

MATLAB[®]

The Language of Technical Computing

If You Are Upgrading to MATLAB 5.2 from ...	Read These Sections ...
MATLAB 5.1	Chapter 1 “Upgrading from MATLAB 5.1 to MATLAB 5.2” in Chapter 4
MATLAB 5.0	Chapters 1 and 2 “Upgrading from MATLAB 5.0 to MATLAB 5.2” and “Upgrading from MATLAB 5.1 to MATLAB 5.2” in Chapter 4
MATLAB 4	All

Computation

Visualization

Programming

MATLAB 5.2 Product Family New Features

Version 5.2

How to Contact The MathWorks:



508-647-7000 Phone



508-647-7001 Fax



The MathWorks, Inc. Mail
24 Prime Park Way
Natick, MA 01760-1500



<http://www.mathworks.com> Web
<ftp.mathworks.com> Anonymous FTP server
<comp.soft-sys.matlab> Newsgroup



support@mathworks.com Technical support
suggest@mathworks.com Product enhancement suggestions
bugs@mathworks.com Bug reports
doc@mathworks.com Documentation error reports
subscribe@mathworks.com Subscribing user registration
service@mathworks.com Order status, license renewals, passcodes
info@mathworks.com Sales, pricing, and general information

MATLAB 5.2 Product Family New Features

© COPYRIGHT 1984 - 1998 by The MathWorks, Inc.

The software described in this document is furnished under a license agreement. The software may be used or copied only under the terms of the license agreement. No part of this manual may be photocopied or reproduced in any form without prior written consent from The MathWorks, Inc.

U.S. GOVERNMENT: If Licensee is acquiring the Programs on behalf of any unit or agency of the U.S. Government, the following shall apply: (a) For units of the Department of Defense: the Government shall have only the rights specified in the license under which the commercial computer software or commercial software documentation was obtained, as set forth in subparagraph (a) of the Rights in Commercial Computer Software or Commercial Software Documentation Clause at DFARS 227.7202-3, therefore the rights set forth herein shall apply; and (b) For any other unit or agency: NOTICE: Notwithstanding any other lease or license agreement that may pertain to, or accompany the delivery of, the computer software and accompanying documentation, the rights of the Government regarding its use, reproduction, and disclosure are as set forth in Clause 52.227-19 (c)(2) of the FAR.

MATLAB, Simulink, Handle Graphics, and Real-Time Workshop are registered trademarks and Stateflow and Target Language Compiler are trademarks of The MathWorks, Inc.

Other product or brand names are trademarks or registered trademarks of their respective holders.

Printing History: January 1998 Revised for MATLAB 5.2

Introduction	xi
How to Use This Document	xi
References and Links to Other Documents	xi
MATLAB 5.2 Product Family Documentation Set	xiii
Manuals Reprinted for 5.2	xiii
Manuals Updated Online for 5.2	xiii

MATLAB 5.2 Enhancements

1 |

What's New in MATLAB 5.2?	1-2
Enhancements to MATLAB	1-2
Upgrades to Simulink, Real-Time Workshop, Toolboxes, and Blocksets	1-3
New Power System Blockset	1-4
MATLAB Language Enhancements	1-5
Support for try/catch	1-5
Warning Messages	1-5
Setting the Recursion Limit	1-5
New Mathematical Functions	1-6
New String Comparison Functions	1-6
M-File Locking	1-6
Persistent Variables	1-7
File and Directory Handling	1-7
Enhancement to load	1-7
Cell Array of Strings	1-8
Enhancement to strjust	1-8
Change in clc and home Behavior	1-8
Additional Functions Changed in MATLAB 5.2	1-8
PC and UNIX Environment Tools Enhancements	1-9
Changes to the MATLAB Editor/Debugger	1-9

Array Editor Added for PC and UNIX Platforms	1-9
More New Tools for UNIX Environments	1-10
SGI64 Fully Supported	1-10
Online Documentation Enhancements	1-11
Full-Text Search Facility	1-11
Reference Page Navigation	1-11
The doc Command	1-11
Japanese Help Desk	1-11
ActiveX Support Enhanced	1-12
HDF File Format Support	1-13
Visualization Enhancements	1-14
Support for OpenGL Renderers	1-14
New View Control Commands	1-14
Complex Camera Operations	1-14
Camera and Axis Control	1-15
New Lighting Convenience Commands	1-16
Support for Predefined Paper Types	1-16
Mechanism to Hide Objects from Selection	1-17
New Behavior for newplot, clf, and cla	1-17
Behavior of newplot	1-17
Behavior of clf and cla	1-18
GUI Development Enhancements	1-19
New Units Property Value	1-19
Tooltips	1-19
Toggle Buttons	1-19
Displaying Truecolor Images on Controls	1-19
Context Menus	1-20
MATLAB Compiler	1-21
Compatibility Release	1-21
Improved Installation and Configuration Process	1-21
Enhanced Support for Windows 95 and NT Compilers	1-21
Building Simulink CMEX S-Functions	1-21
Additional Enhancements	1-22

Documentation	1-22
MATLAB C Math Library 1.2	1-23
Compatibility Release	1-23
New Features	1-23
Documentation	1-24
MATLAB C++ Math Library 1.2	1-25
Compatibility Release	1-25
New Features	1-25
Documentation	1-26
Simulink 2.2	1-27
User Interface	1-27
Toolbar	1-27
Status Bar	1-27
Context-Sensitive Menus	1-27
Automatic Block Connection	1-28
Block Properties Dialog Box	1-28
Undoing Breaking of Library Links	1-28
Simulation	1-28
Block Priorities	1-28
Additional Solvers	1-28
Debugger	1-29
Tunable Mask Parameters	1-29
Level 2 S-Functions	1-29
Merge Block	1-29
Non-Algebraic Feedback Loops	1-29
Model Construction Commands	1-30
Object Parameters	1-30
Dialog Parameters	1-30
Lines/Annotations API	1-30
Printing	1-31
Print Frames	1-31
Real-Time Workshop 2.2	1-32
Asynchronous Processes	1-32
RTWlib	1-32
Merge Block Added	1-32
Level 2 S-Functions	1-33

Stateflow 1.0.6	1-34
Toolboxes and Blocksets	1-35
Power System Blockset 1.0	1-36
Communications Toolbox 1.3	1-36
Control System Toolbox 4.1	1-37
DSP Blockset 2.2	1-38
Data Frames	1-39
Filter Realization Wizard	1-40
New and Enhanced Blocks	1-41
For Users Upgrading from Version 1.0a	1-43
Financial Toolbox 1.1	1-44
Term Structure Functions	1-44
Derivatives Function	1-44
Portfolio Analysis Function	1-45
Date Functions	1-45
Demo of an Excel Link Portfolio Optimizer Tool	1-46
Fuzzy Logic Toolbox 2.0	1-46
Graphical User Interface Enhancements	1-46
Fuzzy Algorithm Improvements	1-47
FIS Represented As MATLAB Structures	1-47
More Dimensions Allowed for User-Defined Membership Functions	1-47
Image Processing Toolbox 2.1	1-48
Interactive Pixel Value Display	1-48
Feature Measurement	1-48
Inverse Radon Transform	1-48
Canny Edge Detector	1-48
Other Enhancements	1-48
Neural Network Toolbox 3.0	1-49
Signal Processing Toolbox 4.1	1-50
Spectral Estimation	1-50
SPTool Graphical User Interface	1-50
Filter Viewer	1-52
General Enhancements	1-53
Spline Toolbox 2.0	1-53
Multivariate Spline Support	1-53
User Interface Enhancements	1-53
Vector-Valued Spline Enhancements	1-53
Additional Enhancements	1-54

What's New in MATLAB 5.1?	2-2
Enhancements to MATLAB	2-2
Upgrades to Simulink, Real-Time Workshop, Toolboxes, and Blocksets	2-3
New Products	2-3
Language and Development Environment Enhancements	2-4
find Returns Empty Matrix	2-4
Multibyte Character Support	2-4
Removal of Microsoft Windows TCP/IP Issues	2-4
Notebook Support for Office 97	2-4
PC Editor/Debugger	2-5
Editing Arrays on Macintosh	2-5
The Path Browser on the Macintosh	2-6
Macintosh Debugger	2-7
Handle Graphics Enhancements	2-8
Scatter Plot Functions Added	2-8
X-Windows Support for uisetcolor	2-8
Previously Undocumented Functions	2-8
Printing Patches and Surfaces	2-8
TIFF and JPEG Device Drivers	2-9
TIFF	2-9
Compression	2-10
JPEG	2-10
Compression	2-11
TIFF Preview Images for Encapsulated PostScript	2-12
Alternate Method on the Macintosh	2-12
API Enhancements for Windows NT	2-13
Setting Up the Compiler Location	2-13
API Enhancements for Macintosh	2-14
Installation Notes for Using CodeWarrior 11 with the MATLAB API	2-14

Using CodeWarrior 10 and 11 with the MATLAB API	2-14
Building Applications on the Power Macintosh and 68K Macintosh	2-14
Stateflow	2-16
Addition to the Simulink Modeling Environment	2-16
Stateflow Coder	2-16
Mapping Toolbox	2-17

MATLAB 5.0 Enhancements

3

MATLAB 5.0 Enhancements	3-2
Enhanced Programming and Application Development Tools	3-2
New Data Types, Structures, and Language Features	3-3
Faster, Better Graphics and Visualization	3-3
More Mathematical and Data Analysis Tools	3-4
Enhancements to Simulink and Application Toolboxes	3-4
New Data Constructs	3-5
Multidimensional Arrays	3-5
Cell Arrays	3-7
Structures	3-7
MATLAB Objects	3-8
Objects	3-8
Character Arrays	3-9
Programming Capabilities	3-10
Flow-Control Improvements	3-10
M-File Programming Tools	3-12
Variable Number of Input and Output Arguments	3-12
Multiple Functions Within an M-File	3-12
M-File Profiler	3-12
Pseudocode M-Files	3-12

New and Enhanced Language Functions	3-14
Subscripting and Assignment Enhancements	3-16
Integer Bit Manipulation Functions	3-16
Dimension Specification for Data Analysis Functions	3-17
Wildcards in Utility Commands	3-18
Empty Arrays	3-18
New Data Analysis Features	3-20
Higher-Dimension Interpolation	3-21
griddata Based on Delaunay Triangulation	3-21
Set Theoretic Functions	3-21
New and Enhanced Handle Graphics Features	3-23
Plotting Capabilities	3-23
Filling Areas	3-23
Bar Chart Enhancements	3-23
Labels for Patches and Surfaces	3-24
Marker Style Enhancement	3-24
Stem Plot Enhancements	3-24
Three-Dimensional Plotting Support	3-24
Data Visualization	3-25
New Viewing Model	3-25
New Method for Defining Patches	3-25
Triangular Meshes and Surfaces	3-25
Improved Slicing	3-25
Contouring Enhancements	3-25
New zoom Options	3-26
Graphics Presentation	3-26
Enhancements to Axes Objects	3-26
Color Enhancements	3-26
Text Object Enhancements	3-27
Improved General Graphics Features	3-28
Lighting	3-28
print Command Revisions	3-29
Additional print Device Options	3-29

Image Support	3-30
Truecolor	3-31
Reading and Writing Images	3-31
8-Bit Images	3-31
Indexed Images	3-32
Colormaps	3-33
Truecolor Images	3-33
New and Enhanced Handle Graphics Object Properties ..	3-34
Improvements to Graphical User Interfaces (GUIs)	3-42
General GUI Enhancements	3-42
Guide	3-43
Enhanced Application Program Interface (API)	3-44
New Fundamental Data Type	3-44
New Functions	3-44
Support for Structures and Cells	3-44
Support for Multidimensional Arrays	3-44
Support for Nondouble Precision Data	3-45
Enhanced Debugging Support	3-45
Enhanced Compile Mechanism	3-45
MATLAB 4 Feature Unsupported in MATLAB 5.0	3-45
Non-ANSI C Compilers	3-45
New Platform-Specific Features	3-46
Microsoft Windows	3-46
Path Browser	3-46
Workspace Browser	3-47
M-File Editor/Debugger	3-47
Command Window Toolbar	3-48
New Dialog Boxes	3-49
16-bit Stereo Sound	3-49

Macintosh	3-49
User Interface Enhancements	3-50
Command Window Features	3-50
Command History Window	3-50
Path Browser	3-51
Workspace Browser	3-52
M-File Debugger	3-53
Editor Features	3-53
UNIX Workstations	3-54
Figure Window Toolbar	3-54
Path Editor	3-55
Simplified Installation Procedure	3-56

Upgrading to MATLAB 5.2

4

Migrating to MATLAB 5.2	4-2
Roadmap for Different Migration Routes	4-2
Toolboxes and Blocksets	4-2
Upgrading from MATLAB 5.1 to MATLAB 5.2	4-3
Change to clear Behavior	4-3
try, catch, and persistent Are Now Keywords	4-3
Matrix Assignment	4-3
Change to Method Search Order	4-3
API Memory Management Compatibility Issue	4-4
Improperly Destroying an mxArray	4-5
Incorrectly Constructing a Cell or Structure mxArray	4-5
Creating a Temporary mxArray with Improper Data	4-6
Potential Memory Leaks	4-7
Recommendation: MEX-Files Should Destroy Their Own Temporary Arrays	4-8
Upgrading from MATLAB 5.0 to MATLAB 5.2	4-9
Upgrading from MATLAB 4 to MATLAB 5.2	4-11
Converting M-Files to MATLAB 5.0	4-11

Converting MATLAB 4 External Interface Programs to the MATLAB 5.0 Application Program Interface	4-25
General Considerations	4-25
Microsoft Windows Considerations	4-28
UNIX Considerations	4-29
Macintosh Considerations	4-29
VMS Considerations	4-29
Conversion	4-30
Recoding C Code for MATLAB 5.0 Compliance	4-34
Upgrading Toolboxes and Blocksets	4-40
Fuzzy Logic Toolbox: Updating FIS Models	4-40
DSP Blockset: Upgrading from Version 1.0a	4-40

Introduction

This document highlights the new features of MATLAB® 5.2 and associated products that have been updated or introduced along with MATLAB 5.2.

Note: This document discusses the whole MATLAB 5.2 product family, including products for which you might not be currently licensed. If you are interested in purchasing a license for a product, please contact your Sales representative at The MathWorks or your distributor.

This document provides the information you need to upgrade to MATLAB 5.2, whether you are currently using MATLAB 4, MATLAB 5.0, or MATLAB 5.1.

How to Use This Document

If You Are Upgrading to MATLAB 5.2 from ...	Read These Sections ...
MATLAB 5.1	Chapter 1 "Upgrading from MATLAB 5.1 to MATLAB 5.2" in Chapter 4
MATLAB 5.0	Chapters 1 and 2 "Upgrading from MATLAB 5.0 to MATLAB 5.2" and "Upgrading from MATLAB 5.0 to MATLAB 5.2" in Chapter 4
MATLAB 4	All

References and Links to Other Documents

Throughout Chapters 1, 2, and 3, there are references to other documents for additional detailed information about new features highlighted in this document.

In the HTML version of this document, Chapter 1 includes several direct links to reference documentation describing specific 5.2 features in more detail. By clicking on those links, you can go directly to the more detailed information. For example, clicking on a highlighted command name in a table displays the documentation for that command. Use your browser's **Back** button to return to this document.

MATLAB 5.2 Product Family Documentation Set

Manuals Reprinted for 5.2

The following manuals have been printed and distributed to existing customers of MATLAB and its optional associated products, as part of their update package:

- *Late-Breaking News for the MATLAB 5.2 Product Family*
- *Application Program Interface Guide*
- *MATLAB Compiler User's Guide*
- *MATLAB C Math Library User's Guide*
- *MATLAB C++ Math Library User's Guide*
- *Communications Toolbox New Features Guide (Version 1.3)*
- *Financial Toolbox User's Guide*
- *Fuzzy Logic Toolbox User's Guide*
- *Neural Network User's Guide*
- *Spline Toolbox User's Guide*

Manuals Updated Online for 5.2

All the updated manuals listed above are also available online, via the Help Desk, in PDF format. The *Late-Breaking News for the MATLAB 5.2 Product Family* and the reference sections of the documentation for most of these toolboxes and blocksets are also available in HTML form.

In addition, the following manuals have been updated for 5.2 in PDF form. The reference section of almost all these manuals is available in HTML format as well.

- *Using MATLAB*
- *Using MATLAB Graphics*
- *MATLAB Function Reference (includes language and graphics)*
- *Application Program Interface Reference*
- *MATLAB Notebook User's Guide*
- *MATLAB C Math Library Reference*
- *MATLAB C++ Math Library Reference*

-
- *Using Simulink*
 - *Real-Time Workshop User's Guide*
 - *Stateflow User's Guide*
 - *Control System Toolbox User's Guide*
 - *Financial Toolbox User's Guide*
 - *Frequency Domain System Identification Toolbox User's Guide*
 - *Image Processing Toolbox User's Guide*
 - *Mapping Toolbox*
 - *Partial Differential Equation Toolbox User's Guide*
 - *Robust Control Toolbox User's Guide*
 - *Signal Processing Toolbox User's Guide*
 - *System Identification Toolbox User's Guide*
 - *Wavelet Toolbox User's Guide*
 - *Power System Blockset User's Guide*
 - *DSP Blockset User's Guide*

MATLAB 5.2 Enhancements

What's New in MATLAB 5.2?	1-2
MATLAB Language Enhancements	1-5
PC and UNIX Environment Tools Enhancements	1-9
Online Documentation Enhancements	1-11
ActiveX Support Enhanced	1-12
HDF File Format Support	1-13
Visualization Enhancements	1-14
GUI Development Enhancements	1-19
MATLAB Compiler	1-21
MATLAB C Math Library 1.2	1-23
MATLAB C++ Math Library 1.2	1-25
Simulink 2.2	1-27
Real-Time Workshop 2.2	1-32
Stateflow 1.0.6	1-34
Toolboxes and Blocksets	1-35

What's New in MATLAB 5.2?

MATLAB 5.2 completes bringing the whole MATLAB product family up to a MATLAB 5 level: the MATLAB Compiler and MATLAB C and C++ Math Libraries now work with MATLAB 5 and its associated products.

MATLAB 5.2 also adds many important application development and visualization features.

In addition, other licensed products have been updated with the release of MATLAB 5.2:

- Simulink[®] 2.2 includes several enhancements, including new user interface features and additional simulation features.
- Real-Time Workshop[®] 2.2 includes important enhancements.
- New versions of most toolboxes are available with MATLAB 5.2 (see the “Toolboxes and Blocksets” on page 1-35 for a complete list).
- The Power System Blockset is introduced with Version 5.2.

Enhancements to MATLAB

The language and development environment enhancements introduced with MATLAB 5.2 include:

- New MATLAB language functions, implementing features such as `try/catch` error handling, additional ODE functions, and M-file locking.
- Expanded application development tools for the Microsoft Windows 95 and NT and UNIX platforms.
- Improved Help Desk, with a much faster search engine that supports full-text searches, and easier HTML reference page navigation.
- Support for two new ActiveX technologies: ActiveX control containment and ActiveX Automation client capabilities, so that now MATLAB can both control and be controlled by other ActiveX components.
- Support for HDF (Hierarchical Data Format) files.

MATLAB 5.2 also includes the following visualization and GUI development enhancements:

- Support for OpenGL rendering to improve performance dramatically for many visualization applications.
- Enhanced visualization features, including enhanced camera control, simplified placement of Light objects, tighter and more consistent control of graphics object hiding.
- A print frame editor enables you to create custom layouts for printing Simulink model diagrams.
- Additional GUI development features, including the ability to define the location and size of user interface objects in units that are based on the size of the default system font, and to add tooltips, toggle buttons, context menus, etc.

Upgrades to Simulink, Real-Time Workshop, Toolboxes, and Blocksets

Simulink 2.2 includes several enhancements, including Level 2 S-function support, new user interface features on the PC and Macintosh platforms, additional simulation features, some new blocks and commands, and the ability to add print frames (header and footer annotation) to printouts of Simulink models.

Real-Time Workshop 2.2 utilizes the Simulink Level-2 S-function feature to support Interrupt Service Routines (ISRs) for VxWorks, customized ISRs for your target system, multiple input/multiple output S-functions, and parameter checking while running.

Almost all toolboxes and blocksets were updated for MATLAB 5.2. Toolboxes and blocksets with especially significant enhancements for the MATLAB 5.2 release include:

- Communications Toolbox 1.3
- Control System Toolbox 4.1
- DSP Blockset 2.2
- Financial Toolbox 1.1
- Fuzzy Logic Toolbox 2.0
- Image Processing Toolbox 2.1

- Neural Network Toolbox 3.0
- Signal Processing Toolbox 4.1
- Spline Toolbox 2.0

New Power System Blockset

The Power System Blockset 1.0 is introduced with MATLAB 5.2. This new blockset is described in more detail later in this chapter.

MATLAB Language Enhancements

Links to Command Descriptions Clicking on the command name in the following tables displays the documentation for that command. Use your browser's **Back** button to return to this document.

Support for try/catch

MATLAB 5.2 adds functions to support try/catch error handling.

Function	Description
catch	Begin catch block.
try	Begin try block.

Warning Messages

The new `lastwarn` function, depending how it is called, returns either a string containing the last warning message issued by MATLAB, an empty string matrix until the next warning is encountered, or sets the last warning message to a specified string.

Setting the Recursion Limit

You can now set the limit for recursion so that you will receive an error instead of being forced out of MATLAB when the recursion limit is reached. The default recursion limit is 500 on PC and UNIX platforms and 200 on Macintosh platforms. To change the recursion limit, change the following line:

```
set(0, 'recursionlimit', limitnumber)
```

in your `matlabrc.m` file in the `toolbox/local` directory.

New Mathematical Functions

MATLAB 5.2 provides these new mathematical functions.

Function	Description
<code>cholinc</code>	Sparse incomplete Cholesky and Cholesky-infinity factorizations.
<code>cholupdate</code>	Rank 1 update to Cholesky factorization.
<code>ifftshift</code>	Inverse fast Fourier transform shift.
<code>ode23t</code>	Solve moderately stiff problems for a solution without numerical damping.
<code>ode23tb</code>	Solve stiff systems using crude error tolerances. May also be used if there is a mass matrix.
<code>qrupdate</code>	Rank 1 update to QR factorization.

New String Comparison Functions

MATLAB 5.2 provides two additional string comparison functions.

Function	Description
<code>strcmpi</code>	Compare strings ignoring case.
<code>strncmpi</code>	Compare first n characters of strings ignoring case.

M-File Locking

You can now lock (and unlock) an M-file so that `clear` does not clear that M-file from memory.

Function	Description
<code>mislocked</code>	True if M-file cannot be cleared.
<code>ml lock</code>	Prevent M-file clearing.
<code>munlock</code>	Permit M-file clearing.

Persistent Variables

A variable may be defined as persistent (with the keyword `persistent`) so that it does not change value from one call to another. Persistent variables may be used within a function only. Persistent variables remain in memory until the M-file is cleared or changed. `persistent` is exactly like `global`, except that the variable name is not in the global workspace, and the value is reset if the M-file is changed or cleared.

Three MATLAB functions support the use of persistent variables (see “M-File Locking” above):

- `mislocked`
- `mllock`
- `munlock`

File and Directory Handling

You can copy a file and make a directory from within MATLAB.

Function	Description
<code>copyfile</code>	Copy a file.
<code>mkdir</code>	Make a directory.

Enhancement to load

MATLAB 5.2 adds a new option to the `load` function:

```
S = load(...)
```

returns the contents of a MAT-file as a structure instead of directly loading the file into the workspace. The field names in `S` match the names of the variables that were retrieved. When the file is ASCII, `S` will be a double precision array.

Cell Array of Strings

You can now use a cell array of strings with the following functions:

- `intersect`
- `ismember`
- `lower`
- `setdiff`
- `setxor`
- `sort`
- `union`
- `unique`
- `upper`

Enhancement to `strjust`

The `strjust` function now does right, left, and center justification.

Change in `clc` and `home` Behavior

The `clc` and `home` commands now both clear the Command Window. After issuing either of these commands, it is no longer possible to scroll back to see previous contents of the command window.

Additional Functions Changed in MATLAB 5.2

In addition to the above functions, the MATLAB 5.2 version of the following functions have minor changes, generally to reflect the addition of the new functions described above (e.g., `clear` does not clear if `mllock` is called first).

- `clear`
- `end`
- `horzcat`
- `lasterr`
- `paren`
- `spline`
- `strcmp`
- `strncmp`

PC and UNIX Environment Tools Enhancements

MATLAB 5.2 provides enhanced environment tools for the PC (Microsoft Windows 95 and NT) platform and introduces environment tools for the UNIX platform. These enhancements are described in detail in Chapter 2 of the online (PDF) version of *Using MATLAB*.

Changes to the MATLAB Editor/Debugger

With MATLAB 5.2, the Editor/Debugger provides a new **Tools** menu for Microsoft Windows 95, Windows NT, and UNIX platforms. Some of the options that were under the **View** menu in previous releases on the PC are now found under the **Tools** menu. MATLAB 5.2 provides a tabbed dialog that allows you to set **General** and **Editor** options. You do so from the **Tools** menu, by selecting **Options**.

You can now use MATLAB to add your own commands to the Editor, by using the **Customize** option that appears as a submenu of the **Tools** menu. Commands that you add will also work with the Path Browser and Array Editor although the results may differ. Chapter 2 of *Using MATLAB* provides a table that explains these differences.

You can set up the Editor so that the values of MATLAB variables are expanded and displayed in the Editor window as the cursor *hovers* over a variable. To do so, under **General** options, check **Show Data Tips**.

Also under **General** options, if you check **Show worksheet style tabs**, the main Editor window displays a tab at the bottom for each open file. This allows quick navigation among all open files.

Also, you can control the Editor's font, style, and size. In previous releases of MATLAB, font control was available only for the Command Window. In MATLAB 5.2, you can select **Font** from the **Tools** menu to control Editor fonts.

Array Editor Added for PC and UNIX Platforms

MATLAB 5.2 provides an Array Editor for both the PC and UNIX platforms. This tool allows you to view and edit two-dimensional numeric arrays.

More New Tools for UNIX Environments

Several tools that were available on PCs in earlier versions of MATLAB are now also available in MATLAB 5.2 for some UNIX machines (Sol2, Sun4, SGI, and HP 700):

- Built-in MATLAB Editor/Debugger
- Workspace Browser
- Path Browser

SGI64 Fully Supported

The SGI64 platform is fully supported for MATLAB 5.2. The SGI64 platform was supported only as a Beta product in previous MATLAB 5 versions.

Note: The Symbolic Math Toolbox 2.0.1 and Extended Symbolic Math Toolbox 2.0.1 are not available for the SGI64 platform; however, the SGI image can be used on the SGI64 platform.

Online Documentation Enhancements

Full-Text Search Facility

The 5.2 Help Desk includes a full-text search facility for the HTML online documentation. You can access the full-text search facility from the top page of the Help Desk or from the “Search” link on reference pages.

Reference Page Navigation

The 5.2 HTML reference pages provide additional navigational aids. The “Examples” and “See Also” links at the top of the first reference page for a function allow you to jump directly to the examples or to links to associated functions.

Also at the top of the reference pages is a “Go to function” edit box. Enter the name of the function and press the **Enter** key to see the reference page for that function.

The doc Command

The `doc` command now accesses the HTML reference documentation for all MathWorks products for which HTML reference documentation has been installed. Before Version 5.2, the `doc` command only accessed the documentation for MATLAB functions.

Japanese Help Desk

MATLAB 5.2 provides a Japanese version of the Help Desk, in addition to the English version.

Note: Most of the Japanese documentation is at the 5.0 or 5.1 level.

During the installation process you can specify what, if any, Japanese documentation you want to install.

If you install any Japanese documentation, the Japanese Help Desk will be displayed when you use the `hel pdesk` command.

ActiveX Support Enhanced

MATLAB 5.2 supports two new ActiveX technologies: ActiveX control containment and ActiveX Automation client capabilities. ActiveX controls are application components, which can be both visually and programmatically integrated into an ActiveX control container; in the MATLAB context, this would be figure windows. Some examples of useful ActiveX controls are the Microsoft Internet Explorer Web Browser control, the Microsoft Windows Communications control for serial port access, and the graphical user interface controls delivered with the Visual Basic development environment.

Prior to 5.2, MATLAB supported ActiveX Automation *server* capabilities. When MATLAB is controlled by another component, it is acting as an automation server. MATLAB 5.2 adds support for ActiveX Automation *client* capabilities. When MATLAB controls another component, MATLAB is the automation client, and the other component is the automation server. In other words, MATLAB 5.2 ActiveX Automation allows MATLAB to both control and be controlled by other ActiveX components.

This feature is described in more detail in Chapter 7 of the *Application Program Interface Guide*.

HDF File Format Support

MATLAB 5.2 extends the support for HDF files beyond that previously provided by `imread` and `imwrite`. This additional support is provided through an interface to different HDF formats via new MATLAB functions that enable you to access the HDF library developed and supported by the National Center for Supercomputing Applications (NCSA). MATLAB 5.2 also provides an extensible gateway for reading and writing HDF files.

To use these functions, you must be familiar with the HDF library. Documentation for the library is available on the NCSA HDF Web page at <http://hdf.ncsa.uiuc.edu>. MATLAB provides extensive command line help for each of these functions.

Function	Interface
<code>hdfan</code>	Multifile annotation
<code>hdfdf24</code>	24-bit raster image
<code>hdfdf8</code>	8-bit raster image
<code>hdfh</code>	HDF H interface
<code>hdfhd</code>	HDF HD interface
<code>hdfhe</code>	HDF HE interface
<code>hdfml</code>	Gateway utilities
<code>hdfsd</code>	Multifile scientific data set
<code>hdfv</code>	Vgroup
<code>hdfvf</code>	Vdata VF functions
<code>hdfvh</code>	Vdata VH functions
<code>hdfvs</code>	Vdata VS functions

Visualization Enhancements

Support for OpenGL Renderers

The OpenGL renderer is available on many computer systems. This renderer is generally faster than MATLAB's painters or zbuffer renderers in some cases. If your system has graphics hardware that is available to OpenGL, MATLAB uses it to achieve even greater performance improvements. This results in greatly improved drawing performance, particularly with graphics cards that support OpenGL. See the Figure Renderer property in the *MATLAB Function Reference* for more information.

New View Control Commands

MATLAB 5.2 contains a number of new commands that simplify camera positioning and aspect ratio control. These commands implement operations similar to those associated with movie camera operation – dollying, panning, rolling, as well as some that are more typically associated with computer graphics, such as orbiting the camera around the scene and selecting a method for projecting the 3-D scene on the computer screen.

Links to Command Descriptions Clicking on the command name in the following tables displays the documentation for that command. Use your browser's **Back** button to return to this document.

Complex Camera Operations

This table lists commands that simplify the process of moving the camera in a well defined manner through three-dimensional space.

Function or Property	Purpose
Camera Graphics Convenience Functions	
<code>camdolly</code>	Translate camera position and camera target (analogous to dollying a movie camera).

Function or Property	Purpose
camorbit	Rotate camera position around camera target (rotation specified in degrees).
campan	Rotate camera target around camera position (rotation specified in degrees).
camroll	Rotate camera about camera viewing axis (rotation specified in degrees).
camzoom	Zoom camera in or out on a scene by specified zoom factor.

Camera and Axis Control

This table lists new commands that provide a convenient way to set Axes properties. These properties control camera positioning as well as axis limits and aspect ratio.

Function or Property	Purpose
campos	Set or get camera position and camera position mode.
camproj	Set or get camera projection type to orthographic or perspective.
camtarget	Set or get camera target and camera target mode.
camup	Set or get camera up vector and camera up vector mode.
camva	Set or get camera view angle and camera view angle mode.
daspect	Set or get data aspect ratio and data aspect ratio mode.
pbaspect	Set or get plot box aspect ratio and plot box aspect ratio mode.

