# MATLAB®

The Language of Technical Computing

Computation

Visualization

Programming

The MATH WORKS Inc.

Release Notes for Release 12

# Contents

## 1 Introducing Release 12

**What's New in Release 12?** . . . . . . . . . . 1-2
General Themes of Release 12 . . . . . . . . . . 1-2
Products with Major Upgrades . . . . . . . . . . 1-4
Products with Minor Upgrades . . . . . . . . . . 1-15
New Products (Since Release 11.0) . . . . . . . . . . 1-16

**Platform Limitations** . . . . . . . . . . 1-17
Differences for the HP and IBM Platforms . . . . . . . . . . 1-17
Problems Entering Accented Characters on Linux . . . . . . . . . . 1-20
Problems Entering Accented Characters on UNIX Platforms 1-20

**Installation Notes** . . . . . . . . . . 1-21

**Linux Distribution Notes** . . . . . . . . . . 1-22

**For More Information About What's New in Release 12** . . 1-23
Printable Version of the Release Notes . . . . . . . . . . 1-23

**MATLAB Software Acknowledgments** . . . . . . . . . . 25

## 2 MATLAB 6.0 Release Notes

**New Features** . . . . . . . . . . 2-2
Development Environment Features . . . . . . . . . . 2-2
Mathematics Features . . . . . . . . . . 2-11
Programming and Data Types Features . . . . . . . . . . 2-22
Graphics Features . . . . . . . . . . 2-26
3-D Visualization Features . . . . . . . . . . 2-30
External Interfaces/API Features . . . . . . . . . . 2-33
Creating Graphical User Interfaces – Features . . . . . . . . . . 2-40

**Major Bug Fixes** . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **2-42**
Figure KeyPressFcn . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **2-42**

**Platform Limitations for HP and IBM** . . . . . . . . . . . . . . . . . . **2-43**
Development Environment Limitations . . . . . . . . . . . . . . . . . . **2-43**
Mathematics Limitations . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **2-44**
Graphics and Printing Limitations . . . . . . . . . . . . . . . . . . . . . **2-45**
GUIDE Limitations . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **2-45**

**Upgrading from an Earlier Release** . . . . . . . . . . . . . . . . . . . . **2-46**
Development Environment Issues . . . . . . . . . . . . . . . . . . . . . . **2-46**
Programming and Data Types Issues . . . . . . . . . . . . . . . . . . . . **2-46**
External Interfaces/API Issues . . . . . . . . . . . . . . . . . . . . . . . . **2-57**
Creating Graphical User Interfaces – Upgrade Issues . . . . . . **2-62**

**Known Software and Documentation Problems** . . . . . . . . . **2-63**
Development Environment Problems . . . . . . . . . . . . . . . . . . . . **2-63**
External Interfaces/API Problems . . . . . . . . . . . . . . . . . . . . . . **2-65**
Graphics Problems . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **2-65**
GUIDE Problems . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **2-66**
Documentation Updates . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **2-66**

**3** **Simulink 4.0 Release Notes**

**New Features** . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **3-2**
Simulink Editor . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **3-2**
Modeling Enhancements . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **3-5**
Simulink Debugger . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **3-6**
Block Library . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **3-6**
SB2SL . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **3-9**

**Platform Limitations for HP and IBM** . . . . . . . . . . . . . . . . . . **3-11**

**Upgrading from an Earlier Release** . . . . . . . . . . . . . . . . . . . . **3-12**
Port Name Property . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **3-12**

# 4 Stateflow 4.0 Release Notes

**New Features** . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **4-2**
Stateflow Works with MATLAB Release 12 . . . . . . . . . . . . . . . . **4-2**
Temporal Logic . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **4-2**
Subcharts . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **4-2**
Graphical Functions . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **4-3**
Symbol Autocreation Wizard . . . . . . . . . . . . . . . . . . . . . . . . . **4-3**
Command Toolbar . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **4-3**
Navigation Toolbar . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **4-3**
Straight Line Transitions . . . . . . . . . . . . . . . . . . . . . . . . . . . . **4-4**
Workspace-Based Data . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **4-4**
Explorer Copy Properties . . . . . . . . . . . . . . . . . . . . . . . . . . . . **4-4**
Library Link Icons . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **4-4**

**Known Software and Documentation Problems** . . . . . . . . . . **4-5**
Many Open Windows Can Cause a Crash or Hang
(Windows 95/98/Me) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **4-5**
Calling MATLAB Functions From Stateflow Charts . . . . . . . . . **4-5**

# 5 Real-Time Workshop 4.0 Release Notes

**Release Summary** . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **5-2**

**New Features** . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **5-3**
Real-Time Workshop Embedded Coder . . . . . . . . . . . . . . . . . . **5-3**
Simulink Data Object Support . . . . . . . . . . . . . . . . . . . . . . . . **5-3**
ASAP2 Support . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **5-4**
Enhanced Real-Time Workshop Page . . . . . . . . . . . . . . . . . . . **5-4**
Other User Interface Enhancements . . . . . . . . . . . . . . . . . . . . **5-4**
Advanced Options Page . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **5-4**
TLC Debugging Page . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **5-5**
Model Parameter Configuration Dialog . . . . . . . . . . . . . . . . . . **5-5**
Tunable Expressions Supported . . . . . . . . . . . . . . . . . . . . . . . **5-5**
S-Function Target Enhancements . . . . . . . . . . . . . . . . . . . . . . **5-6**

External Mode Enhancements . . . . . . . . . . . . . . . . . . . . . . . . . **5-6**
Build Directory . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **5-7**
Code Optimization Features . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **5-7**
Subsystem Based Code Generation . . . . . . . . . . . . . . . . . . . . . . . **5-8**
Nonvirtual Subsystem Code Generation . . . . . . . . . . . . . . . . . . **5-8**
Filename Extensions for Generated Files . . . . . . . . . . . . . . . . . **5-8**
hilite_system and Code Tracing . . . . . . . . . . . . . . . . . . . . . . . . . **5-9**
Generation of Parameter Comments . . . . . . . . . . . . . . . . . . . . . **5-9**
Borland 5.4 Compiler Support . . . . . . . . . . . . . . . . . . . . . . . . . . **5-9**
Enhanced Makefile Include Path Rules . . . . . . . . . . . . . . . . . . . **5-9**
Target Language Compiler 4.0 . . . . . . . . . . . . . . . . . . . . . . . . . **5-10**

**Platform Limitations for HP and IBM** . . . . . . . . . . . . . . . . . **5-13**

**Upgrading from an Earlier Release** . . . . . . . . . . . . . . . . . . . . **5-14**
Column-Major Matrix Ordering . . . . . . . . . . . . . . . . . . . . . . . . **5-14**
Including Generated Files . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **5-14**
Updating Release 11 Custom Targets . . . . . . . . . . . . . . . . . . . **5-14**
hilite_system Replaces locate_system . . . . . . . . . . . . . . . . . . . . **5-15**
TLC Compatibility Issues . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **5-15**

**Known Software and Documentation Problems** . . . . . . . . . **5-18**
Filename Option in Nonvirtual
Subsystem Code Generation . . . . . . . . . . . . . . . . . . . . . . . . . . **5-18**
TLC Debugger Is Disabled . . . . . . . . . . . . . . . . . . . . . . . . . . . . **5-18**
Related Products Table Correction . . . . . . . . . . . . . . . . . . . . . **5-18**

**6** **CDMA Reference Blockset 1.0.2 Release Notes**

**Introduction to the CDMA Reference Blockset** . . . . . . . . . . . **6-2**

**Known Software and Documentation Problems** . . . . . . . . . . **6-3**
Signal Support . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **6-3**

# 7 Communications Blockset 2.0 Release Notes

**New Features** . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **7-2**
Digital Modulation Libraries . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **7-2**
Interleaving Libraries . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **7-2**
Fading Channels . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **7-3**
Enhanced Support for Convolutional Coding . . . . . . . . . . . . . . . **7-3**
Sequence Operations . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **7-3**

**Major Bug Fixes** . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **7-4**

**Upgrading from an Earlier Release** . . . . . . . . . . . . . . . . . . . . . **7-5**
New Block Libraries in Release 12 . . . . . . . . . . . . . . . . . . . . . . . **7-5**
New Signal Support . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **7-7**
Functionality Changes in Specific Blocks . . . . . . . . . . . . . . . . . . **7-7**
Block Name Changes Since Previous Manual . . . . . . . . . . . . . . **7-8**
Obsolete Blocks from Previous Manual . . . . . . . . . . . . . . . . . . **7-12**

**Known Software and Documentation Problems** . . . . . . . . . **7-19**
Code Generation Limitations . . . . . . . . . . . . . . . . . . . . . . . . . . **7-19**
Limited Frame and Matrix Support . . . . . . . . . . . . . . . . . . . . . **7-19**
Errors in Printed Version of Documentation . . . . . . . . . . . . . **7-20**

# 8 Communications Toolbox 2.0 Release Notes

**New Features** . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **8-2**
Convolutional Coding Functions . . . . . . . . . . . . . . . . . . . . . . . . . **8-2**
Gaussian Noise Functions . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **8-2**
Other New Functions . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **8-3**
Enhancements to Existing Functions . . . . . . . . . . . . . . . . . . . . . **8-3**

**Major Bug Fixes** . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **8-4**

**Upgrading from an Earlier Release** . . . . . . . . . . . . . . . . . . . . . **8-5**
Changes in Functionality . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **8-5**

Obsolete Functions . . . . . . . . . . 8-6

**Known Software and Documentation Problems** . . . . . . . . . . **8-8**
Errors in Printed Version of Documentation . . . . . . . . . . **8-8**

9 Control System Toolbox 5.0 Release Notes

**New Features** . . . . . . . . . . **9-2**
SISO Design Tool . . . . . . . . . . **9-2**
LTI Viewer . . . . . . . . . . **9-2**
Property and Preference Editors . . . . . . . . . . **9-3**
Algorithmic Enhancements . . . . . . . . . . **9-3**
New Demo Suite . . . . . . . . . . **9-3**
Documentation . . . . . . . . . . **9-3**

**Platform Limitations for HP and IBM** . . . . . . . . . . **9-5**
Editors . . . . . . . . . . **9-5**
SISO Design Tool . . . . . . . . . . **9-5**

10 Data Acquisition Toolbox 2.0 Release Notes

**New Features** . . . . . . . . . . **10-2**
ComputerBoards Support . . . . . . . . . . **10-2**
Adaptor Kit . . . . . . . . . . **10-2**
New Functions and Properties . . . . . . . . . . **10-3**
Modified Functions and Properties . . . . . . . . . . **10-4**

**Major Bug Fixes** . . . . . . . . . . **10-5**

**Upgrading from an Earlier Release** . . . . . . . . . . **10-6**
Vendor Drivers . . . . . . . . . . **10-6**
Removed and Obsolete Functions . . . . . . . . . . **10-6**

Two daqhwinfo Fields Removed . . . . . . . . . . . . . . . . . . . . . . . . **10-6**

**Known Software Problems** . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **10-7**
Functions . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **10-7**
Vendors . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **10-7**

# 11 Database Toolbox 2.1 Release Notes

**New Features** . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **11-2**
Platforms Supported . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **11-2**
Performance Improvement . . . . . . . . . . . . . . . . . . . . . . . . . . . . **11-2**
Starting the Database Toolbox . . . . . . . . . . . . . . . . . . . . . . . . . **11-2**
Exporting Results to Report Generator . . . . . . . . . . . . . . . . . . **11-2**
Grouping Constraints for a Single Field . . . . . . . . . . . . . . . . . . **11-3**

**Platform Limitations for HP 10.2** . . . . . . . . . . . . . . . . . . . . . . **11-4**

**Upgrading from an Earlier Release** . . . . . . . . . . . . . . . . . . . . **11-5**
Starting the Database Toolbox . . . . . . . . . . . . . . . . . . . . . . . . . **11-5**

# 12 Datafeed Toolbox 1.2 Release Notes

**Introduction to the Datafeed Toolbox** . . . . . . . . . . . . . . . . . . **12-2**
Learning More About the Datafeed Toolbox . . . . . . . . . . . . . . **12-2**

**New Features** . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **12-3**
Support for IDC and Yahoo . . . . . . . . . . . . . . . . . . . . . . . . . . . **12-3**

13 Developer's Kit for Texas Instruments™ DSP 1.0 Release Notes

**Introduction to Developer's Kit for Texas Instruments™ DSP** . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **13-2**

**Known Software and Documentation Problems** . . . . . . . . . **13-4**
Revision 0 TMS320C6701 Evaluation Modules . . . . . . . . . . . . **13-4**
Real-Time Workshop Compiler Optimization Option . . . . . . . **13-4**

14 Dials & Gauges Blockset 1.1 Release Notes

**New Features** . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **14-2**
Real-Time Workshop Support . . . . . . . . . . . . . . . . . . . . . . . . . . **14-2**
External Mode Support . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **14-2**
Block Parameters Dialog Boxes Enhanced . . . . . . . . . . . . . . . . **14-2**
New Demos Added . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **14-3**

**Upgrading from an Earlier Release** . . . . . . . . . . . . . . . . . . . . **14-4**
Double-Clicking on Block Borders . . . . . . . . . . . . . . . . . . . . . . **14-4**
Values Inserted in the "Events on which to output" Field . . . **14-4**

**Known Software and Documentation Problems** . . . . . . . . . **14-5**
Error Message at Installation . . . . . . . . . . . . . . . . . . . . . . . . . . **14-5**
Documentation Update . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **14-5**

15 DSP Blockset 4.0 Release Notes

**New Features** . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **15-2**
Matrix Support . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **15-2**
Frame Support . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **15-4**

New and Enhanced Blocks . . . . . . . . . . . . . . . . . . . . . . . . . . . **15-5**
Library Reorganization . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **15-7**

**Platform Limitations for HP and IBM** . . . . . . . . . . . . . . . . **15-12**

**Upgrading from an Earlier Release** . . . . . . . . . . . . . . . . . . . **15-13**
Adding Release 12 Blocks to Release 11 Models . . . . . . . . . . **15-13**

# 16 Filter Design Toolbox 2.0 Release Notes

**New Features** . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **16-2**
New Quantization Tool for Advanced FIR and IIR Filter Design . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **16-2**
New Objects . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **16-3**
New Filter Design Functions . . . . . . . . . . . . . . . . . . . . . . . . . . **16-3**
New Filter Conversion Functions . . . . . . . . . . . . . . . . . . . . . . **16-4**
New Filter Structures . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **16-4**
New Analysis Methods . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **16-5**
New Functions and Enhancements . . . . . . . . . . . . . . . . . . . . **16-5**

**Platform Limitations for HP 10.2** . . . . . . . . . . . . . . . . . . . . . **16-7**

**Upgrading from an Earlier Release** . . . . . . . . . . . . . . . . . . . **16-8**
Obsolete Functions in Version 2.0 . . . . . . . . . . . . . . . . . . . . . **16-8**

**Known Software and Documentation Problems** . . . . . . . . . **16-9**
Filter Design and Analysis Tool . . . . . . . . . . . . . . . . . . . . . . . **16-9**
Switching Between Design and Quantization Modes in FDATool . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **16-9**

# 17 Financial Derivatives Toolbox 1.0 Release Notes

**Introduction to the Financial Derivitives Toolbox** ...... **17-2**
Interest Rate Models ........ **17-2**
Hedging ........ **17-2**
Financial Instruments ........ **17-2**

# 18 Financial Time Series Toolbox 1.0 Release Notes

**Introduction to the Financial Time Series Toolbox** ...... **18-2**
Creating Financial Time Series Objects ........ **18-2**
Operations ........ **18-2**
Technical Analysis ........ **18-2**
Learning More About the Financial Time Series Toolbox .... **18-3**

# 19 Fixed-Point Blockset 3.0 Release Notes

**New Features** ........ **19-2**
New Blocks ........ **19-2**
New Functions ........ **19-3**
Data Type Support ........ **19-4**
Data Type and Scaling Inheritance ........ **19-4**
Increased Speed, Efficiency, and Accuracy ........ **19-5**
Overflow Logging ........ **19-5**
Simulink Accelerator ........ **19-5**
Model Parameter Configuration ........ **19-5**
Display of Scaling Information ........ **19-5**

20 Fuzzy Logic Toolbox 2.1 Release Notes

**New Features** . . . . . . . . . . 20-2
Wizard For Fuzzy Logic Controller Block . . . . . . . . . . 20-2

**Known Software and Documentation Problems** . . . . . . . . . 20-3
ODE Solver Performance . . . . . . . . . . 20-3

21 GARCH Toolbox 1.0 Release Notes

**Introduction to the GARCH Toolbox** . . . . . . . . . . 21-2

22 Instrument Control Toolbox 1.0 Release Notes

**Introduction to the Instrument Control Toolbox** . . . . . . . . 22-2
Supported Platforms . . . . . . . . . . 22-2
Object-Based Design . . . . . . . . . . 22-2
Reading and Writing Data . . . . . . . . . . 22-3
Events and Actions . . . . . . . . . . 22-3
Data Recording . . . . . . . . . . 22-3
Graphical Tools . . . . . . . . . . 22-3

**Vendor Driver Requirements and Limitations** . . . . . . . . . . 22-4
Driver Requirements . . . . . . . . . . 22-4
GPIB Driver Limitations . . . . . . . . . . 22-5
VISA Driver Limitations . . . . . . . . . . 22-7

**Known Software Problems** . . . . . . . . . . 22-9
Pin Status Event . . . . . . . . . . 22-9
Available Serial Ports . . . . . . . . . . 22-9
Accessing Serial Ports on Solaris . . . . . . . . . . 22-10

# 23 Image Processing Toolbox 2.2.2 Release Notes

**New Features** . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **23-2**
New Demo . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **23-2**
Support For Function Handles . . . . . . . . . . . . . . . . . . . . . . . . **23-2**
Documentation Enhanced . . . . . . . . . . . . . . . . . . . . . . . . . . . **23-2**

**Major Bug Fixes** . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **23-3**
imshow Fixes . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **23-3**
bwlabel Segmentation Violation Eliminated . . . . . . . . . . . . . . **23-3**
dilate And erode Return Correct Answers . . . . . . . . . . . . . . . **23-3**
freqz2 Fixes . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **23-3**
fspecial Function's 'LoG' Option . . . . . . . . . . . . . . . . . . . . . . **23-3**
Improved Display for imcrop, improfile, and roipoly . . . . . . . . **23-4**

# 24 Mapping Toolbox 1.2 Release Notes

**New Features** . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **24-2**
Higher Resolution Atlas Data . . . . . . . . . . . . . . . . . . . . . . . . . **24-2**
External Data Interface Improved . . . . . . . . . . . . . . . . . . . . . **24-2**
New Interactive Interfaces . . . . . . . . . . . . . . . . . . . . . . . . . . . **24-3**
New Analysis Functions for Geographic Data . . . . . . . . . . . . . **24-3**
Other New Functions . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **24-4**

# 25 MATLAB C/C++ Graphics Library 2.1 Release Notes

**New Features** . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **25-2**
Additions . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **25-2**
Unsupported MATLAB Features . . . . . . . . . . . . . . . . . . . . . . . **25-3**

# 26 MATLAB C/C++ Math Library 2.1 Release Notes

**New Features** . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **26-2**
New Features . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **26-2**
Unsupported MATLAB Features . . . . . . . . . . . . . . . . . . . . . . . **26-2**
Documentation Error . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **26-3**

**Upgrading from an Earlier Release** . . . . . . . . . . . . . . . . . . . . **26-4**
Recompile Your Programs . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **26-4**
Changed Features . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **26-4**
For More Information . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **26-5**

# 27 MATLAB Compiler 2.1 Release Notes

**New Features** . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **27-2**
Optimizations . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **27-2**
Dynamically Linking in MEX-Files in the Stand-Alone
Environment . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **27-3**
MATLAB Add-In for Visual Studio® . . . . . . . . . . . . . . . . . . . . **27-4**
mlib Files . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **27-4**
Additional Datatype Support . . . . . . . . . . . . . . . . . . . . . . . . . . **27-4**
Improved Support for load and save . . . . . . . . . . . . . . . . . . . . **27-5**
Faster C/C++ Math Library Applications . . . . . . . . . . . . . . . . . **27-5**
Printing from the C/C++ Graphics Library . . . . . . . . . . . . . . . **27-5**
Additional Language Support . . . . . . . . . . . . . . . . . . . . . . . . . . **27-5**

**Third Party Compilers** . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **27-6**
Exception Handling . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **27-6**

**Upgrading from an Earlier Release** . . . . . . . . . . . . . . . . . . . . **27-8**
Changed Options . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **27-8**
Obsolete Options . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **27-8**

# 28 MATLAB Report Generator 1.1 Release Notes

**New Features** . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **28-2**
New Components . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **28-2**
Enhancements to Components . . . . . . . . . . . . . . . . . . . . . . . . . **28-2**
Property Table Enhancement . . . . . . . . . . . . . . . . . . . . . . . . . . **28-3**
Summary Table Enhancement . . . . . . . . . . . . . . . . . . . . . . . . . **28-3**
Using MATLAB Variables with Components . . . . . . . . . . . . . . **28-3**

**Upgrading from an Earlier Release** . . . . . . . . . . . . . . . . . . . . **28-4**
Use of Existing SGML Files . . . . . . . . . . . . . . . . . . . . . . . . . . . **28-4**

# 29 MATLAB Runtime Server 6.0 Release Notes

**New Features** . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **29-2**

**Upgrading from an Earlier Release** . . . . . . . . . . . . . . . . . . . . **29-3**

**Known Software and Documentation Problems** . . . . . . . . . **29-4**

# 30 MATLAB Web Server 1.2 Release Notes

**New Features** . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **30-2**
Support for Linux Servers . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **30-2**
New Security Feature . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **30-2**

# 31 Motorola DSP Developer's Kit 1.1 Release Notes

**Introduction to the Motorola DSP Developer's Kit** . . . . . . **31-2**
For More Information . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **31-2**

**New Features** . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **31-4**
Matrix and Frame Support . . . . . . . . . . . . . . . . . . . . . . . . . . . **31-4**

**Known Software and Documentation Problems** . . . . . . . . . **31-5**
Interactive Mode Limitations . . . . . . . . . . . . . . . . . . . . . . . . . **31-5**
Stand-Alone GUI Simulator May Hang (Windows NT) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **31-6**
Other Solaris-Specific Limitations . . . . . . . . . . . . . . . . . . . . . **31-6**
Documentation Updates . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **31-7**

# 32 Neural Network Toolbox 4.0 Release Notes

**New Features** . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **32-2**
Control System Applications . . . . . . . . . . . . . . . . . . . . . . . . . . **32-2**
Visual Interface . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **32-2**
New Training Functions . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **32-3**
Design of General Linear Networks . . . . . . . . . . . . . . . . . . . **32-3**
Improved Early Stopping . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **32-3**
Generalization and Speed Benchmarks . . . . . . . . . . . . . . . . . **32-4**
Demonstration of a Sample Training Session . . . . . . . . . . . . **32-4**

**Platform Limitations for HP and IBM** . . . . . . . . . . . . . . . . . **32-5**

**Upgrading from an Earlier Release** . . . . . . . . . . . . . . . . . . . **32-6**
Use of trainwb and trainwb1 Not Recommended . . . . . . . . . . **32-6**

# 33 Optimization Toolbox 2.1 Release Notes

**New Features** . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **33-2**
Toolbox Speed . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **33-2**
Function Handles . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **33-2**
Large Structured Problems . . . . . . . . . . . . . . . . . . . . . . . . . . . **33-2**
Functions with New or Changed Capabilities . . . . . . . . . . . . **33-3**

**Major Bug Fixes** . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **33-4**

**Upgrading from an Earlier Release** . . . . . . . . . . . . . . . . . . . **33-5**

# 34 Power System Blockset 2.1 Release Notes

**New Features** . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **34-2**
Faster Circuit Topology Analysis . . . . . . . . . . . . . . . . . . . . . . **34-2**
Simulation Enhancements . . . . . . . . . . . . . . . . . . . . . . . . . . . . **34-2**
Measurement Enhancements . . . . . . . . . . . . . . . . . . . . . . . . . . **34-2**
Machines Enhancements . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **34-3**
Three-Phase Transformer Blocks Added . . . . . . . . . . . . . . . . **34-3**
Power Electronics Enhancements . . . . . . . . . . . . . . . . . . . . . . **34-4**
Three-Phase Library Enhancements . . . . . . . . . . . . . . . . . . . **34-4**
Elements Library Enhancements . . . . . . . . . . . . . . . . . . . . . . **34-4**
New Control Blocks . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **34-5**
Powergui . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **34-5**
New Demos and Improved Documentation . . . . . . . . . . . . . . **34-5**

# 35 Real-Time Windows Target 2.0 Release Notes

**New Features** . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **35-2**
I/O Device Driver Architecture . . . . . . . . . . . . . . . . . . . . . . . . **35-2**
External Mode Interface . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **35-3**

C Compiler Support . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **35-3**
I/O Device Driver Support . . . . . . . . . . . . . . . . . . . . . . . . . . . . **35-3**

**Upgrading from an Earlier Release** . . . . . . . . . . . . . . . . . . . . **35-5**
Reinstall the Real-Time Windows Kernel . . . . . . . . . . . . . . . . **35-5**
Enter New External Mode Interface Filename . . . . . . . . . . . . **35-5**
Remove I/O Adapter Blocks from Your Simulink Model . . . . . **35-6**
Replace I/O Driver Blocks in Your Simulink Model . . . . . . . . . **35-6**

**Known Software and Documentation Problems** . . . . . . . . . **35-7**
Windows 2000 Not Currently Supported . . . . . . . . . . . . . . . . . **35-7**
Undocumented Procedure for Configuring I/O Drivers . . . . . . **35-7**

**36 Real-Time Workshop Ada Coder 4.0 Release Notes**

**New Features** . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **36-2**
Data-Logging Format Support . . . . . . . . . . . . . . . . . . . . . . . . . **36-2**
Ada83 Support . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **36-2**
S-Function Support . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **36-2**
Block Comment Generation . . . . . . . . . . . . . . . . . . . . . . . . . . **36-3**

**37 Real-Time Workshop Embedded Coder 1.0**

**Introduction to the Real-Time Workshop Embedded Coder** . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **37-2**
Real-Time Workshop Embedded Coder Demos . . . . . . . . . . . . **37-3**

# 38 Requirements Management Interface 1.0 Release Notes

**Introducing the Requirements Management Interface** . . . **38-2**

**Platform Limitations for HP and IBM** . . . . . . . . . . . . . . . . . . **38-3**

# 39 Signal Processing Toolbox 5.0 Release Notes

**New Features** . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **39-2**
Filter Design & Analysis Tool (FDATool) . . . . . . . . . . . . . . . . . **39-3**
SPTool . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **39-4**
Spectrogram Demo . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **39-5**
Automatic Order Adjustment in FIR Filter Functions . . . . . . **39-5**
grpdelay . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **39-6**
hilbert . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **39-6**
lpc . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **39-6**
xcorr . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **39-6**

**Major Bug Fixes** . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **39-7**

**Upgrading from an Earlier Release** . . . . . . . . . . . . . . . . . . . . **39-8**
Changes to remez . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **39-8**

# 40 Simulink Performance Tools 4.0 Release Notes

**Introduction to the Simulink Performance Tools** . . . . . . . **40-2**
The Simulink Accelerator . . . . . . . . . . . . . . . . . . . . . . . . . . . **40-2**
The Model Coverage Tool . . . . . . . . . . . . . . . . . . . . . . . . . . . **40-2**
The Model Differences Tool . . . . . . . . . . . . . . . . . . . . . . . . . **40-2**
The Model Profiler . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **40-2**

**Platform Limitations for HP and IBM** . . . . . . . . . . . . . . . . . . **40-3**

**Known Software and Documentation Problems** . . . . . . . . . **40-4**
Simulink Model Differences Tool . . . . . . . . . . . . . . . . . . . . . . . **40-4**

# 41 Simulink Report Generator 1.1 Release Notes

**New Features** . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **41-2**
New Simulink Components . . . . . . . . . . . . . . . . . . . . . . . . . . . **41-2**
New Stateflow Components . . . . . . . . . . . . . . . . . . . . . . . . . . . **41-3**
New Fixed-Point Components . . . . . . . . . . . . . . . . . . . . . . . . . **41-3**
Enhancements to Components . . . . . . . . . . . . . . . . . . . . . . . . **41-5**

**Upgrading from an Earlier Release** . . . . . . . . . . . . . . . . . . . . **41-6**
Use of Existing SGML Files . . . . . . . . . . . . . . . . . . . . . . . . . . **41-6**

# 42 Spline Toolbox 3.0 Release Notes

**New Features** . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **42-2**
Spline Tool Provides a Visual Interface to the Spline Toolbox **42-2**
Automatic Knot Choice Simplifies Use of spapi and spap2 . . . **42-2**
Automatic Smoothing Parameter Choice Simplifies
Use of csaps . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **42-2**
Use of Rational Splines . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **42-3**
B-Spline Visual Interface . . . . . . . . . . . . . . . . . . . . . . . . . . . . **42-3**
Other New Functions . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **42-3**
Other Enhancements . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **42-3**

**Major Bug Fix** . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **42-5**
spmak and ppmak Use Size Argument to Correctly Construct
Multivariate Splines . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **42-5**

**Upgrading from an Earlier Release** . . . . . . . . . . . . . . . . . . . . **42-7**
optknt and newknt Output No Longer Needs To Be Run Through augknt . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **42-7**

**Known Software and Documentation Problems** . . . . . . . . . **42-8**
Correction for splinetool Reference Page . . . . . . . . . . . . . . . . . **42-8**

## 43 Stateflow Coder 4.0 Release Notes

**New Features** . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **43-2**
Improved Code Generation . . . . . . . . . . . . . . . . . . . . . . . . . . . . **43-2**
Temporal Logic . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **43-2**

## 44 Statistics Toolbox 3.0 Release Notes

**New Features** . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **44-2**
Summary of Enhancements . . . . . . . . . . . . . . . . . . . . . . . . . . . . **44-2**
New Functions . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **44-3**
New Demos . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **44-4**
New Sample Data Files . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **44-5**
Updated Functions for ANOVA-Type Tables . . . . . . . . . . . . . . **44-5**
Other Updated Functions . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **44-6**

45 Symbolic Math Toolbox 2.1.2 Release Notes

**Major Bug Fixes** . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **45-2**

46 System Identification Toolbox 5.0 Release Notes

**New Features** . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **46-2**
Object-Based Design . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **46-2**
Advanced Feature Enhancements . . . . . . . . . . . . . . . . . . . . . . **46-2**

**Upgrading from an Earlier Release** . . . . . . . . . . . . . . . . . . . **46-4**
Theta Models No Longer Supported . . . . . . . . . . . . . . . . . . . . **46-4**

47 Wavelet Toolbox 2.0 Release Notes

**New Features** . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **47-2**
New Visual Interface Tools . . . . . . . . . . . . . . . . . . . . . . . . . . . **47-2**
New Wavelets . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **47-5**
New Functions . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **47-6**
Other Enhancements . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **47-8**

**Major Bug Fixes** . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . **47-10**

**Upgrading from an Earlier Release** . . . . . . . . . . . . . . . . . . **47-11**
Default Extension Mode . . . . . . . . . . . . . . . . . . . . . . . . . . . . **47-11**
Wavelet Packet Structures and Obsolete Functions . . . . . . . **47-11**

**Known Software and Documentation Problems** . . . . . . . . **47-12**
Graphics and Equations in the HTML Documentation . . . . . **47-12**

# 48 xPC Target 1.1 Release Notes

**Introduction to xPC Target and xPC Target Embedded Option** . . . . . **48-2**
xPC Target . . . . . **48-2**
xPC Target Embedded Option . . . . . **48-2**

**New Features** . . . . . **48-3**
Web Browser Interface . . . . . **48-3**
Target PC Command Line Interface . . . . . **48-3**
MATLAB Command Line Interface . . . . . **48-3**
Signal Monitoring . . . . . **48-3**
I/O Device Driver Support . . . . . **48-4**

**Known Software and Documentation Problems** . . . . . **48-5**
Uploading Data with External Mode . . . . . **48-5**
Target Command Line Interface . . . . . **48-5**
Memory-Mapped Devices . . . . . **48-5**

# Introducing Release 12

**What's New in Release 12?** . . . . . . . . . . . . . . . . 1-2
General Themes of Release 12 . . . . . . . . . . . . . . . 1-2
Products with Major Upgrades . . . . . . . . . . . . . . . 1-4
Products with Minor Upgrades . . . . . . . . . . . . . . . 1-15
New Products (Since Release 11.0) . . . . . . . . . . . . 1-16

**Platform Limitations** . . . . . . . . . . . . . . . . . . 1-17
Differences for the HP and IBM Platforms . . . . . . . . . 1-17
Problems Entering Accented Characters on Linux . . . . . 1-20
Problems Entering Accented Characters on Unix Platforms . 1-20

**Installation Notes** . . . . . . . . . . . . . . . . . . . 1-21

**Linux Distribution Notes** . . . . . . . . . . . . . . . 1-22

**For More Information About What's New in Release 12** 1-23
Printable Version of the Release Notes . . . . . . . . . . 1-23

## What’s New in Release 12?

Release 12 includes major updates to MATLAB®, Simulink®, and many other products from The MathWorks. In addition, several new products have been added since Release 11.

This section:

- Provides a general overview of some general themes of Release 12
- Highlights the products with major new versions for Release 12
- Lists the products with minor upgrades
- Lists the new products

### General Themes of Release 12

Release 12 includes many new features and products, as well as a number of bug fixes. Much of what’s new for Release 12 was driven by our desire to:

- Make the software easier to use
- Extend connections to and from the software to other environments
- Improve MathWorks code generation tools

#### Easier to Use Products

Probably the most immediately noticeable change aimed at making MATLAB an easier to use development environment is the new desktop, an integrated set of development tools.

In addition to the new visual interface (graphical user interface (GUI)) for MATLAB, a number of other products have new or improved interfaces. Usability testing was performed as an integral part of the development process for many of the products. Some of the products with major interface enhancements include:

- Simulink
- Real-Time Workshop®
- Control System Toolbox
- Fixed-Point Blockset
- Signal Processing Toolbox

- Filter Design Toolbox
- Power System Blockset
- Neural Network Toolbox
- Spline Toolbox

Release 12 also includes new HTML-based online help that is integrated into the desktop environment, and is accessible from throughout the product via **Help** menus, **Help** buttons, and some context-sensitive help (for example, right-clicking on a block brings up a context-menu from which you can get help on the block). The documentation for several products provides more material to help new users get started using the products more quickly and effectively. For example, you can perform several types of searches, access examples via indices of examples, and take advantage of numerous links to related sections.

## Extended Connections to and from the MathWorks Products

Release 12 provides more ways for you to connect between the MathWorks products and other products, software environments, or hardware environments. For example:

- MATLAB provides a Java interface, which allows you to use MATLAB functions to bring Java classes into the MATLAB environment, construct objects from those classes, call methods on the Java objects, and save Java objects for later reloading.
- MATLAB also provides a serial port feature, which supports direct access to peripheral devices such as modems, printers, and scientific instruments that you connect to your computer's serial port.
- MATLAB, Simulink, and Stateflow® provide a revision control interface for connecting to an external configuration management system.
- The new Requirements Management Interface supports tools such as QSS's DOORS.
- The Data Acquisition Toolbox adds ComputerBoards support.
- The new Instrument Control Toolbox adds GPIB and VISA support.
- The Real-Time Windows Target supports additional device drivers.
- The xPC Target supports remote Web access and additional device drivers.

### Improved Code Generation

Release 12 improves the code generation capabilities and performance of several products used for design automation, including:

- The Simulink Accelerator uses code generation technology to improve simulation speed.
- The Real-Time Workshop has added matrix code generation, faster code generation, and code optimizations.
- The Real-Time Workshop Ada Coder provides Ada83 code generation and full logging support.
- The Stateflow Coder has dramatically improved code efficiency.
- The Fixed-Point Blockset added code generation performance optimizations and enhanced code generation for difficult slope and bias cases.

