nd}); %#ok
```

In MATLAB Version 7.3 (R2006b), you can still use `%#ok` to suppress all messages for the line, or you can add an ID tag to indicate the exact messages to suppress. For example, this suppresses only the first message about `data` growing inside a loop:

```
data{nd} = getfield(flds,fdata{nd}); %#ok<GFLD>
```

To suppress more than one message per line, separate the ID tags with a comma. For example, this suppresses both messages:

```
data{nd} = getfield(flds,fdata{nd}); %#ok<GFLD,AGROW>
```

Compatibility Considerations

In previous versions, the message IDs returned from `mlint` with the `-id` option, were of a form that included a numeric identifier, followed by the category. If you rely on the exact ID values, you will need to make modifications to use the new form.

If you use M-Lint in MATLAB Version 7.3 (R2006b) and then run those files a previous version, M-Lint in the previous version ignores the tag and IDs that follow the `%#ok`, and therefore suppresses all messages for that line, which is the expected behavior for the previous version.

mlintrpt Option Added to Use Preference Settings File

The mlintrpt function now accepts a new option that applies the preferences saved to an M-Lint settings file. It works with the `file` or `dir` syntax:

```
mlintrpt('fullpath_to_file', 'file', 'fullpath_to_configfile.txt')
mlintrpt('fullpath_to_dir', 'dir', 'fullpath_to_configfile.txt')
```

For example, create the file `NoSemiSetting.txt` by saving settings in **File** > **Preferences** > **M-Lint**. Later, use the settings via preferences or M-Lint in the Editor/Debugger, or via `mlintrpt` as in this example:

```
mlintrpt('lengthofline.m', 'file', 'I:\MATLABFiles\NoSemiSettings.txt')
```

which displays an M-Lint report in the MATLAB Web browser for the `lengthofline` file in the current directory using the M-Lint settings in `I:\MATLABFiles\NoSemiSettings.txt`.

Toolbar Refresh Button Removed

The Profiler included a refresh button  on the toolbar. This button has been removed. It performed an action similar to the **Refresh** button that appears in many of the Profiler reports, so use that button instead.

Publishing Results

notebook Setup Arguments Removed

In MATLAB 7.1 (R14SP3), the `notebook` function was enhanced to automatically detect the version of the Microsoft Word application and other required information. This feature changed the `notebook -setup` syntax—arguments to specify the version of the Microsoft Word application (`wordversion`, `wordlocation`, `templatelocation`) were no longer supported. If you used those arguments, MATLAB ignored them and issued a warning. Now in MATLAB 7.3 (R2006b), if you use those arguments, MATLAB errors.

Compatibility Considerations

If you use notebook with the `wordversion`, `wordlocation`, and `templatelocation` arguments in any of your files (for example, `startup.m`), remove them to prevent errors.

Mathematics

New Functions

Function	Description
amd	Interface to the `amd` algorithm. This interface is similar to that used in symamd, but is typically faster than `symamd`.
bvpxtend	Forms a guess structure for extending the boundary value problem solution
ldl	Full ldl factorization and solving for Hermitian matrices

max and min Now Use Magnitudes and Phase Angle for Complex Input

For complex input, min and max are computed using the magnitude, `min(abs(x))` and `max(abs(x))` respectively. In the case of equal magnitude elements, the phase angle, `min(angle(x))` and `max(angle(x))`, is now used.

Compatibility Considerations

In previous versions, `min` and `max` were computed using the magnitude, and in the case of equal magnitude elements, the first element was used. This behavior was indeterministic and not consistent with the sort function.

Code in previous releases that relied on the output of this functionality for the case described above should be updated.

Additional Output from lu

The lu function returns an additional output that helps improve numerical stability of sparse `lu` factorization.

Upper and Lower Factors for chol and ldl

Upper and lower triangular factors for chol and ldl improve performance for sparse `chol`. Referencing the lower triangle and returning the lower triangular factor is more memory efficient in the case of sparse `chol`.

Permutation Vectors with lu, luinc, ldl

Permutation vectors for lu, luinc, and ldl provide memory savings and, for large data, a noticeable performance improvement. You can now store permutation information in a single 1-by-N vector instead of an n-by-n matrix,

Two-Element Threshold for lu, spparms

A new two-element threshold for lu and spparms gives you more control over sparse `lu` and sparse \ behavior.

Lower Triangular Factors from symbfact

Extra arguments exported for symbfact allow the return of the lower triangular symbolic factor.

Support for New Versions of AMD, COLAMD, CHOLMOD, UMFPACK

MATLAB 7.3 supports new versions of the following libraries:

Library Name	Version Supported in MATLAB 7.3
AMD	2.0
COLAMD	2.5
CHOLMOD	1.1
UMFPACK	5.0

Sparse Arrays on 64-Bit Systems

The internal storage of sparse arrays on 64-bit systems has changed in the R2006b release. This change should be invisible to most MATLAB users. However, MEX-file programs that run on a 64-bit system and access sparse arrays must be modified and recompiled in order to run on MATLAB Version 7.3. This applies to existing MEX-files and to any new MEX-files that you write.

If you are affected by this change, you will need to take the following steps to make your code compatible with the new sparse array format:

- Modify all sparse array declaration statements in MEX-files that are to run on a 64–bit system so that they now use the mwSize and mwIndex types instead of more specific type declarations such as `int` or `INTEGER*4`.
- Change any MEX code that assigns a negative value to a variable used to hold a sparse array index or size. An example of this would be code that temporarily uses an array index as a flag, assigning a -1 to it to mark the index as unset. Do not try to cast negative `int` values to `mwSize` or `mwIndex`. Instead, change your code to avoid using negative values.
- Recompile these MEX-files using the `-largeArrayDims` option

For a full description of what you will need to do, please read the section "Sparse Arrays on 64-bit Systems" on page 20-35 in the MATLAB External Interfaces release notes.

Compatibility Considerations

Existing MEX-files that run on 64-bit systems and access sparse arrays in MATLAB will not operate correctly in MATLAB 7.3 unless the required changes outlined above are made and the files are recompiled with the `-largeArrayDims` option. New MEX-files that you create must also adhere to these guidelines.

FFTW Upgraded to Version 3.1.1 in MATLAB

The version of FFTW used in MATLAB has been upgraded from 3.0.2 to 3.1.1 Please read the ??? section, below.

Other changes in MATLAB FFTW support are:

- The default planner method is now `estimate`. Previously, `hybrid` was the default.
- There are new syntaxes for importing from and exporting to the library's internal single- and double-precision wisdom databases. These are as follows and are documented in the reference page for the MATLAB fftw function:
  - `str = fftw('dwisdom')`
  - `str = fftw('swisdom')`
  - `fftw('dwisdom', str)`
  - `fftw('swisdom', str)`

Compatibility Considerations

FFTW version 3.1.1 does not support wisdom produced by previous versions of the FFTW library. For this reason, FFTW wisdom that has been exported from a previous version of MATLAB cannot be imported successfully in MATLAB R2006b. Trying to import "old" wisdom results in a warning.

Future Obsolete Functions

The `betacore` function generates a warning message in R2006b but completes successfully. This function will be made obsolete in a future release.

Compatibility Considerations

You should remove all instances of the `betacore` function from your M-file program code. This function will not be supported in a future release of MATLAB.

Obsolete Functions

The following uses of MATLAB functions are obsolete as of this release. Attempts to use this functionality in the R2006a release generate a warning message but complete successfully. In R2006b, these operations fail and generate an error message.

- The following functions in their entirety: `bvpval`, `quad8`, `table1`, `table8`, `bessela`
- The `sort` function, when used with complex integer inputs
- The `gt`, `ge`, `lt`, and `le` functions, when used with complex integer inputs
- The `beta` function, when used with 3 inputs
- The `gamma` function, when used with 2 inputs
- The `opts.cheb` and `opts.stagtol` fields in the options structure of `eigs`

Compatibility Considerations

You will need to remove all instances that reference these functions at this time. These functions are no longer supported in MATLAB.

max and min No Longer Return Warning Messages for Inputs with Different Data Types

In MATLAB version 7.0 (Release 14), the functions `max` and `min` were changed to return results of a different data type than in previous releases. This behavior is described in max and min Now Have Restrictions on Inputs of Different Data Types. This change in behavior produced warning messages to assist you with diagnosing any resulting issues. In R2006b, these warning messages no longer exist.

Compatibility Considerations

The warning messages for mixed-type inputs to the functions `max` and `min` are no longer produced. Turning warning messages on will no longer display messages for this behavior, and you will no longer be able to depend on the messages for the diagnosis of problems.

Data Analysis

Generate M-File Now Supports Basic Fitting and Data Statistics

The **Generate M-File** option on the **File** menu of MATLAB figure windows now generates code that reproduces plot objects created with the Basic Fitting Tool or the Data Statistics Tool. The generated M-file function accepts new data as input and creates plot objects with the same graphics properties as those in the generating figure. In addition, the M-file recomputes fits and statistics for the new data. The code shows you how to program what you do interactively with the Basic Fitting Tool or the Data Statistics Tool.

New Options for Working with Time Series Objects

- Time Series Tools are now easier to find. From the MATLAB **Start** button, select **MATLAB > Time Series Tools**.
- Time series objects are now fully supported by the Array Editor. When you open a `timeseries` object (using open or the Workspace browser), the editor from Time Series Tools appears in the Array Editor.
- Time series objects can now be opened directly in Time Series Tools. Select a `timeseries` object in the Workspace browser, right click, and choose **Open in Time Series Tools** from the context menu.
- In Time Series Tools, you can now change subplot indices interactively. Click on a plotted line in a time series view and drag and drop it from one subplot to another. To create a new subplot, drag and drop the plotted line below the bottom axes.

Programming

Saving to MAT-Files Larger than 2 GB

With MATLAB R2006b, you can save data items that are over 2 gigabytes in size. This capability is implemented in MATLAB using a new HDF5-based format for MAT-files.

To save to a MAT-file in this format, specify the new `-v7.3` option with the `save` function:

```
save -v7.3 myfile v1 v2
```

Note: MATLAB Version 7.3 does not write MAT-files in HDF5 format by default in this release; you must explicitly specify the `-v7.3` switch.

In this release,

- The default MAT-file format employed by save is the standard MAT-file format used in the R14, R14SP, and R2006a releases. You can explicitly specify this format with the command `save -v7`.
- To write an HDF5-based MAT-file, you must use `save -v7.3`.
- HDF5-based MAT-files are *not* compressed.

In a future release,

- The default MAT-file format will be the HDF5-based format. You can explicitly specify the HDF5-based format with the command `save -v7.3`.
- To write the standard MAT-file format, you must use `save -v7`.
- As of release R2008a, HDF5-based MAT files are compressed.

Here is a summary of the version options that you can use with `save`:

save Option	Description	Available In Versions	Is the Default In Versions
`save -v4`	Save to a MAT-file you can open in MATLAB version 4	V5 and later	V4, V5
`save -v6`	Save to a MAT-file you can open in MATLAB versions 5 or 6	V7 and later	V6

save Option	Description	Available In Versions	Is the Default In Versions
`save -v7`	Save to a MAT-file you can open in MATLAB version 7	V7.3 and later	V7.3
`save -v7.3`	Save to a MAT-file that supports data items ≥2 GB	V7.3 and later	post V7.3

Compatibility Considerations

If you are running MATLAB on a 64-bit computer system and you attempt to `save` a variable that is too large for a version 7 (or earlier) MAT-file, that is, you save without using the `-v7.3` option, MATLAB skips that variable during the `save` operation and issues a warning message to that effect.

If you are running MATLAB on a 32-bit computer system and attempt to `load` a variable from a `-v7.3` MAT-file that is too large to fit in 32–bit address space, MATLAB skips that variable and issues a warning message to that effect.

Import Wizard Generates Import M-Code

The Import Wizard now includes an automated M-code generation option. Click the Generate M-code button located in the lower right of the Import Wizard window and MATLAB generates an M-file that you can use to import additional files that are similar in type to the file you are currently importing. This simplifies the process of importing multiple files into MATLAB.

New Dynamic Regular Expression Syntax

The new (?@cmd) operator specifies a MATLAB command that regexp or regexp is to run while parsing the overall match expression. Unlike the other dynamic expressions in MATLAB, this operator does not alter the contents of the expression it is used in. Instead, you can use this functionality to get MATLAB to report just what steps it's taking as it parses the contents of one of your regular expressions. This can be helpful in debugging regular expressions, for example.

New Reserved MATLAB Keywords

Two new reserved keywords have been added to the MATLAB language in this release:

- `parfor` – designates a `for` loop in the Distributing Computing Toolbox
- `classdef` – signals the beginning of a MATLAB class definition

The iskeyword function returns `true` for each of these functions, thus identifying them as reserved keywords in MATLAB:

Compatibility Considerations

MATLAB keywords are reserved for use internal to MATLAB, and should not be used in your own program code as identifiers or function names. If your code uses either of these two new keywords in this manner, you should modify your code and use words that are not reserved.

Enhancements to Display Generated by whos

The visual output of the information returned by whos looks slightly different in R2006b than in previous releases. Changes are as follows:

- There is a new column in the output called `Attributes` that identifies values that are `sparse`, `complex`, `global`, or `persistent`.
- The words “array” and “object” have been dropped from items under the `Class` heading. Items formerly listed as `double array` or `timer object` , for example, are now displayed as `double`, and `timer`.
- MATLAB no longer includes summary information showing the “Grand total” of elements and bytes at the bottom of the `whos` output display.
- There is an additional field in the structure returned by `whos`. This new field is `'persistent'` and is set to logical 1 for those variables that are persistent, or logical 0 otherwise.

Here is an example of the information displayed by whos in MATLAB 7.3:

```
whos
  Name            Size             Bytes  Class              Attributes

  vCell           5x7               2380  cell
  vComplex        6x2                192  double             complex
  vDouble         6x5x9             2160  double
  vFuncHdl        1x1                 16  function_handle
  vGlobal         0x0                  0  double             global
  vObject         1x1                248  timer
  vPersist        0x0                  0  double             persistent
  vSparse        15x15               244  double             sparse
  vStruct         1x3                804  struct
```

Compatibility Considerations

If any of your programs depend on the displayed output of `whos`, specifically in relation to the changes listed above, you might have to modify your program code. Also, if your code relies on a specific number of fields for the output structure, you should be aware that this release adds a new field: `persistent`.

unique Function Returns First and Last Indices

Using the new `first` option with the unique function returns a vector with elements that represent the *lowest* indices of unique elements in the input vector.

Given the following vector `A`

```
A = [16 7 5 41 2 16 8 2 6 3 16 6 2 5 2 5];
```

Get a sorted vector of the unique elements of `A`:

```
v = unique(A)
v =
     2     3     5     6     7     8    16    41
```

Use the `first` and `last` options to get indices of the first and last elements in `A` that make up the sorted vector `v`:

```
[v, m1] = unique(A, 'first');
m1
m1 =
     5    10     3     9     2     7     1     4

[v, m2] = unique(A, 'last');
m2
m2 =
    15    10    16    12     2     7    11     4
```

save Compression and Unicode Options Removed

In MATLAB Versions 7.0 through 7.2, you could use the switches `–compress`, `–nocompress`, `–unicode`, and `–nounicode` with the save function to enable or disable compression and Unicode character encoding in the MAT-file being generated. In MATLAB Version 7.3, the `save` function no longer supports these switches. Instead, `save` now creates a MAT-file with compression and Unicode character encoding by default, or with the following command:

```
save -v7 filename
```

You can still save to a MAT-file without using compression or Unicode encoding. In fact, you will have to do this in order to create MAT-files that you can load into a MATLAB Version 6 or earlier. To save to a MAT-file without compression or Unicode encoding, type

```
save -v6 filename
```

To disable compression and Unicode encoding for all `save` operations in a MATLAB session, open the MATLAB **Preferences** dialog, select **General** and then **MAT-Files**, and then click the button labelled **Ensure backward compatibility (-v6)**.

Compatibility Considerations

If you have code that uses any of these option switches with the `save` function, you will need to modify that code to use the `–v6` option instead.

Warning Generated by try-catch

To accommodate future changes in the MATLAB error handling capabilities, MathWorks has added a new restriction to the single-line syntax of the try-catch block. In this release, the following syntax operates as it did in previous releases, but now it also generates the following warning message:

```
try try_statements, catch catch_statements, end

Warning: This try-catch syntax will continue to work in R2006b,
but may be illegal or may mean something different in future
releases of MATLAB.
```

To make this single-line `try-catch` work without warning in R2006b, you must insert a separator character (comma, semicolon, or newline) immediately after the word catch

```
try try_statements, catch, catch_statements, end
```

As with previous releases, the recommended syntax for a `try-catch` block is as follows:

```
try
   try_statements
catch
   catch_statements
```

```
end
```

Compatibility Considerations

Your M-file programs may generate this warning if correct syntax for `try` and `catch` is not used.

Case-Sensitivity Warning Removed

The following warning has been removed from MATLAB in release R2006b:

```
"Function call foo invokes /somewhere/on/the/path/foo.m,
however, function /somewhere/ahead/on/the/path/FOO.m that
differs only in case precedes it on the path."
```

In previous versions of MATLAB, this warning message was triggered when you called a function such as `foo`, and all of the following were true:

- There was more than one MATLAB file of this name on the MATLAB path
- The names of these files differed only in letter case, and
- A MATLAB file of this name but with different case (e.g., `FOO.m`) preceded a file of matching case (e.g., `foo.m`) on the path

Earlier versions of MATLAB displayed this warning for the purpose of helping users cope with newly-introduced case sensitive dispatching changes. The warning is being removed at this time under the assumption that users are now sufficiently well-acquainted with the way MATLAB handles case sensitivity in function calls.

Compatibility Considerations

The dispatching behavior regarding to the case sensitivity is NOT changed with the removal of this warning message. If both an exact match and inexact match are present on the path, the exact match is always the one to be invoked.

fprintf(0,...) Now Throws an Error

Commands such as `fprintf(0, ...)` and `fwrite(0, ...)`, in which the file identifier is zero (the same as `stdin`) now result in an error being thrown. In previous releases, MATLAB did not throw an error in response to these commands, even though printing or writing to `stdin` is clearly not a valid option.

Compatibility Considerations

If any of your programs use lower-level MATLAB file I/O functions that send output to `stdin`, because these functions no longer ignore this type of operation, your code will now generate an error. You should modify your program code to use a file identifier other than zero.

Assigning Nonscalar Structure Array Fields to a Single Variable

In the R14 and R14 service pack releases of MATLAB, assigning a nonscalar structure array field to a single variable incorrectly resulted in an error. For example, in the following code, you should be able to assign `S.A` to one, two, three, or four output variables. However, if you assign to just a single variable, MATLAB throws an error:

```
% Create a 1-by-4 structure array S with field A.
S(1).A = 1;   S(2).A = 2;   S(3).A = 3;   S(4).A = 4;

% Assigning S(1).A and S(2).A works as expected.
[x y] = S.A
x =
     1
y =
     2

% Assigning only S(1).A should work, but does not.
x = S.A;
??? Illegal right hand side in assignment. Too many elements.
```

This has been fixed in MATLAB 7.3.

Compatibility Considerations

If any of your programs rely on this error being thrown, you will need to modify those programs.

Comma Separators Not Required in Function Declaration

As of Release 14, the function definition line in a function M-file no longer requires commas separating output variables. This now makes the function definition syntax the same as the syntax used when calling an M-file function:

```
function [A B C] = myfun(x, y)
```

Compatibility Considerations

This new syntax is not valid in MATLAB versions earlier than Release 14. When writing an M-file that you expect to run on versions both earlier and later than R14, be sure to separate any output variables in the function definition line with commas:

```
function [A, B, C] = myfun(x, y)
```

Improved Performance on Certain Platforms and Operations

In this release, MATLAB offers improved performance in the following areas:

- Improved performance on 64-bit Windows XP and Linux platforms. This is independent of the size of data set in use.
- Faster scalar indexing into cell arrays.
- Faster assignment of cell array data to variables.

Graphics and 3-D Visualization

Plotting Tools Are Now Modular Desktop Components

The three MATLAB plotting tools (Figure Palette, Property Editor, and Plot Browser) now function as desktop components like the Workspace Browser and the Array Editor. They dock, however, not to the MATLAB desktop but to a Figures window. Figures windows contain one or more figures, each of which is accessible by a tab. Turning on any plotting tool changes your figure group into a mini-desktop. You can undock, rearrange, tab, and resize the plot tools within the mini-desktop, and their state will persist across MATLAB invocations. There are just one set of plot tools for all your figures, but they only operate on figures contained in the group; undocked figures are free of the plotting tools until you redock them.

For more information see Plotting Tools – Interactive Plotting in the MATLAB Graphics documentation.

Version 6 Property Editor Has Been Removed

The MATLAB Version 6 Property Editor, one of the Plotting Tools, is no longer available. this means that the `'v6'` switch for the `propedit` function now produces an error instead of starting the version 6 property editor. The error message is

```
??? Error using ==> propedit
The Version 6 property editor is no longer available. Sorry!
```

Compatibility Considerations

If you have code that calls the version 6 property editor, you will need to modify it to use the modular plotting tools described above in “Plotting Tools Are Now Modular Desktop Components” on page 20-29. The `propedit` function remain otherwise the same.

New Desktop Printing GUI

Printing MATLAB figures has become easier as a result of combining the **Page Setup**, **Print Setup**, and **Print Preview** dialogs into one tabbed **Print Preview** dialog. You can now specify paper size, plot size and layout, color and line weight, header text,

rendering, and other printing characteristics in this new dialog. The **Page Setup** and **Print Setup** dialogs still exist, and the **Print** dialog that you call from **File —> Print** remains the same. The **Page Setup** dialog is available on all platforms.

For more information, see Using Print Preview in the MATLAB Graphics documentation and printpreview in the MATLAB function reference documentation.

Compatibility Considerations

The **Page Setup** dialog no longer is available from the figure window **File** menu. However, it does continue to exist; you can raise it using the `pagesetupdlg` command. The old **Print Preview** dialog has been removed, however. The old **Print Setup** dialog can be raised using the command

```
print -dsetup
```

When you dismiss the **Print Setup** dialog with **OK**, the settings you made with it are saved, but nothing is printed at that time.

Zoom Mode Now Supports Mouse Scroll Wheel

If your mouse has a center scroll wheel, you can use it to zoom in and out of axes, as well as by clicking and/or dragging.

You can zoom in by positioning the mouse cursor where you want the center of the plot to be and either press the mouse button or rotate the mouse scroll wheel away from you (upward). Zoom out by positioning the mouse cursor where you want the center of the plot to be and either simultaneously press **Shift** and the mouse button, or rotate the mouse scroll wheel toward you (downward). Each mouse click or scroll wheel click zooms in or out by a factor of 2.

Data Cursor Text Can Now Be Programmatically Modified

You can now easily customize the text of datatips. The `datacursormode` function lets you specify the contents and formatting of text displayed by the data cursor tool. When the data cursor tool is active, you can use its context (right-click) menu to edit or specify the text update function that MATLAB executes to display datatips. For more information, see Data Cursor — Displaying Data Values Interactively in the MATLAB Graphics documentation and datacursormode in the MATLAB Function Reference documentation.

Customizing Zoom, Pan, and Rotate3D Data Explore Modes

You can now customize the behavior of data explore modes by modifying the zoom, pan and rotate3d objects that are dereferenced as follows:

```
h = pan(figure_handle)
h = rotate3d(figure_handle)
h = zoom(figure_handle)
```

These syntaxes create *mode objects* that you can use to control the behaviors of the explore tools. Among the effects you can achieve using these explore mode objects are

- Allow, change, or inhibit a mode for a specified axes
- Create callbacks for pre- and post-buttonDown events
- Change callbacks dynamically

See pan, rotate3d, and zoom in the MATLAB Function Reference documentation for details and examples.

Compatibility Considerations

Using mode objects can cause a forward incompatibility. In prior releases, explore modes did not return an argument. Therefore, code such as the above examples that you write to take advantage of the new API will not run in MATLAB versions prior to R2006b. zoom, pan, and rotate3d code written for previous MATLAB versions, however, will run as before (there is no backward incompatibility).

Improvements to MATLAB Graphics Documentation

A new section in the Graphics User Guide, Types of MATLAB Plots, now includes a gallery of graphs that catalogs the kinds of plots that you can create using MATLAB graphics functions. There are two tables containing labeled icons, one for Two-Dimensional Plotting Functions, and one for Three-Dimensional Plotting Functions, classified by plot type. Clicking the function name above any thumbnail plot in the gallery opens the reference documentation for that function.

Reference pages for functions that create, edit, annotate, and save plots have been enhanced in several ways:

- Thumbnail figures at the top of plotting functions and related GUI reference pages illustrate what you see when you invoke these functions.

- *GUI Alternatives* sections above syntax descriptions describe how to invoke a function (or similar capability) from a GUI, and provide links to relevant topics in the Graphics and Desktop Tools User Guides.
- Revised printing documentation in user guides and reference pages (see "New Desktop Printing GUI" on page 20-29, above)
- Numerous corrections and clarifications of details in user guides and reference pages

Creating Graphical User Interfaces (GUIs)

Functions Are Now Obsolete

The following functions are obsolete in MATLAB 7.3 (R2006b): `axlimdlg`, `edtext`, `menubar`, `pagedlg`, `umtoggle`.

Compatibility Considerations

The functions shown in the following table will continue to work but their use will generate a warning message. As soon as possible, replace any occurrences you may have of these functions with the function(s) shown in the following table, if any, or with other suitable code.

Function	Replacement
`axlimdlg`	None
`edtext`	Set the `Editing` property of the `text` object.
`menubar`	Set the `MenuBar` property of the figure to `none`.
`pagedlg`	`pagesetupdlg`
`umtoggle`	Set the `Checked` property of the `uimenu` object.

Colored Buttons on Windows XP

Setting the background color of user interface control (uicontrol) push buttons and toggle buttons on Windows XP now results in flat, colored buttons.

Compatibility Considerations

Prior to this release, if you set the background color of a push button or toggle button to any color other than the factory color on Windows XP, the color displayed only as a border around the button. With this release, any such buttons will display as flat, colored buttons with a simple border.

Documentation Enhancement

The Creating GUIs Programmatically section of the documentation now contains information commensurate with information in the Creating GUIs with GUIDE section. It adds, in workflow order, information and many basic examples about:

- Adding components, menus, and toolbars to your GUI. Placing and aligning components.
- Designing for cross-platform compatibility.
- Initializing the GUI and creating callbacks.
- Callback examples for the different components.

External Interfaces/API

New Types for Declaring Size and Index Values

Version 7.3 (R2006b) defines two new types for API arguments and return values. These are

- mwSize — represents size values, such as array dimensions and number of elements.
- mwIndex — represents index values, such as indices into arrays.

Using these types in array declarations replaces more specific type declarations such as `int` or `INTEGER*4`. In general, using these types consistently in your C or Fortran source files can insulate your code from changes in the API implementation that might take place between different versions of MATLAB.

The `mwSize` and `mwIndex` types are required when working with functions that access sparse arrays on a 64-bit system. This is described in the release note on "Sparse Arrays on 64-bit Systems" on page 20-35, below.

Note: In Fortran, `mwSize` and `mwIndex` are implemented as preprocessor macros. To use these types in Fortran code, add the declaration `#include "fintrf.h"` in your source file and build your MEX-files using the Fortran preprocessor.

Sparse Arrays on 64-bit Systems

The internal storage of sparse arrays on 64-bit systems has changed in the R2006b release. Due to this change, you will need to

- Change declaration statements in your source code so that you use the new types mwSize and mwIndex in place of specific type declarations such as `int` or `INTEGER*4`. This is described in the section "New Types for Declaring Size and Index Values" on page 20-35, above.
- Recompile all MEX-files that interact with sparse arrays using the new `-largeArrayDims` switch. For more information about the `-largeArrayDims` switch, see the section "New MEX Switch" on page 20-36, below.

See the section on ??? to find out if you will need to make any modifications to your existing MEX code to accommodate these changes.

Sparse API Functions Affected By This Change

You will need to recompile any MEX-files that use the following sparse functions on a 64-bit system:

- mxGetIr
- mxGetJc
- mxSetIr
- mxSetJc

New MEX Switch

In order to build MEX-files that use any of the sparse array functions listed above, you need to compile these files with the `-largeArrayDims` switch, as shown here:

```
mex -largeArrayDims filename
```

Also, any existing MEX-files that interact with sparse arrays in MATLAB Version 7.3 must be recompiled using the `-largeArrayDims` switch.

Note: The `-largeArrayDims` option is likely to become the default in a future version of MATLAB.

Compatibility Considerations

If you are using any of the functions listed above, then you should be aware of the following potential compatibility issues if your MEX code uses sparse arrays on a 64-bit system:

- In release R2006b, you must rebuild all MEX-files that use sparse arrays using the new `-largeArrayDims` switch.
- Before building your MEX-files, change your C or Fortran sources to use the mwSize or mwIndex types introduced in this release. See the `mxArray` reference pages for the types to use for each function.
- MEX-files that compiled properly in Version 7.2 (R2006a) and do not use sparse arrays should build and execute correctly in Version 7.3 (R2006b) without changes.
- For more information on how the sparse API is affected, see the `Sparse Arrays on 64-Bit Systems` section in the MATLAB Mathematics release notes.

New MAT-File Format Based on HDF5

In Version 7.3 (R2006b), you can save MAT-files in a format based on HDF5. Unlike earlier MAT-file formats, the HDF5-based format is capable of saving variables that occupy more than 2 GB of storage, including large arrays created on 64-bit systems.

To save a MAT-file in the HDF5-based format, use the `-v7.3` option to the MATLAB save function or the `"w7.3"` mode argument to the C or Fortran matOpen function. The default MAT-file format is the same as that in Version 7.2 (R2006a).

Compatibility Considerations

Earlier versions of MATLAB cannot read MAT-files written in the HDF5-based format.

MAT-files written with MATLAB Version 7.3 (R2006b) on a 64-bit system can be read back into MATLAB 7.3 on a 32-bit system, provided that none of the values stored in the MAT-file require more than 32 bits to store.

-V5 Option to MEX to Be Removed

The `-V5` option to mex is not supported in this and future versions of MATLAB.

Compatibility Considerations

You will no longer be able to build a MEX-file that is compatible with MATLAB Version 5.

Location of mex.bat File Changed

In MATLAB Version 7.3, the Microsoft Windows script `mex.bat` is located in the directory `$MATLAB\bin`.

Compatibility Considerations

You may need to change any scripts or environment variables that relied on the previous location of `mex.bat`.

Changes to Compiler Support

Compaq Visual Fortran version 6.1 is supported in Version 7.3 (R2006b) but will not be supported in a future version of MATLAB.

Compatibility Considerations

To ensure continued support for building your Fortran programs, consider upgrading to another supported compiler. For an up-to-date list of supported compilers, see the Supported and Compatible Compilers Web page.

Actxcontrol Command Now Validates ProgID

Attempting to insert a COM server into a MATLAB figure can result in unpredictable behaviors. To prevent this condition, the `actxcontrol` command now checks the ProgID by looking at the registry's `HKR/CLSID/Control` keyword and throws an error if the ProgID does not belong to a Microsoft ActiveX control.

Compatibility Considerations

Before the validation check was added, some server ProgIDs worked with `actxcontrol` , probably because there are cases where Microsoft software points the server GUID to a control GUID underneath. An example of a ProgID that might not work with `actxcontrol` is the Windows Media® Player server ProgID `Mediaplayer.mediaplayer.1`. Therefore, depending on how Microsoft software registers the control, the following command might now return an error:

```
h = actxcontrol('Mediaplayer.mediaplayer.1'); % Returns an error
```

The correct ProgID to use for Mediaplayer is the ActiveX control ProgID:

```
h = actxcontrol('WMPlayer.OCX.7'); % Use control ProgID
```

Note that methods and properties to open and play files are different when using the control ProgID.

You can disable the validation check with the following command:

```
feature('COM_ActxProgidCheck',0)
```

Or reenable it with:

```
feature('COM_ActxProgidCheck',1)
```

By default, ProgID checking is on.

R2006a

Version: 7.2

New Features

Compatibility Considerations

Desktop Tools and Development Environment

Startup and Shutdown

New features and changes introduced in Version 7.2 (R2006a) are described here.

Installation Directory Structure on Windows Platforms

The installation directory structure on Microsoft Windows platforms is slightly different than in previous versions. By default, the structure now includes a general `MATLAB` top level directory, with a subdirectory for `R2006a`. The root directory for the MATLAB software returned by the `matlabroot` function, is now of the form in this example:

```
D:\Applications\MATLAB\R2006a
```

In previous versions, the top-level directory included the version number, so the root directory for MATLAB, as returned by the `matlabroot` function, was of the form in this example:

```
D:\Applications\MATLAB 7.1
```

Compatibility Considerations

If you relied on the explicit root directory structure for MATLAB in your code, change it to reflect the new structure including the top-level `MATLAB` directory. The `matlabroot` function might be useful.

Error Log Reporting

If MATLAB experiences a segmentation violation, it generates an error log. Upon the next startup, MATLAB prompts you to e-mail the error log to The MathWorks. The MathWorks uses the log to work on resolving the problem. When you send a log, you receive a confirmation e-mail and will only receive further e-mails if The MathWorks develops a fix or workaround for the problem.

There are some situations where the Error Log Reporter does not open, for example, when you start MATLAB with a `-r` option or run in deployed mode. If you experience segmentation violations but do not see the Error Log Reporter on subsequent startups, you can instead e-mail logs by following the instructions at the end of the segmentation violation message in the Command Window.

JVM Software Updated for 64-Bit Linux Platforms

The Sun Microsystems JVM software version that MATLAB uses is now version 1.5.0_04 for 64-bit platforms running the Linux operating system from Linus Torvalds.

Desktop

New features and changes introduced in Version 7.2 (R2006a) are described here.

Preferences Reorganized and New Keyboard Pane Added to Support Command Window and Editor/Debugger

Preferences includes a new pane, **Keyboard**, for setting key bindings, tab completion, and delimiter-matching preferences for the Command Window and Editor/Debugger. Most of these preferences were previously located in the preference panes for the Command Window or Editor/Debugger.

Compatibility Considerations

You no longer access keyboard and indenting preferences for the Command Window and Editor/Debugger from the component preferences, but rather from the new **Keyboard** preferences. In addition, some preferences that were set separately for these components are now shared. For details about the changes, see “Keyboard and Indenting Command Window Preferences Reorganized” on page 21-4, and “Keyboard and Indenting Editor/Debugger Preferences Reorganized” on page 21-6.

Open All Desktop Tools from Desktop Menu

You can now open (and close) all desktop tools from the **Desktop** menu. In previous versions, you could not access document-based tools from the **Desktop** menu. The document-based desktop tools are: Editor/Debugger, Figures, Array Editor, and Web Browser.

Access Login Renamed to MathWorks Account

Use **Help > Web Resources > MathWorks Account** menu items to go to your MathWorks Account if you are registered, or to register online. MathWorks Account was previously called Access Login.

Running Functions — Command Window and Command History

New features and changes introduced in Version 7.2 (R2006a) are described here.

Keyboard and Indenting Command Window Preferences Reorganized

The Command Window Keyboard and Indenting preferences pane was removed. The tab size preference is now on the Command Window preferences pane. The tab completion, keybinding, and parentheses matching preferences were moved to the new Keyboard preferences pane. The parentheses-matching preferences are now called delimiter-matching preferences and are shared with the Editor/Debugger.

Help

New features and changes introduced in Version 7.2 (R2006a) are described here.

help for Model Files

You can now use the `help` function to get the complete description for MDL-files. For example, run

```
help f14_dap.mdl
```

MATLAB displays the description of the F-14 Digital Autopilot High Angle of Attack Mode model in the Simulink software, as defined in its **Model Properties > Description**:

```
Multirate digital pitch loop control for F-14 control design
demonstration.
```

Workspace, Search Path, and File Operations

New features and changes introduced in Version 7.2 (R2006a) are described here.

toolboxdir function added

The toolboxdir function returns the absolute pathname to the specified toolbox. It is particularly useful with the MATLAB Compiler product because the toolbox root directory is different than in MATLAB.

Visual Directory View Removed

The Visual Directory view was removed from the Current Directory browser. Most of the features it provided are accessible from the Current Directory browser standard view.

Editing and Debugging M-Files

New features and changes introduced in Version 7.2 (R2006a) are

Tab Completion — Tab Now Completes Function and Variable Names

You can now use tab completion in the Editor/Debugger to complete function names and variable names that are in the current workspace. When you type the first few characters of a function or variable name and press the **Tab** key, the Editor/Debugger displays a list of all function and variable names that begin with those letters, from which you choose one.

It operates essentially the same way as the existing tab completion feature in the Command Window, with the exception that Editor/Debugger tab completion does not support completion of file and path names.

To enable tab completion in the Editor/Debugger, select **File > Preferences > Keyboard**, and then under **Tab completion**, select **Tab key narrows completions**. By default, the preference is selected.

With tab completion enabled in the Editor/Debugger, you can still include tab spacing, for example, to include a comment at the end of a line. To add tab spacing, include a space after the last character you type and then press **Tab**. Instead of showing possible completions, the Editor/Debugger moves the cursor to the right where you can continue typing.

Compatibility Considerations

If you press the **Tab** key to add spacing within your statements, you might instead get a completion for a function or see a list of possible completions. For example, if the preference for tab completion is on and you want to create this statement

```
if a=mate     %test input value
```

where you press **Tab** after `mate` to achieve the spacing, the following happens instead

```
if a=material
```

This is because the tab completion preference completes `mate`, automatically supplying the `material` function.

To achieve the spacing with **Tab** (as in previous versions), either add a space after `mate` and then press **Tab**, or turn off the preference Tab key narrows completions in Keyboard Preferences.

Go Menu Added; Bookmark and Go To Items Moved from Edit Menu to Go Menu

- To set, clear, and navigate to bookmarks, use the menu items in the new **Go** menu, which were previously located in the **Edit** menu.
- The **Go To** feature for navigating to line numbers, functions in M-files, and cells has moved to the new **Go** menu. It was previously located in the **Edit** menu.

Compatibility Considerations

Use the new **Go** menu items instead of **Edit > Bookmark** features and **Edit > Go To**.

Navigate Back and Forward in Files

Use **Go > Back** (and **Go > Forward**) to go to lines you previously edited or navigated to in a file, in the sequence you accessed them. The main benefit of this feature is going directly to lines of interest. As an alternative to the menu items, use the Back and Forward buttons on the toolbar.

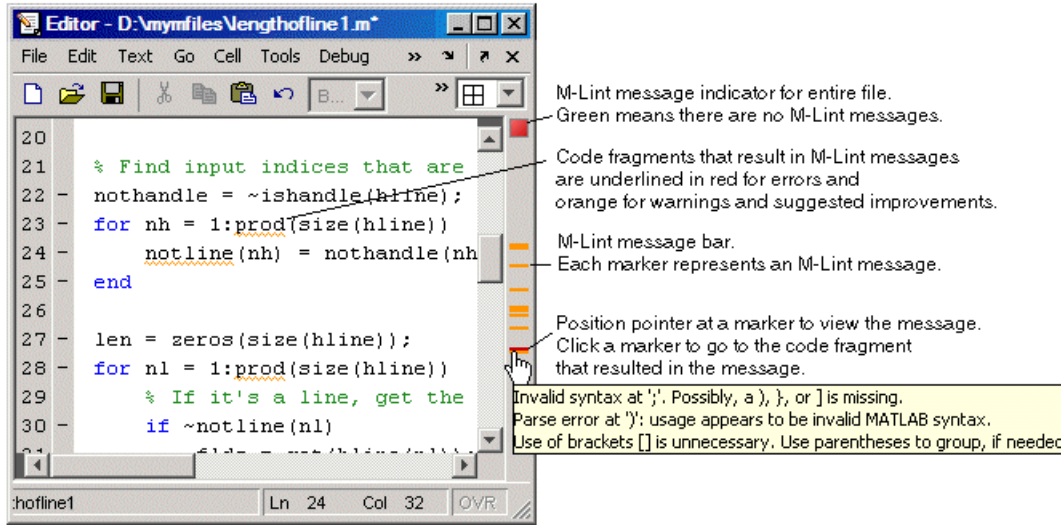
Keyboard and Indenting Editor/Debugger Preferences Reorganized

The Editor/Debugger **Keyboard and Indenting** preferences pane was renamed to **Tab** preferences, and keybinding and parentheses-matching preferences were moved to the new **Keyboard** preferences pane. The parentheses-matching preferences are now called delimiter-matching preferences and are shared with the Command Window.

M-Lint Automatic Code Analyzer Checks for Problems and Suggests Improvements

The M-Lint code analyzer, now built into the Editor/Debugger, continuously checks your code for problems and recommends modifications to maximize performance and maintainability. It performs the same analysis as the existing M-Lint Code Check report, but also provides these features:

- Indicates the problem lines and associated M-Lint messages directly in the M-file rather than in a separate report.
- Identifies (underlines) code fragments within a line that result in M-Lint messages.
- Distinguishes messages that report errors (red) from warnings and suggestions (orange).
- Continually analyzes and updates messages as your work so you can see the effect of your changes without having to save the M-file or rerun an M-Lint report.



To use or turn off M-Lint in the Editor/Debugger, select **File** > **Preferences** > **Editor/Debugger** > **Language**, and for **Language**, select M. Under **Syntax**, select **Enable M-Lint messages**, or clear the check box to turn it off. Use the associated drop-down list to specify the types of code fragments that you want M-Lint to underline, for example, **Underline warnings and errors**.

Debugging Changes

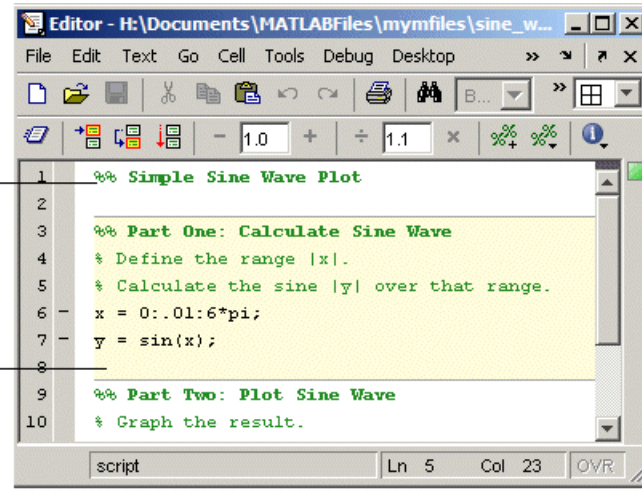
- The `dbstop` function now allows you to stop at, (not in), a non M-file, allowing you to view code and variables near it in your M-file. For example, if you want to stop at the point in your M-file `myfile.m` where the built-in `clear` function is called, run `dbstop in clear; mymfile`. Use this feature with caution because the debugger stops in M-files it uses for running and debugging if they contain the non M-file, and then some debugging features do not operate as expected, such as typing `help functionname` at the `K>>` prompt.

Cell Mode On by Default — Shows Cell Toolbar and Possibly Horizontal Lines and Yellow Highlighting; Cell Information Bar and Button Added

Cell mode, a useful feature in the Editor/Debugger for publishing results and rapid code iteration, is now enabled by default. An M-file cell is denoted by a `%%` at the start of a line. Any M-file that contains `%%` at the start of a line is interpreted as including cells. The Editor/Debugger reflects the cell toolbar state and the cell display preferences, such as yellow highlighting of the current cell and gray horizontal lines between cells.

For quick access to information about using cells in M-files, use the new information  button on the cell toolbar.

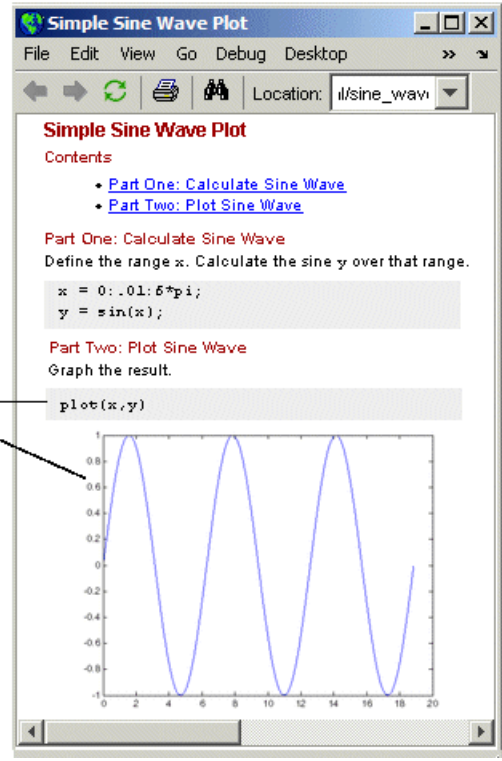
%% at the start of a line denotes a cell, used for publishing or rapid code iteration. The current cell is highlighted in yellow.



Example of M-file with cells published to HTML.

Includes source code and output.

Supported formats are: HTML, Word, PowerPoint, LaTeX, and XML.



If you do not want cell mode enabled, select **Cell > Disable Cell Mode**.

MATLAB remembers the cell mode between sessions. If cell mode is disabled when you quit MATLAB, it will be disabled the next time you start MATLAB, and the converse is true.

In MATLAB Version 7.2, the first time you open an M-file in the Editor/Debugger, the cell toolbar appears. If the M-file contains a line beginning with %%, an information bar appears below the cell toolbar, providing links for details about cell mode. To dismiss the information bar, click the close box on the right side of the bar. To hide the cell toolbar, right-click the toolbar and select **Cell Toolbar** from the context menu.

Compatibility Considerations

In previous versions, cell mode was off by default. The cell toolbar and yellow highlighting or horizontal rules in M-files that contain %% at the start of a line might be unexpected. If you used the %% symbols at the start of a line in M-files for a purpose other than denoting M-file cells, consider replacing the %% symbols with a different indicator, or keep cell mode disabled.

Lines Between Cells

You can set an Editor/Debugger display preference, **Show lines between cells**, to add a faint gray rule above each cell in an M-file. The line does not print or appear in the published M-file.

Cell Titles in Bold Preference Removed

Previous versions included an Editor/Debugger display preference to **Show bold cell titles**. When cleared, cell titles appeared in plain text, rather than bold text. This is no longer a preference you can set — all cell titles now appear in bold text.

Tuning and Managing M-Files

New features and changes introduced in Version 7.2 (R2006a) are

M-Lint and mlint Enhancements and Changes

The M-Lint code analyzer is now built into the Editor/Debugger where it continuously checks your code for problems and recommends modifications to maximize performance and maintainability. For details, see "M-Lint Automatic Code Analyzer Checks for Problems and Suggests Improvements" on page 21-7.

Compatibility Considerations

The mlint function has changed slightly to support its use in the Editor/Debugger. Specifically, the results returned from `mlint` with the `-id` option are of a different form than for previous versions. If you rely on the exact values, you need to make modifications.

For example, this is the form of a message returned in R2006a: `L 22 (C 1-9) 2:AssignmentNotUsed : The value assigned here to variable 'nothandle' might never be used.`

This is the form of the message from R14SP3: `22 (C 1-9) InefficientUsage:AssignmentNotUsed : The value assigned here to variable 'nothandle' might never be used.`

There is now a numeric identifier, followed by the category, for example: `2:AssignmentNotUsed`

If you do rely on the exact values, note that there have been very few changes to the message text itself. For example, both R14SP3 and R2006a use the same text: `The value assigned here to variable 'nothandle' might never be used.`

Because of improvements being made to `mlint`, the values returned using the `-id` option are expected to change in the next version as well, particularly the numeric identifier and category form. Do not rely on the exact values returned using `mlint` with the `-id` option or you will probably need to make modifications.

Profiling Enhancements

nohistory Option Added to profile Function

Use the new `profile -nohistory` option after having previously set the `-history` option to disable further recording of history (exact sequence of function calls). All other profiling statistics continue to accumulate.

Accuracy Improved

The Profiler provides more accurate accounting. The total time you see with the Profiler GUI now matches total wall clock time from when you started profiling until you stopped profiling. Overhead associated with the Profiler itself is now applied evenly.

Statistics for Recursive Functions

The `profile` function now gathers and reports time for recursive functions in the `FunctionTable`'s `TotalTime` for the function. In previous versions, `profile` attempted to break out `TotalRecursiveTime`, which was not always accounted for accurately. The value for `TotalRecursiveTime` in `FunctionTable` is no longer used.

This change is also reflected in the Profiler GUI reports.

PartialData Reported in Results, AcceleratorMessages Removed

The `FunctionTable` now includes the `PartialData` value. A value of `1` indicates the function was modified during profiling, for example, by being edited or cleared, so data was only collected up until the point it was modified.

In previous versions, `FunctionTable` included `AcceleratorMessages` although it was not used. `AcceleratorMessages` is no longer included.

Visual Directory View Removed from Current Directory Browser

The Visual Directory view was removed from the Current Directory browser.

Compatibility Considerations

Most of the features it provided are accessible from the Current Directory browser standard view.

Publishing Results

New features and changes introduced in Version 7.2 (R2006a) are described here.

Insert Italic Text Markup

You can now make designated text comments in cells appear italicized in the published output. Use **Cell > Insert Text Markup > Italic Text**, or use the equivalent markup symbols, underscores, as in `_SAMPLE ITALIC TEXT_`.

publish Function has New catchError Option

The publish function has a new `catchError` option that allows you to continue or stop publishing if the M-file contains an error.

Source Control Interface

New features and changes introduced in Version 7.2 (R2006a) are described here.

PVCS Source Control System Name Change

The PVCS® source control system (from Merant) now has a new name, ChangeMan® (from Serena®), and the source control interface in MATLAB on UNIX platforms reflects this change.

If you use the ChangeMan software on UNIX platforms, the cmopts value returned for it is `pvcs`. If you use PVCS software, select `ChangeMan` in the Source Control Preferences pane.

Compatibility Considerations

PVCS software users on UNIX platforms formerly selected `PVCS` in the Source Control Preferences pane. Now, PVCS software users select `ChangeMan` instead.

Mathematics

New Library CHOLMOD for Sparse Cholesky Factorization

For sparse matrices, MATLAB now uses CHOLMOD version 1.0 to compute the Cholesky factor. CHOLMOD is a set of routines offering improved performance in factorizing sparse symmetric positive definite matrices. See the function reference pages for chol, spparms, and mldivide for more information on how CHOLMOD is used by MATLAB.

New Solver for State-Dependent DDEs

In this release, MATLAB provides a second solver function, ddesd, in addition to the existing dde23 function, for delay differential equations (DDEs). This new solver is for use on equations that have *general* delays. You supply a function in the input argument list that returns the vector of delays to be used by the solver. See the function reference page for `ddesd`, and Delay Differential Equations in the MATLAB Mathematics documentation for more information.

Upgrade to BLAS Libraries

MATLAB now uses new versions of the Basic Linear Algebra Subroutine (BLAS) libraries. For Intel processors on Windows and Linux platforms, MATLAB supports the Math Kernel Library (MKL) version 8.0.1. For AMD processors on Linux platforms, MATLAB uses the AMD Core Math Library (ACML) version 2.7.

New Function for Integer Division

The new idivide function provides division similar to `A./B` on integers except that fractional quotients are rounded to integers according to a specified rounding mode.

New Input to gallery Function

The gallery function has a new, optional input argument called `classname`. The `classname` input is a quoted string that must be either `'single'` or `'double'`. When you specify a `classname` argument in the call to `gallery`, MATLAB produces a matrix of that class.

Improved Algorithm for expm

The expm function now uses an improved algorithm to compute a matrix exponential. This algorithm often requires fewer matrix multiplications.

More Efficient condest for Sparse Matrices

The condest function handles sparse matrices more efficiently when estimating a 1-norm condition number.

accumarray Accepts Cell Vector Input

The accumarray function now accepts a cell vector as the `subs` input. This vector can have one or more elements, each element a vector of positive integers. All the vectors must have the same length. In this case, `subs` is treated as if the vectors formed columns of an index matrix.

Data Analysis

Data Analysis Collection Revised and Expanded

In this release, the MATLAB Data Analysis collection has been thoroughly revised to improve content organization and flow. In addition, most examples have been updated and streamlined.

Reference Pages for timeseries and tscollection Objects

Detailed reference pages are now available for timeseries and tscollection objects, properties, and methods. You can access these reference pages in the Help contents, under Data Analysis in the MATLAB Function Reference.

Text Files Can Be Imported In Time Series Tools

In Time Series Tools, you can now use the Import Wizard to import data from text files, such as `.csv`, `.dat`, and `.txt`.

Linux 64 Platform Fully Enabled for Time Series Tools

Time Series Tools is now fully enabled on the Linux 64 platform.

Compatibility Considerations

On the Linux 64 platform, you no longer need to manually enable the Time Series Tools feature before starting Time Series Tools (as in MATLAB 7.1).

Programming

Larger Data Sets with 64-Bit Windows XP

MATLAB support for Windows XP 64-bit edition enables you to handle much larger data sets. To learn more about memory allocation for arrays, see Memory Allocation.

Using avifile and movie2avi on Windows XP 64

Note You must change the compression setting if you use the avifile or movie2avi function on Windows XP 64.

MATLAB currently defaults to using Indeo codecs to compress video frames when using `avifile/addframe` or `movie2avi`. If you attempt to use `avifile` and addframe, or `movie2avi` on a Windows XP 64-bit platform without specifying the compression type, an error message appears indicating the codec was not found. Nondefault settings must be explicitly passed in when using these functions on Windows XP 64 because Microsoft does not provide Indeo codecs on this platform.

This issue does not affect 32-bit Windows XP installations.

Compatibility Considerations

To work around this issue, do the following:

1. Explicitly specify no compression when creating the `avifile` object or when calling `movie2avi`. Two examples of this are

   ```
   aviobj = avifile('myvideo.avi', 'compression', 'none');
   movie2avi(mov, 'myvideo.avi', 'compression', 'none');
   ```

2. Specify a codec for a compression that is installed. The ones that are included with Windows XP 64 are

   - IYUV — Intel YUV codec (c:\winnt\system32\iyuv_32.dll)
   - MRLE — Microsoft RLE codec (c:\winnt\system32\msrle32.dll)
   - MSVC — Microsoft Video 1 codec (c:\winnt\system32\msvidc32.dll)

For example, to use the Intel YUV codec, use the four-CC code:

```
aviobj = avifile('myvideo.avi', 'compression', 'IYUV');
```

Other codecs can be found at `http://fourcc.org`.

Note there are restrictions with some codecs. For example, some codecs can only be used with grayscale images.

Regular Expressions

MATLAB 7.2 introduces the following new features for regular expressions in MATLAB. For more information on these features, see Regular Expressions in the MATLAB Programming documentation.

New Features

- Dynamic regular expressions — You can now insert MATLAB expressions or commands into regular expressions or replacement strings. The dynamic part of the expression is then evaluated at runtime.
- Generating literals in expressions — Use the new `regexptranslate` function when you want any of the MATLAB regular expression functions to interpret a string containing metacharacters or wildcard characters literally.
- New parsing modes — Four matching modes (case-sensitive, single-line, multiline, and freespacing) extend the parsing capabilities of the MATLAB regular expression functions.
- Warnings display — Use the new `'warnings'` option with the regular expression functions to enable the display of warnings that are otherwise hidden.

Compatibility Considerations

Calling regexp or regexpi with the `'tokenExtents'` and `'once'` options specified now returns a `double` array instead of a cell array. You may need to change your code to accommodate the new return type.

Setting Environment Variables

Use the new setenv function to set the value of an environment variable belonging to the underlying operating system.

issorted Support for Cell Arrays

You can now use the issorted function on a cell array of strings.

XLS Functions Support More Formats

xlsread now supports Excel files having formats other than XLS (e.g., HTML) as long as the COM server is available. Also, xlsfinfo now returns this file format information.

Archiving Functions Accept Files on Path and ~/

Files specified as arguments to gzip, gunzip, tar, and zip can now be specified as partial path names. On UNIX machines, directories can start with `~/` or `~username/`, which expands to the current user's home directory or the specified user's home directory, respectively. The wildcard character `*` can be used when specifying files or directories, except when relying on the MATLAB path to resolve a filename or partial pathname.

sendmail No Longer Requires ASCII Messages

E-mail messages that you send using sendmail are no longer restricted to ASCII character encoding schemes.

MATLAB Warns on Invalid Input to str2func

Due to a bug introduced in MATLAB R14, the str2func function failed to issue a warning or error when called with an invalid function name or a function name that includes a path specification. In the R2006a release, `str2func` now generates a warning under these conditions. In a future version of MATLAB, `str2func` will generate an error under these conditions.

Compatibility Considerations

Any existing code that calls `str2func` with an invalid function name or a function name that includes the path now generates a warning message from MATLAB. In a future version, this will cause an error. You should note any such warnings when using R2006a, and fix the input strings to `str2func` so that they specify a valid function name.

Changes to Character Encoding in File I/O

The fopen function has a new optional argument, a string that specifies a name or alias for the character encoding scheme associated with the file. If this argument is omitted or is the empty string (''), the MATLAB default encoding scheme is used. Given a file identifier as the only argument, `fopen` now returns an additional output value, a string that identifies the character encoding scheme associated with the file.

Low-level file I/O functions that read data from files, including fread, fscanf, fgetl, and fgets, read characters using the encoding scheme associated with the file during the call to `fopen`. Low-level file I/O functions that write data, including fwrite and fprintf, write characters using the encoding scheme associated with the file during the call to `fopen`.

Support for character encoding schemes has these limitations:

- Surrogate pairs are not supported. Each surrogate pair is read as a replacement character, the equivalent of `char(26)`.
- Stateful character encoding schemes are not supported.
- Byte order marks are not interpreted in any special way. Your code must skip them if necessary.
- Scanning numbers, using fscanf, is supported only for character encoding schemes that are supersets of ASCII. (Most popular character encoding schemes, with the exception of UTF-16, are such supersets.)

Compatibility Considerations

In V7.1 (R14SP3), low-level file I/O functions that read and write data treated characters as unsigned bytes. Programs using such functions as fread may have called native2unicode to convert input to MATLAB characters using a particular encoding scheme. Programs using such functions as `fwrite` may have called unicode2native to convert output from MATLAB characters using a particular encoding scheme.

For example, on a Japanese Windows platform, where the default character encoding scheme is Shift-JIS, a program may have used `native2unicode` and `unicode2native` to read and write Japanese text in this way:

```
fid = fopen(file);
data = fread(fid, '*char')';
fclose(fid);
dataU = native2unicode(data);
```

```
% operate on data
outData = unicode2native(dataU);
fid = fopen(file, 'w');
fwrite(fid, outData, 'char');
fclose(fid);
```

Such a program would produce different and possibly incorrect results in V7.2 (R2006a). The calls to `native2unicode` and `unicode2native` are no longer necessary, because the `fread` and `fwrite` functions now convert data to and from MATLAB characters using the character encoding scheme specified in the calls to `fopen`. In V7.2 (R2006a), the example code can be simplified to produce correct results:

```
fid = fopen(file);
dataU = fread(fid, '*char')';
fclose(fid);
% operate on data
fid = fopen(file, 'w');
fwrite(fid, dataU, 'char');
fclose(fid);
```

Changes to code using fread, fgets, fgetl, and fscanf

If your code calls `native2unicode` to convert input to MATLAB characters using a specified (or default) encoding scheme, you can, but do not have to, remove the calls to `native2unicode`.

This applies to reading from an encoded file using any of the following:

- `fread` with `precision` set to `'*char'` or `'char=>char'`
- `fgets` or `fgetl`
- `fscanf` with the `format` specifier set to either `'%s'` or `'%c'`

When you remove a call to `native2unicode`, be sure that the call to `fopen` supplies the same encoding argument (if any) as the call to `native2unicode`.

You may have used code like these examples in V7.1 (R14SP3):

```
indata = native2unicode(fread(fid, '*char')');
indata = native2unicode(fread(fid, 'char=>char')');
indata = native2unicode(fgets(fid));
indata = native2unicode(fgetl(fid));
indata = native2unicode(fscanf(fid, '%s'));
indata = native2unicode(fscanf(fid, '%c'));
```

You can, but do not have to, remove the calls to `native2unicode` in V7.2 (R2006a):

```
indata = fread(fid, '*char')';
indata = fread(fid, 'char=>char')';
indata = fgets(fid);
indata = fgetl(fid);
indata = fscanf(fid, '%s');
indata = fscanf(fid, '%c');
```

Changes to code using fwrite

If your code calls `unicode2native` to convert output from MATLAB characters using a specified (or default) encoding scheme, in most cases you must remove the calls to `unicode2native`. This is especially important if the encoding represents multibyte characters.

This applies to writing an encoded file using `fwrite` with `precision` set to either `'char'` or `'char*1'`.

When you remove a call to `unicode2native`, be sure that the call to `fopen` supplies the same encoding argument (if any) as the call to `unicode2native`.

You may have used code like these examples in V7.1 (R14SP3):

```
fwrite(fid, unicode2native(outbuff), 'char');
fwrite(fid, unicode2native(outbuff), 'char*1');
```

You must remove the calls to `unicode2native` in V7.2 (R2006a):

