

MATLAB Release Notes

Summary by Version	7
Version 7.2 (R2006a) MATLAB	11
Desktop Tools and Development Environment, MATLAB Version 7.2 (R2006a)	13
Mathematics, MATLAB Version 7.2 (R2006a)	27
Data Analysis, MATLAB Version 7.2 (R2006a)	29
Programming, MATLAB Version 7.2 (R2006a)	31
Graphics and 3-D Visualization, MATLAB Version 7.2 (R2006a)	39
Creating Graphical User Interfaces (GUIs), MATLAB Version 7.2 (R2006a)	41
External Interfaces/API, MATLAB Version 7.2 (R2006a) ...	43
Version 7.1 (R14SP3) MATLAB	57
Desktop Tools and Development Environment, MATLAB Version 7.1 (R14SP3)	58
Mathematics, MATLAB Version 7.1 (R14SP3)	73
Data Analysis, MATLAB Version 7.1 (R14SP3)	77
Programming, MATLAB Version 7.1 (R14SP3)	79
Graphics and 3-D Visualization, MATLAB Version 7.1 (R14SP3)	87

Creating Graphical User Interfaces (GUIs), MATLAB Version 7.1 (R14SP3)	89
External Interfaces/API, MATLAB Version 7.1 (R14SP3) ...	91
Version 7.0.4 (R14SP2) MATLAB	97
Desktop Tools and Development Environment, MATLAB Version 7.0.4 (R14SP2)	99
Mathematics, MATLAB Version 7.0.4 (R14SP2)	105
Programming, MATLAB Version 7.0.4 (R14SP2)	107
Graphics and 3-D Visualization, MATLAB Version 7.0.4 (R14SP2)	113
Creating Graphical User Interfaces (GUIs), MATLAB Version 7.0.4 (R14SP2)	115
External Interfaces/API, MATLAB Version 7.0.4 (R14SP2) .	117
Version 7.0.1 (R14SP1) MATLAB	119
Desktop Tools and Development Environment, MATLAB Version 7.0.1 (R14SP1)	121
Mathematics, MATLAB Version 7.0.1 (R14SP1)	127
Programming, MATLAB Version 7.0.1 (R14SP1)	131
Graphics, MATLAB Version 7.0.1 (R14SP1)	137
Creating Graphical User Interfaces (GUIs), MATLAB Version 7.0.1 (R14SP1)	139
External Interfaces/API, MATLAB Version 7.0.1 (R14SP1) .	141

Version 7 (R14) MATLAB	145
Desktop Tools and Development Environment, MATLAB Version 7 (R14)	147
Mathematics, MATLAB Version 7 (R14)	175
Programming, MATLAB Version 7 (R14)	201
Graphics and 3-D Visualization, MATLAB Version 7 (R14)	241
Creating Graphical User Interfaces (GUIs), MATLAB Version 7 (R14)	249
External Interfaces/API, MATLAB Version 7 (R14)	255
Version 6.5.1 (R13SP1) MATLAB	267
Fixed Bugs	277
Compatibility Considerations	289
Compatibility Summary for MATLAB	291
Version 7.2 (R2006a) Compatibility Summary for MATLAB	292
Version 7.1 (R14SP3) Compatibility Summary for MATLAB	294
Version 7.04 (R14SP2) Compatibility Summary for MATLAB	296
Version 7.01 (R14SP1) Compatibility Summary for MATLAB	298

Version 7 (R14) Compatibility Summary for MATLAB	300
Version 6.5.1 (R13SP3) Compatibility Summary for MATLAB	305

Summary by Version

This table provides quick access to what's new in each version. For clarification, see About Release Notes.

Version (Release)	New Features and Changes	Version Compatibility Considerations	Fixed Bugs and Known Problems	Related Documentation at Web Site
Latest Version V7.2 (R2006a)	Yes Details	Yes Summary	Bug Reports at Web site	Printable Release Notes: PDF V7.2 product documentation
V7.1 (R14SP3)	Yes Details	Yes Summary	Bug Reports at Web site	No
V7.0.4 (R14SP2)	Yes Details	Yes Summary	Bug Reports at Web site	No
V7.0.1 (R14SP1)	Yes Details	Yes Summary	Fixed bugs	No
V7 (R14)	Yes Details	Yes Summary	Fixed bugs	No
V6.5.1 (R13SP1)	Yes Details	Yes Summary	Fixed bugs	V6.5.1 product documentation

About Release Notes

Use release notes when upgrading to a newer version to learn about new features and changes, and the potential impact on your existing files and practices. Release notes are also beneficial if you use or support multiple versions.

If you are not upgrading from the most recent previous version, review release notes for all interim versions, not just for the version you are installing. For example, when upgrading from V1.0 to V1.2, review the New Features and Changes, Version Compatibility Considerations, and Bug Reports for V1.1 and V1.2.

New Features and Changes

These include

- New functionality
- Changes to existing functionality
- Changes to system requirements (complete system requirements for the current version are at the MathWorks Web site)
- Any version compatibility considerations associated with each new feature or change

Version Compatibility Considerations

When a new feature or change introduces a known incompatibility between versions, its description includes a **Compatibility Considerations** subsection that details the impact. For a list of all new features and changes that have compatibility impact, see the “Compatibility Summary for MATLAB” on page -291.

Compatibility issues that become known after the product has been released are added to Bug Reports at the MathWorks Web site. Because bug fixes can sometimes result in incompatibilities, also review fixed bugs in Bug Reports for any compatibility impact.

Fixed Bugs and Known Problems

MathWorks Bug Reports is a user-searchable database of known problems, workarounds, and fixes. The MathWorks updates the Bug Reports database as new problems and resolutions become known, so check it as needed for the latest information.

Access Bug Reports at the MathWorks Web site using your MathWorks Account. If you are not logged in to your MathWorks Account when you link to Bug Reports, you are prompted to log in or create an account. You then can view bug fixes and known problems for R14SP2 and more recent releases.

The Bug Reports database was introduced for R14SP2 and does not include information for prior releases. You can access a list of bug fixes made in prior versions via the links in the summary table.

Related Documentation at Web Site

Printable Release Notes (PDF). You can print release notes from the PDF version, located at the MathWorks Web site. The PDF version does not support links to other documents or to the Web site, such as to Bug Reports. Use the browser-based version of release notes for access to all information.

Product Documentation. At the MathWorks Web site, you can access complete product documentation for the current version and some previous versions, as noted in the summary table.

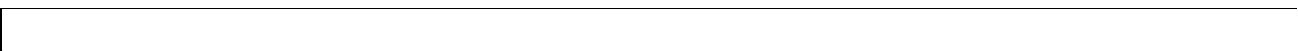
Version 7.2 (R2006a) MATLAB

This table summarizes what's new in Version 7.2 (R2006a):

New Features and Changes	Version Compatibility Considerations	Fixed Bugs and Known Problems	Related Documentation at Web Site
Yes Details below	Yes—Details labeled as Compatibility Considerations in descriptions of new features and changes. See also Summary.	Bug Reports at Web site	Printable Release Notes: PDF V7.2 product documentation

New features and changes introduced in this version are organized by these areas:

- Desktop Tools and Development Environment, MATLAB Version 7.2 (R2006a)
- Mathematics, MATLAB Version 7.2 (R2006a)
- Data Analysis, MATLAB Version 7.2 (R2006a)
- Programming, MATLAB Version 7.2 (R2006a)
- Graphics and 3-D Visualization, MATLAB Version 7.2 (R2006a)
- Creating Graphical User Interfaces (GUIs), MATLAB Version 7.2 (R2006a)
- External Interfaces/API, MATLAB Version 7.2 (R2006a)



Desktop Tools and Development Environment, MATLAB Version 7.2 (R2006a)

New features and changes introduced in this version are organized by these topics:

- Startup and Shutdown
- Desktop
- Running Functions—Command Window and Command History
- Help
- Workspace, Search Path, and File Operations
- Editing and Debugging M-Files
- Tuning and Managing M-Files
- Publishing Results
- Source Control Interface

Startup and Shutdown

New features and changes introduced in Version 7.2 (R2006a) are described here.

Installation Directory Structure on Windows

The installation directory structure on 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 MATLAB® root directory, as 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 MATLAB root directory, 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 MATLAB root directory structure 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 will 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 Updated for 64-Bit Linux Platforms

The Java Virtual Machine (JVM) version for 64-bit Linux platforms that MATLAB uses is now Sun 1.5.0_04.

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](#), and [Keyboard and Indenting Editor/Debugger Preferences Reorganized](#).

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 preference pane was removed. The tab size preference is now on the Command Window preference 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
```

and MATLAB displays the description of the Simulink® F-14 Digital Autopilot High Angle of Attack Mode model, 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 because the toolbox root directory is different than in MATLAB.

Editing and Debugging M-Files

New features and changes introduced in Version 7.2 (R2006a) are

- Tab Completion for Completing Function and Variable Names
- Go Menu Added; Bookmark and Go To Items Moved from Edit Menu to Go Menu
- Navigate Back and Forward in Files
- Keyboard and Indenting Editor/Debugger Preferences Reorganized
- M-Lint Automatic Code Analyzer Checks for Problems and Suggests Improvements
- Debugging Changes
- Cell Mode On by Default—Shows Cell Toolbar and Possibly Horizontal Lines and Yellow Highlighting; Cell Information Bar and Button Added
- Lines Between Cells
- Cell Titles in Bold Preference Removed

Tab Completion for Completing 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 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 use tab completion, select the **Keyboard** preference for it.

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.

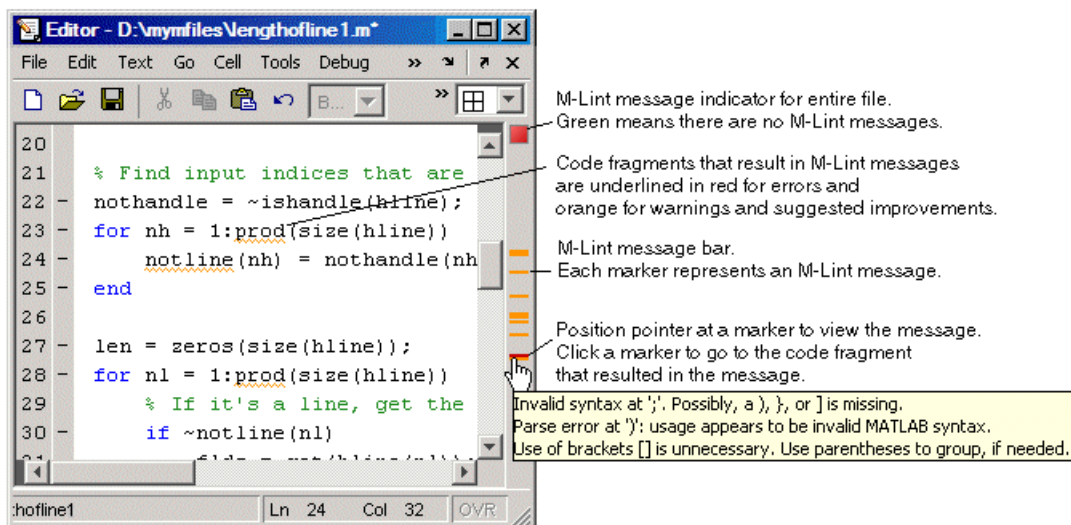
Keyboard and Indenting Editor/Debugger Preferences Reorganized

The Editor/Debugger **Keyboard and Indenting** preference 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 dropdown list to specify the types of code fragments that you want M-Lint to underline, for example, **Underline warnings and errors**.

Debugging Changes

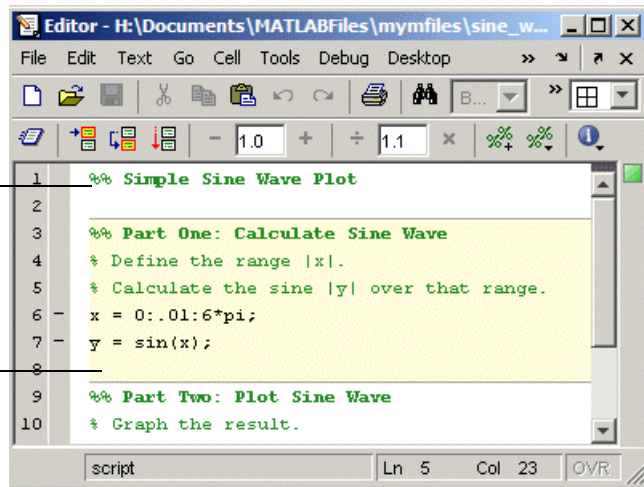
- The `dbstop` function now allows you to stop at, (not in), a non M-file, allowing you to view code and variables near it in your M-file. For example, if you want to stop at the point in your M-file `myfile.m` where the built-in `clear` function is called, run `dbstop in clear; myfile`. Use this feature with caution because the debugger will stop 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 will be interpreted as including cells. The Editor/Debugger will reflect 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.

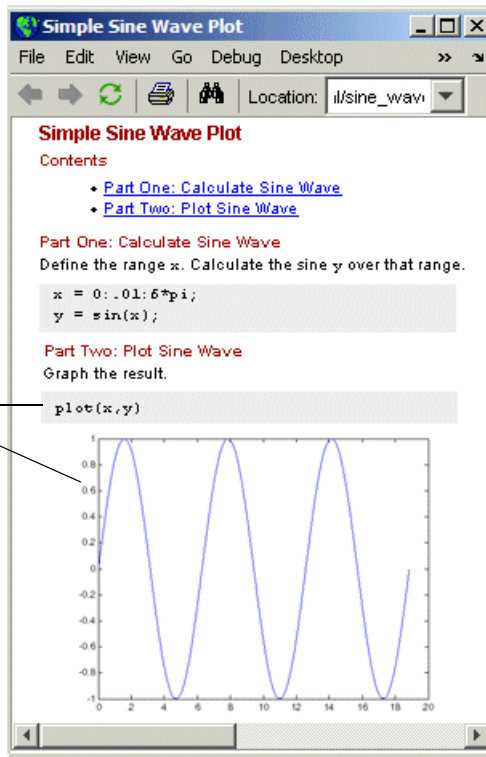
%% at the start of a line denotes a cell, used for publishing or rapid code iteration. The current cell is highlighted in yellow.



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. Users unfamiliar with cell mode will see the cell toolbar and might see yellow highlighting or horizontal rules in M-files that contain `%%` at the start of a line. 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
- Profiling Enhancements
- Visual Directory View Removed from Current Directory Browser

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 .

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 will have 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 'nohandle' might never be used.

while this is the form of the message from R14SP3
22 (C 1-9) InefficientUsage:AssignmentNotUsed : The value assigned here to variable 'nohandle' 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 'nohandle' 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 `-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. If the value is 1, it means 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 MATLAB source control interface on UNIX platforms reflects the change.

If you use ChangeMan on UNIX platforms, the `cmopts` value returned for it is `pvcs`. If you use PVCS, select ChangeMan in the Source Control Preferences pane.

Compatibility Considerations. PVCS users on UNIX platforms formerly selected PVCS in the Source Control Preferences pane. Now, PVCS users select ChangeMan instead.

Mathematics, MATLAB Version 7.2 (R2006a)

New features and changes introduced in this version are:

- New Library CHOLMOD for Sparse Cholesky Factorization
- New Solver for State-Dependent DDEs
- Upgrade to BLAS Libraries
- New Function for Integer Division
- New Input to gallery Function
- Improved Algorithm for expm
- More Efficient condst for Sparse Matrices
- accumarray Accepts Cell Vector Input

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 will return the vector of delays to be used by the solver. See the function reference page for `ddesd`, and “Initial Value Problems for DDEs” 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 condst for Sparse Matrices

The `condst` 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, MATLAB Version 7.2 (R2006a)

New features and changes introduced in this version are

- Data Analysis Collection Revised and Expanded
- Reference Pages for timeseries and tscollection Objects
- Text Files Can Be Imported In Time Series Tools
- Linux 64 Platform Fully Enabled for Time Series Tools

Data Analysis Collection Revised and Expanded

In this release, the 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 MATLAB Functions — By Category.

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, MATLAB Version 7.2 (R2006a)

New features and changes introduced in this version are:

- Larger Data Sets with 64-Bit Windows XP
- Using `avifile` and `movie2avi` on Windows XP 64
- Regular Expressions
- Setting Environment Variables
- Issorted Support for Cell Arrays
- XLS Functions Support More Formats
- Archiving Functions Accept Files on Path and `~/`
- `sendmail` No Longer Requires ASCII Messages
- MATLAB Warns on Invalid Input to `str2func`
- I/O Functions Can Specify and Use Character Encoding Schemes
- Character Encoding Conversion For Native-Encoded Files

Larger Data Sets with 64-Bit Windows XP

MATLAB support for Windows XP 64-bit edition enables you to handle much larger data sets. There remains a 2 GB limit on each variable, but you can store many more such variables in memory at one time.

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, you will see an error message 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 `regexptolate` 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 `regex` 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 pathnames. 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 will now generate 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.

I/O Functions Can Specify and Use Character Encoding Schemes

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 of 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 may have used `native2unicode` to convert input from such functions as `fread` to the MATLAB internal representation of characters using a particular character encoding scheme. Programs may have used `unicode2native` to convert output to such functions as `fwrite` from the MATLAB internal representation of characters using a particular character 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 incorrect results in V7.2 (R2006a). The calls to `native2unicode` and `unicode2native` are no longer necessary, because the

`fread` and `fwrite` functions now convert characters using a specified (or default) character encoding scheme. 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);
```

Character Encoding Conversion For Native-Encoded Files

Reading from a native-encoded file with `fread`, `fgets`, `fgetl`, or `fscanf` in previous versions of MATLAB required that you convert any character encodings in the file by explicitly calling the `native2unicode` function. Likewise, writing to a native-encoded file using `fwrite` required converting the encoding with `unicode2native`.

In this release of MATLAB, you can now read and write native-encoded files directly without having to call `native2unicode` when reading, or `unicode2native` when writing. In most cases, the character encoding conversion to and from Unicode is done for you automatically.

This applies to reading from a native-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'`

or writing to a native-encoded file using

- `fwrite` with precision set to `'char'` or `'char*1'`

Compatibility Considerations

If you have existing programs that perform the character encoding conversion with explicit calls to `native2unicode` and `unicode2native`, in most cases you can continue to use this same code without having to remove these calls from

the code. MATLAB produces the same results whether or not you perform the explicit conversion.

However, if you use `fwrite` with precision set to either `'char'` or `'char*1'` and explicitly convert character encoding using `unicode2native`, then the above paragraph does not hold true. In these cases, you need to modify the program code that performs the `fwrite`, removing any associated calls to the `unicode2native` function.

Example Using `fread`

If you used either of the following commands prior to release R2006a,

```
indata = native2unicode(fread(fid, '*char'));
indata = native2unicode(fread(fid, 'char=>char'));
```

You can, but do not need to, replace this code with

```
indata = fread(fid, '*char');
indata = fread(fid, 'char=>char');
```

Example Using `fgets`, `fgetl`, or `fscanf`

If you used any of the following commands prior to release R2006a,

```
indata = native2unicode(fgets(fid));
indata = native2unicode(fgetl(fid));
indata = native2unicode(fscanf(fid, '%s'));
indata = native2unicode(fscanf(fid, '%c'));
```

You can, but do not need to, replace this code with

```
indata = fgets(fid);
indata = fgetl(fid);
indata = fscanf(fid, '%s');
indata = fscanf(fid, '%c');
```

Example Using fwrite

If you used either of the following commands prior to release R2006a,

```
fwrite(fid, unicode2native(outbuff), 'char');  
fwrite(fid, unicode2native(outbuff), 'char*1');
```

You must replace this code with

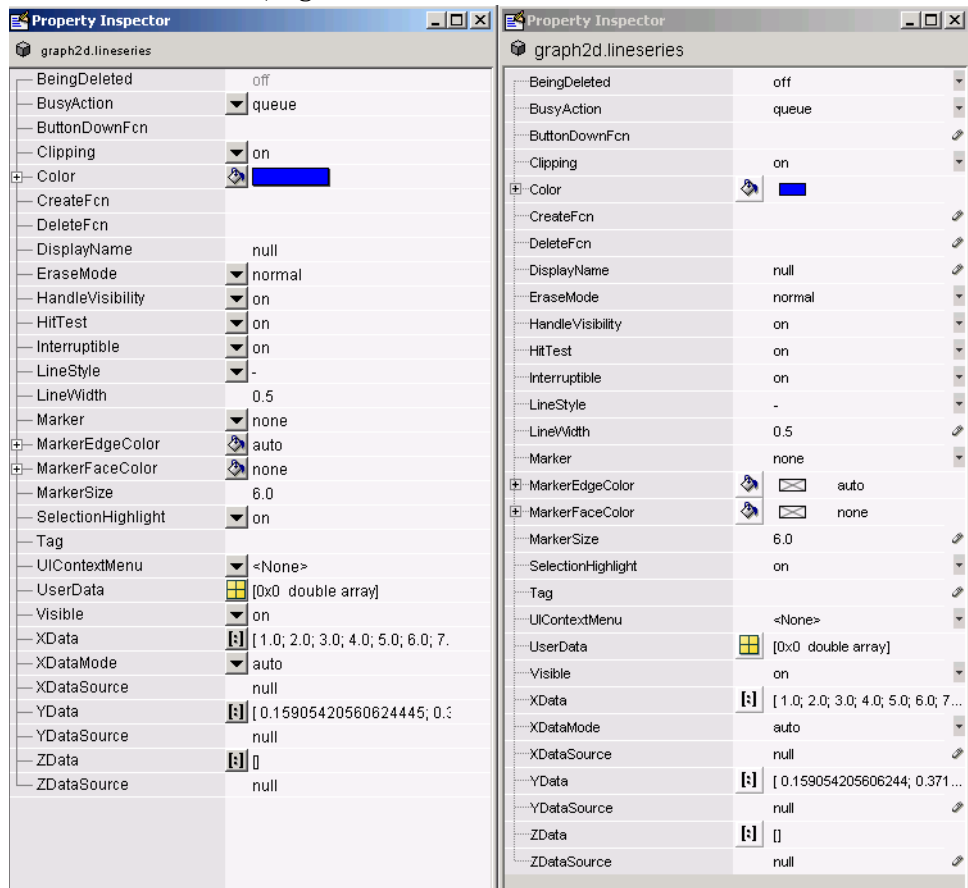
```
fwrite(fid, outbuff, 'char');  
fwrite(fid, outbuff, 'char*1');
```

Graphics and 3-D Visualization, MATLAB Version 7.2 (R2006a)

There is one change introduced in this version:

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 changes in fonts and lines 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), MATLAB Version 7.2 (R2006a)

New features and changes introduced in this version are:

- Treatment of '&' in Menu Label Is Changed
- Major Documentation Revision

Treatment of '&' in Menu Label Is Changed

The use of '&' (andpersand) 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.

Note Following the release of MATLAB version 7.2, Creating Graphical User Interfaces will be further updated and expanded. The PDF and HTML versions of this document will be updated on The MathWorks Web site some time after the release. Check the top page of the HTML document and the title page of the PDF to determine if they have been updated.

External Interfaces/API, MATLAB Version 7.2 (R2006a)

New features and changes introduced in this version are:

- MEX-Files Built with gcc on Linux Must Be Rebuilt
- MEX-Files in MATLAB for Windows x64
- New Microsoft and Intel Compilers Supported
- MWPOINTER Macro for Platform-Independent Fortran Code
- Compaq Visual Fortran Engine and MAT Options File Renamed
- Options Files Removed for Unsupported Compilers
- Obsolete Functions No Longer Documented
- Support for Licensed ActiveX Controls
- Support for VT_Date Type
- Dynamic Linking of External Libraries

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 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 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 (C)

Obsolete Function	Replacement
matDeleteArray (C)	matDeleteVariable (C)
matDeleteMatrix (C)	matDeleteVariable (C)
matGetArray (C)	matGetVariable (C)
matGetArrayHeader (C)	matGetVariableInfo (C)
matGetFull (C)	matGetVariable (C) followed by mxGetM (C), mxGetN (C), mxGetPr (C), mxGetPi (C)
matGetMatrix (C)	matGetVariable (C)
matGetNextArray (C)	matGetNextVariable (C)
matGetNextArrayHeader (C)	matGetNextVariableInfo (C)
matGetNextMatrix (C)	matGetNextVariable (C)
matGetString (C)	matGetVariable (C) followed by mxGetString (C)
matPutArray (C)	matPutVariable (C)
matPutArrayAsGlobal (C)	matPutVariableAsGlobal (C)
matPutFull (C)	mxCreateDoubleMatrix (C) followed by mxSetPr (C), mxSetPi (C), matPutVariable (C)
matPutMatrix (C)	matPutVariable (C)
matPutString (C)	mxCreateString (C) followed by matPutVariable (C)

Obsolete Functions: MX Array Manipulation (C)

Obsolete Function	Replacement
<code>mxClearLogical (C)</code>	None
<code>mxCreateFull (C)</code>	<code>mxCreateDoubleMatrix(C)</code>
<code>mxCreateScalarDouble (C)</code>	<code>mxCreateDoubleScalar(C)</code>
<code>mxFreeMatrix (C)</code>	<code>mxDestroyArray(C)</code>
<code>mxGetName (C)</code>	<code>matGetVariable (C)</code> , <code>mexGetVariable (C)</code> , or <code>engGetVariable (C)</code>
<code>mxIsFull (C)</code>	<code>mxIsSparse(C)</code>
<code>mxIsString (C)</code>	<code>mxIsChar(C)</code>
<code>mxSetLogical (C)</code>	None
<code>mxSetName (C)</code>	<code>matPutVariable (C)</code> , <code>mexPutVariable (C)</code> , or <code>engPutVariable (C)</code>

Obsolete Functions: MEX-Files (C)

Obsolete Function	Replacement
mexAddFlops (C)	None
mexGetArray (C)	mexGetVariable (C)
mexGetArrayPtr (C)	mexGetVariablePtr (C)
mexGetEps (C)	mxGetEps (C)
mexGetFull (C)	mexGetVariable (C) followed by mxGetM (C), mxGetN (C), mxGetPr (C), mxGetPi (C)
mexGetGlobal (C)	mexGetVariablePtr (C)
mexGetInf (C)	mxGetInf (C)
mexGetMatrix (C)	mexGetVariable (C)
mexGetMatrixPtr (C)	mexGetVariablePtr (C)
mexGetNaN (C)	mxGetNaN (C)
mexIsFinite (C)	mxIsFinite (C)
mexIsInf (C)	mxIsInf (C)
mexIsNaN (C)	mxIsNaN (C)
mexPutArray (C)	mexPutVariable (C)
mexPutFull (C)	mxCreateDoubleMatrix (C) followed by mxSetPr (C), mxSetPi (C), mexPutVariable (C)
mexPutMatrix (C)	mexPutVariable (C)

Obsolete Functions: MATLAB Engine (C)

Obsolete Function	Replacement
engGetArray (C)	engGetVariable (C)
engGetFull (C)	engGetVariable (C) followed by mxGetM (C), mxGetN (C), mxGetPr (C), mxGetPi (C)
engGetMatrix (C)	engGetVariable (C)
engPutArray (C)	engPutVariable (C)
engPutFull (C)	mxCreateDoubleMatrix (C) followed by mxSetPr (C), mxSetPi (C), engPutVariable (C)
engPutMatrix (C)	engPutVariable (C)
engSetEvalCallback (C)	None
engSetEvalTimeout (C)	None
engWinInit (C)	None

Obsolete Functions: MAT-File Access (Fortran)

Obsolete Function	Replacement
matDeleteArray (Fortran)	matDeleteVariable (Fortran)
matDeleteMatrix (Fortran)	matDeleteVariable (Fortran)
matGetArray (Fortran)	matGetVariable (Fortran)
matGetArrayHeader (Fortran)	matGetVariableInfo (Fortran)
matGetFull (Fortran)	matGetVariable (Fortran) followed by mxGetM (Fortran), mxGetN (Fortran), mxGetPr (Fortran), mxGetPi (Fortran)
matGetMatrix (Fortran)	matGetVariable (Fortran)
matGetNextArray (Fortran)	matGetNextVariable (Fortran)
matGetNextArrayHeader (Fortran)	matGetNextVariableInfo (Fortran)
matGetNextMatrix (Fortran)	matGetNextVariable (Fortran)
matGetString (Fortran)	matGetVariable (Fortran) followed by mxGetString (Fortran)
matPutArray (Fortran)	matPutVariable (Fortran)
matPutArrayAsGlobal (Fortran)	matPutVariableAsGlobal (Fortran)
matPutFull (Fortran)	mxCreateDoubleMatrix (Fortran) followed by mxSetPr (Fortran), mxSetPi (Fortran), matPutVariable (Fortran)

Obsolete Function	Replacement
matPutMatrix (Fortran)	matPutVariable (Fortran)
matPutString (Fortran)	mxCreateString (Fortran) followed by matPutVariable (Fortran)

Obsolete Functions: MX Array Manipulation (Fortran)

Obsolete Function	Replacement
mxClearLogical (Fortran)	None
mxCreateFull (Fortran)	mxCreateDoubleMatrix(Fortran)
mxCreateScalarDouble (Fortran)	mxCreateDoubleScalar(Fortran)
mxFreeMatrix (Fortran)	mxDestroyArray(Fortran)
mxGetName (Fortran)	matGetVariable (Fortran), mexGetVariable (Fortran), or engGetVariable (Fortran)
mxIsFull (Fortran)	mxIsSparse(Fortran)
mxIsString (Fortran)	mxIsChar(Fortran)
mxSetLogical (Fortran)	None
mxSetName (Fortran)	matPutVariable (Fortran), mexPutVariable (Fortran), or engPutVariable (Fortran)

Obsolete Functions: MEX-Files (Fortran)

Obsolete Function	Replacement
mexGetArray (Fortran)	mexGetVariable (Fortran)
mexGetArrayPtr (Fortran)	mexGetVariablePtr (Fortran)
mexGetEps (Fortran)	mxGetEps (Fortran)
mexGetFull (Fortran)	mexGetVariable (Fortran) followed by mxGetM (Fortran), mxGetN (Fortran), mxGetPr (Fortran), mxGetPi (Fortran)
mexGetGlobal (Fortran)	mexGetVariablePtr (Fortran)
mexGetInf (Fortran)	mxGetInf (Fortran)
mexGetMatrix (Fortran)	mexGetVariable (Fortran)
mexGetMatrixPtr (Fortran)	mexGetVariablePtr (Fortran)
mexGetNaN (Fortran)	mxGetNaN (Fortran)
mexIsFinite (Fortran)	mxIsFinite (Fortran)
mexIsInf (Fortran)	mxIsInf (Fortran)
mexIsNaN (Fortran)	mxIsNaN (Fortran)
mexPutArray (Fortran)	mexPutVariable (Fortran)
mexPutFull (Fortran)	mxCreateDoubleMatrix (Fortran) followed by mxSetPr (Fortran), mxSetPi (Fortran), mexPutVariable (Fortran)
mexPutMatrix (Fortran)	mexPutVariable (Fortran)

Obsolete Functions: MATLAB Engine (Fortran)

Obsolete Function	Replacement
engGetArray (Fortran)	engGetVariable (Fortran)
engGetFull (Fortran)	engGetVariable (Fortran) followed by mxGetM (Fortran), mxGetN (Fortran), mxGetPr (Fortran), mxGetPi (Fortran)
engGetMatrix (Fortran)	engGetVariable (Fortran)
engPutArray (Fortran)	engPutVariable (Fortran)
engPutFull (Fortran)	mxCreateDoubleMatrix (Fortran) followed by mxSetPr (Fortran), mxSetPi (Fortran), engPutVariable (Fortran)
engPutMatrix (Fortran)	engPutVariable (Fortran)

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 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 Runtime 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 [Date Data Type](#) for more information.

Dynamic Linking of External Libraries

MATLAB supports dynamic linking of external libraries only on 32-bit MS-Windows systems and 32-bit Linux systems. See [MATLAB Interface to Generic DLLs](#) for more information.

Version 7.1 (R14SP3) MATLAB

This table summarizes what's new in Version 7.1 (R14SP3):

New Features and Changes	Version Compatibility Considerations	Fixed Bugs and Known Problems	Related Documentation at Web Site
Yes Details below	Yes—Details labeled as Compatibility Considerations in descriptions of new features and changes. See also Summary.	Bug Reports at Web site	No

New features and changes introduced in this version are organized by these areas:

- Desktop Tools and Development Environment, MATLAB Version 7.1 (R14SP3)
- Mathematics, MATLAB Version 7.1 (R14SP3)
- Data Analysis, MATLAB Version 7.1 (R14SP3)
- Programming, MATLAB Version 7.1 (R14SP3)
- Graphics and 3-D Visualization, MATLAB Version 7.1 (R14SP3)
- Creating Graphical User Interfaces (GUIs), MATLAB Version 7.1 (R14SP3)
- External Interfaces/API, MATLAB Version 7.1 (R14SP3)

Desktop Tools and Development Environment, MATLAB Version 7.1 (R14SP3)

New features and changes introduced in this version are organized by these topics:

- Startup and Shutdown
- Desktop
- Running Functions—Command Window and Command History
- Help
- Workspace, Search Path, and File Operations
- Editing and Debugging M-Files
- Tuning and Managing M-Files
- Publishing Results

Startup and Shutdown

New features and changes introduced in this version are described here.