Function or Property	Purpose
<code>xlim</code>	Set or get x -axis limits and x -axis limit mode.
<code>ylim</code>	Set or get y -axis limits and y -axis limits mode.
<code>zlim</code>	Set or get z -axis limits and z -axis limits mode.

New Lighting Convenience Commands

MATLAB 5.2 contains two new commands to simplify the placement of Light objects in the Axes.

Function or Property	Purpose
<code>camlight</code>	Create or move a Light object in the camera's coordinate system. This is useful when you want to place a Light at or near the camera and maintain the same relative position as the camera moves.
<code>lightangle</code>	Create or move a Light object in spherical coordinates (i.e., by specifying azimuth and elevation).

Support for Predefined Paper Types

MATLAB supports a number of new predefined paper types. For a list of these paper types, see the Figure PaperType property.

Mechanism to Hide Objects from Selection

All graphics objects have a new property called `HitTest` that enables you to determine if this object can become the current object or in appropriate cases, the current Figure or current Axes (see the `Figure CurrentObject` and `CurrentAxes` properties and the `Root CurrentFigure` property). This feature is useful to exclude certain graphics objects from user interaction (for example, to prevent MATLAB from selecting text annotations that overlay an image as the user clicks on the image to obtain information returned by a callback routine). See the `HitTest` property for an example.

New Behavior for `newplot`, `clf`, and `cla`

The behavior of the `newplot`, `clf`, and `cla` commands is now clearly defined with respect to hidden-handle objects. Basically, there are three options when drawing graphics in existing Figures:

- Add the new graphics without changing any properties or deleting any objects.
- Delete all existing objects whose handles are not hidden before drawing the new objects.
- Delete all existing objects regardless of whether or not their handles are hidden and reset most properties to their defaults before drawing the new objects.

These features are particularly useful for protecting `Uicontrol` objects that comprise part of a user interface constructed with MATLAB.

Behavior of `newplot`

The `newplot` function now always sets the `Figure NextPlot` property to `add` after obeying the current setting. Previously, `newplot`

- Did not reset the `Figure NextPlot` property if its current value was `replacechildren`.
- Did set the `NextPlot` property to its currently defined default after obeying its value of `replace`. (While the factory default is `add`, user-defined settings can change this.)

With MATLAB 5.2, `newplot`

- Always resets the Figure `NextPlot` property to `add` after obeying the current setting (regardless of user-defined defaults set for `NextPlot`).
- Deletes all handle-visible children (i.e., children whose `HandleVisibility` property is set to `on`) when the Figure or Axes `NextPlot` property is `replacechildren`.
- Deletes all children (regardless of the setting of the `HandleVisibility` property) when the Figure or Axes `NextPlot` property is `replace`.

Behavior of `clf` and `cla`

The behavior of the `clf` command without the `reset` argument has not changed: `clf` deletes all children of the current Figure whose handles are not hidden (i.e., their `HandleVisibility` property is set to `on`).

`clf reset` now deletes all children of the current Figure, regardless of the setting of their `HandleVisibility` property. In addition, `clf reset` also resets all Figure properties to their defaults with the exception of `Position`, `Units`, `PaperPosition`, and `PaperUnits`. Previously, `clf reset` deleted only handle-visible objects.

`cla` behaves in a way directly analogous to that of `clf`: `cla` deletes all children of the current Axes whose handles are not hidden (i.e., their `HandleVisibility` property is set to `on`).

`cla reset` deletes all children of the current Axes, regardless of the setting of their `HandleVisibility` property. In addition, `cla reset` also resets all Axes properties to their defaults with the exception of `Position` and `Units`.

GUI Development Enhancements

MATLAB 5.2 provides a number of new features to make it easier for you to develop effective graphical user interfaces (GUIs) for your applications. In the online HTML version of this document, you can use the highlighted links to get more information about these new features.

New Units Property Value

If you write user interfaces intended to be used on more than one computer platform, you may find that you need to adjust the size of controls to accommodate the differences in the size of the fonts used to label the controls.

The new `characters` value for the `Units` property enables you to define `Uicontrol` objects whose sizes are based on the default system font size.

Tooltips

A tooltip is a small rectangle that contains textual information. A tooltip is associated with a `Uicontrol` and appears below the control when the cursor is held over the control for a certain amount of time (determined by system settings).

You define a tooltip for a `Uicontrol` by specifying a string value for the new `ToolTipString` property.

Toggle Buttons

MATLAB 5.2 provides a new style of `Uicontrol` object called a toggle button. Toggle buttons have two states, down (selected) and up (unselected). When you click on a toggle button, its state changes and its callback is executed.

Displaying Truecolor Images on Controls

MATLAB 5.2 supports the ability to display truecolor images on push buttons and toggle buttons.

Context Menu

A context menu is a menu that is attached to an object and is activated by a right-button click on a Microsoft Windows or UNIX system, or a **Ctrl**-click on a Macintosh system (called an *extended click*). Defining a context menu requires that you define a `Uicontextmenu` object and `Uimenu` children and associate the `Uicontextmenu` with the object to which it is attached.

MATLAB Compiler

Compatibility Release

Version 1.2 of the MATLAB Compiler is a compatibility release that brings the MATLAB Compiler into compliance with MATLAB 5. Although the Compiler works with MATLAB 5, it does not support several of the new features of MATLAB 5.

Improved Installation and Configuration Process

Based on customer feedback, installing and configuring the MATLAB Compiler 1.2 is easier than before. The *MATLAB Compiler User's Guide* includes a complete set of recommended steps to perform during installation to ensure that everything is working properly. It includes troubleshooting sections to help diagnose and correct some of the more common installation problems.

Enhanced Support for Windows 95 and NT Compilers

In MATLAB 5.2 and the Compiler, all of the main Compiler vendors and product releases are supported “out of the box” (no additional steps required). These compilers include:

- Watcom 10.6
- Borland 5.0
- MSVC 4.2

Building Simulink CMEX S-Functions

The MATLAB Compiler now supports building Simulink CMEX S-functions from the MATLAB Function block in Simulink. See the *MATLAB Compiler User's Guide* for details.

Additional Enhancements

Version 1.2 of the MATLAB Compiler also includes these enhancements:

- Loading MATLAB MAT-files (using the `load` command) is now supported.
- Compiler-generated command line applications can accept input arguments (text-strings) from a POSIX shell and return a status. In this way command line M-files can be turned into command line executable applications in a POSIX shell.

Documentation

The *MATLAB Compiler User's Guide* has been updated to reflect the current version of the Compiler.

See `<matlab>/toolbox/compiler/Readme.m` for additional details about using the Compiler.

MATLAB C Math Library 1.2

Compatibility Release

Version 1.2 of the MATLAB C Math Library is a compatibility release that brings the library into compliance with MATLAB 5. Although the library works with MATLAB 5, it does not support several of the new features of MATLAB 5.

Note: Many functions have changed between MATLAB 4 and MATLAB 5. These changes are reflected in the MATLAB C Math Library. If you are using the MATLAB Compiler to generate your C Math Library programs, you will need to regenerate your C files from your M-files before the C files will work with the new libraries. If you have written C Math Library programs by hand, you need to make the changes manually.

New Features

Version 1.2 of the C Math Library adds 47 new functions, providing several significant new features, including:

- Indexing (operations like MATLAB's 'x(:, 4) = 1'). This adds three new functions: `ml fArrayAssign`, `ml fArrayRef`, and `ml fArrayDel` etc.
- The new MATLAB 5 suite of ODE solvers. The Version 1.1 library supported only two ODE routines; Version 1.2 supports six of them.
- String handling functions.
- Support for `feval` with multiple output arguments (return values); Version 1.1 supported `feval` of functions with a single output argument.
- Load and save support.
- Improved user-defined error handling.

Documentation

The *C Math Library User's Guide* has been updated to reflect the new version of the library.

Also, HTML online reference documentation has been written for this library for 5.2.

See the release notes (`release.txt`) that are included with the C Math Library for additional details about this version of the library.

MATLAB C++ Math Library 1.2

Compatibility Release

Version 1.2 of the MATLAB C++ Math Library is a compatibility release that brings the library into compliance with MATLAB 5. Although the library works with MATLAB 5, it does not support several of the new features of MATLAB 5.

Note: Many functions have changed between MATLAB 4 and MATLAB 5. However, through the use of C++ function overloading, most of the old functions remain for backward compatibility, and new functions have been added to handle the new functionality (in most cases, with additional function arguments).

If you are generating C++ Math Library programs using the MATLAB Compiler, this should not affect you very much, since the MATLAB Compiler knows about the new functions, and generates the correct code. You will, in some cases, however, have to regenerate your C++ code from your M-files in order to use the new libraries. If you have written stand-alone programs by hand, you may have to edit some of your code before you can link with the new libraries.

New Features

Version 1.2 of the C++ Math Library adds 47 new functions, providing several significant new features, including:

- The new MATLAB 5 suite of ODE solvers. The Version 1.1 library supported only two ODE routines; Version 1.2 supports six of them.
- String handling functions.
- Support for `feval` with multiple output arguments (return values); Version 1.1 supported `feval` functions with a single output argument.
- Load and save support.
- Improved user-defined error handling.

Documentation

The *C++ Math Library User's Guide* has been updated to reflect the new version of the library.

Also, HTML online reference documentation has been written for this library for 5.2.

See the release notes (release1.txt) that are included with the C++ Math Library for additional details about this version of the library.

Simulink 2.2

Simulink 2.2 provides many enhancements relating to these aspects of the product:

- User Interface
- Simulation
- Model Construction Commands
- Printing

These enhancements are described in more detail in the online (PDF) version of *Using Simulink*; changes are highlighted with change bars.

User Interface

Note: See Chapter 3 of *Using Simulink* (online version) for more information about each of these new user interface features.

Toolbar

The PC (Microsoft Windows 95 and NT) and Macintosh versions of Simulink display an optional toolbar below the menu bar in the model and block library windows. You can use the toolbar's buttons to create, save, edit, print, and run models.

Status Bar

The PC and Macintosh versions of Simulink display an optional status bar at the bottom of model and block library windows. The status bar displays the current time and solver when a simulation is running.

Context-Sensitive Menus

The PC and Macintosh (OS8) versions of Simulink display a pop-up menu when you press the right mouse button over a model or block library window. If a block is selected, the menu displays editing, formatting, and property commands applicable to blocks and annotations; otherwise, the menu displays commands applicable to the model or library as a whole.

Automatic Block Connection

You can insert a block having a single input and output into a model by dropping it onto a line segment.

Block Properties Dialog Box

Simulink 2.2 adds a **Block Properties** dialog box, accessed from the **Edit** menu. You can set the following block parameter values:

- Description
- Priority
- Open function
- Attribute format

Undoing Breaking of Library Links

Simulink 2.2 allows you to undo the breaking of library links.

Simulation

Block Priorities

You can assign evaluation priorities to nonvirtual blocks in a model. Higher priority blocks evaluate before lower priority blocks, though not necessarily before blocks that have no assigned priority. You can do this with either the **Block Properties** dialog box from the **Edit** menu or with the `set_param` command. See Chapter 3 of *Using Simulink* (online version) for more information.

Additional Solvers

Simulink 2.2 adds two stiff solvers, `ode23t` and `ode23tb`. See the “Solvers” section in Chapter 4 of *Using Simulink* (online version) for more information.

Debugger

The Simulink debugger allows you to run a model step by step and inspect the values of any variables at any step. See Chapter 12 of *Using Simulink* (online version) for more information.

Tunable Mask Parameters

You can specify whether a mask parameter is tunable, that is, modifiable while a simulation is running. See Chapter 6 of *Using Simulink* (online version) for more information.

Level 2 S-Functions

Simulink 2.2 supports Level 2 S-functions in a C MEX S-function. In particular, these Level 2 S-functions support:

- Multiple input and output ports
- More simulation S-function routines:
 - mdl ProcessParameters, which is called during simulation after parameters have been changed and verified by mdl CheckParameters
 - mdl Start, which performs actions such as allocating memory and attaching to PWork elements.
- mdl RTW, a method for code generation in which your S-function influences the code generation process.

In mdl RTW, you can write additional subrecords into the model .rtw file for the S-function block record. The Target Language Compiler (TLC) file that inlines your S-function can use this information. For more details about Level 2 S-functions, see *Using Simulink* (online version).

Merge Block

The Merge block allows you to combine multiple input lines into a single output line for reduced memory utilization and increased model flexibility. See Chapter 9 of *Using Simulink* (online version) for more information.

Non-Algebraic Feedback Loops

Prior to Version 2.2, Simulink treated as algebraic loops any loops that involved Triggered Subsystems and that were also composed entirely of blocks with direct feedthrough.

With Version 2.2, for Variable-Step solvers, Simulink now takes advantage of the implicit sequencing inherent in triggered execution (i.e., inputs must be stable prior to the trigger and outputs appear after the trigger) to break such loops, thus

- Eliminating the need to invoke the algebraic loop solver
- Providing more meaningful results.

For Fixed-Step solvers, it is still necessary to insert a memory block in the appropriate location (usually at the output of the triggered subsystem) to break such algebraic loops.

See “Algebraic Loops” in Chapter 10 of *Using Simulink* (online version) for more information.

Model Construction Commands

Object Parameters

The command `get_param(obj, 'ObjectParameters')` where `obj` is an object name returns a cell array describing the object’s parameters. See `get_param` in *Using Simulink* (online version) for more information.

Dialog Parameters

The command `get_param(b, 'DialogParameters')`, where `b` is the name of a block, returns a cell array describing the parameters that appear in a block’s parameter dialog. See `get_param` in *Using Simulink* (online version) for more information.

Lines/Annotations API

You can use the `find_system` command to get handles to all the lines and annotations in a model. The returned handles can be used with `get_param` and `set_param` to read and write the line or annotation properties. See `find_system` in Chapter 11 of *Using Simulink* (online version) for more information.

Printing

Print Frames

You can add print frames (customized headers and footers) to printouts of Simulink model diagrams. To edit a print frame, use the new `framedit` command. See “Printing a Block Diagram” in Chapter 3 of *Using Simulink* (online version) for more information.

Real-Time Workshop 2.2

Asynchronous Processes

The Real-Time Workshop now supports asynchronous interrupt handling in VxWorks and provides templates so that you can create your own interrupt handlers for your target hardware. These blocks include:

- Interrupt block
- Task Synchronization block
- Asynchronous Buffer Reader/Writer blocks
- Asynchronous Rate Transition block

For a discussion of asynchronous processes, see the chapter on RTWlib in the *Real-Time Workshop User's Guide* (online version).

RTWlib

The Real-Time Workshop now has a graphical user interface (GUI), called RTWlib, for quick access to:

- VxWorks Tornado — blocks for Matrix, Xycom, and VME Microsystems I/O support; and asynchronous blocks for ISR support
- DOS — blocks for Keithley-Metrabyte I/O support
- Custom Code — blocks for inserting your custom code into the code that the Real-Time Workshop generates from your model
- Create Your Own Asynchronous Library — templates that use the VxWorks asynchronous blocks as a starting point for developing your own asynchronous blocks
- RTW extras — contains a Function-call Configuration block

The GUI is located in the “Blocksets and Toolboxes” Library in the Simulink window. For more information about RTWlib, see the *Real-Time Workshop User's Guide* (online version).

Merge Block Added

The new Merge block merges multiple signals into one for reduced memory utilization and increased model flexibility.

Level 2 S-Functions

Real-Time Workshop 2.2 supports Level 2 S-functions. In particular, these Level 2 S-functions support:

- Multiple input and output ports
- More simulation S-function routines:
 - `mdlProcessParameters`, which is called during simulation after parameters have been changed and verified by `mdlCheckParameters`
 - `mdlStart`, which performs actions such as allocating memory and attaching to pwork elements. Can only be in a C MEX S-function
- `mdlRTW`, a method for code generation in which your S-function influences the code generation process.

In `mdlRTW`, you can write additional subrecords into the `model.rtw` file for the S-function block record. The Target Language Compiler (TLC) file that inlines your S-function can use this information. For more details about Level 2 S-functions, see *Using Simulink* (online version).

Stateflow 1.0.6

Version 1.0.6 of Stateflow™ and Stateflow™ Coder is shipped with MATLAB 5.2. Version 1.0.6 is essentially the same as the Patch Release 1.0.5 that was made available to Stateflow customers via FTP. However, 1.0.6 fixes some software problems that still existed in the patch release.

Stateflow and Stateflow Coder 1.0.6 are *not* supported on the Macintosh platform.

Toolboxes and Blocksets

Almost all of the toolboxes and blocksets have been updated for release with MATLAB 5.2. For many of these toolboxes and blocksets, the updates simply involved fixing software problems and taking more advantage of MATLAB 5 features.

These toolboxes and blocksets listed have been updated for 5.2. The toolboxes and blocksets with significant updates are highlighted with an asterisk and are discussed in more detail in the rest of this chapter (in alphabetical order).

- Communications Toolbox 1.3*
- Control System Toolbox 4.1*
- DSP Blockset 2.2*
- Extended Symbolic Math Toolbox 2.0.1
- Financial Toolbox 1.1*
- Frequency Domain System Identification 2.0.3
- Fuzzy Logic Toolbox 2.0*
- Higher-Order Spectral Analysis Toolbox 2.0.2
- Image Processing Toolbox 2.1*
- LMI Control Toolbox 1.0.4
- Mapping Toolbox 1.0.1
- Model Predictive Control Toolbox 1.0.3
- Mu-Analysis and Synthesis Toolbox 3.0.3
- NAG Foundation Blockset 1.0.3 (for Sun4, Sol2, Alpha, SGI, and SGI64)
- Neural Network Toolbox 3.0*
- Optimization Toolbox 1.5.2
- Partial Differential Equation Toolbox 1.0.3
- QFT Control Design Toolbox 1.0.3
- Robust Control Toolbox 2.0.5
- Signal Processing Toolbox 4.1*
- Spline Toolbox 2.0*
- Statistics Toolbox 2.1.1

- System Identification Toolbox 4.0.4
- Wavelet Toolbox 1.1

The Power System Blockset is a new blockset introduced with MATLAB 5.2.

Power System Blockset 1.0

The Power System Blockset 1.0 is a modern design tool that allows scientists and engineers to build rapidly and easily models that simulate power systems. The blockset uses the Simulink environment, allowing a model to be built using simple *click-and-drag* procedures. Not only can you draw the circuit topology rapidly, but the analysis of the circuit can include its interactions with mechanical, thermal, control, and other disciplines. This is possible because the electrical portions of the simulation interact with Simulink's extensive modeling library. Since Simulink uses MATLAB as the computational engine, MATLAB's toolboxes can also be used by the designer.

Power System Blockset libraries contain models of typical power equipment such as transformers, lines, machines, and power electronics. Their validity is based on the experience of the Power Systems Testing Laboratory of Hydro-Quebec, a large North American utility located in Canada.

See the *Power System Blockset User's Guide* for information about using this blockset.

Communications Toolbox 1.3

Note: Much of the new functionality of the Communications Toolbox 1.3 requires Simulink 2.2. However, even if you use the Communications Toolbox without Simulink, upgrading to Version 1.3 will let you take advantage of a number of other software quality improvements in the toolbox.

The Communications Toolbox 1.3 includes 22 new Simulink function blocks and 12 new example block diagrams.

The new function blocks are:

- Passband digital modulation/demodulation blocks
- Interleave and scrambler blocks

These new blocks expand the functionality of the Communications Toolbox so that it now provides:

- Five new phase-shift keying modulation/demodulation methods
- Three new phase-shift keying mapping/demapping techniques
- Differential encoding/decoding
- Block interleaving and deinterleaving
- Scrambling/descrambling
- Pseudorandom sequence generation

The Communications Toolbox 1.3 also builds on recent MATLAB and Simulink enhancements. These minor changes to the Communications Toolbox are primarily in the area of graphical scopes such as the Error Rate Meter, Eye-Pattern and Scatter plots, and the Trellis plot in the Convolutional Decode block.

This release of the Communications Toolbox also includes changes made to ensure integration with Version 2.2 of the Real-Time Workshop (RTW). If you are using RTW with the Communications Toolbox 1.3, you need Version 2.2 of RTW. Specifically, a few parameter definitions in the Communications Toolbox have been changed for use with C-coded S-functions in RTW.

See the *Communication Toolbox 1.3 New Features Guide*, available in printed form and online (PDF) for more details on these new features.

Control System Toolbox 4.1

The Control System Toolbox 4.1 contains two main enhancements:

- The Root Locus Design GUI (graphical user interface)
- The Simulink LTI Viewer

The Root Locus Design GUI is an interactive design tool that you can use to

- Implement root locus methods on single input-single output (SISO) LTI models defined using `zpk`, `tf`, or `ss`
- Specify the parameters of a feedback compensator: poles, zeros, and gain
- Examine how changing the compensator parameters affects the root locus, as well as various closed-loop system responses (step response, Bode plot, Nyquist plot, or Nichols chart)

The Root Locus Design GUI is documented in Chapter 6 of the *Control System User's Guide*.

The Simulink LTI Viewer is similar to the Control Systems Toolbox LTI Viewer. The Simulink LTI Viewer is used to analyze portions of a Simulink model. Its features include:

- Drag-and-drop blocks that identify the location for the inputs and outputs of the portion of a continuous-time Simulink model you want to analyze
- The ability to specify the operating conditions about which the Simulink model is linearized
- Access to all time and frequency response tools featured in the regular Control System Toolbox LTI Viewer
- The ability to compare a set of linearized models obtained from the same Simulink diagram by varying either the operating conditions or some model parameter values

The Simulink LTI Viewer is documented in Chapter 4 of the *Control System User's Guide*.

Two additional enhancements are:

- Sharper Root Locus plots
- An Export option for the LTI Viewer

DSP Blockset 2.2

DSP Blockset 2.2 introduces a number of new features and improvements. There are over 30 new and enhanced blocks, a filter design wizard, support for data frames, and expanded support of vector and matrix inputs. This section outlines the new additions, and provides pointers to the complete feature descriptions in the *DSP Blockset User's Guide*. See Chapter 1 of the online *User's Guide* for an overview of the blockset's contents.

Also see the DSP Blockset readme file for a summary of the new additions. To view the readme file, type

```
info dspblks
```

at the MATLAB command line.

Note: Version 2.2 of the DSP Blockset is the first release of this product since Version 2.0 (there was no Version 2.1). This adjustment in version numbering allows the DSP Blockset to begin sharing the same version number as the associated release of Simulink. Therefore, beginning with this release, the DSP Blockset requires the equivalently numbered version of Simulink (e.g., DSP Blockset 2.2 requires Simulink 2.2).

Data Frames

The DSP Blockset now offers support for data *frames*, vectors whose elements represent consecutive time samples from a single signal. Framed data is a common format in real-time systems, where the data acquisition hardware often operates most efficiently by accumulating a large number of signal samples at a high rate, and then propagating these samples to the real-time system as a block, or frame, of data. Data frames can also be constructed through the usual DSP Blockset buffering operations (using the Buffer and Complex Buffer blocks, for example).

Version 2.2 includes two new blocks designed to operate specifically on framed data. They are frame-oriented counterparts to the FIR Rate Conversion and Multichannel IIR Filter blocks, and are distinguished by the word “Frame” in the block name:

- FIR Rate Conversion (Frame)
- Multichannel IIR Filter (Frame)

Use these blocks to directly filter or resample framed data in its native format without the computational expense of unbuffering. Other blocks that operate on framed data include the FFT, DCT, and cepstrum blocks in the Transforms library.

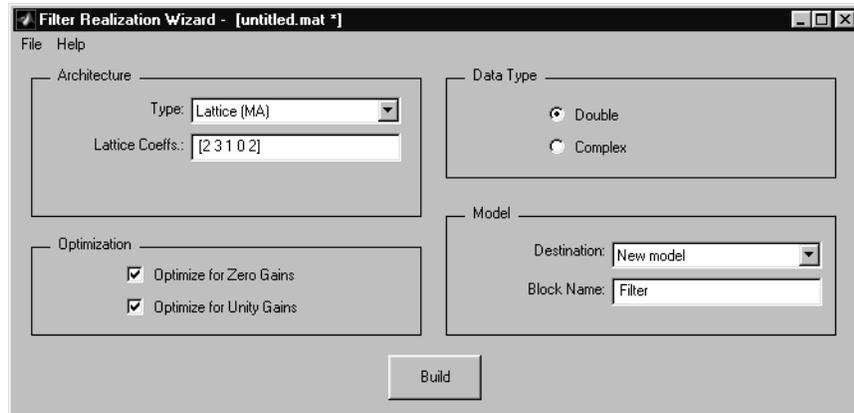
In addition to these frame-based blocks, the data frame format is accepted by all blocks in the blockset that accept vector inputs. Be aware, however, that many blocks implicitly expect the elements of vector inputs to represent *independent channels*, and not consecutive samples. Besides the FIR Rate Conversion and Multichannel IIR Filter blocks, others that expect *non-frame* data include the “running” blocks in the Statistics library, the variable delay

blocks, and the filter design blocks. In general, if a block uses past inputs in generating the current output (and is not specifically designated as a frame-based block), then it considers the elements of a vector (or matrix) input to represent distinct channels, and *not* a frame of consecutive samples.

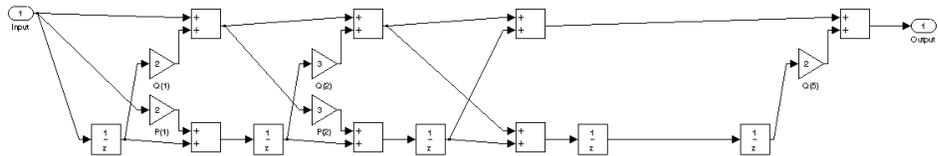
See “Working with Frames” in Chapter 3 of the *User’s Guide* for a complete discussion of this data format.

Filter Realization Wizard

Another new element of the blockset is the Filter Realization Wizard, a graphical user interface (GUI) that allows you to easily construct filters with a variety of different architectures. The GUI is shown below:



When you click the GUI’s **Build** button with the particular settings shown above, the wizard constructs the specified moving average (MA) lattice architecture as a subsystem within a new model window:



You can then alter or optimize the filter to suit your own needs. Additional information about the Filter Realization Wizard can be found in the online Reference.

New and Enhanced Blocks

The table below lists the *new* blocks in Version 2.2. Among the most significant additions are variable delay blocks, discrete cosine transform and cepstrum blocks, linear prediction blocks (LPC, Levinson-Durbin), and new spectral estimation blocks.

Block Library	Block Name	Purpose
DSP Sources	Complex Diagonal Matrix	Generate a square, constant-diagonal complex matrix
DSP Sinks	Triggered Complex Matrix To Workspace	Send a time-sequence of complex matrices to the MATLAB workspace
	Triggered Complex To Workspace	Write the time-sequence of a complex input to the MATLAB workspace
	Triggered Matrix To Workspace	Send a time-sequence of matrices to the MATLAB workspace
	Triggered To Workspace	Write the time-sequence of an input to the MATLAB workspace
Signal Operations	Complex Delay	Delay a complex input by an integer number of sample periods
	Complex Levinson-Durbin	Apply Levinson-Durbin recursion to design an IIR filter with a prescribed autocorrelation sequence
	Complex LPC	Determine the coefficients of an FIR filter that predicts the next sequence value from past and present inputs
	Levinson-Durbin	Apply Levinson-Durbin recursion to design an IIR filter with a prescribed autocorrelation sequence
	LPC	Determine the coefficients of an FIR filter that predicts the next sequence value from past and present inputs
	Variable Fractional Delay	Delay an input by a fractional number of sample periods
	Variable Integer Delay	Delay an input by an integer number of sample periods
Transforms	Complex Cepstrum	Compute the complex cepstrum of an input
	DCT	Compute the DCT of a complex vector input
	IDCT	Compute the complex-valued IDCT of a complex input
	Real Cepstrum	Compute the real cepstrum of an input
	Real DCT	Compute the DCT of a real vector input
	Real IDCT	Compute the IDCT of a real input

Block Library	Block Name	Purpose
Buffers	Shift Register	Convert a scalar time-series into a vector time-series with the same sample period (serial-to-parallel conversion)
	Triggered Shift Register	Convert a scalar time-series into a vector time-series with the same sample period (serial-to-parallel conversion)
Switches and Counters	N-Sample Enable w/Reset	Output 1s for a specified number of sample times
	Sample and Hold	Sample and hold an input signal
Vector Math	Autocorrelation	Compute the autocorrelation of a real vector
	Complex Autocorrelation	Compute the autocorrelation of a complex vector
Complex	Complex Gain	Multiply an input by a complex constant
	Real to Complex	Construct a complex output from a real input
Statistics	Histogram	Compute the histogram (frequency distribution) of values in a vector input
	Median	Find the median value of a vector input
	Running Histogram	Track frequency distribution of values in a vector input over time
	Sort	Sort the elements in a vector by value
Filter Realizations	Filter Realization Wizard	Build an IIR or FIR filter with a particular architecture
	Multichannel IIR Filter (Frame)	Apply an IIR filter to a multichannel input signal
	Time Varying FIR Filter	Apply a variable FIR filter to a multichannel input signal
	Time Varying IIR Filter	Apply a variable IIR filter to a multichannel input signal
Multirate Filters	FIR Rate Conversion (Frame)	Upsample, filter, and downsample a real input
Spectrum Analysis	Burg Method	Compute a parametric estimate of the spectrum using the Burg method
	Yule-Walker AR	Compute a parametric estimate of the spectrum using the Yule-Walker AR method

In addition to the new blocks, several blocks have been enhanced for Version 2.2, and are highlighted in the table below. The most important area of growth

among the existing blocks is in the expanded support of vector and matrix inputs for buffering and unbuffering operations.

Block Library	Block Name	Enhancement
DSP Sources	Diagonal Matrix	Allows specification of a non-constant diagonal
DSP Sinks	Frequency Vector Scope	Offers new menus, and window position memory
	Time Vector Scope	Offers new menus, and window position memory
Signal Operations	Complex Zero Pad	Offers the option of truncating the input to the specified output vector length
	Delay	Accepts an initial condition
	Zero Pad	Offers the option of truncating the input to the specified output vector length
Buffers	Buffer	Supports vector inputs, and accepts an initial condition
	Complex Buffer	Supports vector inputs, and accepts an initial condition
	Complex Partial Unbuffer	Supports matrix inputs
	Complex Unbuffer	Supports matrix inputs
	Partial Unbuffer	Supports matrix inputs
	Unbuffer	Supports matrix inputs
Switches and Counters	Commutator	Supports matrix inputs
	Distributor	Supports vector inputs, and accepts an initial condition
Multirate Filters	FIR Rate Conversion	Supports matrix inputs

For Users Upgrading from Version 1.0a

The DSP Blockset 2.2 is completely compatible with Version 1.0a, but there are some limitations on mixing buffer blocks from the two versions, and you will need to recompile any custom blocks that use C-MEX S-functions so that they work with Simulink 2.2.

See “DSP Blockset: Upgrading from Version 1.0a” in Chapter 4 for more details about upgrading from Version 1.0a.

Financial Toolbox 1.1

The Financial Toolbox 1.1 supports detailed term structure analysis. In addition, this version provides new date functions, coupon date functions, portfolio allocation tools, and a new derivative pricing function. These new functions are summarized below.

For information about these functions, refer to the *Financial Toolbox User's Guide* for Version 1.1.