## Products with Major Upgrades

Release 12 includes major version upgrades to many products, highlighted in the following sections of these *Release Notes*:

- "MATLAB 6.0"
- "MATLAB Runtime Server 6.0"
- "MATLAB Compiler 2.1"
- "Simulink 4.0"
- "Stateflow 4.0"
- "Stateflow Coder 4.0"
- "Real-Time Workshop 4.0"
- "Real-Time Workshop Ada Coder 4.0"
- "Real-Time Windows Target 2.0"
- "Communications Blockset 2.0"
- "Communications Toolbox 2.0"
- "Control System Toolbox 5.0"
- "Data Acquisition Toolbox 2.0"
- "DSP Blockset 4.0"
- "Fixed-Point Blockset 3.0"
- "Neural Network Toolbox 4.0"

- "Power System Blockset 2.0"
- "Signal Processing Toolbox 5.0"
- "Spline Toolbox 3.0"
- "Statistics Toolbox 3.0"
- "System Identification Toolbox 5.0"
- "Wavelet Toolbox 2.0"

## MATLAB 6.0

Major enhancements to MATLAB 6.0 include:

- MATLAB desktop – a new user interface for managing files, tools, and applications associated with MATLAB. The desktop includes:
  - Command History for viewing or using previously run functions
  - Launch Pad for launching product tools, demos, and documentation
  - Help for viewing and searching documentation for the MATLAB product family
  - Current Directory browser for viewing and operating on MATLAB files
- Other enhancements to the MATLAB environment, including:
  - Easier printing, using redesigned **Page Setup** (Microsoft Windows) and **Print** (UNIX) dialog boxes
  - Data statistics and basic fitting (curve fitting) added to the figure window
  - A new context-based **Tab** completion feature that allows you to complete partially entered commands with the **Tab** key
  - Import Wizard for importing and exporting data
  - Source Control System Interface for interfacing MATLAB, Simulink, and Stateflow files with your source control system
- An enhanced, more stable, license manager, including:
  - A new graphical control panel to simplify license manager administration
  - Support for multiple license managers running on the same machine
- Improved performance (from twice to eight times faster) through:
  - Use of the FFTW library for fast Fourier transform computations
  - Use of the LAPACK library for matrix computations

- Language and math enhancements, including
  - Use of the LAPACK library for matrix computations
  - Conversion of the `eigs` function to use the ARPACK library
  - New functions for n-dimensional Delaunay, Voronoi, and convex hull computations, as well as hypersurface fitting
  - New functions for solving boundary value ordinary differential equation (ODE) problems and initial-boundary value problems for partial differential equation (PDE) problems
  - Function handle data type for better performance, accessing more functions, and passing function references (similar to C++ function pointers)
  - The new `continue` statement, which passes control to the next iteration of a `for` or `while` loop, allows more flexible control of program flow and facilitates conversion of algorithms from other languages such as Fortran
- Graphics enhancements, including:
  - Transparency
  - Handle Graphics® Property Editor
  - 3-D visualization extensions
  - OpenGL is now the default renderer. OpenGL is generally faster than the Painters or Z-buffer renderers and accesses graphics hardware on some systems
- MATLAB interface to Java
- Serial port interface for direct access to peripheral devices (e.g., modems)

### MATLAB Runtime Server 6.0

All functionality from MATLAB 6.0 that is required to develop a runtime application is available in the MATLAB Runtime Server 6.0.

### MATLAB Compiler 2.1

The MATLAB Compiler 2.1 has been enhanced so that it now supports:

- Several optimizations including folding scalar and nonscalar valued `mxarray` constants, as well as `for` loops with integer starts and increments
- MLIB files, which make it possible to produce a shared library out of a toolbox and then compile M-files that make calls into that toolbox
- `int` and `uint` datatypes
- Partial support for `eval` and `input`
- `load` and `save` without any variable names
- MEX-files in the stand-alone environment
- The MATLAB Visual Studio add-in, which integrates the MATLAB Compiler 2.1 into Visual C/C++ 5.x or 6.x
- Improved performance of all C/C++ Math Library applications
- Printing from the MATLAB C/C++ Graphics Library

### Simulink 4.0

Major enhancements to Simulink 4.0 include:

- Several new or enhanced tools to facilitate model development and refinement, including:
  - Library Browser enhanced
  - **Find** dialog box added for searching Simulink models and Stateflow charts for objects satisfying specified search criteria
  - A graphical user interface for the Simulink Debugger
  - Single or multiwindow display options, for controlling whether you see the parent and subsystems, or just the current subsystem
  - The icon for a block now optionally displays a number indicating the order in which it is executed relative to other blocks in the diagram.
  - Context menus for blocks and signals, including help
  - The new **Simulink Preferences** dialog box allows you to specify default settings for many diagramming and simulation options.
- Matrix operations on matrix signals are now supported for several blocks.
- You can now use simulation data objects to specify simulation and code generation options on a per-parameter and per-signal basis.

- Enhancements for working with subsystems, including:
  - The enhanced **Configurable Subsystem** menu allows you to choose which block the subsystem represents.
  - Atomic (true, as opposed to virtual) subsystem support
  - Subsystem encryption (requires Real-Time Workshop)
- Library enhancements for editing and instantiation
- Ada S-functions are supported.
- Improved Simulink linearization

**Note** The Simulink Performance Tools, introduced with Release 12, provide additional tools for working in the Simulink environment, including a Simulink Accelerator for faster simulation of models, via compiled code, as well as a Model Profiler, Model Differences tool, and a Test Coverage tool.

### Stateflow 4.0

The major enhancements to Stateflow 4.0 include:

- Subcharts, for creating charts within charts
- Graphical functions (which you define with a flow graph), which are easier to work with than textual custom code functions.
- Temporal logic for expressing temporal order for Stateflow events and to enforce temporal properties such as upper and lower bound requirements in state transition systems
- You can now initialize all Stateflow data to the workspace and copy it back to the workspace at the end of a simulation.

### Stateflow Coder 4.0

The Stateflow Coder 4.0 code generation has been significantly improved:

- The code looks hand-written.
- ROM and RAM size rivals hand-written code.
- Code generation is faster.

The new temporal logic feature is supported.

## Real-Time Workshop 4.0

Major enhancements to the Real-Time Workshop include:

- General code generation enhancements, including:
  - Faster code generation
  - Matrix code generation
  - Nonvirtual subsystem code generation, which replaces **Function management** code generation options; the new approach is more general and flexible
  - Subsystem-based code generation support
- External mode enhancements, including:
  - Simpler target program monitoring
  - External mode is now supported in the Dials & Gauges Blockset, Display blocks, To Workspace blocks, XY Graph blocks, and S-functions. External mode Signal Viewing Subsystems (atomic subsystems that encapsulate processing and viewing signals received from the target system) have been added.
- S-function target support for variable step solver
- Real-Time Workshop Embedded Coder replaces and enhances them embedded real-time (ERT) target. Enhancements include support for singletasking, multirate models.
- Target Language Compiler™ (TLC) enhancements, including:
  - Complete parsing of the TLC file just before execution
  - Faster block parameter generation and other speed improvements
  - New TLC Debugger commands
  - New TLC Profiler
  - Reduced `model.rtw` file size

### Real-Time Workshop Ada Coder 4.0

Major enhancements to the Real-Time Workshop Ada Coder 4.0 include:

- Support for Ada83, in addition to Ada95, code generation
- Simulink now supports noninlined Ada S-functions. For Real-Time Workshop code generation purposes, you can create a wrapper S-function that calls an Ada S-function.
- Full logging support (in Ada95 only) and function-call subsystems

### Real-Time Windows Target 2.0

Major enhancements to the Real-Time Windows Target 2.0 include:

- Support for Microsoft Visual C, Windows 2000, and Windows NT 5
- Additional device drivers
- Frame-based I/O support
- Integration with the Dial & Gauges Blockset and other external mode features

### Communications Blockset 2.0

Major enhancements to the Communications Blockset 2.0 include:

- New digital modulation libraries, including QAM (square and cross), PSK, DPSK, and CPM
- New multipath Rayleigh and Rician fading channel blocks
- A new APP soft-output decoder for convolutional codes and a more general trellis description for the convolutional encoder and Viterbi decoders
- New interleaving libraries that include both block and convolutional interleavers
- Significantly rewritten documentation

### Communications Toolbox 2.0

Major enhancements to the Communications Toolbox 2.0 include:

- New convolutional encoding and Viterbi decoding functions (these functions were implemented as Simulink blocks in the previous version of the Communications Toolbox)
- Significantly rewritten documentation

---

**Note** The Simulink blocks formerly included in the Communications Toolbox now form the basis of a separate product, the Communications Blockset.

---

### Control System Toolbox 5.0

Major enhancements to the Control System Toolbox 5.0 include:

- New SISO Design Tool for performing compensator design using root locus and frequency-domain (Bode) techniques
- LTI Viewer enhancements, including data markers for displaying data and system information, better grids, and sharper plots
- Tools to set preferences and customize plots
- Algorithmic enhancements, including:
  - Stability margins (all crossovers, worst-case, delay margins)
  - Sharper root locus plots
- New *Getting Started with the Control System Toolbox* manual, which introduces the main features of the toolbox through extended examples

### Data Acquisition Toolbox 2.0

The Data Acquisition Toolbox 2.0 now supports ComputerBoards hardware.

### DSP Blockset 4.0

Major enhancements to the DSP Blockset 4.0 include:

- Easier generation and processing of matrix signals
- Frame-based signals support and matrix signals support has been added to many blocks.
- Operations on both row and column dimensions of input are now supported for several blocks.

### Fixed-Point Blockset 3.0

Major enhancements to the Fixed-Point Blockset 3.0 include:

- Fixed-Point Blockset Interface tool improvements, including better logging and information for scaling and ranges, as well as adding targeting options for embedded systems
- Code generation optimizations, including simple conversions, summations, and other simplifications
- New blocks to mirror Simulink's built-in capability (e.g., Multiport Switch, Bitwise Operator, Dead Zone, etc.)

### Neural Network Toolbox 4.0

Major enhancements to the Neural Network Toolbox 4.0 include:

- A new visual interface for:
  - Creating networks
  - Entering data
  - Initializing, training, and simulating networks
  - Exporting and importing data
- Examples of control system applications

### Power System Blockset 2.0

Major enhancements to the Power System Blockset 2.0 include:

- Visual interface improvements
- Faster setup for power system simulation
- Support for descretized power system models, using the fixed-step trapezoidal method
- Enhanced and new blocks relating to machines, universal transformers, power electronics, and control
- Expanded tutorial section of the *Power System Toolbox User's Guide*, with test cases and customization tips

### Signal Processing Toolbox 5.0

Major enhancements to the Signal Processing Toolbox 5.0 include:

- New Filter Design & Analysis Tool (FDATool), for designing and analyzing filters
- SPTool now has a toolbar, which includes quick access to the markers (rulers). Marker readouts have been improved. The Signal Browser now supports playing portions of a signal.
- FIR filter now includes automatic order adjustment.
- Faster, more robust cross-correlation (with the `xcorr` function)

### Spline Toolbox 3.0

Major enhancements to the Spline Toolbox 3.0 include:

- The `splinetool` command invokes a new visual interface for easy, interactive creation and manipulation of various types of splines.
- Rational splines can now be generated with `rsmak` and `rpmak` and operated on by function function commands (e.g., `fn2fm`, `fnbrk`, `fnint`, etc.).
- Automatic knot generation is supported with `spapi` and `spap2`.
- The documentation has been rewritten with more examples in the tutorial section and a new glossary.

### Statistics Toolbox 3.0

Major enhancements to the Statistics Toolbox 3.0 include:

- Expanded support for linear models, including:
  - N-way analysis of variance for unbalanced data
  - Multivariate analysis of variance, with graphics functions for examining multivariate data
  - Multiple comparison of means and other estimates
  - Analysis of covariance, with a graphical user interface
- Generalized linear models
- Robust regression
- Distribution testing and plotting
- Better support for missing data and for grouping variables
- Importing numeric and text data from tab-delimited files

### System Identification Toolbox 5.0

Major enhancements to the System Identification Toolbox 5.0 include:

- New data and model objects
- Simplified command line interface for manipulating time series and identified models

### Wavelet Toolbox 2.0

Major enhancements to the Wavelet Toolbox 2.0 include:

- New visual interface tools for several kinds of analysis, including:
  - Complex continuous wavelet 1-D transforms
  - Wavelet coefficients selection tool for 1-D and 2-D
  - Signal and image de-noising using the stationary wavelet transform
  - Signal and image extension and truncation
  - Scale to frequency converter
- New wavelets for complex and real wavelet families
- New data representation for wavelet packets using MATLAB objects
- Many new functions (e.g., for computing thresholds)

## Products with Minor Upgrades

Release 12 includes several products that had minor upgrades (e.g., bug fixes, use of Release 12 features, etc.), including:

- Communications Toolbox 2.0
- Database Toolbox 2.1
- Fuzzy Logic Toolbox 2.1
- Image Processing Toolbox 2.2.2
- Mapping Toolbox 1.2
- MATLAB C/C++ Graphics Library 2.1
- MATLAB C/C++ Math Library 2.1
- MATLAB Report Generator 1.1
- MATLAB Web Server 1.2
- Optimization Toolbox 2.1
- Simulink Report Generator 1.1
- Symbolic Math Toolbox 2.1.2

## New Products (Since Release 11.0)

The following products are being introduced with Release 12:

- Communications Blockset 2.0
- Filter Design Toolbox 2.0
- Instrument Control Toolbox 1.0
- Simulink Performance Tools 4.0
- Real-Time Workshop Embedded Coder

In addition, Release 12 includes a number of products that have been introduced since Release 11.0 (released in Web-downloadable form and/or on the Release 11.1 CD):

- CDMA Reference Blockset 1.0.2
- Datafeed Toolbox 1.2
- Dials & Gauges Blockset 1.1
- Financial Time Series Toolbox 1.0
- Financial Derivatives Toolbox 1.0
- Motorola DSP Developer's Kit 1.1
- GARCH Toolbox 1.0
- Requirements Management Interface 1.0.1
- xPC Target and xPC Target Embedded Option 1.1

## Platform Limitations

For Release 12, all the features described in these Release Notes and in the Release 12 documentation are supported in *full* for the following platforms:

- Microsoft Windows 95, Windows 98, Windows 2000, Windows Millenium, and Windows NT
- Compaq Alpha
- Linux
- SGI
- Sun Solaris

---

**Note** As in Release 11, there are some minor differences between Windows and UNIX for specific tools, as documented for those tools.

---

For details about each of these platforms, see the Support page of the MathWorks Web page, at `http://www.mathworks.com`.

### Differences for the HP and IBM Platforms

Because of problems with the Java VM, there are limitations for MATLAB on the HP and IBM platforms.

| Platform | Limitations |
|---|---|
| HP 11.0 (HP UX)<br><br>IBM | Graphical user interfaces that are based on Java, such as the MATLAB desktop, are not available. |
| HP 10.2 (HP 700) | Java-based tools and applications are not supported. |

Alternatives for most tools are available and are documented in the following table. The table summarizes the products with limitations for these platforms. For details on limitations and the alternatives, see the sections of the online version of the *Release Notes* for the product(s) of interest to you. If a product is *not* listed in the following table, it has no platform limitations.

**Table 1-1: Products with Limitations on HP and IBM Platforms**

| Product | Limitations on HP and IBM Platforms |
|---|---|
| MATLAB | |
| • Development Environment | The MATLAB desktop and many of its tools are not supported. For details, see “Development Environment Limitations” in the MATLAB section of the online *Release Notes*. |
| • Mathematics | The Basic Fitting interface is not supported. Instead, use the command line curve fitting capabilities. |
| • Graphics | The Property Editor and Data Statistics tool are not supported. For details, see “Graphics and Printing Limitations” in the MATLAB section of the online *Release Notes*. |
| • Printing | The Release 11 **Page Setup**, **Print Setup**, and **Print** dialog boxes are used. |
| • GUIDE | Not supported. Instead, use functions to create graphical user interfaces. |
| Simulink | The following Simulink features are not supported:<br>• Simulink Editor’s **Find** dialog. Use the `find_system` command instead.<br>• GUI interface to the Simulink Debugger. Use the command-line interface instead.<br>• The **View Changes** dialog box for modified library links. See “Platform Limitations for HP and IBM” in the Simulink section of the online *Release Notes* for an alternative.<br>• **Parameter** dialog for the Configurable Subsystem block. Use the `set_param` command instead to set the block’s parameters. |

**Table 1-1: Products with Limitations on HP and IBM Platforms (Continued)**

| Product | Limitations on HP and IBM Platforms |
|---|---|
| Real-Time Workshop | Instead of the **Model Parameter Configuration** dialog box, the **Tunable Parameters** dialog box is used. |
| Control System Toolbox | The Toolbox Preferences Editor and Response Property Editor are not supported. The LTI Viewer Preferences Editor is supported with a reduced set of features (no tools to set preferences for grids, fonts, colors, or phase wrapping).<br><br>The following features are not supported for the SISO Design Tool:<br><br>• Preference and property editing<br>• Compensator Format editing<br>• Storing and retrieving compensators<br><br>In addition, the SISO Tool Export window has fewer choices available for export. |
| Database Toolbox | Not supported on the HP 10.2 (HP 700) platform. |
| DSP Blockset | Same limitations as Simulink. |
| Filter Design Toolbox | Not supported on the HP 10.2 (HP 700) platform. |
| MATLAB C/C++ Graphics Library | Not supported on the IBM RS/6000 platform. |
| Neural Network Toolbox | The Neural Network Tool, a graphical user interface for the Neural Network Toolbox, is not supported. You can achieve the same functionality by using the MATLAB Command Window. |

**Table 1-1: Products with Limitations on HP and IBM Platforms (Continued)**

| Product | Limitations on HP and IBM Platforms |
|---|---|
| Requirements Management Interface | Not supported. |
| Simulink Performance Tools | The Model Differences Tool is not supported. |

## Problems Entering Accented Characters on Linux

On Linux platforms, using an accent key to accent a character in a two-stroke sequence in a MATLAB window, such as ^e for ê or ~n for ñ, may cause MATLAB to hang. To avoid this problem in MATLAB, use `matlab -nodesktop`. With Simulink, avoid the use of these keystrokes. To use characters such as the caret ^ when it is mapped as a dead-accent key, you must paste it into the MATLAB IDE from a non-MATLAB window such as an xterm or use the `power` and `mpower` functions directly.

This problem will be addressed in the next release of MATLAB.

## Problems Entering Accented Characters on UNIX Platforms

Keys for accented characters, such as Ü on German keyboards and Å on Swedish keyboards are ignored in the IDE. Pasting these characters into MATLAB windows results in the character value 128 instead of the actual character value. For example, pasting the string `double('ÅÄÖåäöü')` into MATLAB appears as:

```
>> double('       ')
ans =
128   128   128   128   128   128   128
```

This problem will be addressed in the next release of MATLAB.

## Installation Notes

On Windows 2000, an administrator needs to install and *start* MATLAB before other users run it. If the administrator installs but does not start MATLAB, other users get the following error message about writing to the registry when they start MATLAB. (They can run MATLAB but must first dismiss the error message.)

```
Unable to set Registry value
CLSID\{B3044B60-97FF-11D1-8146-00600815A7AD}\LocalServer32.
You may not have sufficient priveleges. Rerun MATLAB as a user
with Admin access.
```

## Linux Distribution Notes

As of this writing, the following distributions are known to work:

- Redhat 6.2
- Mandrake 7.1
- SuSE 6.4
- Debian 2.1 & 2.2

Slackware 7.0 & 7.1 and Redhat 7.0 appear to have issues with Java. Use

```
matlab -nojvm
```

to start MATLAB. Check with `http://www.mathworks.com/support` for the latest information.

## For More Information About What's New in Release 12

For more detailed information about each product that is included in Release 12, see the sections of the online version of the *Release Notes* for the product(s) of interest to you. In those sections, as applicable, you can find out about a product's:

- New features
- Major bug fixes
- Platform limitations
- Issues relating to upgrading from an earlier release
- Known software and documentation problems

The online *Release Notes* include links to the rest of the online documentation, where you can get detailed information about how to use the new and enhanced features.

### Printable Version of the Release Notes

A printable version of these *Release Notes* is available in PDF format.

# MATLAB Software Acknowledgments

MATLAB and its associated products incorporate the following third-party software:

ARnoldi PACKage (ARPACK)

Rich Lehoucq, Kristi Maschhoff, Danny Sorensen, and Chao Yang
`http://www.caam.rice.edu/software/ARPACK`

Automatically Tuned Linear Algebra Software (ATLAS)

R. Clint Whaley and Jack Dongarra
`http://www.netlib.org/atlas`

`fft` and related MATLAB functions are based on the FFTW library

developed by Matteo Frigo and Steven G. Johnson
© Copyright 1997-1999 Massachusetts Institute of Technology. All rights reserved.
Used under terms of a commercial license
`http://www.fftw.org`

HDF capability in the functions `imread`, `imwrite`, `imfinfo`, and `hdf` is based on code of which portions were developed at

The National Center for Supercomputing Applications at the University of Illinois at Urbana-Champaign.

JPEG capability in the functions `imread`, `imwrite`, `imfinfo`, `print`, and `saveas`

This software is based in part on the work of the Independent JPEG Group.

Linear Algebra PACKage (LAPACK)

`http://www.netlib.org/lapack` (for general information about LAPACK)
For details, see the *LAPACK User's Guide*.
E. Anderson, Z. Bai, C. Bischof, L. S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen
For a printed version of the *LAPACK User's Guide*, go to `http://www.siam.org`.
For an online version of the *LAPACK User's Guide*, go to
`http://www.netlib.org/lapack/lug/lapack_lug.html`.

`qhull` function in MATLAB is based on the Qhull code

Copyright (c) 1993 The National Science and Technology Research Center for Computation and Visualization of Geometric Structures, The Geometry Center, University of Minnesota

e-mail: `software@geom.umn.edu`

For complete copyright information, issue the MATLAB command `help qhull`.

TIFF capability in the functions `imread`, `imwrite`, `imfinfo`, `print`, and `saveas`:

© Copyright 1988-1999 Sam Leffler

© Copyright 1991-1999 Silicon Graphics, Inc.

Permission to use, copy, modify, distribute, and sell this software and its documentation for any purpose is hereby granted without fee, provided that (i) the above copyright notices and this permission notice appear in all copies of the software and related documentation, and (ii) the names of Sam Leffler and Silicon Graphics may not be used in any advertising or publicity relating to the software without the specific, prior written permission of Sam Leffler and Silicon Graphics.

THE SOFTWARE IS PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EXPRESS, IMPLIED OR OTHERWISE, INCLUDING WITHOUT LIMITATION, ANY WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. IN NO EVENT SHALL SAM LEFFLER OR SILICON GRAPHICS BE LIABLE FOR ANY SPECIAL, INCIDENTAL, INDIRECT OR CONSEQUENTIAL DAMAGES OF ANY KIND, OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER OR NOT ADVISED OF THE POSSIBILITY OF DAMAGE, AND ON ANY THEORY OF LIABILITY, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

# MATLAB 6.0 Release Notes

**New Features** . . . . . . . . . . . . . . . . . . . . . . . 2-2
Development Environment Features . . . . . . . . . . . . 2-2
Mathematics Features . . . . . . . . . . . . . . . . . . . . 2-11
Programming and Data Types Features . . . . . . . . . . 2-22
Graphics Features . . . . . . . . . . . . . . . . . . . . . . 2-26
3-D Visualization Features . . . . . . . . . . . . . . . . 2-30
External Interfaces/API Features . . . . . . . . . . . . . 2-33
Creating Graphical User Interfaces – Features . . . . . . 2-39

**Major Bug Fixes** . . . . . . . . . . . . . . . . . . . . . 2-41
Figure KeyPressFcn . . . . . . . . . . . . . . . . . . . . 2-41

**Platform Limitations for HP and IBM** . . . . . . . . . 2-42
Development Environment Limitations . . . . . . . . . 2-42
Mathematics Limitations . . . . . . . . . . . . . . . . . . 2-43
Graphics and Printing Limitations . . . . . . . . . . . . 2-44
GUIDE Limitations . . . . . . . . . . . . . . . . . . . . . 2-44

**Upgrading from an Earlier Release** . . . . . . . . . . 2-45
Development Environment Issues . . . . . . . . . . . . . 2-45
Programming and Data Types Issues . . . . . . . . . . . 2-45
External Interfaces/API Issues . . . . . . . . . . . . . . 2-56
Creating Graphical User Interfaces – Upgrade Issues . . . . 2-61

**Known Software and Documentation Problems** . . . . 2-62
Development Environment Problems . . . . . . . . . . 2-62
External Interfaces/API Problems . . . . . . . . . . . . . 2-64
Graphics Problems . . . . . . . . . . . . . . . . . . . . . 2-64
GUIDE Problems . . . . . . . . . . . . . . . . . . . . . . 2-65
Documentation Updates . . . . . . . . . . . . . . . . . . 2-65

## New Features

This section introduces the new features and enhancements added in MATLAB 6.0 since MATLAB 5.3 (Release 11.0).

For information about MATLAB new features that are incorporated from prior releases, see:

- *Release 11.1 New Features*
  (enhancements introduced between Release 11.0 and Release 11.1)
- *Release 11 New Features*
  (enhancements introduced between Release 9.0 and Release 11.0)

This section about new features is organized into the following subsections:

- "Development Environment Features" on page 2-2
- "Mathematics Features" on page 2-11
- "Programming and Data Types Features" on page 2-22
- "Graphics Features" on page 2-26
- "3-D Visualization Features" on page 2-30
- "External Interfaces/API Features" on page 2-33
- "Creating Graphical User Interfaces – Features" on page 2-40

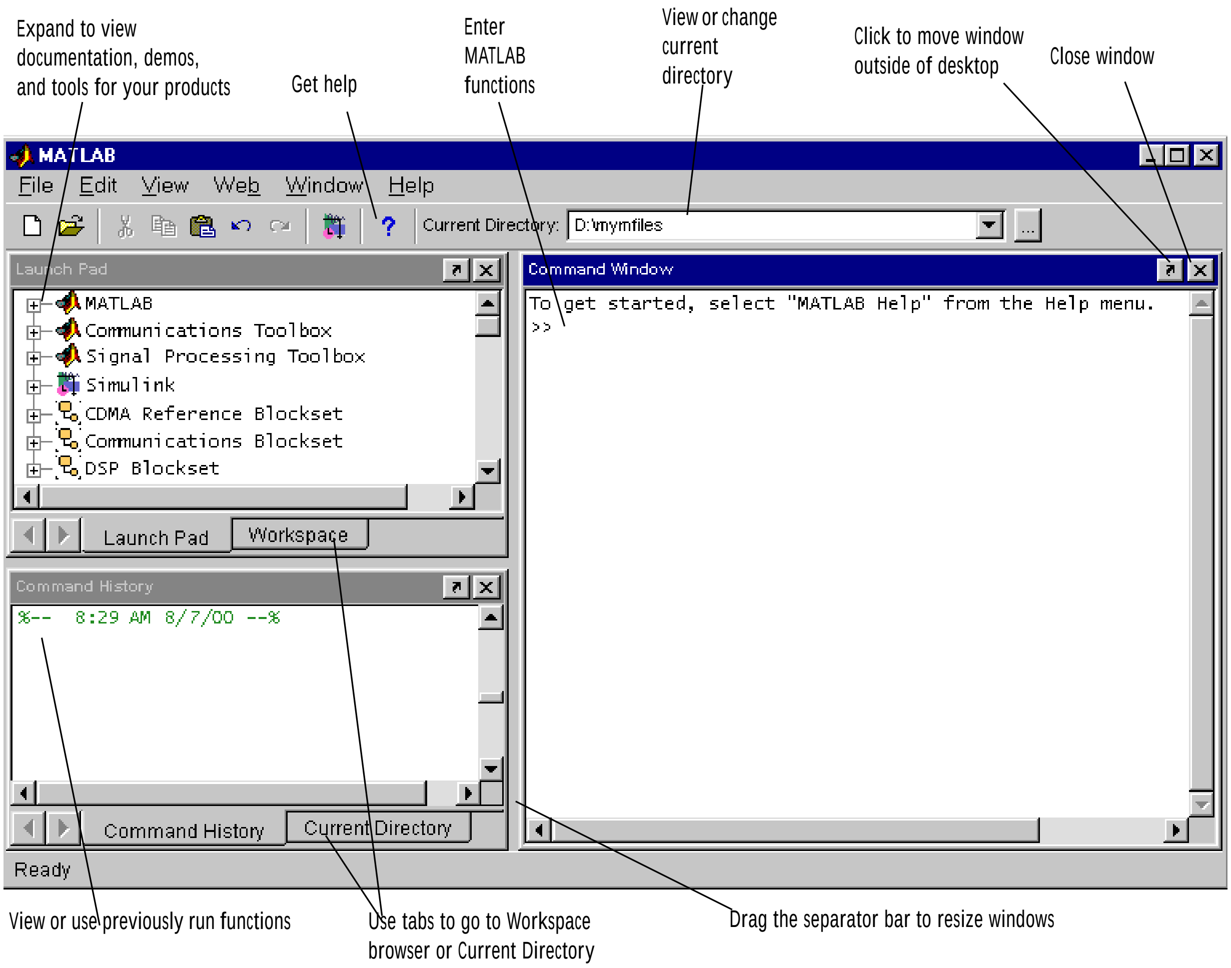
### Development Environment Features

This section includes:

- "MATLAB Desktop" on page 2-2
- "New Online Help" on page 2-6
- "Toolbox Path Cache Reduces MATLAB Startup Time" on page 2-7
- "Import Wizard" on page 2-8
- "Development Environment Functions" on page 2-9

#### MATLAB Desktop

MATLAB has a new environment called the MATLAB desktop, containing tools (graphical user interfaces) for managing files, variables, and applications associated with MATLAB. Think of the desktop as your MATLAB dashboard. The first time MATLAB starts, the desktop appears as shown in the following illustration, although your Launch Pad may contain different entries. You can change the way your desktop looks by opening, closing, and moving tools.


Expand to view documentation, demos, and tools for your products
Get help
Enter MATLAB functions
View or change current directory
Click to move window outside of desktop
Close window
MATLAB
File Edit View Web Window Help
Current Directory: D:\mymfiles
Launch Pad
MATLAB
Communications Toolbox
Signal Processing Toolbox
Simulink
CDMA Reference Blockset
Communications Blockset
DSP Blockset
Launch Pad
Workspace
Command History
Command History
Current Directory
Ready
Command Window
View or use previously run functions
Use tabs to go to Workspace browser or Current Directory
Drag the separator bar to resize windows

```
%--  8:29 AM 8/7/00 --%
```

```
To get started, select "MATLAB Help" from the Help menu.
>>
```

If you prefer a command line interface, you can use functions to perform most of the features found in the MATLAB desktop tools.

The following tools are managed by the MATLAB desktop, but not all of them appear by default when you first start.

| Tool | Purpose | Major New Features |
|---|---|---|
| Command Window | Run MATLAB functions. | • Context menu to evaluate, open, or get help for a selection<br>• Tab completion for function names<br>• Syntax highlighting |
| Command History | View a log of the functions you entered in the command window, copy them, and execute them. | New tool |
| Launch Pad | Easily run tools and access documentation for all your MathWorks products. Expand a listing to show the documentation, demos, and tools for that product. | New tool |
| Help browser | View and search the documentation for MATLAB's full product family, as described in "New Online Help" on page 2-6. | New user interface that replaces the Help Desk |
| Current Directory browser | View MATLAB files and related files. Perform file operations such as open files, and find and replace strings in a file. | New tool |

| Tool | Purpose | Major New Features |
|---|---|---|
| Workspace browser | View and make changes to the contents of the workspace. | • Interface to the Import Wizard<br>• Graph data for a variable using the context menu |
| Array Editor | View array contents in a table format and edit the values. | • Change the display format for variables<br>• Allow strings and cell arrays of strings |
| Editor/ Debugger | Create, edit, and debug M-files (files containing MATLAB functions). | • Show line numbers<br>• Comment or uncomment a selection (multiple lines)<br>• Change colors used for syntax highlighting<br>• Search through multiple files at once for a specified phrase<br>• Option for MATLAB to start a session and open files that were open at shutdown of the previous session<br>• Option to view datatips in edit mode<br>• Breakpoints are maintained when a file is saved. |

To see how many of these new features work, select **Demos** from the **Help** menu in the desktop. The playback demos run in your system's Web browser and illustrate the main features of the new interfaces and tools. The playback demos run faster on Windows platforms than they do on UNIX.

Other MATLAB tools are not managed by the desktop, such as figure windows, toolbox graphical user interfaces, and the following development environment tools.

| Tool | Purpose | Major New Features |
|---|---|---|
| **Set Path** dialog box | View and change the MATLAB search path. | A modified version of the Path Browser user interface |
| Import Wizard | Load binary or text data into the MATLAB workspace. | Graphical user interface to the MATLAB import functions. See “Import Wizard” on page 2-8 for more information. |
| M-File Profiler | Measure where an M-file is spending its time to help you make speed improvements. | Now supports scripts |
| Source Control Interface | Access your source control system from within MATLAB, Simulink, and Stateflow. | New tool |
| Notebook | Access MATLAB’s numeric computation and visualization software from within a word processing environment (Microsoft Word). | Supports Word 2000 |

### New Online Help

Release 12 provides almost all the documentation in online form in HTML (a few products’ online documentation is in PDF form only). The online documentation is at least as up-to-date as any printed documentation shipped with the product, and in several cases is more up-to-date.

**New Help Browser.** Release 12 includes its own Help browser, which allows you to access the online documentation similarly to how you do with Microsoft's HTML Help and Sun's Java Help interfaces. However, the Help browser has been customized to work even more effectively with the whole MATLAB product family. The access methods (via tabs) include:

- An expandable/collapsible table of contents, organized by products
- An index
- A search facility, including a full-text search, as well as searches for word(s) in documentation section titles, function names, or even the Technical Support online knowledge base (via the Web)

Other features include saving favorites (bookmarks) and being able to execute code examples in the online documentation by highlighting the text and using a right-click context menu.

You can print out copies of the documentation by accessing PDF versions of the documentation. For Windows platforms, PDF files are on the Documentation CD.

See "Getting Help" in the MATLAB documentation for complete instructions.

**Context-Sensitive Help.** For several products, you can access context-sensitive help via Help menus, Help buttons, or from a right-click context menu.

## Toolbox Path Cache Reduces MATLAB Startup Time

If you run MATLAB from a network server, you can significantly reduce your startup time by using the new toolbox path cache feature. The toolbox path cache stores path information on all toolbox directories under the MATLAB root directory. During startup, MATLAB obtains this information from the cache rather than by reading it from the remote file system.

The toolbox path cache is used only during the startup of your MATLAB session. It is especially useful if you define your MATLAB path to include many toolbox directories. It takes considerable time to acquire all of this information by scanning directories in the remote file system. Reading it from a pregenerated cache however, is significantly faster. If you have a short toolbox path, there is less benefit to using the cache, but it does still provide a time savings.

When you first install MATLAB, the toolbox path cache must be generated by the system administrator and enabled on those systems for which it is needed. The MATLAB **Preferences** dialog box has a new **Toolbox Caching** panel that assists you in generating and enabling the cache.

See “Reducing Startup Time with Toolbox Path Caching” in the “Development Environment” of the MATLAB documentation for more information.

### Import Wizard

The easiest way to import ASCII text data or binary data into the workspace is to use the new MATLAB Import Wizard. To use the Import Wizard, follow these steps:

1. Start the Import Wizard, by choosing the **Import Data** option on the MATLAB command window **File** menu. The Import Wizard displays a file selection dialog box. Specify the file that contains the data you want to import.

2. The Import Wizard opens the file, processes the data in the file, and displays a preview of the variable (or variables) it has created from the data in the file. If the file contains multiple variables, you can select the variables you want to import. Click **Finish** to import the data into the workspace.

The Import Wizard can process many types of data formats automatically, such as images, sound files, and spreadsheets. The Import Wizard can also process text data files that use commas, spaces, tabs or semicolons as delimiters. (The delimiter, also known as a column-separator, is the character used to separate the individual data items in text data file.) The Import Wizard can process other text files if you specify the delimiter used in the file.

The following table lists the types of data you can import using the Import Wizard.

| **Data Types** | **File Extension** |
|---|---|
| ASCII text data | `.txt`, `.dat`, `.dlm`, and others |
| Audio Video Interleaved (AVI) format | `.avi` |
| CompuServe Graphics Interchange format | `.gif` |

| Data Types | File Extension (Continued) |
|---|---|
| Cursor format | `.cur` |
| HDF raster image | `.hdf` |
| Icon | `.ico` |
| JPEG | `.jpg`, `.jpeg` |
| MATLAB MAT-file | `.mat` |
| Portable Network Graphics | `.png` |
| Sound files | `.wav, .au, .snd` |
| Spreadsheet | `.csv`, `.xls`, `.wk1` |
| Zsoft Paintbrush | `.pcx` |

### Development Environment Functions

This section lists the new and changed development environment functions.

**New Development Environment Functions.** The functions listed in the following table are new in MATLAB 6.0.

| Function | Description |
|---|---|
| `checkin` | Check files into your source control system from MATLAB, Simulink, and Stateflow. |
| `checkout` | Check files out of your source control system into MATLAB, Simulink, and Stateflow. |
| `cmopts` | Get the name of the source control system being used with MATLAB. |
| `customverctrl` | Integrate a version control system not supported with MATLAB. |

| Function | Description (Continued) |
|---|---|
| `filebrowser` | Display the Current Directory browser, a tool for viewing files in the current directory and performing file operations. |
| `helpbrowser` | Display the MATLAB Help browser, which provides access to extensive online help. |
| `undocheckout` | Undo the previous checkout from the source control system. |

**Development Environment Functions That Changed.** The functions listed in the following table have been changed since MATLAB 5.2 (Release 11).

| Function | Description of Change |
|---|---|
| `dbstop` | The function `dbstop if error` no longer stops execution on errors detected within a `try`...`catch` block. MATLAB does not enter debug mode under these circumstances.<br><br>Use the new form of the function, `dbstop if all error` to stop execution and enter debug mode on all types of errors, including those detected within a `try`...`catch`. |
| `doc` | Displays documentation in the Help browser instead of in the Help Desk. If an HTML reference page for a function does not exist, it displays M-file help in the Help browser. |
| `docopt` | Is now only used for:<br>• The `web` function, if the `-browser` option is used for UNIX platforms<br>• The IBM and HP platforms – see “Platform Limitations” on page 1-17 of the *Release Notes* |

| Function | Description of Change (Continued) |
|---|---|
| `helpdesk` | Displays the Help browser instead of the Help Desk. In a future release, the `helpdesk` function will be phased out. |
| `helpwin` | Function listings and descriptions appear in the Help browser instead of in a specialized window. |
| `pathtool` | Opens the new **Set Path** dialog box instead of the Path Browser. |
| `version` | Now has a `-java` flag, which displays the version of Java used by MATLAB. |
| `web` | By default, displays the specified `URL` in the Help browser. Now includes the `-browser` flag, which displays the specified `URL` in your system's default Web browser. |

## Mathematics Features

The following mathematics features have been added or enhanced in MATLAB 6.0:

- Matrix computations
- N-dimensional Delaunay-type functionality
- Differential equations solvers
- Sparse matrices
- Fast Fourier transforms
- Quadrature
- Function functions
- Basic Fitting interface
- Data Statistics interface

These features are described below. At the end of this section are tables that summarize changes to the MATLAB math functions:

- New functions
- Functions with new or changed capabilities

See “Upgrading from an Earlier Release” on page 2-46 for information about obsolete functions.