Windows -nodesktop No Longer has Menu Bar and Toolbar; Use Function Equivalents Instead

The behavior of MATLAB when started on a Windows platform with 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.

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`.

Desktop

New features and changes introduced in this version are organized by these topics:

- Arranging Windows and Documents
- Preferences Directory Added for R14SP3; Supplements R14 Directory
- Preferences Changes for Fonts, Hyperlinks, and -nodesktop
- info.xml File Automatic Validation; Shows Warnings for Invalid Constructs
- Other Desktop Changes

Arranging Windows and Documents

Figure Windows Now Dockable on Macintosh. On 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 **Escape** 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, 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 could 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 will only be used by R14SP3. This means that changes made to files in the directory while running R14SP3 will not be 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 will not be seen 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 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 MATLAB workspace.

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
- Tab Completion No Longer Shows Entries Twice
- Incremental Search Now Supports Removing Characters
- Hyperlink Color Preference Moved

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 a 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 `pl0` does exist in the file.

Hyperlink Color Preference Moved

The preference for specifying the hyperlink color has moved from the Command Windows preference pane to the **Colors** preference 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 Preference Moved
- New Look for Demos, Including Thumbnails and Categories
- Demos Run in Command Window as Scripts and Their Variables Now Created in Base Workspace
- echodemo Function Added to Replace playshow function
- Add Demos to Favorites
- Adding Your Own Demos Type Tag Now Supported
- Bug Reporting System Introduced

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** preference 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 be 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

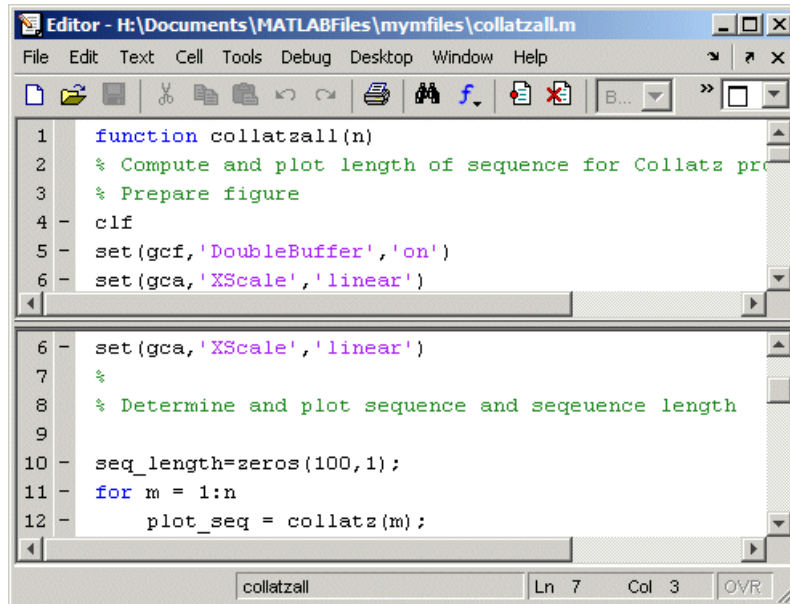
New features and changes introduced in this version are

- Split Screen Display Added
- Highlight Current Line Added
- Comment Lines in Java and C/C++ Files Now Supported
- HTML File Indenting Feature Added as the Default
- Incremental Search Now Supports Removing Characters
- Emacs Key Binding for Select All
- Change Case Added to Menu
- Nested Function Name No Longer in Status Bar

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.



The screenshot shows the MATLAB Editor window with a top/bottom split view. The top pane displays lines 1 through 6 of the script, and the bottom pane displays lines 6 through 12. The script content is as follows:

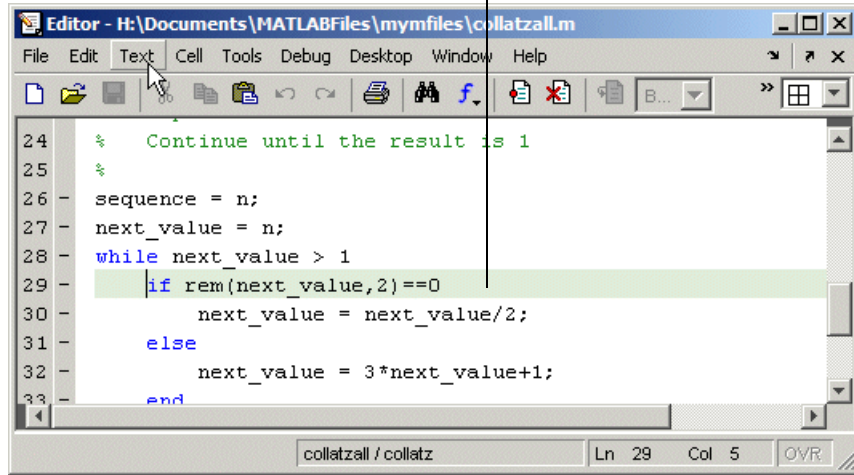
```
1 function collatzall(n)
2 % Compute and plot length of sequence for Collatz pro
3 % Prepare figure
4 clf
5 set(gcf,'DoubleBuffer','on')
6 set(gca,'XScale','linear')
6 set(gca,'XScale','linear')
7 %
8 % Determine and plot sequence and sequence length
9
10 seq_length=zeros(100,1);
11 for m = 1:n
12     plot_seq = collatz(m);
```

The status bar at the bottom of the window indicates the file name 'collatzall', the current line 'Ln 7', the current column 'Col 3', and the view mode 'OVR'.

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.

Current line (line with the caret/blinking cursor) is highlighted.



Comment Lines in Java and C/C++ Files Now Supported

You can now use the **Text -> Comment** feature to comment selected lines in Java and C/C++ files. This adds the // symbols at the front of the selected lines. Similarly, **Text -> Uncomment** removes the // symbols from the front of selected lines in Java and C/C++ 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 Word information from the Windows system registry 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 Microsoft 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 Windows system registry.

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.

Word Versions Supported by Notebook; Word 97 No Longer Supported

MATLAB Notebook supports Word versions 2000, and supports Word 2002 and 2003, both for XP.

Compatibility Considerations. As of MATLAB 7.1 (R14SP3), Notebook no longer supports Microsoft Word 97.

Mathematics, MATLAB Version 7.1 (R14SP3)

New features and changes introduced in this version are organized by these topics:

- New Functions
- Modified Functions
- Changes to `accumarray`
- Imposing Nonnegativity Constraints on Computed ODE Solution
- Mersenne Twister Support in `rand`
- `svd` Returns Economy Decomposition
- New Location for LAPACK Libraries
- Documentation on Data Analysis

New Functions

The following functions are new in R14SP3:

Function	Description
<code>hypot</code>	Square root of sum of squares
<code>mode</code>	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 “Potential Conflict with Function 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
<code>accumarray</code>	Allows more flexibility for input/output classes and functions to be called
<code>odeset</code>	New <code>NonNegative</code> integration property to impose nonnegativity constraints on an ODE solution
<code>rand</code>	Supports the Mersenne Twister algorithm in generating random numbers
<code>svd</code>	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 “Example: Computing Nonnegative Solutions” under “Differential Equations” 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/libmwlpack.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 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, MATLAB Version 7.1 (R14SP3)

New features and changes introduced in this version are described here:

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 time series from MATLAB 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, MATLAB Version 7.1 (R14SP3)

New features and changes are organized by these topics:

- New Functions
- Modified Functions
- Evaluation Functions for Arrays, Structures, Cells
- Using who and whos with Nested Functions
- Date and Time Functions Support Milliseconds
- Stack Trace Provided for lasterror
- isfield Function Supports Cell Arrays; Results Might Differ from Previous Version
- Support for Reading EXIF Data from Image Files
- Performance Improvements to the MATLAB JIT/ Accelerator on Macintosh
- Specifying fread Precision as Number of Bits
- Seconds Field Now Truncated; Results Might Differ
- Built-in Functions No Longer Use .bi; Impacts Output of which Function
- New Warning About Potential Naming Conflict

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 “Potential Conflict with Function 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
<code>cellfun</code>	Applies a given function to each cell of a cell array
<code>datestr</code>	Seconds field truncates instead of rounding
<code>error</code>	Saves stack information that you can retrieve using <code>lasterror</code>
<code>isfield</code>	Supports cell array input
<code>lasterror</code>	Returns stack information on last error
<code>rethrow</code>	Accepts stack information as input
<code>who</code> , <code>whos</code>	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 the reference page for `who` 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 fieldnames. 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, MATLAB Version 7.1 (R14SP3)

This version introduces the following new features and changes:

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), MATLAB Version 7.1 (R14SP3)

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
<code>clruprop</code>	Future version	<code>rmappdata</code>
<code>ctlpanel</code>	Future version	<code>guide</code>
<code>extent</code>	Future version	<code>get(txtobj, 'extent')</code>
<code>figflag</code>	Future version	<code>findobj</code> to determine if figure exists. <code>figure(fighandle)</code> to bring figure to front and give it focus.
<code>getuprop</code>	Future version	<code>getappdata</code>
<code>hthelp</code>	Future version	<code>web</code>
<code>layout</code>	Future version	None provided
<code>matq2ws</code>	Future version	None provided
<code>matqdlg</code>	Future version	None provided
<code>matqparse</code>	Future version	None provided
<code>matqueue</code>	Future version	None provided
<code>menuedit</code>	Future version	<code>guide</code>

Obsolete Function	Removed from Version	Replacement
menulabel	Future version	Use '&' to specify mnemonics and 'Accelerator' property to define accelerator keys.
setupprop	Future version	setappdata
wizard	Future version	None provided
ws2matq	Future version	None provided

External Interfaces/API, MATLAB Version 7.1 (R14SP3)

New features and changes introduced in this version are:

- mex Switches Now Supported on Windows
- New COM Programmatic Identifier
- New File Extension for MEX-Files on Windows
- New Preferences Directory and MEX Options
- Compiler Support
- Import Libraries Moved
- MEX Perl Script Moved
- Linking to System Libraries
- Linking to System Libraries

mex Switches Now Supported on Windows

MATLAB now supports the `-l` and `-L` options to the `mex` command on Windows. In previous releases of MATLAB, these options were supported only on UNIX.

Switch	Description
<code>-l</code>	<p>Specifies additional libraries to link against.</p> <hr/> <p>Note On Windows, the <code>-l</code> option can specify libraries of two forms. For example, specifying <code>-l name</code> matches either <code>name.lib</code> or <code>libname.lib</code>, whereas on UNIX it matches only <code>libname.lib</code>.</p> <hr/>
<code>-L</code>	<p>Specifies a path to use when MATLAB searches for library files specified with the <code>-l</code> option. The <code>-L</code> option must precede the <code>-l</code> option.</p>

For the switches you can use with the `mex` command, see the MEX Script Switches table in the “Custom Building MEX-Files” section of the “Calling C

and Fortran Programs from MATLAB” chapter of *MATLAB External Interfaces*.

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

MATLAB now uses the extension `.mexw32` for MEX-files on 32-bit versions of Windows. In previous versions, MATLAB used the extension `.dll`.

The 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 M-files 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, 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, 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

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 and `mexext.sh` on UNIX. 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 when Microsoft is the compiler vendor:

```
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 95. 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; see “New File Extension for MEX-Files on Windows”.

Compiler Support

The set of compilers that MATLAB supports has changed in MATLAB 7.1. For a complete, up-to-date list of supported compilers, see the following location on the Web:

<http://www.mathworks.com/support/tech-notes/1600/1601.shtml>

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')');
```

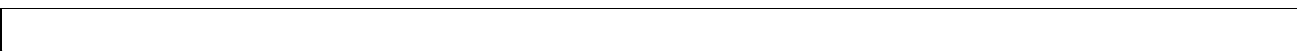

Version 7.0.4 (R14SP2) MATLAB

This table summarizes what's new in Version 7.0.4 (R14SP2):

New Features and Changes	Version Compatibility Considerations	Fixed Bugs and Known Problems	Related Documentation at Web Site
Yes Details below	Yes—Details labeled as Compatibility Considerations in descriptions of new features and changes. See also Summary.	Bug Reports at Web site	No

New features and changes introduced in this version are organized by these areas:

- Desktop Tools and Development Environment, MATLAB Version 7.0.4 (R14SP2)
- Mathematics, MATLAB Version 7.0.4 (R14SP2)
- Programming, MATLAB Version 7.0.4 (R14SP2)
- Graphics and 3-D Visualization, MATLAB Version 7.0.4 (R14SP2)
- Creating Graphical User Interfaces (GUIs), MATLAB Version 7.0.4 (R14SP2)
- External Interfaces/API, MATLAB Version 7.0.4 (R14SP2)



Desktop Tools and Development Environment, MATLAB Version 7.0.4 (R14SP2)

New features and changes are organized by these topics:

- Installation Folder with Spaces
- Startup and Shutdown
- Desktop
- Running Functions—Command Window and History
- Help
- Workspace, Search Path, and File Operations
- Editing and Debugging M-Files
- Source Control Interface
- Publishing Results

Installation Folder with Spaces

In MATLAB 7.0.4 (R14SP2) the following two changes have been made to the MathWorks Installer on Windows systems:

- The Installer now allows a folder name with spaces in the installation path.
- The Installer honors the Windows default installation folder, 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 in the desktop documentation.

JVM Updated

MATLAB is now using Java (JVM) 1.5 on Windows, Linux (32-bit), and Solaris platforms. Java 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 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.)
- View brief Simulink model descriptions in the **Description** column when **Show M and MDL file descriptions** is selected.
- View the complete Simulink model description 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 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 Simulink or Stateflow® files, use the Current Directory browser.

For a summary of the process, see the topic “Source Control Interface on Windows Platforms” 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 Consideration. 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 Word 97 to be Discontinued

Notebook will no longer support Microsoft Word 97 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, MATLAB Version 7.0.4 (R14SP2)

This version introduces the following new features and changes:

- New Vendor BLAS Used for Linear Algebra in MATLAB
- `max` and `min` on Complex Integers Not Supported

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, MATLAB Version 7.0.4 (R14SP2)

This version introduces the following new features and changes:

- Memory-Mapping
- textscan Enhancements
- xlsread Enhancements
- xlsread Imported Date Format Changes
- format Options Added
- Nonscalar Arrays of Function Handles to Become Invalid
- Assigning Nonstructure Variables As Structures Displays Warning

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)
```

See [Example 5 — Passing a Function Handle on 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. See “[Handling Excel Date Values](#)” on the function reference for `xlsread` for information on how to do this.

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” 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);
```


Graphics and 3-D Visualization, MATLAB Version 7.0.4 (R14SP2)

This version introduces the following new feature:

imwrite Now Supports GIF Export

The `imwrite` function now supports exporting image data in Graphics Interchange Format (GIF).

Compatibility Considerations

The MATLAB 7.0.4 graphics features have the the following platform limitations:

Cannot Dock Figures on Macintosh

You cannot dock figures in the Desktop, because MATLAB uses native figure windows on the Macintosh platform.

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.

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 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.

Creating Graphical User Interfaces (GUIs), MATLAB Version 7.0.4 (R14SP2)

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, MATLAB Version 7.0.4 (R14SP2)

New features and changes introduced in this version are described here.

New File Archiving Functions and Functionality

In addition to being able to zip and unzip compressed file archives, MATLAB 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.

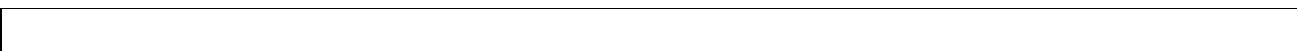
Version 7.0.1 (R14SP1) MATLAB

This table summarizes what's new in Version 7.0.1 (R14SP1):

New Features and Changes	Version Compatibility Considerations	Fixed Bugs and Known Problems	Related Documentation at Web Site
Yes Details below	Yes—Details labeled as Compatibility Considerations in descriptions of new features and changes. See also Summary.	Fixed bugs	No

New features and changes introduced in this version are organized by these areas:

- Desktop Tools and Development Environment, MATLAB Version 7.0.1 (R14SP1)
- Mathematics, MATLAB Version 7.0.1 (R14SP1)
- Programming, MATLAB Version 7.0.1 (R14SP1)
- Graphics, MATLAB Version 7.0.1 (R14SP1)
- Creating Graphical User Interfaces (GUIs), MATLAB Version 7.0.1 (R14SP1)
- External Interfaces/API, MATLAB Version 7.0.1 (R14SP1)



Desktop Tools and Development Environment, MATLAB Version 7.0.1 (R14SP1)

New features and changes are organized by these topics:

- Startup and Shutdown
- Desktop
- Running Functions—Command Window and Command History
- Help
- Workspace, Search Path, and File Operations
- Editing and Debugging M-Files
- Source Control Interface
- Publishing Results

Startup and Shutdown

Constructing Java Classpath Now Uses `librarypath`

When MATLAB starts, it now uses `librarypath.txt` as well as `classpath.txt` to construct the Java classpath.

Compatibility Considerations. If you call Java from MATLAB, refer to [Locating Native Method Libraries](#) in the MATLAB External Interfaces documentation for details. This change was part of MATLAB 7.0.

Desktop

System Web Browser Used for Large Files

The MATLAB Web browser displays files up to 1.5MB. When the Web browser tries to open a file greater than 1.5MB, MATLAB instead automatically displays the file in the system default browser.

Keyboard Access Added for More Desktop Tools

Additional desktop tools provide keyboard access to toolbar buttons and fields via mnemonics. For example, **Alt+K** moves the cursor to the **Stack** field in the Editor/Debugger toolbar. In the Profiler, the **R** in the **Run this code** toolbar field is underlined, indicating that **Alt+R** moves the cursor to this field. You

might need to hold down the **Alt** key while the tool is selected in order to see the mnemonics on the menus and buttons.

Macintosh Menus

On the Macintosh, the location of MATLAB menus has changed. In this version, MATLAB menus are not located at the top of the screen with other Macintosh screen menus. Instead, MATLAB menus appear within the MATLAB desktop and MATLAB tools. This change was made because of a problem with screen menus that caused MATLAB on the Macintosh to crash.

Running Functions – Command Window and Command History

Preferences for Parentheses Matching Added

There are now Command Window preferences you can set to perform parenthesis matching. Select **File -> Preferences -> Command Window -> Keyboard and Indenting** to set them. With the preference **Match parentheses while typing** selected, when you type a parenthesis or another delimiter, MATLAB highlights the matched parenthesis or other delimiter in the pair. With the preference **Match parentheses on arrow key or mouse movement** selected, when you move over a parenthesis or another delimiter, MATLAB highlights the matched parenthesis or other delimiter in the pair. MATLAB also alerts you to mismatches. These preferences also allow you to specify how MATLAB notifies you of matches and mismatches.

Clear Command Window Now Available from Context Menu

You can now select **Clear Command Window** from the context menu in the Command Window. A confirmation dialog box does not appear and the Command Window clears immediately. If you want a confirmation dialog box to appear before the Command Window clears, use **Edit -> Clear Command Window** instead.

End Key Behavior Changes

With the **Display** preference for **Wrap lines** selected, pressing **End** moves the cursor to the end of the current statement. When the **Command line key bindings** preference is set to **Emacs (MATLAB standard)**, you can also do this using **Ctrl+E**. In the previous version, **End** (and **Ctrl+E**) moved the cursor to the end of the current line.

Help

Own Help Files Now Allow Special Icons

When you supply your own help files, you can now specify the type of icon that appears in the Help browser **Contents** pane via the `<help_contents_icon>` tag in the `info.xml` file. For details, see “Adding Your Own Help Files in the Help Browser”.

Workspace, Search Path, and File Operations

Array Editor: F2 Keyboard Shortcut Added to Edit Current Element

A keyboard shortcut to use instead of double-clicking a cell is **F2** (or **Ctrl+U** on Macintosh), which allows you to edit the current element, positioning the cursor at the end of the element.

Array Editor: Single Quotation Marks Now Supplied for Strings During Paste from Excel

When you copy data from Microsoft Excel and paste it into a cell array in the Array Editor using the menu item **Edit -> Paste Excel Data**, MATLAB assumes the values are strings and automatically supplies the single quotation marks if they cannot be interpreted as numeric values.

Current Directory Browser Comments Now Include First Line

When you select the Current Directory preference **Show M-file comments and MAT-file contents**, the help shown now includes the first comment line (also called the H1 line).

Current Directory Browser Auto-Refresh Rate Now Specifiable

The Current Directory browser preference for auto-refresh now allows you to specify the update time. By default, every 2 seconds the Current Directory browser checks for and reflects any changes you made to files and directories in the current directory using other applications.

In some cases when the current directory is on a network and the Current Directory browser is open, MATLAB becomes slow because of the auto-refresh feature. If you experience general slowness in MATLAB and have the Current Directory browser open, increase the default update time to improve responsiveness. If increases do not alleviate the slowness enough, clear the

check box in preferences to turn auto-refresh off. Then you can manually refresh the display selecting **Refresh** from the context menu in the Current Directory browser.

Find Files Field Now Uses Selected Text

You can now select text in the Command Window or Editor and the **Find Files** dialog box enters that text in its **Find files containing text** field.

Editing and Debugging M-Files

Breakpoints Supported in Anonymous Functions

The Editor/Debugger supports breakpoints in anonymous functions. Lines containing anonymous functions can have more than one breakpoint in a line: one for the start of the line and one for each anonymous function in the line. A line that contains multiple breakpoints has a blue breakpoint icon.

dbstatus Supports Anonymous and Nested Functions

The `dbstatus` function now supports anonymous and nested functions, including a new `'-completenames'` argument. Running `dbstatus(' -completenames')` displays, for each breakpoint, the absolute filename and the sequence of functions that nest the function containing the breakpoint.

Colors Now Maintained when Copying From Editor

When you paste a selection from the Editor into another application, such as Word, the Editor now maintains the syntax highlighting colors in the file in the other application. MATLAB pastes the selection to the clipboard in RTF format, which many Windows and Macintosh applications support.

Open Selection Now Works for Current Cursor Position

In an M-file, position the cursor within a subfunction, function, file, variable, or Simulink model, and press **Ctrl+D** (or right-click and select **Open Selection**). The item opens in the appropriate tool. In the previous version of MATLAB, you had to select the complete name in the M-file to use this feature. See “Opening a Selection in an M-File” for more information.

Source Control Interface

verctrl Function Does Not Support Handle

The `verctrl` function, available for Windows platforms only, was documented incorrectly. The documentation stated that you could create a handle, and showed the handle argument in the function syntax. You cannot create a handle, but must instead use a value of 0 for that field.

Publishing Results

Notebook Causes MATLAB to Become Automation Server

If you run Notebook from MATLAB and MATLAB is not an automation server, MATLAB will become an automation server. This is a change from Release 14, where MATLAB spawned a second instance that was an automation server.

Notebook Now Supports Office 2003

Notebook now supports Office 2003 (for XP); it is one of the notebook -setup options.

Notebook Support for Word 97 to Be Discontinued

Word 97 is supported in this release, but will not be supported in future releases.

Compatibility Considerations. If you use Word 97 with Notebook, move to a newer version of Word before moving to the next version of MATLAB.

Mathematics, MATLAB Version 7.0.1 (R14SP1)

This version introduces the following new features and changes:

- New Function — `ordeig`
- More Functions Accept Single-Precision Data Inputs
- New Vendor BLAS Used for Linear Algebra in MATLAB
- Overriding the Default BLAS Library on Sun/Solaris Systems
- FDLIBM Version Upgraded
- Different Results When Solving Singular Linear Systems on Intel Systems; Inconsistent NaN Propagation
- `funm` Returns Status Information; New Output Might Result In Error

New Function — `ordeig`

The new function `ordeig` takes a quasitriangular matrix T or matrix pair (A, B) and returns the vector of eigenvalues in the same order that they appear down the diagonal of T or (A, B) . You can use `ordeig` with the functions `ordschur` and `ordqz`, which reorder the eigenvalues of a Schur factorization or a QZ factorization, respectively.

More Functions Accept Single-Precision Data Inputs

More MATLAB functions now accept single-precision data inputs in addition to the usual double-precision inputs. To determine whether a function works on single precision inputs, look for the `Class support` line in the M-file help for the function. For example, to determine whether the function `mean` accepts single-precision inputs, type

```
help mean
```

The `Class support` line is

```
float: double, single
```

which tells you that `mean` does accept single-precision inputs.

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 `\`.

On some platforms, MATLAB continues to use ATLAS BLAS.

Starting in Release 14, MATLAB 7.0 uses vendor BLAS from the `vecLib` library on the Mac.

Starting in Release 14 with Service Pack 1, MATLAB 7.0.1 uses vendor BLAS from

- The Intel® Math Kernel Library (MKL) Version 7.0 on Intel chips running both Windows and Linux. See the MATLAB 7.0 Release Notes for how to use the multi-threaded capabilities of MKL.
- The AMD Core Math Library (ACML) Version 2.0 library on AMD chips, native 64 bit application

Overriding the Default BLAS Library on Sun/Solaris Systems

MATLAB uses the Basic Linear Algebra Subroutines (BLAS) libraries to speed up matrix multiplication and LAPACK-based functions like `eig`, `svd`, and `\(mldivide)`. At start-up, MATLAB selects the BLAS library to use.

For Release 14 with Service Pack 1, MATLAB still uses the ATLAS BLAS libraries on the Sun Microsystems Solaris Operating System. However, you can switch the BLAS library that MATLAB uses to the Sun Performance Library (Sunperf) BLAS, provided by Sun Microsystems.

If you want to take advantage of the potential performance enhancements provided by the Sun BLAS, you can set the value of the environment variable `BLAS_VERSION` to the name of the Sun Performance Library, `libsunperf.so.4`. MATLAB uses the BLAS specified by this environment variable, if it exists.

To set the `BLAS_VERSION` environment variable, enter the following command at the UNIX prompt.

```
% setenv BLAS_VERSION libsunperf.so.4
```

Then start MATLAB as usual.

To get visual feedback that the BLAS version has changed, also type at the UNIX prompt

```
% setenv LAPACK_VERBOSITY 1
```

before starting MATLAB. This will display diagnostic information while MATLAB is starting up, for example:

```
cpu_id: sun4u
libmwapack: loading libsunperf.so.4
libmwapack: loading lapack.so
```

FDLIBM Version Upgraded

In Release 14, MATLAB used FDLIBM Version 5.2. In R14sp1, MATLAB has been upgraded to use FDLIBM Version 5.3.

Different Results When Solving Singular Linear Systems on Intel Systems; Inconsistent NaN Propagation

In previous releases, when you solved n -by- n linear systems $Ax=b$ using $x = A \setminus b$, where A is singular or contains NaN, the computed result x often contained NaN. In Version 7.0.1, the same command might return 0 in x , due to the way the Intel® Math Kernel Library (MKL) implementation of the BLAS handles this operation.

Compatibility Considerations

Code that relies on the result containing NaN should check for the following warnings instead:

- For singular A, an existing warning is issued.

```
x = [1 2; 0 0] \ [1; 0]
Warning: Matrix is singular to working precision.
x =
     1
     0
```

- For A that contains NaN, a new warning message is issued.

```
x = [1 2; 0 NaN] \ [1; 0]
Warning: Matrix is singular, close to singular or badly scaled.
Results may be inaccurate. RCOND = NaN.
x =
     1
     0
```

funm Returns Status Information; New Output Might Result In Error

Prior to Release 14, the second output of the function `funm` was an error estimate that was sometimes inaccurate. In Release 14, Version 7.0, the second output was replaced by an exit flag that indicates whether the computation was successful.

Compatibility Considerations

Code that was created prior to Release 14 and that uses the second output of `funm`, might not work correctly in Version 7.0 or later.

Programming, MATLAB Version 7.0.1 (R14SP1)

This version introduces the following new features and changes:

- Character Set Conversion Functions Added
- `datevec` Support of Empty String Argument
- `deffun` Function Supports New Options
- `ftell` Returning Invalid Position in Rare Cases
- `fwrite` Saves `uint64` and `int64` Types
- `mat2str` Enhanced to Work with Non-double Types
- `nargin`, `nargout` Operate on Function Handles
- `regexprep` Now Supports Character Representations in Replacement String
- Logical OR Operator `|` in `regexp` Expressions Might Yield Different Results from Previous Version
- Multiple Declarations of Persistent Variables No Longer Supported

Character Set Conversion Functions Added

Unicode is becoming the preferred internal presentation of characters in MATLAB. For example, MATLAB functions such as `disp` require an input string in Unicode to display properly. To facilitate the use of different character sets, MATLAB provides two new functions to convert characters from a native character set to Unicode and back.

The `native2unicode` function converts from either a default, or user specified, native character set to Unicode. The `unicode2native` function does the opposite, converting from Unicode to either a default, or user specified, native character set. Note that any MATLAB string containing only US-ASCII characters does not require any conversion.

Type `doc native2unicode` or `doc unicode2native` for more information on these functions.

`datevec` Support of Empty String Argument

For the purpose of backwards compatibility, invoking the command `datevec('')` now returns an empty vector.

Compatibility Considerations

This behavior was not intentional in previous versions of MATLAB, and it is subject to change in future releases.

depfun Function Supports New Options

The depfun function now supports these options:

Option	Description
'-all'	Computes all possible left-side arguments and displays the results in the report(s). Only the specified arguments are returned.
'-calltree'	Returns a call list in place of a called_from list. This is derived from the called_from list as an extra step.
'-expand'	Includes both indices and full paths in the call or called_from list.
'-print', 'file'	Prints a full report to file.
'-quiet'	Displays only error and warning messages, and not a summary report.
'-toponly'	Examines <i>only</i> the files listed explicitly as input arguments. It does not examine the files on which they depend.
'-verbose'	Outputs additional internal messages.

ftell Returning Invalid Position in Rare Cases

The `ftell` function is likely to return an invalid position when *all* of the following are true. This is due to the way in which the Microsoft Windows C library currently handles its `ftell` and `fgetpos` commands:

- The file you are currently operating on is an ASCII text file.
- The file was written on a UNIX-based system, or uses the UNIX-style line terminator: a line feed (with no carriage return) at the end of each line of text. (This is the default output format for MATLAB functions `dlmwrite` and `csvwrite`.)
- You are reading the file on a Windows system.
- You opened the file with the `fopen` function with mode set to `'rt'`.
- The `ftell` command is directly preceded by an `fgets` command.

Note that this does not affect the ability to accurately read from and write to this type of file from MATLAB.

Compatibility Considerations

This represents a change in behavior.

fwrite Saves uint64 and int64 Types

The `fwrite` function can now save `uint64` and `int64` values. Previously `fwrite` supported these data types only on DEC Alpha systems. Now, it works on all supported MATLAB platforms.

mat2str Enhanced to Work with Non-double Types

In MATLAB 7.0.1, you can use the `mat2str` function to convert nondouble data types to a string that represents the input value. Type `doc mat2str` for more information.

nargin, nargsout Operate on Function Handles

The `nargin` and `nargsout` functions now accept either a function name or function handle as an input argument. When called with a function handle, `nargin` and `nargsout` return the number of input or output arguments you can pass to or receive from the function that the handle maps to.

regexprep Now Supports Character Representations in Replacement String

The `regexprep` function now supports the use of character representations (e.g., `\t` for tab, `\n` for newline) in replacement strings. For example, the following `regexprep` command replaces the `|` character with two horizontal tabs:

```
str = 'Field 1 | Field 2 | Field 3';  
regexprep(str, '\\|', '\\t\\t')  
ans =  
    Field 1   Field 2   Field 3
```

In Version 6, the same command yielded the string

```
Field 1 \\t\\t Field 2 \\t\\t Field 3
```

Logical OR Operator | in regexp Expressions Might Yield Different Results from Previous Version

Be careful about using the logical OR (`|`) operator within square brackets (e.g., `[A|B]`) in regular expressions in MATLAB. The recommended way to match "the letter A or the letter B" in a MATLAB `regexp` expression is to use `[AB]`.

Compatibility Considerations

If you have used `[A|B]` for this purpose in earlier versions of MATLAB, you may get unexpected results when you run your code in version 7.0.

MATLAB versions 6.0 and 6.5 treat `|` as an ordinary character when it is used between square brackets. For example, these versions interpret the expression `[A|B]` as "match 'A', or match '|', or match 'B'." MATLAB 7.0 correctly gives precedence to the logical OR functionality of the `|` operator. Because of this change, MATLAB now interprets `[A|B]` as "match '[A', or match 'B]'."