```
fwrite(fid, outbuff, 'char');
fwrite(fid, outbuff, 'char*1');
```

Graphics and 3-D Visualization

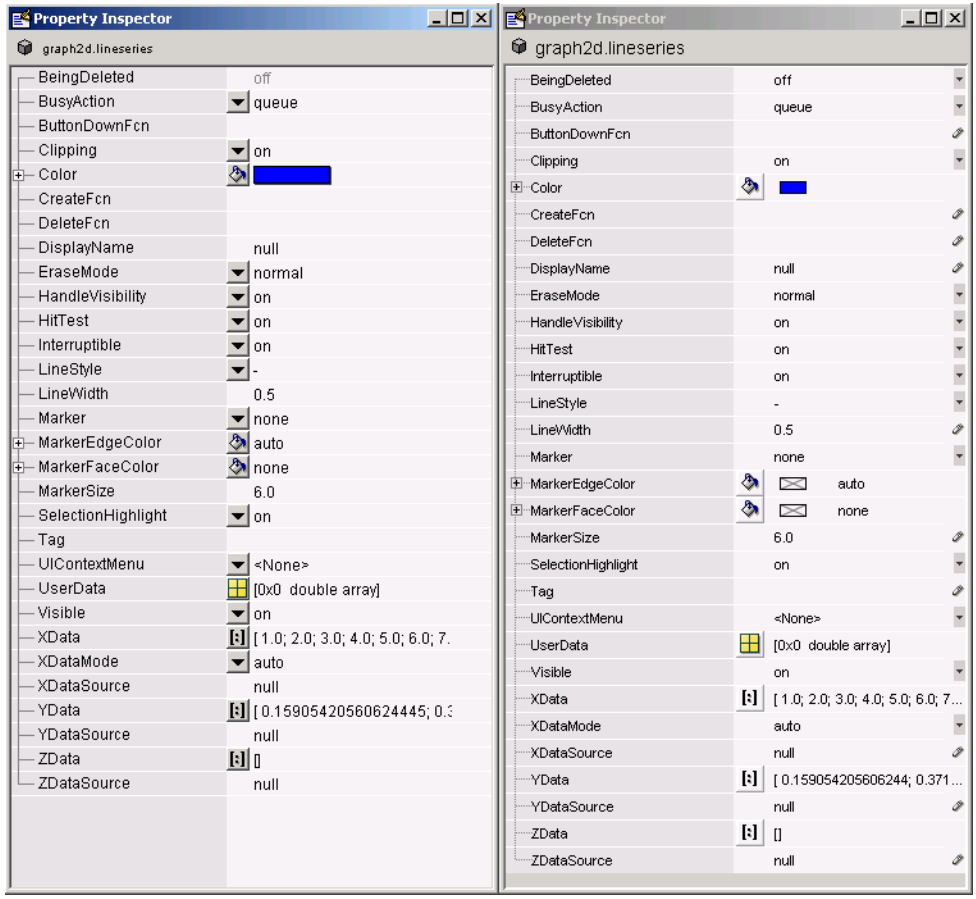
Pasting Cut or Copied Graphic Objects Can Create an Axes

The way in which MATLAB handles copying (or cutting) and pasting children of axes such as lineseries, barseries, or contourgroup objects has changed slightly. In previous releases, if no destination axes was selected prior to pasting one or more such objects, they would be pasted into the current axes (returned by the gca function). MATLAB no longer makes this assumption; if no axes is currently selected when you paste graphic objects, a new axes is created in the destination figure to contain them.

To avoid creating a new axes where you do not want to do so, you should be in plot edit mode and have selected a destination axes before using **Edit -> Paste** or typing **CTRL-V**. You can use the Plot Browser to conveniently select objects to copy and to paste into.

Inspector Has New Look

The Property Inspector (the GUI summoned by the MATLAB `inspect` command) has a new look, but no changed functionality. The inspector enables you to view and change the most commonly used object properties. The figure below compares the previous version (7.1, left) of the Property Inspector with the new version (7.2, right):



Note that in addition to having a smaller font and wider line spacing, the new inspector locates pop-up menus at the right margin instead of between the two columns. Also, some icons have been redesigned.

Creating Graphical User Interfaces (GUIs)

Treatment of & in Menu Label Is Changed

The use of `'&'` (ampersand) in the uimenu `'Label'` property string is changed for cases that use the constructs `'A& B'` and `'A&&B'`. The changes bring these constructs in line with the way `'&'` is used in other `'Label'` constructs. See "Compatibility Considerations" below for specific information.

Compatibility Considerations

Interpretation of `'Label'` property strings that use the following constructs is changed:

- The string `'A& B'` now produces the menu label **A& B** with no underlined mnemonic. Previously, `'A& B'` produced the label **A_B**, in which the space is a mnemonic.
- The string `'A&&B'` now produces the menu label **A & B** with no underlined mnemonic. Previously, `'A&&B'` produced the label **A&B** with no mnemonic.

If you use either construct, `'A& B'` or `'A&&B'`, in your menu labels, verify that the new resulting label is acceptable or change the `'Label'` property to a new string.

Major Documentation Revision

The MATLAB document Creating Graphical User Interfaces is reorganized and rewritten. It now consists of three sections:

- Getting Started—Leads you through the steps needed to create a simple GUI, both programmatically and using GUIDE.
- Creating GUIs with GUIDE—Contains the information, previously included in Creating Graphical User Interfaces, that you need to create a GUI using GUIDE. This section is organized in workflow order with many small examples of the various steps. A final chapter provides advanced examples.
- Creating GUIs Programmatically—For now, this section contains a summary of the available functions and complete code examples for three GUIs.

  One GUI uses a variety of user interface controls to enable a user to calculate the mass of an object after specifying the object's density and volume.

Two other GUIs work together as an icon editor. One GUI, a color palette, is embedded in the other GUI, an icon editor. The color palette passes data to the icon editor whenever the GUI user selects a new color.

External Interfaces/API

MEX-Files Built with gcc on Linux Must Be Rebuilt

In MATLAB V7.2 (R2006a) on Linux and Linux x86-64 platforms, MEX-files built with `gcc` must be recompiled and relinked using `gcc` version 3.4 or later. Rebuilding is required because MATLAB V7.2 (R2006a) on Linux and Linux x86-64 platforms is built with `gcc` version 3.4.

Compatibility Considerations

Changes in `gcc` version 3.4 have caused incompatibilities between MATLAB V7.2 (R2006a) and MEX-files built with `gcc` versions earlier than 3.4.

On Linux and Linux x86-64 platforms, MEX-files built with `gcc` versions earlier than 3.4 cannot be used in MATLAB V7.2 (R2006a).

On Linux and Linux x86-64 platforms, MEX-files built with `gcc` version 3.4 or later cannot be used in versions of MATLAB earlier than V7.2 (R2006a).

MEX-Files in MATLAB for Microsoft Windows x64

With the introduction of MATLAB for Windows x64, you can now build 64-bit MEX-files. These MEX-files have the extension `.mexw64`. The mexext command returns `mexw64` in MATLAB for Windows x64.

Compatibility Considerations

MEX-files built using MATLAB for Windows (32-bit), which have `.mexw32` extensions by default, cannot be used in MATLAB for Windows x64.

By default, when MATLAB for Windows x64 is installed, the `mex.pl` and `mex.bat` scripts build MEX-files for a Windows x64 platform (with `.mexw64` extensions).

New Microsoft and Intel Compilers Supported

MATLAB V7.2 (R2006a) supports new compilers for building MEX-files on Windows and Windows x64 platforms:

- Microsoft Visual C++ 2005, also informally called Visual C++ 8.0, part of Microsoft Visual Studio 2005
- Intel Visual Fortran 9.0

Environment Variables Needed for Intel Visual Fortran

When you build a MEX-file or an Engine or MAT application using Intel Visual Fortran 9.0, MATLAB requires an environment variable to be defined, depending on whether you are building in MATLAB for Windows (32-bit) or MATLAB for Windows x64:

- MATLAB for Windows (32-bit): The environment variable `VS71COMNTOOLS` must be defined. The value of this environment variable is the path to the `Common7\Tools` directory of the Visual Studio .NET 2002 or 2003 installation directory. (Intel Visual Fortran requires Visual Studio .NET 2002 or 2003 on 32-bit Windows platforms.) This environment variable is commonly defined by the Visual Studio .NET 2003 installation program.
- MATLAB for Windows x64: The environment variable `MSSdk` must be defined. The value of this environment variable is the path to the installation directory for Microsoft Platform SDK for Windows Server 2003. (Intel Visual Fortran requires Microsoft Platform SDK for Windows Server 2003 on Windows x64 platforms.) This environment variable is *not* commonly defined by the Microsoft Platform SDK installation program.

MWPOINTER Macro for Platform-Independent Fortran Code

MATLAB provides a preprocessor macro, mwPointer , that declares the appropriate Fortran type representing a pointer to an `mxArray` or to other data that is not of a native Fortran type, such as memory allocated by mxMalloc. On 32-bit platforms, the Fortran type that represents a pointer is `INTEGER*4`; on 64-bit platforms, it is `INTEGER*8`. The Fortran preprocessor translates `MWPOINTER` to the Fortran declaration that is appropriate for the platform on which you compile your file.

Compaq Visual Fortran Engine and MAT Options File Renamed

MATLAB V7.1 (R14SP3) included a Windows Engine and MAT options file named `df66engmatopts.bat`. This file contained options for Compaq Visual Fortran version 6.6 for use in building Fortran engine or MAT stand-alone programs. The file name `df66engmatopts.bat` originated with an earlier version of the Fortran compiler, named Digital Fortran.

In V7.2 (R2006a), this file has been renamed `cvf66engmatopts.bat` to match the Compaq Visual Fortran product name.

Compatibility Considerations

You may need to change any scripts that depend on the earlier name for the options file.

Options Files Removed for Unsupported Compilers

MATLAB V7.1 (R14SP3) included MEX, Engine, and MAT options files for a number of Windows C and Fortran compilers that were untested. These options files are not included in V7.2 (R2006a). The unsupported compilers, and the supported compilers that replace them, are:

Unsupported Compiler	Supported Replacement
Borland 5.0, 5.2, 5.3, 5.4	Borland 5.5, Borland 5.5 Free, Borland 5.6
Digital Visual Fortran 5.0, 6.0	Compaq Visual Fortran 6.1, Compaq Visual Fortran 6.6, Intel Visual Fortran 9.0
Microsoft Visual C++ 5.0, Visual C++ .NET 2002 (7.0)	Microsoft Visual C++ 6.0, Visual C++ .NET 2003 (7.1), Visual C++ 2005 (8.0)
Watcom 10.6, 11	Open Watcom 1.3

Compatibility Considerations

If you were using an untested compiler with a previous version of MATLAB, replace it with a supported compiler. You may need to recompile your MEX-files or applications.

Obsolete Functions No Longer Documented

In V7.1 (R14SP3), many MAT-file access, MX array manipulation, MEX-files, and MATLAB engine functions were declared obsolete in the External Interfaces Reference documentation. These functions are no longer documented in V7.2 (R2006a).

This section lists the obsolete functions removed from the documentation, along with replacement functions, if any.

Obsolete Functions: MAT-File Access

Obsolete Function	Replacement
`matDeleteArray (C and Fortran)`	matDeleteVariable
`matDeleteMatrix (C and Fortran)`	matDeleteVariable
`matGetArray (C and Fortran)`	matGetVariable
`matGetArrayHeader (C and Fortran)`	matGetVariableInfo
`matGetFull (C and Fortran)`	matGetVariable followed by mxGetM, mxGetN, mxGetPr, mxGetPi
`matGetMatrix (C and Fortran)`	matGetVariable
`matGetNextArray (C and Fortran)`	matGetNextVariable
`matGetNextArrayHeader (C and Fortran)`	matGetNextVariableInfo
`matGetNextMatrix (C and Fortran)`	matGetNextVariable
`matGetString (C and Fortran)`	matGetVariable followed by mxGetString
`matPutArray (C and Fortran)`	matPutVariable
`matPutArrayAsGlobal (C and Fortran)`	matPutVariableAsGlobal
`matPutFull (C and Fortran)`	mxCreateDoubleMatrix followed by mxSetPr, mxSetPi, matPutVariable
`matPutMatrix (C and Fortran)`	matPutVariable
`matPutString (C and Fortran)`	mxCreateString followed by matPutVariable

Obsolete Functions: MX Array Manipulation

Obsolete Function	Replacement
`mxClearLogical (C and Fortran)`	None
`mxCreateFull (C and Fortran)`	mxCreateDoubleMatrix
`mxCreateScalarDouble (C and Fortran)`	mxCreateDoubleScalar
`mxFreeMatrix (C and Fortran)`	mxDestroyArray
`mxGetName (C and Fortran)`	None
`mxIsFull (C and Fortran)`	mxIsSparse
`mxIsString (C and Fortran)`	mxIsChar
`mxSetLogical (C and Fortran)`	None
`mxSetName (C and Fortran)`	None

Obsolete Functions: MEX-Files

Obsolete Function	Replacement
`mexAddFlops (C)`	None
`mexGetArray (C and Fortran)`	mexGetVariable
`mexGetArrayPtr (C and Fortran)`	mexGetVariablePtr
`mexGetEps (C and Fortran)`	mxGetEps
`mexGetFull (C and Fortran)`	mexGetVariable followed by mxGetM, mxGetN, mxGetPr, mxGetPi

Obsolete Function	Replacement
`mexGetGlobal (C and Fortran)`	mexGetVariablePtr
`mexGetInf (C and Fortran)`	mxGetInf
`mexGetMatrix (C and Fortran)`	mexGetVariable
`mexGetMatrixPtr (C and Fortran)`	mexGetVariablePtr
`mexGetNaN (C and Fortran)`	mxGetNaN
`mexIsFinite (C and Fortran)`	mxIsFinite
`mexIsInf (C and Fortran)`	mxIsInf
`mexIsNaN (C and Fortran)`	mxIsNaN
`mexPutArray (C and Fortran)`	mexPutVariable
`mexPutFull (C and Fortran)`	mxCreateDoubleMatrix followed by mxSetPr, mxSetPi, mexPutVariable
`mexPutMatrix (C and Fortran)`	mexPutVariable

Obsolete Functions: MATLAB Engine

Obsolete Function	Replacement
`engGetArray (C and Fortran)`	engGetVariable
`engGetFull (C and Fortran)`	engGetVariable followed by mxGetM, mxGetN, mxGetPr, mxGetPi
`engGetMatrix (C and Fortran)`	engGetVariable
`engPutArray (C and Fortran)`	engPutVariable

Obsolete Function	Replacement
`engPutFull (C and Fortran)`	mxCreateDoubleMatrix followed by mxSetPr, mxSetPi, engPutVariable
`engPutMatrix (C and Fortran)`	engPutVariable
`engSetEvalCallback (C)`	None
`engSetEvalTimeout (C)`	None
`engWinInit (C)`	None

Compatibility Considerations

Most of the functions listed as obsolete in this section are unsupported in V6.5 (R13) and later versions. Some obsolete functions are unsupported in earlier versions.

If this section lists a replacement for an obsolete function, change any code that refers to the obsolete function to use the replacement instead.

If you must use an obsolete function in a MEX-file or application, use the `-V5` option to mex when you build the file.

Support for Licensed ActiveX Controls

MATLAB supports the use of Microsoft ActiveX controls that require licensing at both design time and runtime.

See the actxcontrol function for information on how to specify a design-time license key.

See Deploying ActiveX Controls Requiring Run-Time Licenses for information on how to use ActiveX Controls that require runtime licenses in your MATLAB application.

Support for VT_Date Type

MATLAB defines a data type to be used with controls requiring input defined as type VT_DATE. See Using Date Data Type for more information.

Dynamic Linking of External Libraries

MATLAB supports dynamic linking of external libraries only on 32-bit Windows systems and 32-bit Linux systems. See Calling C Shared Library Functions from MATLAB for more information.

R14SP3

Version: 7.1

New Features

Compatibility Considerations

Desktop Tools and Development Environment

Startup and Shutdown

New features and changes introduced in this version are described here.

Startup Option, –nodesktop, on Windows Platforms No Longer has Menu Bar and Toolbar; Use Function Equivalents Instead

The behavior of MATLAB software when started on Microsoft Windows platformswith the `-nodesktop` option has changed. The MATLAB Command Window no longer displays a menu bar or toolbar. This change resolves a number of problems that occurred in previous versions when running MATLAB in `-nodesktop` mode on Windows platforms.

Compatibility Considerations

Use equivalent functions instead of the menu and toolbar.

Instead of using the **File > Preferences** menu to modify the font or colors used in the Command Window, run `preferences -nodesktop`. For more information, see “preferences Function Now Supports -nodesktop Option” on page 22-5.

Desktop

New features and changes introduced in this version are organized by these topics:

Arranging Windows and Documents

Figure Windows Now Dockable on Macintosh Platforms

On Apple Macintosh platforms, figure windows are now dockable.

Resize Multiple Tools at Once

You can now position the pointer at the intersection of three or four tools or documents to resize all of them at once.

Resize and Move Desktop Tools Using the Keyboard

There are now menu items you can select to move and resize the active tool in the desktop. Use the menu item mnemonics to perform those action with the keyboard. For example, if the Command Window is in the desktop along with other tools, press **Ctrl +0** (or click in it) to make the Command Window the active tool. Then press **Alt+D**, **V**, which is the mnemonic equivalent for selecting **Desktop > Move Command Window**. The pointer becomes an arrow. Use the arrow keys to move an outline of the Command Window to a new dockable location. Press **Enter** to dock it there, or press **Esc** to return the Command Window to its original position.

Resize Names in the Document Bar

You can now adjust the width of a name in the document bar when the bar is at the top or bottom of the window.

Positioning Document Bar Menu Item Name Changed

In previous versions, selecting **Desktop > Document Bar** displayed only menu items for positioning the document bar. Now, there are additional menu items. The same change was made to the context menu for the document bar. To access the menu items for positioning the document bar, select **Desktop > Document Bar > Bar Position**.

Keyboard Access Added for Document Bar Options

The **Desktop > Document Bar** now includes these items: **Alphabetize**, **Width**, and **Move documentname On Bar**. With their inclusion in the menu, you can use the keyboard to access these features via mnemonics. For example, on Windows platforms, press **Alt+D**, **M**, **A** as a shortcut to for **Desktop > Document Bar > Alphabetize**.

Left/Right and Top/Bottom Split for Document Arrangements Name Changed

When arranging documents in desktop tools, you can choose **Window > Left/Right Split** or **Window > Top/Bottom Split** to show two documents at once in the tool. Those menu items are now called **Left/Right Tile** and **Top/Bottom Tile**. This change was made to avoid any confusion with the Editor/Debugger's new split screen feature.

Preferences Directory Added for R14SP3; Supplements R14 Directory

There is a new preferences directory, `R14SP3`. This is the directory name returned when you run the `prefdir` function. When you install R14SP3, MATLAB migrates files from

your existing preference directory, `R14`, to the new directory, `R14SP3`. Changes made to files in the directory when you run R14SP3 are not used when you run previous R14 releases.

This represents a change in the preference directory MATLAB uses for a minor release, and was done to prevent serious backwards compatibility problems. It is primarily relevant if you use R14SP3 and previous R14 releases. If you only run R14SP3, or run R14SP3 with R13 or R12 releases, you will not be affected by this change.

In the past, minor releases and the associated major release used the same preferences directory. For example, R13 and R13SP1 shared the `R13` preferences directory. That continues to be true for all previous releases, but is not true for R14SP3 and beyond. The `R14` preferences directory will be shared by the R14 through R14SP2 releases, but the new `R14SP3` preferences directory is only used by R14SP3. This means that changes made to files in the directory while running R14SP3 re not used when you run a previous R14 releases, and the reverse is true. For example, statements added to the Command History when you run R14SP3 are not in the Command History when you run R14SP2.

For more information, see the reference page for prefdir.

Compatibility Considerations

This change was made to prevent major backwards compatibility problems. Use the `R14SP3` preferences directory instead of the `R14` directory. If you use the `prefdir` function and have code that relies on the result being `R14`, you will need to modify that code.

Preferences Changes for Fonts, Hyperlinks, and -nodesktop

Font Antialiasing Preference Added

In **Preferences** > **Fonts**, select the new antialiasing preference to provide a smoother appearance to desktop fonts.

Hyperlink Color Preference Changed

There is a new **Colors** preference for specifying the color of hyperlinks in the Command Window and the Help browser **Index** pane. In previous releases, this preference only applied to the Command Window hyperlinks and was accessed via Command Window preferences.

preferences Function Now Supports -nodesktop Option

Run `preferences -nodesktop` after starting MATLAB on Windows platforms with the `-nodesktop` option to change Command Window font and colors via a special **Preferences** dialog box.

To set other available preferences for the Command Window after starting MATLAB with the `-nodesktop` option, run `preferences` and use the resulting **Preferences** dialog box for all tools and products. Note that changes you make to font and color preferences in this dialog box do not apply to the Command Window.

info.xml File Automatic Validation; Shows Warnings for Invalid Constructs

If you add your own toolbox to the **Start** button, you can use the schema file for its `info.xml` file, *`matlabroot`*`/sys/namespace/info/v1/info.xsd`. MATLAB now automatically validates your `info.xml` file against this schema when you click the **Start** button after updating and refreshing your `info.xml` file.

Compatibility Considerations

If your `info.xml` contains invalid constructs, you will see warnings in the Command Window until you correct the problems.

Other Desktop Changes

Paste Special Menu Item Renamed

In the **Edit** menu, the name of the **Paste Special** item has been replaced by **Paste to Workspace**, but the functionality remains the same. It opens the Import Wizard so you can paste the clipboard contents to the workspace in MATLAB.

Rename Shortcut Categories

You can now rename shortcut categories.

Running Functions — Command Window and Command History

New features and changes introduced in this version are

- "Tab Completion Preference Added" on page 22-6
- "Tab Completion No Longer Shows Entries Twice" on page 22-6
- "Incremental Search Now Supports Removing Characters" on page 22-6

Tab Completion Preference Added

There is a new Command Window preference, **Tab key narrows completion**. When selected, with a list of possible completions in view, type another character and press **Tab** to further narrow the list shown. Repeat to continue narrowing the list. This behavior is similar to tab completion behavior in releases prior to R14.

Tab Completion No Longer Shows Entries Twice

In previous versions, when completing filenames or function names, a name sometimes appeared twice in the completion list, once with the file extension and once without. Now the entry appears only once.

Incremental Search Now Supports Removing Characters

In incremental search, use **Ctrl+G** to remove characters back to the previous successful string of characters found. For example, when searching for the term `plode`, the text is not found and `Failing` appears in the incremental search field. **Ctrl+G** automatically removes the `de` from the search term because `plo` does exist in the file.

Hyperlink Color Preference Moved

The preference for specifying the hyperlink color has moved from the Command Window preferences pane to the **Colors** preferences pane. The hyperlink color now also applies to links in the Help browser **Index** pane.

Compatibility Considerations

Use the **Colors** preference pane to specify the hyperlink color, and be aware that it also impacts the Help browser Index pane color.

Help

New features and changes introduced in this version are

Hyperlink Color in the Index Pane Preference Added

You can now specify the color for links in the Help browser **Index** pane using the **Colors** preferences pane. The hyperlink color also applies to links in the Command Window, so changes you make to the preference apply to both tools.

New Look for Demos, Including Thumbnails and Categories

Stylistic changes were made to the Demos interface in the Help browser. On the summary page for a product, each demo appears with a thumbnail image that provides an indication of the type of output it creates, as well as an icon representing the type of demo (M-file, M-GUI, model, or video).

Demos Run in Command Window as Scripts and Their Variables Now Created in Base Workspace

In this release, all M-file demos include the **Run in the Command Window** link, which executes the demo via `echodemo`.

In previous releases, some M-file demos provided a **Run** hyperlink in the display pane. When you clicked **Run**, the M-file demo executed in a GUI via the `playshow` function. An example of this type of demo is the MATLAB Mathematics Basic Matrix Operations demo, `intro.m`. In this release, the **Run** hyperlink for these M-file demos has been replaced by **Run in the Command Window**. It executes the demo step by step in the Command Window via the `echodemo` function. Double-clicking this type of M-file demo in the Navigator pane no longer runs the M-file demo, but opens the M-file in the Editor/Debugger where you can run it step by step using **Cell > Evaluate Current Cell and Advance**.

Compatibility Considerations

The new **Run in Command Window** hyperlink represent a change in the way demos run.

The `echodemo` function MATLAB uses to run M-file demos in the Command Window runs the demos as scripts. The `playshow` function MATLAB used to run M-file demos in

previous releases ran the demos as a function. This means that now the demo's variables are created in the base workspace. If you have variables in the base workspace when you run an M-file demo, and the demo uses an identical variable name, there could problems with variable name conflicts. For example, your variable could be overwritten. The demo's variables remain in the base workspace after the demo finishes running until you clear them or quit MATLAB. Another change is that figures are not automatically closed when you end the demo.

echodemo Function Added to Replace playshow function

There is a new `echodemo` function that replaces `playshow`. The Demos browser uses `echodemo` to execute M-file demos when you click the **Run in the Command Window** link.

Compatibility Considerations

The `playshow` function is deprecated in favor of the `echodemo` function. In a future release, the `playshow` function will be removed. In practice, both `echodemo` and `playshow` are helper functions for running demos. It is unlikely you would ever call either `playshow` or `echodemo` directly, and especially not in M-files.

Add Demos to Favorites

You now can add published M-file demos to favorites.

Adding Your Own Demos Type Tag Now Supported

If you add demos for your own toolbox, you can use the new `<type>` tag for a `<demoitem>` to identify the type of demo in your toolbox's `demos.xml` file.

Bug Reporting System Introduced

You now can view bugs fixed with this release, as well as any known bugs using the Bug Reports database in the Support section of the MathWorks Web site. The MathWorks continuously updates the database to add any newly found bugs and compatibility issues, as well as any new workarounds and solutions. The system includes bugs found and fixed in R14SP2 and later releases.

Workspace, Search Path, and File Operations

New features and changes introduced in this version are described here.

Find Files Offers Additional Filtering

The Find Files tool has been enhanced. It now allows you to search all file types except those specified. It also lets you ignore files larger than a specified size. Along with enhancements to the Find Files tool, some minor feature changes were made, including the removal of the **Restore Defaults** button.

Visual Directory View to be Removed

In the next release, the Current Directory browser will no longer support the Visual Directory view (accessed using the  toolbar button).

Compatibility Considerations

Some features currently available using the Visual Directory view will not be available in the next release when the feature is removed.

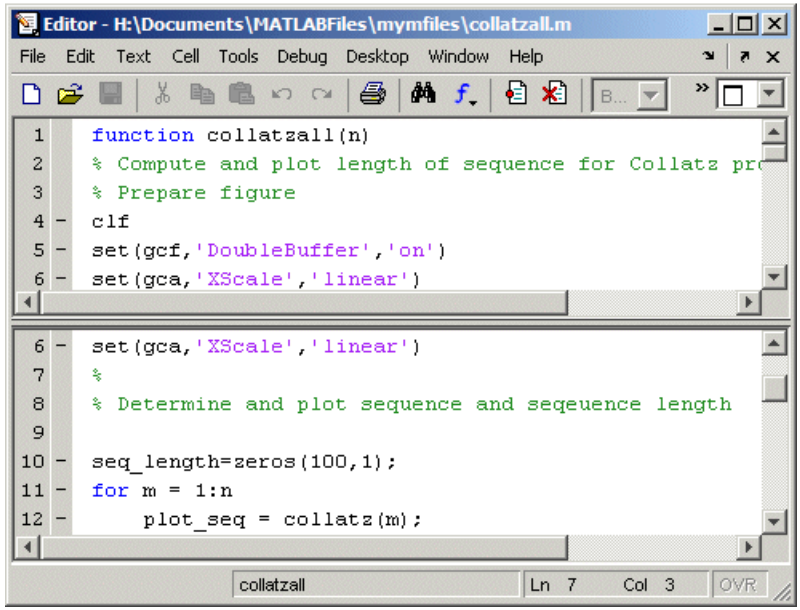
Editing and Debugging M-Files

New features and changes introduced in this version are

Split Screen Display Added

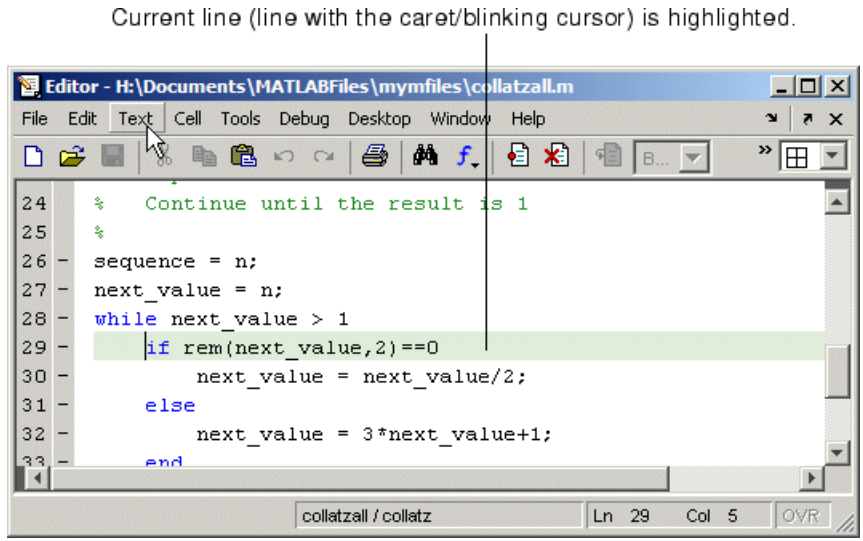
The Editor/Debugger now supports a horizontal or vertical split screen for displaying two different parts of the same document at once. To split the screen, select **Window > Split Screen** and the splitting action you want, for example, **Top/Bottom**. Alternatively, drag the splitter bar that appears above the vertical scroll bar or to the left of the horizontal scroll bar. To remove the splitter, drag it to an edge of the window.

Document with top/bottom split.



Highlight Current Line Added

You can set a preference to highlight the current line, that is, the line with the caret (also called the blinking cursor). This is useful, for example, to help you see where copied text will be inserted when you paste. To highlight the current line, select **Preferences > Editor/Debugger > Display** and under **General Display Options**, select the check box for **Show caret row highlighting**. You can also specify the color used to highlight the line.



Comment Lines in Java, C, or C++ Program Files Now Supported

You can now use the **Text > Comment** feature to comment selected lines in Sun Microsystems Java, ANSI® C, and C++ program files. This adds the // symbols at the start of the selected lines. Similarly, **Text > Uncomment** removes the // symbols from the front of selected lines in Java, C, and C++ program files.

HTML File Indenting Feature Added as the Default

There is a new Editor/Debugger language preference for HTML files to specify block indenting. By default, the preference is selected so block indenting applies when typing text in HTML files.

In addition, you now can select **Text > Smart Indent** to apply smart indenting to selected text in HTML files.

Compatibility Considerations

When typing text in HTML files, you will automatically see block indenting because the preference is selected by default.

Incremental Search Now Supports Removing Characters

In incremental search, use **Ctrl+G** to remove characters back to the previous successful string of characters found. For example, when searching for the term `plode`, the text is not found and `Failing` appears in the incremental search field. **Ctrl+G** automatically removes the `de` from the search term because `plo` does exist in the file.

Emacs Key Binding for Select All

With the Emacs key bindings preference selected, use **Ctrl+X**, **H** to select all.

Change Case Added to Menu

Use new items in the **Text** menu to change the case of selected text. You can also use the keyboard equivalents for changing case that existed in previous versions—these are shown in the menu next to each item.

Nested Function Name No Longer in Status Bar

The Editor/Debugger no longer displays the current nested function name in the status bar. Look in the M-file to view the current nested function name.

Tuning and Managing M-Files

New features and changes introduced in this version are described here.

Directory Reports Uses New Run Buttons

With Directory Reports displayed in the Web browser, you can use these two new buttons:

- **Rerun This Report** — This updates the currently displayed report after you have made changes to the report options or to any files in the current directory.
- **Run Report on Current Directory** — Use this after changing the current directory to run the same type of report for the new current directory.

These new buttons replace the **Refresh** button.

Override %#ok with the New mlint -notok Option

There is a new option for the `mlint` function, `'-notok'` you can use to override any statements that include `%#ok` (the symbol you add to the end of a line instructing `mlint`

to ignore the line). That is, `mlint` will run for all lines in the file and will not ignore any statements.

Hyperlink Now Part of Messages Displayed by mlint

When you run the `mlint` function, the line number in the messages displayed is a hyperlink that when clicked, opens the file in the Editor/Debugger scrolled to that line number.

Profiler Button Added to Toolbar

There is now a button  on the MATLAB desktop toolbar to open the Profiler.

Publishing Results

New features and changes introduced in this version are described here.

Notebook Setup Changes; Some Arguments Removed

The `notebook` function setup behavior and syntax have changed.

When you run `notebook('-setup')`, MATLAB automatically obtains all the information about your Microsoft Word application from the system registry for yourWindows environment and you are no longer prompted to supply the information.

In previous versions, when you configured Notebook, you ran