Term Structure Functions

Function	Description
di sc2zero	Zero rate curve from a discount curve
fwd2zero	Forward rate curve from a zero curve
pyl d2zero	Par yield curve from a zero curve
tbl 2bond	Conversion of TBills to TBond market convention
termfi t	Demo function for smoothing rates with splines
tr2bonds	Conversion of Treasury data to bond input format
zbt pri ce	Bootstrap a zero curve from market bond prices
zbt yi el d	Bootstrap a zero curve from market bond yields
zero2di sc	Discount factors from a zero curve
zero2fwd	Zero curve from a forward curve
zero2pyl d	Zero curve from a par curve

Derivatives Function

Function	Description
bl kpri ce	Black's pricing model

Portfolio Analysis Function

Function	Description
ewcov	Asset covariance estimation with exponential weighting

Date Functions

Function	Description
accrfrac	Accrued interest coupon period fraction
busdate	Next or previous business day
cfdates	Cash flow dates of a security
datefind	Indices of date numbers in a matrix
eomdate	Last date of month
fbusdate	First business date of month
holidays	Holidays and nontrading days
ibusday	True for dates that are business days
lbusday	Last business date of the month
lweekdate	Date of last occurrence of weekday in month
m2xdate	MATLAB serial date number to Excel date number
months	Number of whole months between dates
nweekdate	Date of specific occurrence of weekday in month
yeardays	Number of days in year
x2mdate	Excel serial date number to MATLAB date number

Demo of an Excel Link Portfolio Optimizer Tool

The following files provide a demo of an Excel Link portfolio optimizer tool:

- excel portopt. m
- excel portopt. xls

Fuzzy Logic Toolbox 2.0

The Fuzzy Logic Toolbox 2.0 features several improvements, including

- Additional and enhanced graphical user interfaces (GUIs) for performing a number of tasks
- Several enhanced Fuzzy Logic algorithms
- Fuzzy Inference System (FIS) are represented as MATLAB structures
- More dimensions are allowed in user-defined membership functions

Graphical User Interface Enhancements

Fuzzy Logic Toolbox 2.0 adds or enhances several graphical user interfaces (GUIs):

- GUI for Adaptive Neuro-Fuzzy Inference System (ANFIS) learning.
With this GUI, you can implement an ANFIS and use automatic membership function adaptation without resorting to the command line. The learning process can also be viewed graphically and in real time, so any necessary adjustment can be made efficiently. The ANFIS Editor is also fully integrated with the other GUI tools: the Fuzzy System Editor, Membership Function Editor, Rule Editor, Rule Viewer and Surface Viewer. This GUI is described in Chapter 2 of the *Fuzzy Logic Toolbox User's Guide*.
- Membership Function Editor.
You can click and drag both the shape and the location of your membership functions.
- Rule Editor.
You can point and click to build your rules easily, rather than typing in long rules.
- GUI for fuzzy clustering.
This GUI lets you view both fuzzy c-means clustering and subtractive clustering while they are in progress.

- Rule Viewer for the fuzzy Simulink block.

When a fuzzy inference system is used in Simulink, the Rule Viewer lets you see when each rule is triggered and how each membership function is applied during a simulation.

Fuzzy Algorithm Improvements

The following Fuzzy Logic algorithms have been added or enhanced:

- Backpropagation learning algorithm for ANFIS.

Backpropagation is now available as an ANFIS learning algorithm.

- Constant output membership functions for ANFIS.

You can now use constant output membership functions with ANFIS in addition to linear output membership functions.

- Fuzzy arithmetic.

Basic fuzzy arithmetic functions are now provided for addition, subtraction, multiplication, and division operations among different membership functions.

- Customizable membership function discretization.

Now you can adjust the sampling rate used to discretize the output membership functions of your rules. This gives you control of the accuracy and efficiency of the defuzzification calculations.

FIS Represented As MATLAB Structures

The Fuzzy Inference System (FIS) is now represented as a MATLAB structure. A structure (instead of a flat matrix) is now the basic element in constructing a fuzzy logic system. This fundamental change in the way of representing the fuzzy logic system makes many details of working with the constructed system easier.

A Fuzzy Inference System that you created with a pre-2.0 version of the Fuzzy Logic Toolbox is still usable in 2.0, if you run the `convertfis` function on it. The `convertfis` function automatically converts pre-2.0 Fuzzy Inference Systems to work with Version 2.0.

More Dimensions Allowed for User-Defined Membership Functions

You can now use up to 16 parameters when you define your own customized membership functions.

Image Processing Toolbox 2.1

Interactive Pixel Value Display

The new function `pixval` installs in a figure an interactive display of the data values for whatever image pixel the cursor is currently over. You can also click and drag to display the Euclidean distance between two pixels.

Feature Measurement

The new function `imfeature` computes feature measurements, such as the center of mass and the bounding box, for regions in an image.

Inverse Radon Transform

The new function `iradon` uses the inverse Radon transform to reconstruct images from projection data. In addition, the toolbox has a new function, `phantom`, that generates test images for use with the Radon and inverse Radon transforms.

Canny Edge Detector

The `edge` function now supports the Canny edge detection method. This method is better at detecting weak edges and is less sensitive to noise than the other supported edge-detection methods.

Other Enhancements

- The `bwfill` function can now automatically detect and fill holes in objects.
- The toolbox now supports the YCbCr color space with two new functions, `ycbcr2rgb` and `rgb2ycbcr`.
- You can now easily convert images between double precision and `uint8` using two new functions, `im2double` and `im2uint8`.
- You can control whether `imshow` automatically calls `truesize` by setting the new toolbox preference ' `ImshowTruesize` '.

Neural Network Toolbox 3.0

The Neural Network Toolbox 3.0 provides several important new features, including:

- **Modular network representation.**
All network properties are collected in a single “network object.” Networks can have any number of sets of inputs and layers, any input or layer can be connected to any layer with a weight, and any weight can have a tapped delay.
- **Reduced memory Levenberg-Marquardt (LM) algorithm.**
The fast LM algorithm (by a factor of 10 to 100 over other methods) can be used in much larger problems than in Version 2.0.
- **New algorithms, including**
 - Conjugate gradient
 - R-Prop
 - Two quasi-newton methods
- **New network types, including**
 - Probabilistic
 - Generalized Regression
- **Automatic regularization and new training options, including**
 - Training with on variations of mean square error for better generalization
 - Training against a validation set
 - Training until the gradient of the error reaches a minimum
- **Pre- and post-processing functions, such as Principal Component Analysis.**
- **Better Simulink support: the Neural Network Toolbox now generates network simulation blocks.**

These features are summarized in more detail in the “What’s New in 3.0” section of the updated *Neural Network Toolbox User’s Guide*.

Signal Processing Toolbox 4.1

The Signal Processing Toolbox 4.1 introduces a number of improvements, including a new GUI for the Filter Designer. This section outlines the new additions and provides pointers to the complete feature descriptions in the online (PDF) *Signal Processing Toolbox User's Guide*. The Signal Processing Toolbox readme file also contains a short summary of this information.

To view the readme file, type at the MATLAB command line

```
info signal
```

Spectral Estimation

The *MEM* spectral estimation method (previously implemented by the `pmem` function) has been more accurately renamed the *Yule-Walker AR* method, and is now implemented by the `pyulear` function. The `pmem` function continues to work, but generates the following warning message:

```
Warning: pmem is obsolete and will be discontinued.  
Use pyulear instead.
```

In addition to this name change, the *Burg* method of spectral estimation has been added to the toolbox via the `pburg` function.

SPTool Graphical User Interface

Several areas of the SPTool interactive signal processing environment have been enhanced for Version 4.1. See Chapter 5 in the PDF version of the *User's Guide* for complete instructions on using the new features.

The Filter Designer interface has been revised for improved usability. A signal's spectrum can now be superimposed on any filter response, and a new **Measurements** panel displays the filter's characteristics as it is being designed.

Viewing (zoom) controls

General controls

Specifications panel for setting filter parameters

Apply the specifications, or revert to the previous specifications

Filter magnitude response display area

Overlay a signal's spectrum on the filter response

Measurements panel for viewing filter characteristics

Filter Designer

File Window

Filter: filt2

Zoom In-Y Zoom Out-Y Zoom In-X Zoom Out-X Pass Band Full View Minimize Zoom Help

Algorithm: Equipple FIR Sampling Frequency: 5000 Overlay Spectrum...

Specifications

Minimum Order

Type: bandpass

Passband:

Fp1: 750

Fp2: 1250

Rp: 0.01

Stopband:

Fs1: 500

Fs2: 1500

Rs: 75

Revert Apply

Frequency Response

Magnitude (dB)

Frequency

Measurements

Order: 78

Passband:

Actual Rp: 0.01361

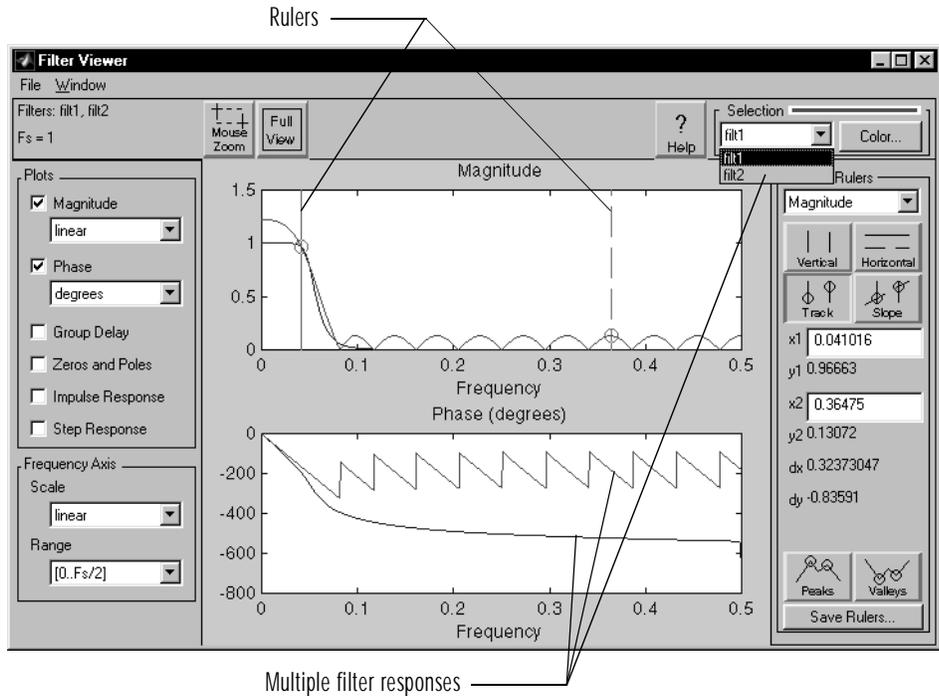
Weight: 1

Stopband:

Actual Rs: 72.31

Weight: 3.237

The Filter Viewer is now capable of displaying multiple filter responses simultaneously, and also benefits from new rulers that can be used for fine measurement on all of the plot types.



The Spectrum Viewer offers two new spectral estimation methods, the fundamental *FFT* method, and the *Burg* method. Additionally, the *MEM* method has been renamed the *Yule-Walker AR* method. The **MEM** option has been retained in the **Method** pop-up menu for backwards compatibility, but will be removed in a future release. Please use **Yule AR** instead.



General Enhancements

The following enhancements and bug-fixes are also included in the 4.1 release.

- The generalized cosine window functions (`hamming`, `hann`, and `blackman`) can now generate both periodic and symmetric windows. Formerly they only generated symmetric windows.
- A bug in `cremez` that produced a complex filter instead of the appropriate real filter has been fixed. Additionally, the `opt.fgrid` and `opt.fextr` outputs are now normalized to the Nyquist frequency.
- The `invfreqz` and `invfreqs` functions now work for complex as well as real filters.

Spline Toolbox 2.0

Multivariate Spline Support

All M-files for the construction of splines (in B-form or `ppform`) have been expanded to handle tensor-product splines in any number of variables. The same is true for most of the M-files that make use of splines. This means that it is now possible to interpolate, approximate, or smooth gridded data in any number of variables and then evaluate, plot, differentiate, or integrate the resulting multivariate spline.

User Interface Enhancements

In the same spirit of keeping the number of commands small (and of object-oriented programming), most of the form-specific commands (such as `spval` or `ppbrk`) have been replaced by generic commands (such as `fnval` or `fnbrk`). The forms themselves are now structures, but that should be irrelevant to the casual user.

Vector-Valued Spline Enhancements

Since splines in the toolbox can be vector-valued, it is now possible to handle certain surfaces as 3-vector-valued bivariate tensor-product splines.

Additional Enhancements

Further new features include:

- Use of sparse matrices wherever appropriate
- Construction, in `spaps`, of the smoothing spline (linear, cubic, or quintic) that fits given data within a given tolerance
- Conversion, in `fnrfn`, of a given form to one on a refined knot or break sequence
- More flexibility in `fncomb` for arithmetic with functions
- More freedom and ease with the input arguments to various M-files
- Optional use of weights in the construction of least-squares and of smoothing splines
- New M-files for helping with the conversion between forms and between breaks and knots and their multiplicities

MATLAB 5.1 Enhancements

What's New in MATLAB 5.1?	2-2
Language and Development Environment Enhancements	2-4
Handle Graphics Enhancements	2-8
TIFF and JPEG Device Drivers	2-9
TIFF Preview Images for Encapsulated PostScript . . .	2-12
API Enhancements for Windows NT	2-13
API Enhancements for Macintosh	2-14
Stateflow	2-16
Mapping Toolbox	2-17

What's New in MATLAB 5.1?

Note: All the features introduced in MATLAB 5.1 are also in MATLAB 5.2.

The main purpose of the MATLAB 5.1 release was to complete upgrades to the 5.0 level of the entire set of toolboxes and blocksets, and to introduce the Stateflow product. However, there were also a number of small but useful enhancements to MATLAB provided with this release.

Enhancements to MATLAB

MATLAB 5.1 added several enhancements to the MATLAB language and development environment, Handle Graphics[®], printing, and the Application Program Interface (API).

The language and development environment enhancements included:

- `find` returns an empty matrix
- Multibyte character support
- Several PC enhancements:
 - Removal of Microsoft Windows TCP/IP requirement
 - Notebook support for Office 97
 - PC Editor/Debugger icons changed
- Several Macintosh enhancements:
 - Editing arrays on the Macintosh
 - Enhancements to the Macintosh Path Browser
 - Enhancements to the Macintosh Debugger

The Handle Graphics and printing enhancements included:

- Scatter plot functions
- X-Windows support for `ui setcolor` (UNIX)
- Patch and surface printing enhancements
- TIFF and JPEG device drivers
- TIFF preview images for Encapsulated Postscript

The Application Program Interface enhancements included:

- Setting up the compiler location for Windows NT
- Support for CodeWarrior 11 for the Macintosh

MATLAB 5.1 also fixed bugs from earlier releases either reported by customers or found through additional internal testing.

Upgrades to Simulink, Real-Time Workshop, Toolboxes, and Blocksets

MATLAB 5.1 completed the upgrades to the entire set of toolboxes. The Fixed-Point Blockset 1.0.2 was introduced with MATLAB 5.1.

New releases of the following products were also produced with MATLAB 5.1; however, these products have been upgraded once again for MATLAB 5.2 (the MATLAB 5.2 version number is shown, since that is the version that is shipped with MATLAB 5.2):

- Simulink 2.2
- Real-Time Workshop 2.2
- DSP Blockset 2.2
- Image Processing Toolbox 2.1
- Symbolic Math Toolboxes 2.0.1
- Communications Toolbox 1.2

New Products

In addition to these toolboxes, two new products were introduced with MATLAB 5.1:

- Stateflow 1.0
- Mapping Toolbox 1.0

Stateflow and the Mapping Toolbox are described in more detail starting on page 2-16.

Language and Development Environment Enhancements

find Returns Empty Matrix

The `find` function returns an empty matrix if nothing is found. Previously it returned `[0, 1]`.

Multibyte Character Support

On the PC and Macintosh, MATLAB 5.1 added support for multibyte characters (including Kanji) for data.

This feature allows you to use multibyte characters in MATLAB strings. You can also use multibyte characters in Handle Graphics property values and Simulink blocks.

Note that you *cannot* use multibyte characters in variable, file, or function names. Also, multibyte text may not be machine independent.



Removal of Microsoft Windows TCP/IP Issues

MATLAB 5.0 for Microsoft Windows 95 required the use of TCP/IP networking software even for non-networked installations. For MATLAB 5.1 this requirement was removed. The portions of the MATLAB user interface that depended upon TCP/IP were recoded to use ActiveX.



Notebook Support for Office 97

MATLAB 5.1 provided Notebook support for Microsoft Office 97.

Note: The MATLAB 5.1 Notebook worked with Windows NT with Microsoft Office 97. However, for Windows 95, due to an Office 97 problem, you may experience problems printing a Notebook document that includes an imported graphic. See “OFF97: Imported EMF Files Are Not Printed Correctly” in the online Microsoft Knowledge Base for details.



PC Editor/Debugger

For MATLAB 5.1 the PC Editor/Debugger provided new debugging icons on the toolbar. The debugging operations are the same as for MATLAB 5.0. The new debugging icons on the toolbar were:

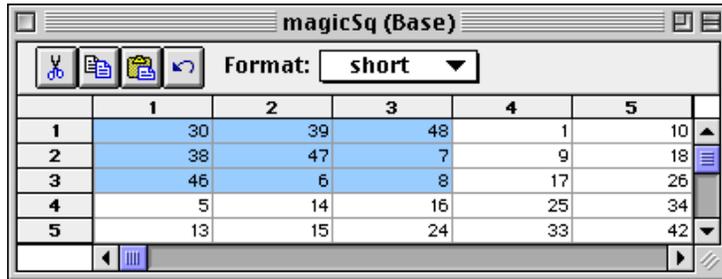
Toolbar Button	Description	Equivalent Command
	Set/Clear Breakpoint: set or clear a breakpoint at the line containing the cursor.	dbstop/ dbclear
	Clear All Breakpoints: clear all breakpoints that are currently set.	dbclear all
	Step In: execute the current line of the M-file and if the line is a call to another function, step into that function.	dbstep in
	Single Step: execute the current line of the M-file.	dbstep
	Continue: continue execution of M-file until completion or until another breakpoint is encountered.	dbcont
	Quit Debugging: exit the debugging state.	dbquit



Editing Arrays on Macintosh

MATLAB 5.1 provided a new capability to allow you to view and edit two-dimensional real and complex double arrays, row vector character arrays, and row or column vector cell arrays of strings. If the Array Editor does not support a variable type, double-click on the variable in the Workspace Browser to display the variable in the Command Window. This tool can be especially useful when debugging M-functions using the M-file Debugger.

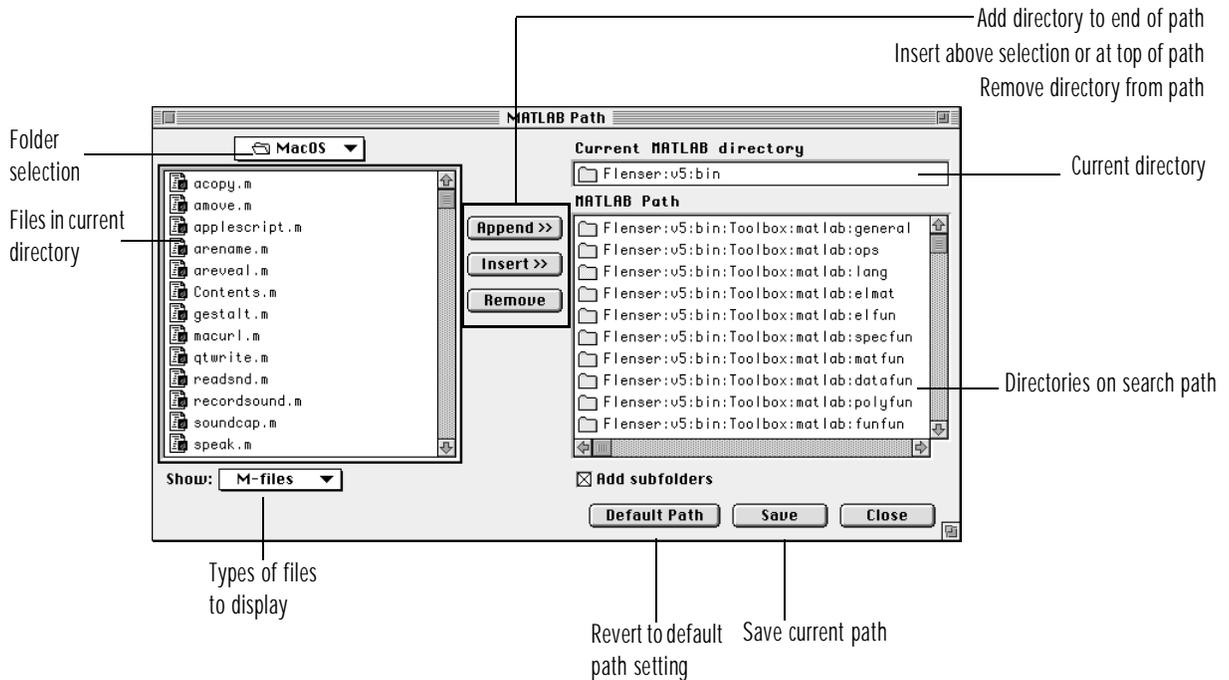
An example of the Array Editor is shown below.



For numeric arrays, the pull-down menu allows you to change the format of the output display as if you were using the format command.



The Path Browser on the Macintosh



In MATLAB 5.1, if you change the path from the Command Window, the Path Browser's contents update automatically. The **Refresh** button was removed from the Path Browser.

If you check the **Add subfolders** box, all subfolders of the selected directory will be added to the path as well.



Macintosh Debugger

The MATLAB 5.1 Debugger provided a new operation, **Step Out**, which was represented by a new icon on the debugger toolbar. The full set of debugging icons is:

Toolbar Button	Description	Equivalent Command
	Single Step: execute the current line of the M-file.	dbstep
	Step In: execute the current line of the M-file and if the line is a call to another function, step into that function.	dbstep in
	Step Out: continue execution until a return to the calling function is encountered.	
	Continue: continue execution of M-file until completion or until another breakpoint is encountered.	dbcont
	Quit Debugging: exit the debugging state.	dbquit

The 5.1 Debugger also introduced the **Go Until Here** menu item on the **Debug** menu. If you place the cursor at a specific line and choose the menu item, execution continues until that line is reached. Option-click on the breakpoint marker to the left of a line to execute a shortcut for this operation.

Handle Graphics Enhancements

MATLAB 5.1 provided some new Handle Graphics functions.

Scatter Plot Functions Added

MATLAB added two new functions, `scatter` and `scatter3`, which enable you to create two-dimensional and three-dimensional scatter plots. Each function allows you to specify the style, size, and color of the marks used to create the scatter diagrams.

See the online *MATLAB Function Reference* for more information about these functions.

X-Windows Support for `uisetcolor`

The `ui setcol` or function is supported on X-Windows systems (UNIX).

Previously Undocumented Functions

These two functions existed in MATLAB 5.0, with command line help, but were not documented in the MATLAB 5.0 online *Function Reference*; they are documented in the MATLAB 5.2 online *Function Reference*:

- `pagedlg` – Dialog box to set page layout properties for printing Figures.
- `printdlg` – Dialog box to manage printing of Figures.

Printing Patches and Surfaces

MATLAB 5.1 added support for printing texture-mapped patches and surfaces (this did not work in Version 5.0).

Also, printing interpolated patches and surfaces is more efficient than in Version 5.0.

TIFF and JPEG Device Drivers

MATLAB 5.1 added new built-in device drivers for producing TIFF (Tagged Image File Format) and JPEG (Joint Photographic Experts Group) graphics files from MATLAB figures. These drivers are available on all platforms.

This table summarizes the command-line switches for these drivers:

Device	Description
<code>-dtiff</code>	TIFF with packbit compression
<code>-dtiffnocompression</code>	TIFF with no compression
<code>-djpeg</code>	Baseline JPEG, quality level 75
<code>-djpegnumber</code>	Baseline JPEG, quality level specified by <i>number</i>

Note: These drivers work with MATLAB figures only. You cannot use these drivers to print Simulink models.

This section summarizes how to use these drivers with the `print` command.

TIFF

To produce a TIFF file from a MATLAB figure, use the `-dtiff` switch. For example, this command produces a TIFF file named `newplot.tif` from the current figure:

```
print -dtiff newplot.tif
```

You can use the `-r` option in conjunction with the `-dtiff` switch to specify the resolution of the output. For example:

```
print -dtiff -r100 newplot.tif
```

If you do not specify the resolution, MATLAB uses the default resolution of 150 dots per inch.

Note that you must specify a filename because TIFF files cannot be sent directly to a printer. If you omit the filename, MATLAB assigns the file a name, such as `figure1.tif`. If you specify a filename that does not include the `.tif` extension, MATLAB appends the extension automatically.

The TIFF files that MATLAB produces are 24-bit truecolor bitmaps. MATLAB renders these graphics using the Z-buffer renderer, regardless of the setting of the figure `Renderer` property. If you use the `-printers` switch with the `print` command, the switch is ignored.

Compression

The TIFF output produced by `-dtiff` uses packbit compression, a lossless compression scheme that is supported by virtually all applications that can import TIFF graphics. If you need to import a TIFF file into an application that does not read packbit-compressed TIFF, use the `-dtiffnocompression` switch to produce an uncompressed TIFF file. (You can abbreviate this switch to `-dtiffn`.) For example:

```
print -dtiffn -r100 newplot.tif
```

An uncompressed TIFF file is often much larger than the same file compressed. For certain plots, the uncompressed file may be more than 10 times the size of the compressed file. (The actual ratio will vary. The size of an uncompressed file depends only on the resolution and the width and height values in the `PaperPosition` figure property; the size of the compressed file also depends on the content of the figure.)

JPEG

To produce a JPEG file from a MATLAB figure, use the `-djpeg` switch. For example, this command produces a JPEG file named `newplot.jpg` from the current figure:

```
print -djpeg newplot.jpg
```

You can use the `-r` option in conjunction with the `-djpeg` switch to specify the resolution of the output. For example:

```
print -djpeg -r100 newplot.jpg
```

If you do not specify the resolution, MATLAB uses the default resolution of 150 dots per inch.

Note that you must specify a filename because JPEG files cannot be sent directly to a printer. If you omit the filename, MATLAB assigns the file a name such as `figure1.jpg`. If you specify a filename that does not include the `.jpg` extension, MATLAB appends the extension automatically.

The JPEG files that MATLAB produces are 24-bit truecolor bitmaps. MATLAB renders these graphics using the Z-buffer renderer, regardless of the setting of the figure `Renderer` property. If you use the `-painters` switch with the `print` command, the switch is ignored.

Compression

JPEG files use a lossy compression scheme that compresses files dramatically with relatively little loss of information. This scheme enables you to make tradeoffs between file size and quality, by specifying a quality level between 0 (minimum quality, maximum compression) and 100 (maximum quality, minimum compression). By default, `-djpeg` uses a quality level of 75; however, you can use a different level by appending the value to the device name. For example, this command produces a JPEG file with a quality level of 50:

```
print -djpeg50 -r100 newplot.jpg
```

Even at the highest quality level, JPEG files are often highly compressed. In fact, depending on the figure, a JPEG file with a quality level of 100 may be considerably smaller than a packbit-compressed TIFF file of the same figure.

TIFF Preview Images for Encapsulated PostScript

MATLAB 5.1 introduced support for TIFF preview images for Encapsulated PostScript (EPS) files. To produce a TIFF preview, use the `-tiff` switch. For example, this command creates an EPS file called `newplot.eps` that contains a TIFF preview:

```
print -deps -tiff newplot.eps
```

The preview image has a resolution of 72 dots per inch, and the colors in the preview match the colors in the EPS file. MATLAB creates the EPS with a loose bounding box (i.e., white space around the figure), so that the size and position of the preview image match the EPS. There may be some differences between the EPS and the TIFF preview because the preview is always rendered using Z-buffer, while the EPS may be rendered with painter's algorithm.

The `-tiff` switch works on all platforms; you can view the resulting preview image within any application that can display TIFF graphics.

Alternate Method on the Macintosh

On Macintosh systems, you can also specify a TIFF preview image by selecting **Save As** from the **File** menu of the figure window. In the dialog box, use the **File Type** pop-up menu to select an EPS format, and then select **TIFF** from the **Preview** pop-up menu.

Also on Macintosh systems, if you use the `print` command to create an EPS file and you do not use the `-tiff` switch, the file is created with a PICT preview. On other platforms, no preview is created.

API Enhancements for Windows NT

Setting Up the Compiler Location

MATLAB 5.1 provided a new switch for the `mex` script. The switch, `setup`, allows you to configure the default options file, `mexopts.bat`, for your system C compiler. This eliminates the need to reinstall MATLAB if you change compilers for your environment.

You can run the `setup` option from either the MATLAB or DOS command prompt, and it can be called anytime to configure the options file.

Executing the `setup` option presents a list of compilers whose options files are currently shipped in the `bin` subdirectory of MATLAB. This example shows how to select the Microsoft Visual C++ compiler:

```
C:\mex -setup
C compilers
[1] Microsoft Visual C++
[2] Borland C/C++
[3] Watcom C/C++

Fortran compilers
[4] Microsoft Powerstation

[0] None

compiler: 1
```

If the selected compiler has more than one options file (due to more than one version of the compiler), you are asked for a specific version. For example,

```
Which version
[1] 4.x
[2] 5.x
version: 1
```

Finally, you are asked to enter the location of your compiler:

```
Please enter the location of your C compiler c:\msdev
```

API Enhancements for Macintosh

Installation Notes for Using CodeWarrior 11 with the MATLAB API

MATLAB 5.1 introduced support for CodeWarrior 10. CodeWarrior 11 is supported with the limitations described below. Previous versions of CodeWarrior are not supported.

In CodeWarrior 11, Metrowerks introduced a new ANSI C library, known as the Metrowerks Standard Library (MSL). Based on difficulties involved in upgrading to MSL, and from similar USENET reports of MSL bugs, MATLAB Application Program Interface (API) support for MSL has been postponed until a future version of CodeWarrior is released.

To use CodeWarrior 11 to generate MEX-files and/or MAT applications, you must install the old CodeWarrior libraries and header files. These files are the same as those shipped with previous versions of CodeWarrior, and can be found on the CodeWarrior 11 CD-ROM in the folder

CW11 Gold Tools: Metrowerks CodeWarrior: (Obsolete ANSI Libraries):

The names of the folders within (Obsolete ANSI Libraries) indicate the CodeWarrior folder to which their contents should be copied.

Using CodeWarrior 10 and 11 with the MATLAB API

The instructions below supersede the instructions for building MAT-file programs when using CodeWarrior 10 and 11 that are given in Chapter 4 of the *MATLAB Application Program Interface Guide*.

Building Applications on the Power Macintosh and 68K Macintosh

To build applications that use the MAT-file API, start with a project made from the stationery MATLAB_API_App. PPC. proj (for Power Macintoshes) or MATLAB_API_App. 68k. proj (for 68K Macintoshes) in the <MATLAB>: extern: src: folder. Then, make these modifications to your project:

- 1 In the **Access Paths** preference panel, add to the User field the folder <MATLAB>: extern: include:

2 Add the source files that call MAT-file API functions to the project.

3 On Power Macintoshes, add to the project the shared libraries `libmx` and `libmat` from the `<MATLAB>extern:lib:PowerMac:`

On 68K Macintoshes, add the static libraries `libmx.lib`, `libmi.lib`, `libbut.lib`, and `libmat.lib` from the `<MATLAB>:extern:lib:68k:Metrowerks: folder` to the project.

You can now build and link your project as normally done in CodeWarrior.

Stateflow

Addition to the Simulink Modeling Environment

Stateflow was introduced with MATLAB 5.1 adding to the Simulink modeling and simulation environment. A graphical tool for designing complex control and supervisory logic systems, Stateflow allows you to model and simulate the behavior of complex reactive, event-based systems based on finite state machine theory. Stateflow lets you add event-driven elements of a system to Simulink continuous or discrete modeling for a single, closed-loop simulation.

Stateflow represents an evolution from finite state machine theory by adding several major improvements, including hierarchy, parallelism, junctions, and history. These changes enable Stateflow to make practical use of finite state machine theory with realistic application to control systems.

A major benefit of Stateflow is its seamless point-and-click interface to Simulink. The control behavior that Stateflow models provides an ideal complement to the algorithmic behavior modeled in Simulink. In Simulink, you develop your model of continuous- and discrete-time dynamic systems using its graphical, block diagram environment. Then you drag and drop the blocks that represent Stateflow diagrams directly into your Simulink model to add event-driven behavior to Simulink simulations.