You can avoid the effects of this bug fix altogether by using the recommended syntax `[AB]` for this type of operation. This syntax returns the correct results in all MATLAB versions.

The following example attempts to find the word Jill or Bill in the string 'My name is Bill'. The syntax used in the expression is incorrect, but `regexp` in MATLAB 6.5 finds a match anyway because of the software bug. This syntax does not work in version 7.0 or 7.0.1 because MATLAB now interprets the expression as the logical OR of the two statements, '[J]' and 'B]ill':

<pre> MATLAB 6.5 str = 'My name is Bill'; expr = '[J B]ill'; [s e] = regexp(str, expr); str(s:e) ans = Bill </pre>	<pre> MATLAB 7.0.1 str = 'My name is Bill'; expr = '[J B]ill'; [s e] = regexp(str, expr); str(s:e) ans = Empty string: 1-by-0 </pre>
--	--

Using the recommended syntax returns the correct results in all MATLAB versions:

```

str = 'My name is Bill';
expr = '[JB]ill';
[s e] = regexp(str, expr);
str(s:e)
ans =
    Bill

```

If you want to use `|` in an expression as an ordinary character, precede it with a backslash:

```

str = 'The | operator performs a logical OR';
expr = 'The [\$ \\| \#] operator';
[s e] = regexp(str, expr);
str(s:e)
ans =
    The | operator

```

Multiple Declarations of Persistent Variables No Longer Supported

You can no longer declare a variable as persistent more than once within a function.

Compatibility Considerations

If you do this, you will need to modify your code.

Graphics, MATLAB Version 7.0.1 (R14SP1)

This version introduces the following new feature:

OpenGL Trouble Shooting

The `opengl` command now enables you to switch from hardware to software-based OpenGL rendering. It also enables you to select various known bug workarounds. See the `opengl` reference page for more information.

Compatibility Considerations

Cannot Dock Figures on Macintosh

You cannot dock figures in the Desktop, because MATLAB uses native figure windows on the Macintosh platform.

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.

Not All Macintosh System Fonts Are Available

MATLAB figures do not support the same fonts as native Macintosh applications. Use the `uifont` functions to see which fonts are available in MATLAB.

Preview Java Figures on the Macintosh

You can preview the use of Java Figure on the Macintosh by starting MATLAB with the `-useJavaFigures` option.

Creating Graphical User Interfaces (GUIs), MATLAB Version 7.0.1 (R14SP1)

This version introduces the following new features and changes:

- FIG-File Format Change
- Panels, Button Groups, and ActiveX Components
- Comments Now Optional for Newly Generated Callback Functions
- Windows XP Display of Push and Toggle Buttons

FIG-File Format Change

GUI FIG-files that are created or modified with MATLAB 7.0 or a later MATLAB version are not automatically compatible with Version 6.5 and earlier versions.

- GUIs Saved from GUIDE or from the Command Line: You can check the **Ensure backward compatibility (-v6)** preference in the **Preferences** dialog box under **General -> MAT-Files**. When this preference is checked, all MAT-files are saved so as to be backward compatible with Version 6.5 and earlier versions.
- Alternative for GUIs Saved from the Command Line: If you do not want to check the **MAT-Files** preference described above, but want to make individual GUI FIG-files backward compatible, use the 'v6' argument when you save the GUI with the `hgsave` function.

Compatibility Considerations

To make FIG-files, which are a kind of MAT-file, backward compatible, you must explicitly specify that you want the backwards compatibility.

Panels, Button Groups, and ActiveX Components

Panels, button groups, and ActiveX components were introduced in MATLAB 7.0. These components are not compatible with versions earlier than 7.0.

Compatibility Considerations

You should not use these components in GUIs that you expect to run in earlier MATLAB versions.

You can export a GUI that contains a panel, button group, or ActiveX component from GUIDE to a single M-file that does not require a FIG-file. However, you will not be able to run that M-file in MATLAB versions earlier than 7.0.

Comments Now Optional for Newly Generated Callback Functions

In prior releases, GUIDE automatically generated comment lines for each callback that you added to an existing GUI M-file. For example:

```
% --- Executes during object deletion, before destroying properties.
function figure1_DeleteFcn(hObject, eventdata, handles)
% hObject    handle to figure1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

Comment lines are now optional for most callbacks. If you want the comments to be generated automatically when you add a callback, check the new preference **Add comments for newly generated callback functions** on the **GUIDE** panel of the **Preferences** dialog box. The factory default is checked.

If this preference is unchecked, GUIDE includes the comment lines only for callbacks that are automatically included for the GUIDE template you chose. No comments are included for any other callbacks that are added to the M-file.

Windows XP Display of Push and Toggle Buttons

Push buttons and toggle buttons with background colors other than the default display differently in Windows XP.

Compatibility Considerations

For Windows XP, GUI push buttons are displayed with a white background. If you have specified a background color other than the default, that color appears as a border around the push button. Unselected toggle buttons are displayed with the specified background color, but selected toggle buttons are displayed with a white background bordered by the background color.

External Interfaces/API, MATLAB Version 7.0.1 (R14SP1)

New features and changes introduced in this version are described here.

Function Handles in COM Event Callbacks

MATLAB now supports function handles as callbacks for ActiveX objects. This example passes a function handle that maps to `samplev` to `registerEvent`:

```
cd $matlabroot\toolbox\matlab\winfun
h = actxcontrol('mwsamp.mwsampctrl.2', [0 0 200 200]);
registerEvent(h, @samplev);           % Click the control.
```

Registering Events for COM Servers and Controls

With MATLAB 7.0.1, you can register events for COM servers as well as for COM controls.

Expanded Support for Web Services (SOAP and WSDL)

This version expands MATLAB support for Web services, that is, Simple Object Access Protocol (SOAP) and Web Services Description Language (WSDL).

These are some of the key enhancements:

- MATLAB now supports document style messages, in addition to the Remote Procedure Call (RPC) style supported in version 7.0.
- MATLAB preserves the case in method, class, and object names.
- Web services functions now decode results that use Base64 encoding.
- The `createClassFromWsd1` function now supports WSDL files that define multiple services.

Specifying the Search Path for Java Native Method DLLs

The mechanism that MATLAB uses to locate native method libraries that are required by Java has changed. MATLAB no longer uses system environment variables to define the paths to these libraries.

Compatibility Considerations

If you presently rely on the `PATH` (for Windows) or `LD_LIBRARY_PATH` (for UNIX) environment variables for this purpose, you will need to use the file `librarypath.txt`, as described below, in its place.

Specifying the Java Library Path

Java classes can dynamically load native methods using the Java method `java.lang.System.loadLibrary("LibFile")`. In order for the JVM to locate the specified library file, the directory containing it must be on the Java Library Path. This path is established when MATLAB launches the JVM at startup, and is based on the contents of the file

```
$matlab/toolbox/local/librarypath.txt
```

(where `$matlab` is the MATLAB root directory represented by the MATLAB keyword `matlabroot`).

You can augment the search path for native method libraries by editing the `librarypath.txt` file. Follow these guidelines when editing this file:

- Specify each new directory on a line by itself.
- Specify only the directory names, not the names of the DLL files. The `LoadLibrary` call does this for you.
- To simplify the specification of directories in cross-platform environments, you can use any of these macros: `$matlabroot`, `$arch`, and `$jre_home`.

MATLAB DDE Server Is Now Disabled By Default

To enable the DDE server start MATLAB with the `/Automation` option.

The outgoing MATLAB DDE commands (`ddeinit`, `ddeterm`, `ddeexec`, `ddereq`, `ddeadv`, `ddeunadv`, `ddepoke`) function normally without the MATLAB DDE server. See

<http://www.mathworks.com/support/solutions/data/1-Q4728.html?solution=1-Q4728> for more information.

Clearing MEX-Functions

The command `clear mex` now clears MEX-functions, but not M- and MEX-functions. Entering `clear mex` does not clear locked functions or functions that

are currently in use. It does however clear breakpoints and persistent variables.

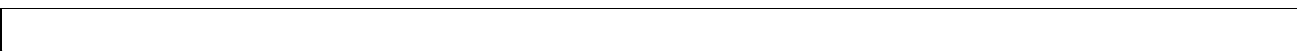
Version 7 (R14) MATLAB

This table summarizes what's new in Version 7 (R14):

New Features and Changes	Version Compatibility Considerations	Fixed Bugs and Known Problems	Related Documentation at Web Site
Yes Details below	Yes—Details labeled as Compatibility Considerations in descriptions of new features and changes. See also Summary.	Fixed bugs	No

New features and changes introduced in this version are organized by these areas:

- Desktop Tools and Development Environment, MATLAB Version 7 (R14)
- Mathematics, MATLAB Version 7 (R14)
- Programming, MATLAB Version 7 (R14)
- Graphics and 3-D Visualization, MATLAB Version 7 (R14)
- Creating Graphical User Interfaces (GUIs), MATLAB Version 7 (R14)
- External Interfaces/API, MATLAB Version 7 (R14)



Desktop Tools and Development Environment, MATLAB Version 7 (R14)

New features and changes introduced in this version are organized by these topics:

- Startup and Shutdown
- Desktop
- Running Functions—Command Window and Command History
- Help
- Workspace, Search Path, and File Operations
- Editing and Debugging M-Files
- Tuning and Managing M-Files
- Source Control Changes
- Publishing Results

Startup and Shutdown

JVM Version Updated

MATLAB is now using Java (JVM) 1.4.2.

Desktop

New features and changes introduced in this version are

- Demo of MATLAB Desktop Added
- Arranging Documents Supports New Options
- Finding Files and Content in Files with New Tools
- MATLAB Shortcuts to Easily Run Group of Statements
- MATLAB Web Browser Added
- Menu Changes
- Keyboard Shortcuts Added to Go to Tools and Documents
- Drag and Drop Supported Between Desktop Tools
- Arranging Columns in Tools
- Font and Color Preferences for Tools
- terminal Function Removed
- license Function Results Modified Slightly

Demo of MATLAB Desktop Added


If you are using the Help browser, watch the new Desktop and Command Window video demo for an overview of the major functionality.

Arranging Documents Supports New Options

The MATLAB desktop now provides you with new options for arranging the following types of documents:

- M-files and other files in the Editor/Debugger
- Arrays in the Array Editor
- Figure windows
- HTML documents in the MATLAB Web browser

You can dock these types of documents in the desktop, undock them from the desktop so each is in its own separate window, or group undocked documents together in their tool. You can now position the documents using these features: tile, left/right split, top/bottom split, floating, or maximized. Use the **Window** menu or toolbar icons to position documents.

Docking Tools and Documents. There are now dock buttons  in the menu bars of undocked tools and documents. Click a dock button to move the tool into the desktop, or to move the document into its tool.

Document Bar. There is now a Document Bar in tools that support documents that you use to go to open documents. It appears when there is more than one maximized document open in a tool. You can hide or move the Document Bar by selecting **Desktop -> Document Bar** menu options.

Saving Layouts. You can save desktop layouts. Select **Desktop -> Save Layout** and provide a name. To restore a saved layout, select **Desktop -> Desktop Layout -> name**.

Launch Pad Removed. The Launch Pad tool was removed. Use the **Start** button instead.

Adding Your Own Toolbox to Start Button. Add your own toolbox to the **Start** button. Select **Start -> Desktop Tools -> View Source Files**. Click **Help** in the resulting dialog box for details.

Finding Files and Content in Files with New Tools

Search for files and directories, as well as for content within files by selecting **Edit -> Find Files** from any desktop tool. For details, see “Finding Files and Content Within Files” in the online documentation.

MATLAB Shortcuts to Easily Run Group of Statements

You can create and run MATLAB shortcuts, where a shortcut is an easy way to run a group of MATLAB statements. A shortcut is like an M-file script, but unlike an M-file, a shortcut does not have to be on the MATLAB search path or in the current directory when you run it.

Create a shortcut by selecting **Start -> Shortcuts -> New Shortcut** and completing the dialog box. Run the shortcut from the **Start** button.

You can also create a shortcut by dragging selected statements to the shortcut toolbar. This adds the shortcut to the toolbar, from where you can then run it. For details, see “Shortcuts for MATLAB” in the online documentation.

MATLAB Web Browser Added

MATLAB now displays HTML documents it produces in a new desktop tool, the MATLAB Web browser. You can display HTML documents in this Web browser using the web function.

The web function now opens the MATLAB Web browser by default, instead of opening the MATLAB Help browser. Use the web function's `-helpbrowser` option to display files in the Help browser.

Menu Changes

Debug Menu Added. You can now access debugging features from the **Debug** menu of most desktop tools.

View, Window, and Desktop Menus. There is no longer a desktop **View** menu, although some tools still have a **View** menu. The **Window** menu in the desktop has changed. Use the new **Desktop** menu to select a layout, and to open and close tools. Use the **Window** menu to access open tools and documents, as well as to position documents. The menus and the menu items in the desktop change, depending on the current tool selected.

Web Menu Items Moved. The **Web** menu was removed. Access the items it contained from **Help -> Web Resources**.

Keyboard Shortcuts Added to Go to Tools and Documents

There is now a keyboard shortcut you can use to go to each tool and to each open document. For example, use **Ctrl+0** to go to the Command Window, and **Ctrl+Shift+0** to go to the most recently used Editor document. See the **Window** menu for the shortcuts to go to currently open tools and documents.

There have been some changes to the keyboard shortcuts you use with desktop tools. For example, **Ctrl+Tab** now moves you to the next open tool or group of tools tabbed together. In previous releases, **Ctrl+Tab** moved you to the next open document or tool. In MATLAB 7, use **Ctrl+Page Down** to move to the next open document or tool in a tabbed group. For the complete list, see "Keyboard Shortcuts" in the online documentation.

Drag and Drop Supported Between Desktop Tools

You can drag selected text or files between desktop tools. For example, you can

- Select text in the Editor and drag it to the Command Window, which cuts and pastes it into the Command Window. You can use **Ctrl** while dragging to copy selected text instead of just moving it.
- Select a file in the Current Directory browser and drag it to the Editor, which opens the file in the Editor.

You can also drag selected text or files between desktop tools and external tools and applications. For example, you can

- Select a MAT-file from the Microsoft Windows Explorer and drag it to the Command Window, which loads the data into the MATLAB workspace.
- Select text from a page displayed in a Netscape browser and drag it to a file in the Editor, which pastes the text into the file in the Editor.

Arranging Columns in Tools

In desktop tools that contain columns, you can drag a column to a new position. For example, this includes the Current Directory browser, and the Help browser **Index** and **Search** panes. Click a column head to sort by that column. For some tools, you can click again to reverse the sort order.

When a column is too narrow to show all the information in it, position the cursor over a long item in that column, and a tooltip displays showing the complete content of the item.

Font and Color Preferences for Tools

Access font and color preferences for all desktop tools in the **Fonts** and **Colors** preference panes. Select **File -> Preferences -> Fonts** or **File -> Preferences -> Colors**. For more information, click the **Help** button in the preferences dialog box, or see **Fonts, Colors, and Other Preferences** in the online documentation.

terminal Function Removed

The `terminal` function was removed.

Compatibility Considerations. If your code refers to the `terminal` function, you need to change it.

license Function Results Modified Slightly

The data returned by the `license` command is now sorted in alphabetical order and uses only lowercase characters.

Compatibility Considerations. If you rely on the data returned by `license`, be sure your code works properly with these changes.

Running Functions—Command Window and Command History

New features and changes introduced in this version are

- Demo of Command Window Features
- Tab Completion Graphical Interface Added; Removed Preference to Limit Completions
- Navigate in Command Window Using Arrow Keys Via New Preference
- Incremental Search Indicates Search Direction and is Now Case Sensitive
- `commandwindow` Function Added to Open or Select the Command Window
- Macintosh: Command +. Now Stops Execution
- Comment After Ellipsis Now Properly Highlighted
- Evaluate Selection from Context Menus No Longer Appends
- Syntax Highlighting Default Colors Modified
- Ctrl+C to Stop Execution is Now More Consistent
- Parentheses Matching Support Removed
- Command History Features

Demo of Command Window Features

If you are using the Help browser, watch the new Desktop and Command Window video demo for an overview of the major functionality.

Tab Completion Graphical Interface Added; Removed Preference to Limit Completions

Tab completion now has a graphical interface. For example, type `cos` and press the **Tab** key. A list of functions that begin with `cos` appears. Double-click the function you want and MATLAB completes the name in the Command Window. Alternatively, when the list of names appears, you can type the next

unique letter in the name, and the first name in the list that matches it is selected. Continue typing unique letters to select the name you want, and press **Enter**. Press **Escape** to clear the list without selecting a name.

With the new interface, there is no longer a preference allowing you to limit the number of tab completions that display. MATLAB always displays all possible completion.

Compatibility Considerations. If you relied on the preference to limit the number of tab completions MATLAB displays, type more characters before pressing **Tab** so fewer possible completions display.

Navigate in Command Window Using Arrow Keys Via New Preference

There is a new preference that allows you to use arrow keys to navigate in the Command Window instead of recalling history.

Incremental Search Indicates Search Direction and is Now Case Sensitive

The incremental search interface now indicates the search direction. It is also case-sensitive when you enter uppercase letters in the search field.

commandwindow Function Added to Open or Select the Command Window

Use the new `commandwindow` function to open the Command Window when it is closed. For example, use this function in an M-file. Or if the Command Window is already open, use the function to select the Command Window, making it the active window.

Macintosh: Command +. Now Stops Execution

On Macintosh platforms, you can now use **Command+.** (**Command** key and period key) to stop execution of a running program.

Comment After Ellipsis Now Properly Highlighted

When you include an ellipsis in a statement so that you can continue the statement on the next line, any text you type after the `...` on the same line is considered to be a MATLAB comment and now is syntax highlighted as a comment. In previous releases, the syntax highlighting did not indicate the text after the `...` as a comment.

Evaluate Selection from Context Menus No Longer Appends

Evaluate selection, available from context menus for various tools, no longer appends the selection to the statement at the prompt, but instead runs the selection. Make a selection and press **Enter** or **Return** to append the selection to the statement at the prompt and execute it.

Syntax Highlighting Default Colors Modified

The default colors for syntax highlighting have been modified. Unterminated strings are now maroon, while terminated strings are now purple. This is the opposite of previous versions. Maroon is considered to be more of an “alerting” color, resembling the default of red for errors, which is the reason for the change. If you prefer the colors used in previous versions, change them using preferences—see Syntax Highlighting Colors in the online documentation.

In addition, arguments in statements entered using command syntax rather than function syntax are highlighted as strings, emphasizing that variables in command syntax are passed as literal strings rather than as their values.

Ctrl+C to Stop Execution is Now More Consistent

Stopping execution using **Ctrl+C** (^C) has changed. Windows and UNIX platforms now respond similarly to **Ctrl+C**, and in general, stop execution without the need for pause or drawnow statements in your M-files. For M-files that run for a long time, or that call built-ins or MEX-files that take a long time, **Ctrl+C** does not always effectively stop execution. In that event, include a drawnow command in your M-file, for example, within a large loop. **Ctrl+C** might be less responsive if you started MATLAB with the -nodesktop option.

Parentheses Matching Support Removed

Matching parentheses in the Command Window is not supported in this version.

Compatibility Considerations. This feature was available in the Command Window in previous versions but you cannot use it in this version.

Command History Features

Demo of Command History Features. If you are using the Help browser, watch the new Command history video demo for an overview of the major functionality.

Syntax Highlighting in Command History. Entries in the Command History tool now appear with syntax highlighting.

Tree View in Command History. Entries in the Command History now appear in a tree view so you can minimize the length of the visible history. The top level nodes of the tree are the dates/times for each session, and beneath that is the history for that session. Click the - to the left of a date/time to hide the history entries for that session. Click the + to the left of a date/time entry to show history entries for that session.

commandhistory Function Added to Go to Tool. Use the new commandhistory function to open the Command History when it is closed, and to select it when it is open.

Save Frequency Higher by Default. The default for saving the history has changed. Now, by default, MATLAB saves the history file after five statements have been added to the history. You can modify the frequency using Command History preferences.

Help

New features and changes introduced in this version are:

- Demo of New Help Browser Features
- Documentation Now Automatically Installed; Not Accessible from CD-ROMs and docroot Not Supported
- Index Pane Adds Alphabetical Links
- Search Type Removed
- Favorites in Help Browser
- Display Pane Find in Page Icon
- Title Field No Longer Supports Web Browsing
- docsearch Function Added to Execute Help Browser Search
- help Function Provides Help for Methods and Classes
- web Function Now Opens MATLAB Web Browser By Default
- HTML Documentation Not Viewable with -nojvm Startup Option

Demo of New Help Browser Features

If you are using the Help browser, watch the new Help and Documentation video demo for an overview of the major functionality.

Documentation Now Automatically Installed; Not Accessible from CD-ROMs and docroot Not Supported

Documentation is automatically installed for all the products you install. Documentation is no longer accessible from CD-ROMs. To access the documentation for products not installed on your system, use The MathWorks Web site,

<http://www.mathworks.com/access/helpdesk/help/helpdesk.shtml>.

Compatibility Considerations. Because of this change, the docroot function is no longer needed and will not be supported.

Index Pane Adds Alphabetical Links

The **Index** pane now has an alphabetical quick index, so you can choose a letter to see entries starting with that letter. You can still type any index term in the text box to go directly to that term. Index entries are now shown as links. Entries that are merely headings do not go to a specific page and do not appear as links. As is true for all desktop tools, you can now drag columns in the **Index** pane to reorder them, or click a column head to sort by that column.

Search Type Removed

In the **Search** pane, you no longer select the type of search. Results are ordered so reference pages appear first, followed by headings that include the search terms. After performing a search, click the link at the bottom of the **Search** pane to look for the same term in the technical support database on The MathWorks Web site. As is true for all desktop tools, you can now drag columns in the **Search** pane to reorder them, or click a column head to sort by that column.

Favorites in Help Browser

Add pages in the Help browser to favorites (also known as bookmarking pages) by selecting **Favorites -> Add to Favorites**. The **Favorites Editor** dialog box opens. Accept the default entries or modify the **Label** and click **Save**. Access favorites from the **Favorites** menu or from the **Start** menu **Shortcuts** item.

Display Pane Find in Page Icon

Click the binoculars icon on the display pane toolbar to search within the page.

Title Field No Longer Supports Web Browsing

The Help browser is now used only for MathWorks documentation installed with your products.

You can no longer enter a URL in the **Title** field of the display pane. Instead run the `web` function to enter a URL in the **Location** field. Links from the documentation to Web pages display the Web pages in the MATLAB Web browser, not in the Help browser.

docsearch Function Added to Execute Help Browser Search

The new `docsearch` function allows you to execute a full text search of the Help browser documentation from the Command Window.

help Function Provides Help for Methods and Classes

The `help` function now allows you to get help for methods and classes. For details, see specific instructions in the release notes about using `help` and `doc` for each product, or type `help help`.

web Function Now Opens MATLAB Web Browser By Default

The `web` function no longer opens the specified URL in the Help browser by default, but instead opens the page in the MATLAB Web browser.

Compatibility Considerations. If you want `web` to open pages in the Help browser, use the `-helpbrowser` option.

HTML Documentation Not Viewable with -nojvm Startup Option

If you start MATLAB using the `-nojvm` option, you cannot view the HTML documentation files from within MATLAB. The `docopt` function no longer supports that option.

Compatibility Considerations. This represents a change from previous versions. You can view the HTML documentation files at the MathWorks Web site.

Workspace, Search Path, and File Operations

New features and changes introduced in this version are organized by these topics:


- Workspace Browser Enhancements and Changes
- Array Editor Enhancements and Changes
- Built-In Functions Now Treated Like Other M-Files on Search Path
- `savepath` Function Added to Replace `path2rc`
- Search Path Functions—Other Enhancements and Changes
- File Operations
- Current Directory Browser Changes and Enhancements
- Visual Directory and Directory Reports in Current Directory Browser

Workspace Browser Enhancements and Changes

Demo of New Workspace Browser Features. If you are using the Help browser, watch the new Workspace Browser video demo for an overview of the major functionality.

Value Column Added to Workspace Browser. The Workspace browser now includes a **Value** column where you can see the content of the variable, or a description of the content. Click the value in the **Value** column to edit the content.

Rename or Duplicate Variable in Workspace Browser. Click a variable name (in the **Name** column) to rename the variable. To create a copy of a variable, right-click and select **Duplicate** from the context menu.

Plotting Selected Variables from Workspace Browser. Click the plot icon  in the Workspace browser toolbar to plot the selected variable. Choose from other applicable plots by clicking the arrow next to the plot button. The function used to create the plot appears in the Command Window so you can use it again later.

Print from Workspace Browser. Click the print button in the Workspace browser toolbar to print a view of the current workspace.

MAT-Files Compressed by Default. MAT-files are now compressed by default. For details on compressing MAT-files, see the Programming release notes.


genvarname Function Added to Construct MATLAB Variable Name. Use the new function `genvarname` to construct a valid MATLAB variable name from a given candidate, where the candidate can be a string or a cell array of strings. For details, type `help genvarname`.

datatipinfo Function Added to Display Information About Variable. The new function `datatipinfo(x)` displays information about the variable, `x`.

Array Editor Enhancements and Changes

Demo of New Array Editor Features. If you are using the Help browser, watch the new Array Editor video demo for an overview of the major functionality.

Cell Arrays and Structures Now Supported. You can now view and edit the content of cell arrays and structures in the Array Editor. For example, double-click a structure in the Workspace browser to open it in the Array Editor. In the Array Editor, double-click an element of the structure to open it as its own Array Editor document. You can then view and edit the contents.

Plotting Multiple Elements. You can select contiguous elements in an array, and then click the plot button  on the Array Editor toolbar to plot only the selected elements. Click the arrow next to the plot button in the toolbar to select from other applicable plots.

Print from Array Editor. You can print an array from the Array Editor. Select **File -> Print** to create a print of the current variable.

Larger Arrays Supported. You can open arrays having up to 2^{19} (524288) elements, which is eight times more than the previous limit, 2^{16} (65536).

Save to MAT-File. You can save a variable to a MAT-file from the Array Editor. Select **File -> Save** and complete the resulting **Save** dialog box.

Built-In Functions Now Treated Like Other M-Files on Search Path

MATLAB now considers built-in files to be the same as other M-files on the search path.

Compatibility Considerations. MATLAB no longer considers built-in functions differently from any other M-files on the search path. MATLAB now looks for a given name first as a variable, then as an M-file in the current directory, and

finally as an M-file on the search path. Previously MATLAB looked for a given name as a built-in function after looking for it as a variable.

If you have a function name that is the same as a MATLAB built-in function, your function might run instead of the built-in function, whereas in previous releases the built-in function would have run. For the built-in function to run, remove or rename your function, or change the directory order in the search path.

savepath Function Added to Replace path2rc

There is a new function, `savepath`, that saves the current search path to a file, `pathdef.m`, so that you can use the same search path in future sessions. Note that this function replaces `path2rc`.

Compatibility Considerations. The `path2rc` function has been replaced by a new function, `savepath`. If you use `path2rc`, it will run `savepath` instead. The new function, `savepath`, performs the same actions as `path2rc` did, but uses a more intuitive name. In addition, `savepath` is case-sensitive on PC platforms, whereas `path2rc` was not. Use `savepath` instead of `path2rc`, and replace existing instances of `path2rc` with `savepath`.

Search Path Functions—Other Enhancements and Changes

restoredefaultpath Function Added for Recover from Problems. There is a new function, `restoredefaultpath`, that helps redefine the search path file, `pathdef.m`, to include only files installed with MathWorks products. Use this function to recover from problems with the path. If that fails, run