```
notebook ('-setup')
```

Notebook then prompted you to specify the version of Word you were using, and if needed, the location of Word and its template directory. You could supply the information using optional arguments to the `notebook` function:

```
notebook('-setup', wordversion, wordlocation, templatelocation)
```

Now, when you run `notebook('-setup')`, MATLAB automatically obtains all the Word information from the registry for yourWindows environment.

Compatibility Considerations

If you use `notebook` with the `wordversion`, `wordlocation`, and `templatelocation` arguments in any of your files (for example, `startup.m`), remove those arguments in

your files. If you specify the optional arguments, the `notebook` function runs and issues a warning, but ignores the values. In a future release, MATLAB will issue an error when it encounters `notebook` with these arguments.

Versions of Microsoft Word Application Supported by Notebook; Microsoft Word 97 No Longer Supported

MATLAB Notebook supports the Microsoft Word version 2000 application. Notebook also supports the Microsoft Word 2002 application and Microsoft Word 2003 application, both for the Microsoft Windows XP platform.

Compatibility Considerations

As of MATLAB 7.1 (R14SP3), Notebook no longer supports the Microsoft Word 97 application.

Mathematics

New Functions

The following functions are new in R14SP3:

Function	Description
hypot	Square root of sum of squares
mode	Finds most frequent values in sample

Compatibility Considerations

A new function name can potentially introduce a backward incompatibility since it can, under certain circumstances, override a variable with the same name as the new function. This is especially true for names that are commonly used as variable names in program code.

An example of such a function name is the mode function, introduced in this release. If you have M-file programs that use `mode` as a variable name, it is possible under certain conditions for MATLAB to interpret these variable names as function names by mistake. Read the section Variable Names in the MATLAB Programming documentation to find out how to avoid having these variables misinterpreted.

If your program code uses a user-written function named `mode`, you may find that MATLAB calls the new MATLAB mode function instead of your own `mode` function. To correct this, modify your MATLAB path by placing the location of your own `mode` function closer to the beginning of the path string than the location of the MATLAB `mode.m` file. The help for the addpath and rmpath functions explains how to modify your MATLAB path.

Modified Functions

The following functions have been modified in MATLAB 7.1:

Function	Modified Behavior
accumarray	Allows more flexibility for input/output classes and functions to be called

Function	Modified Behavior
odeset	New `NonNegative` integration property to impose nonnegativity constraints on an ODE solution
rand	Supports the Mersenne Twister algorithm in generating random numbers
svd	Returns economy decomposition

Changes to accumarray

MATLAB Version 7.1 adds the following new features to the accumarray function:

- The data type for the `val` input can be any numeric type, or logical, or character.
- The data type for the `subs` input can be any numeric type.
- You can use a cell array of separate index vectors for the `subs` input.
- When you specify a `function` input argument, the value returned by `accumarray` is given the same class as the values returned by that function.
- You can control the sparsity of the value returned by `accumarray` by specifying the new input argument `issparse`.

Imposing Nonnegativity Constraints on Computed ODE Solution

There is a new integration property called `NonNegative` that you can use when applying ODE initial value problem solvers. If you need to solve a problem in which certain components of the solution must be nonnegative, use the `NonNegative` property to impose nonnegativity constraints on the computed solutions.

See Nonnegative Solutions under Calculus in the MATLAB Mathematics documentation for more information on this feature.

Mersenne Twister Support in rand

The rand function now supports a method of random number generation called the Mersenne Twister. The algorithm used by this method, developed by Nishimura and Matsumoto, generates double precision values in the closed interval `[2^(-53), 1-2^(-53)]`, with a period of `(2^19937-1)/2`.

For a full description of the Mersenne twister algorithm, see `http://www.math.sci.hiroshima-u.ac.jp/~m-mat/MT/emt.html`.

svd Returns Economy Decomposition

The following feature was released in MATLAB 7.0, but was undocumented until this release.

The command `svd(A,'econ')` returns economy decomposition on matrices having few rows and many columns as well as those with many rows and few columns. `svd(A,0)` continues to behave as it always has, namely to only return economy-sized decomposition on matrices having many rows and few columns.

Note that this does not carry over to the qr function as there is no valid way of cutting out any of the information returned by `qr` to make an economy-sized decomposition of matrices having few rows and many columns.

New Location for LAPACK Libraries

The location of the LAPACK libraries has been changed. These libraries are now located in

```
extern/lib/win32/microsoft/libdflapack.lib
extern/lib/win32/microsoft/libmwlapack.lib
```

This change impacts you only if you build MEX-files that call LAPACK and BLAS functions.

Documentation on Data Analysis

The section of the MATLAB Mathematics documentation on “Data Analysis and Statistics” has been moved to a new MATLAB Data Analysis book. This book documents MATLAB functions and tools that support basic data analysis, including plotting, descriptive statistics, correlation, interpolation, filtering, and Fourier analysis. It also documents the new object-oriented command-line API for analyzing time-series data.

Data Analysis

Data Analysis Documentation

The MATLAB 7.1 documentation includes a new Data Analysis book that describes how to use MATLAB functions and tools for common data-analysis tasks:

- Plotting
- Filtering
- Interpolation
- Descriptive statistics
- Correlation
- Data fitting using linear regression
- Fourier analysis
- Time-series analysis

Some of the content in Data Analysis is incorporated from the Mathematics and Graphics books, such as data plotting, descriptive statistics, data fitting, and Fourier analysis. All information about time-series analysis is new.

Time-Series Analysis

You can analyze time-series data using the new timeseries and tscollection objects and methods, as well as the Time Series Tools graphical user interface. This new functionality supports the following:

- Representation for univariate or multivariate MATLAB time series and Simulink logged-signals data
- Built-in management of time units
- Removal or interpolation of missing data
- Resampling of data
- Arithmetic operations for `timeseries` objects
- Synchronization of time series

Note Due to reported instabilities on the Linux 64 platform, you must manually enable the Time Series Tools feature before starting Time Series Tools.

To manually enable Time Series Tools on the Linux 64 platform, type the following at the MATLAB prompt:

```
rehash toolboxcache
feature('TimeSeriesTools',1)
```

Programming

New Functions

This version introduces the following new functions:

Function	Description
arrayfun	Applies a given function to each element of an array. This is especially useful for arrays of structures.
exifread	Reads EXIF information from JPEG and TIFF image files
structfun	Applies a given function to each field of a structure
swapbytes	Swaps byte ordering
typecast	Converts data types without changing underlying data

Compatibility Considerations

A new function name can potentially introduce a backward incompatibility since it can, under certain circumstances, override a variable with the same name as the new function. This is especially true for names that are commonly used as variable names in program code. Read the section Variable Names in the MATLAB Programming documentation to find out how to avoid having these variables misinterpreted.

Modified Functions

The following functions were modified in this version:

Function	Modified Behavior
cellfun	Applies a given function to each cell of a cell array
datestr	Seconds field truncates instead of rounding
error	Saves stack information that you can retrieve using `lasterror`
isfield	Supports cell array input
lasterror	Returns stack information on last error
rethrow	Accepts stack information as input

Function	Modified Behavior
who, whos	Displays information separately for nested functions

Compatibility Considerations

The following functions might, under certain circumstances, return a different value than what was returned in MATLAB 7.0.4 (R14SP2):

- datestr: Output might differ by 1 second from what was returned in a previous version.
- isfield: Output might differ if you used this feature in a release in which it was not officially supported.

Evaluation Functions for Arrays, Structures, Cells

MATLAB offers the capability to apply a given function to each element of an array, each field of a structure, or each cell of a cell array. See the help on arrayfun, structfun, and cellfun for more information.

Using who and whos with Nested Functions

When you use `who` or `whos` inside of a nested function, MATLAB returns or displays all variables in the workspace of that function, and in the workspaces of all functions in which that function is nested. This applies whether you include calls to `who` or `whos` in your M-file code or if you call `who` or `whos` from the MATLAB debugger. See thewho reference page for more information.

Date and Time Functions Support Milliseconds

The datestr, datenum, and datevec functions now support time specification in milliseconds. Use the symbol `.FFF` to represent milliseconds in any of these three functions. See the table labeled Free-Form Date Format Specifiers on the datestr reference page for more information.

Stack Trace Provided for lasterror

The lasterror function now returns an additional field in the structure that it returns. The new `stack` field contains information from the stack on the M-file, function, and line in which the error occurred.

You can use this stack information to track down the source of an error, or as an input to the rethrow function. When used with `rethrow`, MATLAB sets the stack of the rethrown error to the value contained in the `stack` input.

isfield Function Supports Cell Arrays; Results Might Differ from Previous Version

The isfield function now supports cell array input as shown in this example. Check structure `S` for any of four possible field names. In this case, only the first is found, so the first element of the return value is set to true:

```
S = struct('one', 1, 'two', 2);

fields = isfield(S, {'two', 'pi', 'One', 3.14})
fields =
     1     0     0     0
```

Compatibility Considerations

There might be backward compatibility issues associated with this change if you used `isfield` with cell array input in a previous release. In previous releases, although `isfield` might have worked with this type of input in certain cases, it was not officially a supported feature. If you used this previously unsupported syntax in previous releases, you may see a change in the content and/or size of the return values in this release.

For example, create a structure `s` with three fields `a`, `b`, and `c` created in that order. In MATLAB 7.0.4, `isfield` called with a cell array input returns `true` if any of the elements of the cell array matches a field name, and if that element is in the same position in the cell array as the field is in the structure. This is true for `'c'`:

```
isfield(s, {'b'; 'a'; 'c'})
ans =
     1
```

In MATLAB 7.1, `isfield` returns `true` for each element in the cell array that matches a field name, regardless of where the string is positioned in the cell array. This is true for `'a'`, `'b'`, and `'c'`:

```
isfield(s, {'b'; 'a'; 'c'})
ans =
     1
```

```
     1
     1
```

Support for Reading EXIF Data from Image Files

You can now read EXIF (Exchangeable Image File Format) data from JPEG and TIFF graphics files using the new exifread function. EXIF is a standard used by digital camera manufacturers to store information in the image file, such as the make and model of a camera, the time the picture was taken and digitized, the resolution of the image, exposure time, and focal length.

Performance Improvements to the MATLAB JIT/Accelerator on Macintosh

The JIT/Accelerator for MATLAB, introduced in MATLAB Version 6.5 for Windows and UNIX, is now also supported on Macintosh systems. The JIT/Accelerator affects the performance of MATLAB and can give you a substantial performance increase over earlier MATLAB versions for many MATLAB applications.

Specifying fread Precision as Number of Bits

The following information on the fread function applies to MATLAB 7.1 and also to earlier versions.

MATLAB provides the following method of specifying a `precision` argument in a call to `fread`:

```
input_format=>output_format
```

For example, to read 50 8-bit unsigned integers from a file and convert them to characters, you can use

```
c = fread(fid, 50, 'uint8=>char')'
```

If the input format and output format are the same, you can abbreviate the precision specifier by using

```
*input_format
```

For example, you can replace

```
c = fread(fid, 50, 'uint8=>uint8')'
```

with

```
c = fread(fid, 50, '*uint8')'
```

You can also use this notation with an input stream that is specified as a number of bits (e.g., `bit4` or `ubit18`). MATLAB translates this into an output type that is a signed or unsigned integer (depending on the input type), and which is large enough to hold all of the bits in the source format. For example, `*ubit18` does not translate to `ubit18=>ubit18`, but instead to `ubit18=>uint32`.

Seconds Field Now Truncated; Results Might Differ

When handling time data, MATLAB now truncates the `seconds` field instead of rounding it. This is consistent with the way that MATLAB handles `hours` and `minutes`.

For example, using MATLAB 7.0.4 (R14SP2), datestr returns

```
t = datestr('11:30:01.666')
t =
   01-Jan-2005 11:30:02
```

while MATLAB 7.1 (R14SP3) returns

```
t = datestr('11:30:01.666')
t =
   01-Jan-2005 11:30:01
```

Compatibility Considerations

If your M-files relied on the previous behavior, you might get different results.

Built-in Functions No Longer Use .bi; Impacts Output of which Function

In previous releases, MATLAB function dispatching located built-in functions by means of special files having a `.bi` file extension. MATLAB no longer uses this mechanism to locate built-in functions. All `.bi` files have been removed in MATLAB 7.1.

Compatibility Considerations

If you have M-files that relied on built-in files having a `.bi` extension, your files need to accommodate this change.

There are changes in how MATLAB displays built-in functions using `which`:

In MATLAB 7.0.4 (R14SP2),

```
which -all int32
\\matlab\toolbox\symbolic\@sym\int32.m          % sym method
\\matlab\toolbox\matlab\datatypes\int32.bi      % Shadowed
\\matlab\toolbox\matlab\datatypes\int32.m       % Shadowed
```

In MATLAB 7.1 (R14SP3),

```
which -all int32
built-in (\\matlab\toolbox\matlab\datatypes\int32)
\\matlab\toolbox\symbolic\@sym\int32.m          % sym method
```

New Warning About Potential Naming Conflict

If you change directories (cd) or add a new directory to your current MATLAB path, and the new directory contains an M-file having the same name as a MATLAB built-in function, MATLAB now displays a warning alerting you to the potential naming conflict. For example,

```
Warning: Function D:\test\matlab\disp.m has the same name as a
MATLAB builtin. We suggest you rename the function to avoid a
potential name conflict.
```

In general, any file system event that leads to path refreshing in MATLAB can trigger this warning if the directory involved in this event has such a user function under it.

Compatibility Considerations

MATLAB might generate warnings about naming conflicts that did not appear in previous versions. To avoid this warning, renaming your M-files that have name conflicts with built-in functions.

Graphics and 3-D Visualization

Plot Tools Now Available on Mac Platform

As a consequence of enabling Java figures on Macintosh, the Plot Tools user interface is now available to Mac users, enabling them to interactively add data to plots, change plot symbology, and otherwise customize their data plots.

Documentation for Data Analysis Reorganized

Documentation explaining techniques for analyzing graphical data has been shifted from the Graphics book of the MATLAB documentation to a new book called Data Analysis.

Creating Graphical User Interfaces (GUIs)

Plans for Obsolete Functions

The table below indicates functions that were designated as obsolete prior to R14SP3 and that will be removed in a future version.

Compatibility Considerations

If you use these functions, you should use replacement functions instead.

Obsolete Function	Removed from Version	Replacement
`clruprop`	Future version	rmappdata
`ctlpanel`	Future version	guide
`extent`	Future version	`get(txtobj,'extent')`
`figflag`	Future version	findobj to determine if figure exists. `figure(fighandle)` to bring figure to front and give it focus.
`getuprop`	Future version	getappdata
`hthelp`	Future version	web
`layout`	Future version	None provided
`matq2ws`	Future version	None provided
`matqdlg`	Future version	None provided
`matqparse`	Future version	None provided
`matqueue`	Future version	None provided
`menuedit`	Future version	guide
`menulabel`	Future version	Use `'&'` to specify mnemonics and `'Accelerator'` property to define accelerator keys.
`setuprop`	Future version	setappdata
`wizard`	Future version	None provided
`ws2matq`	Future version	None provided

External Interfaces/API

mex Switches Now Supported on Microsoft Windows

MATLAB now supports the `-l` and `-L` options to the mex command on Windows operating systems. In previous releases of MATLAB, these options were supported only on UNIX systems.

Switch	Description
`-l`	Specifies additional libraries to link against. **Note** On Windows operating systems, the `-l` option can specify libraries of two forms. For example, specifying `-l name` matches either `name.lib` or `libname.lib`, whereas on UNIX it matches only `libname.lib`.
`-L`	Specifies a path to use when MATLAB searches for library files specified with the `-l` option. The `-L` option must precede the `-l` option.

For the switches you can use with the `mex` command, see the MEX Script Switches table in the Custom Building MEX-Files section of Creating C/C++ and Fortran Programs to be Callable from MATLAB (MEX-Files) in the MATLAB External Interfaces documentation.

New COM Programmatic Identifier

There is now a `ProgID` that enables you to use the full desktop version of MATLAB as an automation server.

`Matlab.Desktop.Application` starts an automation server using the most recent version of MATLAB that is installed on your system.

New File Extension for MEX-Files on Windows Systems

MATLAB now uses the extension `.mexw32` for MEX-files on 32-bit versions of Windows systems. In previous versions, MATLAB used the extension `.dll`.

MathWorks recommends that you recompile all MEX-files after installing MATLAB 7.1. MEX-files compiled in MATLAB 7.0.4 with `.dll` extensions should still work in MATLAB 7.1.

There may be two MEX-files with the same name, except that one has a `.mexw32` extension and the other has a `.dll` extension. When these files are both on the MATLAB search path:

- If the two files are in the same directory, MATLAB uses the `.mexw32` file.
- If the two files are in different directories, MATLAB uses the file in the directory that is higher on the search path.

If you want one of these two files to take precedence over the other, ensure that the directory that contains the file you want MATLAB to use is higher on the search path than the directory that contains the file you do not want MATLAB to use.

Compatibility Considerations

Previous versions of MATLAB do not recognize MEX-files compiled in MATLAB 7.1 with `.mexw32` extensions. However, you can use the mex `-output` option in MATLAB 7.1 to build a MEX-file with a `.dll` extension that earlier versions of MATLAB can recognize.

You may need to update any MATLAB scripts or makefiles that explicitly expect `.dll` extensions for compiled MEX-files. You can use the mexext function in MATLAB to obtain the extension for the platform and version you are working on. A new `mexext` script obtains the appropriate extension when executed from outside MATLAB, as in a makefile.

On Windows systems, MATLAB issues warnings at MEX setup time, compile time, and run-time to notify you of possible incompatibilities resulting from the change in MEX-file extension from `.dll` to `.mexw32`.

New mex–output Behavior for Compatibility

The `-output` option to mex specifies the filename of the compiled MEX-file. In general, `mex` ignores any filename extension supplied in the `-output` argument and uses the extension for the compiled file that is appropriate for the architecture. However, on Windows systems, if the `-output` argument specifies a `.dll` extension, the compiled file has this extension instead of `.mexw32`. Previous versions of MATLAB can recognize the resulting compiled file.

Conflicting MEX-Files Renamed Automatically

If two files with the same name but with `.mexw32` and `.dll` extensions exist in the same directory, MATLAB uses the `.mexw32` file. To avoid unintended shadowing, MATLAB automatically renames compiled MEX-files under the following circumstances:

- When you build a MEX-file with a `.mexw32` extension and the directory contains an existing file with the same name, but with a `.dll` extension, the extension of the `.dll` file is changed to `.dll.old`.
- When you build a MEX-file with a `.dll` extension (using the mex `-output` option) and the directory contains an existing file with the same name, but with a `.mexw32` extension, the extension of the `.mexw32` file is changed to `.mexw32.old`.

New Return Value for mexext on Windows Systems

On 32-bit Windows platforms, the mexext function now returns `mexw32`. In MATLAB 7.0.4 it returned `dll`.

New mexext Script to Obtain MEX-File Extension in Makefiles

A new script displays the MEX-file extension in the current version of MATLAB that corresponds to the platform on which the script is executed. It is intended to be used outside MATLAB, in makefiles or scripts, to obtain the appropriate filename extension for compiled MEX-files. Use this script instead of explicitly specifying the MEX-file extension in a makefile.

The script is named `mexext.bat` on Windows platforms and `mexext.sh` on UNIX platforms. It is located in the directory `$matlab/bin`, where `$matlab` represents the string returned from the matlabroot command.

The script displays the MEX-file extension without a leading period. For example, on 32-bit Windows platforms, it returns `mexw32`.

Following is a fragment of a GNU makefile that uses the `mexext` script to obtain the MEX-file extension:

```
ext = $(shell mexext)

yprime.$(ext) : yprime.c
 mex yprime.c
```

New Preferences Directory and MEX Options

The MATLAB preferences directory has changed. In MATLAB 7.1, the preferences directory is named `R14SP3`. In previous R14 releases, the preferences directory was named `R14`. For more information, see the documentation for prefdir, which returns the preferences directory.

Compatibility Considerations

When you install MATLAB 7.1, MATLAB migrates some files from any existing `R14` preferences directory to the new `R14SP3` directory. However, MATLAB does not migrate the MEX options file, `mexopts.bat`. If you want to preserve any MEX options that you have customized in an earlier R14 release, you need to migrate your options to the new `R14SP3` preferences directory.

You can migrate your MEX options in either of two ways:

- If you have customized only a few options: Invoke mex with the `-setup` option to create a new `mexopts.bat` file in the `R14SP3` preferences directory. Edit the new `mexopts.bat` file to customize the MEX options there.
- If you have customized many options: Copy your customized `mexopts.bat` file from the old `R14` preferences directory to the new `R14SP3` directory. Edit at least the settings of the LIBLOC and NAME_OUTPUT linker parameters in the `mexopts.bat` file. These lines should look as follows on Windows systems when using a Microsoft compiler:

  ```
  set LIBLOC=%MATLAB%\extern\lib\win32\microsoft
  set NAME_OUTPUT=/out:"%OUTDIR%%MEX_NAME%%MEX_EXT%"
  ```

The `LIBLOC` parameter has changed because import libraries have moved; see “Import Libraries Moved” on page 22-32. The value of this parameter depends on the platform you are running MATLAB on and the vendor of the compiler you are using.

The `NAME_OUTPUT` parameter has changed because the extension for compiled MEX-Files has changed on Windows systems; see “New File Extension for MEX-Files on Windows Systems” on page 22-28.

Compiler Support

The set of compilers that MATLAB supports has changed in MATLAB 7.1. For an up-to-date list of supported compilers, see the Supported and Compatible Compilers Web page.

Compatibility Considerations

You may need to recompile code compiled with an earlier compiler that is no longer supported.

Import Libraries Moved

The import libraries (`.lib` files) for the MATLAB `dll` files have been moved up a directory level and are no longer specific to the compiler version. The new location for these files is

```
$matlab/extern/lib/$arch/$vendor
```

where the terms `$matlab`, `$arch`, and `$vendor` respectively represent the string returned from the matlabroot command, the platform you are running MATLAB on, and the vendor of the compiler you are using.

Compatibility Considerations

You may need to change any code that depends on the previous library locations.

MEX Perl Script Moved

The MEX Perl script used in building MEX-files is now located in `$matlab/bin`, rather than `$matlab/bin/win32`. (The term `$matlab` represents the string returned by the matlabroot function.) You should not notice any difference, however, as a batch file located in `$matlab/bin/win32` provides backward compatibility.

Linking to System Libraries

MATLAB now links with the system libraries by default. You no longer need to specify them explicitly.

COM Automation Server Now Displays Figure

When using MATLAB as an Automation server, executing MATLAB commands that create figures now displays the figure window.

Previous releases of MATLAB created the figure in the background. To duplicate the old behavior, create a figure with its `Visible` property set to `off`, then set the property to `on` when you want the figure to be visible:

```
h = actxserver('matlab.application');
h.Execute('figure visible off');
h.Execute('plot(1:10)');
h.Execute('set(gcf,''visible'',''on'')');
```

R14SP2

Version: 7.0.4

New Features

Compatibility Considerations

Desktop Tools and Development Environment

Installation Folder with Spaces

In MATLAB 7.0.4 (R14SP2) software, the following two changes have been made to the MathWorks Installer on Microsoft Windows platforms:

- The Installer now allows a folder name with spaces in the installation path.
- The Installer honors the default installation folder for Windows software, which on most machines is `Program Files`.

These changes were made in response to many customer requests and the desire to conform to a widely established industry practice for the PC platform.

Note MathWorks products are used and integrated into many software environments. If you use MathWorks products in conjunction with other third party applications (compilers, other numerical analysis packages, etc.) you might want to continue installing into a folder that does not have spaces in the path until you have tested that those applications work with MathWorks products.

Startup and Shutdown

Confirmation Dialog Box for Quitting Added

When quitting MATLAB, a confirmation dialog box appears if you set a new preference for that purpose. By default, the confirm quitting preference is not set, so the dialog box will not appear. To change the preference, see the instructions for Confirmation Dialogs Preferences in the desktop documentation.

JVM Software Updated

MATLAB is now using version 1.5 of Sun Microsystems Java JVM software on Windows, Linus Torvalds Linux (32-bit), and Sun Microsystems Solaris platforms. Java software is supplied with MATLAB, so this change requires no action on your part.

Compatibility Considerations

If you use a specific version of Java with MATLAB on Windows, Linux 32-bit, or Solaris platforms, this change might affect you.

Desktop

Confirmation Dialog Boxes Preference Introduced

There are new preferences for displaying or not displaying confirmation dialog boxes for desktop tools. In previous versions, some of these preferences existed but were located with other preferences for the associated desktop tool. They are now organized in one preference panel for all desktop tools. Access them by selecting **File > Preferences > General > Confirmation Dialogs**. These preferences work in conjunction with the **Do not show this prompt again** check boxes that appears on various desktop confirmation dialog boxes. For more information, see Confirmation Dialogs Preferences in the desktop documentation.

Running Functions — Command Window and History

Overwrite Mode Now Supported

The Command Window now supports overwrite mode. Press the **Insert** key to enter text in overwrite mode. Press the **Insert** key again to return to entering text in insert mode. View the current state at the far right end of the status bar of the Command Window when it is undocked, or in the desktop when the Command Window is docked and has focus. In insert mode, **OVR** in the status bar is gray and the cursor has a wide block shape.

Hyperlink Color Preference Added

Set the color of hyperlinks that display in the Command Window. Select **File > Preferences > Command Window**, and under **Display**, select **Hyperlink color**.

Help

Subfunction Help Syntax Changed

To get help for a subfunction, use

```
help functionname>subfunctionname
```

Compatibility Considerations

In previous versions, the syntax was `help functionname/subfunctionname`. This change was introduced in R14 (MATLAB 7.0) but was not documented.

Bug Fixes and Known Problems Now on Web; No Longer Found Via Help Search

The Release Notes sections “Major Bug Fixes” and “Known Software and Documentation Problems” no longer include the content in the installed help files. Instead, the sections provide links to these lists on the MathWorks Web site. The lists on the Web site can be updated after the release date to reflect the latest information.

Compatibility Considerations

As a result of this change, the Help browser **Search** feature will not find search terms that are in the content of those reports. Use the MathWorks Web site search features to look for search terms in those reports.

Workspace, Search Path, and File Operations

Formatting Decimal Separator when Copying From the Array Editor

You can now specify how you want decimal numbers to be formatted when you cut or copy cells from the Array Editor and paste them into text files or other applications. You can specify a separator for this purpose in the `Array Editor` panel of the **Preferences** dialog. The **Decimal separator to use when copying** edit field is by default `"."` (period). If you are working in or providing data to a locale that uses a different character to delimit decimals, type that character in this edit field and click **OK** or **Apply**.

Workspace Browser Preference Panel Removed

The Workspace browser preferences panel was removed. The entry on that panel was for confirming deletion of variables. That preference is now part of **General** > **Confirmation Dialogs** preferences.

Compatibility Considerations

Use **File** > **Preferences** > **General** > **Confirmation Dialogs** instead of **File** > **Preferences** > **Workspace Browser**.

Current Directory Browser Preferences Added

There are new Current Directory browser preferences you can access by selecting **File** > **Preferences** > **Current Directory Browser, Browser display options**:

- View the file size by selecting the **Show file sizes** check box. (This is selected by default.)
- For models created with Simulink software, view brief descriptions in the **Description** column when **Show M and MDL file descriptions** is selected.
- For models created with Simulink, view the complete descriptions in the lower pane when the preference for **Show M, MDL and MAT file contents** is selected. This allows you to view information about a model without running Simulink.

Editing and Debugging M-Files

Go To Subfunction or Nested Function

Go directly to a subfunction or nested function within an M-file using the enhanced **Go To** dialog box. Access the dialog box by selected **Edit > Go To**. Click the **Name** column header to arrange the list of functions alphabetically, or click the **Line** column header to arrange the list by the position of the functions in the file.

Help Browser Now Accessible from MATLAB Stand-Alone Editor

You can now access the MATLAB Help browser from the MATLAB stand-alone Editor. This provides you with documentation for MATLAB, including using Editor features and MATLAB functions.

Preference for Editor/Debugger Dialog Moved

The **Show dialog prompt** preference has been moved to **Preferences > General > Confirmation Dialogs**. For more information, see Confirmation Dialogs Preferences in the desktop documentation.

Compatibility Considerations

Use **File > Preferences > General > Confirmation Dialogs** instead of **File > Preferences > Editor/Debugger** to set this preference.

Dragging Text Maintains Font and Highlighting

Now, when you drag text from the Editor/Debugger to another application, it maintains the syntax highlighting and font characteristics.

Source Control Interface

Register Project Feature Added; Add to Source Control Behavior Changed

There is a new source control interface feature for Windows platforms, **Register Project with MATLAB**. Use this to associate all files in a directory with a source control project. You perform this for any file in a directory, which registers the directory and all files in that directory. You only perform this once in a directory, and must perform it before you perform any other source control actions for files in that directory.

Access the feature in the Current Directory browser by right-clicking a file and selecting **Source Control > Register Your Source Control System Project with MATLAB** from the context menu. You can also access it from the Editor/Debugger **File** menu. To access the feature for files created with Simulink or Stateflow® software, use the Current Directory browser.

For a summary of the process, see the topic Source Control Interface on Microsoft Windows in the desktop documentation.

Compatibility Considerations

In previous releases, this feature was part of the **Add to Source Control** feature. You still need to add each file to source control, but you do this after first registering the directory that contains the file.

Project Name Exact Match No Longer Required

The name of the project in the source control system is no longer required to exactly match the name of the directory on disk containing the files.

Publishing Results

Cell Publishing: File Extension Changes

The files created when publishing using cells now have more natural extensions. JPEG files now have a `.jpg` instead of a `.jpeg` extension, and EPSC2 files now have an `.eps` instead of an `.epsc2` extension.

Compatibility Considerations

If you relied on the formerly used file extensions, you need to accommodate the changes.

Cell Publishing: LaTeX Image File Type Changes

Publishing to LaTeX now respects the image file type you specify in preferences rather than always using EPSC2 files.

Cell Publishing: Image Options More Restrictive

The **Publish image options** in Editor/Debugger preferences for **Publishing Images** have changed slightly. The changes prevent you from choosing invalid formats.

Notebook Support for Microsoft Word 97 Application to be Discontinued

Notebook will no longer support the Microsoft Word 97 application starting in the next release of MATLAB.

Compatibility Considerations

If you use Word 97 with Notebook, you will need to migrate to a more recent version.

Mathematics

New Vendor BLAS Used for Linear Algebra in MATLAB

MATLAB uses Basic Linear Algebra Subprograms (BLAS) for its vector inner product, matrix-vector product, matrix-matrix product, and triangular solvers in `\`. MATLAB also uses BLAS behind its core numerical linear algebra routines from Linear Algebra Package (LAPACK), which are used in functions like `chol`, `lu`, `qr`, and within the linear system solver `\`.

Starting in this release

- On Macintosh, MATLAB now uses the Accelerate framework.
- On 64-bit Linux, MATLAB uses Intel Math Kernel Library (MKL) 7.0.1 on Intel chips, and AMD Core Math Library (ACML) 2.0 on AMD chips.

max and min on Complex Integers Not Supported

Using the max and min functions on complex integer inputs (as shown in the example below) is no longer supported. This operation had been supported from release R11 through R14SP1, but now returns an error.

```
max(int8([3-4i 3+4i]))
```

Compatibility Considerations

Any code that calls `max` or `min` on complex integers should be removed from your program files.

Programming

Memory-Mapping

Memory-mapping is a mechanism that maps a portion of a file, or an entire file, on disk to a range of addresses within an application's address space. The application can then access files on disk in the same way it accesses dynamic memory. This makes file reads and writes faster in comparison with using functions such as fread and fwrite.

Another advantage of using memory-mapping in MATLAB is that it enables you to access file data using standard MATLAB indexing operations. Once you have mapped a file to memory, you can read the contents of that file using the same type of MATLAB statements used to read variables from the MATLAB workspace. The contents of the mapped file appear as if they were an array in the currently active workspace. You simply index into this array to read or write the desired data from the file.

Memory-mapped files also provide a mechanism for sharing data between applications. This is achieved by having each application map sections of the same file. This feature can be used to transfer large data sets between MATLAB and other applications.

textscan Enhancements

The textscan function originally read data only from files. As of this release, you can use `textscan` to read from strings as well.

xlsread Enhancements

In this release, you can write a function and pass a handle to this function to xlsread. When `xlsread` executes, it reads from the spreadsheet, executes your function on the data read from the spreadsheet, and returns the final results to you.

You can use either of the following syntaxes:

```
num = xlsread('filename', ..., functionhandle)
[num, txt, raw, X] = xlsread('filename', ..., functionhandle)
```

For an example, see the xlsread reference page.

xlsread Imported Date Format Changes

In MATLAB versions prior to R14, date values read into MATLAB from an Excel spreadsheet using xlsread were always imported as numeric date values. The R14 and later releases of MATLAB import dates in the format in which they were stored in the Excel file. Dates stored in string or date format are now imported as strings by `xlsread`. Dates stored in numeric format are imported as numeric date values.

Compatibility Considerations

Because of a difference in the way Excel and MATLAB compute numeric date values, any numeric dates imported from Excel into MATLAB must be converted to the MATLAB format before being used in the MATLAB application. For more information, see When to Convert Dates from Excel Files.

format Options Added

You can display MATLAB output using two new formats: `short eng` and `long eng`. See the format reference page for more information.

- `short eng` — Displays output in an engineering format that has at least 5 digits and a power that is a multiple of three.
- `long eng` — Displays output in an engineering format that has exactly 16 significant digits and a power that is a multiple of three.

```
format short eng
pi
ans =
     3.1416e+000

format long eng
pi
ans =
     3.14159265358979e+000
```

Nonscalar Arrays of Function Handles to Become Invalid

Creation of nonscalar arrays of function handles by str2func may be invalid or may return different results in future versions of MATLAB, but will continue to work in R14.

Compatibility Considerations

To avoid this warning and prepare for this change, convert the cell array of strings to a cell array of function handles.

For more information, type `help function_handle` and see the section entitled Note on Backward Compatibility.

Assigning Nonstructure Variables As Structures Displays Warning

Assigning to a nonstructure variable as if it were a structure is not recommended in MATLAB. For example, if variable `x` holds a double (as shown below), then attempting to add a fieldname to it, thus converting `x` to a structure, is not good programming practice and should generate an error.

```
x = 10;
x.name = magic(3);
```

Note that if `x` were empty (i.e., `x == []`), then assigning a field to it as if it were already a structure is acceptable.

Behavior Prior to Release R14

Because of a bug in releases of MATLAB prior to R14, you can assign a field to a nonempty, nonstructure variable in those releases without MATLAB generating a warning message or error. The result is that MATLAB quietly converts the variable to a structure:

```
x = 10;
class(x)
ans =
      double

x.name = magic(3);      % Invalid expression completes
                        %    without warning or error.
class(x)
ans =
      struct
```

Behavior In R14 and Later

In the MATLAB R14 and R14 service pack releases, you can still perform this type of operation, but MATLAB now displays a warning message:

```
x = 10;
x.name = magic(3);

Warning: Struct field assignment overwrites a value with class
"double".
```

In a future release of MATLAB, attempting this type of operation will throw an error instead of just displaying a warning message.

Compatibility Considerations

You are encouraged to modify any code that generates this warning. The section “Making a Valid Assignment” on page 23-12 gives instructions on how to do this.

Another Case — Extending the Depth of a Structure

The same rules apply when extending the depth of a structure by adding additional, lower-level fields. The first line of the example shown below creates a structure named `handle` and assigns to it a field of type double named `output`. The line after that treats this double as if it were a structure by attempting to assign a field named `time` to it. The second line is an invalid expression:

```
handle.output = 5;
handle.output.time = 13;
```

As in the case discussed earlier, this assignment does not generate a warning or error in MATLAB releases prior to R14. In the R14 and R14 service pack releases of MATLAB, you get the warning shown in the previous example. Beginning in a future release of MATLAB, this assignment will throw an error.

Making a Valid Assignment

To avoid this warning and future errors, first make x an empty structure or empty array as shown here. Once a variable is established as a structure or empty array, you can assign fields to it without getting an error:

```
x = struct;     or     x = [];
x.name = magic(3);
```

In the case of extending the depth of an existing structure, you can perform this type of assignment without generating a warning or error using the struct function as shown here:

```
handle.output = struct('time', 13);
```

Function Declaration Compatibililty with Pre-R14 M-Files

As of Release 14, the function definition line in a function M-file no longer requires commas separating output variables. However, because this syntax is not compatible with earlier releases, you should always include the comma separators when writing an M-file function that you intend to run on releases both earlier and later than Release 14.

Compatibility Considerations

See Comma Separators Not Required in Function Declaration in the Release 14 release notes.

Graphics and 3-D Visualization

imwrite Now Supports GIF Export

The `imwrite` function now supports exporting image data in Graphics Interchange Format (GIF).

Cannot Dock Figures on Macintosh

You cannot dock figures in the Desktop, because MATLAB uses native figure windows on the Macintosh platform.

Compatibility Considerations

You cannot dock figure in the Desktop on the Macintosh

Plotting Tools Not Working on Macintosh

The plotting tools are not supported on the Macintosh platform. This means the Figure Palette, Plot Browser, and Property Editor do not work these platforms. To use the MATLAB 6 Property Editor, see the propedit command.

Compatibility Considerations

You cannot use plotting tools on the Macintosh.

Not All Macintosh System Fonts Are Available

MATLAB figures do not support the same fonts as native Macintosh applications. Use the uisetfont functions to see which fonts are available in the MATLAB environment.

Compatibility Considerations

Some Macintosh fonts are not available in MATLAB

XDisplay Property Setable on Motif-Based Systems

You can specify the value of the figure `XDisplay` property only on systems using Motif-Based figure windows.

Compatibility Considerations

The figure `XDisplay` property is supported only on systems using Motif.

Creating Graphical User Interfaces (GUIs)

New Callbacks Chapter

The Creating Graphical User Interfaces documentation offers a new chapter, in draft form, that attempts to bring information regarding callbacks into one place. It introduces the concepts and mechanisms with which you work, and explains some basic techniques for programming your GUI's behavior. This chapter is not yet complete, but you may find it useful, even in its current state, particularly if you are new to creating GUIs.

Temporarily, this new chapter appears as Appendix A, “Working with Callbacks (Draft).” It contains some new information, but also duplicates information that can be found in various places throughout the rest of the book. In cases where information has not yet been included in the new chapter, links take you to the main part of the book.

External Interfaces/API

New File Archiving Functions and Functionality

In addition to being able to `zip` and `unzip` compressed file archives, MATLAB software now supports the following archiving functions:

- `gzip`/`gunzip` — Compress/uncompress files in gzip format.

  `gunzip` reads archives from both file systems and URLs.
- `tar`/`untar` — Compress/extract files in a tar-file.

  `untar` reads archives from both file systems and URLs.
- The `unzip` function can now also open a zip archive from a URL.