## Matrix Math in MATLAB 6.0

Matrix computations in MATLAB 6.0 are based on LAPACK, a large, multiauthor Fortran subroutine library for numerical linear algebra. LAPACK extends MATLAB’s matrix computation capabilities and increases its speed on larger problems. It offers MATLAB a larger class of algorithms from which to choose based on the properties of the matrix arguments. Optimized Basic Linear Algebra Subroutines (BLAS), on all MATLAB platforms, speeds up matrix multiplication and the LAPACK routines themselves. Optimized BLAS is provided by Automatically Tuned Linear Algebra Software (ATLAS).

The LAPACK Users’ Guide, Third Edition, is available online at `http://www.netlib.org/lapack/lug/lapack_lug.html`.

**Differing Results.** Matrix functions in earlier versions of MATLAB continue to operate in the same way in MATLAB 6.0, but the results they return may differ. Changes in roundoff errors can be seen in most matrix computations. In cases where quantities are not uniquely determined mathematically, results may differ in order and in normalization.

For example:

- Eigenvalues may be returned in a different order.
- Eigenvectors may be normalized differently.
- The signs of columns of orthogonal matrices may differ.
- `rcond` is a better estimate of the reciprocal condition.
- `lu` can now be used to factor rectangular full matrices.

**Increased Number of Eigenvalue Algorithms.** With MATLAB 6.0, there are now eight different eigenvalue algorithms, determined by:

- `eig(A)` or `eig(A,B)`
- Real or complex matrices
- Symmetric/Hermitian matrices and `B`, if any, positive definite
- Whether eigenvectors are requested or not

For the symmetric and Hermitian problems, the eigenvalues are real, sorted in increasing order and the eigenvectors are normalized so that

`V'*V` or `V'*B*V = I`

**New Decompositions for Real Matrices.** The QZ algorithm returns a newly available real decomposition for real matrices. If `A` and `B` are real and not symmetric,

```
[AA,BB,...] = qz(A,B,'real')
```

returns a real triangular matrix `BB` and a real quasitriangular matrix `AA` with 2-by-2 diagonal blocks corresponding to pairs of complex conjugate eigenvalues. Earlier versions of MATLAB produced complex triangular `AA` and `BB` if there were any complex eigenvalues. You can continue to obtain this behavior with either

```
[AA,BB,...] = qz(A,B)
```

or

```
[AA,BB,...] = qz(A,B,'complex')
```

A similar option for the Schur decomposition of a real matrix

```
T = schur(A,'complex')
```

produces a complex decomposition if `A` has any complex eigenvalues.

**The flops Function.** The incorporation of LAPACK makes it impractical to count floating-point operations. As a result, the `flops` function is inoperative in MATLAB 6.0. It will be discontinued completely in some future version.

With modern computer architectures, floating-point operation counts are no longer a good predictor of execution time. Counts of memory references and cache usage patterns have become more important.

### N-Dimensional Delaunay-Type Functionality

New Qhull-based functions extend Delaunay-type functionality:

- `delaunayn` and `delaunay3` for N-dimensional Delaunay tessellation
- `convhulln` N-dimensional convex hull
- `voronoin` for N-dimensional Voronoi diagrams
- `griddatan` and `griddata3` for data gridding and hyper-surface fitting

### Differential Equation Solvers

New differential equation solvers expand MATLAB's capability:

- `bvp4c` solves two-point boundary value problems for ODEs by collocation. Supporting functions enable you to set options that affect problem solution, form an initial guess, and evaluate the numerical solution obtained with `bvp4c`.
- `pdepe` solves initial-boundary value problems for parabolic-elliptic PDEs in 1-D. A supporting function enables you to evaluate the numerical solution obtained with `pdepe`.

The ODE solvers now take advantage of function handles and can solve problems without using ODE files. The new syntax is

```
solver(@odefun,tspan,y0,options,p1,p2,...)
```

where `odefun`, `tspan` and `y0` are required arguments. See the ODE solver and `odeset` reference pages for details.

MATLAB 6.0 supports use of an ODE file for backwards compatibility, but new functionality is available only with the new syntax.

### Sparse Matrix Computations

New and upgraded routines provide new capabilities and speed up computations:

- `eigs` and `svds` now use the Fortran library ARPACK.
- New routines, `symmlq`, `minres` and `lsqr`, iteratively solve symmetric indefinite systems and least squares problems.
- New routines, `colamd` and `symamd`, provide approximate minimum degree permutations to help reduce the fill-in of their sparse factors.
- `condest` can now produce more accurate condition estimates.

### Fast Fourier Transforms

Fast Fourier transform (FFT) functions now rely on the MIT FFTW library. This results in faster performance for composite, prime, and large prime factor array lengths.

### Quadrature

New quadrature algorithms in `quad` and the new function `quadl` are faster, more accurate and more robust in that they handle singularities better. `quadl` replaces the now obsolete `quad8` function.

### Interpolation

A new one-dimensional interpolation function, `pchip`, based on piecewise cubic Hermite interpolating polynomials, preserves the shape and monotonicity of the underlying data.

### Function Functions

All function functions are now capable of accepting function handles as arguments. Most also accept additional parameters, which they pass to the function that you pass in as an argument.

For information about function handles, see the `function_handle` (`@`), `func2str`, and `str2func` reference pages, and the “Function Handles” section of “Programming and Data Types” in the MATLAB documentation.

### The Basic Fitting Interface

MATLAB supports curve fitting through the Basic Fitting interface. Using this interface, you can quickly perform many curve fitting tasks within the same easy-to-use environment. The interface is designed so that you can:

- Fit data using a spline interpolant, a hermite interpolant, or a polynomial up to degree 10.
- Plot multiple fits simultaneously for a given data set.
- Plot the fit residuals.
- Examine the numerical results of a fit.
- Evaluate (interpolate or extrapolate) a fit.
- Annotate the plot with the numerical fit results and the norm of residuals.
- Save the fit and evaluated results to the MATLAB workspace.

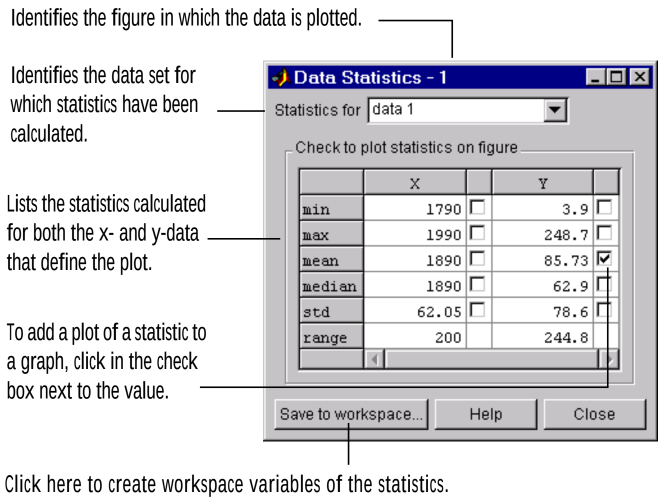
Depending on your specific curve fitting application, you can use the Basic Fitting interface, the command line functionality such as `polyfit` and `polyval`, or both.

### The Data Statistics Interface

MATLAB has a new visual interface that:

- Calculates basic statistics about the central tendency and variability of data plotted in a graph
- Lets you save the statistics to the workspace
- Lets you plot any of the statistics in a graph

When you select **Data Statistics** from the MATLAB figure window **Tools** menu, MATLAB calculates the statistics for each data set plotted in the graph and displays the results in a **Data Statistics** dialog box. To plot a statistic in a graph, click in the check box next to its value. To save a set of statistics as a workspace variable, click on the **Save to workspace...** button. The Data Statistics tool saves the statistics as a structure. The following figure shows the components of this dialog box.



Identifies the figure in which the data is plotted.

Identifies the data set for which statistics have been calculated.

Lists the statistics calculated for both the x- and y-data that define the plot.

To add a plot of a statistic to a graph, click in the check box next to the value.

Click here to create workspace variables of the statistics.

| | X | | Y | |
|---|---|---|---|---|
| min | 1790 | ☐ | 3.9 | ☐ |
| max | 1990 | ☐ | 248.7 | ☐ |
| mean | 1890 | ☐ | 85.73 | ☑ |
| median | 1890 | ☐ | 62.9 | ☐ |
| std | 62.05 | ☐ | 78.6 | ☐ |
| range | 200 | | 244.8 | |

## Math Function Summary Tables

This section summarizes:

- New math functions
- Functions with new or changed capabilities

For more information on these functions, see their respective reference pages or type

```
help function
```

in the MATLAB command window, where `function` is the name of the function about which you want to obtain more information.

---

**Note** See “Upgrading from an Earlier Release” on page 2-46 for information about obsolete functions.

---

## New Math Functions

| Function | Purpose |
|---|---|
| `bvp4c` | Solve two-point boundary value problems (BVPs) for ODEs by collocation |
| `bvpget` | Extract an option from the BVP options structure |
| `bvpinit` | Form the initial guess for `bvp4c` |
| `bvpset` | Create/alter BVP options structure |
| `bvpval` | Evaluate the solution computed by `bvp4c` |
| `colamd` | Column approximate minimum degree permutation |
| `convhulln` | N-dimensional convex hull |
| `delaunay3` | Three-dimensional Delaunay tessellation |
| `delaunayn` | N-dimensional Delaunay tessellation |
| `dsearchn` | N-dimensional nearest point search |
| `griddata3` | Data gridding and hyper-surface fitting for 3-D data |
| `griddatan` | Data gridding and hyper-surface fitting (dimension >= 2) |
| `lsqr` | LSQR implementation of Conjugate Gradients on the Normal Equations |
| `minres` | Solve a system of equations using Minimum Residual Method |
| `pchip` | Piecewise Cubic Hermite Interpolating Polynomial (PCHIP) – preserves monotonicity and the shape of the data. `pchip` is used by `interp1(x,y,xi,'cubic')` |
| `pdepe` | Solve initial-boundary value problems for parabolic-elliptic partial differential equations (PDEs) in one dimension |
| `pdeval` | Evaluate by interpolation the solution computed by `pdepe` |

**New Math Functions (Continued)**

| Function | Purpose |
|---|---|
| `quadl` | Numerically evaluate an integral using adaptive Lobatto quadrature |
| `symamd` | Symmetric approximate minimum degree permutation |
| `symmlq` | Solve a system of equations using symmetric LQ method |
| `voronoin` | Compute N-dimensional Voronoi diagram |

**Math Functions with New or Changed Capabilities**

| Function | Purpose |
|---|---|
| `condest` | `[c,v] = condest(A,t)` specifies a new argument, `t`, a positive integer equal to the number of columns in an underlying iteration matrix. Increasing the number of columns usually gives a better condition estimate but increases the cost. The default is `t = 2`, which almost always gives an estimate correct to within a factor 2. |
| `dblquad` | `dblquad` now accepts extra arguments `p1,p2,...` which it passes to `fun`. For example,<br>`dblquad(fun,xmin,xmax,ymin,ymax,tol,...`<br>`        @quadl,p1,p2,...)` |
| `eig` | For symmetric `A` and symmetric positive definite `B`, `eig(A,B,'chol')` computes the generalized eigenvalues of `A` and `B` using the Cholesky factorization of `B`. `'chol'` is the default.<br><br>`eig(A,B,'qz')` ignores the symmetry, if any, and uses the QZ algorithm. |

**Math Functions with New or Changed Capabilities (Continued)**

| Function | Purpose |
|---|---|
| eigs | Now provides an interface to a subset of the ARPACK capabilities. See the eigs reference page for information about the expanded syntax, and the sigma and options arguments. The MATLAB 5 arbitrary ordering of the inputs B, k, sigma, and options is no longer allowed. |
| fftshift, ifftshift | fftshift(X,dim) and ifftshift(X,dim) can now apply the shift operation along the dimension specified by dim. |
| fminbnd, fminsearch, fzero, lsqnonneg | A new Display options parameter value, 'notify', displays output only if the function does not converge. For these functions, 'notify' is the default. |
| fzero | MATLAB Version 5 changed the calling sequence for fzero. See the instructions for converting your code in “Function Functions” in the MATLAB documentation. |
| gallery | Two new options produce these test matrices:<br>'randcolu' – Random matrix with normalized columns and specified singular values<br>'randcorr' – Random correlation matrix with specified eigenvalues |
| interp1 | interp1(x,y,xi,'cubic') and interp1(x,y,xi,'pchip') use pchip to perform the interpolation. A new flag 'v5cubic' provides the cubic interpolation used in MATLAB 5. The default method is 'linear'.<br>interp1(x,y,xi,method,'extrap') uses the specified method to extrapolate any element of xi that is outside the interval spanned by x.<br>interp1(x,y,xi,method,extrapval) returns the scalar extrapval for out of range values. |

**Math Functions with New or Changed Capabilities (Continued)**

| Function | Purpose |
|---|---|
| `lu` | `lu(X)` can now be used to factor rectangular matrices. |
| `ode45`, `ode23`, `ode113`, `ode15s`, `ode23s`, `ode23t`, `ode23tb` | The ODE solvers can now solve problems without the use of an ODE file. Problem components are passed to the solvers directly as arguments, or provided using parameters in an options structure. See the ODE solvers and `odeset` reference pages for details.<br><br>MATLAB 6.0 supports use of an ODE file for backwards compatibility, but new functionality is available only in the new syntax. |
| `polyval`, `polyfit` | An optional output argument for `polyfit` and an optional input argument to `polyval` provide for centering and scaling, that is, subtracting the mean and normalizing the standard deviation of the independent variable. |
| `quad` | `quad(fun,a,b)` uses a new default tolerance, $10^{-6}$.<br><br>Because of the use of new quadrature algorithms, your results (`q`) and the number of function evaluations (`trace(1) = fcnt`) may differ from MATLAB 5. The new algorithm provides more accurate results and generally result in improved performance. |
| `qz` | For real `A` and `B`, `[AA,BB,Q,Z,V] = qz(A,B,'real')` produces a real decomposition with a quasitriangular `AA`.<br><br>`[AA,BB,Q,Z,V] = qz(A,B,'complex')` produces a possibly complex decomposition with a triangular `AA`. |
| `schur` | For real `X`, `schur(X,'real')` has the real eigenvalues on the diagonal and the complex eigenvalues in 2-by-2 blocks on the diagonal. `schur(X,'complex')` is triangular and is complex if `X` has complex eigenvalues. |

**Math Functions with New or Changed Capabilities (Continued)**

| Function | Purpose |
|---|---|
| `sort` | `sort(S)` now works on other data types, for example `int32`, and has been rewritten to be faster on doubles. `sort(S,dim)` can now sort the elements of both full and sparse matrices along the dimension specified by `dim`. |
| `sqrtm` | `[X, alpha, condest] = sqrtm(A)` returns a stability factor `alpha` and an estimate `condest` of the matrix square root condition number of `X`. |
| `std` | `std([])` no longer returns empty. It now returns `NaN`, and a message, `Warning: Divide by zero`. |

## Programming and Data Types Features

### MATLAB Interface to Java

MATLAB 6.0 provides an interface to Java that enables you to create objects from Java classes and call methods on those objects. You can use existing Java classes or create your own. See “MATLAB Interface to Java” on page 2-33 for a summary of this feature. See “Calling Java from MATLAB” under “External Interfaces/API” in the online help, for full documentation.

### Function Handles

The MATLAB language has a new data type called the function handle. You can create a handle to any MATLAB function and then use that handle as a means of referencing the function. A function handle is typically passed in an argument list to other functions, which can then execute, or *evaluate*, the function using the handle.

A MATLAB function handle is more than just a reference to a function. It often represents a collection of function methods, overloaded to handle different argument types. When you create a handle to a function, MATLAB takes a snapshot of all built-in and M-file methods of that name that are on the MATLAB path and in scope at that time, and stores access information for all of those methods in the handle.

When it comes time to evaluate the function handle, MATLAB considers only those functions that were captured within the handle when it was created. It is the combination of which functions are in the handle and what arguments the handle is evaluated with that determines which is the actual function that MATLAB dispatches to.

Function handles enable you to do all of the following. Each of these items is explained in more detail in “Benefits of Using Function Handles” in the online documentation:

- Pass function access information to other functions
- Capture all methods of an overloaded function
- Allow wider access to subfunctions and private functions
- Ensure reliability when evaluating functions
- Reduce the number of files that define your functions
- Improve performance in repeated operations
- Manipulate handles in arrays, structures, and cell arrays

You construct a function handle in MATLAB using the *at* sign, @, before the function name. The following example creates a function handle for the `humps` function and assigns it to the variable `fhandle`.

```
fhandle = @humps;
```

You pass the handle in the same way you would pass any argument. This example passes the function handle just created to `fminbnd`, which then minimizes over the interval [0.3, 1].

```
x = fminbnd (fhandle, 0.3, 1)
x =
    0.6370
```

The `fminbnd` function evaluates the `@humps` function handle using the MATLAB `feval` function.

For more information, see “Function Handles” under “Programming and Data Types,” in the MATLAB documentation.

### Functions That Operate on Function Handles

The following MATLAB functions now operate on the function handle data type. The `func2str`, `functions`, and `str2func` functions are new to the MATLAB language.

| Function | Purpose |
|---|---|
| `feval` | Evaluate a function through its handle |
| `func2str` | Construct a function name string from a function handle |
| `functions` | Display information about a function handle |
| `isa` | Determine if an object is a function handle |
| `isequal` | Compare function handles for equality |
| `str2func` | Construct a function handle from a function name string |

### CONTINUE Flow Control Statement

The `continue` statement passes control to the next iteration of the `for` or `while` loop in which it appears, skipping any remaining statements in the body of the loop. In nested loops, `continue` passes control to the next iteration of the `for` or `while` loop enclosing it.

The example below shows a `continue` loop that counts the lines of code in the file, `magic.m`, skipping all blank lines and comments. A `continue` statement is used to advance to the next line in `magic.m` without incrementing the count whenever a blank line or comment line is encountered.

```
fid = fopen('magic.m','r');
count = 0;
while ~feof(fid)
    line = fgetl(fid);
    if isempty(line) | strncmp(line,'%',1)
        continue
    end
    count = count + 1;
end
disp(sprintf('%d lines',count));
```

### New MATLAB Programming-Related Functions

The following programming-related functions are new in this release.

| Function | Purpose |
|---|---|
| `beep` | Make your computer beep |
| `genpath` | Generate a path string that includes all directories below a named directory |
| `iskeyword` | Check if the input string is a MATLAB keyword |
| `isvarname` | Check if the input string is valid variable name |
| `nargoutchk` | Validate the number of output arguments |
| `numel` | Returns the number of elements in an object |
| `rehash` | Refresh function and file system caches |
| `support` | Open the MathWorks Technical Support Web page |

### Creating an Object That Inherits from Parent Classes Only

In MATLAB 5, the `class` function allows you to create a new object that inherits fields and methods from one or more parent classes. However, this new object acquires additional fields that belong to the `structure_name` argument passed in the call to `class`.

The syntax for this command is

```
obj = class(structure_name,'class_name',parent1,parent2...)
```

In MATLAB 6.0, you can create a new object that contains no fields other than those that are inherited from the specified parent objects. Do this using the following syntax.

```
obj = class(struct([]),'class_name',parent1,parent2,...)
```

### Code Length Restriction Removed

There is now no limit on the length of a line of M-code.

### Running a Syntax Check on M-Files

You can run a check on the syntax of your M-files before executing the files; to do so, use the following command.

```
check_syntactic_warnings
```

This command checks all M-files in the directories specified by the argument list for all warnings that MATLAB generates when reading the M-file into memory. All @class and private directories contained by the argument list directories will also be processed; class and private directories should not be supplied as explicit arguments to this function.

If no argument list is specified, all files on the MATLAB path and in the current working directory will be checked, except those in toolboxes.

If the argument list is exactly `'-toolboxes'`, all files on the MATLAB path and in the current working directory will be checked, including those in toolboxes.

The following command displays the information given above.

```
help check_syntactic_warnings
```

This command can be especially helpful in locating missing separator characters that are now required between array elements. See “Separators Are Now Required Between Array Elements” on page 2-48.

## Graphics Features

This section is organized into the following subsections:

- “Property Editor” on page 2-26
- “Printing Features” on page 2-28

### Property Editor

MATLAB 6.0 has a new graphical user interface for editing the properties of Handle Graphics objects in MATLAB figures. This tool, called the Property Editor, provides access to many properties of the Handle Graphics objects in a graph, including figures, axes, lines, lights, patches, images, surfaces, rectangles, text, and the root object. Using this tool, you can change the thickness of a line, add titles and axes labels, add lights, and perform many other plot editing tasks.

The following figure shows the components Property Editor interface.

Use these buttons to move back and forth among the graphics objects you have edited.

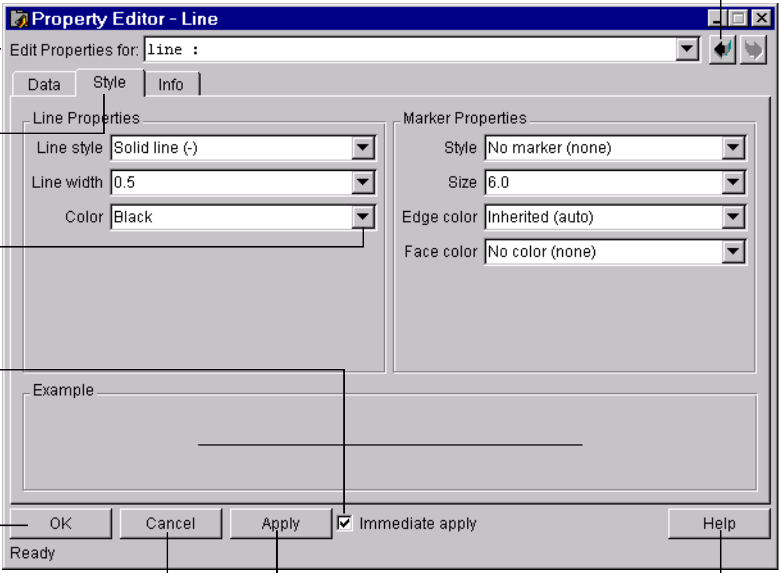
Use the navigation bar to select the object you want to edit.

Click on a tab to view a group of properties.

Click here to view a list of values for this field.

Check this box to see the effect of your changes as you make them.

Click OK to apply your changes and dismiss the Property Editor.



Click Cancel to dismiss the Property Editor without applying your changes.

Click Apply to apply your changes without dismissing the Property Editor.

Click Help to get information about particular properties.

## Starting the Property Editor

There are several ways to start the Property Editor.

If plot editing mode is enabled in the figure, you can start the Property Editor by right-clicking on an object and selecting the **Properties** option from the context menu. You can also start the Property Editor by double-clicking on an object in the graph. (Double-clicking on a text object opens a text editing box around the text but does not start the Property Editor.)

If plot editing mode is not enabled, you can start the Property Editor by selecting either the **Figure Properties**, **Axes Properties**, or **Current Object Properties** from the figure window **Edit** menu. These options automatically enable plot editing mode, if it is not already enabled. You can also start the Property Editor from the command line using the `propedit` function.

---

**Note** Once you start the Property Editor, you can keep it open throughout an editing session. If you click on another object in the graph, the Property Editor displays the set of panels associated with that object type. You can also use the Property Editor's navigation bar to select an object to edit.

---

### Printing Features

**Exporting Figures to the Clipboard.** If you use the clipboard to export your figures to graphics-format files, you can now use the **Figure Copy Template Preferences** panel to optimize your figure. You can apply templates specifically for Microsoft Word and PowerPoint, or you can customize your own template.

**Page Setup Dialog Box.** The **Page Setup** dialog box now presents an enhanced set of options on four tabs.

| Tab | Enables you to... |
|---|---|
| **Size and Position** | • Print at screen size or set the size of the printed figure<br>• Set the top and left margins and the height and width of the figure<br>• Fill the page, restore the aspect ratio, or center the figure<br>• Choose the units in which the settings are made |
| **Paper** | • Choose the paper type or set a custom size<br>• Choose units<br>• Choose paper orientation |

| Tab | Enables you to... (Continued) |
|---|---|
| **Lines and Text** | Set the lines and text in the printed or exported figure to color or black and white |
| **Axis and Figure** | • Keep the same settings as are on the screen, or let MATLAB rescale your axes ticks and labels<br>• Print or not print uicontrols that are part of the figure<br>• Keep the screen background color, or let MATLAB change the background color to white<br>• Override MATLAB's default choice of renderer |

**UNIX Print Dialog Box.** The UNIX **Print** dialog box for MATLAB 6.0 has been rewritten. It has many new features available both from the **Print** dialog box and from a new **Options** dialog box:

- From the **Print** dialog box, you can now set the figure size, select a print driver from a scroll list rather than typing it in, and set whether or not you want MATLAB to rescale your axis and ticks limits when it prints your graphic.
- The new **Print Options** dialog box, which launches from the **Print** dialog, enables you to make many settings, including choosing a renderer and setting the printer resolution.

---

**Note** Note that you can no longer set the paper orientation or the paper type from the **Print** dialog box; you must use the **Page Setup** dialog box instead.

---

**Preference that Controls Color or Black and White Printing.** The **Figure window printing** preference that enables you to set a session-to-session default for sending either black and white or colored lines and text to printers is now located on the **General Preferences** panel.

Note that if your figure does not print correctly in color or black and white according to the figure setting, override the setting from the **File/Preferences** menu. For example, if you specify printing lines and text in color to a color printer, and you don't get your results in color, go to **File/Preferences** and select **Always send as color**.

**PaperSize Property.** The figure property `PaperSize` is now writable. You can also set the new `'<custom>'` paper type from the command line.

---

**Note** See "Basic Printing and Exporting" in the MATLAB graphics documentation for more information about printing and exporting figures in MATLAB Version 6.0.

---

## 3-D Visualization Features

The following table lists new and enhanced volume visualization functions added in MATLAB 6.0.

| Function | Purpose |
|---|---|
| `coneplot` | Create a 3-D coneplot |
| `contourslice` | Draw contours in slice planes |
| `curl` | Compute the curl and angular velocity perpendicular to the flow of a 3-D vector field |
| `divergence` | Compute the divergence of a 3-D vector field |
| `interpstreamspeed` | Interpolate streamline vertices from speed |
| `isocolors` | Compute the colors of isosurface vertices |
| `isosurface` | Extract isosurface |
| `streamparticles` | Draw stream particles from vector data |
| `streamribbon` | Draw stream ribbons from vector data |

| Function | Purpose (Continued) |
|---|---|
| `streamslice` | Draw well-spaced stream lines from vector data |
| `streamtube` | Draw stream tubes from vector data |
| `volumebounds` | Return coordinate and color limits for volume data |

## Graphics Object Transparency

The transparency of an object determines the degree to which you can see through the object's surface and thereby see other objects that are obscured. You can specify a continuous range of transparency from completely transparent (i.e., invisible) to completely opaque (i.e., no transparency). You can specify transparency properties for the following objects:

- Surface
- Patch
- Images

**Transparency Properties.** Transparency values, which range from [0 1], are referred to as *alpha* values. MATLAB handles transparency in a way that is analogous to how it handles color:

- Objects can define alpha data that is used as indices into the alphamap or directly as alpha values.
- Objects can define a single face and edge alpha value or use flat or interpolated transparency based on values in the figure's alphamap.
- Axes define alpha limits that control the mapping of object data to alpha values.
- Figures contain alphamaps, which are m-by-1 arrays of alpha values.

The following table summarizes the object properties that control transparency.

| Property | Purpose |
|---|---|
| `ALim` | Alpha axis limits |
| `ALimMode` | Alpha axis limits mode |
| `AlphaData` | Transparency data for patch, surface, or image |
| `AlphaDataMapping` | Transparency data mapping method |
| `Alphamap` | Figure alphamap |
| `FaceAlpha` | Transparency of the object's faces |
| `EdgeAlpha` | Transparency of the object's edges |
| `FaceVertexAlphaData` | Patch-only alpha data specification |

**Transparency Helper Functions.** This table lists functions that simplify the process of setting transparency properties.

| Function | Purpose |
|---|---|
| `alpha` | Set or query transparency properties for objects in current axes |
| `alphamap` | Specify the figure alphamap |
| `alim` | Set or query the axes alpha limits |

See "Transparency" in the MATLAB "3-D Visualization" documentation for more information.

### Renderer Autoselection

MATLAB automatically selects the rendering method used based on the content of the figure. If it is available, MATLAB selects OpenGL as the renderer when certain criteria are met. OpenGL is available on many computer systems and is generally faster than MATLAB's other renderers (Painters and Z-buffer). Using OpenGL enables MATLAB to access the graphics hardware that is available on some systems.

**Criteria for Autoselection of OpenGL Renderer.** When the figure `RendererMode` property is set to `auto`, MATLAB uses the OpenGL autoselection criteria to determine whether to select the OpenGL renderer.

When the `RendererMode` property is set to `manual`, MATLAB does not change the renderer, regardless of changes to the figure contents.

If you do not want MATLAB to include OpenGL in the renderer autoselection process, issue the following command.

```
opengl neverselect
```

See the `opengl` reference page for more information.

### Camera Toolbar

The Camera Toolbar enables you to perform a number of viewing operations interactively. To display the toolbar, select **Camera Toolbar** from the figure window's **View** menu.

See "View Control with the Camera Toolbar" in the MATLAB "3-D Visualization" documentation for more information.

## External Interfaces/API Features

### MATLAB Interface to Java

The new Java capability enables you to conveniently bring Java classes into the MATLAB environment, to construct objects from those classes, to call methods on the Java objects, and to save Java objects for later reloading — all accomplished with MATLAB functions and commands. You can also develop your own Java class definitions and make them available for use in MATLAB.

The MATLAB Java interface is intended for all MATLAB users who want to take advantage of the special capabilities of the Java programming language. The interface enables you to:

- Access Java application programming interface (API) classes that support essential activities such as I/O and networking. For example, the `URL` class provides convenient access to resources on the internet.

- Easily construct Java objects in MATLAB
- Call Java object methods, using either Java or MATLAB syntax
- Pass data seamlessly between MATLAB variables and Java objects

You construct Java objects in MATLAB by calling the Java class constructor, which has the same name as the class. For example, the following constructor creates a `Frame` object with the title `'Frame A'`. Any other properties are set to their default values.

```
frame = java.awt.Frame('Frame A');
```

You call methods on this object using either Java syntax

```
frame.setTitle('Sample Frame')
```

Or MATLAB syntax.

```
setTitle(frame,'Sample Frame')
```

You can create n-dimensional arrays of Java objects and call methods on the arrays. You reference and assign to elements of those arrays using MATLAB matrix syntax. You can pass either MATLAB data or Java objects to the methods of Java classes. MATLAB performs data type conversion on this data where necessary.

See “Calling Java from MATLAB” under “External Interfaces/API” in the online help, for full documentation of this feature.

### New Functions for the Interface to Java

MATLAB 6.0 provides the following new functions to support the interface to Java.

| Function | Purpose |
|---|---|
| `import` | Add to the current Java packages import list |
| `isjava` | Test whether an object is a Java object |
| `javaArray` | Construct a Java array |
| `javaMethod` | Invoke a Java method (used only in special cases) |

| Function | Purpose |
|---|---|
| `javaObject` | Construct a Java object (used only in special cases) |
| `methodsview` | Display information on all methods implemented by a Java or MATLAB class |

### Existing Functions with Java Functionality Added

The following MATLAB functions now operate on Java objects, classes, and methods.

| Function | Purpose |
|---|---|
| `cell` | Convert a Java object array into a MATLAB cell array |
| `char` | Convert a `java.lang.String` object or array to a `char` array or cell array of `char` arrays |
| `class` | Return a Java class name |
| `clear import` | Remove the Java packages import list |
| `depfun` | Return Java class names that the file and its subordinates use |
| `disp` | Display a Java object |
| `double` | Convert a Java object or array to a double array |
| `exist` | Determine if a Java class exists |
| `fieldnames` | Return the field names of a Java object |
| `inmem` | Return the names of loaded Java classes |
| `isa` | Identify an object as a Java class |
| `isequal` | Compare Java objects for equality |
| `methods` | Return all methods of a Java class |

| Function | Purpose |
|---|---|
| `struct` | Convert a Java object or array to a MATLAB structure or structure array |
| `which` | Return the package, class, and method name for a Java class method |

### Restriction on Unloading Java Classes

If you load a Java class into MATLAB and, sometime later, modify and recompile the class, you must exit and restart MATLAB in order to use the updated class definition. This restriction exists because the Java VM does not provide a way to unload a Java class once it has been loaded. Exiting and restarting MATLAB terminates and restarts the Java VM, which then allows you to load the updated class.

Similarly, the `import` function will not import an updated class definition. You must first exit and restart MATLAB.

### Java Version

The Java version used by MATLAB differs by platform and for Release 12 is as shown in the following table.

| Platform | Java Version |
|---|---|
| Compaq Alpha | 1.1.8-5 |
| IBM | 1.2.2 |
| Linux | 1.1.8 |
| Microsoft Windows | 1.1.8 |
| SGI | 1.1.8-00 |
| Sun Solaris | 1.1.8-09a |

To see the Java version used by MATLAB, type `version -java` in the command window.

### New C Language mx Functions

The following `mx` functions are new in this release.

| Function | Purpose |
|---|---|
| `mxCreateNumericMatrix` | Create a numeric matrix and initialize all its data elements to 0 |
| `mxCreateScalarDouble` | Create a scalar, double-precision array initialized to the specified value |

### Additional Supported Compilers

MATLAB now includes preconfigured options files for these compilers:

- LCC 2.4
- Compaq Fortran 6.1
- Borland's C++Builder 3.0 (Borland C++, Version 5.3)
- Borland's C++Builder 4.0 (Borland C++, Version 5.4)
- Borland's C++Builder 5.0 (Borland C++, Version 5.5)

**Note** The LCC compiler is included in the MATLAB product set.

### Using the Add-In for Visual Studio

The MathWorks provides a MATLAB add-in for the Visual Studio® development system that lets you work easily within Microsoft Visual C/C++ (MSVC). The MATLAB add-in for Visual Studio greatly simplifies using M-files in the MSVC environment. The add-in automates the integration of M-files into Visual C++ projects. It is fully integrated with the MSVC environment.

The add-in for Visual Studio is automatically installed on your system when you run either `mbuild -setup` or `mex -setup` and select Microsoft Visual C/C++ version 5 or 6.

However, there are several steps you must follow in order to use the add-in:

1 To build MEX-files with the add-in for Visual Studio, run the following command at the MATLAB command prompt.

```
mex -setup
```

Follow the menus and choose either Microsoft Visual C/C++ 5.0 or 6.0. This configures `mex` to use the selected Microsoft compiler and also installs the necessary add-in files in your Microsoft Visual C/C++ directories.

2 To configure the MATLAB add-in for Visual Studio to work with Microsoft Visual C/C++:

a Select **Tools** -> **Customize** from the MSVC menu.

b Click on the **Add-ins and Macro Files** tab.

c Click **Browse**, type `<matlab>\bin\win32` as the filename, and select `Add-ins (.dll)` from the **Files of Type** pulldown list.

---

**Note** If Windows is configured to "hide system files," `.dll` files will not be displayed even though they are present on disk. You must change your view options to disable this feature.

---

d Select the `MATLABAddin.dll` file and click **Open**.

e Check **MATLAB for Visual Studio** on the **Add-ins and Macro Files** list and click **Close**. The floating MATLAB add-in for Visual Studio toolbar appears. The checkmark directs MSVC to automatically load the add-in when you start MSVC again.

---

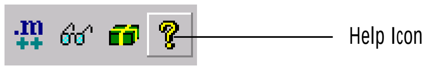
**Note** To run the MATLAB add-in for Visual Studio on Windows 95 or Windows 98 systems, add this line to your `config.sys` file.

```
shell=c:\command.com /e:32768 /p
```

---

For additional information on the MATLAB add-in for Visual Studio:

- See the `MATLABAddin.hlp` file in the `<matlab>\bin\win32` directory, or
- Click on the Help icon in the MATLAB add-in for Visual Studio toolbar.



### Command Line Override

To specify an option on a one-time basis to the `mex` script, you can use the `mex` command line override feature. For example, the string `/WX` is a Microsoft Visual C/C++ compiler option that causes warnings to be treated as errors. The following statement uses the `mex` command line override feature.

```
mex yprime.c COMPFLAGS#"$COMPFLAGS /WX"
```

The string `COMPFLAGS#"$COMPFLAGS /WX"` tells `mex` to take the default value for `COMPFLAGS` and append ` /WX` (space slash W X) to it. To get a list of all environment variables that can be overridden using this feature, use the command

```
mex -v
```

Any variable name in all uppercase listed in the output can be overridden.