```
restoredefaultpath; matlabrc
```

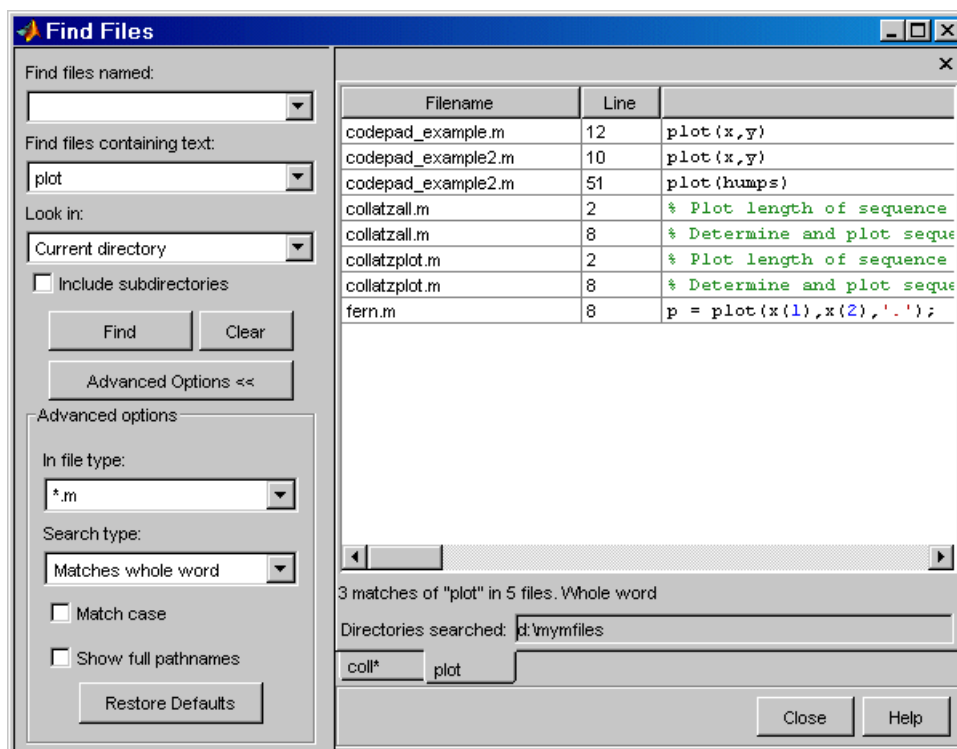
genpath Function Now Includes Empty Directories. The `genpath` function now includes empty directories in the generated path string.

which Function Now Shows Built-In Functions. The `which` function now displays the pathname for built-in functions, as well as for overloaded functions when only the overloaded functions are available

File Operations

Finding Files and Content Within Files. From any desktop tool, select **Edit -> Find Files**. Complete the resulting dialog box to find specified files or files

containing specified text in the directories you choose. Double-click a file in the results listing to open it. For details, see “Finding Files and Content Within Files” in the online documentation.



Preventing Accidental File Deletion. Use the new recycle function or the **General** preference for the delete function to send files you remove using the delete function to the Recycle Bin on Windows, to the Trash Can on Macintosh, or to a /tmp/MATLAB_Files_timestamp directory on UNIX systems. You can then recover any accidentally deleted files from these locations.

Current Directory Browser Changes and Enhancements

Demo of New Current Directory Browser Features. If you are using the Help browser, watch the new Current Directory Browser video demo for an overview of the major functionality.

Source Control Interface Features Accessible from Current Directory Browser. You can access source control system features from the Current Directory browser. Right-click a file or directory, and from the context menu, select **Source Control** and then select the source control function you want to use.

Open File Using External Application. To open a file using an external application, select **Open Outside MATLAB** from the context menu. For example, if you select `myfile.doc`, **Open Outside MATLAB** opens `myfile.doc` in Microsoft Word, assuming you have the `.doc` file association configured to launch Word.

Copy and Paste Directories. Using the Current Directory browser, you can now copy and paste directories, including the entries contents.

Drag to Reorder Columns. As is true for all desktop tools, you can drag columns in the Current Directory browser to reorder them, or click a column head to sort by that column. For an item that does not fit in its column, you can hover over it to see the full name of the item.

Current Directory Text Field Does Not Display When Current Directory Browser is Docked. The current directory field appears in the Current Directory browser only when the Current Directory browser is undocked from the MATLAB desktop. When the Current Directory browser is docked in the MATLAB desktop, use the current directory text field in the desktop toolbar.

Visual Directory and Directory Reports in Current Directory Browser

There are new tools accessible from the Current Directory browser for tuning and managing M-files. For details, see Visual Directory Tool in the Current Directory Browser and Directory Reports in the Current Directory Browser.

Editing and Debugging M-Files

New features and changes introduced in this version are

- Demo of New Editor Features
- Opening, Arranging, and Closing Documents
- Visual Changes
- Entering Statements
- Create Block Comments Using `%{` and `%}`
- Finding and Replacing Text
- Printing M-Files
- Breakpoints and Debugging
- Debugging Functions Enhanced
- `dbstack` Function Supports Nested Functions
- `dbstatus` Function Supports Conditional Breakpoints
- Access Tools from Editor/Debugger
- Rapid Code Iteration Using Cells
- Preferences for the Editor/Debugger

Demo of New Editor Features

If you are using the Help browser, view the Editor new features video demo to see highlights of the major new features.

Opening, Arranging, and Closing Documents

Drag File into Editor. You can drag a file onto the Editor to open it. For example, drag a text file from Windows Explorer onto the Editor.

Automatically Remove Autosave Files—Preference Added. There is now an Editor/Debugger preference you can set to automatically remove autosave files when you close the source file. Select **Preferences -> Editor/Debugger -> Autosave**, and under **Close options**, select the **Automatically delete autosave files** check box.

Toggle Between Command Window and Editor/Debugger. To move from an Editor document to the Command Window, press **Ctrl+0**. To move back to the Editor document, press **Ctrl+Shift+0**.

Editor Remains Open with No Files Open. When you close the last open document in the Editor, the Editor remains open.

Automatic Reloading of Files When Changes Made Outside of MATLAB. When a file is open in the Editor and you open that same file outside of MATLAB and make changes to it, the Editor automatically updates the file to include the changes you made outside the Editor. This only applies if you did not make any changes to the file in the Editor. If you want to be prompted before the Editor updates the file, clear the Editor/Debugger preference for automatically reloading files.

Open Files While Debugging Preference Moved to Debug Menu. In the previous version, you used a preference to automatically open files when debugging. Now, instead of using a preference, you select **Open M-Files When Debugging** from the **Debug** menu in any desktop tool.

With this item selected, when you run an M-file containing breakpoints, the file opens in the Editor/Debugger when MATLAB encounters a breakpoint.

Visual Changes

Syntax Highlighting for Other Languages. The Editor now supports syntax highlighting for other languages, specifically C/C++, Java, and HTML. Use Editor language preferences to change the colors for the syntax highlighting.

Datatips Now Off By Default in Edit Mode. In edit mode, datatips are now off by default. Select the preference to display them in edit mode. Datatips display until you move the cursor. Datatips are always on in debug mode.

Vertical Line in Files. There is now a faint line at column 75, which serves as a useful reminder of where text would be cut off when printing the document. Remove the line or change the column at which the line appears using Editor/Debugger Display preferences.

Balance Delimiters Removed. The feature **Text -> Balance Delimiters** has been removed.

Syntax Highlighting Default Colors Modified. The default colors for syntax highlighting M-files have been modified. Unterminated strings are now maroon, while terminated strings are now purple. This is the opposite of previous versions. Maroon is considered to be more of an “alerting” color, resembling the default of red for errors, which is the reason for the change. If

you prefer the colors used in previous versions, change them using preferences—see “Syntax Highlighting Colors” in the online documentation.

In addition, arguments in statements entered using command syntax rather than function syntax are highlighted as strings, emphasizing that variables in command syntax are passed as literal strings rather than as their values.

Entering Statements

Change Case of Selected Text. To change the case of selected text, select the text and then press:

- **Alt+U, U** to change all text to upper case
- Press **Alt+U, L** to change all text to lower case
- Press **Alt+U, R** to change the case of each letter

Nested Function Indenting Supported. MATLAB now supports nested functions and the Editor provides preferences regarding how to indent them.

Overwrite Mode Supported. When you press the **Insert** key, text entry is done in overwrite mode and the cursor assumes a block shape. Press the **Insert** key again to return to insert mode.

Create Block Comments Using %{ and %}

You can create a block comment in an M-file using any text editor, that is, you can comment out contiguous lines of code. Type `%{` on the line before the first line of the comment and `%}` following the last line of the comment. The lines in between are considered to be comments. Do not include any code on the lines with the block comment symbols. You can also nest block comments. See “Commenting Using Any Text Editor” for details.

Compatibility Considerations. Because of the new block comment symbols, if you have any files with lines that consist only of `%{` and `%}`, they might be misinterpreted as block comment start and end symbols, and might cause errors in your file.

Finding and Replacing Text

Find Files and Content Within Files. You can find directories, files, and content within multiple files. Select **Edit -> Find Files**. For details, see “Finding Files and Content Within Files” in the online documentation.

Incremental Search Now Indicates Direction and is Case Sensitive. The incremental search interface has been updated. It now indicates the search direction. It is also case-sensitive when you enter uppercase letters in the search field.

Printing M-Files

Page setup options differ slightly from previous versions.

Breakpoints and Debugging

Conditional Breakpoints Supported. You can specify conditional breakpoints in an M-file. MATLAB only stops at the line with the breakpoint if the condition is met. Conditional breakpoints have a yellow breakpoint icon, which you can copy and paste to other lines.

Disable (Ignore) Breakpoints. You can disable standard and conditional breakpoints. MATLAB ignores a disabled breakpoint until you enable it again. A disabled breakpoint icon has an X through it.

Error Breakpoints Supported. Set error breakpoints for all files by selecting **Debug -> Stop If Errors/Warnings**, and then completing the resulting dialog box. You can specify a message identifier for an error or warning breakpoint so that MATLAB stops only if it encounters the specified error or warning message.

Debugging Functions Enhanced

Error Support Added to dbstop and dbclear. Enhancements to debugging functions include dbstop if caught error, dbclear if caught error, and dbclear if all error. The dbstop if all error option has been grandfathered and will not be supported in future versions. To specify a message identifier, use dbstop if error ID, dbstop if caught error ID, dbstop if warning ID, and the corresponding dbclear options. The dbstatus function has been updated to reflect the changes to dbstop and dbclear.

dbstop Function Supports Nested and Anonymous Functions. The `dbstop` function has been updated to support nested and anonymous functions. See the `dbstop` reference page for details. You cannot use the Editor/Debugger GUI to set breakpoints in anonymous functions, but must use the `dbstop` function instead. Note that when you save a file in the Editor/Debugger that contains breakpoints in anonymous functions, those breakpoints are cleared. They are also cleared when you run an unsaved file from the Editor/Debugger GUI, because running first saves the file.

Notation for Reporting Path Modified. MATLAB now uses a new notation for reporting the path of functions, subfunctions, and nested functions. As an example, `A/B>C/D` means directory A, file B, (sub)function C within the file B, and nested function D within C.

dbstack Function Supports Nested Functions

The `dbstack` function has been updated to supported nested functions. See the `dbstack` function reference page for more information.

Compatibility Considerations. If you use `dbstack` in M-files, you might need to update your files because of this change. When you run `dbstack` and return results to a structure, there are now three fields, whereas in previous versions, there were only two fields. The fields are:

- `file`, the file in which the function appears
- `name`, the function name within the file
- `line`, the line number in the function

The `file` field does not contain a complete pathname, as the `name` field did in previous versions. To get the complete pathname, use `dbstack(' -completenames')`.

dbstatus Function Supports Conditional Breakpoints

The `dbstatus` function has been updated to support conditional breakpoints. See the `dbstatus` function reference page for more information.

Compatibility Considerations. As a result there have been changes to some of the fields in the structure returned with `s = dbstatus(...)`. If you use `dbstatus` in M-files, you might need to update your files because of this change. For details on the new format, see the `dbstatus` reference page.

Access Tools from Editor/Debugger

You can access useful tools for M-files from the Editor/Debugger. From the **Tools** menu, select **Check Code with M-Lint**, **Show Dependency Report**, or **Open Profiler**. For details about these tools, see *Tuning and Managing M-Files*.

Rapid Code Iteration Using Cells

In the Editor, cell features allow you to easily make changes to values in a section of an M-file to readily see the impact of the changes. First, you define cells in a file, then evaluate a cell or cells, iterate values in the cell, and then reevaluate the cell(s). Cells also allow you to publish M-file code and results to popular formats, such as HTML and Microsoft Word. For details, see “Rapid Code Iteration Using Cells” in the online documentation.

Demo of New Rapid Code Iteration Features. If you are using the Help browser, watch the new Rapid Code Iteration Using Cells video demo for an overview of the major functionality.

Compatibility Considerations. Because of the new symbols for cell publishing, if you have any files with lines that consist only of `%%`, those lines might be misinterpreted as the start of a cell. Your files will still run without problems, but if you publish the M-files, you might need to modify those lines.

Preferences for the Editor/Debugger

Add New Line to End of File Upon Save. There is now a preference that allows you to add a new line to end of a file upon saving.

Open M-Files When Debugging Preference Moved. The feature that instructs M-files to open automatically when debugging is no longer in preferences but is now accessible from the **Debug** menu in all desktop tools.

Tuning and Managing M-Files

Use these tools to fine tune and manage your M-files, and to prepare them for distribution to other users. New features introduced in this version are organized by these topics:



- Demo of New Directory Reports Features
- Visual Directory Tool in the Current Directory Browser
- Directory Reports in the Current Directory Browser
- Profiler for Measuring Performance

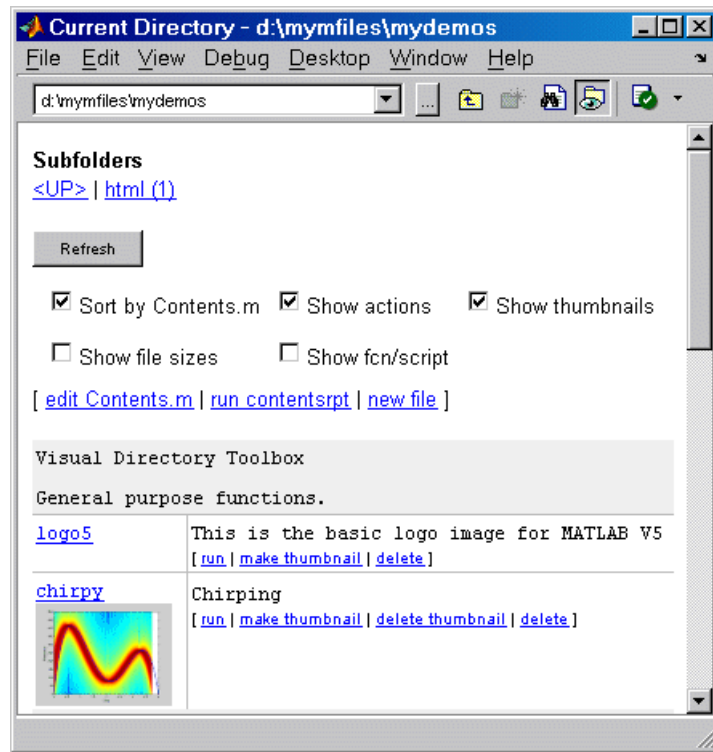
Demo of New Directory Reports Features

If you are using the Help browser, watch the new Directory Reports video demo for an overview of the major functionality.

Visual Directory Tool in the Current Directory Browser

The Visual Directory view of the Current Directory provides useful information about the M-files in a directory. It can help you polish M-files before providing them to others to use.

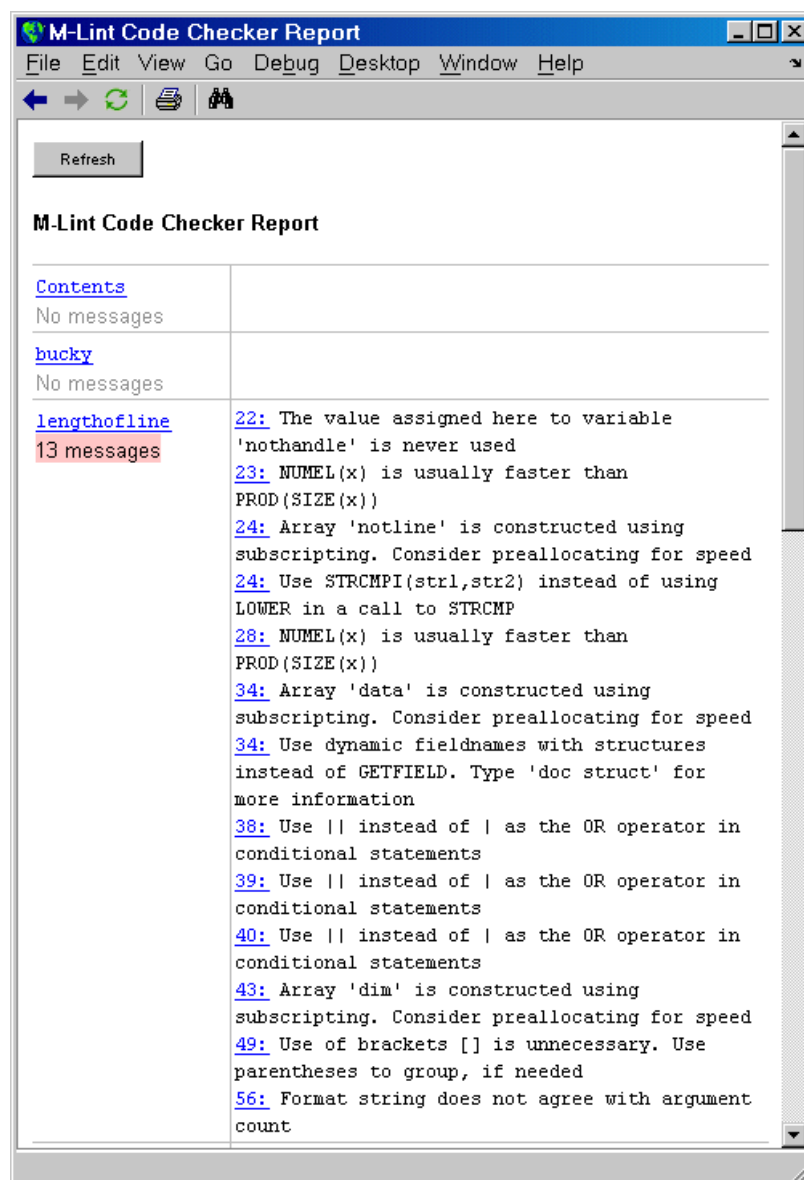
Click the Show Visual Directory button  on the Current Directory browser toolbar. The view changes—see the following figure for an example. To return to the Classic view of the Current Directory browser, click the button  again.



Directory Reports in the Current Directory Browser

In the Current Directory browser, select **View -> Directory Reports** and select the type of report to run. The report appears as an HTML document in the MATLAB Web browser. A summary of the reports follows. For more information, see “Directory Reports in Current Directory Browser” in the online documentation.

M-Lint Code Check Report. The M-Lint report displays potential errors and problems, as well as opportunities for improvement in your code. For example, one common message is that a variable is defined but never used. You can also produce an M-Lint report for specified files using the `mlint` function, or run the M-Lint report from the Editor/Debugger or Profiler.



TODO/FIXME Report. The TODO/FIXME report shows M-files that contain text strings you included as notes to yourself, such as TODO.

Help Report. The Help report presents a summary view of the help component of your M-files. Use this information to help you identify files of interest or to help you identify files that lack help information.

Contents Report. The Contents report displays information about the integrity of the Contents.m file for the directory. A Contents.m file is a file you create that provides a brief description for relevant M-files in the directory. When you type help followed by the directory name, such as help mydemos, MATLAB displays the information in the Contents.m file. Use the Contents report to help you clean up and maintain your Contents.m file. If there is no Contents.m file, use the Contents report to create one.

Dependency Report. The Dependency report shows all M-files called by each M-file, or in other words, shows all children of each M-file. Use this report to determine all files you need to provide to someone who wants to run an M-file.

File Comparison Report. The File Comparison report identifies the differences between two files in the current directory. For example, you can easily compare an autosaved version of a file to the latest version of the file.

Coverage Report. Run the Coverage report after you run the Profile report to identify what percentage of the file was executed when it was profiled.

Profiler for Measuring Performance

Access Profiler from Desktop or Editor/Debugger. Access the Profiler from the Desktop menu or the Editor/Debugger Tools menu.

Profiler Summary Report. In the Profiler summary report, click a column name to sort the report by that column.

Profiler Detail Report. In the Profiler detail report, specify options to show busy lines (lines where the most time was spent) and to show the file listing (the M-file code). Other options allow you to run the M-Lint Code Check report, which provides messages for improving the file, and the Coverage report, which indicates how much of the file was exercised during profiling. For more information about these reports, see “Directory Reports in Current Directory Browser” in the online documentation. After selecting an option in the detail

report, click **Refresh** to update the report. The performance acceleration information in the detail report has been removed.

Compatibility Considerations. The profile report previously supported in MATLAB is no longer available. This was the report you generated by running `profile` report or `profreport`. There is a new function, `profsave` that replaces `profreport`. The `profsave` function saves a static version of the HTML profile report.

Source Control Changes

In MATLAB 6.5 (R13) and MATLAB 7.0 (R14), only source control systems that comply with the Microsoft Common Source Control standard are supported. If there is a compliant source control system installed on your machine, it will be listed in the Source Control options in the MATLAB Preferences dialog.

There are several vendors who provide and interface into Revision Control Systems (RCS), Concurrent Versions System (CVS), and other such tools using Microsoft Source Code Control API. ComponentSoftware provides one such interface layer.

Compatibility Considerations. This represents a change to how MATLAB interfaced with source control systems in prior versions.

Publishing Results

Publishing to HTML, XML, LaTeX, Word, and PowerPoint

You can publish M-files to HTML, XML, LaTeX, Word, and PowerPoint documents. The published documents can include code, formatted comments, and results, such as graphs in Figure windows. Use cells and cell publishing features in the Editor/Debugger. For details, see “Publishing to HTML, XML, LaTeX, PowerPoint, and Word Using Cells” in the online documentation.

Demo of New Publishing Features. If you are using the Help browser, watch the new Publishing M Code from the Editor video demo for an overview of the major functionality.

Notebook

If you currently use Notebook, consider using cell publishing from the Editor instead, which provides more features and flexibility for most applications.

Notebook has been improved with regards to speed and stability, with a few minor changes in operation. The improvements were available via a Web-downloadable update to MATLAB version 6.5, and are now part of MATLAB version 7. For details about the differences, see Solution 36072 on the MathWorks Web site.

Mathematics, MATLAB Version 7 (R14)

This version introduces the following new features and changes:

New and Obsolete Functions

- New Functions
- Obsolete Functions

Nondouble Math

- New Nondouble Mathematics Features
- Nondouble Arithmetic
- New Integer Functions — intmax and intmin
- New Warnings for Integer Arithmetic
- Warning on Concatenating Different Integer Classes
- Integer Data Type Functions Now Round Instead of Truncate
- Changes to Behavior of Concatenation
- Class Input for realmax and realmin
- Class Input for ones, zeros, and eye
- Class and Size Inputs for Inf and NaN
- New Class and Data Inputs for eps
- New Class Inputs for sum
- complex Now Accepts Inputs of Different Data Types
- Bit Functions Now Work on Unsigned Integers

Matrices and Elementary Math

- New Function accumarray for Constructing Arrays with Accumulation
- Enhanced sort Capabilities and Performance
- New Functions for Numerical Data Types
- max and min Now Have Restrictions on Inputs of Different Data Types
- Mathematic Operations on Logical Values

Linear Algebra

- New Function `linsolve` for Solving Systems of Linear Equations
- New Output for `polyeig`
- Enhancements to `lsq`
- New Form for Generalized Hessian

Nonlinear Methods

- New Function `quadv` Integrates Complex, Array-Valued Functions

Trigonometry, Geometry

- New Trigonometric Functions For Angles in Degrees
- Matrix, Trigonometric, and Other Math Functions No Longer Accept Inputs of Type `char`
- New Warnings for Complex Inputs to `atan2`, `log2`, and `pow2`
- Enhanced Functions for Computational Geometry
- New Support for Interpolation Functions

Differential Equations

- New and Enhanced Functions for Ordinary Differential Equations (ODEs)

FFT

- Enhancements to Discrete Fourier Transform Functions
- FFT Functions Applied to Integer Data Types are Becoming Obsolete

Optimization

- New Output Function for Optimization Functions

Specialized Math

- New Input Argument for Incomplete Gamma Function

Other

- Overriding the Default BLAS Library on Intel/Windows Systems
- New Names for Demos expm1, expm2, and expm3

New Functions

This release introduces the following new functions:

Function	Description
accumarray	Construct array with accumulation
cast	Cast variable to different type
expm1	Compute $\exp(x) - 1$ accurately for small values of x
intmax	Largest value of specified integer type
intmin	Smallest value of specified integer type
intwarning	Control state of integer warnings
isfloat	Determine whether input is floating-point array
isinteger	Determine whether input is integer array
linsolve	Solve linear system of equations
log1p	Compute $\log(1+x)$ accurately for small values of x
nthroot	Real n th root of real numbers
ode15i	Solve fully implicit differential equations, variable order method
odextend	Extend solution of initial value problem for ordinary differential equation
quadv	Vectorized quadrature

Obsolete Functions

The functions listed in the left column of the following table are obsolete and will be removed from a future version of MATLAB.

Compatibility Considerations

Use the replacement functions listed in the right column instead.

Obsolete Function	Replacement Function
colmmd	colamd
quad8	quadl
symmmd	symamd

The following obsolete functions are no longer included in MATLAB:

fmin, fmins, icubic, interp4, interp5, interp6, meshdom, nnls,
saxis

New Nondouble Mathematics Features

MATLAB Version 7.0 supports many arithmetic operations and mathematical functions on the following nondouble MATLAB data types:

- single
- int8 and uint8
- int16 and uint16
- int32 and uint32

Most of the built-in MATLAB functions that perform mathematical operations now support inputs of type `single`. In addition, the arithmetic operators and the functions `sum`, `diff`, `colon`, and some elementary functions now support integer data types.

Note In Version 7.0, MATLAB only supports mathematical operations on nondouble data types for built-in functions; it does *not* support these operations for M-file functions unless otherwise stated in the M-file help.

Nondouble Arithmetic

This section describes how MATLAB performs arithmetic on nondouble data types.

Single Arithmetic

You can now combine numbers of type `single` with numbers of type `double` or `single`. MATLAB performs arithmetic as if both inputs had type `single` and returns a result of type `single`. For more information, see “Single-Precision Floating Point” in the online MATLAB Programming documentation.

Integer Arithmetic

You can now combine numbers of an integer data type with numbers of the same integer data type or type `scalar double`. MATLAB performs arithmetic as if both inputs had type `double` and then converts the result to the same integer data type.

MATLAB computes operations on arrays of integer data type using *saturating* integer arithmetic. Saturating means that if the result is greater than the upper bound of the integer data type, MATLAB returns the upper bound. Similarly, if the result is less than the lower bound of the data type, MATLAB returns the lower bound. For more information, see “Integers” in the online MATLAB Programming documentation.

New Integer Functions — `intmax` and `intmin`

Two new functions, `intmax` and `intmin`, return the largest and smallest numbers, respectively, for integer data types. For example,

```
intmax('int8')
ans =
    127
```

returns the largest number of type `int8`. See “Largest and Smallest Values for Integer Data Types” in the online MATLAB documentation for more information.

New Warnings for Integer Arithmetic

This section describes four new warning messages for integer arithmetic in Version 7.0. While these warnings are turned off by default, you can turn them on as a diagnostic tool or to warn of behavior in integer arithmetic that might not be expected.

To turn all four warning messages on at once, enter

```
intwarning on
```

Integer Conversion of Noninteger Values. MATLAB can now return a warning when it rounds up a number in converting to an integer data type. For example,

```
int8(2.7)
Warning: Conversion rounded non-integer floating point value to
nearest int8 value.
```

```
ans =
     3
```

Integer Conversion of NaN. When MATLAB converts NaN (Not-a-Number) to an integer data type, the result is 0. MATLAB can now return a warning when this occurs. For example,

```
int16(NaN)
Warning: NaN converted to int16(0).
```

```
ans =
     0
```

Integer Conversion Overflow. MATLAB can now return a warning when you convert a number to an integer data type and the number is outside the range of the data type. For example,

```
int8(300)
Warning: Out of range value converted to intmin('int8') or
intmax('int8').

ans =
    127
```

Integer Arithmetic Overflow. MATLAB can now return a warning when the result of an operation on integer data types is either NaN or outside the range of that data type. For example,

```
int8(100) + int8(100)
Warning: Out of range value or NaN computed in integer arithmetic.

ans =
    127
```

To turn all of these warnings off at once, enter

```
intwarning off
```

Warning on Concatenating Different Integer Classes

If you concatenate integer arrays of different integer classes, MATLAB displays the warning

```
Concatenation with dominant (left-most) integer class may
overflow other operands on conversion to return class.
```

The class of the resulting array is the same as the dominant (or left-most) value in the concatenation:

```
a = int8([52 37 89; 23 16 47]);
b = int16([74 61 32; 98 73 25]);
```

```
% Combine int8 and int16 (int8 is dominant)
c = [a b];
class(c)
ans =
    int8

% Combine int16 and int8 (int16 is dominant)
c = [b a];
class(c)
ans =
    int16
```

Integer Data Type Functions Now Round Instead of Truncate

The following integer data functions now round noninteger inputs instead of truncating:

```
int8, uint8, int16, uint16, int32, uint32, int64, uint64
```

For example, in MATLAB 7.0,

```
int8(3.7)
```

returns

```
ans =
    4
```

Compatibility Considerations

In previous releases, the same command returned 3. If you have code that contains these functions, it might return different results in Version 7.0 than in previous releases, in particular, results that differ by 1 after converting floating-point inputs to an integer data type.

You can turn the following warning on to help diagnose these differences:

```
warning on MATLAB:intCovertNonIntVal
```

See [New Warnings for Integer Arithmetic](#) for more information about this and other new warning messages.

Changes to Behavior of Concatenation

When you perform concatenation (`[a, b]`, `[a;b]`, and `cat(a,b,dim)`) on mixed integer and other numeric or logical inputs, the left-most integer type among the inputs is the type of the result. As a result, the other inputs might lose values when they are converted to the integer data type. In Version 7.0, MATLAB now returns a warning when you concatenate these mixed data types.

For example,

```
[int8(100) uint8(200)]  
Warning: Concatenation with dominant (left-most) integer class  
may overflow other operands on conversion to return class.  
(Type "warning off MATLAB:concatenation:integerInteraction" to  
suppress this warning.)
```

```
ans =  
    100    127
```

```
class(ans)  
ans =  
    int8
```

Compatibility Considerations

Concatenating an input of any nondouble numeric data type (single and integer data type) with type `char` now returns a result of type `char`. In previous releases, the same operation returned a result of the same type as the numeric data type.

Class Input for `realmax` and `realmin`

You can now call the function `realmax` with the syntax

```
realmax('single')
```

```
ans =
```

```
3.4028e+038
```

which returns the largest single-precision number. Similarly,

```
realmin('single')
```

returns the smallest single-precision number.

The commands `realmax('double')` and `realmin('double')` return the same results as `realmax` and `realmin`, respectively. See “Largest and Smallest Values for Floating-Point Data Types” in the online MATLAB documentation for more information.

Class Input for `ones`, `zeros`, and `eye`

You can now call `ones` or `zeros` with an input argument specifying the data type of the output. For example,

```
ones(m, n, p, ..., 'single')
```

or

```
ones([m, n, p, ...], 'single')
```

returns an m -by- n -by- p -by ... array of type `single` containing all ones. `zeros` uses the same syntax.

You can now call `eye` with this input argument for two-dimensional arrays. For example,

```
eye(m, 'single')
```

returns an m -by- m identity matrix of type `single`. The command

```
eye(m, n, 'int8')
```

returns an m -by- n array of type `int8`.

Class and Size Inputs for Inf and NaN

The functions `Inf` and `NaN` now accept inputs that enable you to create Infs or NaNs of specified sizes and floating-point data types. As examples,

- `Inf('single')` or `NaN('single')` create the single-precision representations of Inf and NaN, respectively.
- `Inf(m,n,p, ...)` or `NaN(m,n,p, ...)` create m-by-n-by-p-by-... arrays of Infs or NaNs, respectively.

See the reference pages for `Inf` and `NaN` for more information.

New Class and Data Inputs for eps

You can now call the function `eps` with the syntax

```
eps(x)
```

If `x` has type `double`, `eps(x)` returns the distance from `x` to the next largest double-precision floating point number. This is a measure of the accuracy of `x` as a double-precision number. `eps(1)` returns the same value as `eps` with no input argument.

You can now replace expressions of the form

```
if Y < eps * abs(X)
```

with

```
if Y < eps(X)
```

If `x` has type `single`, `eps(x)` returns the distance from `x` to the next largest single-precision floating point number. This is a measure of the accuracy of `x` as a single-precision number.

The command

```
eps('single')
ans =
    1.1921e-007
```

returns the same value as `eps(single(1))`. The value of `eps('single')` is the same as `single(2^-23)`. The command `eps('double')` returns the same result as `eps`.

See “Accuracy of Floating-Point Data” in the online MATLAB documentation for more information.

New Class Inputs for `sum`

The following new input arguments for `sum` control how the summation is performed on numeric inputs:

- `s = sum(x, 'native')` and `s = sum(x, dim, 'native')` accumulate in the native type of its input and the output `s` has the same data type as `x`. This is the default for `single` and `double`.
- `s = sum(x, 'double')` and `s = sum(x, dim, 'double')` accumulate in double-precision. This is the default for integer data types.

In Version 7.0, `sum` applied to a vector of type `single` performs `single` accumulation and returns a result of type `single`. In other words, `sum(x)` is the same as `sum(x, 'native')` if `x` has type `single`. This is a change in the behavior of `sum` from previous releases. To make `sum` accumulate in `double`, as in previous releases, use the input argument `'double'`.

Compatibility Considerations

In Version 7.0, `sum` applied to a vector of type `single` performs `single` accumulation and returns a result of type `single`. In previous releases, `sum` performed this operation in `double` accumulation.

To restore the previous behavior, call `sum` with the syntax

```
sum(X, 'double')
```

or

```
sum(X, dim, 'double')
```

complex Now Accepts Inputs of Different Data Types

The function `complex` now accepts inputs of different data types when you use the syntax

```
complex(a,b)
```

according to the following rules:

- If either of `a` or `b` has type `single`, `c` has type `single`.
- If either of `a` or `b` has an integer data type, the other must have the same integer data type or type `scalar double`, and `c` has the same integer data type.

Bit Functions Now Work on Unsigned Integers

The following functions now work on unsigned integer inputs:

```
bitand, bitcmp, bitget, bitor, bitset, bitshift, bitxor
```

Instead of using flints (integer values stored in floating point) to do your bit manipulations, consider using unsigned integers, as a more natural representation of bit strings. Instead of using `bitmax`, use the `intmax` function with the appropriate class name. For example, use `intmax('uint32')` if you are working with unsigned 32 bit integers.

In addition, the function `bitcmp` now accepts the following new syntax for inputs of type `uint8`, `uint16`, and `uint32`:

```
bitcmp(A)
```

`bitcmp` now uses the data type of `A` to determine how to take the bitwise complement.

New Function `accumarray` for Constructing Arrays with Accumulation

The new `accumarray` function enables you to construct an array with accumulation. The following example uses `accumarray` to construct a 5-by-5 matrix `A` from a vector `val`. The function `accumarray` adds the entries of `val` to `A` at the indices specified by the matrix `ind`, which has the same number of rows

as `val`. If an index in `ind` is repeated, the entries of `val` accumulate at the corresponding entry of `A`.