Applications for Stateflow include developing the control logic in embedded systems for electronic and mechanical systems found in automobiles, aircraft, telecommunications systems, computer peripherals, office automation equipment, and medical instrumentation.

Stateflow works with the latest versions of Simulink and Real-Time Workshop to offer an integrated environment for modeling, simulating, and prototyping real-time embedded systems applications.

Stateflow Coder

Stateflow provides automatic C-code generation through the optional Stateflow Coder. C code generated by Stateflow Coder can be used independently or integrated with code from Real-Time Workshop. Thus, Simulink, Stateflow, and Real-Time Workshop are integrated from the design and modeling phase through the code generation stage. The generated code can be executed for rapid prototyping, hardware-in-the-loop testing, or for stand-alone simulations.

Mapping Toolbox

The Mapping Toolbox was introduced with MATLAB 5.2, providing a toolbox for geographic display and cartographic analysis. The toolbox gives you the ability to plot geographically based information as easily as any other type of data that you can plot in MATLAB. Both vector and matrix map data can be displayed, manipulated, and analyzed. The toolbox manages the projection, clipping, and trimming of the data automatically for you, even if you change the projection.

The Mapping Toolbox provides more than 60 map projections. There are extensive geographic analysis functions, such as computations of distance, tracks, great and small circles, intersections, and navigation functions. These computations can be made for a spherical body, or can make use of spheroidal models of the earth and other planets when more accuracy is required. Utility functions allow you to convert easily between different time, distance, and angle units.

The toolbox provides a number of global map data sets, and allows you to import more detailed data from government sources over the Internet and on CD-ROM. These data sets include the Digital Chart of the World, Tiger/Line files, and Digital Elevation Models of the world and the United States.

In addition to the command-line functions, the toolbox also provides an extensive suite of graphical user interfaces (GUIs) for accessing the toolbox functionality. These GUIs allow you to manage data interactively, plot it, modify the display, make measurements, and generate geographic data like tracks and circles. The GUIs are available as an integrated set and also are available individually.

MATLAB 5.0 Enhancements

MATLAB 5.0 Enhancements	3-2
New Data Constructs	3-5
New and Enhanced Language Functions	3-14
New Data Analysis Features	3-20
New and Enhanced Handle Graphics Features	3-23
New and Enhanced Handle Graphics Object Properties	3-34
Improvements to Graphical User Interfaces (GUIs) . . .	3-42
Enhanced Application Program Interface (API)	3-44
New Platform-Specific Features	3-46

MATLAB 5.0 Enhancements

MATLAB 5.0 featured five major areas of new functionality:

- Enhanced programming and application development tools
- New data types, structures, and language features
- Faster, better graphics and visualization
- More mathematical and data analysis tools
- Major enhancements to the MATLAB application toolbox suite and to Simulink

Enhanced Programming and Application Development Tools

MATLAB 5.0 provided new M-file programming enhancements and application development tools that make it easier than ever to develop and maintain applications in MATLAB. Highlights include:

- Integrated M-file editor
- Visual M-file debugger
- M-file performance profiler
- Search path browser/editor
- Workspace browser
- Web-based online Help Desk/documentation viewer
- GUI builder
- Handle Graphics property editor
- Preparsed P-code files (P-files)
- Enhanced, self-diagnosing Application Program Interface (API)

New Data Types, Structures, and Language Features

MATLAB 5.0 introduced new data types and language improvements. These new features make it easy to build much larger and more complex MATLAB applications.

- Multidimensional arrays
- User-definable data structures
- Cell arrays: multitype data arrays
- Character arrays: two bytes per character
- Single byte data type for images
- Object-oriented programming
- Variable-length argument lists
- Multifunction and private M-files
- Function and operator overloading
- `switch/case` statements

Faster, Better Graphics and Visualization

MATLAB 5.0 added powerful new visualization techniques and significantly faster graphics using the Z-buffer algorithm. Presentation graphics were also improved to give you more options and control over how you present your data.

- Visualization
 - Truecolor (RGB) support
 - Fast and accurate Z-buffer display algorithm
 - Flat, Gouraud, and Phong lighting
 - Vectorized patches for three dimensional modeling
 - Camera view model, perspective
 - Efficient 8-bit image display
 - Image file import/export

- Presentation graphics
 - Greek symbols, sub/superscripts, multiline text
 - Dual axis plots
 - Three-dimensional quiver, ribbon, and stem plots
 - Pie charts, three-dimensional bar charts
 - Extended curve marker symbol family

More Mathematical and Data Analysis Tools

With more than 500 mathematical, statistical, and engineering functions, MATLAB gives you immediate access to the numeric computing tools you need. New features with MATLAB 5.0 included:

- New ordinary differential equation solvers (ODEs)
- Delaunay triangulation
- Gridding for irregularly sampled data
- Set theory functions
- Two-dimensional quadrature
- Time and date handling functions
- Multidimensional interpolation, convolution, and FFT's
- Bit-wise operators
- Iterative sparse methods
- Sparse matrix eigenvalues and singular values

Enhancements to Simulink and Application Toolboxes

Significant upgrades introduced with MATLAB 5.0 are listed below (note that all these products have been upgraded again with MATLAB 5.2):

- Simulink 2.0
- Image Processing Toolbox 2.0
- Control System Toolbox 4.0
- Signal Processing Toolbox 4.0
- Optimization Toolbox 2.0

New Data Constructs

MATLAB 5.0 supports these new data constructs:

- Multidimensional arrays
- Cell arrays
- Structures
- Objects

In addition, MATLAB 5.0 features character arrays that incorporate an improved storage method for string data.

Multidimensional Arrays

Arrays (other than sparse matrices) are no longer restricted to two dimensions. You can create and access arrays with two or more dimensions by:

- Using MATLAB functions like `zeros`, `ones`, or `rand`
- Using the new `cat` function
- Using the `repmat` function
- Indexing an existing array

MATLAB functions like `zeros`, `ones`, and `rand` have been extended to accept more than two dimensions as arguments. To create a 3-by-4-by-5 array of ones, for example, use

```
A = ones(3, 4, 5)
```

The new `cat` function enables you to concatenate arrays along a specified dimension. For example, create two rectangular arrays A and B:

```
A = [1 2 3; 4 5 6];  
B = [6 2 0; 9 1 3];
```

To concatenate these along the third dimension:

```
C = cat(3, A, B)
```

```
C(:, :, 1) =
```

```
    1    2    3
    4    5    6
```

```
C(:, :, 2) =
```

```
    6    2    0
    9    1    3
```

You can also create an array with two or more dimensions in which every element has the same value using the `repmat` function. `repmat` accepts the value with which to fill the array, followed by a vector of dimensions for the array. For example, to create a 2-by-2-by-3-by-3 array B where every element has the value pi :

```
B = repmat(pi, [2 2 3 3]);
```

You can also use `repmat` to replicate or “tile” arrays in a specified configuration.

Function	Description
<code>cat</code>	Concatenate arrays.
<code>flipdim</code>	Flip array along specified dimension.
<code>ndgrid</code>	Generate arrays for multidimensional functions and interpolation.
<code>ndims</code>	Number of array dimensions.
<code>permute</code> , <code>i permute</code>	Permute the dimensions of a multidimensional array.
<code>reshape</code>	Change size.
<code>shiftdim</code>	Shift dimensions.

Function	Description
<code>squeeze</code>	Remove singleton array dimensions.
<code>sub2ind</code> , <code>ind2sub</code>	Single index from subscripts; subscripts from linear index.

Cell Arrays

Cell arrays have elements that are containers for any type of MATLAB data, including other cells. You can build cell arrays using:

- The cell array constructor `{}`
- Assignment statements (for instance, `A{2, 2} = 'string'`)
- The new `cell` function

Function	Description
<code>cell</code>	Create cell array.
<code>cell2struct</code>	Cell array to structure array conversion.
<code>celldisp</code>	Display top-level structure of cell array.
<code>cellplot</code>	Graphically display the structure of a cell array.
<code>num2cell</code>	Convert a matrix into a cell array.

Structures

Structures are constructs that have named fields containing any kind of data. For example, one field might contain a text string representing a name (`patient.name = 'Jane Doe'`), another might contain a scalar representing a billing amount (`patient.billing = 127.00`), and a third might hold a matrix of medical test results. You can organize these structures into arrays of data.

Create structure arrays by using individual assignment statements or the new `struct` function.

Function	Description
<code>fieldnames</code>	Field names of structure array.
<code>getfield</code>	Get field of structure array.
<code>rmfield</code>	Remove structure fields.
<code>setfield</code>	Set field of structure array.
<code>struct</code>	Create structure array.
<code>struct2cell</code>	Structure to cell array conversion.

MATLAB Objects

Object-oriented programming within the MATLAB environment was introduced with MATLAB 5.0.

Objects

The MATLAB programming language does not require the use of data types. For many applications, however, it is helpful to associate specific attributes with certain categories of data. To facilitate this, MATLAB allows you to work with *objects*. Objects are typed structures. A single *class* name identifies both the type of the structure and the name of the function that creates objects belonging to that class.

Objects differ from ordinary structures in two important ways:

Data Hiding. The structure fields of objects are not visible from the command line. Instead, you can access structure fields only from within a *method*, an M-file associated with the object class. Methods reside in class directories. Class directories have the same name as the class, but with a prepended `@` symbol. For example, a class directory named `@inline` might contain methods for a class called `inline`.

Function and Expression Overloading. You can create methods that override existing M-files. If an object calls a function, MATLAB first checks to see if there is a method of that name before calling a supplied M-file of that name. You can also provide methods that are called for MATLAB operators. For objects *a* and *b*, for instance, the expression *a + b* calls the method `plus(a, b)` if it exists.

Function	Description
<code>class</code>	Create or return class of object.
<code>isa</code>	True if object is a given class.
<code>inferiorto</code>	Inferior class relationship.
<code>superiorto</code>	Superior class relationship.

Character Arrays

Strings take up less memory than they did previously: MATLAB 4 required 64 bits per character for string data, whereas MATLAB 5.0 (and later releases) requires only 16 bits per character.

Function	Description
<code>base2dec</code>	Convert base <i>B</i> to decimal number.
<code>bin2dec</code>	Convert binary to decimal number.
<code>char</code>	Convert numeric values to string.
<code>dec2base</code>	Convert decimal number to base.
<code>dec2bin</code>	Convert decimal to binary number.
<code>mat2str</code>	Convert a matrix into a string.
<code>strcat</code>	String concatenation.
<code>strmatch</code>	Find possible matches for a string.

Function	Description
<code>strncmp</code>	Compare the first <code>n</code> characters of two strings.
<code>strvcat</code>	Vertical concatenation of strings.

Programming Capabilities

MATLAB 5.0 includes flow-control improvements and new M-file programming tools.

Flow-Control Improvements

MATLAB 5.0 features:

- A new flow-control statement, the `switch` statement
- More efficient evaluation of `if` expressions

The `switch` statement is a convenient way to execute code conditionally when you have many possible cases to choose from. It is no longer necessary to use a series of `elseif` statements:

```
switch input_num
    case -1
        disp('negative one');
    case 0
        disp('zero');
    case 1
        disp('positive one');
    otherwise
        disp('other value');
end
```

Only the first matching case is executed.

`switch` can handle multiple conditions in a single case statement by enclosing the case expression in a cell array. For example, assume `method` exists as a string variable:

```
switch lower(method)
    case {'linear', 'bilinear'}, disp('Method is linear')
    case 'cubic', disp('Method is cubic')
    case 'nearest', disp('Method is nearest')
    otherwise, disp('Unknown method.')
```

Command	Description
<code>case</code>	Case switch.
<code>otherwise</code>	Default part of switch statement.
<code>switch</code>	Conditionally execute code, switching among several cases.

MATLAB 5.0 evaluates `if` expressions more efficiently than MATLAB 4. For example, consider the expression `if a|b`. If `a` is true, then MATLAB will not evaluate `b`. Similarly, MATLAB won't execute statements following the expression `if a&b` in the event `a` is found to be false.

Operator	Description
<code>iscell</code>	True for a cell array.
<code>isequal</code>	True if arrays are equal.
<code>isfinite</code>	True for finite elements.
<code>islogical</code>	True for logical arrays.
<code>isnumeric</code>	True if input is a numeric array.
<code>isstruct</code>	True for a structure.
<code>logical</code>	Convert numeric values to logical vectors.

M-File Programming Tools

MATLAB 5.0 added three features to enhance MATLAB's M-file programming capabilities.

Variable Number of Input and Output Arguments

The `varargin` and `varargout` commands simplify the task of passing data into and out of M-file functions. For instance, the statement `function varargout = myfun(A, B)` allows M-file `myfun` to return an arbitrary number of output arguments, while the statement `function [C, D] = myfun(varargin)` allows it to accept an arbitrary number of input arguments.

Multiple Functions Within an M-File

MATLAB 5.0 supports having subfunctions within the body of an M-file. These are functions that the primary function in the file can access but that are otherwise invisible.

M-File Profiler

This utility lets you debug and optimize M-files by tracking cumulative execution time for each line of code. Whenever the specified M-file executes, the profiler counts how many time intervals each line uses.

Pseudocode M-Files

The `pcode` command saves a pseudocode version of a function or script to disk for later sessions. This *pseudocode* version is ready-to-use code that MATLAB can access whenever you invoke the function.

Function	Description
<code>addpath</code>	Append directory to MATLAB's search path.
<code>assignin</code>	Assign variable in workspace.
<code>edit</code>	Edit an M-file.
<code>editpath</code>	Modify current search path.
<code>evalin</code>	Evaluate variable in workspace.
<code>fullfile</code>	Build full filename from parts.

Function	Description
<code>inmem</code>	Functions in memory.
<code>inputname</code>	Input argument name.
<code>mfilename</code>	Name of the currently running M-file.
<code>mexext</code>	Return the MEX filename extension.
<code>pcode</code>	Create pseudocode file (P-file).
<code>profile</code>	Measure and display M-file execution profiles.
<code>rmpath</code>	Remove directories from MATLAB's search path.
<code>varargin</code> , <code>varargout</code>	Pass or return variable numbers of arguments.
<code>warning</code>	Display warning message.
<code>web</code>	Point Web browser at file or Web site.

New and Enhanced Language Functions

MATLAB 5.0 added a large number of new language functions as well as enhancements to existing functions.

Table 3-1: New Elementary and Specialized Math Functions

Function	Description
airy	Airy functions.
besselh	Bessel functions of the third kind (Hankel).
condeig	Condition number with respect to eigenvalues.
condest	1-norm matrix condition estimate.
dblquad	Numerical double integration
mod	Modulus (signed remainder after division).
normest	2-norm estimate.

Table 3-2: New Time and Date Functions

Function	Description
calendar	Calendar.
datenum	Serial date number.
datestr	Create date string.
date tick	Date formatted tick labels.
datevec	Date components.
eomday	End of month.
now	Current date and time.
weekday	Day of the week.

Table 3-3: New Ordinary Differential Equation Functions

Function	Description
ode45, ode23, ode113, ode23s, ode15s	Solve differential equations, low- and high-order methods.
odefile	Define a differential equation problem for ODE solvers.
odeget	Extract options from an argument created with odeset.
odeset	Create and edit input arguments for ODE solvers.

Table 3-4: New Matrix Functions

Function	Description
cholinc	Incomplete Cholesky factorization.
gallery	More than 50 new test matrices.
luintc	Incomplete LU factorization.
repmat	Replicate and tile an array.
sprand	Random uniformly distributed sparse matrices.

Table 3-5: New Methods for Sparse Matrices

Method	Description
bicg	BiConjugate Gradients method.
bicgstab	BiConjugate Gradients Stabilized method.
cgs	Conjugate Gradients Squared method.
eigs	Find a few eigenvalues and eigenvectors.
gmres	Generalized Minimum Residual method.

Table 3-5: New Methods for Sparse Matrices (Continued)

Method	Description
pcg	Preconditioned Conjugate Gradients method.
qmr	Quasi-Minimal Residual method.
svds	A few singular values.

Subscripting and Assignment Enhancements

In MATLAB 5.0, you can:

- Access the last element of an array using the end keyword.
- Obtain consistent results for indexing expressions consisting of all ones.
- Use scalar expansion in subarray assignments.

A statement like `A(ones([m, n]))` always returns an m -by- n array in which each element is `A(1)`. In previous versions, the statement returned different results depending on whether `A` was or was not an m -by- n matrix.

In previous releases, expressions like `A(2:3, 4:5) = 5` resulted in an error. MATLAB 5.0 automatically “expands” the 5 to be the right size (that is, `5*ones(2, 2)`).

Integer Bit Manipulation Functions

The `ops` directory contains commands that permit bit-level operations on integers. Operations include setting and unsetting, complementing, shifting, and logical AND, OR, and XOR.

Function	Description
<code>bitand</code>	Bitwise AND.
<code>bitcmp</code>	Complement bits.
<code>bitget</code>	Get bit.
<code>bitmax</code>	Maximum floating-point integer.
<code>bitor</code>	Bitwise OR.

Function	Description
<code>bitset</code>	Set bit.
<code>bitshift</code>	Bitwise shift.
<code>bitxor</code>	Bitwise XOR.

Dimension Specification for Data Analysis Functions

MATLAB's basic data analysis functions enable you to supply a second input argument. This argument specifies the dimension along which the function operates. For example, create an array A:

```
A = [3 2 4; 1 0 5; 8 2 6];
```

To sum along the first dimension of A, incrementing the row index, specify 1 for the dimension of operation:

```
sum(A, 1)

ans =

    12     4    15
```

To sum along the second dimension, incrementing the column index, specify 2 for the dimension:

```
sum(A, 2)

ans =

     9
     6
    16
```

Other functions that accept the dimension specifier include `prod`, `cumprod`, and `cumsum`.

Wildcards in Utility Commands

The asterisk (*) can be used as a wildcard in the `clear` and `whos` commands. This allows you, for example, to clear only variables beginning with a given character or characters, as in

```
clear A*
```

Empty Arrays

Earlier versions of MATLAB allowed for only one empty matrix, the 0-by-0 matrix denoted by `[]`. MATLAB 5.0 provides for matrices and arrays in which some, but not all, of the dimensions are zero. For example, 1-by-0, 10-by-0-by-20, and `[3 4 0 5 2]` are all possible array sizes.

The two-character sequence `[]` continues to denote the 0-by-0 matrix. Empty arrays of other sizes can be created with the functions `zeros`, `ones`, `rand`, or `eye`. To create a 0-by-5 matrix, for example, use

```
E = zeros(0, 5)
```

The basic model for empty matrices is that any operation that is defined for m -by- n matrices, and that produces a result with a dimension that is some function of m and n , should still be allowed when m or n is zero. The size of the result should be that same function, evaluated at zero.

For example, horizontal concatenation

```
C = [A B]
```

requires that A and B have the same number of rows. So if A is m -by- n and B is m -by- p , then C is m -by- $(n+p)$. This is still true if m or n or p is zero.

Many operations in MATLAB produce row vectors or column vectors. With MATLAB 5.0 it is possible for the result to be the empty row vector

```
r = zeros(1, 0)
```

or the empty column vector

```
c = zeros(0, 1)
```

Some MATLAB functions, like `sum` and `max`, are *reductions*. For matrix arguments, these functions produce vector results; for vector arguments they produce scalar results. Backwards compatibility issues arise for the argument `[]`, which in MATLAB 4 played the role of both the empty matrix and the empty vector. In MATLAB 5.0, empty inputs with these functions produce these results:

- `sum([])` is 0
- `prod([])` is 1
- `max([])` is `[]`
- `min([])` is `[]`

New Data Analysis Features

MATLAB 5.0 provides an expanded set of basic data analysis functions.

Function	Description
convhull	Convex hull.
cumtrapz	Cumulative trapezoidal numerical integration.
delaunay	Delaunay triangularization.
dsearch	Search for nearest point.
factor	Prime factors.
inpolygon	Detect points inside a polygonal region.
isprime	True for prime numbers.
nchoosek	All possible combinations of n elements taken k at a time.
perms	All possible permutations.
polyarea	Area of polygon.
primes	Generate a list of prime numbers.
sortrows	Sort rows in ascending order.
tsearch	Search for enclosing Delaunay triangle.
voronoi	Voronoi diagram.

MATLAB 5.0 also offers expanded data analysis in the areas of:

- Higher-dimension interpolation
- Extended `griddata` functionality based on Delaunay triangulation
- New set theoretic functions

Higher-Dimension Interpolation

The new functions `interp3` and `interpN` let you perform three-dimensional and multidimensional interpolation. `ndgrid` provides arrays that can be used in multidimensional interpolation.

Function	Description
<code>interp3</code>	Three-dimensional data interpolation (table lookup).
<code>interpN</code>	Multidimensional data interpolation (table lookup).
<code>ndgrid</code>	Generate arrays for multidimensional functions and interpolation.

griddata Based on Delaunay Triangulation

`griddat` supports triangle-based interpolation using nearest neighbor, linear, and cubic techniques. It creates smoother contours on scattered data using the cubic interpolation method.

Set Theoretic Functions

The functions `union`, `intersect`, `ismember`, `setdiff`, and `unique` treat vectors as sets, allowing you to perform operations like union $A \cup B$, intersection $A \cap B$, and difference $A - B$ of such sets. Other set-theoretical operations include location of common set elements (`ismember`) and elimination of duplicate elements (`unique`).

Function	Description
<code>intersect</code>	Set intersection of two vectors.
<code>ismember</code>	Detect members of a set.
<code>setdiff</code>	Return the set difference of two vectors.
<code>setxor</code>	Set XOR of two vectors.

Function	Description
uni on	Set union of two vectors.
uni que	Unique elements of a vector.

New and Enhanced Handle Graphics Features

MATLAB 5.0 features significant improvements to Handle Graphics. For details on all graphics functions, see *Using MATLAB Graphics*.

Plotting Capabilities

MATLAB's basic plotting capabilities were improved and expanded in MATLAB 5.0.

Function	Description
area	Filled area plot.
bar3	Vertical 3-D bar chart.
bar3h	Horizontal 3-D bar chart.
barh	Horizontal bar chart.
pie	Pie chart.
pie3	Three-dimensional pie chart.
plotyy	Plot graphs with Y tick labels on left and right.

Filling Areas

The area function plots a set of curves and fills the area beneath the curves.

Bar Chart Enhancements

bar3, bar3h, and barh draw vertical and horizontal bar charts. These functions, together with bar, support multiple filled bars in grouped and stacked formats.

Labels for Patches and Surfaces

legend can label any solid-color patch and surface. You can place legends on line, bar, ribbon, and pie plots, for example.

Function	Description
box	Axes box.
date tick	Display dates for Axes tick labels.

Marker Style Enhancement

A number of new line markers are available, including, among others, a square, a diamond, and a five-pointed star. These can be specified independently from line style.

Stem Plot Enhancements

stem and stem3 plot discrete sequence data as filled or unfilled stem plots.

Three-Dimensional Plotting Support

quiver3 displays three-dimensional velocity vectors with (u,v,w) components. The ribbon function displays data as three-dimensional strips.

Function	Description
quiver3	Three-dimensional quiver plot.
ribbon	Draw lines as 3-D strips.
stem3	Three-dimensional stem plot.

Data Visualization

MATLAB 5.0 features many new and enhanced capabilities for data visualization.

New Viewing Model

Axes camera properties control the orthographic and perspective view of the scene created by an Axes and its child objects. You can view the Axes from any location around or in the scene, as well as adjust the rotation, view angle, and target point.

New Method for Defining Patches

You can define a Patch using a matrix of faces and a matrix of vertices. Each row of the face matrix contains indices into the vertex matrix that define the connectivity of the face. Defining Patches in this way reduces memory consumption because you no longer need to specify redundant vertices.

Triangular Meshes and Surfaces

The new functions `tri mesh` and `tri surf` create triangular meshes and surfaces from `x`, `y`, and `z` vector data and a list of indices into the vector data.

Function	Description
<code>tri surf</code>	Triangular surface plot.
<code>tri mesh</code>	Triangular mesh plot.

Improved Slicing

`slice` supports an arbitrary slicing surface.

Contouring Enhancements

The contouring algorithm supports parametric surfaces and contouring on triangular meshes. In addition, `clabel` rotates and inserts labels in contour plots.

Function	Description
<code>contourf</code>	Filled contour plot.

New zoom Options

The `zoom` function supports two new options:

- `scale_factor` – zooms by the specified scale factor relative to the current zoom state (e.g., `zoom(2)` zooms in by a factor of two).
- `fill` – zooms to the point where the objects contained in the Axes are as large as they can be without extending beyond the Axes plot box from any view. Use this option when you want to rotate the Axes without seeing an apparent size change.

Graphics Presentation

MATLAB 5.0 provides improved control over the display of graphics objects.

Enhancements to Axes Objects

MATLAB 5.0 provides more advanced control for three-dimensional Axes objects. You can control the three-dimensional aspect ratio for the Axes' plot box, as well as for the data displayed in the plot box. You can also zoom in and out from a three-dimensional Axes using viewport scaling and Axes camera properties.

The `axis` command supports a new option designed for viewing graphics objects in 3-D:

```
axis vis3d
```

This option prevents MATLAB from stretching the Axes to fit the size of the Figure window and otherwise altering the proportions of the objects as you change the view.

In a two-dimensional view, you can display the x -axis at the top of an Axes and the y -axis at the right side of an Axes.

Color Enhancements

`colordf white` or `colordf black` changes the color defaults on the root so that subsequent figures produce plots with a white or black axes background color. The figure background color is changed to be a shade of gray, and many other defaults are changed so that there will be adequate contrast for most

`plots.colordf` none sets the defaults to their MATLAB 4 values. In addition, a number of new colormaps are available.

Table 3-6: New Figure and Axis Color Control

Function	Description
<code>colordf</code>	Select Figure color scheme.

Table 3-7: New Colormaps

Function	Description
<code>autumn</code>	Shades of red and yellow colormap.
<code>colcube</code>	Regularly spaced colors in RGB colorspace, plus more steps of gray, pure red, pure green, and pure blue.
<code>lines</code>	Colormap of colors specified by the Axes' <code>ColorOrder</code> property.
<code>spring</code>	Shades of magenta and yellow colormap.
<code>summer</code>	Shades of green and yellow colormap.
<code>winter</code>	Shades of blue and green colormap.

Text Object Enhancements

MATLAB 5.0 supports a subset of TeX commands. A single Text graphics object can support multiple fonts, subscripts, superscripts, and Greek symbols. See the `text` function in the online MATLAB Function Reference for information about the supported TeX subset.

You can also specify multiline character strings and use normalized font units so that Text size is a fraction of an Axes' or Uicontrol's height. MATLAB supports multiline text strings using cell arrays. Simply define a string variable as a cell array with one line per cell.

Improved General Graphics Features

The MATLAB startup file sets default properties for various graphics objects so that new Figures are aesthetically pleasing and graphs are easier to understand.

Command	Description
<code>dialog</code>	Create a dialog box.

Z-buffering is available for fast and accurate three-dimensional rendering.

MATLAB 5.0 provides built-in menus on X Window systems. Figure MenuBar 'figure' is supported on UNIX.

Lighting

MATLAB supports a new graphics object called a Light. You create a Light object using the `light` function. Three important Light object properties are:

- `Color` – the color of the light cast by the Light object
- `Style` – either infinitely far away (the default) or local
- `Position` – the direction (for infinite light sources) or the location (for local light sources)

You cannot see Light objects themselves, but you can see their effect on any Patch and Surface objects present in the same Axes. You can control these effects by setting various Patch and Surface object properties.

`AmbientStrength`, `DiffuseStrength`, and `SpecularStrength` control the intensity of the respective light-reflection characteristics;

`SpecularColorReflectance` and `SpecularExponent` provide additional control over the reflection characteristics of specular light.

The Axes `AmbientLightColor` property determines the color of the ambient light, which has no direction and affects all objects uniformly. Ambient light effects occur only when there is a visible Light object in the Axes.

The Light object's `Color` property determines the color of the directional light, and its `Style` property determines whether the light source is a point source (`Style` set to `local`), which radiates from the specified position in all directions,

or a light source placed at infinity (`Style` set to `infinite`), which shines from the direction of the specified position with parallel rays.

You can also select the algorithm used to calculate the coloring of the lit objects. The `Patch` and `Surface` `EdgeLighting` and `FaceLighting` properties select between no lighting, and flat, Gouraud, or Phong lighting algorithms.

print Command Revisions

The `print` command has been extensively revised for MATLAB 5.0. Consult *Using MATLAB Graphics* for a complete description of `print` command capabilities. Among the new options available for MATLAB 5.0:

- The `-loose` option makes the PostScript bounding box equal to the Figure's `PaperPosition` property. PICT (Macintosh) and EPSI (X Window systems) previews are the same size as the generated PostScript drawing.
- Z-buffer images can be printed at user-selectable resolution.
- The `-dmeta` option supports Enhanced Windows Metafiles.
- `print -dmfile` generates an M-file that recreates a Figure.
- Uicontrol objects print by default unless suppressed with the `-noui` option. In earlier versions of MATLAB, uicontrols did not appear when you printed Figures. If you specify the `-noui` option with the `print` command, MATLAB ignores uicontrols and prints only Axes and Axes children.

Additional print Device Options

MATLAB 5.0 introduced several new device options for the `print` command:

Device	Description
<code>-dljet4</code>	HP LaserJet 4 (defaults to 600 dpi)
<code>-ddesket</code>	HP DeskJet and DeskJet Plus
<code>-ddjet500</code>	HP Deskjet 500
<code>-dcdj500</code>	HP DeskJet 500C
<code>-dcdj550</code>	HP Deskjet 550C
<code>-dpjxl</code>	HP PaintJet XL color printer

Device	Description
-dpj xl 300	HP PaintJet XL300 color printer
-ddnj 650c	HP DesignJet 650C
-dbj 200	Canon BubbleJet BJ200
-dbj c600	Canon Color BubbleJet BJC-600 and BJC-4000
-di bmpro	IBM 9-pin Proprinter
-dbmp256	8-bit (256-color) BMP file format
-dbmp16m	24-bit BMP file format
-dpcxmono	Monochrome PCX file format
-dpcx24b	24-bit color PCX file format, three 8-bit planes
-dpbm	Portable Bitmap (plain format)
-dpbmr aw	Portable Bitmap (raw format)
-dpgm	Portable Graymap (plain format)
-dpgmr aw	Portable Graymap (raw format)
-dppm	Portable Pixmap (plain format)
-dppmr aw	Portable Pixmap (raw format)

Image Support

MATLAB 5.0 provides a number of enhancements to image support. These enhancements include:

- Truecolor support
- New functions for reading images from, and writing images to, graphics files
- 8-bit image support

Truecolor

In addition to indexed images, in which colors are stored as an array of indices into a colormap, MATLAB 5.0 supports truecolor images. A truecolor image does not use a colormap; instead, the color values for each pixel are stored directly as RGB (red, green, blue) triplets. In MATLAB, the `CData` property of a truecolor Image object is a three-dimensional (m -by- n -by-3) array. This array consists of three m -by- n matrices (representing the red, green, and blue color planes) concatenated along the third dimension.

Reading and Writing Images

The `imread` function reads image data into MATLAB arrays from graphics files in various standard formats, such as TIFF. You can then display these arrays using the `image` function, which creates a Handle Graphics Image object. You can also write MATLAB image data to graphics files using the `imwrite` function. `imread` and `imwrite` both support a variety of graphics file formats and compression schemes.

8-Bit Images

When you read an image into MATLAB using `imread`, the data is stored as an array of 8-bit integers. This is a much more efficient storage method than the double-precision (64-bit) floating-point numbers that MATLAB typically uses.