### Serial I/O

MATLAB's serial port interface provides direct access to peripheral devices such as modems, printers, and scientific instruments that you connect to your computer's serial port. This interface is established through a serial port object, which you create with the `serial` function. The serial port object supports functions and properties that allow you to:

- Configure serial port communications
- Use serial port control pins
- Write and read data
- Use events and actions
- Record information to disk

**Note** The serial port interface is supported only for Microsoft Windows, Linux, and Sun Solaris platforms.

## Creating Graphical User Interfaces – Features

### New Graphical User Interface Development Environment

GUIDE, MATLAB's graphical user interface development environment has been redesigned since MATLAB 5.3. The GUIDE toolset for creating graphical user interfaces (GUIs), or visual interfaces, consists of:

- Layout Editor – add and arrange objects in the figure window.
- Alignment Tool – align objects with respect to each other.
- Property Inspector – inspect and set property values.
- Object Browser – observe a hierarchical list of the Handle Graphics objects in the current MATLAB session.
- Menu Editor – create window menus and context menus.

You access all of these tools from the Layout Editor. To start the Layout Editor, use the `guide` command.

### New Code Architecture

GUIDE now employs a FIG-file to save layout information separately from the M-file that programs the GUI. This approach has several advantages:

- Faster GUI loading
- Generated code provides the framework for application M-files with application options to include a number of useful GUI programming techniques, including:
  - Cross-platform compatibility with respect to GUI figure size, screen location, colors, and fonts
  - Access to object handles without saving global variables or using `findobj`
  - Single or multiple instances of GUI
  - Function prototypes for callback routines
  - GUI switchyard approach to the application program

See “Creating Graphical User Interfaces” in the MATLAB documentation for more information.

## Major Bug Fixes

MATLAB 6.0 includes several bug fixes, including the following descriptions (online only) of particularly important bug fixes.

### Figure KeyPressFcn

The figure `KeyPressFcn` now generates an event for all keys pressed. Previously, modifiers keys (such as, **Control** and **Esc**) did not generate an event to executed the `KeyPressFcn` callback.

## Platform Limitations for HP and IBM

The MATLAB functionality described in these *Release Notes* and in the MATLAB documentation applies to MATLAB 6.0 with the exception of the limitations listed below for the HP and IBM platform. For background information, see “Platform Limitations” on page 1-17:

- “Development Environment Limitations” on page 2-43
- “Mathematics Features” on page 2-11
- “Graphics and Printing Limitations” on page 2-45
- “GUIDE Limitations” on page 2-45

**Note** To run MATLAB on HP-UX 11.0, you must install a patch available from Hewlett-Packard. To get the patch, go to `www.itrc.hp.com`, the IT Resource Center page. The patch is available to registrered customers from the individual patches link. The patch name is:
PHSS_21959 1.0 X/Motif 32 bit
Runtime 2000 Periodic Patch.

### Development Environment Limitations

The MATLAB desktop and most of the development environment tools are not available on the HP and IBM platform. Following are the specific limitations for each tool and available alternatives.

| Feature | Limitation and Alternatives |
|---|---|
| Desktop | Not supported. Instead, the MATLAB prompt appears in an X window. Use function alternatives for various tools. |
| Array Editor | Not supported. Instead, view and edit variables at the command line. |
| Command History | Not supported. To recall previous lines, use the up arrow key, or use the `diary` function or the `logfile` startup option. |

| Feature | Limitation and Alternatives (Continued) |
|---|---|
| Current Directory browser | Not supported. Use function alternatives documented for the Current Directory browser, including `cd`, `delete`, `ls`, and `mkdir`. |
| Editor/ Debugger | Not supported. For editing M-files, use another text editor, such as Emacs, and the `edit` function – see the `edit` reference page to specify the other text editor. To debug M-files, use MATLAB debugging functions. |
| Help browser | Not supported. Help displays in your default browser. The Index and Search features are not available. You will get a broken link message from your browser if you try to access documentation that you don't have installed. |
| Import Wizard | Not supported. Use import function equivalents for the various features. |
| Launch Pad | Not supported. Access documentation and demos using functions, such as `help` and `demo`. |
| Preferences | Not supported. Set location of help files using `docopt`. |
| Set Path dialog box | Not supported. Use the `path`, `addpath`, and `rmpath` functions instead. |
| Source Control menu items | Not supported. Use the `checkin`, `checkout`, `cmopts`, and `undocheckout` functions instead. |
| Workspace browser | Not supported. Use `who`, `whos`, `save`, `load`, and `clear` functions instead. |

## Mathematics Limitations

The Basic Fitting interface is not supported. Instead, use curve fitting functions.

### Graphics and Printing Limitations

| Feature | Limitation and Alternatives |
|---|---|
| Property Editor | Not supported. Similar graphical user interfaces provide access to figure, line and text objects. Use the `set` and `get` functions to modify Handle Graphics object properties. |
| Data Statistics | Not supported. |
| Printing | Uses the Release 11 **Page Setup**, **Print Setup**, and **Print** dialog boxes. For information about these interfaces, see "Printing MATLAB Graphics" in the online MATLAB documentation. |

### GUIDE Limitations

The Application Options dialog configures the M-file generated by GUIDE. If you comment-out lines of code in this M-file, the Application Options dialog may not be able to correctly represent the state of the generated code. In addition, the M-file may not function correctly.

If you want to remove code that has been added due to the settings of the Application Options dialog, you should make changes to the Application Options dialog. GUIDE will update the M-file the next time you save your layout.

If you want to prevent certain lines of generated code from executing, you should remove these lines from the M-file instead of making these lines into comments.

Note that these limitations apply only to code generated by GUIDE, not user-written code.

## Upgrading from an Earlier Release

This section describes the upgrade issues involved in moving from MATLAB 5.3 (Release 11.0) to MATLAB 6.0.

For information about upgrading from a release earlier than 5.3, see *Release 11 New Features*.

This section about upgrading from an earlier release is organized into the following subsections:

- "Development Environment Issues" on page 2-46
- "Programming and Data Types Issues" on page 2-46
- "External Interfaces/API Issues" on page 2-57
- "Creating Graphical User Interfaces – Upgrade Issues" on page 2-62

### Development Environment Issues

#### Preferences

Your preferences from Release 11 are not maintained in Release 12. Specify new preferences for Release 12.

The **Tools** menu in the Release 11 Editor/Debugger, which allowed you to create customized menu items, is no longer supported.

#### The dos Function Now Returns an Accurate Error Status

The dos function now returns a nonzero status when it encounters an error condition. In the past, dos always returned zero, indicating success, regardless of whether an error had actually occurred.

### Programming and Data Types Issues

#### Evaluating Function Names

The `feval` function can be used with either function handles or function name strings. Evaluation of function handles is preferred over evaluation of the function name. For example, of the following two lines of code that evaluate the humps function, the second is considered to be the preferable mechanism to use.

```
feval('humps', 0.5674);      % Uses a function name string

feval(@humps, 0.5674);       % Uses a function handle
```

To support backward compatibility, `feval` still accepts a function name string as a first argument and evaluates the function named in the string. However, function handles offer you the additional performance, reliability, and source file control benefits listed in the earlier section "Programming and Data Types Features".

### Argument Name Length for feval, load, and save

Most names in MATLAB are truncated to 31 characters. In previous versions of MATLAB, this was not true for arguments passed to the `feval`, `load`, and `save` functions. As a result, MATLAB was able to distinguish between names passed into these functions when these names were identical up to the thirty first character but unique at some point after that.

In MATLAB 6.0, arguments (other than filenames) that are passed to `feval`, `load`, and `save` are truncated to a maximum of 31 characters. If your code uses names in which the first 31 characters are identical, MATLAB will no longer distinguish between these names within these functions.

Filenames do not hold to this rule. MATLAB uniquely identifies files with names of any length.

### Limit on Line Length of M-code Removed

There is no longer any limit on the length of a line of M-code in MATLAB.

### feval Accepts Only Simple Function Names

In previous versions of MATLAB, you could call `feval` passing the full path to the function to be evaluated. You could also call a subfunction or private function directly using this method. For example, the following command executes a subfunction that is defined within the file, `subfile`.

```
feval('subfile/mysubfunction')
```

In MATLAB 6.0, `feval` accepts only simple function names. To evaluate subfunctions and private functions from beyond their usual scope, you can use function handles.

### Assigning to an Empty Structure Field

Attempting to assign a structure to a field of another structure now results in an error if both of the following conditions are true:

- The field being assigned to has been initialized to an empty matrix.
- The field being assigned to is referenced in the assignment using an array index.

For example,

```
mystruct.emptyfield = [];
mystruct.emptyfield(1) = struct('f1', 25);
??? Conversion to double from struct is not possible.
```

This operation did not return an error in previous versions of MATLAB.

### Comparing With an Empty Structure Field

Attempting a string comparison between a string and a structure field now results in an error, if all of the following conditions are true:

- The structure field being compared has been initialized to an empty cell array.
- The structure field being compared is two or more levels down in the structure array.
- The structure array is referenced in the `strcmp` using an array index.

```
mystruct.field1 = struct('field2',{});
strcmp('string', mystruct(1).field1.field2);
??? Error using ==> strcmp
Not enough input arguments.
```

This operation did not return an error in previous versions of MATLAB.

### Separators Are Now Required Between Array Elements

MATLAB 5 allowed two or more elements of a cell array, vector, or matrix to appear together without any separating text such as a comma or whitespace, under certain circumstances. For example, the following were all legal expressions.

```
[[1][2]]                      % equivalent to [ 1 2 ]
['hello'sprintf(' world')]  % equivalent to [ 'hello world' ]
{{1}fft(2)}                   % equivalent to { {1} 2 }
```

MATLAB 6.0 issues a warning if it detects this situation upon first loading the M-file into memory. Future versions of MATLAB will likely error at that point instead. To remedy the situation, place a comma or whitespace between the elements of the cell array, vector or matrix, as in

```
[[1],[2]]
['hello',sprintf(' world')]
{{1},fft(2)}
```

or

```
[[1] [2]]
['hello' sprintf(' world')]
{{1} fft(2)}
```

The following command displays the information given above.

```
help matrix_element_separators
```

---

**Note** The section “Running a Syntax Check on M-Files” on page 2-26 describes how MATLAB can help you locate missing element separators in your M-files.

---

### Logical AND and OR Precedence

Starting in MATLAB 6.0, the precedence of the logical AND (&) and logical OR (|) operators now obeys the standard relationship, where AND has a higher precedence than OR. This precedence agrees with the formal rules of Boolean algebra as implemented in most other programming languages, as well as in Simulink and Stateflow.

Previously, MATLAB would incorrectly treat the expression

```
y = a&b | c&d
```

as

```
y = (((a&b) |c) &d);
```

It now correctly treats it as

```
y = (a&b) | (c&d);
```

The only case where the new precedence will impact the result obtained is when | appears before & within the same expression (without parentheses). For example,

```
y = 1 | x & 0;
```

In MATLAB 5.3 and earlier, this statement would yield 0, being evaluated as

```
y = (1 | x) & 0;
```

In MATLAB 6.0 and beyond, this expression yields a 1 as the result, being evaluated as

```
y = 1 | (x & 0);
```

---

**Note** We strongly recommend that you add parentheses to all expressions of this form to avoid any potential problems with the interpretation of your code between different versions of MATLAB.

---

A feature has been provided to allow the system to produce an error for any expression that would change behavior from the old to the new interpretation.

```
feature('OrAndError', 0)  % Warning for any expression that
                          % changed behavior (default)

feature('OrAndError', 1)  % Error for any expression that
                          % changed behavior
```

### Breaking From Try-Catch in a Loop

The MATLAB `break` function terminates the execution of a `for` or `while` loop. If a `for` or `while` loop contains a `try` – `catch` statement, with a `break` placed within the `try` – `catch`, the `break` statement terminates the `for` or `while` loop, not the `try` – `catch`.

In the following example, a `break` has been placed in a `try` – `catch` statement and the `try` – `catch` is within a `for` loop. In MATLAB 5, execution of the `break` causes an incorrect branch from the `try` to the line, `x = 5`. The `for` loop is not terminated, in this case.

In MATLAB 6.0, execution of the `break` within the `try` – `catch` terminates the `for` loop. The next line executed following the `break` is `y = 10`.

```
for i = 1:10
        try
            <statement>
            break
        catch
            <statement>
        end
    x = 5                % Break to here in MATLAB 5
    end
y = 10                   % Break to here in MATLAB 6.0
```

### Specifying Field Names with SETFIELD and GETFIELD

The `setfield` and `getfield` functions set and get the value of a specified field in a MATLAB structure array. In MATLAB 5, you could use the following undocumented syntaxes to set `fieldn` to the value, `newvalue` and to read back that value.

```
setfield(structurename.field1.fieldn,newvalue)
getfield(structurename.field1.fieldn)
```

These syntaxes are not supported in MATLAB 6.0. You should use the following instead.

```
setfield(structurename,'field1','fieldn',newvalue)
getfield(structurename,'field1','fieldn')
```

For example,

```
a.b.c = 7;

a = setfield(a,'b','c',10);

getfield(a,'b','c')

ans =

    10
```

### Using GETFIELD on a Structure Array

The `getfield` function returns the contents of a specified field in a MATLAB structure array. In MATLAB 5, you would get an error if you requested more than one return value on a single `getfield` call. For example, the following call to `getfield` requests the contents of the `x` field for all three elements of the structure array, `a`.

```
a(1).x=5;
a(2).x=12;
a(3).x=27;

getfield(a,'x')
```

In MATLAB 6.0, requesting more than one output value from `getfield` returns a single output value and does not result in an error. The value returned is the first of all output values requested in the call. In the example above, `getfield` now returns the contents of `a(1).x`, because that is the first field to which `(a,'x')` applies.

```
getfield(a,'x')
ans =
    5
```

You can obtain the contents of another structure array element by specifically indexing to that element. For example,

```
getfield(a,{3},'x')
ans =
    27
```

### Temporary Variables Are Anonymous

Individual cells of a cell array, fields of a MATLAB structure, and all temporary variables are now guaranteed to be anonymous.

For example, the following function returns the name of the first input argument passed to `myfun`.

```
function myfun(a)
inputname(1)
```

In MATLAB 5, a temporary variable, used to pass the value of `eval('x')`, retains the variable name, `x`. In MATLAB 6.0, this variable is anonymous.

```
x = [1 2 3];

myfun(eval('x'’))        % Running MATLAB 5, ...
x                        %   the output is x

myfun(eval('x'’))        % Running MATLAB 6.0, ...
''                       %   the output is ''
```

This change may result in a small difference in the size of some MAT files.

### Platform Name Changes for the computer Function

The names returned by the `computer` function for some UNIX platforms have changed for MATLAB 6.0. If you have code that uses the `computer` function for special handling on some UNIX platforms, that code may behave differently because of these platform name changes.

| Platform | Pre-6.0 String | String for MATLAB 6.0 |
|---|---|---|
| Linux | `LNX86` | `GLNX86` |
| SGI | `SGI` | `SGI` (no change) |
| SGI64 | `SGI64` | `SGI` |
| HPUX 11.x | `HP700` | `HPUX` |
| HPUX 10.20 | `HP700` | `HP700` (no change) |

### Processes and Memory Usage on Linux

On Linux, you may notice that MATLAB can start up a large number of processes, even to handle simple tasks. These are actually threads, but due to the “one process per thread” model on Linux, they appear as processes. When you use the ps or top commands in Linux to show the processes running on your system, you will see each of these threads listed as a separate process.

It is not uncommon for Linux to create a large number of threads for an application. Even a simple Java program using AWT and native threads on Linux will create nine threads.

Also displayed is the amount and percentage of memory used by each thread. You should be aware that the memory used by the threads started by MATLAB is shared between threads. The memory consumption reported for each thread in this display actually represents the total memory used by all of the MATLAB threads together.

## Obsolete Functions

The following MATLAB functions have either been renamed or have become obsolete. For backwards compatibility, they have not been removed from the language at this time. However, these functions may be removed in a future release, so you are encouraged to discontinue use of the functions, or use the functions that replace them.

| Function | Description |
|---|---|
| `errortrap` | Replace with `try ... catch` |
| `flops` | Floating-point operation count. Now returns `0`. |
| `fmin` | Minimize a function of one variable. Replace with `fminbnd`. See instructions for converting your code in "Function Functions" in the MATLAB documentation. |
| `fmins` | Minimize a function of several variables. Replace with `fminsearch`. See instructions for converting your code in "Function Functions" in the MATLAB documentation. |
| `foptions` | Default parameters used by the optimization routines. Replace with `optimget`, `optimset`. See instructions for converting your code in "Function Functions" in the MATLAB documentation. |
| `interp4`, `interp5`, `interp6` | Various two-dimensional data interpolation. Use `interp2` instead. |
| `isdir` | Replace with `exist` |
| `isieee` | Return logical true (`1`) on machines with IEEE arithmetic and logical false (`0`) on machines without IEEE arithmetic. Now returns `1` in all cases. |
| `isstr` | Replace with `ischar` |

| Function | Description (Continued) |
|---|---|
| meshdom | Generate X and Y arrays for three-dimensional plots. Use meshgrid instead. |
| nnls | Nonnegative least squares. Replace with lsqnonneg. See instructions for converting your code in "Function Functions" in the MATLAB documentation. |
| quad8 | Numerically evaluate integral. Replaced by quadl. |
| saxis | Sound axis scaling. Now has no effect on the output of sound. |
| setstr | Replace with char |
| str2mat | Replace with char |
| table1 | One-dimensional table lookup. Use interp1 instead. |
| table2 | Two-dimensional table lookup. Use interp2 instead. |

### The errortrap Function Is Disabled But Not Removed

The errortrap builtin function is obsolete in MATLAB 6.0. It is disabled, but has not been removed from the language for purposes of backward compatibility.

### Specifying Ordinary Differential Equation (ODE) Problems

The ODE problem components that were passed to the solver through an ODE file now are passed directly as arguments or need to be specified in an options structure. The new syntax is

```
solver(@odefun,tspan,y0,options,p1,p2,...)
```

where odefun, tspan and y0 are required arguments. See the ODE solver and odeset reference pages for details.

MATLAB 6.0 supports use of an ODE file for backwards compatibility, but new functionality is available only with the new syntax.

### Output from Background and Foreground Commands (UNIX)

In Release 12 on UNIX platforms, a background command (i.e., any system command after which you add a &), such as

```
! cat startup.m &
```

no longer produces any output. Prior to Release 12, a background command sent output to the command window.

If you need to see the output from a command, either do not make the command a background command (i.e., remove the &), or run the background command in a separate xterm. To start another xterm, issue the following command.

```
! xterm &
```

In Release 12, foreground functions (i.e., non-background functions) send their output to the diary, if the `diary` function has been issued. The output is also displayed in the command window (prior to Release 12, foreground function output was only displayed in the command window).

#### matlab_helper Process

To make the `!` and `unix` commands operate more efficiently, in Release 12 MATLAB creates a secondary process, called `matlab_helper`, at startup.

This `matlab_helper` contains those elements of MATLAB necessary to run the `!` and `unix` commands.

## External Interfaces/API Issues

#### Recompile Fortran MEX-Files

Due to changes in the MATLAB/Fortran interface, Fortran MEX-files that were built with versions of MATLAB prior to Release 12 (MATLAB 6.0) will not work unless they are rebuilt with MATLAB 6.0.

#### MEX File Compatibility

The Release 12 MATLAB Application Program Interface (API) has several compatibility issues relating to existing MEX-files:

- C MEX-files built under MATLAB 4.x may work with MATLAB 6.0, but are no longer supported. You should rebuild these files using MATLAB 6.0.
- You will need to rebuild all MATLAB 5.x C MEX-files on the IBM_RS in MATLAB 6.0 in order to run those MEX-files in MATLAB 6.0.
- MATLAB 6.0 C and Fortran MEX-files will not run on previous versions of MATLAB.

#### Preference File Location for mex and mbuild

On Windows platforms, the `mex` and `mbuild` commands now look for their preferences file in

```
C:\Winnt\profiles\<user>\application data\mathworks\matlab\R12
```

as opposed to the location used by previous versions.

```
C:\Winnt\profiles\<user>\application data\mathworks\matlab
```

On UNIX platforms, the `mex` and `mbuild` commands now look for their preferences file in

```
/home/<user>/.matlab/R12
```

Therefore, to restore functionality to `mex` and `mbuild`, you will need to either:

- Run `mex -setup` and/or `mbuild -setup` (easy way), or
- Manually move your options files from the directory used for preferences in previous releases to the directory now used.

### Linux MEX-files

For Release 12, you must rebuild Release 11 MEX-files on Linux.

Due to compatibility issues between versions of the GNU standard C libraries, for Linux the file extension for MEX-files is now `.mexglx`. The file extension for Release 11 was `.mexlx`. This change in file extensions means that the Release 11 MEX-files for Linux will be ignored by Release 12, unless you rebuild the Release 11 MEX-files in Release 12.

### SGI MEX-files

You will need to rebuild all MATLAB 5.x C MEX-files on the SGI in MATLAB 6.0 in order to run those MEX-files in MATLAB 6.0. There is no longer a difference between SGI and SGI64 MEX-files; they are all built as `n32`.

### Unsupported Compilers

MATLAB no longer supports the following compilers:

- Digital Visual Fortran version 6.0
- Microsoft Visual C/C++ version 4.2

In addition, The MathWorks will drop support for following compilers in a future version of MATLAB. The MathWorks no longer tests with these compilers:

- Watcom C/C++ version 11

- Watcom C/C++ version 10.6

### Using the move Command for ActiveX Controls

In previous versions of MATLAB, the move command returned its first two output variables in reversed order. In the following command,

```
pos = move(h);
```

the returned value `pos`, was `[y x xsize ysize]` where it should have been `[x y xsize ysize]`.

This has been corrected in MATLAB 6.0. If you have changed your code to accommodate the reversed order in earlier versions of MATLAB, you should correct the order of these variables for MATLAB 6.0.

### Return Values for Methods Invoked on ActiveX Objects

In previous releases of MATLAB, invoking a method on an ActiveX object always returned a value of type `double`. Given the example,

```
h = actxcontrol('MWSAMP.MwsampCtrl.1');
val = invoke(h, 'GetI4');
```

Instead of returning an `int32` value in `val`, MATLAB converted the type of the object obtained from COM to a `double`.

MATLAB 6.0 returns the same type of object that was passed to it from COM during method invocation. For example, in R12, `val` will be an `int32` value instead of a `double`.

This was done to be consistent with the method signature for a given method. The method signatures for all methods of an interface can be obtained using `invoke` on the handle to an ActiveX object. For example,

```
invoke(h)
```

will list all methods and their signatures for the `mwsamp` control.

If you prefer the former behavior, where MATLAB converts the return value to `double`, you will need to explicitly call `double` on the method parameter you wish to convert. For example,

```
val = double(invoke(h,'GetI4'));
```

yields the old behavior.

### mex Function Now Returns Accurate Error Status

The mex syntax

```
mex myprog.c
```

now throws an error when it encounters an error condition.

The mex syntax

```
stat = mex('myprog.c')
```

now returns a nonzero value to stat when it encounters an error condition.

In the past, on Microsoft Windows platforms, mex always either successfully exited or returned zero (indicating success), regardless of whether an error had actually occurred.

To ensure code from before Release 12 works properly in Release 12, either use try/catch logic to deal with error conditions, or use a form of mex that returns an error status instead of throwing an error. Specifically

```
try
  mex something.c
catch
  disp(‘something failed’);
end
```

or

```
status = mex(‘something.c’);
if status ~= 0
  disp(‘something failed’);
end
```

### Using engEvalString with GUI-Intensive Applications

If you have graphical user interface (GUI) intensive applications that execute a lot of callbacks through the MATLAB engine, you should force these callbacks to be evaluated in the context of the base workspace. Use evalin to specify that the base workspace is to be used in evaluating the callback expression, as follows.

```
engEvalString(ep, ”evalin('base', expression)”)
```

Specifying the base workspace in this manner ensures that MATLAB will process the callback correctly and return results for that call.

This does not apply to computational applications that do not execute callbacks.

### Starting MATLAB in gdb on GLINUX

If you debug MATLAB using gdb on a GLINUX system, your session will stop execution whenever it creates a new thread. You can avoid these interruptions by telling the system not to stop on SIGCONT events. You can then proceed to debug your MATLAB code using nodesktop mode. You will still be notified whenever a new thread is created, but your session will continue to run without interruption.

Use the following statement in gdb to turn off triggering on SIGCONT events.

```
handle SIGCONT nostop
```

You can choose to debug in gdb without specifying SIGCONT nostop, but you will have to type continue at each interruption to proceed with your session. The following is a sample debug session in which SIGCONT nostop is specified.

```
% matlab -Dgdb

(gdb) handle SIGCONT nostop
Signal Stop Print Pass to program Description
SIGCONT No   Yes   Yes             Continued

(gdb) run -nodesktop
[New Thread 4219]
Program received signal SIGCONT, Continued.
[New Thread 4220]
Program received signal SIGCONT, Continued.
   .
   .
   .
```

### MEX-File Extension Changes

The MEX-file extensions have changed for the Linux, SGI, and HP700 platforms for MATLAB 6.0.

| Platform | Pre-6.0 Extension | Extension for MATLAB 6.0 |
|---|---|---|
| Linux | `.mexlx` | `.mexglx` |
| SGI | `.mexsg` | `.mexsg` (no change) |
| SGI64 | `.mexsg64` | `.mexsg` |
| HPUX 11.x | `.mexhp7` | `.mexhpux` |
| HPUX 10.20 | `.mexhp7` | `.mexhp7` (no change) |

### Obsolete C Language MEX Functions

The following API function is obsolete and should not be used in MATLAB programs. This function may not be available in a future version of MATLAB.

```
mexAddFlops
```

## Creating Graphical User Interfaces – Upgrade Issues

### Editing Version 5 GUIs with Version 6 GUIDE

In Version 5 GUIDE, GUI layouts were saved as MAT-file/M-file pairs. In Version 6, GUIDE saves GUI layouts as FIG-files and M-files. The GUI layout is defined in the FIG-file; there is no generated M-file containing layout information.

# Known Software and Documentation Problems

This section updates the MATLAB 6.0 documentation set, reflecting known MATLAB 6.0 software and documentation problems.

This section about software and documentation problems is organized into the following subsections:

- “Development Environment Problems” on page 2-63
- “External Interfaces/API Problems” on page 2-65
- “Graphics Problems” on page 2-65
- “GUIDE Problems” on page 2-66
- “Documentation Updates” on page 2-66

## Development Environment Problems

### Many Open Windows Can Cause a Crash or Hang (Windows 95/98/Me)

On Microsoft Windows 95/98/Me platforms, if you keep many windows open, MATLAB may crash or hang. For example, if you keep open about 12 Stateflow windows, or 25 figure windows, or 50 Simulink windows, you may experience this problem. Note that these numbers are only estimates; the actual number of open windows that may cause this problem depends on the resources currently in use by other components and applications.

### Maximized Desktop Window Not Remembered on Startup

When you start MATLAB, the desktop configuration is the same as when you last closed MATLAB. However, if the desktop was maximized when you closed MATLAB, it is not maximized upon startup.

### Workspace Browser with Over 1000 Variables

If there are over 1000 variables in the workspace and the Workspace browser is open, you might experience performance problems. It is suggested you close the Workspace browser if you expect to have over 1000 variables in the workspace.

### Help Browser Doesn’t Support Mouse Wheel

The wheel on your mouse will not work in the Help browser.

### UNIX Display Problems when UNIX Client and Server Platforms Differ

If you run on UNIX and the platform for the server is different than that for the client, there may be problems with the display of graphics on the client. See the Technical Support Web page for a solution that lists the combinations tested and any known display problems with them.

### UNIX Help Browser Search Results Not Highlighted

On UNIX systems, when you perform a full text search using the Help browser, the search returns all of the pages that contain the search term, however the search term is not highlighted when you view a page. To find the term on a page, use the **Find in page** field in the Help browser display pane.

### Sun Solaris 16-Bit Display Not Supported

Sun's Java VM for Solaris does not support 16-bit displays. Therefore you cannot use this configuration with Release 12. Use another display mode instead.

### Sun Solaris Arrow Keys Not Working

On some Sun Solaris systems, the arrow keys on the main keyboard are not working properly. Instead, try the arrow keys in the numeric keypad.

### ALPHA Shortcut Problems When Using Emacs Key Bindings in Editor

On the Alpha platform, if you set the Editor/Debugger preference for key bindings to Emacs, the shortcuts for **Undo** (**Ctrl+_**) and **Copy** (**~+W**) do not work.

### Display Problems with Xoftware

If you use Xoftware on a PC to run MATLAB on a UNIX platform, you need to do the following to avoid display problems:

1. Go to the Xoftware **Control Panel**.
2. From the **Options** menu, select **Configuration**.
3. Select the **Window** tab.
4. From the **Options** listing, select `Concurrent Window Manager`.
5. Under **Settings**, select **Off**.

6 Click **OK**.

## External Interfaces/API Problems

### Available Serial Ports on Windows 95 and Windows 98

On Windows 95 and Windows 98 platforms, you can access no more than four serial ports. The four serial ports are labeled COM1 through COM4.

The COM1 and COM2 serial ports are standard components of your Windows platform. You can add two additional serial ports, COM3 and COM4. If you have more than four serial ports, you can access only the first four ports. If you try to access a port that does not exist, MATLAB will return the following error message after the `fopen` function is issued.

```
s = serial('COM3');
fopen(s)
??? Error using ==> serial/fopen
Error using ==> fopen
Cannot connect to the COM3 port. Possible reasons are another
application is connected to the port or the port does not exist.
```

### Accessing Serial Ports on Solaris

If you repeatedly open and close one or more serial ports on Solaris, MATLAB will become unresponsive. To minimize the chance of encountering this problem, you should:

- Use only one serial port on your platform.
- Connect the serial port object to your device once per MATLAB session.

## Graphics Problems

### The uimenu Function on Linux

If you use a string containing embedded spaces with the `uimenu` function, MALTAB produces a segmentation violation on the Linux platform.

Here is an example that reproduces the problem.

```
hfig = figure;
uimenu = uimenu(hfig,’Label’,[‘Test’ setstr(9) ‘Menu’]);
```

## GUIDE Problems

This section lists know problems with GUIDE:

- If you try to open a file from the **File** menu of the Layout Editor and then cancel the open operation, MATLAB displays an error dialog. Disregard this message.
- On DEC Alpha, there is no Object Browser available. It will be available in the final release of GUIDE.
- On DEC Alpha, figures activated from the Layout Editor are not the correct size.

## Documentation Updates

### Ctrl-q Quits Without Issuing a Warning

If MATLAB is the active window in your system, using **Ctrl-q** forces MATLAB to quit without issuing any warning.

### interp1 Extrapolation of Out of Range Values

A new argument enables `interp1` to perform extrapolation for out of range values for all methods. It also enables you to specify a scalar to be returned for out of range values.

The PDF version of the `interp1` reference page incorrectly states that the default for all methods is for `interp1` to perform extrapolation for out of range values. In fact, `interp1` performs extrapolation as the default only for the `'spline'`, `'pchip'`, and `'cubic'` methods. For all other methods, it returns `NaN` for out of range values. This behavior is unchanged from Version 5.

The HTML reference page for `interp1` is correct.

# Simulink 4.0 Release Notes

**New Features** . . . . . . . . . . . . . . . . . . . . . 3-2
Simulink Editor . . . . . . . . . . . . . . . . . . . . . 3-2
Modeling Enhancements . . . . . . . . . . . . . . . . . . 3-5
Simulink Debugger . . . . . . . . . . . . . . . . . . . . 3-6
Block Library . . . . . . . . . . . . . . . . . . . . . 3-6
SB2SL . . . . . . . . . . . . . . . . . . . . . . . . 3-9

**Platform Limitations for HP and IBM** . . . . . . . . . 3-11

**Upgrading from an Earlier Release** . . . . . . . . . . 3-12
Port Name Property . . . . . . . . . . . . . . . . . . 3-12

## New Features

This section introduces the new features and enhancements added in Simulink 4.0 since Simulink 3.0 (Release 11.0).

For information about Simulink new features that are incorporated from prior releases, see *Release 11 New Features* (enhancements introduced between Release 9.0 and Release 11.0).

---

**Note** For information about closely related products that extend Simulink, see the sections on the Simulink Performance Tools and the Real-Time Workshop, as well as information about the blocksets from The MathWorks.

---

This section about new Simulink features is organized into the following subsections:

- "Simulink Editor" on page 3-2
- "Modeling Enhancements" on page 3-5
- "Simulink Debugger" on page 3-6
- "Block Library" on page 3-6
- "SB2SL" on page 3-9

### Simulink Editor

This section describes enhancements to the Simulink Editor.

#### Preferences

The Simulink **Preferences** dialog box allows you to specify default settings for many options (see "Setting Simulink Preferences" in *Using Simulink*).

#### Text Alignment

Simulink 4.0 allows you to choose various alignments for annotation text. To choose an alignment for an annotation, select the annotation and then select **Text Alignment** from the editor menubar or context (right-click) menu (see "Annotations" in *Using Simulink*).

### UNIX Context Menus

The UNIX version of Simulink 4.0 now has context menus for block diagrams. Click the right button on your mouse to display the menu.

### Library Link Enhancements

Simulink 4.0 optionally displays an arrow in each block that represents a library link in a model. Simulink 4.0 also allows you to modify a link in a model and propagate the changes back to the library (see “Modifying a Linked Subsystem” in *Using Simulink*).

---

**Note** Simulink displays “Parameterized Link” on the parameter dialog box of a masked subsystem whose parameters differ from the library reference block to which the masked subsystem is linked. This feature, which is not documented in *Using Simulink*, allows you to determine quickly whether a library link differs from its reference.

---

### Find Dialog Box

The **Find** dialog box enables you to search Simulink models and Stateflow charts for objects that satisfy specified search criteria. You can use the dialog box to find annotations, blocks, signals, states, state transitions, etc. To invoke the **Find** dialog, select **Find** from Simulink’s **Edit** menu (see “Searching for Objects” in *Using Simulink*).

### Model Browser

The Model Browser’s toolbar includes the following new buttons:

- Show Library Links

  Shows library links as nodes in the browser tree.
- Look Under Masks

  Shows the contents of masked blocks as nodes in the browser tree.

### Single Window Mode

Simulink now provides two modes for opening subsystems. In multiwindow mode, Simulink opens each subsystem in a new window. In single-window mode, Simulink closes the parent and opens the subsystem (see “Window Reuse” in *Using Simulink*).

### Keyboard Navigation

Simulink 4.0 provides the following new keyboard shortcuts.

| Key | Action |
|---|---|
| **Tab** | Selects the next block in the block diagram. |
| **Shift+Tab** | Selects the previous block in the block diagram. |
| **Ctrl+Tab** | Cycles between the browser tree pane and the diagram pane when the model browser is enabled. |
| **Enter** | Opens the currently selected subsystem. |
| **Esc** | Opens the parent of the current subsystem. |

### Enhanced Library Browser

The Library Browser incorporates the following new features:

- Blocks no longer appear as browser tree nodes. Instead, they appear as icons in the preview pane.
- The preview pane has moved from beneath the library tree pane to beside the tree pane. You can create instances of blocks displayed in the preview pane by dragging them from the preview pane and dropping them in a model.
- Splitter bars now divide the browser’s panes, allowing the panes to be independently resized.
- Double-clicking a block’s icon opens the block’s parameter dialog box with all fields disabled. This allows you to inspect, but not modify, a library block’s parameters.
- Double-clicking a library block opens the library in the preview pane.
- You can now insert a block in the topmost model on your screen by right-clicking the block in the preview pane and selecting **Insert in...** from the context menu that appears. If no model is open or the topmost model is a locked library, the Library Browser offers to create a model in which to insert the block.

- The browser now contains a menu with **File**, **Edit**, and **Help** options.
- The block help text pane has moved from the bottom of the Library Browser to the top.
- Selecting **Find** from the Library Browser's **Edit** menu displays a modeless **Find** dialog box.
- The browser's search feature is much faster and supports regular expressions.

### Help Menus

Simulink 4.0 adds a Help menu to the menu bar on model and library windows. The help item on a block context menu displays a help page for the block. The help item on the model context menu displays the first page of the *Using Simulink* book.

## Modeling Enhancements

### Hierarchical Variable Scoping

This release extends Simulink's ability to resolve references to variables in masked subsystems. Previously Simulink could resolve references only to variables in a block's local workspace.With this release, Simulink will resolve references to variables located anywhere within the workspace hierarchy containing the block (see “The Mask Workspace” in *Using Simulink*).

**Note** In some cases, hierarchical scoping will cause some models to behave differently in the current release than in previous releases of Simulink.

### Matrix Signals

Many Simulink blocks can now accept or output matrix signals. A matrix signal is a two-dimensional array of signal elements represented by a matrix. Each matrix element represents the value of the corresponding signal element at the current time step. In addition to matrix signals, Simulink also supports scalar (dimensionless) signals and vector signals (one-dimensional arrays of signals). Simulink can optionally thicken (select **Wide Lines** from the **Format** menu) and display the dimensions of lines (select **Line Dimensions** from the **Format** menu) that carry vector or matrix signals. When you select the **Line**

**Dimensions** option, Simulink displays a label of the form `[r x c]` above a matrix signal line, where `r` is the number of rows and `c` is the number of columns. For example, the label `[2 x 3]` indicates that the line carries a two-row by three-column matrix signal.

You can use Simulink source blocks, such as a Sine Wave or a Constant block, to generate matrix signals. For example, to create a time-invariant matrix signal, insert a Constant block in your model and set its Constant Value parameter to any MATLAB expression that evaluates to a matrix, e.g., `[1 2; 3 4]`, that represents the desired signal. See “Working with Signals” in Using Simulink for more information.

### Simulink Data Objects

Simulink data objects allow a model to capture user-defined information about parameters and signals, such as minimum and maximum values, units, and so on (see “Working with Data Objects” in *Using Simulink*).