```
ind = [1 2 5 5;1 2 5 5]';
val = [10.1 10.2 10.3 10.4]';
A = accumarray(ind, val)
```

A =

```
10.1000     0     0     0     0
     0 10.2000     0     0     0
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0 20.7000
```

To get the (5,5) entry of `A`, `accumarray` adds the entries of `val` corresponding to repeated pair of indices (5,5).

$$A(5, 5) = 10.3 + 10.4$$

In general, if `ind` has `ndim` columns, `A` will be an `N`-dimensional array with `ndim` dimensions, whose size is `max(ind)`.

Enhanced sort Capabilities and Performance

Improved Performance

`sort` performance has been improved for numeric arrays of randomly ordered data. Although there is some performance improvement for all such numeric arrays, you should see the greatest improvement for integer arrays and multidimensional arrays.

Sort Direction

A new argument, `mode`, lets you specify whether `sort` returns the sorted array in ascending or descending order.

New Functions for Numerical Data Types

MATLAB 7.0 contains three new functions for detecting and converting data types:

- `cast` enables you to cast a variable to a different data type or class
- `isfloat` enables you to detect floating-point arrays. `isfloat(A)` returns 1 if `A` has type `double` or `single` and 0 otherwise. `isfloat(A)` is the same as `isa(A, 'float')`.
- `isinteger` enables you to detect integer arrays. `isinteger(A)` returns 1 if `A` has integer data type and 0 otherwise. `isinteger(A)` is the same as `isa(A, 'integer')`

`max` and `min` Now Have Restrictions on Inputs of Different Data Types

In MATLAB 7.0, the functions `max` and `min` now have the following restrictions on inputs of different data types:

- If any input has an integer data type, all other inputs must have the same integer data type or type `scalar double`.
- If any input is of type `single`, all other inputs must have type `double` or `single`.

Other combinations of inputs now return an error message. In previous releases, inputs to `max` or `min` could have any combination of data types.

For the allowed mixed-type combinations listed above, `max` and `min` now return results of a different data type than in previous releases:

- If one input has an integer data type, while another has type `double`, the result now has the same integer data type. In previous releases, the result had type `double`.
- If one input has type `single`, while another has type `double`, the result now has type `single`. In previous releases, the result had type `double`.

You can turn on the following warning messages to diagnose any issues that might result from this change in behavior:

- warning on MATLAB:max:mixedIntegersScalarDoubleInputs
- warning on MATLAB:max:mixedSingleDoubleInputs
- warning on MATLAB:min:mixedIntegersScalarDoubleInputs
- warning on MATLAB:min:mixedSingleDoubleInputs

Compatibility Considerations

Combinations of inputs other than those listed above now return an error message. Also, for these allowed mixed-type combinations, `max` and `min` now return results of a different data type than in previous releases:

Mathematic Operations on Logical Values

Most mathematic operations are not supported on logical values.

New Function `linsolve` for Solving Systems of Linear Equations

The new `linsolve` function enables you to solve systems of linear equations of the form $Ax = b$ more quickly when the matrix of coefficients A has a special form, such as upper triangular. When you specify one of these special types of systems, `linsolve` is faster than `mldivide` or `\` (backslash) because it does not check whether the matrix actually has the form you specify.

Note If the matrix A does not have the form you specify in `opts`, `linsolve` returns incorrect results because it does not perform error checking. If you are unsure of the form of A , use `mldivide`, or `\` instead.

New Output for `polyeig`

The function `polyeig` can now return a vector of condition numbers for the eigenvalues, when you call it with the syntax

```
[X,E,S] = polyeig(A0,A1,...,Ap)
```

At least one of `A0` and `Ap` must be nonsingular. Large condition numbers imply that the problem is close to one with multiple eigenvalues.

Enhancements to `lscov`

The command

```
lscov(A,b,V)
```

now accepts either a weight vector or a covariance matrix for `V`. If you enter `lscov(A,b)` without a third argument, `lscov` uses the identity matrix for `V`.

The command `lscov(A, b, V, alg)` now enables you to specify the algorithm used to compute the result when `V` is a matrix. You can specify `alg` to be one of the following:

- 'chol' uses the Cholesky decomposition of `V`
- 'orth' uses the orthogonal decomposition of `V`

The command

```
[x stdx mse] = lscov(...)
```

now returns `mse`, the mean squared estimate (MSE).

The command

```
[x stdx mse S] = lscov(...)
```

now returns `S`, the estimated covariance matrix of `x`.

In addition, `lscov` can now accept a design matrix `A` that is rank deficient and a covariance matrix, `V`, that is positive semidefinite.

New Form for Generalized Hessian

The function `hess` has a new syntax of the form

$$[AA, BB, Q, Z] = \text{hess}(A, B)$$

where A and B are square matrices, and returns an upper Hessenberg matrix AA , an upper triangular matrix BB , and unitary matrices Q and Z such that

$$Q * A * Z = AA$$

and

$$Q * B * Z = BB$$

New Function `quadv` Integrates Complex, Array-Valued Functions

The new function `quadv` integrates complex, array-valued functions.

New Trigonometric Functions For Angles in Degrees

The following new functions compute trigonometric functions of arguments in degrees.

Function	Purpose
<code>sind</code>	Compute the sine of an argument in degrees
<code>cosd</code>	Compute the cosine of an argument in degrees
<code>tand</code>	Compute the tangent of an argument in degrees
<code>cotd</code>	Compute the cotangent of an argument in degrees
<code>secd</code>	Compute the secant of an argument in degrees
<code>cscd</code>	Compute the cosecant of an argument in degrees

The following new functions compute the inverse trigonometric functions and return the answer in degrees:

Function	Purpose
asind	Compute the inverse sine of an argument and return answer in degrees
acosd	Compute the inverse cosine of an argument and return answer in degrees
atand	Compute the inverse tangent of an argument and return answer in degrees
acotd	Compute the inverse cotangent of an argument and return answer in degrees
asecd	Compute the inverse secant of an argument and return answer in degrees
acscd	Compute the inverse cosecant of an argument and return answer in degrees

Matrix, Trigonometric, and Other Math Functions No Longer Accept Inputs of Type char

Matrix functions, such as `chol`, `lu`, and `svd`, and trigonometric functions, such as `sin` and `cos`, no longer accept inputs of type `char`. In previous releases, these functions simply converted `char` inputs to `double` before performing operations on them.

Compatibility Considerations

To restore the previous behavior of these functions, create an M-file that converts its input to `double` before applying the function. For example, to restore the behavior of `sin`,

- 1 Create a directory called `@char` in a directory on the MATLAB path, for example, your work directory.

2 Create an M-file with the following commands:

```
function s = sin(x)
s = sin(double(x));
```

3 Save the file as `sin.m` in the directory `@char`.

New Warnings for Complex Inputs to `atan2`, `log2`, and `pow2`

The following functions now return a warning for inputs that are not real numbers:

- `atan2(y,x)`
- `[f,e] = log2(x)`
- `pow2(f,e)`

Enhanced Functions for Computational Geometry

The following functions, which perform geometric computations on a set of points in N-dimensional space, now provide many new options:

- `convhull` — Compute convex hulls
- `convhulln` — Compute N-dimensional convex hulls
- `delaunay` — Construct Delaunay triangulation
- `delaunay3` — Construct 3-dimensional Delaunay tessellations
- `delaunayn` — Construct N-dimensional Delaunay tessellations
- `griddata` — Data gridding and surface fitting
- `griddata3` — Data gridding and surface fitting for 3-dimensional data
- `griddatan` — Data gridding and hypersurface fitting (dimensions ≥ 2)
- `voronoi` — Construct Voronoi diagrams
- `voronoin` — Construct N-dimensional Voronoi diagrams

These functions now accept an input cell array `options` that gives you greater control over how they perform calculations. These functions use the software `Qhull`, created at the National Science and Technology Research Center for Computation and Visualization of Geometric Structures (the Geometry

Center). For more information on the available options, see <http://www.qhull.org/>.

New Support for Interpolation Functions

The following interpolation functions now have enhanced features:

- `interp1` — The command `YI = interp1(X,Y,XI)` now accepts a multidimensional array `Y` and returns an array of the correct dimensions. If `Y` is an array of size `[n,m1,m2,...,mk]`, `interp1` performs interpolation for each `m1-by-m2-by-...-mk` value in `Y`. If `XI` is an array of size `[d1,d2,...,dj]`, `YI` has size `[d1,d2,...,dj,m1,m2,...,mk]`.

The command `pp = interp1(X,Y,'method','pp')` uses the specified method to generate the piecewise polynomial form (ppform) of `Y`. See the reference page for `interp1` for information about the available methods.

- `interp2`, `interp3`, and `interpN` — You can now pass in a scalar argument, `ExtrapVal`, which these functions return for any values of `XI` and `YI` that lie outside the range of values spanned by `X` and `Y` defining the grid. For example,

```
ZI = interp2(X,Y,Z,XI,YI,'method',ExtrapVal)
```

returns the value of `ExtrapVal` for any values of `XI` or `YI` that are outside the range of values spanned by `X` and `Y`.

- `ppval` now accepts multidimensional arrays returned by the `spline` function using the syntax

```
YY = ppval(spline(X,Y), XX)
```

Each entry of `YY` is obtained by evaluating `spline(X,Y)` at the corresponding value of `XX`.

- `spline` — The command `YY = spline(X,Y,XX)` now accepts a multidimensional array `Y` and returns an array of the correct dimensions. Note that `YY = spline(X,Y,XX)` is the same as `YY = ppval(spline(X,Y), XX)`.

If `spline(X, Y)` is scalar-valued, then `YY` is of the same size as `XX`. If `spline(X, Y)` is `[D1,...,Dr]`-valued, and `XX` has size `[N1,...,Ns]`, then `YY` has size `[D1,...,Dr, N1,...,Ns]`, where `YY(:,...,:, J1,...,Js)` is the value of `spline(X, Y)` at `XX(J1,...,Js)`. There are two exceptions to this rule:

New and Enhanced Functions for Ordinary Differential Equations (ODEs)

MATLAB 7.0 provides two new functions for solving implicit ODEs and extending solutions to ODEs, along with several enhancements to existing ODE-related functions:

- `ode15i`, which is new in Version 7.0, provides the capability to solve fully implicit ODE and DAE problems of the form $f(t, y, y') = 0$ with consistent initial conditions, i.e., $f(t, y_0, y'_0) = 0$. `ode15i` provides an interface that is similar to that of the other MATLAB ODE solvers and is as easy to use. A supporting function `decic` helps you calculate consistent initial conditions. The existing functions `odeset` and `odeget` enable you to set integration properties that affect the problem solution. `deval` evaluates the numerical solution obtained with `ode15i`.
- `odextend`, which is new in Version 7.0, enables you to extend the solution to an ODE created by an ODE solver.
- `bvp4c` can now solve multipoint boundary value problems. To see an example of how to solve a three-point boundary value problem, enter `threebvp`. To see the code for the example, enter `edit threebvp`. Enter `help bvp4c` to learn more about `bvp4c`.
- `deval` can now evaluate the derivative of the solution to an ODE as well as the solution itself. The command

```
[psxint, spxint] = deval(sol,xint)
```

returns `spxint`, the value of the derivative to `sol`.

Enhancements to Discrete Fourier Transform Functions

The new function `fftw` enables you to optimize the speed of the discrete Fourier transform (DFT) functions `fft`, `ifft`, `fft2`, `ifft2`, `fftn`, and `ifftn`. You can use `fftw` to set options for a tuning algorithm that experimentally determines the fastest algorithm for computing a discrete Fourier transform of a particular size and dimension at run time.

The functions `ifft`, `ifft2`, and `ifftn` now accept the input argument `'symmetric'`, which causes these functions to treat the array `X` as conjugate symmetric. This option is useful when `X` is not exactly conjugate symmetric, merely because of round-off error.

FFT Functions Applied to Integer Data Types are Becoming Obsolete

In previous releases, the following fast Fourier transform (FFT) and related functions cast integer inputs of type `uint8` and `uint16` to `double`, used the `double` algorithm, and returned a `double` result:

- `fft`
- `fftn`
- `ifft`
- `ifftn`
- `conv2`

In Version 7.0, these operations return warning messages that recommend convert the inputs to `double` before applying the function, for example, by `fft(double(x))`.

New Output Function for Optimization Functions

In MATLAB 7.0, you can create an output function for several optimization functions in MATLAB. The optimization function calls the output function at each iteration of its algorithm. You can use the output function to obtain information about the data at each iteration or to stop the algorithm based on the current values of the data. You can use the output function with the following optimization functions:

- `fminbnd`
- `fminsearch`
- `fzero`

See “Output Functions” for an example of how to use the output function.

- `N1` is ignored if `XX` is a row vector, that is, if `N1` is 1 and `s` is 2.
- `spline` ignores any trailing singleton dimensions of `XX`.

New Input Argument for Incomplete Gamma Function

The incomplete gamma function, `gammainc`, now accepts the input argument `tail`, using the syntax

```
Y = gammainc(X,A,tail)
```

`tail` specifies the tail of the incomplete gamma function when `X` is non-negative. The choices are for `tail` are 'lower' (the default) and 'upper'. The upper incomplete gamma function is defined as

$$1 - \text{gammainc}(x,a)$$

Overriding the Default BLAS Library on Intel/Windows Systems

Note Intel has used aggressive optimization to compile MKL. This optimization causes NaNs to be treated as zeros in some situations. Calculations that do not involve NaNs are done correctly. In some calculations that do involve NaNs, the NaNs will not propagate.

MATLAB uses the Basic Linear Algebra Subroutines (BLAS) libraries to speed up matrix multiplication and LAPACK-based functions like `eig`, `svd`, and `\(mldivide)`. At start-up, MATLAB selects the BLAS library to use.

For R14, MATLAB still uses the ATLAS BLAS libraries, however, on Windows systems running on Intel processors, you can switch the BLAS library that MATLAB uses to the Math Kernel Library (MKL) BLAS, provided by Intel.

If you want to take advantage of the potential performance enhancements provided by the Intel BLAS, you can set the value of the environment variable `BLAS_VERSION` to the name of the MKL library, `mk1.d11`. MATLAB uses the BLAS specified by this environment variable, if it exists.

To set the `BLAS_VERSION` environment variable, follow this procedure:

- 1 Click the **Start** button, go to the **Settings** menu, and select **Control Panel**.
- 2 On the **Control Panel** menu, select **System**.

- 3 In the **System Properties** dialog box, click the **Advanced** tab.
- 4 On the **Advanced** panel, click the **Environment Variables** button.
- 5 In the **Environment Variables** dialog box, click the **New** button in the User variables section.
- 6 In the **New User Variable** dialog box, enter the name of the variable as `BLAS_VERSION` and set the value of the variable to the name of the MKL library: `mk1.dll`.

Multithreading Disabled in Intel Math Kernel Library (MKL) BLAS

The Intel Math Kernel Library (MKL) is multithreaded in several areas. By default, this threading capability is disabled. To enable threading in the MKL library, set the value of the `OMP_NUM_THREADS` environment variable. Intel recommends setting the value of the `OMP_NUM_THREADS` variable to the number of processors you want to use in your application.

To set the value of this environment variable, follow the instructions outlined in “Overriding the Default BLAS Library on Intel/Windows Systems” on page 198.

Before enabling multithreading, read the Intel Math Kernel Library 6.1 for Windows Technical User Notes that explains certain limitations of this capability.

New Names for Demos `expm1`, `expm2`, and `expm3`

The demos `expm1`, `expm2`, and `expm3` have been renamed `expmdemo1`, `expmdemo2`, and `expmdemo3`, to avoid a name conflict with the new function `expm1`.

Compatibility Considerations

If you have code that relies on these function names, you will need to change the names in your code.

Programming, MATLAB Version 7 (R14)

Caution If you have saved data to a MAT-file using MATLAB Release 14 Beta 2, please read “MAT-Files Generated By Release 14 Beta2 Must Be Reformatted” on page 205.

This version introduces the following new features and changes:

Save and Load

- MATLAB Stores Character Data As Unicode; Making Release 14 MAT-files Readable in Earlier Versions
- MAT-Files Generated By Release 14 Beta2 Must Be Reformatted
- Unicode-Based Character Classification
- Character Rendering on Linux
- Additional Bytes Reserved in MAT-File Header; Do Not Write To Reserved Space
- Compressed Data Support in MAT-Files
- Saving Structures with the save Function

Function Dispatching

- Case-Sensitivity in Function and Directory Names; Can Affect Which Function MATLAB Selects
- Differences Between Built-Ins and M-Functions Removed; Can Affect Which Function MATLAB Selects
- Warning on Naming Conflict
- Change to How evalin Evaluates Dispatch Context

Functions and Scripts

- Summary of New Functions
- New Function Type — Anonymous Functions
- New Function Type — Nested Functions
- Calling Private Functions From Scripts
- New Calling Syntax for Function Handles; Replace eval With New Syntax
- Arrays of Function Handles
- Calling nargin and narginout with Built-In Functions

Changes to Specific Functions

- getfield and setfield Not To Be Deprecated
- isglobal Function To Be Discontinued
- Recycle to Protect Files from Unwanted Deletion
- bin2dec Ignores Space Characters
- dbstop crashes Are Now Resolved
- Bit Functions on Unsigned Integers
- inmem Returns Path Information

Specific Data Types

- New Features for Nondouble Data Types
- Mathematic Operations on Logical Values
- Accessing Cell and Structure Arrays Without deal

Characters and Strings

- New Features in Regular Expression Support
- Functions that Use Regular Expressions
- Regular Expressions Accept String Vector; No Longer Support Character Matrix Input
- Cell Array Support for String Functions
- Additional Class Output From `mat2str`
- String Properties
- Using `strtok` on Cell Arrays of Strings
- Colon Operator on `char` Now Returns a `char`

Dates and Times

- `datestr` Returns Date In Localized Format
- Form and Locale for weekday
- Freestyle Date String Format
- Reading Date Values with `xlsread`; Conversion No Longer Necessary

File I/O

- Comprehensive Function for Reading Text Files
- New Inputs and Outputs to `xlsread`
- New Inputs and Syntax for `dlmwrite`
- Change in Output from `xlsinfo`
- Importing Complex Arrays
- Using `imread` to Import Subsets of TIFF Images
- Getting Information about Multimedia Files
- All-Platform Audio Recording and Playback
- FTP File Operations
- Web Services (SOAP)
- 64-Bit File Handling on MacIntosh

Error Handling

- Changes to Error Message Format
- nargchk Has a New Format for Error Messages
- Enabling and Disabling Warning Messages
- Catching Ctrl+C in try-catch Statements

Other Topics

- MATLAB Performance Acceleration
- “Using MATLAB” Documentation Is Now Three Books

MATLAB Stores Character Data As Unicode; Making Release 14 MAT-files Readable in Earlier Versions

Prior releases of MATLAB represented character data in memory using a system default character encoding scheme that was padded out to 16-bits. This was the case both in memory and in MAT-files. If this data needed to be accessible to multiple users, each user’s system had to use the same character encoding scheme. For those users whose default encoding scheme differed, the exchange of character-oriented information was not possible.

In Release 14, this limitation is removed by adopting the Unicode character encoding scheme in `mxAArrays` and their storage in MAT-files. For more information regarding Unicode, consult the Unicode Consortium web site at <http://www.unicode.org>.

For more information on saving character data using Unicode encoding, see “Writing Character Data” in the External Interfaces documentation. For information on the internal formatting of MAT-files, see the “MAT-File Format” document in the MATLAB documentation available in PDF format

Compatibility Considerations

Release 14 MATLAB writes character and figure data to MAT-files using Unicode character encoding by default. This is now the default encoding used by MATLAB when writing to MAT-files with the `save` and `hgsave` functions or with the MAT-file external interface functions.

Unicode encoded MAT-files are not readable by earlier versions of MATLAB. Thus, if you save data to a MAT-file using MATLAB Release 14, and you intend

to load this MAT-file into an earlier release of MATLAB, you must override the default encoding during the save, as explained in this section.

You can override the default encoding by using the `-v6` switch with `save` and `hgsave`:

```
save filename -v6
hgsave filename -v6
```

or, when saving with MAT functions, by setting the mode to "wL" on the `matOpen` operation:

```
matOpen(filename, "wL");
```

MAT-Files Generated By Release 14 Beta2 Must Be Reformatted

Any MAT-files that you created with Release 14 Beta 2 were written using an internal format that is no longer supported by MATLAB.

Compatibility Considerations

If you need to import data from these files using any release besides Release 14 Beta 2, you must first regenerate the files as described in this section. You cannot read these files using other releases of MATLAB 7.0, and attempting to read them with MATLAB 6.5 or 6.5.1 will corrupt memory.

There are two ways in which you can regenerate your MAT-file:

- If you want to use the MAT-file with earlier versions of MATLAB, regenerate the file using the local character set for your system. To do this, run MATLAB R14 prerelease or MATLAB R14 Beta 2, load the MAT-file, and rewrite the file using the command

```
save filename -v6
```

- If you want to use the MAT-file with R14 LCS or later, regenerate the file using Unicode character encoding. To do this, run MATLAB R14 prerelease, load the MAT-file, and rewrite it using the following command that uses the `-unicode` default.

```
save filename
```

Caution The final R14 release of MATLAB does not allow you to import a MAT-file written with Release 14 Beta 2. You will get an error if you attempt to do this. To use a Beta 2 MAT-file with Release 14, you must first reformat the file with MATLAB R14 prerelease as described above.

If you no longer have access to Release 14 Beta2 or the R14 prerelease, then you must regenerate the data and save it again.

Unicode-Based Character Classification

Unicode-based character classification APIs are now provided in MATLAB. The new character classification functions work with any locale or language and resolve all locale-specific issues that existed in prior releases.

Character Rendering on Linux

Character data rendering has been improved for Linux operating systems that are configured with a UTF-8 default character set.

Additional Bytes Reserved in MAT-File Header; Do Not Write To Reserved Space

In previous releases of MATLAB, the last 4 bytes of the 128-byte MAT-file header were reserved for use by the MathWorks. In Release 14, the last 12 bytes of this header are reserved. See the PDF file “MAT-File Format” for more information.

Compatibility Considerations

If you have programs that write to any of the last 12 bytes of the MAT-file header, or if they rely on the state of the 8 additional header bytes that are reserved as of this release, you will probably need to change your code. These bytes are now used by MATLAB. If your code writes to these bytes, MATLAB is likely to overwrite this data. If your code reads these bytes, they might not be in the same state as they were in previous releases of MATLAB.

Compressed Data Support in MAT-Files

The `save` function compresses your workspace variables as they are saved to a MAT-file. When writing a MAT-file that you will need to load using an earlier version of MATLAB, be sure to use the `save -v6` command. When you use the `-v6` switch, MATLAB saves the data without compression and without Unicode character encoding. This makes the resulting file compatible with MATLAB Version 6 and earlier.

You can also compress data when using MAT-file interface library functions (`matPut*`) to write to a MAT-file by opening the file with the command `matOpen wz`. See the section “Data Compression” in the MATLAB Programming documentation for more information on this feature.

Saving Structures with the `save` Function

Two new syntaxes for the `save` function enable you to save individual fields of a structure to a file. See the function reference for `save` for more information.

To save all fields of the scalar structure `s` as individual variables within the file, `myfile.mat`, use

```
save('myfile', '-struct', 's')
```

To save as individual variables only those structure fields specified (`s.f1`, `s.f2`, ...), use

```
save('myfile', '-struct', 's', 'f1', 'f2', ...)
```

Case-Sensitivity in Function and Directory Names; Can Affect Which Function MATLAB Selects

Prior to this release, filenames for MATLAB functions and Simulink® models, (M, P, MEX, DLL, and MDL files) and also directory names were interpreted somewhat differently by MATLAB with regards to case sensitivity, depending upon which platform you were running on. Specifically, earlier versions of MATLAB handled these names with case sensitivity on UNIX, but without case sensitivity on Windows.

This release addresses the issue of case sensitivity in an effort to make MATLAB consistent across all supported platforms. By removing these differences, we hope to make it easier for MATLAB users to write platform independent code.

Case Sensitivity in MATLAB 6 and Earlier

There are several rules regarding case sensitivity that were already consistent across all platforms in MATLAB 6, and remain in effect on all platforms in MATLAB 7. MATLAB interprets each of the following with case sensitivity on both Windows and UNIX:

- Function names that correspond to MATLAB built-ins
- M-file subfunction names
- The names of functions imported from another language environment, such as Java or COM

UNIX. On all UNIX platforms, including the new implementation on MacIntosh, all function, model, and directory names were case sensitive and required an exact match. This rule remains true for UNIX systems in MATLAB 7.

Windows. On Windows platforms, MATLAB 6 obeys the following rules. These rules are changing in MATLAB 7:

- Function and model names were not case sensitive.
- Directory names, including MATLAB class directory names (e.g., @MyClass) and private directory names (e.g., prIVAtE) were not case sensitive.

Case Sensitivity in MATLAB 7

MATLAB 7 removes the platform specific behaviors by adopting its UNIX case sensitivity rules on Windows systems. MATLAB running on Windows now gives preference to an exact (case sensitive) name match, but falls back to an inexact (case insensitive) match when no exact match can be found.

Compatibility Considerations

There are four main conditions under which MATLAB 7 interprets directory or function names differently in regards to case sensitivity:

- “Two Files of the Same Name” on page 209
- “Two Method Files of the Same Name” on page 209
- “One File with an Inexact Match” on page 210
- “Private Directory Names” on page 210

In any of these cases, each described below, there is a potential for MATLAB to select a file other than the one you had intended.

Whenever MATLAB 7 detects a potential naming conflict related to case sensitivity, it issues a warning. If you get one of these warnings when running a MATLAB program, you may want to modify the related code to eliminate the warning, or you may wish to simply disable the warning. To disable this type of warning, see “Turning Off Warnings Caused by Case Mismatch” on page 211

Two Files of the Same Name. Consider the situation in which there are two or more directories on the MATLAB path that contain a function or model file of the same name. The names of these M-files differ only in letter case:

```
H:\released\myTestFun.m
K:\under_test\mytestfun.m
```

Of these two directories, H:\released is closer to the beginning of the MATLAB path and thus has priority over the other:

```
path = H:\released; K:\under_test; ...
```

On Windows Platforms —

- In MATLAB 6, executing the function `mytestfun` invokes `H:\released\myTestFun.m`.
- In MATLAB 7, executing the function `mytestfun` invokes `K:\under_test\mytestfun.m` and also displays the following warning:

```
Function call mytestfun invokes K:\under_test\mytestfun.m
however, function H:\released\myTestFun.m, that differs only in
case, precedes it on the path.
```

On UNIX Platforms —

MATLAB 7 does the same as on Windows, except that the warning message is disabled by default.

Two Method Files of the Same Name. In this case, there are two M-files of the same name that implement methods of a MATLAB base class and one of its subclasses:

```
@baseclass/my_method.m
@subclass/My_Method.m
```

On Windows Platforms —

- In MATLAB 6, the command `my_method(subclass)` invokes `@subclass/My_Method`.
- In MATLAB 7, the same command invokes `@baseclass/my_method` because it is an exact match.

On UNIX Platforms —

MATLAB 7 does the same as on Windows.

One File with an Inexact Match. Another situation that MATLAB now handles differently involves just one function or model file that matches the function being called:

```
H:\released\myTestFun.m
```

However, the name of this M-file does not match the called function (`mytestfun`) in letter case.

On Windows Platforms —

- In MATLAB 6, calling the function `mytestfun` invokes `H:\released\myTestFun.m`.
- In MATLAB 7, calling the function `mytestfun` invokes the same M-file but also displays the following warning:

```
Function call mytestfun invokes inexact match  
H:\released\myTestFun.m.
```

On UNIX Platforms —

- In MATLAB 6, calling the function `mytestfun` results in an error.
- In MATLAB 7, calling `mytestfun` invokes `H:/released/myTestFun.m` and generates the following warning:

```
Function call mytestfun invokes inexact match  
H:/released/myTestFun.m.
```

Private Directory Names. Private functions must reside in a directory named `private` that is one level down from the directory of any calling function. As of this release, the directory name `private` is case sensitive on Windows as it has always been on UNIX.

On Windows Platforms —

- In MATLAB 6, calling function `myprivfun` in an environment where only a subdirectory named `\PriVate` contains the M-file `myprivfun.m` invokes `\PriVate\myprivfun` without displaying a warning.
- MATLAB 7 does the same as MATLAB 6, except that it also displays the following warning:

```
Wrong case spelling of 'private' as a directory name in
\released\PriVate\myprivfun.m.
```

On UNIX Platforms —

- In MATLAB 6, calling function `myprivfun` in an environment where only a subdirectory named `\PriVate` contains the M-file `myprivfun.m` results in an error.
- In MATLAB 7, calling `myprivfun` in this same environment invokes `/Private/myprivfun` and also displays the following warning:

```
Wrong case spelling of 'private' as a directory name in
/released/Private/myprivfun.m.
```

Turning Off Warnings Caused by Case Mismatch

You can disable most warnings caused by case mismatch with the following command:

```
warning off MATLAB:dispatcher:InexactMatch
```

To disable this warning for all of your MATLAB sessions, add this command to your `startup.m` or `matlabrc.m` file.

If you continue to get case sensitivity warnings after entering this command, you can disable a wider range of warnings with the following command:

```
warning off ...
MATLAB:dispatcher:CaseInsensitiveFunctionPrecedesExactMatch
```

Note This latter warning alerts you when, for case sensitivity reasons, MATLAB may have selected a different M-file from the one you had intended to run. It is recommended that you leave this warning enabled.

Differences Between Built-Ins and M-Functions Removed; Can Affect Which Function MATLAB Selects

MATLAB implements many of its core functions as built-ins. In previous releases of MATLAB, there have been several significant differences between the way MATLAB handles built-in and M-file functions. As of this release, MATLAB handles both types of functions the same. This change affects the following:

- “Function Dispatching” on page 212
- “Return Value from the functions Function” on page 213
- “Output from the which Function” on page 213

Function Dispatching

MATLAB now dispatches both built-in and M-file functions according to the same precedence rules, (see “Function Precedence Order” in the *Programming and Data Types* section of the MATLAB documentation). In previous releases, subfunctions, private functions, and class constructor functions took precedence over M-functions of the same name, but not over built-ins. In this release, built-in functions follow the same rules given to M-functions, and thus are lower in precedence than the three function types named above.

This change addresses a potential problem in that changes to the internal implementation of MATLAB functions could potentially affect the operation of your own M-code. For example, if a new version of MATLAB were to change an internal function from being M-based to being built-in, the function in the new version would now be subject to different precedence rules. If one of your M-code modules had a subfunction with the same name as this function (now obeying the built-in rules), then this subfunction would never be called.

This release resolves this potential conflict by using the same precedence rules for both M-functions and built-ins.

Compatibility Considerations. If any of your programs have subfunctions, private functions, or class constructor functions that have the same name as a built-in function that has the same function scope, MATLAB now gives precedence to the subfunction, private function, or constructor rather than the built-in. If this is not corrected, you may find instances where your program calls a function other than the one you had intended. You can avoid such problems by renaming any functions that may conflict with a MATLAB built-in function.

Return Value from the functions Function

The MATLAB `functions` function returns information about a function handle such as the function name, type, and filename. In previous releases, functions returned the filename for a built-in function as the string

```
'MATLAB built-in function'
```

In this release, MATLAB associates each built-in function with a placeholder file that has a `.bi` extension (for example, `reshape.bi` for the built-in `reshape` function).

Output from the which Function

The `which` function now displays the pathname for built-in functions, as well as for overloaded functions when only the overloaded functions are available.

Warning on Naming Conflict

The following warning was added to identify the case when you first use a name as a function and later use it as a variable:

```
Warning: File: D:\Work\MATLAB XL\theworks\my_yprime.m Line: 17  
Column: 1 Variable 'getdata' has been previously used as a  
function name.  
(Type "warning off MATLAB:mir_warning_variable_used_as_function:  
to suppress this warning.)
```

For example, this code generates such a warning:

```
X = i; % Calls the function i() to get sqrt(-1)  
for i = 1:10 % uses i as a variable. This produces the warning.  
... end
```

Change to How `evalin` Evaluates Dispatch Context

In Release 13 and earlier, the `evalin` function evaluated its input in the specified workspace, but not the workspace's corresponding dispatching context. Hence, running the following example used to succeed, calling the subfunction `MySubfun` but using the value of `x` from the base workspace:

```
function demo
    evalin('base', 'MySubfun(x)')

function MySubfun(in)
    disp(in)
```

When you call `evalin` in Release 14, MATLAB tries to find a function named `MySubfun` that is accessible in the base workspace, i.e. at the command prompt. Since `MySubfun` is a subfunction and therefore not in scope at the command prompt, MATLAB errors, reporting that `MySubfun` is undefined.

Compatibility Considerations

There are two ways to change your existing code to work with this new behavior. First, if your code only needs to get the value of the subfunction's inputs from the base workspace (as `demo.m` does above), and does not care what context `MySubfun` is run in, then you can change your code to use `evalin` only to get the values of the inputs from the base workspace, like this:

```
function demo_workaround1
    MySubfun(evalin('base', 'x'))

function MySubfun(in)
    disp(in)
```

If, however, it is important that the subfunction itself be run in the context of the base workspace, you can place a function handle to the subfunction in the base workspace and then evaluate that:

```
function demo_workaround2
    assignin('base', 'MySubfunHandle', @MySubfun);
    evalin('base', 'MySubfunHandle(x)')

function MySubfun(in)
    disp(in)
```

You can also substitute 'caller' for 'base' in the workaround code if your original code uses `evalin('caller', ...)`.

Summary of New Functions

These functions are new in this release.

Function	Description
<code>addtodate</code>	Modify a particular field of a date number
<code>genvarname</code>	Construct valid variable name from string
<code>intmax</code>	Return largest possible integer value
<code>intmin</code>	Return smallest possible integer value
<code>intwarning</code>	Control state of integer warnings
<code>isfloat</code>	Detect floating-point arrays
<code>isinteger</code>	Detect whether an array has integer data type
<code>isscalar</code>	Determine if item is a scalar
<code>isstrprop</code>	Determine the content of each element of a string
<code>isvector</code>	Determine if item is a vector
<code>mmfileinfo</code>	Get information about multimedia file
<code>recycle</code>	Set option to move deleted files to recycle folder
<code>restoredefaultpath</code>	Restore default search path
<code>strtrim</code>	Remove leading and trailing whitespace from string
<code>textscan</code>	Read data from text file, convert and write to cell array
<code>xlswrite</code>	Write matrix to a Microsoft Excel spreadsheet

New Function Type – Anonymous Functions

Anonymous functions give you a quick means of creating simple functions without having to create M-files each time. You can construct an anonymous function either at the MATLAB command line or from within another function or script.

Refer to “Anonymous Functions” in the MATLAB Programming documentation for more complete coverage of this topic. For more information on anonymous functions, open the M-file `anondemo.m` in the MATLAB Editor by typing