The Handle Graphics Image object has been enhanced to support 8-bit `CData`. This means you can display 8-bit images without having to convert the data to double precision. MATLAB 5.0 also supports a limited set of operations on these 8-bit arrays. You can view the data, reference values, and reshape the array in various ways. To perform any mathematical computations, however, you must first convert the data to double precision, using the `double` function.

Note that, in order to support 8-bit images, certain changes have been made in the way MATLAB interprets image data. This table summarizes the conventions MATLAB uses:

Image Type	Double-Precision Data (Double Array)	8-Bit Data (uint8 Array)
Indexed (colormap)	Image is stored as a 2-D (m-by-n) array of integers in the range [1, length(colormap)]; colormap is an m-by-3 array of floating-point values in the range [0, 1].	Image is stored as a 2-D (m-by-n) array of integers in the range [0, 255]; colormap is an m-by-3 array of floating-point values in the range [0, 1].
Truecolor (RGB)	Image is stored as a 3-D (m-by-n-by-3) array of floating-point values in the range [0, 1].	Image is stored as a 3-D (m-by-n-by-3) array of integers in the range [0, 255].

Note that MATLAB interprets image data very differently depending on whether it is double precision or 8-bit. The rest of this section discusses factors you should keep in mind when working with image data to avoid potential pitfalls. This information is especially important if you want to convert image data from one format to another.

Indexed Images

In an indexed image of class `double`, the value 1 points to the first row in the colormap, the value 2 points to the second row, and so on. In a `uint8` indexed image, there is an offset; the value 0 points to the first row in the colormap, the value 1 points to the second row, and so on. The `uint8` convention is also used in graphics file formats, and enables 8-bit indexed images to support up to 256 colors. Note that when you read in an indexed image with `imread`, the resulting image array is always of class `uint8`. (The colormap, however, is of class `double`; see below.)

If you want to convert a `uint8` indexed image to `double`, you need to add 1 to the result. For example:

```
X64 = double(X8) + 1;
```

To convert from `double` to `uint8`, you need to first subtract 1, and then use `round` to ensure that all the values are integers:

```
X8 = uint8(round(X64 - 1));
```

The order of the operations must be as shown in these examples because you cannot perform mathematical operations on `uint8` arrays.

When you write an indexed image using `imwrite`, MATLAB automatically converts the values if necessary.

Colormaps

Colormaps in MATLAB are always `m`-by-3 arrays of double-precision floating-point numbers in the range [0, 1]. In most graphics file formats, colormaps are stored as integers, but MATLAB does not support colormaps with integer values. `imread` and `imwrite` automatically convert colormap values when reading and writing files.

Truecolor Images

In a truecolor image of class `double`, the data values are floating-point numbers in the range [0, 1]. In a truecolor image of class `uint8`, the data values are integers in the range [0, 255].

If you want to convert a truecolor image from one data type to the other, you must rescale the data. For example, this call converts a `uint8` truecolor image to `double`:

```
RGB64 = double(RGB8) / 255;
```

This call converts a `double` truecolor image to `uint8`:

```
RGB8 = uint8(round(RGB*255));
```

The order of the operations must be as shown in these examples, because you cannot perform mathematical operations on `uint8` arrays.

When you write a truecolor image using `imwrite`, MATLAB automatically converts the values if necessary.

New and Enhanced Handle Graphics Object Properties

This section lists new graphics object properties supported in MATLAB 5.0. It also lists graphics properties whose behavior has changed significantly. *Using MATLAB Graphics* provides a more detailed description of each property.

Table 3-8: Properties of All Graphics Objects

Property	Description
BusyAction	Controls events that potentially interrupt executing callback routines.
Children	Enhanced behavior allows reordering of child objects.
CreateFcn	A callback routine that executes when MATLAB creates a new instance of the specific type of graphics object.
DeleteFcn	A callback routine that executes when MATLAB deletes the graphics object.
HandleVisibility	Controls scope of handle visibility.
Interruptible	On by default.
Parent	Enhanced behavior allows reparenting of graphics objects.
Selected	Indicates whether graphics object is in selected state.
SelectOnHighlight	Determines if graphics objects provide visual indication of selected state.
Tag	User-specified object label.

Table 3-9: Axes Properties

Property	Description
AmbientLightColor	Color of the surrounding light illuminating all Axes child objects when a Light object is present.
CameraPosition	Location of the point from which the Axes is viewed.
CameraPositionMode	Automatic or manual camera positioning.
CameraTarget	Point in Axes viewed from camera position.
CameraTargetMode	Automatic or manual camera target selection.
CameraUpVector	Determines camera rotation around the viewing axis.
CameraUpVectorMode	Default or user-specified camera orientation.
CameraViewAngle	Angle determining the camera field of view.
CameraViewAngleMode	Automatic or manual camera field of view selection.
DataAspectRatio	Relative scaling of x -, y -, and z -axis data units.
DataAspectRatioMode	Automatic or manual axis data scaling.
FontUnits	Units used to interpret the FontSize property (allowing normalized text size).
Layer	Draw axis lines below or above child objects.
NextPlot	Enhanced behavior supports add, replace, and replacechildren options.
PlotBoxAspectRatio	Relative scaling of Axes plot box.

Table 3-9: Axes Properties (Continued)

Property	Description
PlotBoxAspectRatioMode	Automatic or manual selection of plot box scaling.
Projection	Select orthographic or perspective projection type.
TickDirMode	Automatic or manual selection of tick mark direction (allowing you to change view and preserve the specified TickDir).
XAxisLocation	Locate <i>x</i> -axis at bottom or top of plot.
YAxisLocation	Locate <i>y</i> -axis at left or right side of plot.

[../ref/figure_props.html#](#)

Table 3-10: Figure Properties

Property	Description
CloseRequestFcn	Callback routine executed when you issue a <code>close</code> command on a Figure.
Dithermap	Colormap used for truecolor data on pseudocolor displays.
DithermapMode	Automatic dithermap generation.
IntegerHandle	Integer or floating-point Figure handle.
NextPlot	Enhanced behavior supports <code>add</code> , <code>replace</code> , and <code>replacechildren</code> options.
PaperPositionMode	WYSIWYG printing of Figure.
PointerShapeCData	User-defined pointer data.
PointerShapeHotSpot	Active point in custom pointer.
Renderer	Select painters or Z-buffer rendering.

Table 3-10: Figure Properties (Continued)

Property	Description
RendererMode	Enable MATLAB to select best renderer automatically.
Resize	Determines if Figure window is resizable.
ResizeFcn	Callback routine executed when you resize the Figure window.

Table 3-11: Image Properties

Property	Description
CData	Enhanced behavior allows truecolor (RGB values) specification.
CDataMapping	Select direct or scaled interpretation of indexed colors.
EraseMode	Control drawing and erasing of Image objects.

Table 3-12: Light Properties

Property	Description
Color	Color of the light source.
Position	Place the light source within Axes space.
Style	Select infinite or local light source.

Table 3-13: Line Properties

Property	Description
Marker	The marker symbol to use at data points (markers are separate from line style).
MarkerEdgeColor	The color of the edge of the marker symbol.
MarkerFaceColor	The color of the face of filled markers.

Table 3-14: Patch Properties

Property	Description
AmbientStrength	The strength of the Axes ambient light on the particular Patch object.
CData	Enhanced behavior allows truecolor (RGB values) specification.
CDataMapping	Select direct or scaled interpretation of indexed colors.
DiffuseStrength	Strength of the reflection of diffuse light from Light objects.
FaceLightingAlgorithm	Lighting algorithm used for Patch faces.
Faces	The vertices connected to define each face.
FaceVertexCData	Color specification when using the Faces and Vertices properties to define a Patch.
LineStyle	Type of line used for edges.
Marker	Symbol used at vertices.
MarkerEdgeColor	The color of the edge of the marker symbol.
MarkerFaceColor	The color of the face of filled markers.
MarkerSize	Size of the marker.

Table 3-14: Patch Properties (Continued)

Property	Description
Normal Mode	MATLAB generated or user-specified normal vectors.
SpecularColorReflectance	Control the color of the specularly reflected light from Light objects.
SpecularExponent	Control the shininess of the Patch object.
SpecularStrength	Strength of the reflection of specular light from Light objects.
VertexNormals	Definition of the Patch's normal vectors.
Vertices	The coordinates of the vertices defining the Patch.

Table 3-15: Root Properties

Property	Description
CallbackObject	Handle of object whose callback is currently executing.
ErrorMessage	Text of the last error message issued by MATLAB.
ErrorType	The type of the error that last occurred.
ShowHiddenHandles	Show or hide graphics object handles that are marked as hidden.
TerminalHideGraphCommand	Command to hide graphics window when switching to command mode.
TerminalDimensions	Size of graphics terminal.
TerminalShowGraphCommand	Command to expose graphics window when switching from command mode to graphics mode.

Table 3-16: Surface Properties

Property	Description
AmbientStrength	The strength of the Axes ambient light on the particular Surface object.
CData	Enhanced behavior allows truecolor (RGB values) specification.
CDataMapping	Selects direct or scaled interpretation of indexed colors.
DiffuseStrength	Strength of the reflection of diffuse light from Light objects.
FaceLightingAlgorithm	Lighting algorithm used for Surface faces.
Marker	Symbol used at vertices.
MarkerEdgeColor	The color of the edge of the marker symbol.
MarkerFaceColor	The color of the face of filled markers.
MarkerSize	Size of the marker.
NormalMode	MATLAB generated or user-specified normal vectors.
SpecularColorReflectance	Control the color of the specularly reflected light from Light objects.
SpecularExponent	Control the shininess of the Surface object.
SpecularStrength	Strength of the reflection of specular light from Light objects.
VertexNormals	Definition of the Surface's normal vectors.
Vertices	The coordinates of the vertices defining the Surface.

Table 3-17: Text Properties

Property	Description
FontUnits	Select the units used to interpret the FontSize property (allowing normalized text size).
Interpreter	Allows MATLAB to interpret certain characters as TeX commands.

Table 3-18: Uicontrol Properties

Property	Description
Enable	Enable or disable (gray out) uicontrols.
FontAngle	Select character slant.
FontName	Select font family.
FontSize	Select font size.
FontUnits	Select the units used to interpret the FontSize property (allowing normalized text size).
FontWeight	Select the weight of text characters.
ListboxTop	Select the listbox item to display at the top of the listbox.
SliderStep	Select the size of the slider step.
Style	Enhanced to include listbox device.

Table 3-19: Uimenu Properties

Property	Description
Enable	Enable or disable (gray out) uicontrols.

Improvements to Graphical User Interfaces (GUIs)

General GUI Enhancements

MATLAB 5.0 provides general enhancements that are useful in the GUI area:

- Starting MATLAB with the `-nosplash` argument suppresses the splash screen on UNIX.
- Using the `CloseRequestFcn` callback can abort a `Figure close` command.
- Stacking of `Figure` and `Axes` graphics objects can be varied to affect the order in which MATLAB displays these objects.
- The mouse pointer can be set to a number of different symbols or you can create a custom `Figure` pointer.
- On the Microsoft Windows platforms, edit controls have a three-dimensional appearance.

MATLAB 5.0 provides features that make it easier to create MATLAB GUIs. Major enhancements include `List Box` objects to display and select one or more list items. You can also create modal or non-modal error, help, and warning message boxes. In addition, `uicontrol` edit boxes support multiline text.

Function	Description
<code>dragrect</code>	Drag predefined rectangles.
<code>inputdlg</code>	Display a dialog box to input data.
<code>msgbox</code>	Display message box.
<code>questdlg</code>	Question dialog.
<code>rbbox</code>	Rubberband box.
<code>selectmoveresize</code>	Interactively select, move, or resize objects.

MATLAB 5.0 also provides more flexibility in callback routines. You can specify callbacks that execute after creating, changing, and deleting an object.

Function	Description
<code>ui resume</code>	Resume suspended M-file execution.
<code>ui wai t</code>	Block program execution.
<code>wai tfor</code>	Block execution until a condition is satisfied.

Guide

Guide is a GUI design tool. The individual pieces of the Guide environment are designed to work together, but they can also be used individually. For example, there is a Property Editor (invoked by the command `propedi t`) that allows you to modify any property of any Handle Graphics object, from a figure to a line. Point the Property Editor at a line and you can change its color, position, thickness, or any other line property.

The Control Panel is the centerpiece of the Guide suite of tools. It lets you “control” a figure so that it can be easily modified by clicking and dragging. As an example, you might want to move a button from one part of a figure to another. From the Control Panel you put the button’s figure into an editable state, and then it’s simply a matter of dragging the button into the new position. Once a figure is editable, you can also add new uicontrols, uimenu, and plotting axes.

Tool	Command	Description
Control Panel	<code>gui de</code>	Control figure editing.
Property Editor	<code>propedi t</code>	Modify object properties.
Callback Editor	<code>cbedi t</code>	Modify object callbacks.
Alignment Tool	<code>al i gn</code>	Align objects.
Menu Editor	<code>menuedi t</code>	Modify figure menus.

Enhanced Application Program Interface (API)

The MATLAB 5.0 API introduces data types and functions not present in MATLAB 4. This section summarizes the important changes in the API. For details on any of these topics, see the *MATLAB Application Program Interface Guide*.

New Fundamental Data Type

The MATLAB 4 `Matrix` data type is obsolete. MATLAB 5.0 programs use the `mxArray` data type in place of `Matrix`. The `mxArray` data type has extra fields to handle the richer data constructs of MATLAB 5.0.

Functions that expected `Matrix` arguments in MATLAB 4 expect `mxArray` arguments in MATLAB 5.0.

New Functions

The MATLAB 5.0 API introduced many new functions that work with the C language to support MATLAB 5.0 features.

Support for Structures and Cells

MATLAB 5.0 introduced structure arrays and cell arrays. Therefore, the MATLAB 5.0 API introduced a broad range of functions to create structures and cells, as well as functions to populate and analyze them. See Chapter 4 of this document for a complete listing of these functions.

Support for Multidimensional Arrays

The MATLAB 4 `Matrix` data type assumed that all matrices were two-dimensional. The MATLAB 5.0 `mxArray` data type supports arrays of two or more dimensions. The MATLAB 5.0 API provides two different `mxCreate` functions that create either a two-dimensional or a multidimensional `mxArray`.

In addition, MATLAB 5.0 introduces several functions to get and set the number and length of each dimension in a multidimensional `mxArray`.

Support for Nondouble Precision Data

The MATLAB 4 `Matrix` data type represented all numerical data as double-precision floating-point numbers. The MATLAB 5.0 `mxArray` data type can store numerical data in six different integer formats and two different floating-point formats.

Note: Although the MATLAB API supports these different data representations, MATLAB itself does not currently provide any operations or functions that work with nondouble-precision data. Nondouble precision-data may be viewed, however.

Enhanced Debugging Support

MATLAB 5.0 includes more powerful tools for debugging C MEX-files. The `-argcheck` option to the `mex` script provides protection against accidental misuse of API functions (such as passing NULL pointers). In addition, there is increased documentation on troubleshooting common problems.

Enhanced Compile Mechanism

MATLAB 5.0 replaced the old `cmex` and `fmex` scripts with `mex`, which compile C or Fortran MEX-files. All compiler-specific information was moved to easily readable and highly configurable options files. The `mex` script has a configurable set of flags across platforms and can be accessed from within MATLAB via the `mex.m` M-file.

MATLAB 4 Feature Unsupported in MATLAB 5.0

Non-ANSI C Compilers

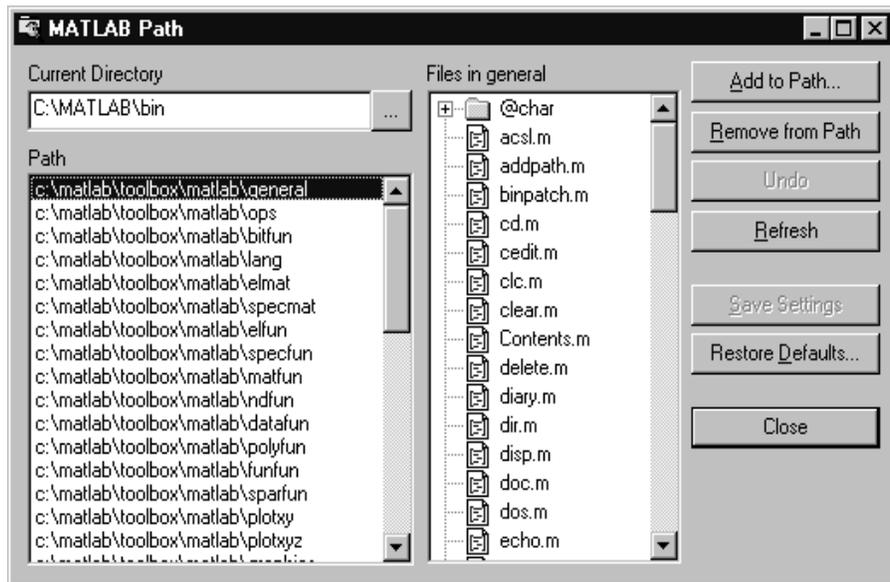
MATLAB 4 let you compile MATLAB applications with non-ANSI C compilers. MATLAB 5.0 requires an ANSI C compiler.

New Platform-Specific Features

Microsoft Windows

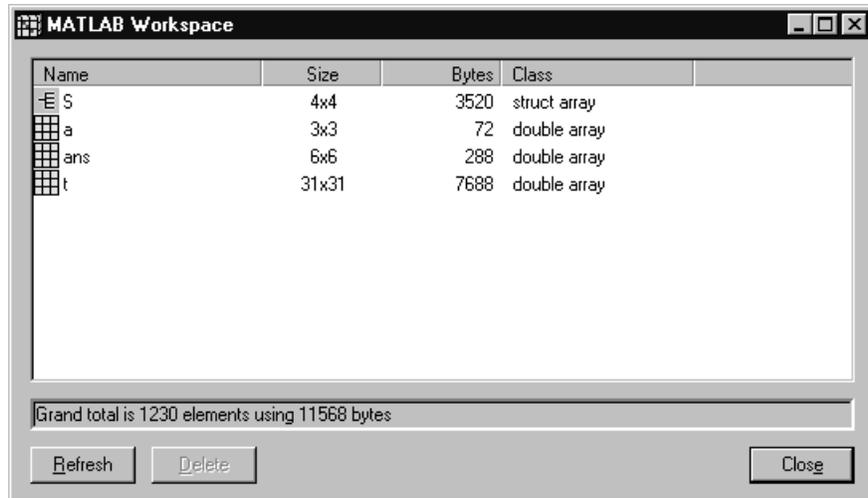
Path Browser

The Path Browser lets you view and modify the MATLAB search path. All changes take effect in MATLAB immediately.



Workspace Browser

The Workspace Browser lets you view the contents of the current MATLAB workspace. It provides a graphical representation of the traditional whos output. In addition, you can clear workspace variables and rename them.



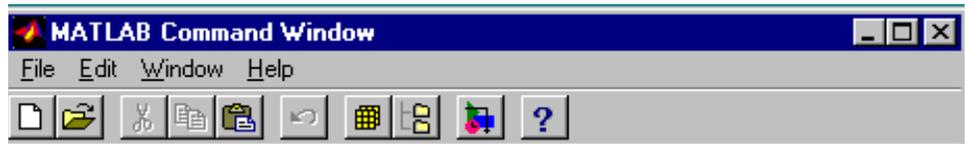
M-File Editor/Debugger

The graphical M-file Editor/Debugger allows you to set breakpoints and single-step through M-code. The M-file Editor/Debugger starts automatically when a breakpoint is hit. When MATLAB is installed, this program becomes the default editor.



Command Window Toolbar

A toolbar is optionally present for the Command Window. The toolbar provides single-click access to several commonly used operations:

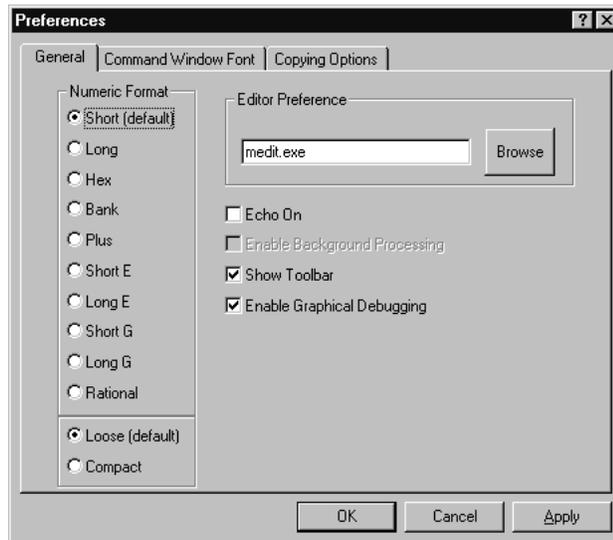


- Open a new editor window
- Open a file for editing
- Cut, copy, paste, and undo
- Open the Workspace Browser
- Open the Path Browser
- Create new Simulink model (if Simulink is installed)
- Access the Help facility

New Dialog Boxes

New **Preferences** dialog boxes are accessible through the **File** menu. Some of these were previously available through the **Options** menu in MATLAB 4. There are three categories of preferences:

- General
- Command Window Font
- Copying Options



16-bit Stereo Sound

MATLAB 5.0 supports 16-bit stereo sound on the Windows platform.

Macintosh

Two new features available on the Macintosh platform are:

- Japanese characters
It is possible to generate annotation and string constants that use Japanese characters.
- 16-bit stereo sound
MATLAB 5.0 supports 16-bit stereo sound.

User Interface Enhancements

- Optional toolbars in the Command Window, Editor windows, and M-file debugger allow rapid access to commonly used features.



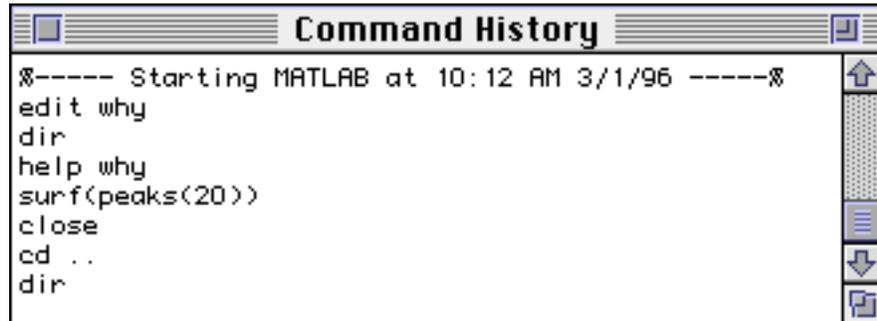
- Color syntax highlighting in the Command Window, Editor windows, and M-file debugger provides visual cues for identifying blocks of code, comments, and strings.
- Almost all lists and text items in the Command Window, Editor, Path Browser, Workspace Browser, M-file debugger, and Command History Window have optional dynamic or “live” scrolling; the display is scrolled as the scroll box of a scrollbar is moved.
- Macintosh Drag and Drop is supported throughout MATLAB for rapid and easy exchange of text between windows.

Command Window Features

- Typing on the current command line can be undone and redone. This includes cutting, clearing, overtyping, dragging, and dropping.
- Placing the caret on an error message and pressing **Enter** opens the M-file in the Editor, positioned to the offending line.

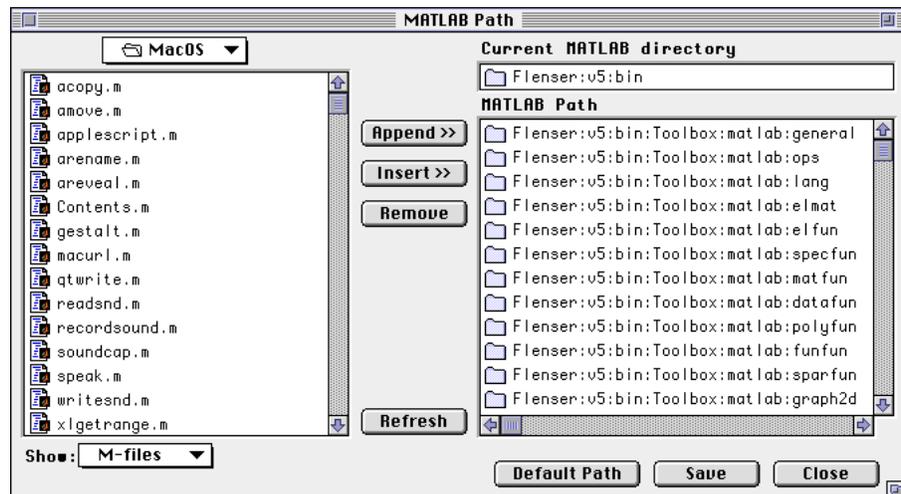
Command History Window

The Command History window contains a list of all commands executed from the Command Window. Commands are saved between MATLAB sessions, so you can select and execute a group of commands from a previous day’s work to continue quickly from where you left off.



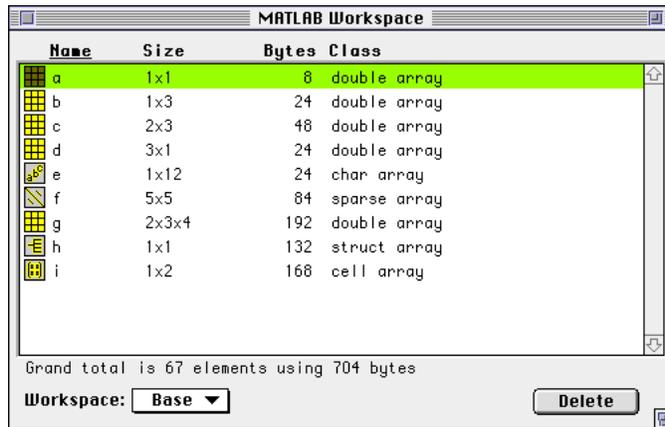
Path Browser

The Path Browser provides an intuitive, easy-to-use graphical interface for viewing and modifying the MATLAB search path. You can reorder or modify the search path simply by dragging items in the path list. Similarly, you can change the current MATLAB directory by dragging any folder into the current MATLAB directory area.



Workspace Browser

The Workspace Browser allows you to view the contents of the current MATLAB workspace. It provides a graphic representation of the traditional whos output. You can delete variables from the workspace and sort the workspace by various criteria. Double-clicking on a workspace variable displays that variable's contents in the Command Window.



M-File Debugger

MATLAB 5.0 includes a graphical M-file debugger, which allows you to set breakpoints and single-step through M-code. Selecting text in the debugger window and pressing the **Enter** (not the **Return**) key evaluates that text in the Command Window.

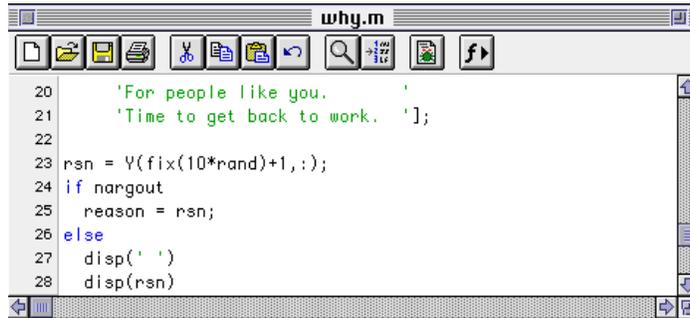
```

1  function reason = why
2  %WHY Provides succinct answers to any questions.
3  % WHY displays a succinct answer.
4  %
5  % S = WHY returns the reason in the string S.
6
7  % Copyright (c) 1984-96 by The MathWorks, Inc.
8  % $Revision: 5.4 $ $Date: 1996/04/24 20:24:53 $
9
10 ● Y = ['How the hell should I know?'
11      'Why not? '
12      'It feels good. '
13      'R.T.F.M. '
14      'Why? Because we like you! '
15      'Stupid question. '
16      'Jack made me do it. '
17      'Because it''s there. '
18      'For people like you. '
19      'Time to get back to work. '];
20 → rsn = Y(fix(10*rand)+1,:);
21 if nargin
22     reason = rsn;
23 else
24     disp(' ')
25     ' / \

```

Editor Features

- Command-clicking in the title of an Editor window displays a pop-up menu containing the full path to the M-file. Selecting a folder from the pop-up menu opens that folder in the Finder.
- Selecting text in an Editor window and pressing **Enter** evaluates that text in the Command Window.
- Typing a close parenthesis, bracket, or brace briefly highlights the matching open parenthesis, bracket, or brace.
- Double-clicking a parenthesis, bracket, or brace selects all text within the matching parenthesis, bracket, or brace.
- Line numbers can be optionally displayed

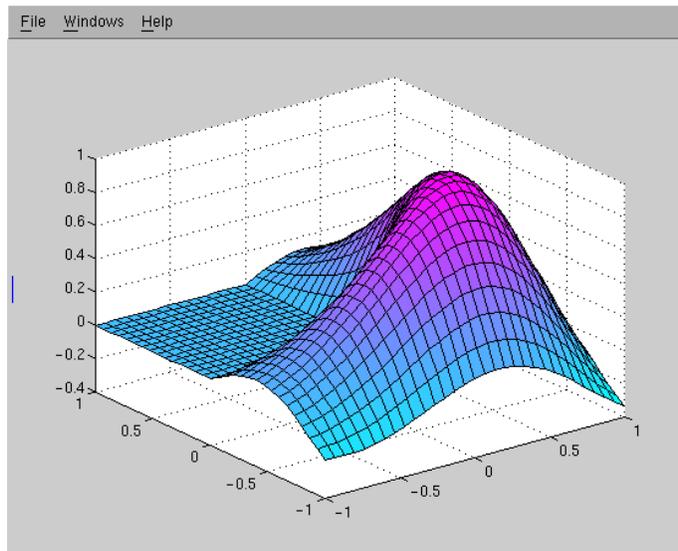


```
20     'For people like you.     '  
21     'Time to get back to work.  '];  
22  
23 rsn = V(fix(10*rand)+1,:);  
24 if nargin  
25     reason = rsn;  
26 else  
27     disp(' ')  
28     disp(rsn)
```

UNIX Workstations

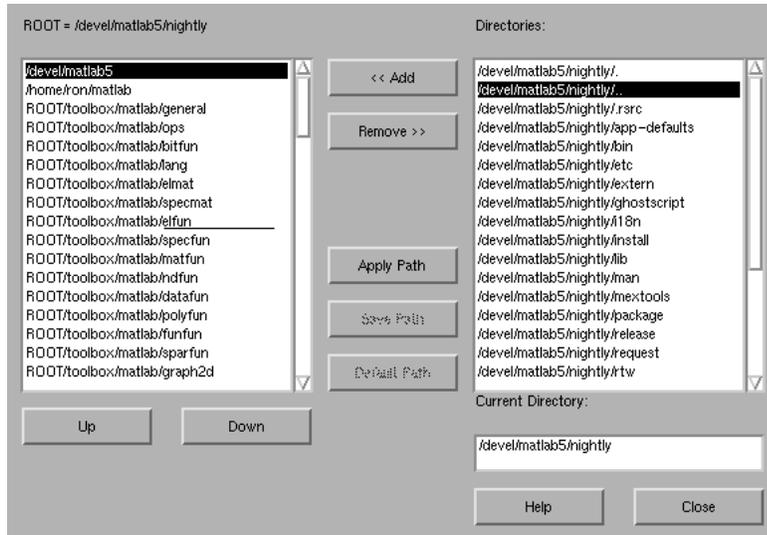
Figure Window Toolbar

The MATLAB 5.0 Figure window provides a toolbar with a **File** pull-down menu. Selecting the **Print** option on the **File** menu activates a set of push buttons that allows easy setting of the most frequently used print options.