### Block Execution Order

Simulink now optionally displays the execution order of each block on the model’s block diagram (see “Displaying Block Execution Order” in Using Simulink).

## Simulink Debugger

This section describes enhancements to the Simulink debugger.

### GUI Debugger Interface

Simulink 4.0 introduces a graphical user interface (GUI) for the Simulink Debugger. For more information, see “Simulink Debugger” in the online help for Simulink (see “Simulink Debugger” in *Using Simulink*).

## Block Library

This section describes enhancements to the Simulink block libraries.

### Product Block

The Product block now supports both element-by-element and matrix multiplication and inversion of inputs. The block’s parameter dialog includes a new **Multiplication** parameter that allows you to specify whether the block should multiply or invert inputs element-by-element or matrix-by-matrix.

### Gain Block

The Gain block now supports matrix as well as element-wise multiplication of the input signal by a gain factor. Both input signals and gain factors can be matrices. The block's parameter dialog includes a new **Multiplication** parameter that allows you to choose the following options:

- `K.*u` (element-wise product)
- `K*u` (matrix product with the gain as the left operand)
- `u*K` (matrix product with the gain as the right operand)

### Math Function Block

The Math Function block adds two new matrix-specific functions: transpose and Hermitian. The first function outputs the transpose of the input matrix. The second function outputs the complex conjugate transpose (Hermitian) of the input matrix.

### Reshape Block

Simulink 4.0 introduces the Reshape block, which changes the dimensionality of its input signals, based on an **Output dimensionality** parameter that you specify. For example, the block can change an n-element vector to a 1-by-N or N-by-1 matrix signal and vice versa. You can find the Reshape block in Simulink's Signals & Systems library.

### Multiplexing Matrix Signals

Simulink's Mux, Demux, and Bus Selector blocks have been enhanced to support multiplexing of matrix signals.

### Function Call Iteration Parameter

Simulink 4.0 adds a **Number of iterations** parameter to the Function Call Generator block. This parameter allows you to specify the number of times the target block is called per time step.

### Probing Signal Dimensionality

The Probe block now optionally outputs the dimensionality of the signal connected to its input.

### Configurable Subsystem

The Configurable Subsystem block has been reimplemented to make it easier to use. The configurable subsystem block now has a **Blocks** menu that allows you to choose which block the subsystem represents. To display the menu, select the configurable subsystem and then **Blocks** from the Simulink editor's **Edit** or context (right click) menu.

### Look-Up Table Blocks

This release provides four new Look-Up Table (LUT) blocks.

- Direct Look-Up Table (n-D)
- Look-Up Table (n-D)
- Prelook-Up Index Search
- Interpolation (n-D) Using PreLook-Up

The blocks reside in Simulink's Functions and Tables block library.

### Polynomial Block

The Polynomial block outputs a polynomial function of its input. The block resides in Simulink's Functions and Tables block library.

### Signal Specification

The Signal Specification block allows you to specify the attributes that the input signal must satisfy. If the input signal does not meet the specification, the block generates an error.

### ADA S-Functions

Simulink now supports S-functions coded in ADA. See "Creating Ada S-Functions" in *Writing S-Functions* for more information.

### Bitwise Logical Operator Block

The Bitwise Logical Operator block is a new block that logically masks, inverts, or shifts the bits of an unsigned integer signal. See the online Simulink documentation for details.

### Atomic Subsystems

Simulink 4.0 allows you to designate subsystems as *atomic* as opposed to *virtual*. An *atomic subsystem* is a true subsystem. When simulating a model, Simulink executes all blocks contained by an atomic subsystem block before executing the next block of the containing model (or atomic subsystem).

By declaring a subsystem atomic, you guarantee that Simulink completes execution of the subsystem before executing any other blocks at the same level in the model hierarchy. See "Atomic Versus Virtual Subsystems" in *Using Simulink* for more information.

---

**Note** Conditionally executed subsystems are inherently atomic. Simulink does not allow you to specify them as atomic or virtual.

---

## SB2SL

### SB2SL Extends Code Generation Support

SB2SL, which is included as part of Simulink, allows you to translate SystemBuild SuperBlocks to Simulink models.

For Release 12, SB2SL 2.1 has been enhanced to provide more complete support for use with the Real-Time Workshop. If you use the Real-Time Workshop 4.0 to generate code for models you have converted from SystemBuild to Simulink (using SB2SL), then code is generated for most translated blocks in the model.

The blocks that do *not* support code generation through the Real-Time Workshop 4.0 are:

- ConditionBlock
- Decoder
- Encoder

- GainScheduler
- Interp Table (Archive library)
- ShiftRegister

---

**Note** SB2SL 2.1 also includes a number of important bug fixes.

---

## Platform Limitations for HP and IBM

The following features are not supported on the HP and IBM platforms. For background information, see “Platform Limitations” on page 1-17.

- Simulink Editor’s **Find** dialog

  Use the `find_system` command instead.
- GUI interface to the Simulink Debugger

  Use the command-line interface instead.
- The **View Changes** dialog box for modified library links

  Instead, select the modified link and execute `ld=get_param(gcb,'LinkData')` to get a structure that lists the parameter differences between the library and local instance of the block. Edit this structure and execute `set_param(gcb,'LinkData',ld)` to apply the changes.
- **Parameter** dialog for the Configurable Subsystem block.

  Use the `set_param` command instead to set the block’s parameters.

## Upgrading from an Earlier Release

This section describes the upgrade issues involved in moving from Simulink 3.0 (Release 11.0) to Simulink 4.0.

For information about upgrading from a release earlier than 3.0, see *Release 11 New Features*.

### Port Name Property

In previous releases, the name property of ports and lines referred to the label of the line connected to the port. In the current release, a port's name property refers to the port's (and line's) name, which, in the current release, can differ from the line's label.If you need to get the line's label, invoke

```
get_param(p, 'label')
```

where `p` is the handle of the port.

4

# Stateflow 4.0 Release Notes

**New Features** . . . . . . . . . . . . . . . . . . . . . . 4-2
Stateflow Works with MATLAB Release 12 . . . . . . . . . 4-2
Temporal Logic . . . . . . . . . . . . . . . . . . . . . . 4-2
Subcharts . . . . . . . . . . . . . . . . . . . . . . . . . 4-2
Graphical Functions . . . . . . . . . . . . . . . . . . . 4-3
Symbol Autocreation Wizard . . . . . . . . . . . . . . . 4-3
Command Toolbar . . . . . . . . . . . . . . . . . . . . . 4-3
Navigation Toolbar . . . . . . . . . . . . . . . . . . . . 4-3
Straight Line Transitions . . . . . . . . . . . . . . . . . 4-4
Workspace-Based Data . . . . . . . . . . . . . . . . . . 4-4
Explorer Copy Properties . . . . . . . . . . . . . . . . . 4-4
Library Link Icons . . . . . . . . . . . . . . . . . . . . . 4-4

**Known Software and Documentation Problems** . . . . 4-5
Many Open Windows Can Cause a Crash or Hang
(Windows 95/98/Me) . . . . . . . . . . . . . . . . . 4-5
Calling MATLAB Functions From Stateflow Charts . . . . 4-5

## New Features

This section introduces the new features and enhancements added in Stateflow 4.0 since Stateflow 2.0.

---

**Note** For information about closely-related products that extend Stateflow, see the section about the Stateflow Coder.

---

### Stateflow Works with MATLAB Release 12

Stateflow 4.0 upgrades all features introduced in Stateflow 3.0 to work with Release 12 of the MATLAB product family.

---

**Note** The following features were introduced in Stateflow 3.0.

---

### Temporal Logic

You can now use temporal conditions (before, after, at, every time) to determine the activation of transitions and duration of state activation. Temporal logic provides a simple paradigm for event scheduling.

Temporal conditions allow your Stateflow model to express clearly and simply the time-dependent behavior of a system. See “Temporal Logic Operators” in the *Stateflow User's Guide* for more information.

### Subcharts

Stateflow now allows you to create charts within charts. A chart that is embedded in another chart is called a subchart. A subchart can contain anything a top-level chart can, including other subcharts. In fact, you can nest subcharts to any level. A subchart appears as a labeled block in the chart that contains it. You can create transitions among objects residing in different subcharts existing at the same level or at different levels. A transition that crosses subchart boundaries in this fashion is called a supertransition.

Subcharts enable you to reduce a complex chart to a set of simpler, hierarchically organized diagrams. This makes the chart easier to understand and maintain, without changing the semantics of the chart in any way.

Stateflow ignores subchart boundaries when simulating and generating code from Stateflow models. See “Working with Subcharts” in the *Stateflow User’s Guide* for more information.

## Graphical Functions

A graphical function is a function defined by a flow graph. Graphical functions are similar to textual functions, such as MATLAB and C functions. Like textual functions, graphical functions can accept arguments and return results. You invoke graphical functions in transition and state actions in the same way you invoke MATLAB and C functions. Unlike C and MATLAB functions, however, graphical functions are full-fledged Stateflow objects. You use the Stateflow editor to create them and they reside in your Stateflow model along with the diagrams that invoke them. This makes graphical functions easier to create, access, and manage than textual functions, whose creation requires external tools and whose definitions reside separately from the model. See “Working with Graphical Functions” in the *Stateflow User’s Guide* for more information.

## Symbol Autocreation Wizard

The Symbol Autocreation Wizard helps you to add missing data and events to your Stateflow charts. When you parse the diagram or run the simulation, this wizard detects whenever data and events have not been previously defined in the Stateflow Explorer. The wizard then opens automatically and heuristically recommends attributes for the unresolved data or events to help you to quickly define these symbols.

## Command Toolbar

The Stateflow editor now contains a toolbar containing buttons for Stateflow’s most commonly used editing and simulation commands. The toolbar saves searching through menus for these commands.

## Navigation Toolbar

This toolbar contains buttons for navigating a chart hierarchy.

## Straight Line Transitions

You can now create straight lines between junctions. Transitions that are almost straight are automatically snapped straight during edit time. The snap-to-grid functionality helps align connected junctions vertically and horizontally.

## Workspace-Based Data

Data items can now be initialized from identically named variables in the MATLAB workspace and/or copied back to the workspace at the end of a simulation. Workspace-initialized constants consume no memory in generated code. See “Setting Data Properties” in the *Stateflow User's Guide* for more information.

## Explorer Copy Properties

Stateflow Explorer now allows you to pick up properties from one data/event/ target item and apply them to another data/event/target item or a group of items. This speeds up the process of creating diagrams that have objects with similar sets of properties. See “Transferring Object Properties” in the *Stateflow User's Guide* for more information.

## Library Link Icons

An arrow distinguishes icons of library links from those of actual charts in the Stateflow Explorer. Clicking a library link icon opens the library chart in the Stateflow Editor. Stateflow 4.0 requires Release 12, including Simulink 4.0.

## Known Software and Documentation Problems

This section updates the Stateflow 4.0 documentation set, reflecting known Stateflow 4.0 software and documentation problems.

### Many Open Windows Can Cause a Crash or Hang (Windows 95/98/Me)

On Microsoft Windows 95/98/Me platforms, if you keep about 12 Stateflow windows open, Stateflow may crash or hang. Note that the actual number of open windows that may cause this problem depends on the resources currently in use by other components and applications.

### Calling MATLAB Functions From Stateflow Charts

Calling MATLAB functions from Stateflow charts using either the MATLAB namespace operator `ml.` or by using the function form `ml()` with an argument whose datatype is not `double` causes erroneous results. The workaround is to cast the argument as a `double`. For example:

```
ml.myfun(x,y,1)
```

where `x` and `y` are Stateflow data items whose data types are `uint8` and `int32` and the third argument is an integer has to be rewritten as

```
ml.myfun(double(x),double(y),double(1))
```

inserting casts around the non-double arguments.

# Real-Time Workshop 4.0 Release Notes

**Release Summary** . . . . . . . . . . . . . . . . . . 5-2

**New Features** . . . . . . . . . . . . . . . . . . . . 5-3
Real-Time Workshop Embedded Coder . . . . . . . . . . 5-3
Simulink Data Object Support . . . . . . . . . . . . . 5-3
ASAP2 Support . . . . . . . . . . . . . . . . . . . . 5-4
Enhanced Real-Time Workshop Page . . . . . . . . . . 5-4
Other User Interface Enhancements . . . . . . . . . . 5-4
Advanced Options Page . . . . . . . . . . . . . . . . 5-4
TLC Debugging Page . . . . . . . . . . . . . . . . . 5-5
Model Parameter Configuration Dialog . . . . . . . . . 5-5
Tunable Expressions Supported . . . . . . . . . . . . 5-5
S-Function Target Enhancements . . . . . . . . . . . 5-6
External Mode Enhancements . . . . . . . . . . . . . 5-6
Build Directory . . . . . . . . . . . . . . . . . . . 5-7
Code Optimization Features . . . . . . . . . . . . . 5-7
Subsystem Based Code Generation . . . . . . . . . . . 5-8
Nonvirtual Subsystem Code Generation . . . . . . . . . 5-8
Filename Extensions for Generated Files . . . . . . . . 5-8
hilite_system and Code Tracing . . . . . . . . . . . . 5-9
Generation of Parameter Comments . . . . . . . . . . 5-9
Borland 5.4 Compiler Support . . . . . . . . . . . . . 5-9
Enhanced Makefile Include Path Rules . . . . . . . . . 5-9
Target Language Compiler 4.0 . . . . . . . . . . . . . 5-10

**Platform Limitations for HP and IBM** . . . . . . . . 5-13

**Upgrading from an Earlier Release** . . . . . . . . . 5-14
Column-Major Matrix Ordering . . . . . . . . . . . . . 5-14
Including Generated Files . . . . . . . . . . . . . . 5-14
Updating Release 11 Custom Targets . . . . . . . . . . 5-14
hilite_system Replaces locate_system . . . . . . . . . 5-15
TLC Compatibility Issues . . . . . . . . . . . . . . . 5-15

**Known Software and Documentation Problems** . . . . 5-18
File Name Option in Nonvirtual
Subsystem Code Generation . . . . . . . . . . . . . 5-18
TLC Debugger Is Disabled . . . . . . . . . . . . . . . 5-18
Related Products Table Correction . . . . . . . . . . 5-18

## Release Summary

Release 4.0 of the Real-Time Workshop is a major upgrade, incorporating significant new and enhanced features and many improvements in the quality of generated code. These include:

- Significantly faster Target Language Compiler (TLC) code generation process
- TLC Profiler report for debugging TLC programs
- New efficiencies in generated code include improved signal storage reuse, constant block elimination, and parameter pooling.
- New Real-Time Workshop Embedded Coder add-on product replaces and significantly enhances the Embedded Real-Time (ERT) target.
- User interface improvements, including a redesigned Real-Time Workshop page and Model Parameter Configuration (tunable parameters) dialog
- Support for additional Simulink blocks, including Look-Up table blocks with very efficient generated code
- S-Function Target support for variable step solvers and parameter tuning
- Support for matrix operations for most Simulink blocks
- Support for frame-based processing for DSP blocks
- External mode support for many additional block types for signal uploading
- Automatic generation of S-function wrappers for embedded code, allowing for validation of generated code in Simulink
- Support for generation of code and executables from subsystems
- Support for Simulink data objects in code generation
- Support for generation of ASAP2 data definition files

## New Features

This section introduces the new features and enhancements added in the Real-Time Workshop 4.0 since the Real-Time Workshop 3.0.1.

For information about the Real-Time Workshop new features that are incorporated from prior releases, see:

- *Release 11.1 New Features*
  (enhancements introduced between Release 11.0 and Release 11.1)
- *Release 11 New Features*
  (enhancements introduced between Release 9.0 and Release 11.0)

---

**Note** For information about closely related products that extend the Real-Time Workshop, see the sections about the Real-Time Workshop Ada Coder, Real-Time Workshop Embedded Coder 1.0, Real-Time Windows Target, and xPC Target.

---

### Real-Time Workshop Embedded Coder

The Real-Time Workshop Embedded Coder is a new add-on product that replaces and enhances the Embedded Real-Time (ERT) target.

The Real-Time Workshop Embedded Coder is 100% compatible with the ERT target. In addition to supporting all previous functions of the ERT target, the Real-Time Workshop Embedded Coder includes many enhancements. For a summary of these enhancements, see the "Real-Time Workshop Embedded Coder 1.0" section in the Release Notes.

See "The Real-Time Workshop Embedded Coder" in the *Real-Time Workshop User's Guide* for full documentation of the Real-Time Workshop Embedded Coder.

### Simulink Data Object Support

The Real-Time Workshop supports the new Simulink data objects feature. Simulink provides the built-in `Simulink.Parameter` and `Simulink.Signal` classes for use with the Real-Time Workshop. Using these classes, you can create parameter and signal objects and assign storage classes and storage type qualifiers to the objects. These properties control how the generated code represents signals and parameters. The `Simulink.Parameter` and `Simulink.Signal` classes can be extended to include user-defined properties.

See “Simulink Data Objects and Code Generation” in the *Real-Time Workshop User's Guide* for complete details.

## ASAP2 Support

ASAP2 is a data definition standard proposed by the Association for Standardization of Automation and Measuring Systems (ASAM). This standard is used for data measurement, calibration, and diagnostic systems.

The Real-Time Workshop now lets you export an ASAP2 file containing information about your model during the code generation process. See “Generating ASAP2 Files” in the Real-Time Workshop online documentation.

## Enhanced Real-Time Workshop Page

The Real-Time Workshop page of the **Simulation Parameters** dialog box has been reorganized and made easier to use. See “Overview of the Real-Time Workshop User Interface” in the *Real-Time Workshop User's Guide* for complete details.

## Other User Interface Enhancements

The **Tools** menu of the Simulink window now contains a **Real-Time Workshop** submenu with shortcuts to frequently used features. See “Real-Time Workshop Submenu” in the *Real-Time Workshop User's Guide* for details.

You can now select a target configuration from the System Target File Browser by double-clicking on the desired entry in the target list. The previous selection method — selecting an entry and clicking the **OK** button — is still supported.

## Advanced Options Page

An Advanced options page has been added to the **Simulation Parameters** dialog box. The Advanced page contains new code generation options, as well as options formerly located in the Diagnostics and Real-Time Workshop pages. See “Advanced Options Page” for details.

## TLC Debugging Page

A TLC debugging options category has been added to the Real-Time Workshop page. The TLC debugging options are of interest to those who are developing TLC programs. See “TLC Debugging Page” for details.

---

**Note** In Release 12, the TLC debugger is still under test and is disabled. The TLC Debugging Page has been left in place for future release.

---

## Model Parameter Configuration Dialog

The **Model Parameter Configuration** dialog box replaces the **Tunable Parameters** dialog box. The **Model Parameter Configuration** dialog box enables you to declare individual parameters to be tunable and to control the generated storage declarations for each parameter. See “Parameters: Storage, Interfacing and Tuning” in the *Real-Time Workshop User’s Guide* for details.

---

**Note** On certain platforms, the release 11 **Tunable Parameters** dialog box is displayed instead of the **Model Parameter Configuration** dialog box. See “Platform Limitations for HP and IBM” for information.

---

## Tunable Expressions Supported

A tunable expression is an expression that contains one or more tunable parameters. Tunable expressions are now supported during simulation and in generated code.

Tunable expressions are allowed in masked subsystems. You can use tunable parameter names or tunable expressions in a masked subsystem dialog. When referenced in lower-level subsystems, such parameters remain tunable.

See “Tunable Expressions” in the *Real-Time Workshop User’s Guide* for a detailed description of the use of tunable parameters in expressions.

## S-Function Target Enhancements

S-function target enhancements include:

- The S-function target now supports variable-step solvers.
- The S-function target now supports tunable parameters.
- The new **Generate S-function** feature lets you automatically generate an S-function from a subsystem.

The S-function target is now documented the *Real-Time Workshop User's Guide*. See the chapter "The S-Function Target" for a full description of S-function target features.

## External Mode Enhancements

Several new features have been added to external mode:

- The default operation of the **External Signal & Triggering** dialog box has been changed to make monitoring the target program simpler. See "External Signal & Triggering Dialog Box" in the *Real-Time Workshop User's Guide* for details.

- Signal Viewing Subsystems have been implemented to let you encapsulate processing and viewing of signals received from the target system. Signal Viewing Subsystems run only on the host, generating no code in the target system. This is useful in situations where you want to process or condition signals before viewing or logging them, but you do not want to perform these tasks on the target system. See "Signal Viewing Subsystems" in the *Real-Time Workshop User's Guide* for details.
- Previously, only Scope blocks could be used in external mode to receive and view signals uploaded from the target program. The following now support external mode:
  - Dials & Gauges Blockset
  - Display blocks
  - To Workspace blocks
  - Signal Viewing Subsystems
  - S-functions

  See "External Mode Compatible Blocks and Subsystems" in the *Real-Time Workshop User's Guide* for details.

- In Release 12, we have documented the external mode communications application program interface (API). If you want to implement external mode communications via your own low-level protocol, see "Creating an External Mode Communications Channel" in the *Real-Time Workshop User's Guide*.

## Build Directory

The Real-Time Workshop now creates a *build directory* within your working directory. The build directory stores generated source code and other files created during the build process. The build directory name, *model*_*target*_`rtw`, derives from the name of the source model and the chosen target.

See "Directories Used in the Build Process" in the *Real-Time Workshop User's Guide* for details.

---

**Note** If you have created custom targets for the Real-Time Workshop under Release 11, you must update your custom system target files and template makefiles to create and utilize the build directory. See "Updating Release 11 Custom Targets" on page 5-14.

---

## Code Optimization Features

This section describes new or modified code generation options that are designed to help you optimize your generated code. The options described are located on the Advanced page of the **Simulation Parameters** dialog box. See "Advanced Options Page" for full details.

- **Block reduction**: When the **Block reduction** option is selected, Simulink collapses certain groups of blocks into a single, more efficient block, or removes them entirely. This results in faster model execution during simulation and in generated code.
- **Parameter pooling**: When multiple block parameters refer to storage locations that are separately defined but structurally identical, you can use this option to save memory.
- **Signal storage reuse**: This option replaces the (Enable/Disable) **Optimized block I/O storage** option of previous releases. **Signal storage reuse** is functionally identical to the older feature. Turning **Signal storage reuse** on is equivalent to enabling **Optimized block I/O storage**.

See the chapter “Optimizing the Model for Code Generation” in the *Real-Time Workshop User’s Guide* for further information on code optimization.

## Subsystem Based Code Generation

The Real-Time Workshop now generates code and builds an executable from any subsystem within a model. The build process uses the code generation and build parameters of the root model. See “Generating Code and Executables from Subsystems” in the *Real-Time Workshop User’s Guide* for details.

## Nonvirtual Subsystem Code Generation

The Real-Time Workshop now lets you generate code modules at the subsystem level. This feature applies only to nonvirtual subsystems. With nonvirtual subsystem code generation, you control how many files are generated, as well as the file and function names. To set options for nonvirtual subsystem code generation, you use the subsystem’s **Block Parameters** dialog.

Nonvirtual subsystem code generation is a more general and flexible method of controlling the number and size of generated files than the **Function management** code generation options (**File splitting** and **Function splitting**) used in previous releases. The **Function management** code generation options have been replaced by nonvirtual subsystem code generation.

See “Nonvirtual Subsystem Code Generation” in the *Real-Time Workshop User’s Guide* for details.

---

**Note** The operation of the `Auto` option of the **RTW file name options** menu in the **Block Parameters** dialog has changed since the printed version of the *Real-Time Workshop User’s Guide* went to press See “Filename Option in Nonvirtual Subsystem Code Generation” on page 5-18.

---

## Filename Extensions for Generated Files

In previous releases, some generated files were given special filename extensions, such as `.prm` or `.reg`. All the Real-Time Workshop generated code and header files now use standard filename extensions (`.c` and `.h`). The file naming conventions for the following generated files have changed:

- Model registration file (formerly `model.reg`) is now named `model_reg.h`.
- Model parameter file (formerly `model.prm`) is now named `model_prm.h`.
- `BlockIOSignals` struct file (formerly `model.bio`) is now named `model_bio.c`.
- `ParameterTuning` file (formerly `model.pt`) is now named `model_pt.c`.
- External mode data type transition file (formerly `model.dt`) is now named `model_dt.c`.

## hilite_system and Code Tracing

The Real-Time Workshop writes system/block identification tags in the generated code. The tags are designed to help you identify the block, in your source model, that generated a given line of code. In previous releases, the `locate_system` command was used to trace a tag back to the generating block.

The new `hilite_system` command replaces `locate_system`, for the purposes of tracing the Real-Time Workshop identification tags. You should use the `hilite_system` command to trace a tag back to the generating block. For further information on identification tags and code tracing, see “Tracing Generated Code Back to Your Simulink Model.”

## Generation of Parameter Comments

The **Force generation of parameter comments** option in the **General code generation options** category of the Real-Time Workshop page controls the generation of comments in the model parameter structure (`rtP`) declaration in `model_prm.h`. This lets you reduce the size of the generated file for models with a large number of parameters. See “Force Generation of Parameter Comments Option” in the *Real-Time Workshop User's Guide* for details.

## Borland 5.4 Compiler Support

The Real-Time Workshop now supports Version 5.4 of the Borland C/C++ compiler.

## Enhanced Makefile Include Path Rules

Two new rules and macros have been added to Real-Time Workshop template makefiles. These rules let you add source and include directories to makefiles generated by Real-Time Workshop without having to modify the template makefiles themselves. This feature is useful if you need to include your code when building S-functions.

See “Customizing the Makefile Include Path” in the Real-Time Workshop online documentation for details.

## Target Language Compiler 4.0

### TLC File Parsing Before Execution

The Target Language Compiler 4.0 completes parsing of the TLC file just before execution. This aids development because syntax errors are caught the first time the TLC file is run instead of the first time the offending line is reached.

### Enhanced Speed

The Target Language Compiler 4.0 features speed improvements throughout the software. In particular, the speed of block parameter generation has been enhanced.

### Build Directory

The Target Language Compiler 4.0 creates and uses a `build` directory. The `build` directory is in the current directory and prevents generated code from clashing with other files generated for other targets, and keeps your model directories maintenance to a minimum.

### TLC Profiler

An entirely new TLC Profiler has been added to the Target Language Compiler to help you find performance problems in your TLC code.

### model.rtw Changes

This release contains a new format and changes to the *model*`.rtw` file. The size of the *model*`.rtw` file has been reduced.

### Block Parameter Aliases

Aliases have been added for block parameters in the *model*`.rtw` file.

### Improved Text Expansion

This release of the Target Language Compiler contains new, flexible methods for text expansion from within strings.

### Column-Major Ordering

Two-dimensional signal and parameter data now use column-major ordering.

### Improved Record Handling

The Target Language Compiler 4.0 utilizes new record data handling.

### New TLC Language Semantics

Many changes have been made to the language including:

- Improved `EXISTS` behavior (see "TLC Compatibility Issues" on page 5-15)
- New TLC primitives for record handling
- Functions can return records.
- Records can be printed.
- Records can be empty.
- Record aliases are available.
- Built-in functions cannot be undefined via `%undef`.
- Short circuit evaluation for Boolean operators, `%if-%elseif-%endif`, and `?:` expressions are handled properly
- Conversions of values to and from MATLAB. (See "FEVAL Function" in the Target Language Compiler documentation for more information.)
- Relational operators can be used with nonfinite values.
- Loop control variables are local to loop bodies.

### New Built-In Functions

The following built-in functions have been added to the language.

`FIELDNAMES`, `GENERATE_FORMATTED_VALUE`, `GETFIELD`, `ISALIAS`, `ISEMPTY`, `ISEQUAL`, `ISFIELD`, `REMOVEFIELD`, `SETFIELD`

### New Built-In Values

The following built-in values have been added to the language.

`INTMAX`, `INTMIN`, `TLC_FALSE`, `TLC_TRUE`, `UINTMAX`

### Added Support for Inlined Code

Support has been added for two-dimensional signals in inlined code.

## Platform Limitations for HP and IBM

On the HP and IBM platforms, the Real-Time Workshop opens the Release 11 **Tunable Parameters** dialog box in place of the **Model Parameter Configuration** dialog box. Although they differ in appearance, both dialogs present the same information and support the same functionality.

For background information, see “Platform Limitations” on page 1-17 in the *Release Notes*.

## Upgrading from an Earlier Release

This section describes the upgrade issues involved in moving from the Real-Time Workshop 3.0 (Release 11.0) to the Real-Time Workshop 4.0.

For information about upgrading from a release earlier than 3.0, see *Release 11 New Features*

### Column-Major Matrix Ordering

The Real-Time Workshop now uses column-major ordering for two-dimensional signal and parameter data. In previous releases, the ordering was row-major.

If your hand-written code interfaces to such signals or parameters via `ExportedGlobal`, `ImportedExtern`, or `ImportedExternPointer` declarations, make sure to review any code that relies on row-major ordering, and make appropriate revisions.

### Including Generated Files

Filename extensions for certain generated files have changed. If your application code uses `#include` statements to include the Real-Time Workshop generated files (such as `model.prm`), you may need to modify these statements. See “Filename Extensions for Generated Files” on page 5-8.

### Updating Release 11 Custom Targets

If you have created custom targets for the Real-Time Workshop under Release 11, you must update your custom system target files and template makefiles to create and utilize the build directory. See `matlabroot/rtw/c/grt` for examples.

To update a Release 11 target:

**1** Add the following to your system target file.

```
/%
BEGIN_RTW_OPTIONS
...
rtwgensettings.BuildDirSuffix = '_grt_rtw';
END_RTW_OPTIONS
%/
```

**2** Add `".."` to the `INCLUDES` rule in your template makefile. The following example is from `grt_lcc.tmf`.

```
INCLUDES = -I. -I.. $(MATLAB_INCLUDES) $(USER_INCLUDES)
```

The first `-I.` gets files from the build directory, and the second `-I..` gets files (e.g., user written S-functions) from the current working directory.

Conceptually, think of the current directory and the build directory as the same (as it was in Release 11). The current working directory contains items like user written S-functions. The reason `".."` must be added to the `INCLUDES` rule is that `make` is invoked in the build directory (i.e., the current directory was temporarily moved).

**3** Place the generated executable in your current working directory. The following example is from `grt_lcc.tmf`.

```
PROGRAM = ../$(MODEL).exe
$(PROGRAM) : $(OBJS) $(RTWLIB)
   $(LD) $(LDFLAGS) -o $@ $(LINK_OBJS) $(RTWLIB) $(LIBS)
```

## hilite_system Replaces locate_system

If you use the `locate_system` command, in MATLAB programs for tracing the Real-Time Workshop system/block identification tags, you should use `hilite_system` instead. See “hilite_system and Code Tracing” on page 5-9.

## TLC Compatibility Issues

In bringing Target Language Compiler files from Release 11 to Release 12, the following changes may affect your TLC code base:

- Nested evaluations are no longer supported. Expressions such as

  ```
  %<LibBlockParameterValue(%<myVariable>,"", "", "")>
  ```

  are no longer supported. You will have to convert these expressions into equivalent non-nested expressions.

- Aliases are no longer automatically created for Parameter blocks while reading in the Real-Time Workshop files.

- You cannot change the contents of a “Default” record after it has been created. In the previous TLC, you could change a “Default” record and see the change in all the records that inherited from that default record.
- The `%codeblock` and `%endcodeblock` constructs are no longer supported.
- `%defines` and macro constructs are no longer supported.
- Use of line continuation characters (`...` and `\`) are not allowed inside of strings. Also, to place a double quote (`"`) character inside a string, you must use `\"`. Previously, the Target Language Compiler allowed you to use `"""` to get a double quote in a string.
- Semantics have been formalized to `%include` files in different contexts (e.g., from generate files, inside of `%with` blocks, etc.) `%include` statements are now treated as if they were read in from the global scope.
- The previous the Target Language Compiler had the ability to split function definitions (and other directives) across include file boundaries (e.g., you could start a `%function` in one file and `%include` a file that had the `%endfunction`). This no longer works.
- Nested functions are no longer allowed. For example,

```
%function foo ()
  %function bar ()
  %endfunction
%endfunction
```

- Built-in functions cannot be undefined via `%undef`. It is possible to undefine built in values, but this practice is not encouraged.
- Recursive records are no longer allowed. For example,

```
Record1    {
  Val      2
  Ref      Record2
}
Record2    {
  Val      3
  Ref      Record1
}
```

- Semantics of the `EXISTS` function have changed. In the previous release of TLC, `EXISTS(var)` would check if the variable represented by the string value in `var` existed. In the current release of TLC, `EXISTS(var)` checks to see if `var` exists or not.

To emulate the behavior of `EXISTS` in the previous release, replace

```
EXISTS(var)
```

with

```
EXISTS(“%<var>”)
```

## Known Software and Documentation Problems

This section updates the Real-Time Workshop documentation set, reflecting known software and documentation problems.

### Filename Option in Nonvirtual Subsystem Code Generation

To set options for nonvirtual subsystem code generation, you use the subsystem's **Block Parameters** dialog. The operation of the `Auto` option of the **RTW file name options** menu in the **Block Parameters** dialog has changed since the printed version of the *Real-Time Workshop User's Guide* went to press.

In the online version of the *Real-Time Workshop User's Guide*, we have corrected the description of this option. See the "Nonvirtual Subsystem Code Generation" section of the online guide.

We repeat the corrected description here:

`Auto`: The Real-Time Workshop does *not* generate a separate file for the subsystem. Code generated from the subsystem is generated within the code module generated from the subsystem's parent system. If the subsystem's parent is the model itself, code generated from the subsystem is generated within *model*`.c`.

### TLC Debugger Is Disabled

The TLC debugger is not available in Release 12. The MathWorks has been improving TLC code generation speed and enhancing TLC language semantics. The improvements were not compatible with the debugger included in a Release 11.1+ release or one of the Release 12 Betas. We are working on an debugger that will support the new language syntax. The revised TLC debugger is still under test and is disabled in Release 12. The TLC Debugging Page, and references to the TLC debugger, have been left in place for a future release.

### Related Products Table Correction

In the printed version of the *Real-Time Workshop User's Guide*, the Related Products table incorrectly listed the MATLAB Communications Toolbox instead of the Simulink Communications Blockset. This has been corrected in the online version of the *Real-Time Workshop User's Guide*.

# CDMA Reference Blockset 1.0.2 Release Notes

**Introduction to the CDMA Reference Blockset** . . . . 6-2

**Known Software and Documentation Problems** . . . . 6-3
Signal Support . . . . . . . . . . . . . . . . . . . . 6-3

## Introduction to the CDMA Reference Blockset

The CDMA Reference Blockset was introduced as a Web downloadable product after Release 11.1. It is now part of Release 12.

The CDMA Reference Blockset is a collection of Simulink blocks for creating and simulating the CDMA IS-95A standard for wireless communications. You can create and simulate an entire, end-to-end (transmitter-to-receiver) model of a wireless system.

## Known Software and Documentation Problems

This section updates the CDMA Reference Blockset 1.0.2 documentation set, reflecting known CDMA Reference Blockset 1.0.2 software problems.

### Signal Support

The CDMA Reference Blockset does not support frame-based signals or matrix signals. That is, it supports only sample-based one-dimensional signals. If your model combines blocks from the CDMA Reference Blockset with blocks from the Communications Blockset or the DSP Blockset, then you may need to change signal attributes in appropriate parts of your model. Useful blocks for changing signal attributes are:

- Convert 1-D to 2-D, in the DSP Blockset
- Frame Status Conversion, in the DSP Blockset
- Reshape, in Simulink

# Communications Blockset 2.0 Release Notes

**New Features** . . . . . . . . . . . . . . . . . . . . . 7-2
Digital Modulation Libraries . . . . . . . . . . . . . . . . . 7-2
Interleaving Libraries . . . . . . . . . . . . . . . . . . . . 7-2
Fading Channels . . . . . . . . . . . . . . . . . . . . . 7-3
Enhanced Support for Convolutional Coding . . . . . . . . 7-3
Sequence Operations . . . . . . . . . . . . . . . . . . . 7-3

**Major Bug Fixes** . . . . . . . . . . . . . . . . . . . . 7-4

**Upgrading from an Earlier Release** . . . . . . . . . . 7-5
New Block Libraries in Release 12 . . . . . . . . . . . . 7-5
New Signal Support . . . . . . . . . . . . . . . . . . . 7-7
Functionality Changes in Specific Blocks . . . . . . . . . 7-7
Block Name Changes Since Previous Manual . . . . . . . 7-8
Obsolete Blocks from Previous Manual . . . . . . . . . . 7-12

**Known Software and Documentation Problems** . . . . 7-19
Code Generation Limitations . . . . . . . . . . . . . . 7-19
Limited Frame and Matrix Support . . . . . . . . . . . 7-19
Errors in Printed Version of Documentation . . . . . . . . 7-20

## New Features

The Communications Blockset is a new product being introduced with Release 12. The Communications Blockset incorporates the functionality of the blocks that were included in the Communications Toolbox 1.4 (Release 11), with the addition of the new features summarized below.

---

**Note** The Communications Toolbox is described in a separate section.

---

### Digital Modulation Libraries

The digital modulation libraries have been replaced with new ones. The new libraries contain baseband and passband sublibraries for:

- Amplitude modulation (PAM, QAM)
- Phase modulation (PSK, DPSK)
- Frequency modulation (FSK)
- Continuous phase modulation (CPM), including MSK and GMSK

For a list of blocks, see the reference sections for the baseband and passband digital modulation libraries. For a discussion of the capabilities of the new libraries, see “Digital Modulation” in the *Communications Blockset User’s Guide*.

### Interleaving Libraries

A new Interleaving library contains sublibraries for block interleaving and convolutional interleaving. These sublibraries support general block interleavers and general multiplexed interleavers, as well as several special cases of these. For more information, see “Interleaving” in the *Communications Blockset User’s Guide*.