```
edit anondemo
```

Syntax

The syntax for creating an anonymous function from an expression is

```
fhandle = @(arglist) expr
```

where `arglist` is a comma-separated list of input variables, and `expr` is any valid MATLAB expression. The constructor returns a function handle, `fhandle`, that is mapped to this new function. Creating a function handle for an anonymous function gives you a means of invoking the function. It is also useful when you want to pass your anonymous function in a call to some other function.

Note Function handles not only provide access to anonymous functions. You can create a function handle to any MATLAB function. The constructor uses a different syntax: `fhandle = @functionname` (e.g., `fhandle = @sin`). To find out more about function handles, see “Function Handles” in the MATLAB Programming documentation.

You can use the function handle for an anonymous function in the same way as any other MATLAB function handle.

A Simple Example

To create a simple function `sqr` to calculate the square of a number, use

```
sqr = @(x) x.^2;
```


To execute the function, type the name of the function handle, followed by any input arguments enclosed in parentheses:

```
a = sqr(5)
a =
    25
```

Since `sqr` is a function handle, you can pass it to other functions. The code shown here passes the function handle for anonymous function `sqr` to the MATLAB `quad` function to compute its integral from zero to one:

```
quad(sqr, 0, 1)
ans =
    0.3333
```

Arrays of Anonymous Functions

To store multiple anonymous functions in an array, use a cell array. See “Arrays of Anonymous Functions” in the MATLAB Programming documentation.

Examples

You can find more examples of how to use anonymous functions in MATLAB under “Examples of Anonymous Functions”.

New Function Type — Nested Functions

You can now define one or more functions within another function in MATLAB. These inner functions are said to be *nested* within the function that contains them. You can also nest functions within other nested functions.

Refer to “Nested Functions” in the MATLAB Programming documentation for more complete coverage of this topic. For more information on nested functions, open the M-file `nesteddemo.m` in the MATLAB Editor by typing

```
edit nesteddemo
```

Writing a Nested Function

To write a nested function, simply define one function within the body of another function in an M-file. Like any M-file function, a nested function contains any or all of the usual function components. In addition, you must always terminate a nested function with an end statement:

```
function x = A(p1, p2)
...
    function y = B(p3)
        ...
    end
...
end
```

Characteristics of Nested Functions

Two characteristics unique to nested functions are

- A nested function has access to the workspaces of all functions inside of which it is nested. A variable that has a value assigned to it by the primary function can be read or overwritten by a function nested at any level within the primary. Similarly, a variable that is assigned in a nested function can be read or overwritten by any of the functions containing that function.
- When you construct a function handle for a nested function, the handle not only stores the information needed to access the nested function; it also stores the values of all variables shared between the nested function and those functions that contain it. This means that these variables persist in memory between calls made by means of the function handle.

Examples

You can find examples of how to use nested functions in MATLAB under “Examples of Nested Functions”.

Calling Private Functions From Scripts

You can now invoke a private function from a script, provided that the script is called from another M-file function, and that the private function being called by the script is within the scope of this M-file function.

New Calling Syntax for Function Handles; Replace `eval` With New Syntax

You can now call functions by means of their related function handles using standard calling syntax rather than having to use `feval`. When calling a function using its handle, specify the function handle name followed by any input arguments enclosed in parentheses.

For example, if the handle to a function was stored in variable `h`, you would call the function as if the handle `h` were a function name:

```
h(arg1, arg2, ...)
```

For the parabola function shown here, construct a function handle `h` and call the parabola function by means of the handle:

```
function y = parabola(a, b, c, x)
y = a*x.^2 + b*x + c;

parabHandle = @parabola;

parabHandle(1.3, .2, 30, 25)
```

When calling functions that take no input arguments, you must use empty parentheses after the function handle:

```
parabHandle()
```

Compatibility Considerations

Using `feval` for the purpose of invoking functions via function handle is no longer necessary and is, in fact, slower. However, for purposes of backward compatibility, the use of `feval` to evaluate function handles is still supported in this release.

Parenthesis notation on a non-scalar function handle means subscripting, just as in Release 13, while the same notation on scalar function handles means function call, as described above. Incompatibility can arise only if you construct a scalar array of function handles and actually index it, necessarily with an index of 1.

Arrays of Function Handles

Previous releases of MATLAB supported arrays of function handles. You created such an array using the `[]` operator, and indexed into the array with the `()` operator:

```
x = [@sin @cos @tan];  
plot(feval(x(2), -pi:.01:pi));
```

In Release 14, MATLAB supports arrays of functions handles using cell arrays. You create and index into a function handle array using the `{}` operator:

```
x = {@sin @cos @tan};  
plot(x{2}(-pi:.01:pi));
```

Compatibility Considerations

For purposes of backward compatibility, standard arrays of function handles are still supported in this release.

Calling `nargin` and `nargout` with Built-In Functions

When you pass the name of a function to `nargin` or `nargout`, MATLAB returns the number of declared inputs or outputs for that function. For example, passing the function name 'normest' to `nargin` returns 2, (the number of inputs declared in `normest.m`):

```
nargin('normest')      % normest is an M-function.  
ans =  
     2
```

In this release, you can also use this feature with MATLAB built-in functions. The following use of `nargin` returned an error in previous versions of MATLAB because `norm` is implemented as a built-in function. In this release, `nargin` returns the number of inputs declared by the `norm` function:

```
nargin('norm')        % norm is a built-in function.  
ans =  
     2
```

The same applies to `nargout`.

getfield and setfield Not To Be Deprecated

There are no plans to remove the `getfield` and `setfield` functions from the MATLAB language, as stated in the release notes for MATLAB Release 13.

isglobal Function To Be Discontinued

Support for the `isglobal` function will be removed in a future release of MATLAB. In Release 14, invoking `isglobal` generates the following warning:

```
Warning: isglobal is obsolete and will be discontinued.  
Type "help isglobal" for more details.
```

Recycle to Protect Files from Unwanted Deletion

To protect yourself from unintentionally deleting any files that you want to keep, use the new `recycle` function to turn on file recycling. When file recycling is on, MATLAB moves all files that you delete with the `delete` function to either the recycle bin (on the PC or Macintosh) or a temporary folder (on UNIX). When file recycling is off, any files you delete are actually removed from the system.

You can turn recycling on for all of your MATLAB sessions using the **Preferences** dialog box (Select **File** -> **Preferences** -> **General**). Under the heading **Default behavior of the delete function**, select **Move files to the Recycle Bin**.

bin2dec Ignores Space Characters

The `bin2dec` function now ignores any space (' ') characters in the input string. Thus, the binary string '010 111' now yields the same result as the string '010111'.

In Release 13, `bin2dec` interpreted space characters as zeros:

```
bin2dec('010 111')  
ans =  
    39
```

In this release, `bin2dec` ignores all space characters:

```
bin2dec('010 111')  
ans =  
    23
```

dbstop crashes Are Now Resolved

In previous versions, a MATLAB session would terminate prematurely when attempting to execute certain P-code files if you had set a debugger breakpoint in the function represented by that file. For example, an attempt to run Guide would terminate your MATLAB session if you had used the `dbstop` function to set a breakpoint in the corresponding M-file:

```
dbstop in guide
guide
```

This bug has been fixed in this release enabling you to debug these files successfully.

Bit Functions on Unsigned Integers

MATLAB bit functions now work on unsigned integers. Instead of using flints (integer values stored in floating point) to do your bit manipulations, consider using unsigned integers. See “Bit Functions Now Work on Unsigned Integers” in the MATLAB Mathematics release notes.

inmem Returns Path Information

The `inmem` function now returns not only the names of the currently loaded M- and MEX-files, but the path and filename extension for each as well. Use the `-completenames` option to obtain this additional information:

```
inmem('-completenames')
```

New Features for Nondouble Data Types

The section *New Nondouble Mathematics Features* describes new features affecting the `nondouble` (`single` and `integer`) data types. These changes affect `single` and `integer` arithmetic operations, and also conversion of `single` and `double` data types to integers.

Mathematic Operations on Logical Values

Most mathematic operations are not supported on logical values.

Accessing Cell and Structure Arrays Without deal

In many instances, you can access the data in cell arrays and structure fields without using the `deal` function. Here is an example that reads each of the cells of a cell array into a separate output:

```
C = {rand(3) ones(3,1) eye(3) zeros(3,1)};
```

Use either of the following to access the cells in `C`:

```
[a,b,c,d] = deal(C{:})  
[a,b,c,d] = C{:}
```

Here is an example that reads each of the fields of a structure array into a separate output:

```
A.name = 'Pat'; A.number = 176554;  
A(2).name = 'Tony'; A(2).number = 901325;
```

Use either of the following to access the name field:

```
[name1,name2] = deal(A(:).name)  
[name1,name2] = A(:).name
```

New Features in Regular Expression Support

This version of MATLAB introduces the following new features in regular expression support:

- **Multiple Input Strings** — You can use any of the MATLAB regular expression functions with cell arrays of strings as well as with single strings. Any or all of the input parameters (the string, expression, or replacement string) can be a cell array of strings.
- **Selective Outputs** — To select what type of data you want the `regexp` and `regexpi` to return (string indices or text, token indices or text, or token data by name) use one or more of the six qualifiers for these functions.
- **Lookaround Operators** — Lookahead and lookbehind operators enable you to match a pattern only if it is preceded, or followed, by another pattern.
- **New Logical Operators** — New operators for grouping, inserting comments, and finding alternative match patterns
- **New Quantifiers** — Lazy quantifiers match a minimum number of characters in a string. Possessive quantifiers do not reevaluate parts of the string that have already been evaluated
- **Element Grouping** — Group elements together using either `(...)` to group and capture, or `(?:...)` for grouping alone
- **Named Capture Grouping** — Capture characters in a token and assign a name to the token
- **Conditional Expressions** — Process a string in different ways depending on a stated condition
- **New character representations** — New symbolic representations such as `\e` for escape, or `\xN` for a character of hexadecimal value N, are available in this release.
- **Default Tokenizing** — The `regexprep` function now tokenizes by default. There is no longer a `'tokenize'` option

Refer to “Regular Expressions” in the MATLAB Programming documentation.

Functions that Use Regular Expressions

The `who`, `whos`, `save`, `load`, and `clear` functions now accept regular expressions as input. This feature enables you to be more selective concerning which variables they operate on.

For example, this statement saves to a MAT-file only those variables with a name that either starts with the letters A or B, or contains ten or more characters:

```
save('mydata.mat', '-regexp', '^([AB].|'.{10,})');
```

If the workspace contains the following four variables, two of the four meet the requirements of the regular expression:

```
whos
  Name                Size          Bytes  Class
  A_stats             10x5           400  double array
  X23456789           1x1            12  char array
  ab                  3x1           536  struct array
  longerVariableName  1x4             8  char array
```

When you perform the save operation and then check the contents of the MAT-file, you see that the variables with names that either start with A or have at least ten characters were saved:

```
save('mydata.mat', '-regexp', '^([AB].|'.{10,})');
```

```
whos -file mydata.mat
  Name                Size          Bytes  Class
  A_stats             10x5           400  double array
  longerVariableName  1x4             8  char array
```

Refer to the reference pages for these functions for more information and examples.

Regular Expressions Accept String Vector; No Longer Support Character Matrix Input

You can now pass a vector of strings in a cell array to any of the MATLAB regular expression functions (`regexp`, `regexp_i`, and `regprep`).

Compatibility Considerations

Because this is the preferred method of passing a string vector, MATLAB no longer supports using character matrices for this purpose.

Cell Array Support for String Functions

You can now pass a cell array of strings to the `strfind` function. MATLAB searches each string in the cell array for occurrences of the pattern string, and returns the starting index of each such occurrence.

Additional Class Output From `mat2str`

The statement `str = mat2str(A, 'class')` creates a string with the name of the class of `A` included. This option ensures that the result of evaluating `str` will also contain the class information.

Change the 16-bit integer matrix to a string that includes `'int16'`. Next, evaluate this string and verify that you get the same matrix that you started with:

```
x1 = int16([-300 407 213 418 32 -125]);

A = mat2str(x1, 'class')
A =
    int16([-300 407 213 418 32 -125])
x2 = eval(A);

isa(x2, 'int16') && all(x2 == x1)
ans =
     1
```

String Properties

Use the new `isstrprop` function to see what parts of a string or array of strings are alphabetic, alphanumeric, numeric digits, hexadecimal digits, lowercase, uppercase, white-space characters, punctuation characters, contain control characters, or contain graphic characters.

For example, to test for alphabetic characters in a two-dimensional cell array, use

```
A = isstrprop({'abc123def'; '456ghi789'}, 'alpha')
A =
    [1x9 logical]
    [1x9 logical]

A{:,:}
ans =
    1 1 1 0 0 0 1 1 1
    0 0 0 1 1 1 0 0 0
```

Using `strtok` on Cell Arrays of Strings

You can now use the `strtok` function on a cell array of strings. When used with a cell array of strings, `strtok` returns a token output that is also a cell array of strings, each containing a token for its corresponding input string.

See the `strtok` reference page to see an example of how this works.

Colon Operator on `char` Now Returns a `char`

Applying the colon operator to inputs of type `char` now returns a result of type `char`. For example,

```
'a':'g'

ans =

abcdefg
```

Compatibility Considerations

In previous releases, the same operation returned a result of type `double`. You may need to change your code if it relies on type `double` being returned.

datestr Returns Date In Localized Format

The statement `str = datestr(..., 'local')` returns the date string in a localized format. See the `datestr` reference page for more information.

Form and Locale for weekday

The `weekday` function now takes two new inputs that control the output format. These arguments enable you to get a full or abbreviated day name, and a local or US English output.

Freestyle Date String Format

When converting between serial date numbers, date vectors, and date strings with the `datenum`, `datevec`, and `datestr` functions, you can specify a format for the date string from the Free-Form Date Format Specifiers table shown on the `datestr` reference page.

Reading Date Values with xlsread; Conversion No Longer Necessary

There are two changes that affect importing date information from Excel with the `xlsread` function:

- Date Information Returned as Cell Array of Char
- Conversion of Date Values Is No Longer Necessary

Date Information Returned as Cell Array of Char

Prior to this release, `xlsread` imported date information from an Excel file and returned the results as a `double`. In Release 14, `xlsread` returns this information as a cell array containing data of class `char`. The reason for this is that MATLAB now imports Excel files using an Excel COM server. Excel returns dates as strings, and there is really no indication that what is returned is a date.

Compatibility Considerations. You will need to change your program code to accept a cell array of type `char` instead of an array of `double` when using `xlsread` to import date information from Excel.

Conversion of Date Values Is No Longer Necessary

When reading date fields from a Microsoft Excel file using earlier versions of MATLAB, it was necessary to convert the Excel date values into MATLAB date values. This was necessary because Excel and MATLAB calculated date values based on a different reference date. This is explained in the section, “Handling Excel Date Values” in the function reference for `xlsread`.

With MATLAB 7.0, you no longer have to do this conversion because `xlsread` now imports dates as strings rather than as numerical values.

Compatibility Considerations. If your existing code converts Excel date values to MATLAB values, you will need to remove this step so that you end up with the correct results.

Comprehensive Function for Reading Text Files

The new `textscan` function reads data from an open text file into a cell array. MATLAB parses the data into fields and converts it according to conversion specifiers passed to `textscan` in the argument list.

The `textscan` function is similar to `textread` but differs from `textread` in the following ways:

- The `textscan` function offers better performance than `textread`, making it a better choice when reading large files.
- With `textscan`, you can start reading at any point in the file. Once the file is open, (`textscan` requires that you open the file first), you can seek to any position in the file and begin the `textscan` at that point. The `textread` function requires that you start reading from the beginning of the file.
- Subsequent `textscan` operations start reading the file at the point where the last `textscan` left off. The `textread` function always begins at the start of the file, regardless of any prior `textread`.
- `textscan` returns a single cell array regardless of how many fields you read. With `textscan`, you don't need to match the number of output arguments to the number of fields being read as you would with `textread`.
- `textscan` offers more choices in how the data being read is converted.
- `textscan` offers more user-configurable options.

New Inputs and Outputs to `xlsread`

The table below shows new input and output arguments to the `xlsread` function. See the function reference for `xlsread` for more information. With the exception of the **basic** input argument, these arguments are supported only on computer systems capable of starting Excel as a COM server from MATLAB.

New Input Arguments	Description
-1	Opens the Excel file in an Excel window, enabling you to interactively select the worksheet to be read and the range of data to import from the worksheet.
range	Reads data from the rectangular region of a worksheet specified by range.
basic	Imports data from the spreadsheet in basic import mode.

New Output Argument	Description
rawdata	Returns unprocessed cell content in a cell array. This includes both numeric and text data.

New Inputs and Syntax for `dlmwrite`

The `dlmwrite` function now has several new input arguments plus an optional attribute-value format in which to enter these arguments. You can now enter input arguments to `dlmwrite` in an attribute-value format. This format enables you to specify just those arguments that you need and omit any others. This new syntax for `dlmwrite` is

```
dlmwrite('filename', M, attribute1, value1, ...
        attributeN, valueN)
```

The former syntax for `dlmwrite` is still supported for arguments that were available in earlier versions of MATLAB.

The table below shows new input arguments to the `dlmwrite` function. You must specify these new arguments using the attribute-value format. See the function reference for `dlmwrite` for more information.

Attribute	Value
append	Either overwrite or append to the file
delimiter	Delimiter string to be used in separating matrix elements
newline	Character(s) to use in terminating each line
roffset	Offset, in rows, from the top of the destination file to where matrix data is to be written
coffset	Offset, in columns, from the left side of the destination file to where matrix data is to be written
precision	Numeric precision to use in writing data to the file

For example, to export matrix `M` to file `myfile.txt`, delimited by the tab character, and using a precision of six significant digits, type

```
dlmwrite('myfile.txt', M, 'delimiter', '\t', 'precision', 6)
```

Change in Output from `xlsfinfo`

`xlsfinfo` now returns the names of all worksheets in an Excel file instead of just the ones with numbers in them (as in Release 13).

Importing Complex Arrays

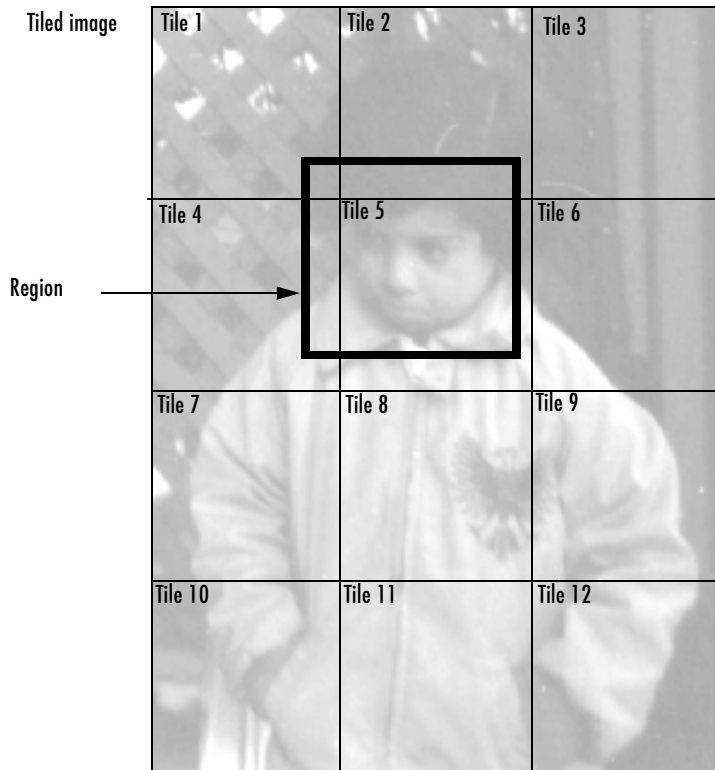
The `csvread`, `dlmread`, and `textscan` functions import any complex number as a whole into a complex numeric field, converting the real and imaginary parts to the specified numeric type. Valid forms for a complex number are

Form	Example
<code>-<real>-<imag>i j</code>	<code>5.7-3.1i</code>
<code>-<imag>i j</code>	<code>-7j</code>

Using `imread` to Import Subsets of TIFF Images

When using the `imread` function with the `'PixelRegion'` parameter, you can now read in portions of an image stored in TIFF format. Specify a cell array containing two vectors, `ROWS` and `COLS`, for the value of this parameter. Each vector can either be a two-element vector specifying the extent of the region, `[START STOP]`, or a three-element vector that enables downsampling, `[START INCREMENT STOP]`.

When used with tiled images, `'PixelRegion'` subsetting can improve memory usage and performance because it only reads in the tiles that encompass the region. For example, in the following figure, if you specify the region defined by the box, `imread` would only read in tiles 1, 2, 4, and 5.



Getting Information about Multimedia Files

MATLAB now includes a function, named `mmfileinfo`, that returns information about the contents of a multimedia file. The file can contain audio data, video data, or both.

This function is only available on Windows platforms.

All-Platform Audio Recording and Playback

The MATLAB `audiorecorder` and `audioplayer` functions can now be used on Windows and UNIX platforms. These functions were previously only available on Windows systems.

Note The `audiorecorder` and `audioplayer` objects are now implemented as MATLAB objects on all platforms. The methods supported by these objects are overloaded functions. You must use standard MATLAB function calling syntax to call methods of these objects; you cannot use dot notation.

FTP File Operations

From within MATLAB, you can connect to an FTP server to perform remote file operations. For more information, see the `ftp` reference page.

Web Services (SOAP)

MATLAB can now consume Simple Object Access Protocol-based (SOAP) Web services with the `createClassFromWSDL` function. For more information, see “Using Web Services in MATLAB” in the online documentation.

64-Bit File Handling on Macintosh

The release notes for MATLAB Release 13 should have included Macintosh in the list of those platforms that support 64-bit file handling. This support is available on the following platforms:

- Windows
- Solaris
- Linux 2.4.x
- HP-UX 11.0, 9000/785
- Macintosh

Changes to Error Message Format

The last two lines of MATLAB error messages have changed for Release 14. Error messages now

- Display functions and subfunctions differently than in R13.
- Display nested functions.
- Call out the error in a string that you can use as input to other functions, like `dbstop`.
- Have a hot link to the source of the error.

Each of these changes is discussed below. Examples show the errors generated by both the previous release (V6.5) and current release (7.0) of MATLAB for the purpose of comparison.

Display of Functions and Subfunctions

MATLAB now calls out the source of the error using a consistent format. One of the features of this format is that you can place the string of the message into other MATLAB commands. See “Using the Error Message String as Input to Other Functions” on page 236.

Errors Generated by the Primary Function. Errors generated by the primary function of an M-file are displayed as shown below. In version 7.0, the path is not shown in most cases (private functions are one exception). Filename extension is also not shown. The failing line number is shown on third line.

In MATLAB V6.5 —

```
??? Error using ==> strcmp
Too many input arguments.
```

```
Error in ==> B:\MATLAB_V70\work\errmsgtest.m
On line 11 ==> strcmp('aa','bb','cc');
```

In MATLAB V7.0 —

```
??? Error using ==> strcmp
Too many input arguments.
```

```
Error in ==> errmsgtest at 11
strcmp('aa','bb','cc');
```

Errors Generated by a Subfunction. Errors generated by a subfunction of an M-file are displayed in the previous release and current release of MATLAB as shown below. Comments for primary functions apply here as well. Also, the name of the failing subfunction follows the > character instead of being put in parentheses.

In MATLAB V6.5 —

```
??? Error using ==> strcmp
Too many input arguments.
```

```
Error in ==> B:\MATLAB_V70\work\errmsgtest.m (subFun1)
On line 17 ==> strcmp('aa','bb','cc');
```

In MATLAB V7.0 —

```
??? Error using ==> strcmp
Too many input arguments.
```

```
Error in ==> errmsgtest>subFun1 at 17
strcmp('aa','bb','cc');
```

Error Messages Display Nested Functions

This example shows an error that comes from a nested function (`nestFun2`) called by another nested function (`nestFun1`). It uses the following syntax, where the `>` character follows the name of the primary function and precedes the names of any nested functions.

```
fun>nestfun1/nestfun2/etc at lineno.
```

In MATLAB V6.5 —

Nested functions are not supported prior to version 7.0.

In MATLAB V7.0 —

```
??? Error using ==> strcmp
Too many input arguments.
```

```
Error in ==> errmsgtest>nestFun1/nestFun2 at 6
strcmp('aa','bb','cc');
```

Using the Error Message String as Input to Other Functions

You can copy the text of the line that calls out the source of an error and use this string as input to some of the MATLAB debugging functions. The example shown below uses the string in a call to the `dbstop` function.

Copy the text that begins after

```
Error in ==>
```

In MATLAB V6.5 —

This feature is not supported prior to version 7.0.

In MATLAB V7.0 —

```
??? Error using ==> strcmp
Too many input arguments.
```

```
Error in ==> errmsgtest>nestFun1/nestFun2 at 6
strcmp('aa','bb','cc');
```

Copy and paste text of this error message into the `dbstop` command:

```
dbstop errmsgtest>nestFun1/nestFun2 at 6
```

Hot Link to the Source of an Error

Error messages now contain a blue-underlined hot link to the failing line of the M-file being executed.

Compatibility Considerations

If any of your programs rely on specific text in the types of error messages described here, you may have to modify your program code.

nargchk Has a New Format for Error Messages

When the `nargchk` function detects an error condition, it returns information on the error in either a string or a MATLAB structure. Use one of these two command syntaxes to specify which format to return. If neither is specified, `nargchk` returns a string:

```
msgstring = nargchk(minargs, maxargs, numargs, 'string')
msgstruct = nargchk(minargs, maxargs, numargs, 'struct')
```

The return structure has two fields: the message string, and a message identifier. When too few inputs are supplied, these fields are

```
message: 'Not enough input arguments.'
identifier: 'MATLAB:nargchk:notEnoughInputs'
```

When too many inputs are supplied, the structure fields are

```
message: 'Too many input arguments.'
identifier: 'MATLAB:nargchk:tooManyInputs'
```

Enabling and Disabling Warning Messages

The following message, which MATLAB appended to all warning messages in the previous release, is no longer displayed.

(Type "warning off <msgid:msgstr>" to suppress this warning.)

Use the off and on options of the warning function to control the display of all or selected warnings. This example disables a selected warning, and then enables all warnings:

```
% All warnings are enabled by default.
A = 5/0;
Warning: Divide by zero.

% Disable the most recent warning
[msgstr msgid] = lastwarn;
warning('off', msgid);

% Try it again. This time there is no warning.
A = 5/0;

% Enable all warnings
warning('on', 'all')

% Verify that the warning is reenabled.
A = 5/0;
Warning: Divide by zero.
```

Catching Ctrl+C in try-catch Statements

In previous releases of MATLAB, typing **Ctrl+C** while executing the try part of a try-catch statement resulted in the program branching to the catch part of that statement. In this release, typing **Ctrl+C** is purposely not caught by try-catch statements.

The reason for this change is that, under certain circumstances, this behavior in try-catch statements was found to adversely affect internal MATLAB code. In these cases, this resulted in MATLAB code catching the **Ctrl+C** rather than responding appropriately to it by terminating the current operation.

MATLAB Performance Acceleration

Release 13 introduced a new performance acceleration feature built into MATLAB. Enhancing performance in MATLAB is an ongoing development project that continues to show significant improvements in the performance of MATLAB programs.

The Performance Acceleration documentation written for Release 13 included suggestions on specific techniques to make the most of this feature. In this release, many of those techniques are no longer necessary. This documentation has been replaced with more general suggestions on how to improve the performance of your programs.

“Using MATLAB” Documentation Is Now Three Books

Due to the increasing size of the printed “Using MATLAB” manual, we have divided it up into three separate printed books in version 7.0 to make it more manageable. The titles for these books (and their corresponding headings in the MATLAB Help Browser) are

- Desktop Tools and Development Environment
- Mathematics
- Programming

The online structure of this documentation is very similar to what it has been in previous releases, although some topics are now covered more thoroughly. We hope that you find this new format easier to use.

Graphics and 3-D Visualization, MATLAB Version 7 (R14)

If you are using the Help browser, view the Graphics new features video demo to see highlights of the new features.

This version introduces the following new features and changes:

- Plotting Tools
- Code Generation
- Data Exploration Tools
- Annotation Features
- Plot Objects
- Group Objects
- Linking Graphics Object Properties
- New Behavior for Hold Command
- Enhancements to findobj
- New Axes Properties
- New Figure Properties
- New Rootobject Property

Plotting Tools

If you are using the Help browser, watch the new Plotting Tools video demo for an overview of the major functionality.

The following list links to new or redesigned plotting tool features.

- Figure Toolbars — figure toolbars that provide data exploration, plot editing, and annotation tools
- Interactive Plotting Tools — overview of plotting tools
 - Figure Palette
 - Plot Browser
 - Property Editor

Related functions:

- `plottools`
- `figurepalette`
- `plotbrowser`
- `propertyeditor`

Code Generation

You can save a graph as an M-file that contains the code to regenerate the graph. See [Generating an M-File to Recreate a Graph](#) for more information.

Data Exploration Tools

The following list links to the documentation for the data exploration tools.

- [Data Cursor](#) — displaying data values interactively
- [Zooming](#) — 2-D and 3-D zoom tools
- [Panning](#) — repositioning your view of the graph
- [Rotate 3D](#) — interactive rotation of 3-D views
- [Camera Toolbar](#) — mouse-controlled 3-D view manipulation.

Annotation Features

The following list links to the documentation for annotation features and properties of the annotation objects.

- Overview of annotation features
- Rectangles and ellipses
 - Rectangle properties
 - Ellipse properties
- Textbox annotations
 - Textbox properties
- Lines and arrows
 - Line properties
 - Arrow properties
 - Textarrow properties
 - Doublearrow properties
- Adding a Colorbar to a graph — new positioning options and colormap modification.
colorbar — new command options
- Adding a legend to a graph — new positioning and appearance options
legend — new command options
- Pinning — attaching annotation objects to a point in the figure
- Aligning and Distributing graphics objects

See the `annotation` function for information on programmatic access to annotation objects.

See Annotation Objects for an overview of this type of graphics object.

Plot Objects

Plot Objects are composite graphics objects that simplify the modification of graphs that employ them. The following list links to reference pages for modified graphing functions and to property descriptions of the new plot objects.

See Plot Objects for an overview of this type of graphics object.

Functions That Use Plot Objects

- area
- bar
- contour
- errorbar
- plot, plot3, loglog, semilogx, semilogy
- quiver, quiver3
- scatter, scatter3
- stairs
- stem, stem3
- surf, and mesh group

Note that all of the above functions have a 'v6' optional argument that causes each function to return the core graphics objects that were created in previous releases. See the reference pages for more information.

Plot Objects

- areaseries
- barseries
- contourgroup
- errorbarseries
- lineseries
- quivergroup
- scattergroup
- stairseries
- stemseries
- surfaceplot

Refreshing Data Source Properties

The refreshdata function enables you to take advantage of the XDataSource, YDataSource, and ZDataSource plot objects properties to update graph data when workspace variables change values.

See Specifying a Data Source for more information.

Group Objects

Group objects enable you to treat a number of objects as one, with respect to certain properties.

See `Group Objects` for an overview of this type of graphics object.

Group Object Functions

- `hgroup`
- `hgtransform`
- `makehgtform`

Linking Graphics Object Properties

You can link the corresponding properties of graphics objects so that changing any one object's properties makes the same change to all the linked objects.

See `linkprop` and `linkaxes` for more information.

New Behavior for Hold Command

The `hold` command has a new option `all`. This option holds the plot and the current line color and line style so that subsequent graphing commands do not reset the `ColorOrder` or `LineStyleOrder` property values to the beginning of the list.

Enhancements to findobj

The `findobj` function now supports logical operators and regular expressions. See the `findobj` reference page for more information.

New Axes Properties

You can control the behavior of an axes within a resized figure using the following new axes properties.

- `OuterPosition` — The boundary of the axes including the axis labels, title, and a margin. For figures with only one axes, this is the interior of the figure.
- `ActivePositionProperty` — Specifies whether to use the `OuterPosition` or the `Position` property as the size to preserve when resizing the figure containing the axes.
- `TightInset` — The margins added to the width and height of the `Position` property to include text labels, title, and axis labels.

See [Automatic Axes Resize](#) for more information.

New Figure Properties

There are three new figure properties described below.

Figure `KeyPressFcn` Property

The figure `KeyPressFcn` property now supports an event structure that returns information about the key press event. See the `KeyPressFcn` description for more information.

`DockControls` and `WindowState` Properties

Figures now have a `DockControls` property that determines if the **Desktop** menu appears on the figure. Setting `dockable` to `on` causes the menu to be displayed, the default setting of `off` prevents the menu from being displayed. You can always dock and undock the figure by setting the figure `WindowState` property.

Note that, depending on your preference settings, the figure might first be grouped into a Document window, which can then be docked in the Desktop.

See [Docking Figures in the Desktop](#) for more information.

New `Rootobject` Property

The `MonitorPosition` property enables you to get the position (width, height, and location) of multiple monitors connected to your computer.

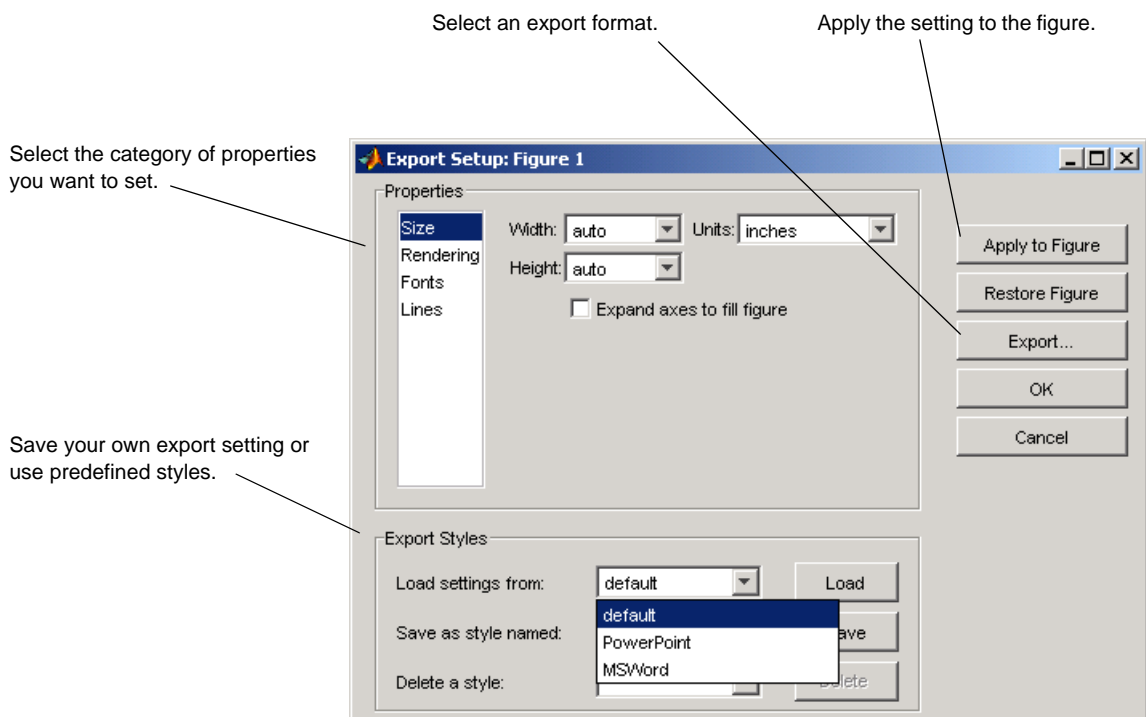
New Dialog for Exporting Figures

You can export MATLAB figures to a variety of standard file formats using the Export Setup dialog. To display the dialog, select **Export Setup** from the figure **File** menu.

Export Setup provides easy access to the graphics properties that affect exported figures. For example, it enables you to control the size of the figure, the font size and type, whether to use line styles or solid lines, and so on.

You can save your own export setting as an export style, or you can use predefined options optimized for PowerPoint and MSWord.

The following picture shows the major components of the Export Setup dialog.



Compatibility Considerations

Plotting Tools Not Working on Macintosh

The plotting tools 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.

Using `-nojvm` Option Prevents Plotting Tools Use

If you use the `-nojvm` option when starting MATLAB, the plotting tools are not available.

MATLAB 6 Version Property Editor

The MATLAB 6 Property Editor is being replaced by a new Property Editor, which is available in this release. However, you can still access the MATLAB 6 Property Editor by issuing the following command.