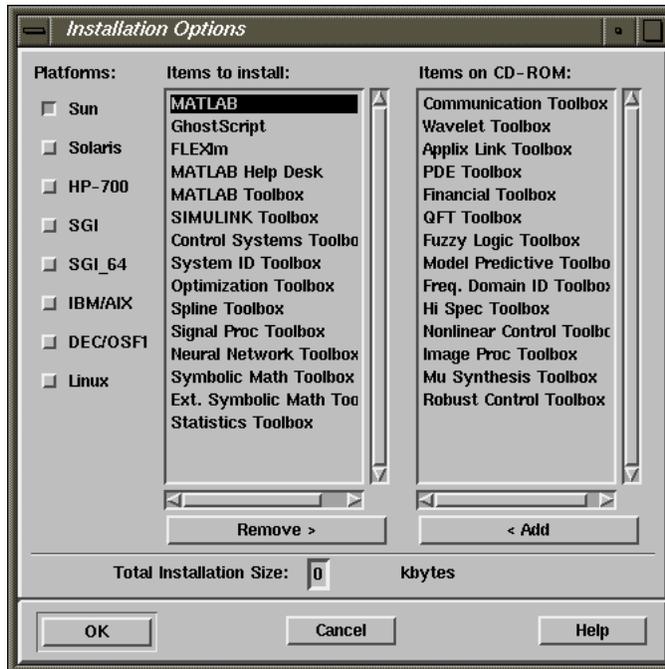
Path Editor

The `editpath` command displays a GUI that allows you to view and modify your MATLAB search path.



Simplified Installation Procedure

The MATLAB 5.0 installation procedure uses a GUI to select or deselect products and platforms.



Upgrading to MATLAB 5.2

Migrating to MATLAB 5.2	4-2
Roadmap for Different Migration Routes	4-2
Toolboxes and Blocksets	4-2
Upgrading from MATLAB 5.1 to MATLAB 5.2	4-3
Change to clear Behavior	4-3
try, catch, and persistent Are Now Keywords	4-3
Matrix Assignment	4-3
Change to Method Search Order	4-3
API Memory Management Compatibility Issue	4-4
Upgrading from MATLAB 5.0 to MATLAB 5.2	4-9
Upgrading from MATLAB 4 to MATLAB 5.2	4-11
Converting M-Files to MATLAB 5.0	4-11
Converting MATLAB 4 External Interface Programs to the MATLAB 5.0 Application Program Interface	4-25
Upgrading Toolboxes and Blocksets	4-40
Fuzzy Logic Toolbox: Updating FIS Models	4-40
DSP Blockset: Upgrading from Version 1.0a	4-40

Migrating to MATLAB 5.2

It is useful to introduce two terms in discussing this migration. The first step in converting your code to MATLAB 5.2 is to make it MATLAB 5.2 *compatible*. This involves a rather short list of possible changes that let your M-files run under MATLAB 5.2. The second step is to make it MATLAB 5.2 *compliant*. This involves making further changes so that your M-file is not using obsolete, but temporarily supported, features of MATLAB. It also can mean taking advantage of MATLAB 5.2 features like the new data constructs, graphics, and so on.

There are a relatively small number of things that are likely to be in your code that you must change to make your M-files MATLAB 5.2 compatible. Most of these are in the graphics area.

There are a somewhat larger number of things you can do (but don't have to) to make your M-files fully MATLAB 5.2 compliant. To help you gradually make your code compliant, MATLAB 5.2 displays warning messages when you use functions that are obsolete, even though they still work correctly.

Roadmap for Different Migration Routes

The sections of this chapter that you need to read depend on the version of MATLAB from which you are upgrading.

If You Are Upgrading to MATLAB 5.2 from...	Read These Sections ...
MATLAB 5.1	"Upgrading from MATLAB 5.1 to MATLAB 5.2"
MATLAB 5.0	"Upgrading from MATLAB 5.0 to MATLAB 5.2" "Upgrading from MATLAB 5.1 to MATLAB 5.2"
MATLAB 4	All

Toolboxes and Blocksets

The last section discusses upgrade issues for toolboxes and blocksets. Only the Signal Processing Toolbox and the DSP Blockset have any upgrade issues.

Upgrading from MATLAB 5.1 to MATLAB 5.2

This section describes some changes you can make to your code to eliminate error messages and warnings due to incompatible and noncompliant statements.

Change to clear Behavior

The `clear` function does not remove an M-file from the MATLAB workspace if that M-file is locked with the `ml lock` function, introduced in MATLAB 5.2.

Although pre-Version 5.2 code does not use `ml lock`, if that code is modified to use `ml lock`, `clear` will not behave as it did in previous versions of MATLAB (i.e., it will not be guaranteed to clear all M-files in the workspace). To unlock an M-file, use `munlock`.

try, catch, and persistent Are Now Keywords

You can no longer use `try`, `catch`, and `persistent` as variable names in MATLAB 5.2. In previous releases MATLAB did not treat these as keywords.

Matrix Assignment

In pre-Version 5.2, for

```
A(:) = b
```

where `b` is a scalar or vector, the resulting type was the type of `b`. In MATLAB 5.2, the resulting type is the type of `A`. So, if `A` is a `uint8` array to begin with, and `b` is a `double`, the result is that `A` is still a `uint8`.

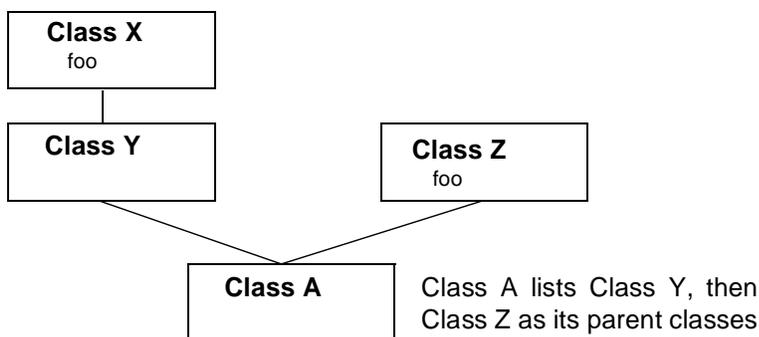
This change to preserve `A`'s type was made to ensure consistent indexing behavior.

Change to Method Search Order

When an object inherits from two different classes, MATLAB 5.2 performs a depth-first search when a method is called (this is how the method search was documented as working in previous versions). MATLAB 5.2 exhausts the search in one parent, then goes up the class hierarchy for each class from which the object inherits (from left to right, as appears in the class definition). In

previous releases, MATLAB actually performed a modified breadth-first search.

In this hierarchy:



If function `foo` is called:

- In MATLAB 5.2, Class X's `foo` would be invoked.
- In MATLAB 5.0 or 5.1, Class Z's `foo` would be invoked.

API Memory Management Compatibility Issue

To address performance issues, the internal MATLAB memory management model has been changed somewhat. These changes support future enhancements to the MEX-file API.

With this release, MATLAB now implicitly calls `mxDestroyArray`, the `mxArray` destructor, at the end of a MEX-file's execution on any `mxArrays` that are not returned in the left-hand side list (`plhs[]`). You are now warned if MATLAB detects any misconstructured or improperly destroyed `mxArrays`.

We highly recommend that you fix code in your MEX-files that produces any of the warnings discussed in the following sections. For additional information, see the "Memory Management" section in Chapter 3 of the *Application Program Interface Guide*.

The rest of this discussion describes situations in which you would receive such warning messages. The discussion of each situation includes an example and a solution.

Improperly Destroying an mxArray

You cannot use `mxFree` to destroy an mxArray.

Warning

Warning: You are attempting to call `mxFree` on a <class-id> array. The destructor for mxArrays is `mxDestroyArray`; please call this instead. MATLAB will attempt to fix the problem and continue, but this will result in memory faults in future releases.

Note: In MATLAB 5.2, these warnings are enabled by default for compatibility reasons. They can be disabled with the command

```
feature('MEXFileCompat', 0)
```

Disabling the code that detects and fixes these error conditions may slightly improve performance, but will cause serious problems if your MEX-file actually causes any of these errors.

Example That Causes Warning

```
mxArray *temp = mxCreateDoubleMatrix(1, 1, mxREAL);  
...  
mxFree(temp); /* INCORRECT */
```

`mxFree` does not destroy the array object. This operation frees the structure header associated with the array, but MATLAB will still operate as if the array object needs to be destroyed. Thus MATLAB will try to destroy the array object, and in the process, attempt to free its structure header again.

Solution

Call `mxDestroyArray` instead:

```
mxDestroyArray(temp); /* CORRECT */
```

Incorrectly Constructing a Cell or Structure mxArray

You cannot call `mxSetCell` or `mxSetField` variants with `prhs[]` as the member array.

Warning

Warning: You are attempting to use an array from another scope (most likely an input argument) as a member of a cell array or structure. You need to make a copy of the array first. MATLAB will attempt to fix the problem and continue, but this will result in memory faults in future releases.

Example That Causes Warning

```
>> myfunction('hello')
/* myfunction is the name of your MEX-file and your code */
/* contains the following: */

    mxArray *temp = mxCreateCellMatrix(1,1);
    ...
    mxSetCell(temp, 0, prhs[0]); /* INCORRECT */
```

When the MEX-file returns, MATLAB will destroy the entire cell array. Since this includes the members of the cell, this will implicitly destroy the MEX-file's input arguments. This can cause several strange results, generally having to do with the corruption of the caller's workspace, if the right-hand side argument used is a temporary array (i.e., a literal or the result of an expression).

Solution

Make a copy of the right-hand side argument with `mxDuplicateArray` and use that copy as the argument to `mxSetCell` (or `mxSetField` variants); for example:

```
mxSetCell(temp, 0, mxDuplicateArray(prhs[0])); /* CORRECT */
```

Creating a Temporary mxArray with Improper Data

You cannot call `mxDestroyArray` on an mxArray whose data was not allocated by an API routine.

Warning

Warning: You have attempted to point the data of an array to a block of memory not allocated through the MATLAB API. MATLAB will attempt to fix the problem and continue, but this will result in memory faults in future releases.

Example That Causes Warning

If you call `mxSetPr`, `mxSetPi`, `mxSetData`, or `mxSetImagData` with memory as the intended data block (second argument) that was not allocated by `mxMalloc`, `mxMalloc`, or `mxRealloc`, as shown below:

```

mxArray *temp = mxCreateDoubleMatrix(0, 0, mxREAL);
double data[5] = {1, 2, 3, 4, 5};
...
mxSetM(temp, 1); mxSetN(temp, 5); mxSetPr(temp, data);
/* INCORRECT */

```

then when the MEX-file returns, MATLAB will attempt to free the pointer to real data and the pointer to imaginary data (if any). Thus MATLAB will attempt to free memory, in this example, from the program stack. This will cause the above warning when MATLAB attempts to reconcile its consistency checking information.

Solution

Rather than use `mxSetPr` to set the data pointer, instead create the `mxArray` with the right size and use `memcpy` to copy the stack data into the buffer returned by `mxGetPr`:

```

mxArray *temp = mxCreateDoubleMatrix(1, 5, mxREAL);
double data[5] = {1, 2, 3, 4, 5};
...
memcpy(mxGetPr(temp), data, 5*sizeof(double)); /* CORRECT */

```

Potential Memory Leaks

Prior to Version 5.2, if you created an `mxArray` using one of the API creation routines and then you overwrote the pointer to the data using `mxSetPr`, MATLAB would still free the original memory. This is no longer the case.

For example:

```

pr = mxMalloc(5*5, sizeof(double));
... <load data into pr>
plhs[0] = mxCreateDoubleMatrix(5, 5, mxREAL);
mxSetPr(plhs[0], pr); /* INCORRECT */

```

will now leak $5*5*8$ bytes of memory, where 8 bytes is the size of a `double`.

You can avoid that memory leak by changing the code:

```
plhs[0] = mxCreateDoubleMatrix(5, 5, mxREAL);  
pr = mxGetPr(plhs[0]);  
... <load data into pr>
```

or alternatively:

```
pr = mxMalloc(5*5, sizeof(double));  
... <load data into pr>  
plhs[0] = mxCreateDoubleMatrix(5, 5, mxREAL);  
mxFree(mxGetPr(plhs[0]));  
mxSetPr(plhs[0], pr);
```

Note that the first solution is more efficient.

Similar memory leaks can also occur when using `mxSetPi`, `mxSetData`, `mxSetImagData`, `mxSetIr`, or `mxSetJc`. You can address this issue as shown above to avoid such memory leaks.

Recommendation: MEX-Files Should Destroy Their Own Temporary Arrays

In general, we recommend that MEX-files destroy their own temporary arrays and clean up their own temporary memory. All inconsistent `mxArrays` except those returned in the left-hand side list and the return from the `mxGetArrayPtr` may be safely destructed. This approach is consistent with other MATLAB API applications (i.e., MAT-file applications, Engine applications, and MATLAB Compiler generated applications).

Upgrading from MATLAB 5.0 to MATLAB 5.2

This table describes some changes you can make to your code to eliminate error messages and warnings due to incompatible and noncompliant statements in MATLAB 5.0 code that you are upgrading to MATLAB 5.2.

Note: If you are upgrading from MATLAB 5.0 to MATLAB 5.2, in addition to this section, you should read the previous section, called “Upgrading from MATLAB 5.1 to MATLAB 5.2.”

Table 4-1: MATLAB 5.1 Language Changes

Function	Change	Action
<code>find</code>	<code>find</code> was modified for sparse row vectors. <code>find(sparse_row)</code> was a column in MATLAB 4 and MATLAB 5.0. In 5.1 it produces a row when the input is a row. All other cases still return columns.	Update code.
<code>lasterr</code>	In MATLAB 5.1, <code>lasterr</code> doesn't contain the ??? that prints out when you get an error. Some types of errors in MATLAB 5.0 erroneously contained ???.	None required.
<code>plot</code>	In MATLAB 5.0, <code>plot</code> incorrectly accepted arrays with more than two dimensions, but treated them as 2-D arrays. In 5.1, this causes an error.	Do not use arrays of more than two dimensions as arguments for <code>plot</code> .

Table 4-1: MATLAB 5.1 Language Changes (Continued)

Function	Change	Action
Set functions: intersect, setdiff, setxor, union, unique	These functions now error out if the inputs aren't vectors, and you aren't using the rows flag.	Update code if required.
size	An empty string created within a MEX-file is now of size 0,0, consistent with the rest of MATLAB. (This change occurred with MATLAB 5.1.)	Update code accordingly.

Upgrading from MATLAB 4 to MATLAB 5.2

MATLAB 5.0 was a major upgrade to MATLAB 4. Although The MathWorks endeavors to maintain full upwards compatibility between subsequent releases of MATLAB, inevitably there are situations where this is not possible. In the case of MATLAB 5.0, there were a number of changes that you need to know about in order to migrate your code from MATLAB 4 to MATLAB 5.0.

Note: In addition to this section, you should read the rest of this chapter as well if you are upgrading from MATLAB 4.0 to MATLAB 5.2.

Converting M-Files to MATLAB 5.0

This section describes some changes you can make to your code to eliminate error messages and warnings due to incompatible and noncompliant statements.

Table 4-2: MATLAB 5.0 Language Changes

Function	Change	Action
auread, aawrite	New syntax.	Change call to use new syntax.
bessel functions	The bessel functions no longer produce a table for vector arguments of the same orientation.	For example, in <code>besselj(nu, x)</code> , specify <code>nu</code> as a row and <code>x</code> as a column to produce a table.
case, otherwise, switch	<code>case</code> , <code>otherwise</code> , and <code>switch</code> cannot be used as variable names.	Rename your variables.
dialog	<code>dialog.m</code> now creates a modal dialog.	Use the <code>msgbox</code> function instead.
echo	<code>echo</code> does not display multiline matrices.	Update code.

Table 4-2: MATLAB 5.0 Language Changes (Continued)

Function	Change	Action
end	extra end statements.	Remove redundant end statements.
eps	eps is a function. eps = 0 no longer redefines eps for other functions (it makes a local variable called eps in the current workspace). Functions that base their tolerance on an externally defined eps won't work.	Change code accordingly.
for	for loop variable different after loop for empty loops. In MATLAB 4: i = 10; for i = 1:0, %goes nowhere end i produces i = 10. In MATLAB 5.0 it produces i = [].	Protect the for loop with an i empty call: i = 10; if ~i empty(n) for i=1:n end end i
global	Undefined globals	Define globals before they are used. Always put the global statement at the top of the M-file (just below the help comments). MATLAB 4 produced a link in the workspace to an uninitialized global. It shows up in whos but exist returns 0. Do not use exist to test for the first time the global has been accessed. Use isempty.
gradient	gradient no longer produces complex output.	Use two outputs in the 2-D case.

Table 4-2: MATLAB 5.0 Language Changes (Continued)

Function	Change	Action
<code>i nput</code>	<code>i nput(' prompt', ' s')</code> no longer outputs an initial line feed. Prompts now show up on the same line.	Update code accordingly if this causes a display problem. Add <code>\n</code> in the prompt string to force a line feed.
<code>i nterp1</code>	The old <code>i nterp1</code> syntax (<code>i nterp1(x, n)</code>) no longer calls <code>i nterpft</code> . A warning was in place in MATLAB 4.	Update code accordingly.
	<code>i nterp1</code> now returns a row vector when given a row vector. It used to return a column vector.	Transpose the output of <code>i nterp1</code> to produce the MATLAB 4 result when <code>xi</code> is a row vector.
	<code>i nterp1(' spl i ne')</code> returns NaN's for out of range values.	Use <code>spl i ne</code> directly.
<code>i nterp2</code>	The old <code>i nterp2</code> syntax (<code>i nterp2(x, y, xi)</code>) no longer calls <code>i nterp1</code> . A warning was in place in MATLAB 4.	Update code accordingly.
<code>i nterp3</code>	The old <code>i nterp3</code> syntax (<code>i nterp3(z, m, n)</code> or <code>i nterp3(x, y, z, xi, yi)</code>) no longer calls <code>gri ddat a</code> . A warning was in place in MATLAB 4. <code>i nterp3</code> is now 3-D interpolation.	Update code accordingly.
Automeshing interpolation commands	Interpolation automeshing has changed. <code>gri ddat a</code> , <code>i nterp2</code> , <code>i nterp3</code> , <code>i nterpn</code> , and <code>bessel *</code> now automesh if <code>(xi, yi)</code> or <code>(nu, z)</code> are vectors of different orientations. Previously they automeshed if the vectors were different size.	When <code>xi</code> and <code>yi</code> are vectors of the same orientation but different lengths, change calls such as <code>i nterp2(..., xi, yi)</code> to <code>i nterp2(..., xi, yi')</code> .

Table 4-2: MATLAB 5.0 Language Changes (Continued)

Function	Change	Action
<code>isempty</code>	<code>A == []</code> and <code>A ~= []</code> as a check for an empty matrix produce warning messages.	Use <code>isempty(A)</code> or <code>~isempty(A)</code> . In a future version <code>A == []</code> will produce an empty result.
<code>isspace</code>	<code>isspace</code> only returns true (1) on strings. <code>isspace(32)</code> is 0 (it was 1 in MATLAB 4).	Wrap your calls to <code>isspace</code> with <code>char</code> .
<code>logical</code>	Some masking operations where the mask isn't defined using a logical expression now produce an out of range index error.	Wrap the subscript with a call to <code>logical</code> or use the logical expression <code>A~=0</code> to produce MATLAB 4 behavior.
	Boolean indexing is no longer directly supported.	Use <code>logical</code> to create the index array.
<code>matlabrc</code>	On the PC, MATLAB no longer stores the path in <code>matlabrc</code> .	All platforms except the Macintosh now use <code>pathdef.m</code> . The Macintosh still stores its path in the MATLAB <code>Settings</code> file (usually in the Preferences folder).
<code>max</code>	<code>max(size(v))</code> , as a means to determine the number of elements in a vector <code>v</code> , fails when <code>v</code> is empty. <code>max</code> ignores NaNs.	Use <code>length(v)</code> in place of <code>max(size(v))</code> .
<code>min</code>	<code>min</code> ignores NaNs.	Change code if necessary.

Table 4-2: MATLAB 5.0 Language Changes (Continued)

Function	Change	Action
nargi n,nargout	nargi n and nargout are functions.	nargout = nargout-1 (and any similar construction) is an error. To work around this change, assign nargi n to a local variable and increment that variable. Rename all occurrences of nargi n to the new variable. The same holds true for all functions.
ones	A(ones(size(A))) no longer produces A.	This statement produces copies of the first element of A. Use A(ones(size(A))~=0) or just A to produce the MATLAB 4 behavior.
	No longer accepts column vector.	The size vector must be a row vector with integer elements.
	Functions such as ones, eye, rand, and zeros give an error if supplied with a matrix argument (such as zeros(A)).	Use the syntax ones(size(A)) instead.
polyfi t	Second output now a structure.	Change code to access structure component.
pri nt	-ocmyk is now -cmyk.	Update code accordingly.
	-psdefcset is now -adobecset.	Update code accordingly.
	GIF format no longer supported.	Use alternate format.
	Texture mapped surfaces do not print with painter's algorithm.	Use -zbuffer.
rand	rand(' normal ') and rand(' uni form ') no longer supported.	Use randn for normally distributed and rand for uniformly distributed random numbers.

Table 4-2: MATLAB 5.0 Language Changes (Continued)

Function	Change	Action
round	Subscripts must be integers.	To reproduce MATLAB 4 behavior, wrap noninteger subscripts with round(). Strings are no longer valid subscripts (since they are not integers in the strict sense).
slice	slice no longer requires the number of columns (ncols) argument.	Update code accordingly.
sound	Doesn't autoscale.	Use soundsc.
strcmp strncmp	strcmp and strncmp now return false (0) when any argument is numeric. They used to perform an isequal.	Call isequal for all nonstrings you want to compare.
wavread, wavwrite	New syntax.	Change call to use new syntax.
zeros	No longer accepts column vector.	Size vector must be a row vector with integer elements.

Note: The following language changes do not directly apply to specific functions.

	a(:) = b where a doesn't exist creates an error. This used to do the same thing as a = b(:) when a didn't exist.	Either initialize a or use a = b(:) instead.
	Must use an explicitly empty matrix to delete elements of an array, as in a(i) = [] or a(i, :) = []. This syntax works for all built-in data types (including cell arrays and structures).	Change code accordingly.

Table 4-2: MATLAB 5.0 Language Changes (Continued)

Function	Change	Action
	The syntax <code>a(i) = B</code> , when B is empty, no longer deletes elements.	Use <code>a(i) = []</code> instead.
	An attempt to delete elements of an array outside its range is no longer (incorrectly) ignored. An error is generated.	Change code accordingly.
	Undefined variables.	To reproduce MATLAB 4 behavior, initialize your variable to the empty matrix (<code>[]</code>) or empty string (<code>''</code>).
	Undefined outputs.	To reproduce MATLAB 4 behavior, initialize your outputs to the empty matrix (<code>[]</code>).
	Indices must be integers. Strings are no longer valid indices.	Use <code>a(round(ind))</code> to get MATLAB 4 behavior.
	<code>_</code> , <code>^</code> , <code>{</code> , and <code>}</code> are now interpreted, not displayed.	Use <code>_</code> , <code>\^</code> , <code>\{</code> , and <code>\}</code> .
	Concatenating a string and a double truncates the double.	Use <code>double</code> to convert the string before concatenating.
	Input arguments are no longer evaluated left to right.	Evaluate input arguments before passing them to a function.
	String handling difference. In MATLAB 4 <code>a = 32*ones(1, 10);</code> <code>a(1:5) = 'hello'</code> produces 'hello'. In MATLAB 5.1, it produces: 104 101 108 11 32 32 32 32 32.	Initialize <code>a</code> to be a character array or convert it after assignment.

Table 4-2: MATLAB 5.0 Language Changes (Continued)

Function	Change	Action
	Using inline matrix constants and continued matrix constants inside function calls: <pre>fun(arg1, [1 2 3 3 4 5, 5 6 6])</pre> is a syntax error.	Put continuation dots and semicolon after each matrix line.

Table 4-3: MATLAB 5.0 Obsolete Language Functions

Obsolete Function	Action
casesen	Remove the call.
csvread, csvwrite	Use <code>dlmread(filename, ',')</code> and <code>dlmwrite(filename, ',')</code> .
ellipk	Replace with <code>ellipke</code> .
extent	Replace with Extent property.
figflag	Use <code>findobj</code> .
finite	Rename to <code>isfinite</code> . <code>finite</code> will continue to work in MATLAB 5.2 but will probably be removed in a later release.
fwhich	Use <code>which</code> .
hthelp	<code>hthelp</code> works in MATLAB 5.2, but will not be further developed or supported. Use <code>helpwin</code> .
htpp	Use <code>helpwin</code> .
inquire	Use <code>set</code> and <code>get</code> to obtain the current state of an object or of MATLAB.
inverf	Rename to <code>erfinv</code> .
isdir	Use <code>dir</code> .

Table 4-3: MATLAB 5.0 Obsolete Language Functions (Continued)

Obsolete Function	Action
layout	No replacement in MATLAB 5.2.
loadhtml	Use helpwin or doc.
matq2ws	Replace with assign and evalin.
matqdlg	Replace with assign and evalin.
matqparse	Replace with assign and evalin.
matqueue	Replace with assign and evalin.
menulabel	Bug in Handle Graphics is now fixed.
mexdebug	Rename to dbmex.
ode23p	Use ode23 with no left-hand arguments or set an output function with odeset.
polyl ine, polymark	Use the line object or plot.
printmenu	No replacement in MATLAB 5.2.
saxiss	Use soundsc.
ws2matq	Replaced by assign and evalin.

Table 4-4: MATLAB 5.0 Graphics Function Changes

Function	Change	Action
figure	In MATLAB 4 if a figure extended past the top of the window, it was adjusted to be visible. MATLAB 5.0 performs no adjustment.	Avoid hardcoded Figure positions. Unless you know the screen size, it is better to position the Figure by first querying the Root ScreenSize property.
get	get(h, 'currentfigure') and get(h, 'currentaxes') no longer create a Figure or an Axes if one doesn't exist. They return [] in that case.	gcf and gca always return a valid handle. Use gcf and gca instead of the get function in this context.
	In MATLAB 4 you could determine if a graphics object had a default value set by passing its handle in a query like get(gca, 'DefaultAxesColor').	In MATLAB 5.0 make the query on the object's ancestor, e.g., get(gcf, 'DefaultAxesColor') or get(0, 'DefaultAxesColor')
plot	MATLAB 4 plots may have elements that are the wrong color. MATLAB 5.0 defaults to a white background on all platforms. (MATLAB 4 defaulted to white background on the Macintosh and black everywhere else.)	Use colordef to control your color defaults. Typically, you will put a call to colordef in startup.m. To get the MATLAB 4 defaults, use colordef none.
	plot line styles c1 through c15 and i are no longer supported	Use a 1-by-3 RGB ColorSpec instead. i is the same as get(gca, 'color') or get(gcf, 'color') when the Axes color is 'none'.
rotate	rotate alpha is reversed from MATLAB 4.	If your call looked like rotate(h, [theta phi], alpha), change to rotate(h, [theta phi], -alpha[0 0 0]).

Table 4-4: MATLAB 5.0 Graphics Function Changes (Continued)

Function	Change	Action
text	text(S) when S is a multirow character array formerly produced one handle per row. Now it produces one multiline text handle.	Rewrite code so that it doesn't assume a specific number of handles.
ui control	The default Uicontrol text horizontal alignment is centered in MATLAB 5.0. (MATLAB 4 used to left align text and ignore the alignment property.)	Explicitly set the horizontal alignment when you create Uicontrol Text objects.
	In MATLAB 4, Uicontrols of style 'edit' executed their callback routine whenever you moved the pointer out of the edit box. In MATLAB 5.0, edit controls execute their callbacks after you perform a specific action.	The callback is called when: <ul style="list-style-type: none"> • Return key is pressed (single-line edits only) • Focus is moved out of the edit by: <ul style="list-style-type: none"> -Clicking elsewhere in the Figure (on another Uicontrol or on another graphical object) -Clicking in another Figure -Clicking on the menu bar (X Window systems only)
Note: The following change does not directly apply to a specific function.		
	MATLAB 5.0 sets font size selection to match platform conventions. A MATLAB 4 font selection may be a different size in MATLAB 5.0.	Resize font appropriately.

Table 4-5: MATLAB 5.0 Graphics Property Changes

Property	Object Type	Change	Action
AspectRati o	Axes	Obsolete	Replace with DataAspectRati o and Pl otBoxAspectRati o.
BackgroundCol or	Uimenu	Obsolete	Do not use.
CurrentMenu	Figure	Becoming obsolete. No warning message produced.	Replace with the function gcbo.
EraseMode	Image, Line, Patch, Surface, Text	Now use xor against the Axes color rather than the Figure color. For non-normal erasemode, MATLAB recomputes Axes limits only when you fully update the display (e.g., with a drawnow command).	Modify code as appropriate. Set the Axes limits (and other properties you depend upon) before using non-normal modes to create animation.
ExpFontAngl e	Axes, Text	Obsolete	Do not use.
ExpFontName	Axes, Text	Obsolete	Do not use.
ExpFontSi ze	Axes, Text	Obsolete	Do not use.
ExpFontSt ri keThrough	Axes, Text	Obsolete	Do not use.
ExpFontUnderl i ne	Axes, Text	Obsolete	Do not use.
ExpFontUni ts	Axes, Text	Obsolete	Do not use.
ExpFontWei ght	Axes, Text	Obsolete	Do not use.
FontSt ri keThrough	Axes, Text	Obsolete	Do not use.
FontUnderl i ne	Axes, Text	Obsolete	Do not use.

Table 4-5: MATLAB 5.0 Graphics Property Changes (Continued)

Property	Object Type	Change	Action
LineStyle	Line, Patch, Surface	<p>Setting the LineStyle property to a marker value (such as '+') now produces a warning.</p> <p>Setting the marker style of a line now affects the Marker property instead of the LineStyle property. Although you will be able to set a line marker using the LineStyle property (with a warning), you will not be able to get marker style information from LineStyle.</p>	<p>Set the Marker property instead. Note that plot will continue to take line-color marker line styles.</p> <p>If your code relies on markers in the LineStyle, you'll have to change it to use the Marker instead.</p>
NextPlot	Axes, Figure	Use of value 'new' is obsolete. Produces warning message.	Use HandleVisibility to protect user interfaces from command line users.
PaletteModel	Image, Patch, Surface	Property renamed and value of bypass removed	Use CDatamapping
RenderLimits	Axes	Obsolete	Do not use. Limits are now always accurate.
SelectionType	Figure	Right mouse button went from Extended in MATLAB 4 to Alternate in MATLAB 5.0.	None required.

Table 4-5: MATLAB 5.0 Graphics Property Changes (Continued)

Property	Object Type	Change	Action
Units	Axes, Figure, Text, Uicontrol	Units/Position is always order dependent for all objects. In MATLAB 4, it was inconsistent.	The Units property should precede any properties that depend upon it. A command such as <code>axes('position', [100 200 300 100], 'units', 'pixels')</code> is not the same as <code>axes('units', 'pixels', 'position', [100 200 300 100])</code> . In the first case the numbers are interpreted in normalized coordinates.
WindowID	Figure	Possibly becoming obsolete.	May be removed in a future release.
XLim, XTick	Axes	Values must be monotonically increasing.	Sort the ticks: <code>set(gca, 'xtick', sort([3 2 1]))</code>
XTickLabels	Axes	Renamed	Use XTickLabel
YLim, YTick	Axes	Values must be monotonically increasing.	Sort the ticks: <code>set(gca, 'ytick', sort([3 2 1]))</code>
YTickLabels	Axes	Renamed	Use YTickLabel

Table 4-5: MATLAB 5.0 Graphics Property Changes (Continued)

Property	Object Type	Change	Action
ZLim, ZTick	Axes	Values must monotonically increase.	Sort the ticks: <code>set(gca, 'ztick', sort([3 2 1]))</code>
ZTickLabels	Axes	Renamed	ZTickLabel

Converting MATLAB 4 External Interface Programs to the MATLAB 5.0 Application Program Interface

MATLAB 4 External Interface programs, including MEX-files, MAT-file programs, and Engine programs may run without any modification on the MATLAB 5.0 Application Program Interface (API), or they may require modification and/or recompilation. The following pages and flowcharts describe how to determine which of these possibilities applies in your situation and how to choose the appropriate conversion technique.