## Fading Channels

The new Multipath Rayleigh Fading Channel and Rician Fading Channel blocks implement baseband simulations of fading propagation channels. These blocks model real-world mobile communication effects and are useful for modeling mobile wireless communication systems. For more information, see “Fading Channels” in the *Communications Blockset User's Guide*.

## Enhanced Support for Convolutional Coding

The new APP Decoder block implements *a posteriori probability* decoding. The enhanced Convolutional Encoder and Viterbi Decoder blocks now support a more general class of convolutional codes by accepting a trellis parameter in their dialog boxes. The new `poly2trellis` function in the Communications Toolbox supports this enhancement, by converting a polynomial description of an encoder into a corresponding trellis description. For more information, see “Convolutional Coding” in the *Communications Blockset User's Guide*.

## Sequence Operations

These new blocks in the Sequence Operations library manipulate data sequences in various ways:

- Bit to Integer Converter and Integer to Bit Converter convert between integers and their binary representations.
- Complex Phase Shift and Complex Phase Difference manipulate or analyze the phase of a complex signal.
- Derepeat is an inverse of the DSP Blockset's Repeat block.
- Interlacer and Deinterlacer can be useful for combining or separating in-phase and quadrature components of a signal.
- Puncture and Insert Zero are useful for processing punctured codes.

## Major Bug Fixes

The Communications Blockset includes several bug fixes, including the following descriptions (online only) of particularly important bug fixes.

## Upgrading from an Earlier Release

This section describes the upgrade issues involved in moving from the Communications Toolbox 1.4 (Release 11) to the Communications Blockset 2.0. Such issues relate to:

- The new set of block libraries
- Support of new signal types, particularly frame-based signals and two-dimensional (that is, matrix) signals
- Functionality changes in specific blocks
- Blocks from the previous *Communications Toolbox User's Guide* that have changed their names in Version 2.0 of the Communications Blockset
- Blocks from the previous *Communications Toolbox User's Guide* that are not in Version 2.0 of the Communications Blockset

Note that before Release 12, the Communications Toolbox was a collection of functions and blocks, so that the previous *Communications Toolbox User's Guide* documented both functions and blocks. Release 12 is the first release of the Communications Blockset as a distinct product.

### New Block Libraries in Release 12

The Communications Blockset uses a new set of block libraries, although it also includes the previous set of block libraries for backwards compatibility. The new set of libraries is what appears in the Simulink Browser (on PC) and what opens if you type `commlib` at the MATLAB prompt. You should build new models using this new set. New blocks in the new set of libraries are described in "New Features" on page 7-2.

Your previous models link to the previous set of libraries unless you choose to replace individual blocks manually. You can access the previous set of libraries by typing `commlib 1.5` at the MATLAB prompt.

## Reorganization of Utility Functions in New Set of Libraries

The Utility Functions library has been reorganized. The table below lists blocks in Release 12 that were in the Release 11 Utility Functions library.

| Block | New Location |
|---|---|
| Data Mapper | Utility Functions |
| Derepeat | Sequence Operations sublibrary |
| Descrambler | Sequence Operations sublibrary |
| Differential Decoder | Source Coding |
| Differential Encoder | Source Coding |
| Discrete Modulo Integrator (formerly called Discrete Time Modulo Integrator) | Integrators sublibrary |
| Discrete-Time VCO | Comm Sources |
| Windowed Integrator | Integrators sublibrary |
| Modulo Integrator | Integrators sublibrary |
| Integrate and Dump (formerly called Scheduled Reset Integrator) | Integrators sublibrary |
| Scrambler | Sequence Operations sublibrary |
| Voltage-Controlled Oscillator | Comm Sources |

The Sequence Operations and Integrators sublibraries are in the Basic Comm Functions library.

Some blocks from the Release 11 Utility Functions library have become obsolete; for possible substitutions in Release 12, see “Obsolete Blocks from Previous Manual.”

## New Signal Support

As of Release 12, Simulink supports matrix signals in addition to one-dimensional arrays, and frame-based signals in addition to sample-based signals. The Communications Blockset processes certain kinds of matrix and frame-based signals.

Because a future release is planned to include more comprehensive matrix and frame support, some Release 12 blocks avoid conflict with future features by using strict guidelines to determine the kinds of signals that they now accept. As a consequence, if you used vector signals in a model before Release 12, then you might need to use a particular kind of vector signal in Release 12 (such as a frame-based column vector, a frame-based row vector, or a sample-based vector of a particular shape or dimension).

As another consequence of frame support, the AWGN Channel and Derepeat blocks no longer have the **Frame-based inputs** check box and the **Number of channels** parameter as in the Communications Toolbox 1.5. Instead, these blocks inherit the frame status and number of channels from their inputs.

The *Communications Blockset User's Guide* includes more detail about signal support. To learn the terminology associated with signal attributes, see the Technical Conventions section. For general information about signal support in the Communications Blockset, see the Signal Support section.

## Functionality Changes in Specific Blocks

Aside from signal support and naming issues, some blocks have changed the way that they behave:

- The Continuous-Time Eye and Scatter Diagrams and Discrete-Time Eye and Scatter Diagrams blocks process *complex* signals, whereas their counterparts before Release 12 (called Eye-Diagram Scatter Plot and Sample-Time Eye-Diagram Scatter) processed real vectors that listed in-phase and quadrature components separately.
- The blocks for Reed-Solomon and BCH coding no longer have a second input port for an enabler signal. The change affects the Binary-Input RS Encoder, Binary-Output RS Decoder, Integer-Input RS Encoder, Integer-Output RS Decoder, and BCH Decoder blocks.
- The Scrambler, Descrambler, and PN Sequence Generator blocks no longer have a trigger input. The Scrambler and Descrambler blocks no longer have

a state output. The PN Sequence Generator block produces output from the last register in the generator, not the first.

- The Convolutional Encoder and Viterbi Decoder blocks have new interfaces because they can now accept a more general trellis description of a convolutional encoder.
- The Version 1.4 Error Rate Calculation block considers a vector input to be a sample, whereas the current block considers a vector input to be a frame of multiple samples. For vector inputs of length n, a **Receive delay** parameter value of k in the Version 1.4 block is equivalent to a **Receive delay** of k*n in the current block.
- The Voltage-Controlled Oscillator block now uses the cosine, not sine, function to produce its waveform. This change affects the phase of the output signal.
- The blocks in the Synchronization library no longer use a **Gain at the output** parameter. The remaining parameters that define characteristics of the voltage-controlled oscillator have changed slightly. Also, the Baseband PLL and Linearized Baseband PLL blocks now include three output ports instead of one, to match the Phase-Locked Loop and Charge Pump PLL blocks.

## Block Name Changes Since Previous Manual

The table below lists the old and new names of blocks that were part of the Communications Toolbox before Release 12 and that have changed their names. The old names are from the last printed version of the *Communications Toolbox User's Guide*. Because the libraries have been reorganized since that document was printed, the third column of the table lists the current library name for each block.

**Table 1-1: Names of Blocks in Version 1.x and Version 2, Where Different**

| Old Block Name (Version 1.x) | New Block Name (Version 2) | Library Location |
|---|---|---|
| ADM with Carrier | DSB AM Demodulator Passband | Analog Passband |
| ADM with Carrier CE | DSB AM Demodulator Baseband | Analog Baseband |
| AM with Carrier | DSB AM Modulator Passband | Analog Passband |

**Table 1-1: Names of Blocks in Version 1.x and Version 2, Where Different (Continued)**

| Old Block Name (Version 1.x) | New Block Name (Version 2) | Library Location |
|---|---|---|
| AM with Carrier CE | DSB AM Modulator Baseband | Analog Baseband |
| BCH Decode Vector In/Out | BCH Decoder | Block Codes |
| BCH Encode Vector In/Out | BCH Encoder | Block Codes |
| Baseband Model PLL | Baseband PLL | Synchronization |
| Bernoulli Random Binary Noise Generator | Bernoulli Random Binary Generator | Comm Sources |
| Binary Error Channel | Binary Symmetric Channel | Channels |
| Cyclic Decode Vector In/Out | Binary Cyclic Decoder | Block Codes |
| Cyclic Encode Vector In/Out | Binary Cyclic Encoder | Block Codes |
| DPCM Decode | DPCM Decoder | Source Coding |
| DPCM Encode | DPCM Encoder | Source Coding |
| DSB-SC ADM | DSBSC AM Demodulator Passband | Analog Passband |
| DSB ADM CE | DSBSC AM Demodulator Baseband | Analog Baseband |
| DSB-SC AM | DSBSC AM Modulator Passband | Analog Passband |
| DSB AM CE | DSBSC AM Modulator Baseband | Analog Baseband |
| Discrete Time VCO | Discrete-Time VCO | Comm Sources |
| Discrete Time Modulo Integrator | Discrete Modulo Integrator | Integrators |
| Eye-Pattern & Scatter Plot | Continuous-Time Eye and Scatter Diagrams | Comm Sinks |
| FDM | FM Demodulator Passband | Analog Passband |

**Table 1-1: Names of Blocks in Version 1.x and Version 2, Where Different (Continued)**

| Old Block Name (Version 1.x) | New Block Name (Version 2) | Library Location |
|---|---|---|
| FDM CE | FM Demodulator Baseband | Analog Baseband |
| FM | FM Modulator Passband | Analog Passband |
| FM CE | FM Modulator Baseband | Analog Baseband |
| Gaussian Random Noise Generator | Gaussian Noise Generator | Comm Sources |
| Hamming Decode Vector In/Out | Hamming Decoder | Block Codes |
| Hamming Encode Vector In/Out | Hamming Encoder | Block Codes |
| Linear Block Decode Vector In/Out | Binary Linear Decoder | Block Codes |
| Linear Block Encode Vector In/Out | Binary Linear Encoder | Block Codes |
| Linearized Baseband Model PLL | Linearized Baseband PLL | Synchronization |
| μ-Law Compressor | Mu-Law Compressor | Source Coding |
| μ-Law Expander | Mu-Law Expander | Source Coding |
| PDM | PM Demodulator Passband | Analog Passband |
| PDM CE | PM Demodulator Baseband | Analog Baseband |
| PLL | Phase-Locked Loop | Synchronization |
| PM | PM Modulator Passband | Analog Passband |
| PM CE | PM Modulator Baseband | Analog Baseband |
| Poisson Random Integer Generator | Poisson Int Generator | Comm Sources |
| Quantization Decode | Quantizer Decode | Source Coding |

**Table 1-1: Names of Blocks in Version 1.x and Version 2, Where Different (Continued)**

| Old Block Name (Version 1.x) | New Block Name (Version 2) | Library Location |
|---|---|---|
| Reed-Solomon Decode Binary Vector In/Out | Binary-Output RS Decoder | Block Codes |
| Reed-Solomon Decode Integer Vector In/Out | Integer-Output RS Decoder | Block Codes |
| Reed-Solomon Encode Binary Vector In/Out | Binary-Input RS Encoder | Block Codes |
| Reed-Solomon Encode Integer Vector In/Out | Integer-Input RS Encoder | Block Codes |
| Rician Random Noise Generator | Rician Noise Generator | Comm Sources |
| SSB ADM | SSB AM Demodulator Passband | Analog Passband |
| SSB ADM CE | SSB AM Demodulator Baseband | Analog Baseband |
| SSB-AM | SSB AM Modulator Passband | Analog Passband |
| SSB-AM CE | SSB AM Modulator Baseband | Analog Baseband |
| Sample Time Eye-Pattern Diagram & Scatter Plot | Discrete-Time Eye and Scatter Diagrams | Comm Sinks |
| Scheduled Reset Integrator | Integrate and Dump | Integrators |
| Signal Quantizer | Sampled Quantizer Encode | Source Coding |
| Triggered Signal Quantizer | Enabled Quantizer Encode | Source Coding |
| Uniform Random Noise Generator | Uniform Noise Generator | Comm Sources |
| Uniform Random Integer Generator | Random-Integer Generator | Comm Sources |
| VCO | Voltage-Controlled Oscillator | Comm Sources |

## Obsolete Blocks from Previous Manual

The table below lists blocks that appear in the previous version of the *Communications Toolbox User's Guide* but that are not included in the Release 12 Communications Blockset. Where applicable, the second column lists blocks that provide similar functionality. In some cases, the similar block requires different parameter settings, data formats, or signal attributes compared to the original block. Therefore, you should read the documentation for the similar block before using it in your model.

**Table 7-1: Blocks Not in v2, and Similar v2 Blocks**

| Obsolete Block | Similar Block(s), if Any |
|---|---|
| Array Function | See Math library in Simulink |
| BCH Code View Table | Use `bchpoly` in Communications Toolbox |
| BCH Decode Sequence In/Out | BCH Decoder. See “Using Serial Signals” in the *Communications Blockset User's Guide* |
| BCH Encode Sequence In/Out | BCH Encoder. See “Using Serial Signals” in the *Communications Blockset User's Guide* |
| Coherent MFSK Corr Demod | |
| Coherent MFSK Demod | |
| Coherent MFSK Demod CE | |
| Complex Filter | See Filtering library in DSP Blockset |
| Convolutional Decode Sequence In/Out | Viterbi Decoder. See “Using Serial Signals” in the *Communications Blockset User's Guide* |
| Convolutional Decode Vector In/Out | Viterbi Decoder |

**Table 7-1: Blocks Not in v2, and Similar v2 Blocks (Continued)**

| Obsolete Block | Similar Block(s), if Any |
|---|---|
| Convolutional Encode Sequence In/Out | Convolutional Encoder. See "Using Serial Signals" in the *Communications Blockset User's Guide* |
| Convolutional Encode Vector In/Out | Convolutional Encoder |
| Cyclic Decode Sequence In/Out | Binary Cyclic Decoder. See "Using Serial Signals" in the *Communications Blockset User's Guide* |
| Cyclic Encode Sequence In/Out | Binary Cyclic Encoder. See "Using Serial Signals" in the *Communications Blockset User's Guide* |
| DPSK Demod | M-DPSK Demodulator Passband |
| DPSK Mod | M-DPSK Modulator Passband |
| D-TDMA Demux | |
| D-TDMA Mux | |
| Edge Detector | Edge Detector in DSP Blockset |
| Envelope Detector | Maximum, Minimum in DSP Blockset |
| Error Counter | Counter, in DSP Blockset |
| Error Rate Meter | Error Rate Calculation |
| Hamming Decode Sequence In/Out | Hamming Decoder. See "Using Serial Signals" in the *Communications Blockset User's Guide* |

**Table 7-1: Blocks Not in v2, and Similar v2 Blocks (Continued)**

| Obsolete Block | Similar Block(s), if Any |
|---|---|
| Hamming Encode Sequence In/Out | Hamming Encoder. See “Using Serial Signals” in the *Communications Blockset User's Guide* |
| Hilbert Filter | Remez FIR Filter Design in DSP Blockset |
| Integer Scalar to Vector | Integer to Bit Converter |
| Integer Vector to Scalar | Bit to Integer Converter |
| Interleave | Matrix Interleaver |
| K-Step Delay | Integer Delay in DSP Blockset |
| Limited Binary Error Channel | Binary Vector Noise Generator |
| Linear Block Decode Sequence In/Out | Binary Linear Decoder. See “Using Serial Signals” in the *Communications Blockset User's Guide* |
| Linear Block Encode Sequence In/Out | Binary Linear Encoder. See “Using Serial Signals” in the *Communications Blockset User's Guide* |
| MASK Demap | |
| MASK Demod | M-PAM Demodulator Passband |
| MASK Demod CE | M-PAM Demodulator Baseband |
| MASK Map | |
| MASK Mod | M-PAM Modulator Passband |
| MASK Mod CE | M-PAM Modulator Baseband |
| Mean and Variance | Mean, Variance in DSP Blockset |

**Table 7-1: Blocks Not in v2, and Similar v2 Blocks (Continued)**

| Obsolete Block | Similar Block(s), if Any |
|---|---|
| Mean and Std | Mean, Standard Deviation in DSP Blockset |
| MFSK Map | |
| MFSK Mod | M-FSK Modulator Passband |
| MFSK Mod CE | M-FSK Modulator Baseband |
| Min/Max Demap | |
| Min/Max Index | Maximum, Minimum in DSP Blockset |
| Modulo | Math Function in Simulink |
| MPSK Correlation Demodulation | |
| MPSK Demod | M-PSK Demodulator Passband |
| MPSK Demod CE | M-PSK Demodulator Baseband |
| MPSK Map | |
| MPSK Mod | M-PSK Modulator Passband |
| MPSK Mod CE | M-PSK Modulator Baseband |
| MSK Demod | MSK Demodulator Passband |
| MSK Mod | MSK Modulator Passband |
| Noncoherent MFSK Corr Demod | |
| Noncoherent MFSK Demod | M-FSK Demodulator Passband |
| Noncoherent MFSK Demod CE | M-FSK Demodulator Baseband |
| Number Counter | Counter, in DSP Blockset |
| OQPSK Demod | OQPSK Demodulator Passband |
| OQPSK Mod | OQPSK Modulator Passband |

Table 7-1: Blocks Not in v2, and Similar v2 Blocks (Continued)

| Obsolete Block | Similar Block(s), if Any |
|---|---|
| QADM | General QAM Demodulator Passband |
| QADM CE | General QAM Demodulator Baseband |
| QAM | General QAM Modulator Passband |
| QAM CE | General QAM Modulator Baseband |
| QASK Demap Arbitrary Constellation | |
| QASK Demap Circle Constellation | |
| QASK Demap Square Constellation | |
| QASK Demod Arbitrary Constellation | General QAM Demodulator Passband |
| QASK Demod CE Arbitrary Constellation | General QAM Demodulator Baseband |
| QASK Demod CE Circle Constellation | General QAM Demodulator Baseband |
| QASK Demod CE Square Constellation | Rectangular QAM Demodulator Baseband |
| QASK Demod Circle Constellation | General QAM Demodulator Passband |
| QASK Demod Square Constellation | Rectangular QAM Demodulator Passband |
| QASK Map Arbitrary Constellation | |
| QASK Map Square Constellation | |

**Table 7-1: Blocks Not in v2, and Similar v2 Blocks (Continued)**

| Obsolete Block | Similar Block(s), if Any |
|---|---|
| QASK Mod Arbitrary Constellation | General QAM Modulator Passband |
| QASK Mod CE Arbitrary Constellation | General QAM Modulator Baseband |
| QASK Mod CE Circle Constellation | General QAM Modulator Baseband |
| QASK Mod CE Square Constellation | Rectangular QAM Modulator Baseband |
| QASK Mod Circle Constellation | General QAM Modulator Passband |
| QASK Mod Square Constellation | Rectangular QAM Modulator Passband |
| Raised Cosine Filter | |
| Rayleigh Fading CE Channel | Multipath Rayleigh Fading Channel |
| Rayleigh Noise CE Channel | Rayleigh Noise Generator |
| Reed-Solomon Decode Binary Sequence In/Out | Binary-Output RS Decoder. See “Using Serial Signals” in the *Communications Blockset User’s Guide* |
| Reed-Solomon Decode Integer Sequence In/Out | Integer-Output RS Decoder. See “Using Serial Signals” in the *Communications Blockset User’s Guide* |
| Reed-Solomon Encode Binary Sequence In/Out | Binary-Input RS Encoder. See “Using Serial Signals” in the *Communications Blockset User’s Guide* |

**Table 7-1: Blocks Not in v2, and Similar v2 Blocks (Continued)**

| Obsolete Block | Similar Block(s), if Any |
|---|---|
| Reed-Solomon Encode Integer Sequence In/Out | Integer-Input RS Encoder. See “Using Serial Signals” in the *Communications Blockset User’s Guide* |
| Register Shift | Queue in DSP Blockset |
| Rician Noise CE Channel | Rician Noise Generator |
| Sampled Read From Workspace | Signal From Workspace in DSP Blockset |
| Sinc | |
| Time-Share Demux | |
| Time-Share Mux | |
| Triggered Read from Workspace | Triggered Signal From Workspace in DSP Blockset |
| Triggered Write to Workspace | Triggered To Workspace in DSP Blockset |
| Varying AWGN Channel | |
| Varying Rayleigh Fading CE Channel | |
| Varying Rayleigh Noise CE Channel | |
| Varying Rician Noise CE Channel | |
| Vector Pulse | Discrete Pulse Generator in Simulink |
| Vector Redistributor | |

## Known Software and Documentation Problems

This section updates the Communications Blockset 2.0 documentation set, reflecting known Communications Blockset 2.0 software and documentation problems.

### Code Generation Limitations

Several blocks are incompatible with Real-Time Workshop. As a result, Real-Time Workshop cannot generate code for models that include these blocks:

- Discrete-Time Eye and Scatter Diagrams block
- Continuous-Time Eye and Scatter Diagrams block
- Voltage-Controlled Oscillator block
- Multipath Rayleigh Fading Channel block
- Rician Fading Channel block
- Blocks in the Analog Passband and Digital Passband sublibraries of the Modulation library
- Blocks in the CPM sublibrary of the Digital Baseband sublibrary of the Modulation library

Furthermore, blocks in the Analog Passband sublibrary of the Modulation library are not compatible with the Simulink Accelerator.

### Limited Frame and Matrix Support

The Communications Blockset provides limited support for matrix and frame-based signals. In a future release, more blocks will support multichannel behavior, and more blocks will be optimized for faster frame-based processing. Release 12 strives to be forward-compatible in the sense that future signal support modes should not invalidate current modes and should minimize the difficulty of upgrading from Release 12 to a future release.

As a consequence of this forward-looking view, some blocks now use strict guidelines to determine the kinds of signals that they accept. One consideration is that if a block will ultimately support frame-based multichannel signals, then a sample-based vector input might potentially represent either a frame of data from a single channel or a set of samples from multiple channels. Therefore, even if such a block does not currently provide such comprehensive signal support, it accepts only frame-based vectors, whose interpretation is unambiguous.

## Errors in Printed Version of Documentation

The printed version of the *Communications Blockset User's Guide* contains these errors:

- In the example, "Passband Digital Modulation," the instructions on page 2-82 for configuring the Mean block should say, "Set **Reset port** to **None**" instead of "Uncheck the **Reset port** check box." The electronic version of the instructions are correct.
- Reference pages for some blocks in the CPM modulation libraries omit the **Symbol prehistory** and/or **Phase offset** parameters. The electronic versions of the reference pages are correct. The affected blocks are:
    - CPM Demodulator Baseband
    - GMSK Demodulator Baseband
    - MSK Demodulator Baseband
    - CPFSK Demodulator Baseband
    - CPM Demodulator Passband
    - GMSK Demodulator Passband

# Communications Toolbox 2.0 Release Notes

**New Features** . . . . . . . . . . . . . . . . . . . . . . 8-2
Convolutional Coding Functions . . . . . . . . . . . . . . 8-2
Gaussian Noise Functions . . . . . . . . . . . . . . . . . 8-2
Other New Functions . . . . . . . . . . . . . . . . . . . . 8-3
Enhancements to Existing Functions . . . . . . . . . . . 8-3

**Major Bug Fixes** . . . . . . . . . . . . . . . . . . . . 8-4

**Upgrading from an Earlier Release** . . . . . . . . . . 8-5
Changes in Functionality . . . . . . . . . . . . . . . . . 8-5
Obsolete Functions . . . . . . . . . . . . . . . . . . . . 8-6

**Known Software and Documentation Problems** . . . . 8-8
Errors in Printed Version of Documentation . . . . . . . . 8-8

## New Features

The Communications Toolbox 2.0 and the Communications Blockset 2.0 are now separate products (that is, the Communications Toolbox no longer includes blocks).

This section introduces the new features and enhancements added in the Communications Toolbox 2.0 since the Communications Toolbox 1.4.

---

**Note** The Communications Blockset is described in a separate section.

---

### Convolutional Coding Functions

The Communications Toolbox processes feedforward and feedback convolutional codes that can be described by a trellis structure or a set of generator polynomials. It uses the Viterbi algorithm to implement hard-decision and soft-decision decoding. These new functions support convolutional coding:

- `convenc` creates a convolutional code from binary data.
- `vitdec` decodes convolutionally encoded data using the Viterbi algorithm.
- `poly2trellis` converts a polynomial description of a convolutional encoder to a trellis description.
- `istrellis` checks if the input is a valid trellis structure representing a convolutional encoder.

For more information about using these functions, see “Convolutional Coding” in the *Communications Toolbox User's Guide*.

### Gaussian Noise Functions

These new functions create Gaussian noise:

- `awgn` adds white Gaussian noise to the input signal to produce a specified signal-to-noise ratio.
- `wgn` generates white Gaussian noise with a specified power, impedance, and complexity.

## Other New Functions

These functions are also new in Release 12:

- `eyediagram` plots an eye diagram.
- `marcumq` implements the generalized Marcum Q function.
- `oct2dec` converts octal numbers to decimal numbers.
- `randerr` generates bit error patterns. This is similar to the obsolete function `randbit`, but it accepts a more intuitive set of input arguments and uses an upgraded random number generator.
- `randsrc` generates random matrices using a prescribed alphabet.
- `scatterplot` produces a scatter plot.
- `syndtable` generates syndrome decoding tables. This is similar to the obsolete function `htruthtb`, but it is not limited to single-error-correction codes.

## Enhancements to Existing Functions

The following functions have been enhanced in Release 12:

- `biterr` and `symerr` provide a third output argument that indicates the results of individual comparisons. These functions also provide more comprehensive support for comparisons between a vector and a matrix.
- `de2bi` and `bi2de` use an optional input flag to indicate the ordering of bits. If you omit the flag from the list of input arguments, then the default behavior matches that of Release 11.
- `randint` can operate without input arguments. Also, it can accept a negative value for the optional third input argument.

## Major Bug Fixes

The Communications Toolbox includes several bug fixes, including the following descriptions (online only) of particularly important bug fixes.

## Upgrading from an Earlier Release

This section describes the upgrade issues involved in moving from the Communications Toolbox 1.4 (Release 11) to the Communications Toolbox 2.0.

### Changes in Functionality

The table below lists functions whose behavior has changed.

| Function | Change in Functionality |
|---|---|
| `bi2de` | Distinguishes between rows and columns as input vectors. Treats column vector as separate numbers, not as digits of a single number. To adapt your existing code, transpose the input vector if necessary. |
| `biterr` | Input argument `k` must be large enough to represent all elements of the input arguments `x` and `y`. |
| `biterr`, `symerr` | Distinguish between rows and columns as input vectors. To adapt your existing code, transpose the input vector if necessary. |
| | Use different strings for the input argument that controls row-wise and column-wise comparisons. |
| | Produce vector, not scalar, output if one input is a vector. See these functions' reference pages for more information. |
| `de2bi` | Second input argument, if it appears, must not be smaller than the number of bits in any element of the first input argument. Previously, the function produced a truncated binary representation instead of an error. To adapt your existing code, specify a sufficiently large number for the second input argument and then truncate the answer manually. |
| `ddemod` | Default behavior uses no filter, not a Butterworth filter. Regardless of filtering, the function uses an integrator to perform demodulation. |

| Function | Change in Functionality |
|---|---|
| dmod, ddemod, dmodce, ddemodce, modmap, demodmap | For frequency shift keying method, the default separation between successive frequencies is Fd, not 2*Fd/M. For minimum shift keying method, the separation between frequencies is Fd/2, not Fd. |
| encode, decode | No longer support convolutional coding. Use convenc and vitdec instead. |
| gflineq | If the equation has no solutions, then the function returns an empty matrix, not a matrix of zeros. |
| randint | Uses state instead of seed to initialize random number generator. See rand for more information about initializing random number generators. |
| rcosflt | The 'wdelay' flag is superfluous. The function now behaves as the Release 11 function behaved with the 'wdelay' flag. |

## Obsolete Functions

The table below lists functions that are obsolete. Although they are included in Release 12 for backward compatibility, they might be removed in a future release. Where applicable, the second column lists functions that provide similar functionality. In some cases, the similar function requires different arguments or produces different results compared to the original function.

| Function | Similar Function, if Any |
|---|---|
| commgui | |
| convdeco | vitdec |
| convenco | convenc |
| eyescat | eyediagram, scatterplot |
| flxor | bitxor |

| Function | Similar Function, if Any |
|---|---|
| gen2abcd | |
| htruthtb | syndtable |
| imp2sys | |
| oct2gen | |
| randbit | randerr |
| sim2gen | |
| sim2logi | |
| sim2tran | |
| viterbi | vitdec |

## Known Software and Documentation Problems

This section updates the Communications Toolbox 2.0 documentation set, reflecting known documentation problems.

### Errors in Printed Version of Documentation

The printed version of the *Communications Toolbox User's Guide* contains several errors that have been corrected in the electronic documentation:

- On page 2-54 (under Example: A Rate-2/3 Feedforward Encoder), the first line of code is missing a right parenthesis before the semicolon. The line should read

  ```
  trel = poly2trellis([5 4],[27 33 0;0 5 13]); % Define trellis.
  ```

- The `ddemod` reference page indicates that the function's default behavior uses a Butterworth filter. As of Release 12, if `num` is empty, zero, or absent, then the function does not use a filter.
- The `gfadd` and `gfsub` reference pages incorrectly say, " If `len` is a negative number, then the length of the sum is not altered." In fact, negative values of `len` cause the function to truncate the arithmetic result.
- The `gflineq` reference page omits the syntax for solving a linear equation over GF(p) when p > 2. Also, an example there shows an output of `x2 =[0; 0; 0]` in the case when the linear equation has no solution. As of Release 12, the function returns an empty matrix, `x2 = []`, in this case.
- The example on the `dmodce` reference page shows incorrect output. The electronic version has a revised example with accurate output.
- Two documentation examples that use `vitdec` produce better error statistics than the printed documentation shows. Incorrect output values are on the `vitdec` reference page and on page 2-52 (under Example: Soft-Decision Decoding).

# Control System Toolbox 5.0 Release Notes

**New Features** . . . . . . . . . . . . . . . . . . . . 9-2
SISO Design Tool . . . . . . . . . . . . . . . . . . . 9-2
LTI Viewer . . . . . . . . . . . . . . . . . . . . . . 9-2
Property and Preference Editors . . . . . . . . . . . . . 9-3
Algorithmic Enhancements . . . . . . . . . . . . . . . 9-3
New Demo Suite . . . . . . . . . . . . . . . . . . . . 9-3
Documentation . . . . . . . . . . . . . . . . . . . . 9-3

**Platform Limitations for HP and IBM** . . . . . . . . . 9-5
Editors . . . . . . . . . . . . . . . . . . . . . . . 9-5
SISO Design Tool . . . . . . . . . . . . . . . . . . . 9-5

## New Features

This section introduces the new features and enhancements added in the Control System Toolbox 5.0 since the Control System Toolbox 4.2.1 (Release 11.0).

For information about Control System Toolbox new features that are incorporated from prior releases, see *Release 11 New Features* (for enhancements introduced between Release 10.0 and Release 11.0).

### SISO Design Tool

The SISO Design Tool is a new visual interface that greatly simplifies the design of compensators for single-input, single-output (SISO) linear systems. Using editable root locus and Bode diagram views, you can graphically adjust the compensator gain, poles, and zeros, while monitoring the closed-loop responses and stability margins.

Other features include storing intermediate designs, discretizing the compensator, and visualizing design constraints.

For detailed examples of how to use the SISO Design Tool, see “Designing Compensators” in *Getting Started with the Control System Toolbox*.

### LTI Viewer

Enhancements to the LTI Viewer include:

- Data markers for scanning data off response plots and annotating plots with critical response values
- Adaptive S and Z grids for pole/zero and root locus plots
- Sharper Nyquist and Nichols plots with less user intervention
- Option to display all crossover frequencies for gain and phase margins

For detailed examples of how to use the LTI Viewer, see “Analyzing Models” in *Getting Started with the Control System Toolbox*.

## Property and Preference Editors

You can use new Property and Preference Editors to set plot options such as titles, fonts, units, and grids. You can set toolbox preferences to persist from session to session, or set tool preferences for instances of the LTI Viewer and SISO Design Tool during a single session. You can also customize individual response plots using the Property Editor.

For a discussion of how to set properties and preferences, see “Setting Plot Properties and Preferences” in the Control System Toolbox online help.

## Algorithmic Enhancements

Enhancements to controls algorithms include:

- A new function, `allmargin`, for computing all crossover frequencies and corresponding gain margins, phase margins, and delay margins
- A new algorithm for computing stability margins of SISO systems, including systems with delays
- An enhanced root locus plotting algorithm

## New Demo Suite

The Control System Toolbox demos have been upgraded and considerably expanded. The new demo suite includes basic "getting started" tutorials, interactive demos, web-based GUI demos, and an extensive set of case studies covering dc motors, op amplifiers, disk drives, aircraft autopilots, heat exchangers, and steel rolling mills.

## Documentation

The Control System Toolbox documentation has been thoroughly revamped and now includes the following:

- A new *Getting Started with the Control System Toolbox* manual, which introduces the main features of the toolbox through extended examples. The focus of this book is on basic control engineering tasks, including building models, analyzing model responses, and designing compensators. The examples presented use the LTI Viewer and SISO Design Tool extensively.

- Online documentation, including a complete function reference and a set of design case studies that demonstrate the advanced capabilities of the Control System Toolbox.

## Platform Limitations for HP and IBM

The following features are not supported on the HP and IBM platforms. For background information, see “Platform Limitations” on page 1-17.

### Editors

The Toolbox Preferences Editor and Response Property Editor are not supported. The LTI Viewer Preferences Editor is supported with a reduced set of features (no tools to set preferences for grids, fonts, colors, or phase wrapping).

### SISO Design Tool

The following features are not supported for the SISO Design Tool:

- Preference and property editing
- Compensator format editing
- Storing and retrieving compensators

In addition, the SISO Tool Export window has fewer export options.

# Data Acquisition Toolbox 2.0 Release Notes

**New Features** . . . . . . . . . . . . . . . . . . . . . . 10-2
ComputerBoards Support . . . . . . . . . . . . . . . . . . 10-2
Adaptor Kit . . . . . . . . . . . . . . . . . . . . . . . 10-2
New Functions and Properties . . . . . . . . . . . . . . . . 10-3
Modified Functions and Properties . . . . . . . . . . . . . 10-4

**Major Bug Fixes** . . . . . . . . . . . . . . . . . . . . 10-5

**Upgrading from an Earlier Release** . . . . . . . . . . 10-6
Vendor Drivers . . . . . . . . . . . . . . . . . . . . . 10-6
Removed and Obsolete Functions . . . . . . . . . . . . . 10-6
Two daqhwinfo Fields Removed . . . . . . . . . . . . . . 10-6

**Known Software Problems** . . . . . . . . . . . . . . . 10-7
Functions . . . . . . . . . . . . . . . . . . . . . . . . 10-7
Vendors . . . . . . . . . . . . . . . . . . . . . . . . . 10-7

## New Features

This section introduces the new features and enhancements added in the Data Acquisition Toolbox 2.0 since the Data Acquisition Toolbox 1.0.1 (Release 11.1).

For information about Data Acquisition Toolbox new features that are incorporated from prior releases, see *Release 11.1 New Features* (enhancements introduced between Release 11.0 and Release 11.1).

### ComputerBoards Support

The Data Acquisition Toolbox 2.0 supports ComputerBoards hardware. You can create device objects associated with the analog input, analog output, and digital I/O subsystems for these boards.

You should use the Universal Library 5.1 drivers or the associated release of the InstaCal software with your ComputerBoards hardware (`http://www.computerboards.com/`).

#### Example: Creating a ComputerBoards Analog Input Object

To create the analog input object `ai` associated with your ComputerBoards hardware, you must supply the ComputerBoards adaptor name and the device ID to the `analoginput` creation function.

```
ai = analoginput('cbi',1);
```

You can use the `set` function to display the properties associated with `ai`.

```
set(ai)
```

### Adaptor Kit

An adaptor kit for writing additional adaptors for the toolbox is included with this release. The kit includes source code for an example adaptor and documentation of the API. Additionally, the source code for the ComputerBoards, National Instruments, ComputerBoards, Agilent Technologies, and Windows sound card adaptors is included in the `toolbox/daq/daq/src` directory. You can find the adaptor kit software in the `toolbox/daq/daqadaptor` directory. You can download the adaptor kit documentation via the MathWorks FTP server, `ftp://ftp.mathworks.com/pub/mathworks/toolbox/daq/adaptor_kit/AdaptorKit.pdf`.

## New Functions and Properties

The new Data Acquisition Toolbox 2.0 functions are described below.

| Function | Description |
|---|---|
| `addmuxchannel` | Add channels when using a National Instruments AMUX-64T multiplexer. |
| `muxchanidx` | Return multiplexed scanned channel index. |
| `waittilstop` | Wait for the device object to stop running |

The new Data Acquisition Toolbox 2.0 properties are described below.

| Property | Description |
|---|---|
| `ManualTriggerHwOn` | Control when the hardware starts. Valid values are `Start` and `Trigger`. A value of `Start` starts the hardware after the `start` function executes. A value of `Trigger` starts the hardware after the `trigger` function executes. Use the `Trigger` value to accurately start multiple hardware devices. |
| `NativeOffset` | Specify the offset to use when converting data from native format to doubles. |
| `NativeScaling` | Specify the scaling to use when converting data from native format to doubles. |

Both `NativeOffset` and `NativeScaling` are channel properties for analog input and analog output objects. Therefore, you can configure these properties on a per-channel basis.