```
propedit(object_handle, 'v6')
```

Without the `v6` argument, `propedit` displays the new Property Editor. See the `propedit` function for more information. Note that the Property Editor might not work with all objects.

Cannot Dock Figures on Macintosh

You cannot dock figures in the Desktop, because MATLAB uses native figure windows on the Macintosh platform.

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 MATLAB.

Creating Graphical User Interfaces (GUIs), MATLAB Version 7 (R14)

This version introduces the following new features and changes:

- New Container Components
- ActiveX Controls
- New Toolbar Component
- Menu Editor Enhancements
- Layout Resize Behavior
- Key Press Detection
- Edit Text Box Scroll Bar
- Setting Uicontrol Focus
- Multiple Selection in `uigetfile`
- Program Suspension Time-Out
- Standard Dialog Box Push Buttons
- New Syntax for `uigetfile` and `uiputfile`
- Frames Not Available in GUIDE Layout Editor

New Container Components

MATLAB 7.0 introduces two new container components,

- Panel — Groups components
- Button group — Groups components and manages exclusive selection behavior for radio buttons and toggle buttons.

These components are available in the GUIDE Layout Editor and via the functions `uipanel` and `uibuttongroup`.

A container component can be the child of a figure or another container. In general, containers can have as children the same components as figures, including other containers. However, they cannot have menu bars, toolbars, or ActiveX controls as children. The `Position` property of the child of a panel or button group is interpreted relative to the panel or button group. If you move the panel or button group, the components it contains automatically move with it and maintain their positions.

Panel properties and button group properties enable you to control the color, size, border, and position of the panel or button group, assign a title, and specify a context menu. In general, panels and button groups have many of the same properties as `uicontrol` objects.

For information about working with panels and button groups in GUIDE, see the following topics in the Creating Graphical User Interfaces collection of the MATLAB documentation.

Compatibility Considerations

You can export a GUI that contains a panel or button group from GUIDE to a single M-file that does not require a FIG-file. However, you will not be able to run that M-file in MATLAB versions earlier than 7.0.

ActiveX Controls

GUIDE now enables you to insert an ActiveX control into your GUI if you are running MATLAB on Microsoft Windows. When you drag an ActiveX component from the component palette into the layout area, GUIDE displays a dialog in which you can select any registered ActiveX control on your system. When you select an ActiveX control and click **Create**, the control appears as a small box in the Layout Editor. You can then program the control to do what you want it to.

See [MATLAB COM Client Support](#) in the online MATLAB documentation and [ActiveX Controls](#) in the GUIDE documentation to learn more about ActiveX controls.

New Toolbar Component

A new function, `uitoolbar`, enables you to add a toolbar to a figure. You can add your own push tools and toggle tools to the toolbar with the `uipushtool` and `uitoggletool` functions.

`Uipushtool` properties enable you to provide a callback that responds to a mouse click. `Uitoggletool` properties enable you to provide callbacks that respond to the tool being set on, off, or toggled to either position. Properties for both `uipushtools` and `uitoggletools` provide for tooltip strings, separators, and truecolor images to display on the tools. In general, `uitoolbar`, `uipushtool`, and `uitoggletool` objects have many of the same properties as `uicontrol` objects.

Menu Editor Enhancements

The GUIDE Menu Editor now enables you to:

- Choose a keyboard accelerator for a menu item from a pop-up menu.
- Set an item's Enabled property on or off when the menu is first opened. If the property value is off, the item appears dimmed and the user cannot select it.
- Open the Property Inspector where you can change all uimenu properties.
- Display the callback subfunction in an editor. If the callback does not yet exist, GUIDE creates it before displaying it.

The Menu Editor is now better synchronized with other GUIDE tools:

- Property changes made in the Menu Editor or in the Property Inspector are immediately reflected in the other.
- uimenu objects in the Menu Editor now also appear in the Object Browser. If you select a uimenu object in either, it is automatically selected in the other.
- If a component is selected in the Layout Editor and you select a menu item in the Menu Editor, the component is deselected in the Layout Editor.

See Menu Editor in the MATLAB documentation for more information.

Layout Resize Behavior

In the GUIDE Layout Editor, components you have placed in the layout area now maintain their visual position relative to the upper left corner of their parent container (figure, panel, or button group) when you resize the container in the Layout Editor. However, the values of the Position property are determined relative to the lower left corner, and these values will change accordingly when you increase or decrease the height of the container.

Key Press Detection

A new uicontrol callback property, KeyPressFcn, specifies a key press callback function with which you can detect a key press when the callback's uicontrol object has focus. If no uicontrol has focus, the figure's key press callback function, if any, is invoked. This property is available in the uicontrol function and in GUIDE.

If you specify the `KeyPressFcn` property as an M-file, the callback routine can query the figure's `CurrentCharacter` property to determine what particular key was pressed and thereby limit the callback execution to specific keys. If you specify the `KeyPressFcn` property as a function handle, the callback routine can retrieve information about the key that was pressed from its `eventdata` structure argument.

As an example, you can use this property to enable a user to press **Enter**, rather than the space bar, after giving focus to a `uicontrol` push button. Use the push button's key press callback function to determine if the user pressed the **Enter** key. If it was the **Enter** key, call the push button callback.

See the `Uicontrol Properties` for more information.

Edit Text Box Scroll Bar

For `uicontrol` editable text fields, i.e. the `Style` property is set to `'edit'`, if `Max-Min>1`, then multiple lines are allowed. For multi-line edit boxes, a vertical scroll bar enables you to scroll the text. You can also use the arrow keys to scroll.

Setting Uicontrol Focus

The `uicontrol` function now enables you to transfer focus programmatically to a specified `uicontrol` object. The syntax `uicontrol(uich)` transfers focus to the `uicontrol` object with handle `uich`.

Multiple Selection in uigetfile

The `uigetfile` function can now create a dialog that enables the user to select and retrieve multiple files using the **Shift** and **Ctrl** keys. You can turn this capability on or off using the new `'MultiSelect'` parameter. The default setting is off.

Program Suspension Time-Out

A new `uiwait` argument, `timeout`, enables you to specify the number of seconds after which program execution will resume, unless `uiresume` is called first or the specified figure is deleted. For example,

```
uiwait(h,5)
```

causes the suspended program to resume execution, if it has not already, after five seconds.

Standard Dialog Box Push Buttons

For standard dialog boxes with more than one `uicontrol` push button, you can now give focus to another button while retaining the default button. Focus is denoted by a border or a dotted border, respectively, in UNIX and Microsoft Windows. The default button has a shadow.

In such a case, if the user presses the space bar, the button with focus gets the key press and can choose to execute its own callback or the callback of the default button. If the user presses **Enter**, the default push button gets the key press and its callback executes. This code provides an example.

```
ButtonName=questdlg('What is your wish?', ...  
                    'Genie Question', ...  
                    'Food', 'Clothing', 'Money', 'Money')
```

New Syntax for `uigetfile` and `uiputfile`

The `uigetfile` and `uiputfile` syntax that enables you to position dialog boxes that are used to retrieve and save files is changed. The new syntaxes are `uigetfile('FilterSpec', 'DialogTitle', 'Location', [x y])` and `uiputfile('FilterSpec', 'DialogTitle', 'Location', [x y])`. Previously, the syntaxes were, `uigetfile('FilterSpec', 'DialogTitle', x, y)` and `uiputfile('FilterSpec', 'DialogTitle', x, y)`

Compatibility Considerations

You are encouraged to change to the new `uigetfile` and `uiputfile` syntax. The earlier syntaxes continue to be valid but may be removed in a later release.

Frames Not Available in GUIDE Layout Editor

The frame component no longer appears in the GUIDE Layout Editor component palette. It has been replaced by the panel and button group components. See “New Container Components” on page 249 for information about these new components.

Compatibility Considerations

GUIDE continues to support frames in those GUIs that contain them, but it is recommended that you replace them with panels or button groups.

External Interfaces/API, MATLAB Version 7 (R14)

New features and changes introduced in this version are organized by these topics:

- Importing and Exporting
- ActiveX and COM
- MATLAB Interface to Java
- General Features

Importing and Exporting

Saving Character Data with Unicode Encoding

The save function now saves character data to a MAT-file using Unicode character encoding by default. You can use your system's default character encoding scheme instead by specifying the -v6 option with save.

Compatibility Considerations. MAT-files saved in MATLAB version 7.0 without using the new -v6 flag will not be readable in previous versions of MATLAB.

Saving Data in Compressed Format

The save function now saves data to a MAT-file in a compressed format by default.

Large File I/O for MEX-Files

MATLAB supports the use of 64-bit file I/O operations in your MEX-file programs. This enables you to read and write data to files that are up to and greater than 2 GB ($2^{31}-1$ bytes). Note that some operating systems or compilers may not support files larger than 2 GB.

See “Large File I/O” in the External Interfaces documentation for more information.

ActiveX and COM

Automatic Registration of Automation Server on Installation

When installing previous versions of MATLAB, system administrators also had to run MATLAB at least once on each machine to register the Automation server. In MATLAB 7.0, the MATLAB installation software does the Automation server installation for you.

Support for Multiple COM Type Libraries

MATLAB now fully supports importing additional type libraries from within an IDL file. Any COM object that depends on an imported type library is now handled correctly.

COM Interface Supports Custom Interfaces

MATLAB now supports custom interfaces to a server component in configurations where MATLAB is the client controlling an ActiveX control, or an in-process or out-of-process server. For those COM components that implement one or more custom interfaces, you can list the interfaces in MATLAB using the new `interfaces` function:

```
h = actxserver('ComponentA.CustomObject')
h =
    COM.componenta.customobject

customlist = interfaces(h)
customlist =
    ICustomObject1
    ICustomObject2
```

Once you select the custom interface that you want, use the `invoke` function to get a handle to it:

```
c1 = invoke(h, 'ICustomObject1')
c1 =
    Interface.componenta_Type_Library.ICustomObject1_Interface
```


You can now use this handle with most of the COM client functions to access the properties and methods of the object through this custom interface. For example, to list the methods available through the `ICustomObject1` interface, use

```
invoke(c1)
    Add = double Add(handle, double, double)
    CustomMethod1 = HRESULT CustomMethod1(handle, int32)
    CustomMethod2 = HRESULT CustomMethod2(handle, int32)
    TripleAdd = [double, double] TripleAdd(handle, double, double)
    method3 = [string, int32, string, string] method3(
        handle, int16, int32, double, string)
    outin = [double, double, double, double] outin(
        handle, double, double)
    strings = string strings(handle, string)
```

You can read more about this feature in the section, “Getting Interfaces to the Object” in the External Interfaces documentation.

COM Data Type Support for Scripting Languages

In previous versions of MATLAB, a COM client program written in VBScript could not retrieve numeric data from or write data to the workspace of a MATLAB client. This was because VBScript does not support the `SAFEARRAY` data type used by MATLAB to pass numeric data to and from the server workspace using the `GetFullMatrix` and `PutFullMatrix` functions.

Release 14 adds two new functions, `GetWorkspaceData` and `PutWorkspaceData`, that pass data using the variant data type, a type that is supported by VBScript. You can use these new functions to pass either numeric or string data to any workspace in the COM server running MATLAB.

Refer to “Exchanging Data with the Server” in the External Interfaces documentation.

Additional ProgIDs for Latest MATLAB Version

There are three additional COM programmatic identifiers (ProgIDs) in MATLAB 7.0:

```
MATLAB.Autoserver
MATLAB.Autoserver.Single
MATLAB.Autoserver.7
```

Using any of these identifiers with the `actxserver` function guarantees that the MATLAB server you create always runs the latest version of MATLAB (version 7.0).

Note These new ProgIDs do not replace the `MATLAB.Application` identifier used in previous versions of MATLAB. You can continue using this ProgID, but there is no guarantee that `actxserver` will create a server running MATLAB 7.0.

Connecting to an Existing MATLAB Server

Instead of having to create new instances of a MATLAB server, clients can connect to an existing MATLAB automation server using the `GetObject` command. This sample Visual Basic program connects to a running MATLAB automation server, returning a handle `h` to that server. It then executes a simple plot command in the server:

```
Dim h As Object

' Call GetObject (omit first argument).
Set h = GetObject(, "matlab.application")

' Handle h should be valid now. Test it by calling Execute
h.Execute ("plot([0 18], [7 23])")
```

Graphical Interface to Listing Available ActiveX Controls

The `actxcontrollist` function enables you to see what COM controls are currently installed on your system. Type

```
list = actxcontrollist;
```

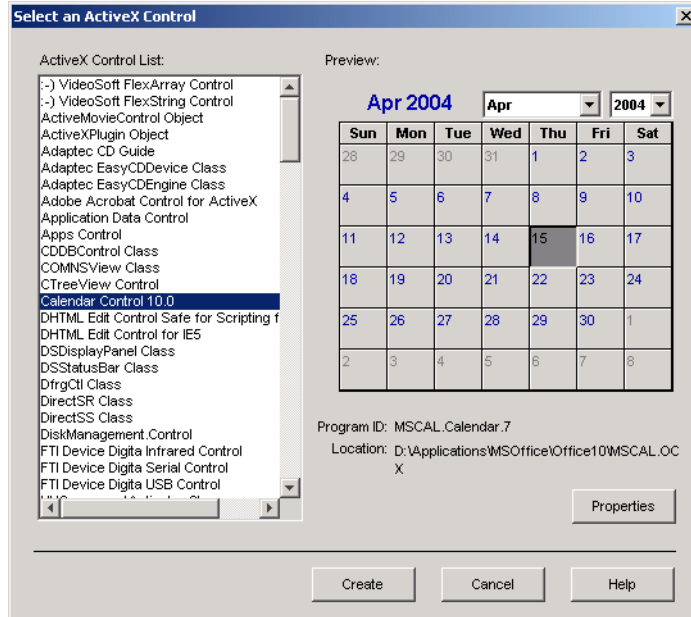
and MATLAB returns a list of each control, including its name, programmatic identifier (or ProgID), and filename, in the output cell array.

Refer to “Finding Out What Controls Are Installed” in the External Interfaces documentation.

Graphical Interface to Creating ActiveX Controls

The simplest way to create a control object is to use the `actxcontrolselect` function. This function displays a graphical interface that lists all controls installed on the system and creates the one that you select from the list.

The `actxcontrolselect` interface has a selection panel at the left of the window and a preview panel at the right. Click on one of the control names in the selection panel to see a preview of the control displayed. (If MATLAB cannot create the control, an error message is displayed in the preview panel.) Select an item from the list and click the **Create** button.



Refer to “Creating Control Objects Using a Graphical Interface” in the External Interfaces documentation.

New Functions for the MATLAB COM Interface

There are five new COM client functions.

Function	Description
<code>actxcontrollist</code>	List all currently installed ActiveX controls
<code>actxcontrolselect</code>	Display graphical interface for creating an ActiveX control
<code>interfaces</code>	List custom interfaces to a COM server
<code>iscom</code>	Determine if input is a COM or ActiveX object
<code>isinterface</code>	Determine if input is a COM interface

There are three new COM server functions. When invoked by a MATLAB or Visual Basic client, these functions execute in the server associated with the specified handle parameter.

Function	Description
<code>Feval</code>	Evaluate MATLAB function call in the server
<code>GetWorkspaceData</code>	Get data from server workspace
<code>PutWorkspaceData</code>	Store data in server workspace

See the function reference pages in the “External Interfaces Reference” documentation for more information.

COM Interface Supports Dot Syntax in Commands

You can now use a simpler form of syntax when invoking either MATLAB COM functions or methods belonging to COM objects. In this *dot syntax* (as it is referred to in the MATLAB documentation), you specify the object name, a dot (`.`), and then the name of the function or method you are calling. Enclose any input arguments in parentheses after the function name. Specify output arguments to the left of the equals sign:

```
outputvalue = object.function(arg1, arg2, ...)
```

For example, Release 13 syntax for invoking the `addproperty` function on a COM object with handle `h` was

```
invoke(h, 'addproperty', 'Position');
```

You can now perform the same operation using

```
h.addproperty('Position');
```

The get and set operations are even simpler:

```
** R13 SYNTAX **                               ** R14 SYNTAX **
x = get(h, 'Radius');                          x = h.Radius;
set(h, 'Radius', 50);                          h.Radius = 50;
```

Refer to “Invoking Commands on a COM Object” in the External Interfaces documentation.

Enumeration in COM Method Arguments

In addition to supporting enumeration for the properties of a COM object, MATLAB now supports enumeration for parameters passed to methods of a COM object. The only restriction is that the type library in use must report the parameter as `ENUM`, and only as `ENUM`.

Refer to “Specifying Enumerated Parameters” in the External Interfaces documentation.

Event Handling for COM Servers

In addition to handling events from ActiveX controls, MATLAB now handles events fired by Automation servers as well. Use the same event handling functions that you have been using for events from controls.

Function	Description
<code>eventlisteners</code>	Return a list of events attached to listeners
<code>events</code>	List all events, both registered and unregistered, a control or server can generate
<code>isevent</code>	Determine if an item is an event of a COM object
<code>registerevent</code>	Register an event handler with a control or server event
<code>unregisterallevents</code>	Unregister all events for a control or server
<code>unregisterevent</code>	Unregister an event handler with a control or server event

Refer to “How to Prepare for and Handle Events from a COM Server” and “Example — Responding to Events from an Automation Server” in the External Interfaces documentation.

Callbacks to COM Event Handlers Written as Subfunctions

Instead of having to maintain a separate M-file for every event handler routine you write, you can consolidate some or all of these routines into a single M-file using M-file subfunctions.

Refer to “Writing Event Handlers Using M-File Subfunctions” in the External Interfaces documentation.

Event Handlers Can Be Function Handles

In this release, you can now implement ActiveX event handlers as function handles.

Optional Input Arguments to COM Methods

When calling a method that takes optional input arguments, you can skip any optional argument by specifying an empty array ([]) in its place. The syntax for invoke with the second argument (arg2) not specified is as follows:

```
invoke(handle, 'methodname', arg1, [], arg3);
```

See the section, “Optional Input Arguments” in the External Interfaces documentation for more information.

Display of Interface Handles

MATLAB has changed the way it displays a COM interface in this release. For example, the string used to represent an interface in MATLAB 6.5 was

```
[1x1 Interface.excel.application.Workbooks]
```

MATLAB 7.0 represents this same interface with the following string:

```
[1x1 Interface.Microsoft_Excel_9.0_Object_Library.Workbooks]
```

Compatibility Considerations. You may need to change any code that depends on the previous behavior.

MATLAB Interface to Java

Java Interface Adds Dynamic Java Class Path

MATLAB loads Java class definitions from files that are on the Java class path. The Java class path now consists of two segments: the *static path*, and a new segment called the *dynamic path*.

The static path is loaded from the file `classpath.txt` at the start of each MATLAB session and cannot be changed without restarting MATLAB. This was the only path available in previous versions of MATLAB. Thus, there was no way to change the Java path without restarting MATLAB.

The dynamic Java class path can be loaded at any time during a MATLAB session using the `javaclasspath` function. You can define the dynamic path (using `javaclasspath`), modify the path (using `javaaddpath` and `javarmppath`), and refresh the Java class definitions for all classes on the dynamic path (using `clear java`) without restarting MATLAB. See the function reference pages for more information on how to use these functions.

The `javaclasspath` function, when used with no arguments, displays both the static and dynamic segments of the Java class path:

```
javaclasspath

      STATIC JAVA PATH

D:\Sys0\Java\util.jar
D:\Sys0\Java\widgets.jar
D:\Sys0\Java\beans.jar
      .
      .

      DYNAMIC JAVA PATH

User4:\Work\Java\ClassFiles
User4:\Work\Java\mywidgets.jar
      .
      .
```

You can read more about this feature in the sections, “The Java Class Path” and “Making Java Classes Available to MATLAB” in the External Interfaces documentation.

Locating Java Native Method DLLs with File `librarypath.txt`

Previous versions of MATLAB required that you set a system environment variable to enable Java to locate the shared libraries supporting any native methods you need to use. This environment variable was `PATH` on Windows systems, and `LD_LIBRARY_PATH` on UNIX systems. This is no longer necessary.

Now you can enter the names of those directories that contain native method libraries in a new file called `librarypath.txt` using one line per directory. The `librarypath.txt` file resides adjacent to the similar file `classpath.txt` in the `$matlab/toolbox/local` directory.

General Features

New `mx` Functions

New functions `mxIsInt64` and `mxIsUInt64` return true if an `mxArray` represents its data as signed or unsigned 64-bit integers respectively.

Identifying Dependencies When MEX-Files Do Not Load

If MEX-files don't load on the PC the error message is not informative. The dependency is a very useful tool distributed with MSVC. It is also freely available from www.dependencywalker.com. In R15, we will incorporate some kind of dependency walker into our MEX loader but for now, we will point users to the web site.

Recompile MEX-Files on GLNX86 and Macintosh

In Release 14, MATLAB uses C++ exception handling. MEX-files built prior to R14 did not support C++ exceptions.

For example, write a C MEX-file that just calls `mexErrMsgTxt`. If you build this with a release prior to Release 14 and run it, the program aborts MATLAB. If you build this with Release 14 and run it, MATLAB will handle the exception correctly.

Compatibility Considerations. On GLNX86 and Macintosh systems, all MEX-files that can throw errors need to be recompiled for R14.

Shared Libraries Now In /bin/\$ARCH

Shared libraries previously residing in directory `$MATLAB/extern/lib/$ARCH` are now in `$MATLAB/bin/$ARCH`.

Compatibility Considerations. You may need to change any code that depends on the previous behavior.

Version 6.5.1 (R13SP1) MATLAB

This table summarizes what's new in Version 6.5.1 (R13SP1):

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

New features and changes introduced in this version are organized by these areas:

- MATLAB Interface to Generic DLLs
- Reading HDF5 Files
- Relational Operators Work with int64, uint64
- Reading and Writing Data with JPEG Lossless Compression
- Reading and Writing L*a*b* Color Data

MATLAB Interface to Generic DLLs

A shared library is a collection of functions that are available for use by one or more applications running on a system. On Windows systems, the library is precompiled into a dynamic link library (.dll) file. At run-time, the library is loaded into memory and made accessible to all applications. The MATLAB Interface to Generic DLLs enables you to interact with functions in dynamic link libraries directly from MATLAB.

Documentation

For help on this new feature, see “MATLAB Interface to Generic DLLs” in the External Interfaces documentation.

The examples used in the documentation use library (.dll) and header (.h) files located in the MATLABROOT\extern\examples\shrlib directory. To use these example files, first add this directory to your MATLAB path with the following command:

```
addpath([matlabroot '\extern\examples\shrlib'])
```

Or you can make this your current working directory with this command:

```
cd([matlabroot '\extern\examples\shrlib'])
```

Restrictions for This Release

- At this time, the MATLAB Interface to Generic DLLs is supported on Windows systems only.
- Passing a void ** argument to a function in a dynamic link library is not supported in this release.
- Passing a complex structure argument to a function in a dynamic link library is not supported in this release. (The term *complex structure argument* refers to a structure constructed from other structures.)
- Passing an array of pointers, is not supported in this release. An example of an array of pointers is an array of strings.
- MATLAB does not support manipulation of pointers returned by functions in a dynamic link library at this time. An example of this type of operation is the addition or subtraction of pointers.

Function and Data Type Names in Generic DLL Interface

Minor changes have been made to the naming of some functions and data types in the Generic DLL interface. If you are upgrading from the post-release 13 download of MATLAB, see “Function and Data Type Names in Generic DLL Interface” on page 289 of these release notes.

Reading HDF5 Files

This release includes support for reading files that use the Hierarchical Data Format, Version 5 (HDF5). HDF5 is a product of the National Center for

Supercomputing Applications (NCSA). The NCSA develops software and file formats for scientific data management.

This section includes this information:

- An overview of the structure of an HDF5 file
- Determining the contents of an HDF5 file
- Reading data from an HDF5 file
- Mapping HDF5 data types to MATLAB data types

Note MATLAB has supported reading and writing HDF files for several releases. The HDF and HDF5 specifications are not compatible.

Overview of HDF5 File Structure

HDF 5 files can contain multiple *datasets*. A dataset is a multidimensional array of data elements. Datasets can have associated metadata. HDF5 files store the datasets and attributes in a hierarchical structure, similar to a directory structure. The directories in the hierarchy are called *groups*. A group can contain other groups, datasets, attributes, links, and data types.

To illustrate this structure, the following figure shows the contents of the sample HDF5 file included with MATLAB, `example.h5`.

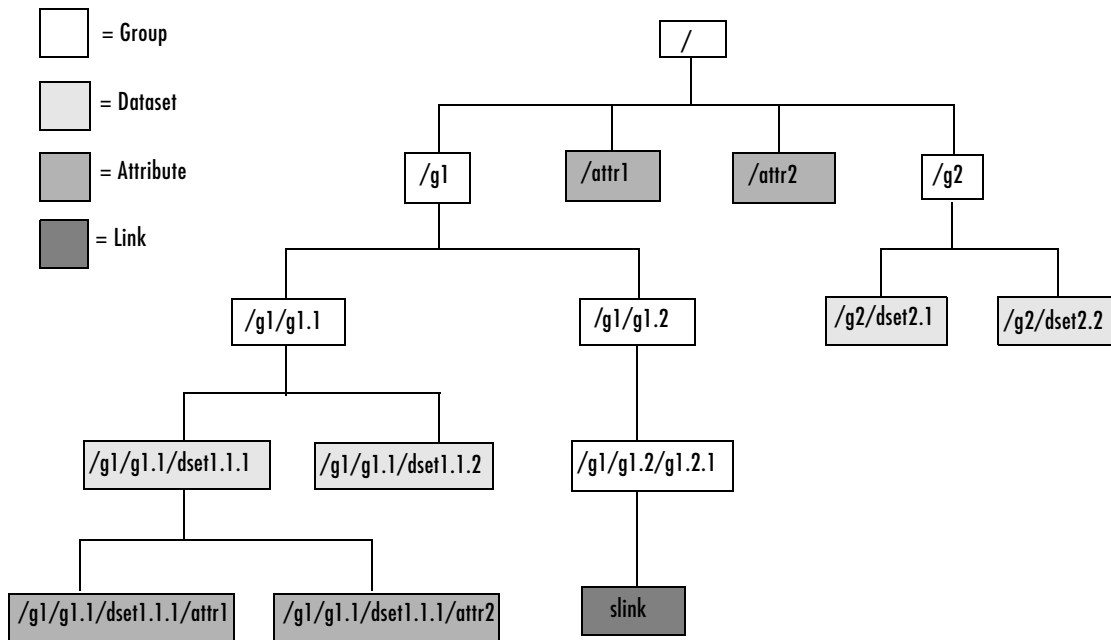


Figure 0-1: Hierarchical Structure of example.h5 HDF5 File

Determining the Contents of an HDF5 File

To extract an attribute or dataset from an HDF5 file, you must know the name of the attribute or dataset. You specify the name as an argument to the `hdf5read` function, described in “Reading Data from an HDF5 File” on page 272.

To find the names of all the datasets and attributes contained in an HDF5 file, you can use the `hdf5info` function. For example, to find out what the sample HDF5 file, `example.h5`, contains, use this syntax.

```
fileinfo = hdf5info('example.h5');
```

The `fileinfo` structure returned by `hdf5info` contains various information about the HDF5 file, including the name of the file and the version of the HDF5 library that MATLAB is using.

```
fileinfo =  
    Filename: 'example.h5'  
    LibVersion: '1.4.2'  
    Offset: 0  
    FileSize: 8172  
    GroupHierarchy: [1x1 struct]
```

To explore the contents of the file, examine the GroupHierarchy field.

```
level1 = fileinfo.GroupHierarchy  
  
level1 =  
  
    Filename: 'C:\matlab\toolbox\matlab\demos\example.h5'  
    Name: '/'  
    Groups: [1x2 struct]  
    Datasets: []  
    Datatypes: []  
    Links: []  
    Attributes: [1x2 struct]
```

The GroupHierarchy structure describes the top-level group in the file, called the root group. HDF5 uses the UNIX convention and names this top-level group / (forward slash), as seen in the Name field. The other fields in the structure describe the contents of the group. In the example, the root group contains two groups and two attributes. All the other fields, such as the Datasets field, are empty. To traverse further down the file hierarchy, look at one of the structures in the Groups field.

```
level2 = level1.Groups(2)  
  
level2 =  
  
    Filename: 'C:\matlab\toolbox\matlab\demos\example.h5'  
    Name: '/g2'  
    Groups: []  
    Datasets: [1x2 struct]  
    Datatypes: []  
    Links: []  
    Attributes: []
```

In this group, the Groups field is empty and the Datasets field contains two structures. To get the names of the datasets, examine the Name field of either of these Dataset structures. This structure provides other information about the dataset including how many dimensions it contains (Dims) and the data type of the data in the dataset (Datatype).