General Considerations

Non-ANSI C Compilers. MATLAB 4 let you compile External Interface programs with non-ANSI C compilers. MATLAB 5.0 API header files include strict prototyping of API functions and require an ANSI C compiler.

MATLAB Character Strings. MATLAB 4 and MATLAB 5.0 represent string data in different ways. MATLAB 4 supported only one data type. All data was represented as double-precision, floating-point numbers, even individual characters in a string array. A numerical array and a character array differed only by how MATLAB displayed these values. MATLAB 5.0 represents each character in a string array as an `mxChar`, a 16-bit unsigned integer data type. If the `mxArray`'s class is `mxCHAR_CLASS`, the API treats each number in the `mxArray` as an element from the current character set. Character sets are platform specific.

External Interface programs that call the API routines `mxGetString()` and `mxCreateString()` to manipulate strings continue to work. All MEX-files that directly manipulate strings must be rewritten. MAT-file and Engine applications that directly manipulate strings need not be rewritten. However, to recompile these applications under MATLAB 5.0, any code that directly

manipulates strings must be rewritten. We highly recommend that you use the API string access and creation routines to do this. To help in this endeavor, you can use the new C routine, `mxCreateCharMatrixFromStrings()`, in addition to `mxGetString()` and `mxCreateString()`, to make it easy to create two-dimensional string matrices.

MEX-File Argument Complexification. In MATLAB 4, if one argument to a MEX-function was complex, all arguments were passed as complex. This is not true in MATLAB 5.0. For example, consider a MEX-function, `myeig(A, B, C)`, that calculates eigenvalues of three matrices. In MATLAB 4, if matrix A is complex, B and C are assumed to be complex matrices as well. In this instance, additional memory is allocated for the complex part of B and C, and initialized with zero values.

MATLAB 5.0 does not allocate this memory for you. If your MEX-file assumes argument complexification, you will have to rewrite your MEX-file. Each argument to a MEX-function needs to be tested with `mxIsComplex()` to guarantee that an argument indeed has a complex component.

Type Imputation by Process of Elimination. In MATLAB 4 the only way to determine if a matrix was a full, nonstring matrix was by a process of elimination. For example, if your code checked a variable and found that it was a full matrix (nonsparse), and was not a string, you could also assume that the variable was double-precision, floating-point and two-dimensional. In MATLAB 5.0, with the addition of several new data types and support for multidimensional data, this assumption is no longer valid. `mxIs*` routines and `mxGetNumberOfDimensions()` have been added to the C interface so that now you can explicitly check arguments for specific data types and shapes. `mxIsDouble()`, which always returned 1 in MATLAB 4, now correctly returns 1 only if the `mxArray` is of type double precision floating point. `mxIsDouble()` is available in C as well as Fortran. `mxGetN()` returns the total number of elements in dimensions 2-n, and therefore works correctly with multidimensional as well as two-dimensional arguments.

Version 3.5 MEX-Files. MEX-files generated under MATLAB 4 using the `-v3.5` switch to the `cmex` or `fmex` script are no longer supported and must be rewritten. Refer to both the MATLAB 4 *External Interface Guide* for information on upgrading to the MATLAB 4 syntax and the section later in this document on recoding for MATLAB 5.0 compliance.

Simulink. By design, Simulink S-functions written in C must be recompiled under the latest version of Simulink. The code is source compatible, but the binary must be recompiled. If you attempt to run a Simulink 1.3 C S-function under Simulink 2.1, you get an appropriate warning message. Simulink S-functions written in Fortran do not have this restriction.

Fortran MEX-File Considerations. The MATLAB Fortran API has not changed from MATLAB 4 to MATLAB 5.0. However, Fortran mex users on Windows, 68K Macintosh and the Sun4 must recompile their applications using the mex script for programs that call `mxCreateString()` or `mxGetString()`. Only these platforms, which statically link in the Fortran interface, are affected. No recompilation is required on other platforms.

Rebuilding MEX-Files Loaded in Memory. In MATLAB 4, on Windows and UNIX, you could execute `cmex` or `fmex` from within MATLAB by using the `!` command or execute in another window. These methods are no longer supported with MATLAB 5.0. However, the M-file that builds MEX-files, `mex.m`, is available on all platforms and can safely rebuild loaded MEX-files by unloading them before proceeding with the build. The interface provided by the M-file is identical to the external mex script.

Default MEX-File Optimization. MATLAB 4 MEX-files, by default, were built with no optimization flags passed to the compiler. In MATLAB 5.0 the default is to optimize MEX-files. See the *MATLAB Application Program Interface Guide* for more information.

Debugging MEX-Files. The `-debug` switch to `cmex/fmex` is no longer supported. Instead, on all platforms, MATLAB 5.0 supports MEX-file debugging from within a MATLAB session. (See the debugging sections of the *MATLAB Application Program Interface Guide* for more details). In addition, the mex script has been enhanced with the `-argcheck` switch. That switch provides a way for C mex users to generate code to check input and output arguments at runtime and issue appropriate error messages if invalid data is detected.

MAT-file External Applications. MAT-file external applications built under MATLAB 4 will continue to work under MATLAB 5.0. Note that the MAT-file format has changed between MATLAB 4 and MATLAB 5.0. Although MATLAB will be able to read MATLAB 4 MAT-files generated by a stand-alone program, stand-alone programs will not be able to read MAT-files in a MATLAB 5.0 format. You can generate MATLAB 4 MAT-files from MATLAB 5.0 by specifically passing the `-v4` switch to the `save` command.

Microsoft Windows Considerations

MEX-files are generated under MATLAB 5.0 as 32-bit DLLs. The `cmex` and `fmex` batch files have been superseded by `mex` (a PERL script) and a set of compiler-specific option files. These compilers are fully supported for creating MEX or stand-alone applications through MathWorks-supplied option files:

- Watcom C
- Microsoft Visual C++
- Borland C

Microsoft Fortran, currently the only supported Fortran compiler, is supported for MEX applications only. You will not be able simply to recompile your MATLAB 4 Fortran MEX-file source with this new compiler. See “Creating Fortran MEX-files” in the *MATLAB Application Program Interface Guide* for further information.

NDP Fortran, previously supported under MATLAB 4, is not supported in this release.

16-bit DLL MEX-files are no longer supported and cannot be generated. Existing MATLAB 4 REX MEX-files are usable but cannot be created under MATLAB 5.0.

See “Fortran MEX-file Considerations” above for Windows-specific restrictions on creating and accessing MATLAB character strings.

MATLAB 4 C language Engine binaries will not run with MATLAB 5.0. Programs must be recompiled. MATLAB 5.0 data types are currently not supported on the PC from an Engine program. There is currently no support for Fortran Engine or MAT-file programs.

See the *MATLAB Application Program Interface Guide* for instructions on compiling stand-alone programs in MATLAB 5.0.

UNIX Considerations

The `cmex` and `fmex` Bourne shell scripts for building MEX-files have been superseded by `mex`, also a Bourne shell script, that sources an options file, `mexopts.sh`, for all platform compiler-specific information. The options file contains all the pertinent compiler and linker switches for building ANSI C and Fortran MEX-file applications. The `.mexrc.sh` file is no longer supported and must be converted to the new format. The `mexdebug` MATLAB command that allows UNIX users to debug their MEX-files while MATLAB is running has been changed to `dbmex`. The behavior of `dbmex` under MATLAB 5.0 is identical to `mexdebug` under MATLAB 4.

See “Fortran MEX-file Considerations” above for Sun4-specific restrictions on creating and accessing MATLAB character strings.

You can use your existing UNIX Engine and MAT-file binary files unmodified; no recompilation is necessary. Note that MATLAB 4 Engine programs have no access to new MATLAB 5.0 data types. If you try to invoke `engGetMatrix(ep, my_variable)`, and `my_variable` is a cell array, structure array, object, etc., the operation automatically fails.

Macintosh Considerations

The Absoft MacFortran and Symantec Think C compilers are no longer supported, as of MATLAB 5.0.

MEX-files built under MATLAB 4 with MPW C on the 68K Macintosh that call the following I/O routines must be recompiled: `putc()`, `getc()`, `ferror()`, `feof()`, `clearerr()`. These exist as macros in MPW's `stdio.h`. In MATLAB 5.0, these functions have been reimplemented as function callbacks to MATLAB.

See “Fortran MEX-file Considerations” above for 68K Macintosh-specific restrictions on creating and accessing MATLAB character strings.

VMS Considerations

MEX-files compiled for MATLAB 4 on the VMS platform are not binary compatible. You must regenerate these files from source code before using them with MATLAB 5.0.

Conversion

Rebuilding MEX-Files. The simplest strategy for converting C MEX-file programs is to rebuild them with the special `-V4` option of `mex`. This option uses `mex` to define a macro `V4_COMPAT` that supports MATLAB 4 syntax and function calls. Therefore, any ANSI C MEX-file source code that compiled cleanly under MATLAB 4 should compile cleanly with the `-V4` option. The resulting MEX-file should run under MATLAB 5.0 just as it ran under MATLAB 4. For example, given C MEX-file MATLAB 4 source code in file `MyEi.g.c`, recompiling under UNIX with

```
mex -V4 myei.g.c
```

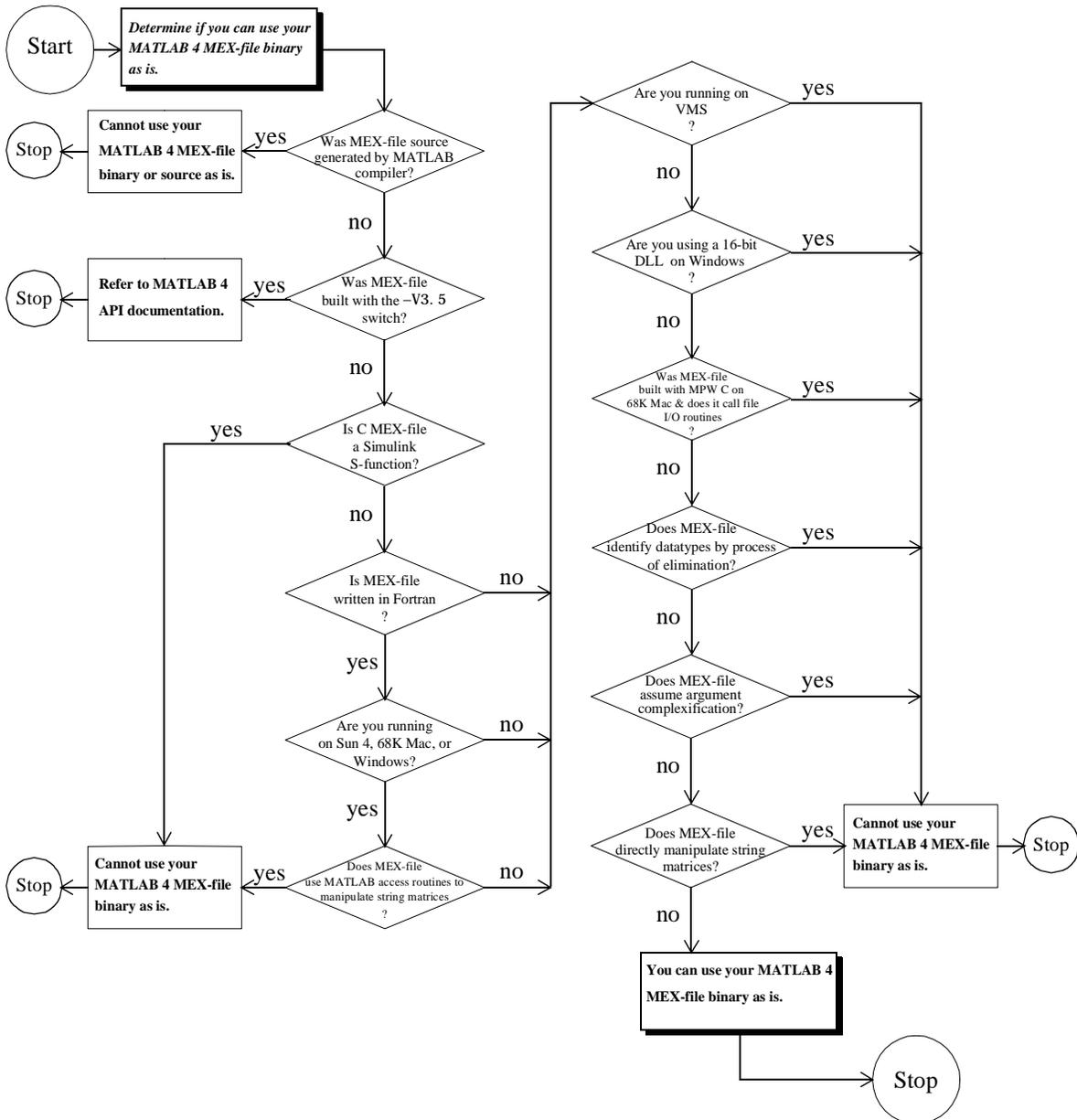
yields a MEX-file that MATLAB 5.0 can execute. It is also possible to use `cmex` and `fmex` for compiling C and Fortran source code, but both of these functions simply call `mex`.

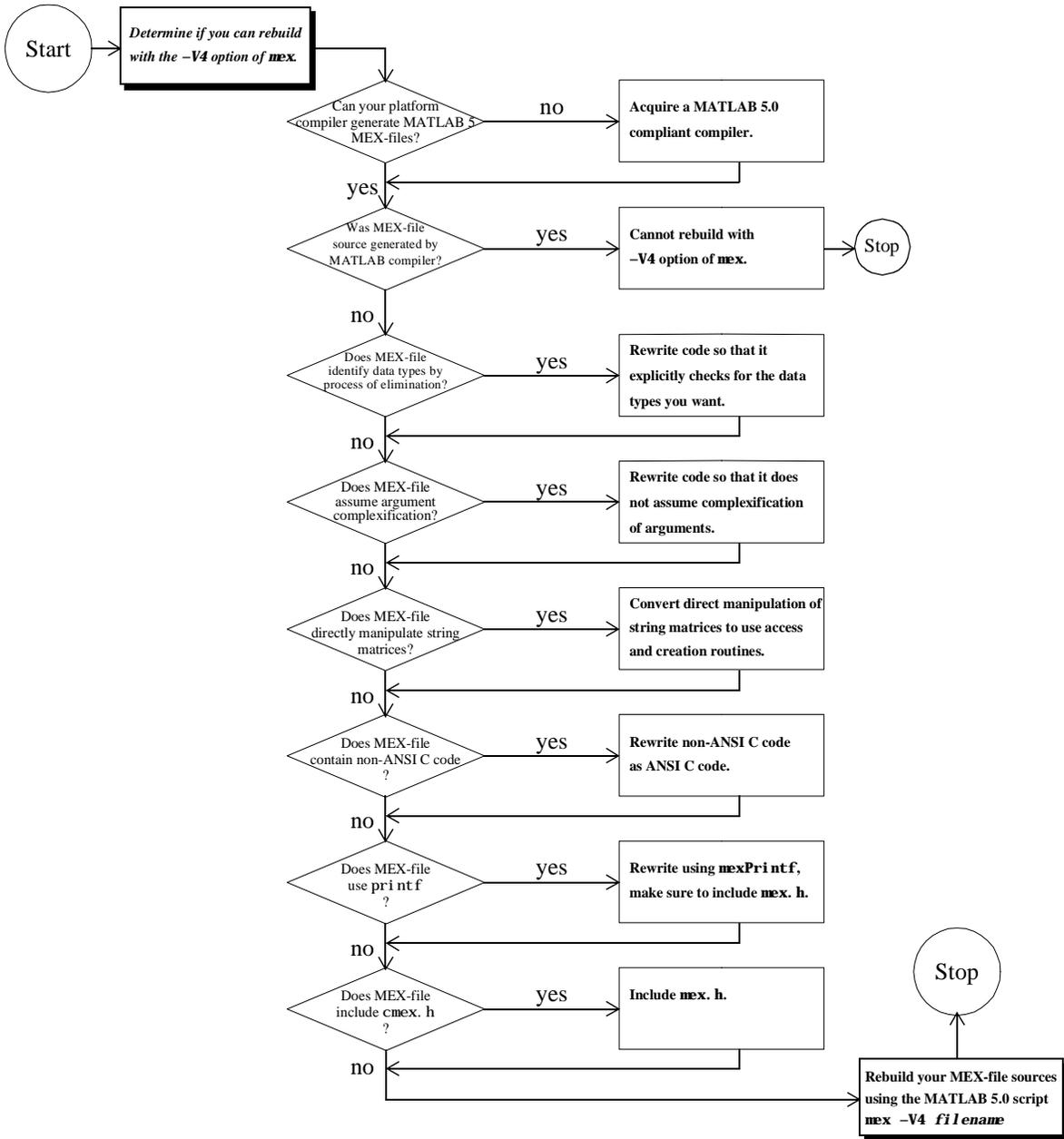
The obvious advantage to the `-V4` strategy is that it requires very little work on your part. However, this strategy provides only a temporary solution to the conversion problem; there is no guarantee that future releases of MATLAB will continue to support the `-V4` option. If you have the time, recoding for MATLAB 5.0 compliance is a better strategy. See “Recoding C Code for MATLAB 5.0 Compliance” below.

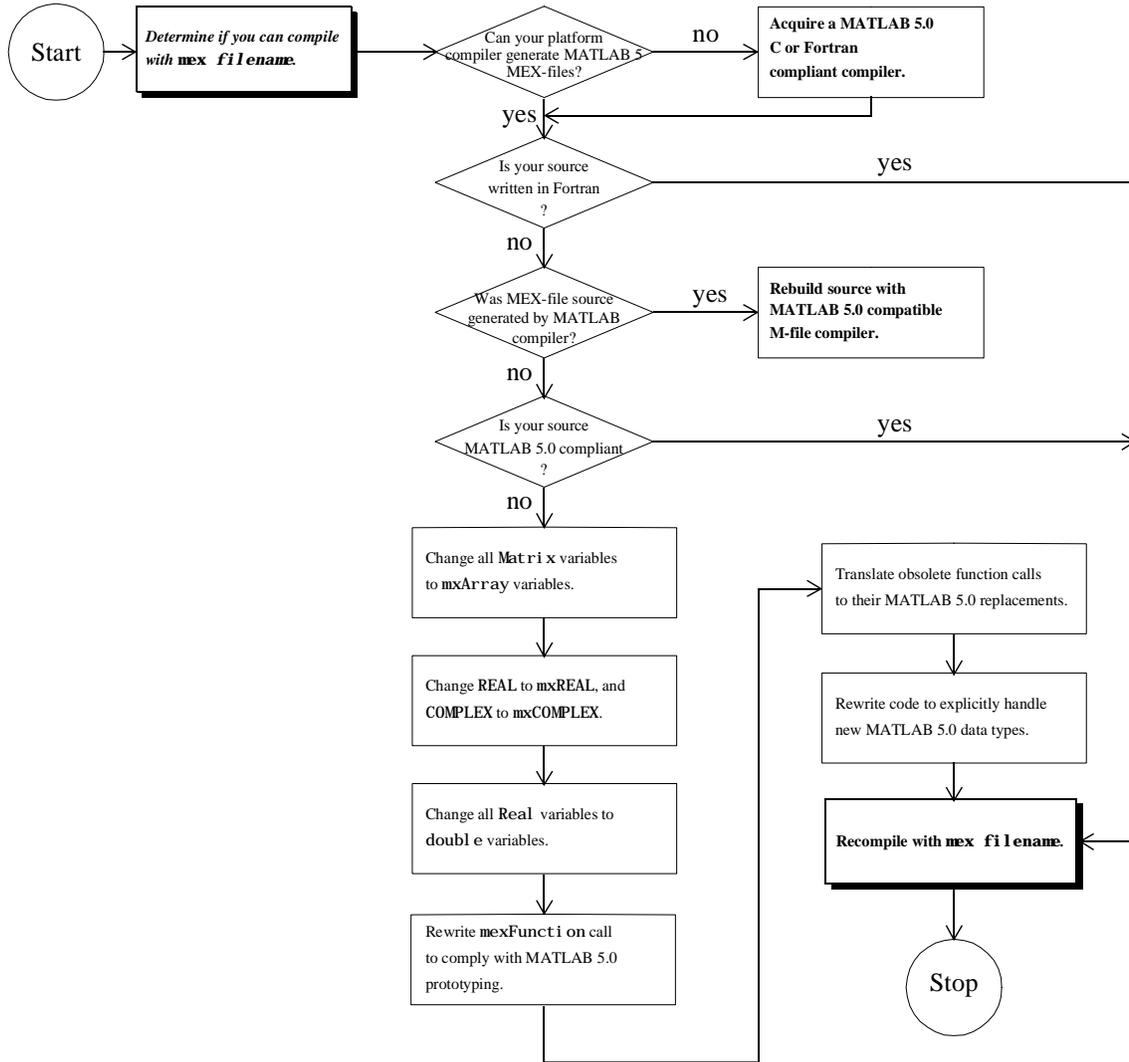
Rebuilding Stand-Alone MAT-File and Engine Programs. If your source code is ANSI compliant, you can recompile your source without modification by using the compiler flag `-DV4_COMPAT`. This allows you to avoid recoding, such as rewriting obsolete function calls as their MATLAB 5.0 equivalents. However, the resulting program will have the same restrictions as existing binary files.

See the *MATLAB Application Program Interface Guide* for instructions on compiling stand-alone programs in MATLAB 5.0.

MEX-File Conversion Flowcharts. The flowcharts below help you determine what steps you should take to run your MATLAB 4 MEX-files under MATLAB 5.0.







Recoding C Code for MATLAB 5.0 Compliance

Recoding your MATLAB 4 C code for MATLAB 5.0 compliance involves:

- Rewriting any non-ANSI C code as ANSI C code. (For details, see an ANSI C book.)
- Changing all `Matrix` variables to `mxArray` variables.

The MATLAB 4 `Matrix` data type is obsolete; you must change all `Matrix` variables to `mxArray` variables. For example, the `mxCreateSparse` function returns a `Matrix` pointer in MATLAB 4:

```
Matrix *MySparse;  
MySparse = mxCreateSparse(10, 10, 110, REAL);
```

To be MATLAB 5.0 compliant, change the code to:

```
mxArray *MySparse;  
MySparse = mxCreateSparse(10, 10, 110, mxREAL);
```

- Rewriting all function prototypes.

The function prototype of almost every MATLAB 4 function is different in MATLAB 5.0. The two primary prototype changes are

- All `Matrix` arguments are now `mxArray` arguments.
 - Pointers to read-only data are now declared as `const *`.
- For MEX-files, rewriting `mexFunction` to take a constant `mxArray *` as a fourth argument.
 - Changing `REAL` to `mxREAL` and `COMPLEX` to `mxCOMPLEX`.

In any function that requires the specification of real or complex data types, instead of `REAL` and `COMPLEX`, use `mxREAL` and `mxCOMPLEX`. For example, in MATLAB 4 you would write

```
mxCreateSparse(m, n, nzmax, REAL);
```

to create an m-by-n sparse matrix with `nzmax` nonzero real elements. In MATLAB 5.0, the correct syntax for this same function is:

```
mxCreateSparse(m, n, nzmax, mxREAL);
```

- Translating obsolete function calls into their MATLAB 5.0 replacements.
A number of functions have become obsolete. However, MATLAB 5.0 offers replacements for nearly all obsolete functions.

- Handling MATLAB 5.0 new data types.

You should explicitly check the data type of your input arguments to ensure that you have what you want.

Table 4-6 lists MATLAB 4 External Interface functions along with a description of how to recode those functions to work with MATLAB 5.0.

Table 4-6: Recoding MATLAB 4 Functions for MATLAB 5.0 Compliance

MATLAB 4 Function	MATLAB 5.0 Replacement
engGetMatrix	engGetArray
engGetFull	engGetArray followed by calls to the appropriate mx* routines
engPutMatrix	engPutArray
engPutFull	mxCreateDoubleMatrix followed by engPutArray
engSetEvalCallback	(Windows platform only.) Obsolete in 5.0.
engSetEvalTimeout	(Windows platform only.) Obsolete in 5.0.
engWinInit	(Windows platform only.) Obsolete in 5.0.
matGetMatrix	matGetArray
matGetNextMatrix	matGetNextArray
matGetFull	matGetArray followed by calls to the appropriate mx* routines
matPutMatrix	matPutArray
matPutFull	mxCreateDoubleMatrix followed by matPutArray
mexAtExit	No change
mexCallMATLAB	Second and fourth arguments are mxArray *
mexErrMsgTxt	No change
mexEvalString	No change

Table 4-6: Recoding MATLAB 4 Functions for MATLAB 5.0 Compliance

MATLAB 4 Function	MATLAB 5.0 Replacement
<code>mexFunction</code>	Second and fourth arguments are <code>mxArray *</code> Fourth argument is a <code>const</code>
<code>mexGetEps</code>	Obsolete; call <code>mxGetEps</code> instead
<code>mexGetFull</code>	Obsolete; call this sequence instead: <pre> mexGetArray(array_ptr, "caller"); name = mxGetName(array_ptr); m = mxGetM(array_ptr); n = mxGetM(array_ptr); pr = mxGetPr(array_ptr); pi = mxGetPi(array_ptr); </pre>
<code>mexGetGlobal</code>	Obsolete; call <code>mexGetArrayPtr</code> instead, setting the second argument to <code>"global"</code> . Note: it is better programming practice to call <code>mexGetArray("global")</code> ;
<code>mexGetInf</code>	Obsolete; call <code>mxGetInf</code> instead
<code>mexGetMatrix</code>	Call <code>mexGetArray(name, "caller")</code> ;
<code>mexGetMatrixPtr</code>	Call <code>mexGetArrayPtr(name, "caller")</code> ;
<code>mexGetNaN</code>	Obsolete; call <code>mxGetNaN</code> instead
<code>mexIsFinite</code>	Obsolete; call <code>mxIsFinite</code> instead
<code>mexIsInf</code>	Obsolete; call <code>mxIsInf</code> instead
<code>mexIsNaN</code>	Obsolete; call <code>mxIsNaN</code> instead
<code>mexPrintf</code>	No change

Table 4-6: Recoding MATLAB 4 Functions for MATLAB 5.0 Compliance

MATLAB 4 Function	MATLAB 5.0 Replacement
<code>mexPutFull</code>	<p>Obsolete; call this sequence instead:</p> <pre> mxArray *parray; int retval; parray = mxCreateDouble(0, 0, 0); if(parray == (mxArray*)0) return(1); mxSetM(parray, m); mxSetN(parray, n); mxSetPr(parray, pr); mxSetPi(parray, pi); mxSetName(parray, name); retval = mexPutArray(parray, "caller"); mxFree(parray); return(retval); </pre>
<code>mexPutMatrix</code>	Obsolete; call <code>mexPutArray</code> instead.
<code>mexSetTrapFlag</code>	No change
<code>mxCalLoc</code>	No change
<code>mxCreateFull</code>	Obsolete; call <code>mxCreateDoubleMatrix</code> instead
<code>mxCreateSparse</code>	Returns <code>mxArray *</code>
<code>mxCreateString</code>	Returns <code>mxArray *</code>
<code>mxFree</code>	No change
<code>mxFreeMatrix</code>	Obsolete; call <code>mxDestroyArray</code> instead
<code>mxGetIr</code>	First argument is <code>mxArray *</code>
<code>mxGetJc</code>	First argument is <code>mxArray *</code>
<code>mxGetM</code>	First argument is <code>mxArray *</code>
<code>mxGetN</code>	First argument is <code>mxArray *</code>

Table 4-6: Recoding MATLAB 4 Functions for MATLAB 5.0 Compliance

MATLAB 4 Function	MATLAB 5.0 Replacement
<code>mxGetName</code>	First argument is <code>mxArray</code> *
<code>mxGetNzmax</code>	First argument is <code>mxArray</code> *
<code>mxGetPi</code>	First argument is <code>mxArray</code> *
<code>mxGetPr</code>	First argument is <code>mxArray</code> *
<code>mxGetScalar</code>	First argument is <code>mxArray</code> *
<code>mxGetString</code>	First argument is <code>mxArray</code> *
<code>mxIsComplex</code>	First argument is <code>mxArray</code> *
<code>mxIsDouble</code>	First argument is <code>mxArray</code> * Note that MATLAB 4 stores all data as doubles; MATLAB 5.0 stores data in a variety of integer and real formats.
<code>mxIsFull</code>	Obsolete; call <code>!mxIsSparse</code> instead
<code>mxIsNumeric</code>	First argument is <code>mxArray</code> *
<code>mxIsSparse</code>	First argument is <code>mxArray</code> *
<code>mxIsString</code>	Obsolete; call <code>mxIsChar</code> instead
<code>mxSetIr</code>	First argument is <code>mxArray</code> *
<code>mxSetJc</code>	First argument is <code>mxArray</code> *
<code>mxSetM</code>	First argument is <code>mxArray</code> *
<code>mxSetN</code>	First argument is <code>mxArray</code> *
<code>mxSetName</code>	First argument is <code>mxArray</code> *
<code>mxSetNzmax</code>	First argument is <code>mxArray</code> *
<code>mxSetPi</code>	First argument is <code>mxArray</code> *

Table 4-6: Recoding MATLAB 4 Functions for MATLAB 5.0 Compliance

MATLAB 4 Function	MATLAB 5.0 Replacement
<code>mxSetPr</code>	First argument is <code>mxArray</code> *
<code>mxSetString</code>	Obsolete; MATLAB 5.0 provides no equivalent call since the <code>mxArray</code> data type does not contain a string flag. Use <code>mxCreateCharMatrixFromStrings</code> to create multidimensional string <code>mxArrays</code> .

Upgrading Toolboxes and Blocksets

The versions of the toolboxes and blocksets shipped with MATLAB 5.2 are compatible with the previous versions of the respective toolboxes/blocksets. However, there are a minor upgrade issues for the

- Fuzzy Logic Toolbox 2.0
- DSP Blockset 2.2

The Fuzzy Logic Toolbox 2.0 includes an upgrade function to update FIS models from previous versions of the toolbox.

The DSP Blockset 2.2 is completely compatible with Version 1.0a, but there are some limitations on mixing buffer blocks from the two versions, and you will need to recompile any custom blocks that use C-MEX S-functions so that they work with Simulink 2.2.

Fuzzy Logic Toolbox: Updating FIS Models

In the Fuzzy Logic Toolbox 2.0, the Fuzzy Inference System (FIS) is represented as a MATLAB structure. A structure (instead of a flat matrix) is now the basic element in constructing a fuzzy logic system. This fundamental change in the way of representing fuzzy logic system makes many details of working with the constructed system easier.

A Fuzzy Inference System that you created with a pre-2.0 version of the Fuzzy Logic Toolbox is still usable in 2.0, if you run the `convertfis` function on it. The `convertfis` function automatically converts pre-2.0 Fuzzy Inference Systems to work with Version 2.0.

DSP Blockset: Upgrading from Version 1.0a

The DSP Blockset 2.2 is completely compatible with Version 1.0a of the blockset, and Version 2.2 blocks can be used directly in models containing Version 1.0a blocks. However, because of changes to the buffering operation in 2.2, you should avoid using 2.2 buffer blocks in the same model with 1.0a buffer blocks. If you wish to continue using the 1.0a Buffer, Complex Buffer, Unbuffer, and Complex Unbuffer blocks in a model, do not use the 2.2 version of these blocks, or the 2.2 Partial Unbuffer and Complex Partial Unbuffer blocks. If you want to use the 2.2 versions in your model, we recommend that you replace any 1.0a buffering blocks with the 2.2 versions as well.

Also note that if you have custom blocks that use C-MEX S-functions, you will need to recompile the C-MEX S-functions to work with Simulink 2.2. All of the Version 2.2 DSP Blockset C-MEX S-Functions are completely compatible with Simulink 2.2. Additionally, DSP Blockset 2.2 includes updated C-MEX S-functions for the Version 1.0a blocks that use them, so that these 1.0a blocks will continue to work with Simulink 2.2. However, the updated 1.0a C-MEX S-functions will generate warnings indicating that these functions are obsolete, and we recommend that you replace these old blocks with the 2.2 versions when convenient.