## Modified Functions and Properties

The modified Data Acquisition Toolbox functions are described below.

| Function | Description |
|---|---|
| `daqhwinfo` | The `ConversionExtraScaling` and `ConversionOffset` fields have been removed. You can use the `NativeScaling` and `NativeOffset` properties to convert native data to doubles. |
| `flushdata` | You can now specify that data corresponding to an integral number of triggers is flushed from the data acquisition engine. |
| `getdata` | The event data structure includes only the events associated with the data being retrieved. Previously, the entire event structure of the device object was returned. |

The modified Data Acquisition Toolbox properties are described below.

| Property | Description |
|---|---|
| `TriggerChannel` | When using National Instruments 611x series boards, you can now specify any channel in the channel list as the trigger source. |
| `TriggerRepeat` | You can now define trigger repeats for hardware trigger types. |

## Major Bug Fixes

The Data Acquisition Toolbox includes several bug fixes, including the following descriptions (online only) of particularly important bug fixes.

## Upgrading from an Earlier Release

This section describes the upgrade issues involved in moving from the Data Acquisition Toolbox 1.0.1 to the Data Acquisition Toolbox 2.0.

### Vendor Drivers

If you are not using the latest drivers for your hardware, and you experience problems with the toolbox, then you should upgrade your drivers:

- For National Instruments hardware, you should use NI-DAQ 6.7.0 (`http://www.ni.com/nidaq`). Note that you should be able to use NI-DAQ 6.8, although the toolbox has not been fully tested against this driver version.
- For Agilent Technologies hardware, you should use VXI Plug and Play Drivers version A.06.04 (`http://www.tm.agilent.com`).

### Removed and Obsolete Functions

The `nidaq`, `winsound`, and `hpe1432` functions have been removed. The information contained in these M-files is included in these *Release Notes*.

The `obj2code` function has been renamed to `obj2mfile`. `obj2code` will still work but it now produces a warning.

### Two daqhwinfo Fields Removed

The `ConversionExtraScaling` and `ConversionOffset` fields of the `daqhwinfo` function have been removed. You can use the `NativeScaling` and `NativeOffset` properties to convert native data to doubles.

## Known Software Problems

This section updates the Data Acquisition Toolbox 2.0 documentation set, describing known software problems.

### Functions

#### The putdata Function

You should not modify the `BitsPerSample`, `InputRange`, `SensorRange`, and `UnitsRange` properties after calling `putdata`. If these properties are modified, all data is deleted from the data acquisition engine.

If you add a channel after calling `putdata`, then all data will be deleted from the buffer.

#### The daqread Function

Information that you store in the `UserData` property is not returned from a .daq file.

### Vendors

#### National Instruments Hardware

- If you use the Data Acquisition Toolbox and National Instruments' Measurement and Automation (M&A) Explorer at the same time, a conflict will occur and you will not be able to access your board. To avoid a conflict, you should access your board using either the toolbox or the M&A Explorer, and close the other software application.
- If you install NI-DAQ on your computer, and then install LabVIEW 6i on the same computer, you will need to reinstall NI-DAQ.
- You must use the PFI7 pin to input an external STARTSCAN signal. You must use the PFI2 pin to input an external CONVERT* signal.
- When running at a sampling rate of 5000 Hz or higher and with a `TransferMode` property value of `Interrupt`, there may be a considerable decline in system performance.
- You should configure the `SampleRate` property with the `setverify` function just before starting the hardware. Note that the `SampleRate` value depends

on the number of channels added to the device object, and the ChannelSkew property value depends on the SampleRate value.

- When using the 1200 series hardware, you must add channels in reverse order. If you specify invalid channels, the data acquisition engine will create the number of requested channels with valid hardware IDs. You can determine the hardware IDs with the object's display or with the HwChannel property.
- Only one digital I/O (DIO) object should be associated with a given DIO subsystem. To perform separate tasks with the hardware lines, you should add all the necessary lines to the DIO object, but partition them into separate line groups based on the task.
- All channels contained within an analog input object must have the same polarity. In other words, the InputRange property for these channels must have all unipolar values or all bipolar values.
- When using mux boards, you must add channels in a specific order using the addmuxchannel function.
- If you have trouble acquiring data with the DAQPad-MIO-16XE-50, you should increase the size of the engine buffer with the BufferingConfig property.

### ComputerBoards Hardware

- For boards that do not have a channel gain list, an error occurs at startup if all the channel input ranges are not the same or the channel scan order is not contiguous. However, if ClockSource is software, this rule does not apply.
- You should configure the SampleRate property with the setverify function just before starting the hardware. Note that the SampleRate value is dependent upon the number of channels added to the device object.
- For boards that do not support continuous background transfer mode (i.e., the board does not have hardware clocking), the only available ClockSource property value is software.
- When running at a sampling rate of 5000 Hz or higher and with a TransferMode property value of InterruptPerPoint, there may be a considerable decline in system performance.

- Most boards do not support simultaneous input and output. However, if software clocking is used, then this limitation does not apply.
- To use hardware digital triggers with the PCI-DAS4020/12 board, you must first configure the appropriate trigger mode with InstaCal.
- Some boards may return erroneous digital I/O ports – usually with an ID of 0.
- Expansion boards are not supported. This includes the CIO-EXP family of products.
- MEGA-FIFO hardware is not supported.

### Agilent Technologies Hardware

- When you start an analog input object associated with an E1432A or E1433A board that has the Arbitrary Source Output option, the source is automatically started. Therefore, you should not use a `TriggerType` property value of `Manual` with hardware having this option.
- For analog output objects, you should configure the `SampleRate` and `Span` properties with the `setverify` function just before starting the hardware, since these property values depend on the number of channels contained by the analog output object.
- The first time you connect a device object to an Agilent board, a list of available hardware is determined and all the hardware is initialized. However, this list is not updated during a MATLAB session. Therefore, if you install a new board or remove an existing board while MATLAB is running, you will not see the new configuration. To see the new configuration, you must restart MATLAB. When all device objects are deleted from the data acquisition engine, all the hardware is closed.
- For the E1433A, the minimum sampling rate is 20 Hz and the minimum span is 7.8125.
- The first channel in the `TriggerChannel` property list is used to trigger the object.
- For the E1434A, channels 1 and 2 and channels 3 and 4 share a 56000 DSP. Therefore, certain operational aspects are coupled between the channels in each pair. For example, both channels in each pair will have the same `RampRate` property value.
- You must add channels in increasing order and a channel array cannot contain repeated channels.

- If you create a device object that spans multiple boards, the device object should list the logical addresses using the same order as returned by the `daqhwinfo` function. To determine the logical address order, use the `daqhwinfo` function with no input arguments and examine the `InstalledBoardIds` field.

### Windows Sound Cards

- The maximum sampling rate depends on the `StandardSampleRates` property value. If `StandardSampleRates` is `On`, the maximum `SampleRate` property value is `44100`. If `StandardSampleRates` is `Off`, the maximum `SampleRate` property value is `96000` if supported by the sound card.

  For some sound cards that allow nonstandard sampling rates, certain value above 67,000 Hz will cause your computer to hang.
- If you are acquiring data when `StandardSampleRates` is `Off`, one of these messages may be returned to the command line depending on the specific sound card you are using:
  - `"Invalid format for device winsound"` occurs when the sound card does not allow for any nonstandard value.
  - `"Device Winsound already in use"` occurs when a nonstandard sampling rate is specified and the device takes longer than expected to acquire data.

# Database Toolbox 2.1 Release Notes

**New Features** . . . . . . . . . . . . . . . . . . . . . . 11-2
Platforms Supported . . . . . . . . . . . . . . . . . . . 11-2
Performance Improvement . . . . . . . . . . . . . . . . . . 11-2
Starting the Database Toolbox . . . . . . . . . . . . . . 11-2
Exporting Results to Report Generator . . . . . . . . . . 11-2
Grouping Constraints for a Single Field . . . . . . . . . 11-3

**Platform Limitations for HP 10.2** . . . . . . . . . . . 11-4

**Upgrading from an Earlier Release** . . . . . . . . . 11-5
Starting the Database Toolbox . . . . . . . . . . . . . . 11-5

## New Features

This section introduces the new features and enhancements added in the Database Toolbox 2.1 since the Database Toolbox 2.0 (Release 11.0).

For information about Database Toolbox new features that are incorporated from prior releases, see:

- *Release 11.1 New Features*
  (enhancements introduced between Release 11.0 and Release 11.1)
- *Release 11 New Features*
  (enhancements introduced between Release 9.0 and Release 11.0)

### Platforms Supported

The Database Toolbox now runs on all platforms that support MATLAB 6, with the exception of the HP 10.2 (HP 700).

### Performance Improvement

Version 2.1 performance for fetching data from your database has increased by a factor of roughly 100 over Version 2.0. This improvement was first introduced in Version 2.0.1.

### Starting the Database Toolbox

Do not run `feature('dispatchjava',1)` to start the Database Toolbox, as was required for Version 2.0. Instead, begin by running the Database Toolbox function you want to use.

### Exporting Results to Report Generator

When using the Visual Query Builder, you can now export query results using the Report Generator, if the Report Generator product is part of your system configuration. To use it, select **Report Generator** from the Visual Query Builder **Display** menu.

## Grouping Constraints for a Single Field

A **Group** button has been added to the **Where**, **Subquery**, and **Having** dialog boxes. Use the **Group** button to group constraints for a single field, especially when using the `OR` operator. Basically, the **Group** button allows you to evaluate a set of constraints as a whole.

## Platform Limitations for HP 10.2

The Database Toolbox is not supported on the Hewlett-Packard 10.2 platform. For background information, see “Platform Limitations” on page 1-17.

## Upgrading from an Earlier Release

This section describes the upgrade issues involved in moving from the Database Toolbox 2.0.1 to the Database Toolbox 2.1.

For information about upgrading from a release earlier than 2.0.1, see *Release 11 New Features*.

### Starting the Database Toolbox

Do not run `feature('dispatchjava',1)` to start the Database Toolbox, as was required for Version 2.0. Instead, begin by running the Database Toolbox function you want to use.

# Datafeed Toolbox 1.2 Release Notes

**Introduction to the Datafeed Toolbox** . . . . . . . . . 12-2
Learning More About the Datafeed Toolbox . . . . . . . . . 12-2

**New Features** . . . . . . . . . . . . . . . . . . . 12-3
Support for IDC and Yahoo . . . . . . . . . . . . . . . 12-3

## Introduction to the Datafeed Toolbox

The Datafeed Toolbox was introduced after Release 11.1. It was distributed via Web download.

The Datafeed Toolbox effectively turns your MATLAB workstation into a financial data acquisition terminal. Using the Datafeed Toolbox, you can download a wide variety of security data from financial data servers into your MATLAB workspace. Then, you can pass this data to MATLAB or to another toolbox, such as the Financial Time Series Toolbox, for further analysis.

### Learning More About the Datafeed Toolbox

The Datafeed Toolbox documentation describes the product in detail. You can also read about the Datafeed Toolbox in the products section of the MathWorks Web page (`http://www.mathworks.com`).

## New Features

This section introduces the new features and enhancements added in the Datafeed Toolbox 1.2 after the Datafeed Toolbox 1.0 (Release 11.1).

### Support for IDC and Yahoo

The Datafeed Toolbox now supports connections to Interactive Data Corporation (IDC) and Yahoo data servers in addition to Bloomberg.

# Developer’s Kit for Texas Instruments™ DSP 1.0 Release Notes

**Introduction to Developer’s Kit for Texas Instruments™ DSP** . . . . . . . . . . . . . 13-2

**Known Software and Documentation Problems** . . . . 13-3
Revision 0 TMS320C6701 Evaluation Modules . . . . . . . 13-3
Real-Time Workshop Compiler Optimization Option . . . . 13-3

## Introduction to Developer’s Kit for Texas Instruments™ DSP

Developer’s Kit for Texas Instruments™ DSP (DKTI™) is a new product that is being introduced in Release 12.

**Note** The Developer’s Kit for Texas Instruments DSP product is not part of Release 12, but is targeted to be available in the near future.

Composed of three component collections, DKTI lets you develop signal processing systems for the C6701 EVM, and use MATLAB to access and load data to and from the digital signal processor on a target signal processor. The three libraries are

| Product Name | Library Name | Description |
|---|---|---|
| Target for C6701 EVM | | Contains the blockset needed to use Simulink and RTW to create applications for the C6701 EVM. Comprising four blocks, you use this blockset to configure the coder-decoder (codec) and LEDs on the C6701 EVM. |
| Link for Code Composer Studio IDE | | Contains the Target for Code Composer Studio IDE interface software, and ???? |
| Link for RTDX Interface | | Contains MATLAB objects and methods for communicating with target digital signal processors on boards installed in your PC, such as the C6701 EVM. |

DKTI lets you use Simulink to model floating-point digital signal processing algorithms from blocks in the DSP Blockset, and then use the Real-Time Workshop (RTW) to build ANSI C code targeted to the TMS320C6701 EVM (C6701 EVM). DKTI takes the generated C code and uses Texas Instruments (TI) tools to build C6701 EVM-specific machine code. The build process downloads the targeted machine code to the evaluation module and runs the executable on the TMS320C6701 Digital Signal Processor.

After downloading the code to the board, your digital signal processing application or algorithm runs automatically on the evaluation module. You have the option of compiling your model without downloading the code to the target.

Texas Instruments produces the C6701 EVM to help you create DSP applications for the Texas Instruments TMS320C6701 processor. You can create, test, and deploy your signal processing algorithms or filters on the target processor without the difficulties inherent in starting with the digital signal processor itself and building the support hardware to test the application on the processor. Instead, the C6701 EVM provides the input hardware, output hardware, timing circuitry, memory, and power for the C6701 digital signal processor. Texas Instruments provides the software tools, such as the C compiler, linker, and assembler, and Code Composer Studio (CCS), for you to use your PC to develop, download, and test your algorithms and applications on their C6701 EVM. If you want, you can launch Code Composer Studio Integrated Development Environment (CCS IDE) directly from DKTI by using the CCS IDE target. When you use the CCS IDE option, DKTI launches the integrated development environment and populates a new development project with the files from your RTW build process.

## Known Software and Documentation Problems

This section updates the Developer's Kit for Texas Instruments DSP (DKTI) 1.0 documentation, reflecting known software problems.

### Revision 0 TMS320C6701 Evaluation Modules

Due to a bug in the C6701 Digital Signal Processor on the evaluation modules, a number of restrictions apply when you use DKTI with revision 0 TMS320C67701 Evaluation Modules (EVM).

- The demonstration programs provided with the toolbox – Reverberation and Wavelet Denoising – will not run on revision 0 C6701 EVMs. For more information about this problem, refer to "Using DKTI with Rev 0 C6701 Evaluation Modules" in *Developer's Kit for Texas Instruments™ DSP User's Guide*.
- You are restricted to using internal memory on the revision 0 EVMs. Refer to "Using DKTID with Rev 0 C6701 Evaluation Modules" in *Developer's Kit for Texas Instruments™ DSP User's Guide* for more information.

### Real-Time Workshop Compiler Optimization Option

Although some figures in the documentation show an option used to optimize the code generated by the Texas Instruments Compiler, the option is not available in the product.

# Dials & Gauges Blockset 1.1 Release Notes

**New Features** . . . . . . . . . . . . . . . . . . . . . . . 14-2
Real-Time Workshop Support . . . . . . . . . . . . . . . 14-2
External Mode Support . . . . . . . . . . . . . . . . . . . . 14-2
Block Parameters Dialog Boxes Enhanced . . . . . . . . 14-2
New Demos Added . . . . . . . . . . . . . . . . . . . . . 14-3

**Upgrading from an Earlier Release** . . . . . . . . . 14-4
Double-Clicking on Block Borders . . . . . . . . . . . . . 14-4
Values Inserted in the “Events on which to output” Field . . 14-4

**Known Software and Documentation Problems** . . . . 14-5
Error Message at Installation . . . . . . . . . . . . . . 14-5
Documentation Update . . . . . . . . . . . . . . . . . 14-5

## New Features

This section introduces the new features and enhancements added in the Dials & Gauges Blockset 1.1 since the Dials & Gauges Blockset 1.0 (Release 11.1).

### Real-Time Workshop Support

You can now use Real-Time Workshop 4.0 to generate code from models that include Dials & Gauges Blockset blocks.

For dials, the code you generate contains static values (i.e., the value specified at the time of code generation). Gauges are ignored during code generation, except through the use of external mode (see below). If you want to manipulate dials and view the gauges, you can do so through the Real-Time Workshop's external mode.

### External Mode Support

The Dials & Gauges Blockset 1.1 support for external mode allows you to incorporate dials and gauges into any target that you can connect to through external mode (e.g., the xPC Target and Real-Time Windows Target environments; see the documentation for those products for details).

For more information about external mode, see the external mode section of the *Real-Time Workshop User's Guide*.

### Block Parameters Dialog Boxes Enhanced

A new field, **Event on which to output**, has been added to the **Block Parameters** dialog box for dials.

This field has been added to allow dial controls to be more efficient. In the Dials & Gauges Blockset 1.0, at each time step Simulink queried the dial for its value. Now, in the Dials & Gauges Blockset 1.1, when you move a dial, an event occurs that changes the output value of the block. This new event-driven approach is more efficient than the former approach of repeatedly requesting the same information at successive time steps.

The **Event on which to output** field allows you to specify what events will cause the value of the output to be updated.

**Note** The field that was called **Event** in the Dials & Gauges Blockset 1.0 has been renamed in the Dials & Gauges Blockset 1.1; it is now called **Other events and handlers**.

## New Demos Added

The Global Majic ActiveX Library, `dng_gmslib`, contains two new demo sublibraries:

- Demo Aircraft Instruments
- Demos Joystick Control

These sublibraries contain ActiveX controls that use time-limited evaluation licenses from Global Majic, Inc. Contact The MathWorks for details about purchasing full licenses for those controls.

## Upgrading from an Earlier Release

This section describes the upgrade issues involved in moving from the Dials & Gauges Blockset 1.0 to the Dials & Gauges Blockset 1.1.

### Double-Clicking on Block Borders

In the Dials & Gauges Blockset 1.0, double-clicking on the border of a block displayed the **Block Parameters** dialog box. Now, in the Dials & Gauges Blockset 1.1, double-clicking on a block that is supplied with the blockset (i.e., a built-in block) displays the ActiveX Control property sheet. If you double-click on a user-created block, the **Block Parameters** dialog box is displayed (i.e., the behavior is the same as it was for the Dials & Gauges Blockset 1.0).

### Values Inserted in the "Events on which to output" Field

When you open a Dials & Gauges Blockset 1.0 model with the Dials & Gauges Blockset 1.1, default values may be automatically inserted in the **Events on which to output** field. This occurs for built-in Dials & Gauges Blockset blocks when this field is empty. See "Block Parameters Dialog Boxes Enhanced" for details.

## Known Software and Documentation Problems

This section updates the Dials & Gauges Blockset documentation set, reflecting known software and documentation problems.

### Error Message at Installation

When you install the Dials & Gauges Blockset, if you receive an error message referring to an `.ocx` component, similar to the following message,

```
Copying Dials & Gauges Blockset files
ads.ocx self registering file did not register
```

see Solution Number 24876 in the Support section of the MathWorks Web site (`http://www.mathworks.com`) for information on how to address this problem.

### Documentation Update

The next to last sentence in the “Notes on Third-Party ActiveX Control Blocks” on page 3-8 of the printed *Dials & Gauges Blockset User's Guide* that says “Assign the MATLAB command to that event as a callback, for easy editing of the control.” is not accurate.

For updated information, see the online version of that section.

# DSP Blockset 4.0 Release Notes

**New Features** . . . . . . . . . . . . . . . . . . . . . . 15-2
Matrix Support . . . . . . . . . . . . . . . . . . . . . . 15-2
Frame Support . . . . . . . . . . . . . . . . . . . . . . 15-4
New and Enhanced Blocks . . . . . . . . . . . . . . . . . . 15-5
Library Reorganization . . . . . . . . . . . . . . . . . . 15-7

**Platform Limitations for HP and IBM** . . . . . . . . 15-12

**Upgrading from an Earlier Release** . . . . . . . . . 15-13
Adding Release 12 Blocks to Release 11 Models . . . . . 15-13

## New Features

This section introduces the new features and enhancements added in the DSP Blockset 4.0 since the DSP Blockset 3.1 (Release 11.0).

For information about the DSP Blockset 4.0 new features that are incorporated from prior releases, see:

- *Release 11.1 New Features*
  (enhancements introduced between Release 11.0 and Release 11.1)
- *Release 11 New Features*
  (enhancements introduced between Release 9 and Release 11)

Version 4.0 of the DSP Blockset offers complete support for the Simulink 4.0 enhancements that provide seamless propagation and processing of both matrix and frame-based signals. A number of new blocks are also provided.

This section is organized into the following subsections:

- “Matrix Support” on page 15-2
- “Frame Support” on page 15-4
- “New and Enhanced Blocks” on page 15-5
- “Library Reorganization” on page 15-7

### Matrix Support

Matrix support is now provided natively within Simulink, which allows the DSP Blockset to generate and process matrix signals in a much more natural manner than was previously possible.

#### Generating Matrix Signals

Matrix signals can now be generated in the same way as vector or scalar signals, using appropriate source blocks such as Signal From Workspace or DSP Constant. Where you would have previously specified a vector or scalar, you can now use standard MATLAB notation to specify a matrix. For example, enter `[1 2 3;4 5 6]` in the **Constant value** parameter of the DSP Constant block to generate the 2-by-3 output matrix.

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$$

When **Signal dimensions** is selected from the model window **Format** menu, Simulink displays the dimensions of an M-by-N matrix signal as a pair of bracketed integers, `[MxN]`. Simulink *does not display* any size information for a scalar signal. Dimension information for a signal can also be displayed in a model by attaching a Probe block with **Probe signal dimensions** selected.

Row vectors and column vectors are matrices with M=1 or N=1, respectively. These are generated by the DSP Constant block when a vector is specified in the **Constant value** parameter and the **Interpret vector parameters as 1-D** check box is *not* selected.

In addition, Simulink recognizes one-dimensional (1-D) vectors, which are ordered lists of values. These are generated by the DSP Constant block when a vector is specified in the **Constant value** parameter and the **Interpret vector parameters as 1-D** check box is selected. When **Signal dimensions** is selected from the model window **Format** menu, Simulink displays the size of a 1-D vector signal as an unbracketed integer. For convenience, DSP Blockset blocks treat 1-D vectors as column vectors for most operations.

### Processing Matrix Signals

The most noticeable benefit of the new matrix enhancements is that you no longer need to manually specify the size of matrix inputs to a block. All DSP Blockset blocks now automatically detect input dimensions, and will generate an error if those dimensions are not appropriate for the specific operation.

Since input dimensions are detected automatically by new (Version 4.0) blocks, a matrix output of an older block may need to be altered before you use is as an input to a new block, and vice versa. See “Adding Release 12 Blocks to Release 11 Models” on page 15-13 for more information.

The interpretation of sample-based and frame-based matrices is the same as in previous releases. An M-by-N sample-based matrix represents M*N independent channels, each containing one sample. An M-by-N frame-based matrix represents M consecutive samples from each of N independent channels. That is, each column of a frame-based matrix represents M consecutive samples from a single channel.

## Frame Support

Support for frame-based signals is also now provided natively within Simulink, which allows for straightforward propagation of frame-based data throughout a model.

### Generating Frame-Based Signals

Frame-based signals are generated in much the same way that they were in previous DSP Blockset releases. In constant blocks such as DSP Constant or Constant Diagonal Matrix, the **Frame-based output** check box confers frame-based status when selected. In other source blocks, such as Signal From Workspace, Chirp, and Random Source, the output is frame based whenever the **Samples per frame** setting is greater than 1.

Note that 1-D vectors cannot be frame based. See the DSP Constant block reference for information about the interaction of the **Frame-based output** and **Interpret vector parameters as 1-D** parameters.

### Processing Frame-Based Signals

Frame-based status is now propagated automatically throughout a model. Most blocks do not change the frame status of a signal; frame-based signals remain frame based, and sample-based signals remain sample based. The exceptions to this rule are blocks, like Buffer and Frame Status Conversion, that are specifically intended to alter the frame status of a signal.

Since frame status is now automatically detected by the new (Version 4.0) blocks, a frame-based output from an older block may need to be altered before you use it as an input to a new block, and vice versa. See “Adding Release 12 Blocks to Release 11 Models” on page 15-13 for more information.

As mentioned above, the interpretation of sample-based and frame-based matrices is the same as in previous releases. An M-by-N sample-based matrix represents M*N independent channels, each containing one sample. An M-by-N frame-based matrix represents M consecutive samples from each of N independent channels. For convenience, certain vector-oriented blocks such as Autocorrelation treat sample-based 1-by-N matrices (i.e., row vectors) as N-by-1 matrices (i.e., column vectors). See the documentation for the particular block in which you are interested.

Simulink displays sample-based signals using a single line, $\rightarrow$, and frame-based signals using a double line, $\Rightarrow$. The sample period ($T_s$) of a sample-based signal, or the frame period ($T_f$) of a frame-based signal, can be displayed by connecting a Probe block to the line. As always, the sample period of a frame-based signal is $T_f$ / M, where M is the signal's frame size (i.e., the number of rows in the frame-based M-by-N matrix).

## New and Enhanced Blocks

Table 15-1 lists the blocks that are new in Version 4.0, as well as those (*) that received substantial enhancements above and beyond the general upgrade for matrix and frame support. Library levels are separated by a forward slash.

**Table 15-1: New and Enhanced Blocks in the DSP Blockset 4.0**

| Block Library | Block Name | Description |
|---|---|---|
| DSP Sources | Chirp* | Generate a swept-frequency cosine. Version 4.0 adds bidirectional sweep. |
| | Constant Ramp | Generate a ramp signal with length based on input dimensions. |
| | Window Function* | Compute a window, and/or apply a window to an input signal. Version 4.0 can compute user-defined windows. |
| DSP Sinks | Signal To Workspace | Replaces To Workspace. |
| | Spectrum Scope | Replaces FFT Frame Scope and Buffered FFT Frame Scope. |
| | Vector Scope | Replaces Time Frame Scope, User-Defined Frame Scope, and Frequency Frame Scope. |
| Estimation / Linear Prediction | Autocorrelation LPC | Determine the coefficients of an Nth-order forward linear predictor. |
| Math Functions / Matrices and Linear Algebra / Linear System Solvers | SVD Solver | Solve the equation AX = B using singular value decomposition. |
| Math Functions / Matrices and Linear Algebra / Matrix Factorizations | Singular Value Decomposition | Compute the singular value decomposition of a matrix. |
| Math Functions / Matrices and Linear Algebra / Matrix Inverses | Cholesky Inverse | Compute the inverse of a Hermitian positive definite matrix using Cholesky factorization. |
| | LDL Inverse | Compute the inverse of a Hermitian positive definite matrix using LDL factorization. |
| | LU Inverse | Compute the inverse of a square matrix using LU factorization. |
| | Pseudoinverse | Compute the Moore-Penrose pseudoinverse. |
| Math Functions / Matrices and Linear Algebra / Matrix Operations | Identity Matrix | Generate a matrix with ones on the main diagonal and zeros elsewhere. |
| | Submatrix* | Select a subset of elements (submatrix) from a matrix input. Version 4.0 offers a major redesign of the block's interface. |

**Table 15-1: New and Enhanced Blocks in the DSP Blockset 4.0 (Continued)**

| Block Library | Block Name | Description |
|---|---|---|
| Math Functions / Polynomial Functions | Least Squares Polynomial Fit | Compute the coefficients of the polynomial that best fits the input data in a least-squares sense. |
| | Polynomial Stability Test | Determine whether all roots of the input polynomial are inside the unit circle using the Schur-Cohn algorithm. |
| Signal Management / Buffers | Buffer* | Rebuffer the input sequence to a smaller or larger frame size. In Version 4.0, the Buffer block incorporates all the capabilities of the former Rebuffer block. See “Minor Enhancements” below for more information. |
| | Delay Line | Rebuffer a sequence of inputs with a one-sample shift. Previously called Shift Register. |
| | Triggered Delay Line | Rebuffer a sequence of inputs with a one-sample shift. Previously called Triggered Shift Register. |
| Signal Management / Indexing | Multiport Selector | Distribute rows or columns of an input to multiple output ports. |
| Signal Management / Signal Attributes | Check Signal Attributes | Generate an error if input attributes differ from (or match) those specified. |
| | Convert 1-D to 2-D | Reshape a 1-D or 2-D input to a 2-D matrix with the specified dimensions. |
| | Convert 2-D to 1-D | Convert a 2-D matrix input to a 1-D vector. |
| | Frame Status Conversion | Specify the frame status of the output, sample-based or frame-based. |
| Signal Operations | Pad | Alter the input size by padding or truncating rows and/or columns. |
| Simulink / Signals & Systems | Matrix Concatenation | Concatenate inputs horizontally or vertically. |

## Minor Enhancements

**Rowwise Operations.** A number of blocks were enhanced to operate along both the row and column dimensions of inputs. Among the improved blocks are Zero Pad, Flip, Difference, and Cumulative Sum. All these blocks now allow you to select **Rows** or **Columns** as the dimension along which to operate.

**Buffers Library Reorganization.** The Buffers library has undergone a minor reorganization. The Buffer block now incorporates all the capabilities of the former Rebuffer block, which has been removed. The former Shift Register and Triggered Shift Register blocks have also been renamed to Delay Line and Triggered Delay Line, respectively.

## Library Reorganization

The DSP Blockset libraries have been reorganized for clarity and accessibility. The revised library hierarchy is shown below:

- DSP Sinks
- DSP Sources
- Estimation
  - Linear Prediction
  - Parametric Estimation
  - Power Spectrum Estimation
- Filtering
  - Adaptive Filters
  - Filter Designs
  - Filter Structures
  - Multirate Filters
- Math Functions
  - Math Operations
  - Matrices and Linear Algebra
    - Linear System Solvers
    - Matrix Factorizations
    - Matrix Inverses
    - Matrix Operations
  - Polynomial Functions
- Quantizers
- Signal Management
  - Buffers
  - Indexing
  - Signal Attributes
  - Switches and Counters
- Signal Operations
- Statistics
- Transforms

Use the Simulink Library Browser to access the blockset directly through this hierarchical library list.

As a result of the reorganization, a number of existing blocks have been moved to new locations. Table 15-2 below lists the changes.

**Table 15-2: New Block Locations in Version 4.0**

| Old Block Name | Old Library Location | New Block Name | New Library Location |
|---|---|---|---|
| Analog Filter Design | Filtering / Filter Designs | same | Filtering / Filter Structures |
| Analytic Signal | General DSP / Signal Operations | same | Transforms |
| Autocorrelation | Math Functions / Vector Functions | same | Statistics |
| Backward Substitution | Math Functions / Linear Algebra | same | Math Functions / Matrices and Linear Algebra / Linear System Solvers |
| Biquadratic Filter | Filtering / Filter Realizations | same | Filtering / Filter Structures |
| Buffered FFT Frame Scope | DSP Sinks | Spectrum Scope | same |
| Cholesky Factorization | Math Functions / Linear Algebra | same | Math Functions / Matrices and Linear Algebra / Matrix Factorizations |
| Cholesky Solver | Math Functions / Linear Algebra | same | Math Functions / Matrices and Linear Algebra / Linear System Solvers |
| Commutator | General DSP / Switches and Counters | obsolete | obsolete |
| Complex Exponential | Math Functions / Elementary Functions | same | Math Functions / Math Operations |
| Constant Diagonal Matrix | DSP Sources, Math Functions / Matrix Functions | same | Math Functions / Matrices and Linear Algebra / Matrix Operations |
| Contiguous Copy | Math Functions / Elementary Functions | same | Signal Management / Signal Attributes |
| Convert Complex DSP To Simulink | Math Functions / Elementary Functions | obsolete | obsolete |
| Convert Complex Simulink To DSP | Math Functions / Elementary Functions | obsolete | obsolete |
| Convolution | Math Functions / Vector Functions | same | Signal Operations |
| Correlation | Math Functions / Vector Functions | same | Statistics |
| Create Diagonal Matrix | Math Functions / Matrix Functions | same | Math Functions / Matrices and Linear Algebra / Matrix Operations |
| Cumulative Sum | Math Functions / Vector Functions | same | Math Functions / Math Operations |

**Table 15-2: New Block Locations in Version 4.0 (Continued)**

| Old Block Name | Old Library Location | New Block Name | New Library Location |
|---|---|---|---|
| dB | Math Functions / Elementary Functions | dB Conversion | Math Functions / Math Operations |
| dB Gain | Math Functions / Elementary Functions | same | Math Functions / Math Operations |
| Detrend | General DSP / Signal Operations | same | Statistics |
| Difference | Math Functions / Vector Functions | same | Math Functions / Math Operations |
| Direct-Form II Transpose Filter | Filtering / Filter Realizations | same | Filtering / Filter Structures |
| Discrete Constant | DSP Sources | DSP Constant | same |
| Distributor | General DSP / Switches and Counters | obsolete | obsolete |
| Downsample | General DSP / Signal Operations | same | Signal Operations |
| Extract Diagonal | Math Functions / Matrix Functions | same | Math Functions / Matrices and Linear Algebra / Matrix Operations |
| Extract Triangular Matrix | Math Functions / Matrix Functions | same | Math Functions / Matrices and Linear Algebra / Matrix Operations |
| FFT Frame Scope | DSP Sinks | Spectrum Scope | same |
| Filter Realization Wizard | Filtering / Filter Realizations | same | Filtering / Filter Structures |
| Flip | Math Functions / Vector Functions | same | Signal Management / Indexing |
| Forward Substitution | Math Functions / Linear Algebra | same | Math Functions / Matrices and Linear Algebra / Linear System Solvers |
| Frequency Frame Scope | DSP Sinks | Vector Scope | same |
| Inherit Complexity | Math Functions / Elementary Functions | same | Signal Management / Signal Attributes |
| Integer Delay | General DSP / Signal Operations | same | Signal Operations |
| LDL Factorization | Math Functions / Linear Algebra | same | Math Functions / Matrices and Linear Algebra / Matrix Factorizations |
| LDL Solver | Math Functions / Linear Algebra | same | Math Functions / Matrices and Linear Algebra / Linear System Solvers |
| Levinson Solver | Math Functions / Linear Algebra | same | Math Functions / Matrices and Linear Algebra / Linear System Solvers |
| LPC | General DSP / Signal Operations | Autocorrelation LPC | Estimation / Linear Prediction |
| LU Factorization | Math Functions / Linear Algebra | same | Math Functions / Matrices and Linear Algebra / Matrix Factorizations |

Table 15-2: New Block Locations in Version 4.0 (Continued)

| Old Block Name | Old Library Location | New Block Name | New Library Location |
|---|---|---|---|
| LU Solver | Math Functions / Linear Algebra | same | Math Functions / Matrices and Linear Algebra / Linear System Solvers |
| Matrix 1-Norm | Math Functions / Matrix Functions | same | Math Functions / Matrices and Linear Algebra / Matrix Operations |
| Matrix Constant | DSP Sources, Math Functions / Matrix Functions | DSP Constant | DSP Sources |
| Matrix From Workspace | DSP Sources, Math Functions / Matrix Functions | From Workspace | DSP Sources |
| Matrix Multiplication | Math Functions / Matrix Functions | Matrix Multiply | Math Functions / Matrices and Linear Algebra / Matrix Operations |
| Matrix Product | Math Functions / Matrix Functions | same | Math Functions / Matrices and Linear Algebra / Matrix Operations |
| Matrix Scaling | Math Functions / Matrix Functions | same | Math Functions / Matrices and Linear Algebra / Matrix Operations |
| Matrix Square | Math Functions / Matrix Functions | same | Math Functions / Matrices and Linear Algebra / Matrix Operations |
| Matrix Sum | Math Functions / Matrix Functions | same | Math Functions / Matrices and Linear Algebra / Matrix Operations |
| Matrix To Workspace | DSP Sinks, Math Functions / Matrix Functions | To Workspace | DSP Sinks |
| Multiphase Clock | DSP Sources, General DSP / Switches and Counters | same | DSP Sources |
| Normalization | Math Functions / Vector Functions | same | Math Functions / Math Operations |
| Overlap-Add FFT Filter | Filtering / Filter Realizations | same | Filtering / Filter Structures |
| Overlap-Save FFT Filter | Filtering / Filter Realizations | same | Filtering / Filter Structures |
| Partial Unbuffer | General DSP / Buffers | Buffer / Unbuffer | same |
| Permute Matrix | Math Functions / Matrix Functions | same | Math Functions / Matrices and Linear Algebra / Matrix Operations |
| Polynomial Evaluation | Math Functions / Elementary Functions | same | Math Functions / Polynomial Functions |
| QR Factorization | Math Functions / Linear Algebra | same | Math Functions / Matrices and Linear Algebra / Matrix Factorizations |
| QR Solver | Math Functions / Linear Algebra | same | Math Functions / Matrices and Linear Algebra / Linear System Solvers |
| Rebuffer | General DSP / Buffers | Buffer/Unbuffer | same |

**Table 15-2: New Block Locations in Version 4.0 (Continued)**

| Old Block Name | Old Library Location | New Block Name | New Library Location |
|---|---|---|---|
| Reciprocal Condition | Math Functions / Linear Algebra | same | Math Functions / Matrices and Linear Algebra / Matrix Operations |
| Repeat | General DSP / Signal Operations | same | Signal Operations |
| Reshape | Math Functions / Matrix Functions | same | Simulink / Signals & Systems |
| Sample and Hold | General DSP / Switches and Counters | same | Signal Operations |
| Shift Register | General DSP / Buffers | Delay Line | same |
| Submatrix | Math Functions / Matrix Functions | same | Math Functions / Matrices and Linear Algebra / Matrix Operations |
| Time Frame Scope | DSP Sinks | Vector Scope | same |
| Time-Varying Direct-Form II Transpose Filter | Filtering / Filter Realizations | same | Filter Structures |
| Time-Varying Lattice Filter | Filtering / Filter Realizations | same | Filter Structures |
| Toeplitz | Math Functions / Matrix Functions | same | Math Functions / Matrices and Linear Algebra / Matrix Operations |
| Transpose | Math Functions / Matrix Functions | same | Math Functions / Matrices and Linear Algebra / Matrix Operations |
| Triggered Matrix To Workspace | DSP Sinks | Triggered To Workspace | same |
| Triggered Shift Register | General DSP / Buffers | Triggered Delay Line | same |
| Unwrap | Math Functions / Vector Functions | same | Signal Operations |
| Upsample | General DSP / Signal Operations | same | Signal Operations |
| User-Defined Frame Scope | DSP Sinks | Vector Scope | same |
| Variable Fractional Delay | General DSP / Signal Operations | same | Signal Operations |
| Variable Integer Delay | General DSP / Signal Operations | same | Signal Operations |
| Variable Selector | Math Functions / Elementary Functions | same | Signal Management / Indexing |
| Window Function | DSP Sources, General DSP / Signal Operations | same | Signal Operations |
| Zero Pad | General DSP / Signal Operations | same | Signal Operations |

## Platform Limitations for HP and IBM

On the HP and IBM platforms, the DSP Blockset has the same limitations imposed by Simulink. See “Platform Limitations for HP and IBM” in the “Simulink 4.0 Release Notes” section for more information.