```
dataset1 = level2.Datasets(1)
```

```
dataset1 =
```

```
    Filename: 'L:\matlab\toolbox\matlab\demos\example.h5'  
      Name: '/g2/dset2.1'  
      Rank: 1  
    Datatype: [1x1 struct]  
      Dims: 10  
    MaxDims: 10  
    Layout: 'contiguous'  
Attributes: []  
      Links: []  
    Chunksize: []  
    Fillvalue: []
```

Reading Data from an HDF5 File

To read an HDF5 file, use the `hdf5read` function, specifying the name of the file and the name of the dataset as arguments. For information about finding the name of a dataset, see “Determining the Contents of an HDF5 File” on page 270.

For example, to read the dataset, `/g2/dset2.1` from the HDF5 file `example.h5`, use this syntax:

```
data = hdf5read('example.h5', '/g2/dset2.1');
```


The return value `data`, contains the values in the dataset, in this case a 1-by-10 vector of single precision values.

```
data =

    Columns 1 through 8

    1.0000    1.1000    1.2000    1.3000    1.4000    1.5000
    1.6000    1.7000

    Columns 9 through 10

    1.8000    1.9000
```

Mapping HDF5 Data Types to MATLAB Data Types

The `hdf5read` function maps HDF5 data types to MATLAB data types, depending on whether the data in the dataset is in an *atomic* data type or a *non-atomic* data type.

HDF5 Atomic Data Types. If the data in the dataset is stored in one of the HDF5 atomic data types, `hdf5read` uses the equivalent MATLAB data type to represent the data. Each dataset contains a `Datatype` field that names the data type. For example, the dataset `/g2/dset2.2` in the sample HDF5 file includes this data type information.

```
dtype = dataset1.Datatype
dtype =

    Name: []
    Class: 'H5T_IEEE_F32BE'
    Elements: []
```

The `H5T_IEEE_F32BE` class name indicates the data is a four-byte, big-endian, IEEE floating point data type. (See the HDF5 specification for more information about atomic data types.)

HDF5 Non-Atomic Data Types. If the data in the dataset is stored in one of the HDF5 non-atomic data types, `hdf5read` represents the dataset in MATLAB as an object. To access the data in the dataset, you must access the `Data` field in the object.

To illustrate, this example uses `hdf5read` to read a dataset called `/dataset2` from the HDF5 file, `my_hdf5_file.h5`. The dataset contains four elements; each element is an HDF5 array.

```
data = hdf5read('my_hdf5_file.h5', '/dataset2');
```

In MATLAB, the `hdf5read` function creates a 1x4 array of `hdf5.h5array` objects to represent this data.

```
whos
```

Name	Size	Bytes	Class
data	1x4		hdf5.h5array

```
Grand total is 4 elements using 0 bytes
```

Index into the MATLAB array to view the first element in the dataset.

```
data(1)
```

```
hdf5.h5array:
```

```
Name: ''  
Data: [4x5x3 int32]
```

To look at the raw data in the HDF5 array element, access the Data field in the object.

```
data(1).Data

ans(:,:,1) =
  0  1  2  3  4
 10 11 12 13 14
 20 21 22 23 24
 30 31 32 33 34

ans(:,:,2) =
 100 101 102 103 104
 110 111 112 113 114
 120 121 122 123 124
 130 131 132 133 134

ans(:,:,3) =
 200 201 202 203 204
 210 211 212 213 214
 220 221 222 223 224 230 231 232 233 234
```

The `hdf5read` function uses any of the following objects to represent HDF5 non-atomic data types.

- `hdf5.h5array`
- `hdf5.h5enum`
- `hdf5.h5vlen`
- `hdf5.h5compound`
- `hdf5.h5string`

Relational Operators Work with int64, uint64

All relational operators such as `<`, `>`, `<=`, `>=`, `~=`, and `==` now support `int64` and `uint64` data types.

Reading and Writing Data with JPEG Lossless Compression

MATLAB now supports reading and writing data that has been compressed using JPEG lossless compression. With lossless compression, you can recover

the original image from its compressed form. Lossless compression, however, achieves lower compression ratios than its counterpart, lossy compression.

Using the `imread` function, you can read data that has been compressed using JPEG lossless compression.

Using the `imwrite` function, you can write data to a JPEG file using lossless compression. For the `imwrite` function, you specify the `Mode` parameter with the `'lossless'` value.

Reading and Writing $L^*a^*b^*$ Color Data

The `imread` function can now read color data that uses the $L^*a^*b^*$ color space from TIFF files. The TIFF files can contain $L^*a^*b^*$ values that are in 8-bit or 16-bit CIELAB encodings or in 8-bit or 16-bit ICCLAB encodings.

If a file contains 8-bit or 16-bit CIELAB data, `imread` automatically converts the data into 8-bit or 16-bit ICCLAB encoding. The 8-bit or 16-bit CIELAB data cannot be represented as a MATLAB array because it contains a combination of signed and unsigned values.

The `imwrite` function can write $L^*a^*b^*$ data to a file using either the 8-bit or 16-bit CIELAB encoding or the 8-bit or 16-bit ICCLAB encoding. You select the encoding by specifying the value of the `ColorSpace` parameter.

Fixed Bugs

MATLAB 6.5.1 includes these major bug fixes:

- “Seeking Within a File” on page 277
- “Reshaping to More Than Two Dimensions” on page 277
- “mkdir No Longer Fails On Windows NT” on page 278
- “Using sqrt with Complex Input” on page 278
- “Multiplying Matrices with Non-Double Entries” on page 278
- “Sorting a Sparse Row Vector or Matrix” on page 278
- “diff Produces Correct Results with Logical Inputs” on page 279
- “Opening Modal Dialog with Third-Party GUI Open” on page 279
- “Serial Port Object with Latest Windows Service Pack” on page 279
- “OpenGL Problem Using Notebook” on page 279
- “Lcc C Compiler Fixed to Handle Large C Files” on page 279
- “Bug Fixes in MATLAB Interface to COM” on page 280
- “Bug Fixes in Creating GUIs” on page 286

Note In addition to the bug fixes described on this page, there are several bug fixes relating to MATLAB mathematics that are documented in a separate HTML bug-fix report.

Seeking Within a File

In Release 13, when you opened a file in write-only ('wb') mode, you could not seek to a position in the file without first seeking to the beginning of the file. The `fseek` function has been fixed to allow seeking from any position of the file.

Reshaping to More Than Two Dimensions

In Release 13, under certain circumstances, reshaping an array to have more than two dimensions produced a two dimensional result. This has been corrected.

mkdir No Longer Fails On Windows NT

In Release 13, if on Windows NT you called the `dir`, `exist`, `isdir`, or what function on a nonexistent directory name on a network drive, it caused a windows handle to remain open to that directory name until you exit the MATLAB session. This condition caused any attempts to use `mkdir` on that directory to fail. This problem also affected the `mkdir` command when run from a DOS command prompt. This condition would persist until you exited MATLAB, thus freeing the handle.

This bug is fixed in this release.

Using sqrt with Complex Input

In Release 13, under certain circumstances, the `sqrt` function incorrectly produced a real result when called with a complex input. This bug has been corrected.

Multiplying Matrices with Non-Double Entries

In Release 13, MATLAB gave an incorrect answer or crashed for expressions of the following forms:

- $A' * B$
- $A * B'$
- $A' * B'$
- $A.' * B$
- $A * B.'$
- $A.' * B.'$
- $A' * B.'$
- $A.' * B'$

when either A or B was a numeric, non-double value (`single`, `int32`, etc.). This has been fixed for this release.

Sorting a Sparse Row Vector or Matrix

In Release 13, a segmentation violation occurred when you used the command `sort(S,2)` to sort a sparse row vector or to sort a sparse matrix along its rows. This bug is fixed in this release.

diff Produces Correct Results with Logical Inputs

In Release 13, the `diff` function could produce an incorrect result when you passed a logical array to it. This bug is fixed in this release.

Opening Modal Dialog with Third-Party GUI Open

In Release 13, MATLAB would occasionally hang if the user tried to open a modal dialog box when a third-party GUI was open. This no longer happens.

Serial Port Object with Latest Windows Service Pack

Under certain hardware configurations, or when using the latest Service Pack from Microsoft Windows, the serial port object in both MATLAB and the Instrument Control Toolbox could cause MATLAB to crash or hang. This problem is resolved in this release.

Several additional problems affecting the serial port have also been identified and fixed:

- 1 The serial port object now obeys all supported parity configurations.
- 2 The serial port object now obeys all supported flow control configurations.
- 3 On Windows, serial ports higher than COM8 were not recognized by MATLAB. As of this release, MATLAB supports up to 256 ports.
- 4 The serial port object generates output empty events after running the serial port object continuously.

OpenGL Problem Using Notebook

This version of MATLAB uses an improved algorithm for selecting pixel formats when using the `UseGenericOpenGL` feature on Windows. This improvement fixes rendering problems seen with Notebook.

For information on graphics rendering, see Tech Note 1201.

Lcc C Compiler Fixed to Handle Large C Files

Lcc version 2.4.1 MathWorks patch 1.29 corrects a bug encountered when compiling very large C files. Although this bug has only been observed when

using large Stateflow[®] models, we suggest that you upgrade to the new version to avoid potential problems when compiling MEX-files.

If you choose not to upgrade your version of Lcc, you can select a different C compiler using `mex -setup` from the MATLAB command line.

Bug Fixes in MATLAB Interface to COM

This release includes the following bug fixes in the COM interface:

- “Blank Spreadsheet Cells Returned as NaNs” on page 280
- “Importing Excel Worksheets Containing Currency Format” on page 281
- “Getting the Forms Font Interface” on page 281
- “Programmatic Identifiers Containing Space Characters” on page 281
- “Naming of Interfaces Returned by `invoke` or `get`” on page 281
- “Optional Input and Output Arguments Supported” on page 282
- “Memory Leak with MATLAB as COM Client” on page 282
- “Support for Multiple Type Libraries” on page 282
- “MATLAB Now Supports Skipping an Optional Argument” on page 283
- “Saving COM Objects Created with `actxserver`” on page 283
- “Creating Certain Servers That Do Not Have Type Libraries” on page 284
- “Creating Microsoft Controls” on page 285
- “ActiveX Controls Created with Visual Basic 6.0” on page 285
- “Type Mismatch Error Fixed” on page 285

Blank Spreadsheet Cells Returned as NaNs

When reading from a Microsoft Excel spreadsheet in a COM environment where MATLAB is the COM client and Excel the server, MATLAB now returns any empty cells in the spreadsheet as NaNs. In MATLAB 6.5 (Release 13), this same operation had returned a matrix of empty (`[]`) values.

For example, if the range A1 to D3 in a currently active workbook sheet contains no data, MATLAB 6.5.1 returns the following matrix of NaN values:

```
eActiveSheet = get(e, 'ActiveSheet');
eActiveSheetRange = get(eActiveSheet, 'Range', 'A1', 'D3');

eActiveSheetRange.Value
ans =
    [NaN]    [NaN]    [NaN]    [NaN]
    [NaN]    [NaN]    [NaN]    [NaN]
    [NaN]    [NaN]    [NaN]    [NaN]
```

Importing Excel Worksheets Containing Currency Format

In MATLAB 6.5, using a COM interface to Excel to import worksheet data containing currency format failed with either a field access error or segmentation violation. This bug is fixed in this release.

Getting the Forms Font Interface

In MATLAB 6.5, attempts to get the Font interface from a `forms.textbox.1` control, as done in the second line below, caused MATLAB to crash.

```
h=actxcontrol('forms.textbox.1')
font = h.Font
```

This bug is fixed in this release.

Programmatic Identifiers Containing Space Characters

Using the `actxcontrol` function with a ProgID argument containing one or more spaces failed in MATLAB 6.5. This bug is fixed in this release. For example, the following command now works:

```
h = actxcontrol('rmocx.RealPlayer G2 Control.1')
h =
    COM.rmocx.realplayer g2 control.1
```

Naming of Interfaces Returned by `invoke` or `get`

In MATLAB 6.5, interfaces returned by the `invoke` and `get` functions were given a name composed of the programmatic identifier (ProgID) for the component and the name of the method or property being invoked. In cases

where a method or property implemented multiple interface types, this naming algorithm resulted in interface names that were not always unique.

For example, when invoking a method that returns an Excel and a Word interface, you could obtain any number of either type of interface (Excel or Word), but you could not obtain interfaces of both types. In such cases, you might be unable to access methods and properties of this interface.

In this release, interface names constructed by MATLAB are composed of the name of the type library and the class name, thus ending this potential naming conflict. If you invoke the method described in the last paragraph, MATLAB now returns the following for any Excel interfaces that you request:

```
Interface.Microsoft_Excel_9.0_Object_Library._Application
```

And MATLAB returns a different handle for Word interfaces:

```
Interface.Microsoft_Word_9.0_Object_Library._Application
```

Optional Input and Output Arguments Supported

MATLAB now supports optional input and output arguments passed in COM method calls. These arguments are declared as [in, optional] and [out, optional] respectively.

Memory Leak with MATLAB as COM Client

In Version 6.5, a memory leak developed under certain circumstances when MATLAB was configured as a COM client. This was caused by internal MATLAB code failing to release memory allocated by the method `StringFromCLSID`. This bug is fixed in this release.

Support for Multiple Type Libraries

MATLAB now supports multiple type libraries. If a COM object has many interfaces that are described in multiple type libraries, MATLAB can now retrieve the information correctly.

MATLAB Now Supports Skipping an Optional Argument

When calling ActiveX automation server methods, you can skip any optional arguments in the argument list by specifying that argument value as an empty matrix (`[]`). For example, the `Add` method shown below accepts as many as four optional arguments:

```
Add(Before, After, Count, Type)
```

To call this method, specifying values for `After` and `Count`, but no values for `Before` or `Type`, use this syntax.

```
addedsheet = invoke(Sheets, 'Add', [], Sheet1, 5);
```

Use `[]` for any arguments you skip, and that also precede the ones you do specify in the argument list. In this case, the `Before` argument is not specified but two subsequent arguments are.

Saving COM Objects Created with `actxserver`

Release 13 does not support saving COM objects that have been created with the `actxserver` function. You can use `save only` on control objects (created with `actxcontrol`). Attempting to use `save` on a COM server object causes MATLAB to hang temporarily, and eventually crash.

This bug has been fixed in this release so that if you now attempt to save a COM server object, MATLAB saves the object and any base properties of the object, but does not attempt to save any interfaces that might exist.

The same behavior applies to the `pack` function on COM objects.

This example creates a server running Microsoft Excel, adds a new property to the object, and saves it to the file `excelserver.mat`. It then reloads the server from the MAT-file.

```
e = actxserver ('Excel.Application');
addproperty(e, 'NewProperty');
set(e, 'NewProperty', 500);
get(e, 'NewProperty')
ans =
    500

save('excelserver.mat')
clear
get(e, 'NewProperty')
??? Undefined function or variable 'e'.

load('excelserver.mat')
get(e, 'NewProperty')
ans =
    500
```

Creating Certain Servers That Do Not Have Type Libraries

In the Release 12.1 and Release 13 releases, the `actxserver` function generated an error when attempting to create a COM object for certain servers. One error commonly returned by `actxserver` in these releases was

```
h = actxserver('msdev.application')
??? Error using ==> actxserver
Cannot find type library. COM object creation failed.
```

This has now been fixed in this release.

```
h = actxserver('msdev.application')
h =
    COM.msdev.application
```

Creating Microsoft Controls

Earlier versions of MATLAB would crash if you attempted to create certain Microsoft COM controls with the `actxcontrol` function. Examples of these controls, by programmatic identifier (ProgID), are shown below. MATLAB now successfully creates the controls.

<code>mschart20lib.mschart</code>	<code>msdatalistlib.datacombo</code>
<code>msdatagridlib.datagrid</code>	<code>MSComCtl2.DTPicker.2</code>
<code>msdatalistlib.datalist</code>	<code>MSHierarchicalFlexGridLib.MSHFlexGrid.6</code>

ActiveX Controls Created with Visual Basic 6.0

In Release 13, if you attach a callback routine to an event, and this event is eventually fired by a control created in Visual Basic 6.0, an error dialog box appears with the message “Run-Time error.”

This has been fixed in this release.

Type Mismatch Error Fixed

Some COM objects may define methods that pass scalar inputs by reference. This might appear in a type library signature as shown here for the `x` input:

```
functionname(double *x, [out] double *y)
```

Note that when input or output is not specifically stated, as is the case here for `x`, MATLAB defaults to input (`[in]`). So the line shown above is interpreted by MATLAB as

```
functionname([in] double *x, [out] double *y)
```

In MATLAB, the `[in]` and by-reference (`*`) specifications are considered incompatible for scalar arguments. In Release 13, MATLAB ignores the by-reference specifier for scalar inputs and passes such arguments by value instead. Thus, any modified value for such an argument is not received by the calling function. You may also see a type mismatch error displayed, even when trying to access valid control methods.

MATLAB 6.5.1 fixes this bug by treating the `[in]` specifier for scalar references as if it were `[in,out]`.

In this example using MATLAB syntax, the `GetWinVersionX` function passes six double arguments by reference, yet none are returned in MATLAB 6.5:

```
GetWinVersionX = int32 GetWinVersionX(  
    handle, double, double, double, double, double)
```

In MATLAB 6.5.1, all scalar reference arguments specified (or defaulting to `[in]`) are treated as `[in,out]`, and all references cause a value to be returned:

```
GetWinVersionX = [int32, double, double, double, double,  
    double, double] GetWinVersionX(  
    handle, double, double, double, double, double)
```

Note that this bug affects only scalar arguments. The `VT_DISPATCH` and `VT_VOID` types are not affected.

Bug Fixes in Creating GUIs

This release includes the following bug fixes related to creating, converting, and exporting GUIs:

- “Converting a MATLAB 5.3 GUI to MATLAB 6.5” on page 286
- “Using GUIDE on Existing GUIs with Empty Tag Property” on page 287
- “Exporting GUIs from GUIDE to a Single M-file” on page 287
- “MATLAB Hangs when Using Property Inspector from GUIDE” on page 287
- “Recursion Limit Error when Running Existing GUIs from GUIDE” on page 288

Converting a MATLAB 5.3 GUI to MATLAB 6.5

Converting a MATLAB 5.3 (R11) GUI to MATLAB 6.5 sometimes resulted in the error:

```
Unhandled internal error in guidemfile. Reference to non-existent  
field 'blocking'
```

This problem has been fixed.

Using GUIDE on Existing GUIs with Empty Tag Property

In MATLAB Version 6.5, editing a GUI that contained a uicontrol whose **Tag** property was set to [] (empty) sometimes generated the following error message:

```
Unhandled internal error in guidefunc.  
Error using ==> set  
Value must be a string
```

This problem has been fixed.

Exporting GUIs from GUIDE to a Single M-file

In MATLAB Version 6.5, some GUIs exported from GUIDE failed to open. In other cases, attempting to export a GUI resulted in one of the following errors:

```
??? Error using ==> guidefunc  
Error using ==> ==  
Matrix dimensions must agree.
```

```
??? Error using ==> guidefunc  
Error using ==> ==  
Function '==' is not defined for values of class 'struct'.
```

These problems have been fixed.

MATLAB Hangs when Using Property Inspector from GUIDE

Using the Property Inspector from GUIDE sometimes caused MATLAB Version 6.5 to hang. This problem has been fixed.

Recursion Limit Error when Running Existing GUIs from GUIDE

In MATLAB Version 6.5, running some existing GUIs from GUIDE generated the following error message:

```
??? Error using ==> guidatafunc
Maximum recursion limit of 500 reached. Use
set(0,'RecursionLimit',N) to change the limit. Be aware that
exceeding your available stack space can crash MATLAB and/or
your computer.
```

```
Could not create figure:
127
```

This problem has been fixed.

Compatibility Considerations

These changes might cause incompatibilities when you upgrade from an earlier version, or when you use files on multiple versions.

Rebuild Macintosh MEX-files

Macintosh MEX-files (named `.mex`) built with MATLAB 5.2 or older will not work with MATLAB 6.5 or later. You must recompile these files, creating a new file with the file extension `.mexmac`.

Function and Data Type Names in Generic DLL Interface

The following functions have been renamed since the initial download release of the Generic DLL Interface:

- The `libmethods` function is now called `libfunctions`.
- The `libmethodsview` function is now called `libfunctionsview`.

All data types ending in `Ref` are now suffixed with `Ptr`. For example, `doubleRef` is now called `doublePtr`, and `int16Ref` is now `int16Ptr`.

All data types ending in `RefPtr` are now suffixed with `PtrPtr`. For example, `doubleRefPtr` is now called `doublePtrPtr`, and `int16RefPtr` is now `int16PtrPtr`.

Compatibility Summary for MATLAB

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

- [Version 7.2 \(R2006a\) Compatibility Summary for MATLAB](#)
- [Version 7.1 \(R14SP3\) Compatibility Summary for MATLAB](#)
- [Version 7.04 \(R14SP2\) Compatibility Summary for MATLAB](#)
- [Version 7.01 \(R14SP1\) Compatibility Summary for MATLAB](#)
- [Version 7 \(R14\) Compatibility Summary for MATLAB](#)
- [Version 6.5.1 \(R13SP3\) Compatibility Summary for MATLAB](#)

Version 7.2 (R2006a) Compatibility Summary for MATLAB

For other versions, see the Compatibility Summary for MATLAB.

Area	New Features and Changes with Version Compatibility Impact
Desktop Tools and Development Environment	<p>See the Compatibility Considerations subheading for each of these new features or changes:</p> <ul style="list-style-type: none"> • Installation Directory Structure on Windows • Preferences Reorganized and New Keyboard Pane Added to Support Command Window and Editor/Debugger • Go Menu Added; Bookmark and Go To Items Moved from Edit Menu to Go Menu • Cell Mode On by Default—Shows Cell Toolbar and Possibly Horizontal Lines and Yellow Highlighting; Cell Information Bar and Button Added • M-Lint and mlint Enhancements and Changes • Visual Directory View Removed from Current Directory Browser • PVCS Source Control System Name Change
Mathematics	None
Data Analysis	<p>See the Compatibility Considerations subheading for each of these new features or changes:</p> <ul style="list-style-type: none"> • Linux 64 Platform Fully Enabled for Time Series Tools
Programming	<p>See the Compatibility Considerations subheading for each of these new features or changes:</p> <ul style="list-style-type: none"> • Using avifile and movie2avi on Windows XP 64 • Regular Expressions • MATLAB Warns on Invalid Input to str2func • I/O Functions Can Specify and Use Character Encoding Schemes

Area	New Features and Changes with Version Compatibility Impact
Graphics and 3-D Visualization	None
Creating GUIs	See the Compatibility Considerations subheading for each of these new features or changes: <ul style="list-style-type: none">• Treatment of '&' in Menu Label Is Changed
External Interfaces	See the Compatibility Considerations subheading for each of these new features or changes: <ul style="list-style-type: none">• MEX-Files Built with gcc on Linux Must Be Rebuilt• MEX-Files in MATLAB for Windows x64• Compaq Visual Fortran Engine and MAT Options File Renamed• Options Files Removed for Unsupported Compilers• Obsolete Functions No Longer Documented

Version 7.1 (R14SP3) Compatibility Summary for MATLAB

For other versions, see the Compatibility Summary for MATLAB.

Area	New Features and Changes with Version Compatibility Impact
Desktop Tools and Development Environment	<p>See the Compatibility Considerations subheading for each of these new features or changes:</p> <ul style="list-style-type: none"> • Windows -nodesktop No Longer has Menu Bar and Toolbar; Use Function Equivalents Instead • Preferences Directory Added for R14SP3; Supplements R14 Directory • info.xml File Automatic Validation; Shows Warnings for Invalid Constructs • Hyperlink Color Preference Moved • Demos Run in Command Window as Scripts and Their Variables Now Created in Base Workspace • echodemo Function Added to Replace playshow function • Visual Directory View to be Removed • HTML File Indenting Feature Added as the Default • Notebook Setup Changes; Some Arguments Removed • Word Versions Supported by Notebook; Word 97 No Longer Supported
Mathematics	<p>See the Compatibility Considerations subheading for each of these new features or changes:</p> <ul style="list-style-type: none"> • New Functions
Data Analysis	None

Area	New Features and Changes with Version Compatibility Impact
Programming	<p>See the Compatibility Considerations subheading for each of these new features or changes:</p> <ul style="list-style-type: none"> • New Functions • Modified Functions • isfield Function Supports Cell Arrays; Results Might Differ from Previous Version • Seconds Field Now Truncated; Results Might Differ Built-in Functions No Longer Use .bi; Impacts Output of which Function New Warning About Potential Naming Conflict
Graphics and 3-D Visualization	None
Creating GUIs	<p>See the Compatibility Considerations subheading for each of these new features or changes:</p> <ul style="list-style-type: none"> • Plans for Obsolete Functions
External Interfaces	<p>See the Compatibility Considerations subheading for each of these new features or changes:</p> <ul style="list-style-type: none"> • New File Extension for MEX-Files on Windows • New Preferences Directory and MEX Options • Compiler Support • Import Libraries Moved

Version 7.04 (R14SP2) Compatibility Summary for MATLAB

For other versions, see the Compatibility Summary for MATLAB.

Area	New Features and Changes with Version Compatibility Impact
Desktop Tools and Development Environment	<p>See the Compatibility Considerations subheading for each of these new features or changes:</p> <ul style="list-style-type: none"> • JVM Updated • Subfunction Help Syntax Changed • Bug Fixes and Known Problems Now on Web; No Longer Found Via Help Search • Workspace Browser Preference Panel Removed • Preference for Editor/Debugger Dialog Moved • Register Project Feature Added; Add to Source Control Behavior Changed • Cell Publishing: File Extension Changes • Notebook Support for Word 97 to be Discontinued
Mathematics	<p>See the Compatibility Considerations subheading for each of these new features or changes:</p> <ul style="list-style-type: none"> • max and min on Complex Integers Not Supported
Programming	<p>See the Compatibility Considerations subheading for each of these new features or changes:</p> <ul style="list-style-type: none"> • xlsread Imported Date Format Changes • Nonscalar Arrays of Function Handles to Become Invalid • Assigning Nonstructure Variables As Structures Displays Warning

Area	New Features and Changes with Version Compatibility Impact
Graphics and 3-D Visualization	See the Compatibility Considerations subheading for each of these new features or changes: <ul style="list-style-type: none">• Cannot Dock Figures on Macintosh• Plotting Tools Not Working on Macintosh• Not All Macintosh System Fonts Are Available• XDisplay Property Setable on Motif-Based Systems
Creating GUIs	None
External Interfaces	None

Version 7.01 (R14SP1) Compatibility Summary for MATLAB

For other versions, see the Compatibility Summary for MATLAB.

Area	New Features and Changes with Version Compatibility Impact
Desktop Tools and Development Environment	<p>See the Compatibility Considerations subheading for each of these new features or changes:</p> <ul style="list-style-type: none"> • Constructing Java Classpath Now Uses librarypath • Notebook Support for Word 97 to Be Discontinued
Mathematics	<p>See the Compatibility Considerations subheading for each of these new features or changes:</p> <ul style="list-style-type: none"> • Different Results When Solving Singular Linear Systems on Intel Systems; Inconsistent NaN Propagation • funm Returns Status Information; New Output Might Result In Error
Programming	<p>See the Compatibility Considerations subheading for each of these new features or changes:</p> <ul style="list-style-type: none"> • datevec Support of Empty String Argument • ftell Returning Invalid Position in Rare Cases • Logical OR Operator in regexp Expressions Might Yield Different Results from Previous Version • Multiple Declarations of Persistent Variables No Longer Supported
Graphics and 3-D Visualization	<p>See the Compatibility Considerations subheading for each of these new features or changes:</p> <ul style="list-style-type: none"> • Cannot Dock Figures on Macintosh • Plotting Tools Not Working on Macintosh • Not All Macintosh System Fonts Are Available • Preview Java Figures on the Macintosh

Area	New Features and Changes with Version Compatibility Impact
Creating GUIs	See the Compatibility Considerations subheading for each of these new features or changes: <ul style="list-style-type: none">• FIG-File Format Change• Panels, Button Groups, and ActiveX Components• Windows XP Display of Push and Toggle Buttons
External Interfaces	See the Compatibility Considerations subheading for this new feature or change: <ul style="list-style-type: none">• Specifying the Search Path for Java Native Method DLLs

Version 7 (R14) Compatibility Summary for MATLAB

For other versions, see the Compatibility Summary for MATLAB.

Area	New Features and Changes with Version Compatibility Impact
Desktop Tools and Development Environment	<p data-bbox="457 309 1302 371">See the Compatibility Considerations subheading for each of these new features or changes:</p> <ul data-bbox="464 392 1322 1194" style="list-style-type: none"><li data-bbox="464 392 832 418">• terminal Function Removed<li data-bbox="464 440 1010 466">• license Function Results Modified Slightly<li data-bbox="464 489 1317 552">• Tab Completion Graphical Interface Added; Removed Preference to Limit Completions<li data-bbox="464 574 991 600">• Parentheses Matching Support Removed<li data-bbox="464 623 1297 685">• Documentation Now Automatically Installed; Not Accessible from CD-ROMs and docroot Not Supported<li data-bbox="464 708 1243 734">• web Function Now Opens MATLAB Web Browser By Default<li data-bbox="464 756 1292 782">• HTML Documentation Not Viewable with -nojvm Startup Option<li data-bbox="464 805 1326 831">• Built-In Functions Now Treated Like Other M-Files on Search Path<li data-bbox="464 854 1040 880">• savepath Function Added to Replace path2rc<li data-bbox="464 902 995 928">• Create Block Comments Using %{ and %}<li data-bbox="464 951 1044 977">• dbstack Function Supports Nested Functions<li data-bbox="464 999 1141 1025">• dbstatus Function Supports Conditional Breakpoints<li data-bbox="464 1048 896 1074">• Rapid Code Iteration Using Cells<li data-bbox="464 1097 928 1123">• Profiler for Measuring Performance<li data-bbox="464 1145 788 1171">• Source Control Changes

Area	New Features and Changes with Version Compatibility Impact
Mathematics	<p>See the Compatibility Considerations subheading for each of these new features or changes:</p> <ul style="list-style-type: none">• Obsolete Functions• Integer Data Type Functions Now Round Instead of Truncate• Changes to Behavior of Concatenation• New Class Inputs for sum• max and min Now Have Restrictions on Inputs of Different Data Types• Matrix, Trigonometric, and Other Math Functions No Longer Accept Inputs of Type char• New Names for Demos expm1, expm2, and expm3

Area	New Features and Changes with Version Compatibility Impact
Programming	<p>See the Compatibility Considerations subheading for each of these new features or changes:</p> <ul style="list-style-type: none"> • MATLAB Stores Character Data As Unicode; Making Release 14 MAT-files Readable in Earlier Versions • MAT-Files Generated By Release 14 Beta2 Must Be Reformatted • Additional Bytes Reserved in MAT-File Header; Do Not Write To Reserved Space • Case-Sensitivity in Function and Directory Names; Can Affect Which Function MATLAB Selects • Differences Between Built-Ins and M-Functions Removed; Can Affect Which Function MATLAB Selects • Change to How evalin Evaluates Dispatch Context • New Calling Syntax for Function Handles; Replace eval With New Syntax • Arrays of Function Handles • Regular Expressions Accept String Vector; No Longer Support Character Matrix Input • Colon Operator on char Now Returns a char • Reading Date Values with xlsread; Conversion No Longer Necessary • Changes to Error Message Format
3-D Graphics and Visualization	<p>See the Compatibility Considerations subheading for each of these new features or changes:</p> <ul style="list-style-type: none"> • Plotting Tools Not Working on Macintosh • Using -nojvm Option Prevents Plotting Tools Use • MATLAB 6 Version Property Editor • Cannot Dock Figures on Macintosh • Not All Macintosh System Fonts Are Available

Area	New Features and Changes with Version Compatibility Impact
Creating GUIs	See the Compatibility Considerations subheading for each of these new features or changes: <ul style="list-style-type: none">• New Container Components• New Syntax for uigetfile and uiputfile• Frames Not Available in GUIDE Layout Editor
External Interfaces	See the Compatibility Considerations subheading for each of these new features or changes: <ul style="list-style-type: none">• Saving Character Data with Unicode Encoding• Display of Interface Handles• Recompile MEX-Files on GLNX86 and Macintosh• Shared Libraries Now In /bin/\$ARCH

Version 6.5.1 (R13SP3) Compatibility Summary for MATLAB

For other versions, see the Compatibility Summary for MATLAB.

New Features and Changes with Version Compatibility Impact

The compatibility considerations are

- Rebuild Macintosh MEX-files
 - Function and Data Type Names in Generic DLL Interface
-