A

- ActiveX support 1-12
- alignment tool 3-43
- API
 - cell array support 3-44
 - compiler location 2-13
 - converting to MATLAB 5 interface 4-25
 - enhancements for the Macintosh 2-13
 - MATLAB 5.1
 - API enhancements 2-14
 - fundamental data type 3-44
 - memory leaks 4-7
 - memory management 4-4
 - MEX-files 4-8
 - mexopts. bat file 2-13
 - multidimensional array support 3-44
 - nonANSI C compilers 3-45
 - nondouble data 3-45
 - See also* function, API
 - setup option 2-13
 - structure support 3-44
- Application Program Interface
 - See* API
- Application Program Interface *See* API
- application toolboxes
 - enhancements 1-3, 2-3
- area filling 3-23
- area function 3-23
- array
 - empty 3-18
 - string 3-9
- array editing
 - on the Macintosh 2-5
- Array Editor 2-6
- AspectRatio property 4-22
- assignment 3-16
 - matrix 4-3

- asterisk
 - as wildcard 3-18
- auread function 4-11
- awrite function 4-11
- Axes object 3-26, 4-22

B

- BackgroundColor property 4-22
- bar charts 3-23
- bar3 function 3-23
- bar3h function 3-23
- barh function 3-23
- bessel functions 4-11
- bit-manipulation 3-16
- Block Properties dialog box (Simulink) 1-28
- blocks
 - automatic connection 1-28
- browser
 - path 3-46, 3-51
 - workspace 3-47, 3-52

C

- C Math Library 1.2 1-23
- C++ Math Library 1.2 1-25
- callback editor 3-43
- camera 1-14
- camera properties 3-25
- Canny edge detection 1-48
- case statement 3-11
- cat function 3-5
- catch 4-3
- CDat aMapping property 4-23
- cell array 3-7
 - API support 3-44

- use by functions 1-8
- cells function 3-7
- cessen function (obsolete) 4-18
- character set, Japanese 3-49
- characters Units 1-19
- Cholesky factorization 1-6
- cla function 1-17, 1-18
- clabel function 3-25
- clc function 1-8
- clear 4-3
- clf function 1-17, 1-18
- CodeWarrior 11 2-14
- color enhancements 3-26
- colorddef function 3-26
- colormaps 3-33
- Command Window 3-48, 3-50
 - clearing 1-8
- Communications Toolbox 1.3
 - enhancements 1-36
 - integration with Real-Time Workshop 2.2 1-37
 - integration with Simulink 2.2 1-37
 - interleave and scrambler blocks 1-36
 - passband digital modulation/demodulation blocks 1-36
- compatibility 4-2
- Compiler 1.2 1-21
- compiler location 2-13
- compilers
 - CodeWarrior 11 2-14
- complex data
 - creating
 - from real data 1-42
- compliance 4-2
- context menus 1-20
- Continue
 - in debugger 2-5
- contouring 3-25

- Control System Toolbox 4.1
 - Root Locus Design GUI 1-37
 - Simulink LTI Viewer 1-38
- control, flow 3-10
- convertfis function 1-47, 4-40
- csvread function (obsolete) 4-18
- csvwrite function (obsolete) 4-18
- cumprod function 3-17
- cumsum function 3-17
- Current Menu property 4-22

D

- data analysis features 3-20
- data constructs 3-5
 - cell array 3-7
 - multidimensional array 3-5
 - structure 3-7
- data frames, DSP Blockset 1-39
- data hiding 3-8
- data types
 - mxArray 3-44
- data visualization 3-25
- date functions
 - Financial Toolbox 1-45
- dbmex function 4-19
- debugger
 - Continue 2-5
 - M-file (Macintosh) 3-53
 - Simulink 1-29
 - step 2-5
- debugger, MATLAB (Macintosh) 3-53
- derivatives functions
 - Financial Toolbox 1-44
- device drivers
 - JPEG 2-9, 2-10
 - TIFF 2-9

- device options
 - print command 3-29
- dialog box
 - modal 3-42, 4-11
 - non-modal 3-42
- dialog function 4-11
- dimension specification 3-17
- directory handling 1-7
- dlmread function 4-18
- dlmwri te function 4-18
- doc command 1-11
- documentation
 - online 1-11
- DSP Blockset 2.2
 - data frames 1-39
 - enhanced blocks 1-42
 - enhancements (summary) 1-38
 - Filter Realization Wizard 1-40
 - new blocks 1-41
 - relationship to Simulink 1-39, 4-41
 - upgrading from Version 1.0a 4-40
 - version numbering 1-39
- E**
- echo function 4-11
- editor
 - callback 3-43
 - Macintosh 3-53
 - menu 3-43
 - property 3-43
- Editor/Debugger 2-5
 - for Microsoft Windows 3-47
- ellipk function (obsolete) 4-18
- ellipke function 4-18
- empty array 3-18
 - checking for 4-14
 - multidimensional 3-18
- empty matrix 3-18
 - checking for 4-14
- Encapsulated PostScript 2-12
- end statements, extra 4-12
- engGetFull function 4-35
- engGetMatrix function 4-35
- engPutFull function 4-35
- engPutMatrix function 4-35
- engSetEvalCallback function 4-35
- engSetEvalTimeout function 4-35
- engWinInit function 4-35
- EPS files 2-12
- eps function 4-12
- EraseMode property 4-22
- erfinv function 4-18
- Excel Link Portfolio Optimizer Tool
 - demo 1-46
- ExpFontAngle property 4-22
- ExpFontName property 4-22
- ExpFontSize property 4-22
- ExpFontStrikeThrough property 4-22
- ExpFontUnderline property 4-22
- ExpFontUnits property 4-22
- ExpFontWeight property 4-22
- extent function (obsolete) 4-18
- eye function 4-15
- F**
- features
 - Macintosh 3-49
 - Microsoft Windows 3-46
 - UNIX workstations 3-54
- feedback loops
 - Simulink 1-29
- figflag function (obsolete) 4-18

- figure function 4-20
- Figure Window 3-54
- file handling 1-7
- filling areas 3-23
- Filter Designer
 - Signal Processing Toolbox 1-50
- Filter Realization Wizard, for DSP Blockset 1-40
- Filter Viewer
 - Signal Processing Toolbox 1-52
- Financial Toolbox 1.1
 - date functions 1-45
 - derivatives functions 1-44
 - enhancements 1-44
 - portfolio analysis functions 1-45
 - term structure functions 1-44
- find function 2-4, 4-9
- find_system command 1-30
- finite function (obsolete) 4-18
- FIS
 - conversion function 1-47
- Fixed-Point Blockset 1.0.2 2-3
- flow control 3-10
 - case 3-11
 - switch 3-10
- FontStrikeThrough property 4-22
- FontUnderline property 4-22
- for 4-12
- full-text search 1-11
- function
 - dimension specification 3-17
- functions

API

- dbmex 4-19
- engGetFull 4-35
- engGetMatrix 4-35
- engPutFull 4-35
- engPutMatrix 4-35
- engSetEvalCallback 4-35
- engSetEvalTimeout 4-35
- engWinInit 4-35
- matGetFull 4-35
- matGetMatrix 4-35
- matGetNextMatrix 4-35
- matPutFull 4-35
- matPutMatrix 4-35
- mexAtExit 4-35
- mexCallMATLAB 4-35
- mexdebug 4-19
- mexErrMsgTxt 4-35
- mexEvalString 4-35
- mexFunction 4-36
- mexGetEps 4-36
- mexGetFull 4-36
- mexGetGlobal 4-36
- mexGetInf 4-36
- mexGetMatrix 4-36
- mexGetMatrixPtr 4-36
- mexGetNaN 4-36
- mexIsFinite 4-36
- mexIsInf 4-36
- mexIsNaN 4-36
- mexPrintf 4-36
- mexPutFull 4-37
- mexPutMatrix 4-37
- mexSetTrapFlag 4-37
- mxCallLoc 4-37
- mxCreateFull 4-37
- mxCreateSparse 4-37

mxCreateString 4-37
mxFree 4-5, 4-37
mxFreeMatrix 4-37
mxGetIr 4-37
mxGetJc 4-37
mxGetM 4-37
mxGetN 4-37
mxGetName 4-38
mxGetNzmax 4-38
mxGetPi 4-38
mxGetPr 4-38
mxGetScalar 4-38
mxGetString 4-38
mxIsComplex 4-38
mxIsDouble 4-38
mxIsFull 4-38
mxIsNumeric 4-38
mxIsSparse 4-38
mxIsString 4-38
mxSetCell 4-5
mxSetData 4-7, 4-8
mxSetField 4-5
mxSetImagData 4-7, 4-8
mxSetIr 4-8, 4-38
mxSetJc 4-8, 4-38
mxSetM 4-38
mxSetN 4-38
mxSetName 4-38
mxSetNzMax 4-38
mxSetPi 4-7, 4-8, 4-38
mxSetPr 4-7, 4-39
mxSetString 4-39
area 3-23
auread 4-11
auwrite 4-11
bar3 3-23
bar3h 3-23
barh 3-23
bessel 4-11
cat 3-5
cells 3-7
cessen 4-18
cla 1-17, 1-18
clabel 3-25
clc 1-8
clf 1-17, 1-18
colordf 3-26
convertfis 1-47, 4-40
csvread 4-18
csvwrite 4-18
cumprod 3-17
cumsum 3-17
diag 4-11
dlmread 4-18
dlmwrite 4-18
doc 1-11
echo 4-11
ellipk 4-18
ellipke 4-18
eps 4-12
erfinv 4-18
extent 4-18
eye 4-15
figflag 4-18
figure 4-20
find 2-4, 4-9
finite 4-18
fwhich 4-18
gca 4-20
gcf 4-20
get 4-18, 4-20
gradient 4-12
griddata 3-21
home 1-8

hthelp 4-18
http 4-18
input 4-13
inquire 4-18
integer bit manipulation 3-16
interp1 4-13
interp2 4-13
interp3 3-21, 4-13
interp4 3-21
interpolation 3-21
intersect 3-21
inverf 4-18
isdir 4-18
isempty 4-14
ismember 3-21
isspace 4-14
lastwarn 1-5
layout 4-19
load 1-7
loadhtml 4-19
matq2ws 4-19
matqdlg 4-19
matqparse 4-19
matqueue 4-19
max 3-19, 4-14
menulabel 4-19
min 3-19, 4-14
msgbox 4-11
munlock 4-3
nargin 4-15
nargout 4-15
newplot 1-17
ode23 4-19
ode23p 4-19
odeset 4-19
ones 4-15
ordinary differential equations 1-6
pathedit 3-55
pcode 3-12
plot 4-9, 4-19, 4-20
plotting 3-23
polyline 4-19
polymark 4-19
print 2-9, 2-12
printmenu 4-19
prod 3-17, 3-19
quiver3 3-24
rand 4-15
saxis 4-19
scatter 2-8
set 4-18
setdiff 3-21, 4-10
setxor 4-10
size 4-10
slice 3-25, 4-16
sound 4-16
soundsc 4-16, 4-19
stem 3-24
stem3 3-24
strcmp 4-16
strjust 1-8
strncmp 4-16
structs 3-8
sum 3-17, 3-19
tremesh 3-25
ui setcolor 2-8
union 3-21, 4-10
unique 3-22, 4-10
use of cell arrays 1-8
wavwrite 4-16
ws2matq 4-19
zeros 4-15
fundamental data type, API 3-44
Fuzzy Logic Toolbox 2.0

- algorithm improvements 1-47
- enhancements 1-46
- FIS represented by MATLAB structures 1-47
- GUI enhancements 1-46
- user-defined membership functions 1-47
- fwhi ch function (obsolete) 4-18

G

- gca function 4-20
- gcf function 4-20
- general graphics features 3-28
- get function 4-18, 4-20
- get_param command 1-30
- global variable 4-12
- gradient function 4-12
- graphical user interface *See* GUI
- graphics
 - OpenGL 1-14
- graphics object
 - Axes 3-27, 4-22
 - defaults 3-28
 - Line 4-19
 - Patch 3-25
 - Text 3-27
- graphics objects handles, hiding 1-17
- griddata function 3-21
- GUI building
 - enhancements 3-42
- Guide 3-43

H

- Handle Graphics 3-23
 - object properties 3-34
 - zoom options 3-26
- handles, of graphics objects 1-17

HDF

- MATLAB support 1-13
- Help Desk 1-11
 - Japanese 1-11
- home function 1-8
- ht help function (obsolete) 4-18
- HTML
 - reference pages 1-11
- http function (obsolete) 4-18

I

- if expressions 3-11
- Image Processing Toolbox 2.1
 - Canny edge detection method 1-48
 - converting images 1-48
 - enhancements 1-48
 - feature measurements 1-48
 - handling holes in objects 1-48
 - inverse Radon transform 1-48
 - YCbCr color space support 1-48
- images
 - 8-bit 3-31
 - indexed 3-32
 - reading and writing 3-31
 - support for 3-30
 - truecolor 3-31, 3-33
- images, truecolor 1-19
- indexed images 3-32
- initializing
 - outputs 4-17
 - variables 4-16, 4-17
- input function
 - no initial linefeed 4-13
- inquire function (obsolete) 4-18
- integer bit manipulation functions 3-16
- integer subscripts 4-16

- interleave and scrambler blocks 1-36
- interp1 function 4-13
- interp2 function 4-13
- interp3 function 3-21, 4-13
- interpn function 3-21
- interpolation
 - higher-dimension 3-21
 - triangle-based 3-21
- interpolation functions 3-21
- intersect 4-10
- intersect function 3-21
- inverf function (obsolete) 4-18
- inverse Radon transform 1-48
- isdir function (obsolete) 4-18
- isempty function 4-14
- ismember function 3-21
- isspace function 4-14

J

- Japanese character set 3-49
- Japanese Help Desk 1-11
- JPEG device driver 2-9
- justification, string 1-8

K

- Kanji 2-4
- Keithley-Metrabyte I/O support 1-32

L

- language enhancements
 - MATLAB 5.2 1-5
- lasterr 4-9
- lastwarn function 1-5
- layout function (obsolete) 4-19

- legend function 3-24
- Level 2 S-functions 1-29, 1-33
- Light objects 3-28
- lighting, convenience commands for 1-16
- line markers 3-24
- Line object 4-19
- Line properties
 - LineStyle 4-23
 - Marker 4-23
- line styles 4-20
- LineStyle property 4-23
- List Box object 3-42
- load function 1-7
- loadhtml function (obsolete) 4-19
- locking M-files 1-6
- LTI Viewer 1-38

M

- Macintosh
 - Array Editor 2-6
 - editing arrays 2-5
 - features 3-49
 - Path Browser 2-6
 - TIFF preview 2-12
- MALAB 5.0
 - three-dimensional plotting 3-24
- Mapping Toolbox 2-17
- Marker property 4-23
- mask parameter
 - Simulink 1-29
- masking 4-14
- matGetFull function 4-35
- matGetMatrix function 4-35
- matGetNextMatrix function 4-35
- MATLAB 5.0
 - API enhancements 3-44

- assignment enhancements 3-16
 - Axes objects 3-26
 - bar chart enhancements 3-23
 - cat function 3-5
 - character arrays 3-9
 - color enhancements 3-26
 - Command Window 3-48, 3-50
 - contouring enhancements 3-25
 - data constructs added 3-5
 - debugger 3-53
 - dimension specification for data analysis 3-17
 - Editor/Debugger 3-47
 - empty arrays 3-18
 - Figure Window 3-54
 - filling areas 3-23
 - flow-control enhancements 3-10
 - GUI building enhancements 3-42
 - Guide enhancements 3-43
 - Handle Graphics
 - enhancements 3-23
 - image support 3-30
 - integer-bit manipulation 3-16
 - interpolation functions 3-21
 - Light objects 3-28
 - line markers 3-24
 - Macintosh enhancements 3-49
 - M-file profiler 3-12
 - M-file programming tools 3-12
 - Microsoft Windows tools 3-46
 - multidimensional arrays 3-5
 - mxArray data type 3-44
 - object property enhancements 3-34
 - object-oriented programming 3-8
 - Patch definition 3-25
 - Path Browser 3-46, 3-51
 - Path Editor 3-55
 - plotting capabilities 3-23
 - printing enhancements 3-29
 - slicing surfaces 3-25
 - subfunctions 3-12
 - subscripting enhancements 3-16
 - TeX commands 3-27
 - Text object enhancements 3-27
 - triangular meshes 3-25
 - UNIX enhancements 3-54
 - viewing models 3-25
 - visualization enhancements 3-25
 - wildcards in utility commands 3-18
 - Workspace Browser 3-47
 - zoom options 3-26
- MATLAB 5.1**
- API enhancements 2-13
 - Array Editor 2-6
 - Editor/Debugger 2-5
 - Encapsulated Postscript files 2-12
 - enhancements (summary) 2-2
 - find function 2-4
 - Handle Graphics enhancements 2-8
 - JPEG device driver 2-10
 - Kanji 2-4
 - multibyte characters 2-4
 - Path Browser 2-6
 - TCP/IP 2-4
 - TIFF device driver 2-9
 - visualization enhancements 2-8
- MATLAB 5.2**
- ActiveX support 1-12
 - API memory management 4-4
 - blockset upgrades 1-35
 - C Math Library 1.2 1-23
 - C++ Math Library 1.2 1-25
 - camera enhancements 1-14
 - cell array support 1-8
 - cla enhancement 1-18

- clc and home changes 1-8
- clf enhancement 1-18
- Compiler 1-21
- context menus 1-20
- directory handling 1-7
- Editor/Debugger enhancements 1-9
- file handling 1-7
- HDF file format support 1-13
- Help Desk enhancements 1-11
- hiding graphics objects 1-17
- language enhancements 1-5
- lighting convenience commands 1-16
- mathematical functions 1-6
- matrix assignment 4-3
- method search order 4-3
- M-file locking 1-6
- Microsoft Windows tool enhancements 1-9
- newplot enhancement 1-17
- ODE functions 1-6
- online documentation enhancements 1-11
- paper types 1-16
- PC tools enhancements 1-9
- persistent variables 1-8
- recursion limit 1-5
- string comparison 1-6
- strjust enhancement 1-8
- toggle buttons 1-19
- toolbox upgrades 1-35
- tooltips 1-19
- Units property value 1-19
- UNIX tool enhancements 1-10
- view control commands 1-14
- warning messages 1-5
- MATLAB Editor/Debugger 1-9
- matPutFull function 4-35
- matPutMatrix function 4-35
- matq2ws function (obsolete) 4-19
- matqdlg function (obsolete) 4-19
- matqparse function (obsolete) 4-19
- matqueue function (obsolete) 4-19
- matrix
 - empty 3-18
- matrix assignment 4-3
- max function 4-14
 - with empty argument 3-19
- mdlProcessParameters S-function 1-29, 1-33
- mdlRTWS-function 1-29, 1-33
- mdlStart S-function 1-29, 1-33
- measuring features in images 1-48
- memory leaks 4-7
- memory management
 - API 4-4
- menlabeled function 4-19
- menu editor 3-43
- Merge block 1-29
- meshes
 - and triangulation 3-25
- method search order 4-3
- mexAtExit function 4-35
- mexCallMATLAB function 4-35
- mexdebug function (obsolete)
 - obsolete 4-19
- mexErrMsgTxt function 4-35
- mexEvalString function 4-35
- mexFunction function 4-36
- mexGetEps function (obsolete) 4-36
- mexGetFull function (obsolete) 4-36
- mexGetGlobal function 4-36
- mexGetGlobal function (obsolete) 4-36
- mexGetInf function (obsolete) 4-36
- mexGetMatrix function 4-36
- mexGetMatrixPtr function 4-36
- mexGetNaN function (obsolete) 4-36
- mexIsFinite function (obsolete) 4-36

- mexIsInf function (obsolete) 4-36
 - mexIsNaN function (obsolete) 4-36
 - mexPrintf function 4-36
 - mexPutFull function (obsolete) 4-37
 - mexPutMatrix function (obsolete) 4-37
 - mexSetTrapFlag function 4-37
 - M-file
 - clearing 1-6
 - locking 1-6
 - profiling 3-12
 - programming tools 3-12
 - pseudocode 3-12
 - variable number of arguments 3-12
 - with multiple functions 3-12
 - M-Files
 - converting to MATLAB 5 4-11
 - min function 4-14
 - with empty argument 3-19
 - ml ock 4-3
 - modal dialog box 3-42, 4-11
 - Mode property 4-23
 - mouse pointer 3-42
 - msgbox function 4-11
 - multibyte characters 2-4
 - multidimensional array 3-5
 - API support 3-44
 - empty 3-18
 - multiple functions within an M-file 3-12
 - munl ock function 4-3
 - mxArray
 - memory management 4-4
 - mxArray data type 3-44
 - mxCall loc function 4-37
 - mxCreateFull function 4-37
 - mxCreateSparse function 4-37
 - mxCreateString function 4-37
 - mxFree function 4-5, 4-37
 - mxFreeMatrix function (obsolete) 4-37
 - mxGetIr function 4-37
 - mxGetJc function 4-37
 - mxGetM function 4-37
 - mxGetN function 4-37
 - mxGetName function 4-38
 - mxGetNzmax function 4-38
 - mxGetPi function 4-38
 - mxGetPr function 4-38
 - mxGetScalar function 4-38
 - mxGetString function 4-38
 - mxIsComplex function 4-38
 - mxIsDouble function 4-38
 - mxIsFull function (obsolete) 4-38
 - mxIsNumeric function 4-38
 - mxIsSparse function 4-38
 - mxIsString function (obsolete) 4-38
 - mxSetCell function 4-5
 - mxSetData function 4-7, 4-8
 - mxSetField function 4-5
 - mxSetImagData function 4-7, 4-8
 - mxSetIr function 4-8, 4-38
 - mxSetJc function 4-8, 4-38
 - mxSetM function 4-38
 - mxSetN function 4-38
 - mxSetName function 4-38
 - mxSetNzmax function 4-38
 - mxSetPi function 4-7, 4-8, 4-38
 - mxSetPr function 4-7, 4-39
 - mxSetString function (obsolete) 4-39
- N**
- nargin function 4-15
 - nargout function 4-15
 - Neural Network Toolbox 3.0
 - enhancements 1-49

- improved Simulink support 1-49
- modular network representation 1-49
- new algorithms 1-49
- new network types 1-49
- new training options 1-49
- newplot function 1-17
- NextPlot property 4-23
- nonANSI C compilers 3-45
- nondouble data
 - API support 3-45
- non-modal dialog box 3-42
- Notebook 2-4

O

- object
 - Axes 3-26
 - List Box 3-42
 - Patch 3-25
 - Text 3-27
- object-oriented programming 3-8
- obsolete functions 4-18
- ode23 function (obsolete) 4-19
- ode23p function (obsolete) 4-19
- odeset function (obsolete) 4-19
- Office 97 2-4
- ones function
 - functions
 - ones 4-15
 - with matrix inputs 4-15
- online documentation 1-11
 - Japanese 1-11
- OpenGL renderer 1-14
- ordinary differential equation functions 1-6
- outputs
 - initializing 4-17
- overloading 3-9

P

- PaletteMode property 4-23
- paper types for printing 1-16
- passband digital modulation/demodulation blocks 1-36
- Patch object 3-25
- patches
 - printing 2-8
- Path Browser 3-46, 3-51
 - on the Macintosh 2-6
- Path Editor 3-55
- pathedit function 3-55
- pcode function 3-12
- persistent 4-3
- persistent keyword 1-8
- persistent variables 1-8
- plot function 4-9, 4-19, 4-20
- plotting
 - three-dimensional 3-24
- plotting capabilities 3-23
- polylines function (obsolete) 4-19
- polymark function (obsolete) 4-19
- portfolio analysis functions
 - Financial Toolbox 1-45
- Postscript 2-12
- Power System Blockset 1.0 1-36
- print command 3-29
- print frames, in Simulink 1-31
- print function 2-9, 2-12
 - changes to 4-15
- print options
 - generating M-file to recreate figure 3-29
 - PostScript bounding box 3-29
 - Uicontrol objects 3-29
 - user-selectable Z-buffer resolution 3-29
- printing 2-9, 2-12
 - paper types 1-16

- patches 2-8
- Simulink 1-31
- surfaces 2-8
- printmenu function (obsolete) 4-19
- prod function 3-17
 - with empty argument 3-19
- profiler 3-12
- programming
 - object-oriented 3-8
- programming tools 3-12
- property
 - AspectRatio 4-22
 - BackgroundColor 4-22
 - CDatamapping 4-23
 - CurrentMenu 4-22
 - EraseMode 4-22
 - ExpFontAngle 4-22
 - ExpFontName 4-22
 - ExpFontSize 4-22
 - ExpFontStrikeThrough 4-22
 - ExpFontUnderline 4-22
 - ExpFontUnits 4-22
 - ExpFontWeight 4-22
 - FontStrikeThrough 4-22
 - FontUnderline 4-22
 - Mode 4-23
 - NextPlot 4-23
 - PaletteMode 4-23
 - RenderLimit 4-23
 - SelectionType 4-23
 - Style 3-28
 - Units 4-24
 - WindowID 4-24
 - XLim 4-24
 - XTick 4-24
 - XTickLabel 4-24
 - XTickLabels 4-24

- YLim 4-24
- YTick 4-24
- YTickLabel 4-24
- YTickLabels 4-24
- ZLim 4-25
- ZTick 4-25
- ZTickLabel 4-25
- ZTickLabels 4-25
- property editor 3-43
- pseudocode 3-12

Q

- quiver3 function 3-24

R

- Radon 1-48
- rand function 4-15
 - with matrix inputs 4-15
- random number generation 4-15
- Real-Time Workshop
 - relationship to Stateflow 2-16
- Real-Time Workshop 2.2
 - asynchronous interrupt handling 1-32
 - Keithley-Metrabyte I/O support 1-32
 - Level 2 S-functions 1-29, 1-33
 - Merge block 1-32
 - RTWlib 1-32
 - summary of enhancements 1-3
 - VxWorks 1-32
 - VxWorks Tornado 1-32
 - with Communications Toolbox 1.3 1-37
- recursion limit 1-5
- reference pages
 - navigation 1-11
- renderer

- OpenGL 1-14
- RenderLimit property 4-23
- Root Locus Design GUI 1-37
- RTWlib 1-32

- S**
- saxis function (obsolete) 4-19
- scalar expansion for subarray assignments 3-16
- scatter function 2-8
- search
 - full text 1-11
- SelectiOnType property 4-23
- set function 4-18
- set theoretic functions 3-21
- setdiff function 3-21, 4-10
- setxor function 4-10
- S-functions
 - Level 2 1-29, 1-33
 - mdl ProcessParameters 1-29, 1-33
 - mdl RTW 1-29, 1-33
 - mdl Start 1-29, 1-33
- SGI64 1-10
- Signal Processing Toolbox 4.1
 - enhancements 1-50, 1-53
 - Filter Designer interface enhancements 1-50
 - Filter Viewer enhancements 1-52
 - spectral estimation enhancements 1-50
 - Spectrum Viewer enhancements 1-52
 - SPTool enhancements 1-50
- simulation 1-28
- Simulink
 - relationship to Stateflow 2-16
- Simulink 2.2
 - additional solvers 1-28
 - automatic block connection 1-28
 - Block Properties dialog box 1-28
 - context-sensitive menus 1-27
 - debugger 1-29
 - dialog parameters 1-30
 - editing frames 1-31
 - enhancements (summary) 1-3
 - feedback loop handling 1-29
 - find_system 1-30
 - get_param 1-30
 - lines and annotations API 1-30
 - mask parameter 1-29
 - Merge block 1-29
 - model construction 1-30
 - object parameters 1-30
 - printing frames 1-31
 - printing title blocks 1-31
 - simulation 1-28
 - status bar 1-27
 - toolbar 1-27
 - undoing breaking of library links 1-28
- Simulink LTI Viewer 1-38
- size function 4-10
- slice function 3-25, 4-16
- slicing surfaces 3-25
- solvers
 - Simulink 1-28
- sound function 4-16
- soundsc function 4-16, 4-19
- Spectrum Viewer
 - Signal Processing Toolbox 1-52
- splash screen
 - suppressing on UNIX system 3-42
- Spline Toolbox 2.0
 - multivariate spline support 1-53
 - user interface enhancements 1-53
 - vector-valued spline enhancements 1-53
- startup file 3-28
- Stateflow

- relationship to Real-Time Workshop 2-16
- relationship to Simulink 2-16
- Stateflow 1.0.6 1-34
- Stateflow Coder 2-16
- stem function 3-24
- stem3 function 3-24
- step
 - in debugger 2-5
- stereo sound
 - Macintosh 3-49
 - Windows 3-49
- strcmp function
 - with numeric inputs 4-16
- string array 3-9
- string comparison 1-6
- strings
 - comparing 1-6
 - justification 1-8
- strjust function 1-8
- strncmp function
 - with numeric inputs 4-16
- structs function 3-8
- structure 3-7
 - API support 3-44
- Style property 3-28
- subfunctions 3-12
- subscripting 3-16
- subscripts
 - must be integers 4-16
- sum function
 - dimension specifier 3-17
 - with empty argument 3-19
- surface slicing 3-25
- surfaces
 - printing 2-8
 - triangulation 3-25
- switch statement 3-10

T

- TCP/IP 2-4
- term structure functions
 - Financial Toolbox 1-44
- TeX commands 3-27
- text 4-21
- Text object 3-27
 - TeX commands 3-27
- texture-mapped patches and surfaces 2-8
- three-dimensional plotting 3-24
- TIFF
 - device driver 2-9
 - preview images for encapsulated PostScript 2-12
- title blocks
 - printing 1-31
- toggle buttons 1-19
- toolboxes
 - enhancements 1-35
 - updated for MATLAB 5.2 1-3
- tools
 - UNIX 1-10
- tooltips 1-19
- ToolTipString property 1-19
- triangle-based interpolation 3-21
- triangular meshes 3-25
- triangular surfaces 3-25
- tri mesh function 3-25
- truecolor 3-31
- truecolor images 1-19, 3-31, 3-33
- try 4-3

U

- Uicontrol
 - text alignment 4-21
- uicontrol function 4-21

Uicontrol object

- List Box 3-42
- ui set col or function 2-8
- undoing breaking of library links
 - Simulink 1-28
- union function 3-21, 4-10
- unique function 3-22, 4-10
- Units property 4-24
- Units property 1-19
- UNIX tools 1-10
- Upgrading from DSP Blockset 1.0a 4-40
- Upgrading from MATLAB 4 4-11
- Upgrading from MATLAB 5.0 4-9
- Upgrading from MATLAB 5.1 4-3

V

- varargin 3-12
- varargout 3-12
- variable number of inputs to M-files 3-12
- variable number of outputs for M-files 3-12
- variables
 - global 4-12
 - initializing 4-17
 - names 4-11
- version
 - compatibility 4-2
 - compliance 4-2
- view control 1-14
- viewing models 3-25
- vis3d option 3-26
- visualization, data 3-25
- VxWorks
 - support in Real-Time Workshop 2.2 1-32
- VxWorks Tornado 1-32

W

- warning messages 1-5
- wavwrite function 4-16
- wildcard for utility commands 3-18
- WindowID property 4-24
- Workspace Browser 3-47, 3-52
- ws2matq function (obsolete) 4-19

X

- XLim property 4-24
- XTick property 4-24
- XTickLabel property 4-24
- XTickLabels property 4-24

Y

- YCbCr color space 1-48
- YLim property 4-24
- YTick property 4-24
- YTickLabel property 4-24
- YTickLabels property 4-24

Z

- Z-buffering 3-28
 - printing Z-buffer images 3-29
- zeros function
 - with matrix inputs 4-15
- ZLim property 4-25
- zoom options 3-26
- ZTick property 4-25
- ZTickLabel property 4-25
- ZTickLabels property 4-25