## Upgrading from an Earlier Release

This section describes the upgrade issues involved in moving from the DSP Blockset 3.1 to the DSP Blockset 4.0.

For information about upgrading from a release earlier than 3.1, see *Release 11 New Features*.

### Adding Release 12 Blocks to Release 11 Models

All DSP Blockset Version 4.0 blocks support the Simulink 4.0 matrix and frame enhancements. Thus, when you add new (Version 4.0) blocks to a model containing older blocks, some of the signals may need to be altered. Otherwise, an older block may not correctly recognize the output from a new block, and vice versa. See “Using Version 4.0 Blocks with Blocks from Earlier Versions”, “Checking Block Versions”, and “Finding Replacement Blocks” below for details.

#### Using Version 4.0 Blocks with Blocks from Earlier Versions

Input dimension and frame status are detected automatically by new (Version 4.0) blocks but not by older blocks from previous releases. Thus, when both new and older blocks are used in a model, some signals may need to be passed through conversion blocks so the dimension and frame status of signals are recognized by their receiving blocks.

**Using Outputs from Older Blocks as Inputs to New Blocks.** All outputs of blocks from previous releases are 1-D vectors. You can use an output of an older block as an input to a new block without altering the signal. In this case, the new block will usually treat the 1-D input vector as a sample-based 2-D column vector (check the documentation for the particular block in which you are interested). You can also change the 1-D output of an older block to a 2-D sample- or frame-based matrix before feeding it into a new block: simply insert the Convert 1-D to 2-D block in the DSP Blockset Signal Attributes library. See the documentation for the Convert 1-D to 2-D block for more information.

**Using Outputs from New Blocks as Inputs to Older Blocks.** Older blocks can only accept 1-D inputs. To use an output from a new block as an input to an older block, pass the new output through the Convert 2-D to 1-D block found in the DSP Blockset Signal Attributes library. You can set the frame status and dimensions of the resulting 1-D signal in the parameters dialog box of the older block. See the documentation for the Convert 2-D to 1-D block for more information.

### Checking Block Versions

When you begin adding Version 4.0 blocks to an older model, use the `dsp_links` function to check the versions of all blocks in that model. When you type

```
dsp_links
```

at the command line, the blocks in current model are color coded to reflect the version of the DSP Blockset to which they are linked. By default, Version 4.0 blocks are displayed in green. If the color coding reveals that a Version 4.0 block is providing the input to a Version 3 or Version 2 block (displayed in yellow or red by default), the older block should be replaced with a current version. Alternatively, you can use conversion blocks as described previously in “Using Version 4.0 Blocks with Blocks from Earlier Versions”.

### Finding Replacement Blocks

In most cases, you can simply replace the older block with the current block of the same name. In general, you should open the block’s dialog box and verify that the settings are correct for the intended operation. (Many block dialogs have undergone minor refinements.)

In certain cases, a block may have changed name or location in Version 4.0. If you cannot find the block you are looking for, check Table 15-2 to see if it may have been renamed or relocated.

# Filter Design Toolbox 2.0 Release Notes

**New Features** . . . . . . . . . . . . . . . . . . . . . 16-2
New Quantization Tool for
Advanced FIR and IIR Filter Design . . . . . . . . . 16-2
New Objects . . . . . . . . . . . . . . . . . . . . . 16-3
New Filter Design Functions . . . . . . . . . . . . . . 16-3
New Filter Conversion Functions . . . . . . . . . . . . 16-4
New Filter Structures . . . . . . . . . . . . . . . . . 16-4
New Analysis Methods . . . . . . . . . . . . . . . . . 16-5
New Functions and Enhancements . . . . . . . . . . . 16-5

**Platform Limitations for HP 10.2** . . . . . . . . . . . 16-7

**Upgrading from an Earlier Release** . . . . . . . . . 16-8
Obsolete Functions in Version 2.0 . . . . . . . . . . . 16-8

**Known Software and Documentation Problems** . . . . 16-9
Filter Design and Analysis Tool . . . . . . . . . . . . 16-9
Switching Between Design and
Quantization Modes in FDATool . . . . . . . . . . . 16-9

## New Features

Incorporating functionality from the Quantized Filtering Toolbox, Filter Design Toolbox 2.0 is a new product for digital filter design, implementation, and analysis using fixed-point or arbitrary-precision floating-point arithmetic. This toolbox includes advanced FIR and IIR filter design methods and a new quantization tool for designing and analyzing quantized FIR and IIR filters. This section introduces the features and enhancements added in the Filter Design Toolbox 2.0 since the Quantized Filtering Toolbox 1.0 was released in Release 11.1.

---

**Note** The Quantized Filtering Toolbox 1.1 has been integrated into the Filter Design Toolbox 2.0. The information below summarizes the changes to the Quantized Filtering Toolbox since Version 1.0 of that product. These changes are incorporated into the new Filter Design Toolbox 2.0.

---

You must have the Signal Processing Toolbox installed to use Filter Design Toolbox.

### New Quantization Tool for Advanced FIR and IIR Filter Design

Signal Processing Toolbox includes an interactive filter design tool called Filter Design and Analysis Tool (FDATool). You use this visual interface to design, convert, quantize, import, and export FIR and IIR filters. FDATool integrates filter design and analysis functions in a single tool, letting you do all your filter design tasks from within the tool instead of using the MATLAB command line. Filter Design Toolbox adds quantization analysis to FDATool.

When you install Filter Design Toolbox, FDATool operates in three modes:

- Filter Design for designing double-precision FIR and IIR filters.
- Import for importing existing filters from your MATLAB workspace or by entering an expression.
- Quantization for quantizing filters in FDATool. You can quantize any filter in FDATool, whether you designed or imported the filter.

To enable you to convert filters from one structure to another, for example, from direct form to coupled allpass, FDATool offers smart conversion tools. When you convert a filter, FDATool offers you a selection of conversion structures that depends on the current filter type.

To launch FDATool, enter `fdatool` at the MATLAB command prompt or use the Launch Pad. For more information on the new tool, refer to “Filter Design and Analysis Tool” in the *Signal Processing Toolbox User's Guide* or use Help to review the online documentation.

## New Objects

The Filter Design Toolbox 2.0 includes these new objects:

- Quantizer object

  Use the quantizer to quantize data using fixed-point or custom floating-point arithmetic. Use the `quantizer` function to construct a quantizer, and use the `quantize` function to quantize the data according to your quantizer specifications. There are also many functions that calculate the characteristics associated with a given quantizer. For example, `eps` determines the precision of the quantizer, and `realmax` determines the maximum number the quantizer can quantize without overflow.
- Quantized FFT object

  Use the new quantized FFT object to create and simulate fixed-point and custom floating-point FFTs. Construct a quantized FFT to specify the quantizing characteristics of the FFT method that you want to apply to data. Use the function `qfft` to construct a quantized FFT, and the function `fft` to apply an FFT to data as specified by your quantized FFT.

## New Filter Design Functions

The Filter Design Toolbox 2.0 includes these new filter design functions:

- `firlpnorm` – Design a least P-norm FIR filter
- `gremez` – Use generalized Remez techniques to design FIR filters
- `iirgrpdelay` – IIR filter design with prescribed group delay value
- `iirlpnorm` – Design a least P-norm IIR filter
- `iirlpnormc` – Design a constrained least P-norm IIR filter

## New Filter Conversion Functions

The Filter Design Toolbox 2.0 includes these new filter conversion functions:

- `ca2tf` – Convert coupled allpass transfer function forms to transfer function forms
- `cl2tf` – Convert coupled allpass lattice forms to transfer function forms
- `iirpowcomp` – Compute a power-complementary IIR filter from a given IIR filter
- `sos`: Convert the filter topology of a quantized filter to second-order sections with scaling
- `tf2ca` – Convert transfer function forms to coupled allpass transfer function forms
- `tf2cl` – Convert transfer function forms to coupled allpass lattice forms

- Quantized filter coefficient conversion

  You can now convert quantized filter coefficients to a hex or binary format, using these functions:

  - `num2hex` for hexadecimal conversion
  - `num2bin` for binary conversion

  These functions are overloaded for quantizers and quantized FFTs.

## New Filter Structures

Filter Design Toolbox 2.0 includes five new filter structures:

- Direct form FIR transposed
- Direct form antisymmetric FIR
- Lattice coupled-allpass
- Second-order sections
- Lattice coupled-allpass power-complementary

To learn more about these structures, refer to “Filter Structures” in the *Filter Design Toolbox User’s Guide*.

## New Analysis Methods

Filter Design Toolbox 2.0 includes two new analysis methods:

- Limit Cycle – The new function `limitcycle` uses Monte Carlo techniques to detect limit cycles in quantized filters.
- Noise Loading Method – Use function `nlm` to use the noise loading method to calculate the frequency response of a quantized filter. Compare the response to the results from `freqz`, which calculates the theoretical frequency response. To generate noise signals that contain complete frequency content across the spectrum, `nlm` uses a series of Monte Carlo trials.

## New Functions and Enhancements

Filter Design Toolbox 2.0 includes the following new functions and enhancements:

- Added `copyobj` function for copying quantized filters, quantized FFTs, and quantizers. Copies are independent of the original items, but have the same property values as the original.
- Changed the `CoefficientFormat` property to default to `Quantizer` rather than `Unitquantizer` for quantized filters, quantized FFTs, and quantizers.
- Added `NOperations`, a new data-related read-only property for quantizers that counts the number of data points quantized by a quantizer.
- Updated the quantized filter data display to indicate over- and underflow conditions in filter coefficients, as well as adding the `NOperations` read-only values to the displayed information.
- Changed the default rounding method for the `CoefficientFormat` property for quantized filters and FFTs to `'round'` rather than `'floor'`.
- Included new quantized filtering demos. Use the `demo` function to access the new demos.
- Modified the data format properties.
- You can now set the following properties individually for each data path in a quantizer, quantized filter, or quantized FFT:
  - `Mode`
  - `RoundMode`
  - `OverflowMode`
  - `OptimizeUnityGains`

- Removed the following properties:
  - `DisplayType`
  - `WarnIfOverflow`
- Allow cascades of all filter structures.

## Platform Limitations for HP 10.2

Filter Design Toolbox is not supported on the Hewlett-Packard 10.2 (HP 700) platform. For background information, see “Platform Limitations” on page 1-17.

## Upgrading from an Earlier Release

This section describes the upgrade issues involved in moving from Quantized Filtering Toolbox to Filter Design Toolbox 2.0.

You can freely upgrade from all versions of Quantized Filtering Toolbox, and Filter Design and Quantization Toolbox to Filter Design Toolbox 2.0.

### Obsolete Functions in Version 2.0

Filter Design Toolbox 2.0 makes obsolete the following functions that were part of Quantized Filter Design Toolbox:.

| Obsolete Function | Suggested Replacement |
|---|---|
| `propinfo` | Use `help constructor/propertyname` to get help about a function. Or use the Help browser. |
| `qfiltlog` | Use `qreport` to get information about quantized filters, quantized FFTs, and quantizers. |
| `qhelp` | Use `help constructor/propertyname` to get help about a function. Or use the Help browser. |

## Known Software and Documentation Problems

This section updates the Filter Design Toolbox 2.0 documentation, reflecting known Filter Design Toolbox 2.0 software or documentation problems.

### Filter Design and Analysis Tool

The quantization dialog in FDATool does not provide context-sensitive help in this release. To get help on a control in the dialog, use the Help browser. Select an option, such as **FDATool Help**, from the **Help** menu in FDATool. Or type `helpdesk` at the MATLAB prompt to open the online help system.

### Switching Between Design and Quantization Modes in FDATool

After you scale a quantized filter, or convert a quantized filter to second-order sections, do not switch between quantized mode and filter design mode in FDATool. Checking and clearing the **Turn quantization on** option (to shift between the quantization and design modes) after you have scaled or converted a quantized filter can corrupt the coefficients for your reference filter and your quantized filter.

To recover your reference filter design and quantized filter design if they have been corrupted, either:

- Import the filter from your workspace again
- Use FDATool to repeat the filter design

# Financial Derivatives Toolbox 1.0 Release Notes

**Introduction to the Financial Derivitives Toolbox** . . . 17-2
Interest Rate Models . . . . . . . . . . . . . . . . . . 17-2
Hedging . . . . . . . . . . . . . . . . . . . . . . . . 17-2
Financial Instruments . . . . . . . . . . . . . . . . . 17-2

## Introduction to the Financial Derivitives Toolbox

The Financial Derivatives Toolbox extends the Financial Toolbox in the areas of fixed income derivatives and of securities contingent upon interest rates. The toolbox provides components for analyzing individual financial derivative instruments and portfolios composed of them. Specifically, it provides the necessary functions for calculating prices and sensitivities, for hedging, and for visualizing results.

### Interest Rate Models

The Financial Derivatives Toolbox computes pricing and sensitivities of interest rate contingent claims based upon sets of zero coupon bonds or the Heath-Jarrow-Morton (HJM) evolution model of the interest rate term structure.

### Hedging

The Financial Derivatives Toolbox also includes hedging functionality, allowing the rebalancing of portfolios to reach target costs or target sensitivities, which may be set to zero for the case of a neutral-sensitivity portfolio. Optionally, the rebalancing process can be self-financing or directed by a set of user-supplied constraints.

### Financial Instruments

The toolbox provides a set of functions that perform computations upon portfolios containing up to seven types of financial instruments.

**Bond.** A long-term debt security with preset interest rate and maturity, by which the principal and interests must be paid.

**Bond Options.** Puts and calls on portfolios of bonds.

**Fixed Rate Note.** A long-term debt security with preset interest rate and maturity, by which the interests must be paid. The principal may or may not be paid at maturity. In this version of the Financial Derivatives Toolbox, the principal is always paid at maturity.

# Financial Time Series Toolbox 1.0 Release Notes

**Introduction to the Financial Time Series Toolbox** . . 18-2
Creating Financial Time Series Objects . . . . . . . . . . 18-2
Operations . . . . . . . . . . . . . . . . . . . . . . . . 18-2
Technical Analysis . . . . . . . . . . . . . . . . . . . . 18-2
Learning More About the Financial Time Series Toolbox . . 18-3

## Introduction to the Financial Time Series Toolbox

The Financial Time Series Toolbox for MATLAB is a collection of tools for the analysis of time series data in the financial markets. The toolbox contains a financial time series object constructor and several methods that operate on and analyze the object. Financial engineers working with time series data, such as equity prices or daily interest fluctuations, can use this toolbox for more intuitive data management than by using regular vectors or matrices.

### Creating Financial Time Series Objects

The Financial Time Series Toolbox provides two ways to create a financial time series object:

1. At the command line using the object constructor `fints`
2. From a text data file through the function `ascii2fts`

The structure of the object minimally consists of a description field, a frequency indicator field, the date vector, and a data series vector.

### Operations

Several MATLAB functions have been overloaded to work with financial time series objects. The overloaded functions include basic arithmetic functions such as addition, subtraction, multiplication, and division as well as other functions such as arithmetic average, filter, and difference. Also, specific methods have been designed to work with the financial time series object.

### Technical Analysis

The technical analysis functions in this toolbox are tools to help analyze your investments. Technical analysis (or charting) is used by some investment managers to help manage portfolios. Technical analysis relies heavily on the availability of historical data. Investment managers calculate different indicators from available data and plot them as charts. Observations of price, direction, and volume on the charts assist managers in making decisions on their investment portfolios.

## Learning More About the Financial Time Series Toolbox

The Financial Time Series Toolbox documentation describes the product in detail.

# Fixed-Point Blockset 3.0 Release Notes

**New Features** . . . . . . . . . . . . . . . . . . . . . . . 19-2
New Blocks . . . . . . . . . . . . . . . . . . . . . . . . . 19-2
New Functions . . . . . . . . . . . . . . . . . . . . . . . 19-3
Data Type Support . . . . . . . . . . . . . . . . . . . . . . 19-4
Data Type and Scaling Inheritance . . . . . . . . . . . . 19-4
Increased Speed, Efficiency, and Accuracy . . . . . . . . 19-5
Overflow Logging . . . . . . . . . . . . . . . . . . . . . 19-5
Simulink Accelerator . . . . . . . . . . . . . . . . . . . . 19-5
Model Parameter Configuration . . . . . . . . . . . . . . 19-5
Display of Scaling Information . . . . . . . . . . . . . . 19-5

## New Features

This section introduces the new features and enhancements added in the Fixed-Point Blockset 3.0 since the Fixed-Point Blockset 2.0 (Release 11.0).

For information about Fixed-Point Blockset 3.0 new features that are incorporated from prior releases, see *Release 11 New Features* (enhancements introduced between Release 9.0 and Release 11.0).

This section is organized into these subsections:

- "New Blocks" on page 19-2
- "New Functions" on page 19-3
- "Data Type Support" on page 19-4
- "Data Type and Scaling Inheritance" on page 19-4
- "Increased Speed, Efficiency, and Accuracy" on page 19-5
- "Overflow Logging" on page 19-5
- "Simulink Accelerator" on page 19-5
- "Model Parameter Configuration" on page 19-5
- "Display of Scaling Information" on page 19-5

### New Blocks

The Fixed-Point Blockset 3.0 includes the new blocks listed below.

| Block Name | Description |
|---|---|
| FixPt Absolute Value | Output the absolute value of the input. |
| FixPt Bitwise Operator | Perform the specified bitwise operation on the inputs. |
| FixPt Data Type Propagation | Configure the data type and scaling of the propagated signal based on information from the reference signals. |
| FixPt Dead Zone | Provide a region of zero output. |
| FixPt Dot Product | Generate the dot product. |

| Block Name | Description |
|---|---|
| FixPt Dynamic Look-Up Table | Approximate a one-dimensional function using a selected look-up method and a dynamically specified table. |
| FixPt Gateway In Inherited | Convert a Simulink data type to a Fixed-Point Blockset data type, and inherit the data type and scaling. |
| FixPt Integer Delay | Delay a signal N sample periods. |
| FixPt MinMax | Output the minimum or maximum input value. |
| FixPt Multiport Switch | Switch output between different inputs based on the value of the first input. |
| FixPt Sign | Indicate the sign of the input. |
| FixPt Tapped Delay | Delay a scalar signal multiple sample periods, and output all the delayed versions. |
| FixPt Unary Minus | Negate the input. |

## New Functions

The Fixed-Point Blockset 3.0 includes the new functions listed below.

| Function Name | Description |
|---|---|
| `fixptbestexp` | Determine the exponent that gives the best precision fixed-point representation of a value. |
| `fixptbestprec` | Determine the maximum precision available for the fixed-point representation of a value. |
| `fixpt_convert` | Convert Simulink models and subsystems to fixed-point equivalents. |

| Function Name | Description |
|---|---|
| `fixpt_convert_prep` | Prepare a Simulink model for more complete conversion to fixed point. |
| `fixpt_restore_links` | Restore links for fixed-point blocks. |

## Data Type Support

The Fixed-Point Blockset 3.0 has expanded its data type support:

- You can connect Simulink blocks directly to Fixed-Point Blockset blocks provided the signals use built-in Simulink data types. However, a fixed-point signal consisting of 8-, 16-, or 32-bit integers is compatible with Simulink only when its scaling is given by a slope of `1` and a bias of `0`.
- Many important Simulink blocks have added support for fixed-point signals. Key blocks include Enable, Trigger, Scope, Display, and To Workspace.
- Fixed-point blocks now support complex numbers.

## Data Type and Scaling Inheritance

Many fixed-point blocks support the inheritance of data type and scaling information from other blocks. The two possible inheritance options are:

- Inherit via internal rule – The output data type and scaling are inherited from the block input(s). The goal of the inheritance rule is to select the “natural” data type and scaling for the output. The specific rule that is used depends on the block operation. For example, if you are multiplying two signed 16-bit signals, the FixPt Product block produces the natural output of a signed 32-bit data type.
- Inherit via back propagation – The output data type and scaling are inherited by back propagation. In many cases, you will use the FixPt Data Type Propagation block with this option.

## Increased Speed, Efficiency, and Accuracy

- Simulation speed has been increased by approximately 2.5 times.
- On average, fixed-point signals use 20 times less memory.
- Interpolation is more accurate and the number of quantization steps has been reduced.
- Generated code is significantly simpler.
- Unsigned numbers avoid overflow in these situations:
  - Interpolation when the slope of the output data is negative
  - Conversion when the bias adjustment is negative
  - Addition and subtraction when the bias adjustment is negative

## Overflow Logging

Indirect logging of overflows and saturations via percent range error has been replaced by explicit logging.

## Simulink Accelerator

You can now use the Simulink Accelerator in conjunction with the Fixed-Point Blockset. However, you cannot log minimum and maximum simulation values or overflows.

## Model Parameter Configuration

You can now use the **Model Parameter Configuration** dialog box to override the **Inline parameters** option for selected parameters. You access this feature by selecting the **Configure** button on the **Advanced** tab of the **Simulation Parameters** dialog box. Previously, code generation for fixed point blocks required all parameters to be inline, or all parameters to be tunable.

## Display of Scaling Information

To significantly improve simulation speed, the scaling information associated with fixed-point blocks is no longer updated. Since this scaling information can become obsolete, you should remove it by running `fixpt_clear_tag` or `slupdate`. To view data type and scaling information, you should select **Port data types** under the model's **Format** menu.

# Fuzzy Logic Toolbox 2.1 Release Notes

**New Features** . . . . . . . . . . . . . . . . . . . . . . 20-2
Wizard For Fuzzy Logic Controller Block . . . . . . . . . . 20-2

**Known Software and Documentation Problems** . . . . 20-3
ODE Solver Performance . . . . . . . . . . . . . . . . . 20-3

## New Features

This section introduces the new features and enhancements added in the Fuzzy Logic Toolbox 2.1 since the Fuzzy Logic Toolbox 2.0.1 (Release 11.0).

For information about Fuzzy Logic Toolbox 2.1 enhancements that are incorporated from prior releases, see *Release 11 New Features* (enhancements introduced between Release 9.0 and Release 11.0)

### Wizard For Fuzzy Logic Controller Block

The Fuzzy Logic Toolbox supports a new wizard for the Fuzzy Logic Controller block. This block now automatically generates a customized block diagram representation for most Fuzzy Inference Systems (FISs). This block diagram uses only built-in Simulink blocks and allows for generation of efficient and compact code using Real-Time Workshop.

## Known Software and Documentation Problems

This section updates the Fuzzy Logic Toolbox 2.1 documentation set, reflecting known Fuzzy Logic Toolbox 2.1 software and documentation problems.

### ODE Solver Performance

Because most FIS contain Min, Max, and other blocks with zero-crossing detection, the default ODE solver can perform poorly in some cases. Two remedies are available:

- Choose a stiff solver. Several choices are available on the **Solver** page of the **Simulation Parameters** dialog box in Simulink.
- Disable zero crossing detection on the **Advanced** page of the **Simulation Parameters** dialog box

# GARCH Toolbox 1.0 Release Notes

**Introduction to the GARCH Toolbox** . . . . . . . . . . 21-2

## Introduction to the GARCH Toolbox

The GARCH Toolbox was introduced in Release 11.1.

MATLAB and the GARCH Toolbox provide an integrated computing environment for modeling the volatility of univariate economic time series. The GARCH Toolbox uses a general ARMAX/GARCH composite model to perform simulation, forecasting, and parameter estimation of univariate time series in the presence of conditional heteroskedasticity. Supporting functions perform tasks such as pre- and post-estimation diagnostic testing, hypothesis testing of residuals, model order selection, and time series transformations. Graphics capabilities let you plot correlation functions and visually compare matched innovations, volatility, and return series.

GARCH modeling provides accurate forecasts of variances and covariances of asset returns through its ability to model time-varying conditional variances. As a consequence, you can apply GARCH models to such diverse fields as risk management, portfolio management and asset allocation, option pricing, foreign exchange, and the term structure of interest rates.

Using the GARCH Toolbox, you can:

- Perform Monte Carlo simulation of univariate returns, innovations, and conditional volatilities
- Specify conditional mean models of general ARMAX form and conditional models of general GARCH form for univariate asset returns
- Estimate parameters of general ARMAX/GARCH composite models via the maximum likelihood method
- Generate minimum mean square error forecasts of the conditional mean and conditional variance of univariate return series
- Perform pre- and post-estimation diagnostic and hypothesis testing, such as Engle's ARCH test, Ljung-Box Q-statistic test, likelihood ratio tests, and AIC/BIC model order selection
- Perform graphical correlation analysis, including auto-correlation, cross-correlation, and partial auto-correlation
- Convert price/return series to return/price series, and transform finite-order ARMA models to infinite-order AR and MA models

# Instrument Control Toolbox 1.0 Release Notes

**Introduction to the Instrument Control Toolbox** . . . 22-2
Supported Platforms . . . . . . . . . . . . . . . . . . . 22-2
Object-Based Design . . . . . . . . . . . . . . . . . . . 22-2
Reading and Writing Data . . . . . . . . . . . . . . . . 22-3
Events and Actions . . . . . . . . . . . . . . . . . . . . 22-3
Data Recording . . . . . . . . . . . . . . . . . . . . . . 22-3
Graphical Tools . . . . . . . . . . . . . . . . . . . . . . 22-3

**Vendor Driver Requirements and Limitations** . . . . 22-4
Driver Requirements . . . . . . . . . . . . . . . . . . . 22-4
GPIB Driver Limitations . . . . . . . . . . . . . . . . . 22-5
VISA Driver Limitations . . . . . . . . . . . . . . . . . 22-7

**Known Software Problems** . . . . . . . . . . . . . . 22-9
Pin Status Event . . . . . . . . . . . . . . . . . . . . . 22-9
Available Serial Ports . . . . . . . . . . . . . . . . . . 22-9
Accessing Serial Ports on Solaris . . . . . . . . . . . . 22-10

## Introduction to the Instrument Control Toolbox

The Instrument Control Toolbox 1.0 is a new product from The MathWorks that provides a framework for communicating with instruments that support the GPIB interface (IEEE-488, HP-IB), the VISA standard, and the serial port interface (RS-232, RS-422, and RS-485).

**Note** The toolbox extends the serial port features included with MATLAB.

### Supported Platforms

The supported platforms and associated toolbox features are given below.

| Platform | Supported Toolbox Features |
|---|---|
| Microsoft Windows 95, Windows 98, Windows 2000, and Windows NT 4.0 | All features |
| Sun Solaris | National Instruments GPIB |
| | National Instruments VISA |
| | Serial port |
| Linux | Serial port |

### Object-Based Design

Based on MATLAB's object technology, the Instrument Control Toolbox provides functions for creating objects directly associated with your instrument.

You can create objects for instruments that support the GPIB interface, the VISA standard, and serial port interface. For example, the following command creates the GPIB object `g` associated with a National Instruments GPIB controller with board index 0, and an instrument with primary address 1.

```
g = gpib('ni',0,1);
```

`g` contains all the necessary parameters for communicating with the instrument.

## Reading and Writing Data

Functions are available for transferring data between MATLAB and your instrument:

- The data can be binary (numerical) or text.
- Text data can be any command used by your instrument such as a command given by the Standard Commands for Programmable Instruments (SCPI) language.
- The transfer can be synchronous and block the command line, or asynchronous and not block the command line.

## Events and Actions

You can enhance the power and flexibility of your instrument control application by using events. An event occurs after a condition is met and may result in one or more actions. For example, you can use events to display a message, display data, or analyze data.

You configure actions through action properties and action functions. All event types have an associated action property. Action functions are M-file functions that you construct to suit your specific application needs.

Event types supported by the Instrument Control Toolbox include error events, timer events, and bytes-available events.

## Data Recording

Functions are provided for recording instrument commands, data, and events to one or more disk files. Recording this information to disk provides a permanent record of your instrument control session, and is an easy way to debug your application.

## Graphical Tools

Tools are provided that facilitate instrument control in an easy-to-use graphical environment. Using these tools, you can:

- Create and configure an instrument object
- Communicate with your instrument via text commands

## Vendor Driver Requirements and Limitations

The requirements and limitations for the vendor GPIB and VISA drivers supported by the Instrument Control Toolbox are described below.

**Note** The limitations described in this section are restricted to the limitations directly associated with using the Instrument Control Toolbox.

### Driver Requirements

You can use the Instrument Control Toolbox with the GPIB and VISA drivers listed below.

| Interface | Vendor | Driver Requirements |
|---|---|---|
| GPIB | Agilent | SICL version H.012.02.00 |
| | ComputerBoards | GPIB-32 version 2.12 |
| | Capital Equipment Corporation | CEC-488 version 5.08 |
| | IOTech | IOTech version 1.0 |
| | Keithley | Keithley-488 version 5.08 |
| | National Instruments | NI-488 version 1.41.7649 |
| VISA | Agilent | Agilent-VISA version 1.1 |
| | National Instruments | NI-VISA version 2.0 |

See the following sections for a description of:

- “GPIB Driver Limitations” on page 22-5
- “VISA Driver Limitations” on page 22-7

## GPIB Driver Limitations

### Agilent Technologies

The Agilent GPIB driver has these limitations:

- Asynchronous read and write operations are not supported. Therefore, Agilent GPIB objects do not support the following toolbox functionality:
  - The `readasync` function
  - The `async` flag for the `fprintf` and `fwrite` functions
  - `BytesAvailableAction` and `OutputEmptyAction` properties
- The End Or Identify (EOI) line is not asserted when the End-Of-String (EOS) character is written to the hardware. Therefore, when the `EOSMode` property is configured to `write` and the `EOIMode` property is configured to `on`, the EOI line is not asserted when the `EOSCharCode` property value is written to the hardware.
- All eight bits are used for the EOS comparison. Therefore, the only value supported by the `CompareBits` property is `8`.
- A board index value of `0` is not supported.
- An error is not reported for an invalid primary address. Instead, the read and write operations will time out.

### ComputerBoards

The ComputerBoards GPIB driver does not support asynchronous notification for the completion of read and write operations. Therefore, ComputerBoards GPIB objects do not support the following toolbox functionality:

- The `readasync` function
- The `async` flag for the `fprintf` and `fwrite` functions
- The `BytesAvailableAction` and `OutputEmptyAction` properties

### Capital Equipment Corporation

The Capital Equipment Corporation (CEC) GPIB driver has these limitations:

- Asynchronous read and write operations are not supported. Therefore, CEC GPIB objects do not support the following toolbox functionality:
  - The `readasync` function
  - The `async` flag for the `fprintf` and `fwrite` functions
  - The `BytesAvailableAction` and `OutputEmptyAction` properties
- The Handshake and Bus Management line values are not provided. The `BusManagementStatus` and `HandshakeStatus` properties always return the line values as `on`.
- The EOI line is not asserted when the EOS character is written to the hardware. Therefore, when the `EOSMode` property is configured to `write` and the `EOIMode` property is configured to `on`, the EOI line is not asserted when the `EOSCharCode` property value is written to the hardware.
- All eight bits are used for the EOS comparison. Therefore, the only value supported by the `CompareBits` property is `8`.
- You should not simultaneously use a GPIB controller address of 0 and an instrument primary address of 0.

### IOTech

The IOTech GPIB driver has these limitations:

- Asynchronous read and write operations are not supported. Therefore, IOTech GPIB objects do not support the following toolbox functionality:
  - The `readasync` function
  - The `async` flag for the `fprintf` and `fwrite` functions
  - The `BytesAvailableAction` and `OutputEmptyAction` properties.
- Incorrect values are returned for the REN and IFC bus management lines. The `BusManagementStatus` property always returns a value of `on` for the `RemoteEnable` and the `InterfaceClear` fields.
- The EOI line is not asserted when the EOS character is written to the hardware. Therefore, when the `EOSMode` property is configured to `write` and the `EOIMode` property is configured to `on`, the EOI line will not be asserted when the `EOSCharCode` property value is written to the hardware.

### Keithley

The Keithley GPIB driver has these limitations:

- Asynchronous read and write operations are not supported. Therefore, Keithley GPIB objects do not support the following toolbox functionality:
  - The `readasync` function
  - The `async` flag for the `fprintf` and `fwrite` functions
  - The `BytesAvailableAction` and `OutputEmptyAction` properties
- The Handshake and Bus Management line values are not provided. The `BusManagementStatus` and `HandshakeStatus` properties always return the line value as `on`.
- The EOI line is not asserted when the EOS character is written to the hardware. Therefore, when the `EOSMode` property is configured to `write` and the `EOIMode` property is configured to `on`, the EOI line will not be asserted when the `EOSCharCode` property value is written to the hardware.
- All eight bits are used for the EOS comparison. Therefore, the only value supported by the `CompareBits` property is `8`.
- You should not simultaneously use a GPIB controller address of 0 and an instrument primary address of 0.

## VISA Driver Limitations

### Agilent Technologies

The Agilent VISA driver has these known limitations:

- Asynchronous read and write operations for GPIB, GPIB-VXI and VXI VISA objects are not supported. Therefore, the following toolbox functionality is not supported for these objects:
  - The `readasync` function
  - The `async` flag for the `fprintf` and `fwrite` functions
  - The `BytesAvailableAction` and `OutputEmptyAction` properties
- All eight bits are used for the EOS comparison. Therefore, the only value supported by the `CompareBits` property is `8`.

### National Instruments

The National Instruments VISA driver uses all eight bits for the EOS comparison. Therefore, the only value supported by the `CompareBits` property is `8`.

## Known Software Problems

This section updates the Instrument Control Toolbox documentation, reflecting known software problems.

### Pin Status Event

On the Microsoft Windows 95 and Windows 98 platforms, a pin-status event is not produced when the ring indicator pin changes value.

### Available Serial Ports

#### Windows 95 and Windows 98

On Windows 95 and Windows 98 platforms, you can use only the COM1 through COM4 ports. Additionally, the `instrhwinfo` function always indicates that these ports exist. To find out the actual serial ports that are available, examine the `AvailableSerialPorts` field.

```
out = instrhwinfo('serial');
out.AvailableSerialPorts
out =
    'COM1'
    'COM2'
```

#### Solaris

On Solaris, all serial ports are returned by `instrhwinfo` – including ports that you cannot access. For example, `instrhwinfo` may return the following ports.

```
out = instrhwinfo('serial')
out.AvailableSerialPorts
ans =
'/dev/term/a'
'/dev/term/b'
'/dev/cua/a'
'/dev/cua/b'
```

However, only `/dev/term/a` and `/dev/term/b` are available for communicating with instruments.

## Accessing Serial Ports on Solaris

If you repeatedly open and close one or more serial ports on Solaris, MATLAB will become unresponsive. To minimize the chance of encountering this problem, you should:

- Use only one serial port on your platform.
- Connect the serial port object to your instrument once per MATLAB session.

Alternatively, if you have NI-VISA installed, you can completely avoid this problem by using a VISA-serial object.

# Image Processing Toolbox 2.2.2 Release Notes

**New Features** . . . . . . . . . . . . . . . . . . . . . . 23-2
New Demo . . . . . . . . . . . . . . . . . . . . . . . . 23-2
Support For Function Handles . . . . . . . . . . . . . . . 23-2
Documentation Enhanced . . . . . . . . . . . . . . . . . . 23-2

**Major Bug Fixes** . . . . . . . . . . . . . . . . . . . . . 23-3
imshow Fixes . . . . . . . . . . . . . . . . . . . . . . . 23-3
bwlabel Segmentation Violation Eliminated . . . . . . . . . 23-3
dilate And erode Return Correct Answers . . . . . . . . . . 23-3
freqz2 Fixes . . . . . . . . . . . . . . . . . . . . . . . 23-3
fspecial Function's 'LoG' Option . . . . . . . . . . . . . . 23-3
Improved Display for imcrop, improfile, and roipoly . . . . 23-4

## New Features

The focus of the Image Processing Toolbox 2.2.2 is on bug fixes (see “Major Bug Fixes” below).

A number of important new features will be released in the Image Processing Toolbox 3.0, which will be made available in a Web downloadable version after Release 12.0 is released.

### New Demo

The Image Processing Toolbox 2.2.2 includes the new `landsatdemo` function, which is a demo that illustrates how to construct color composite images from multispectral Landsat data.

### Support For Function Handles

The following functions have been updated to support function handles, a new MATLAB 6.0 language feature:

- `blkproc`
- `colfilt`
- `nlfilter`
- `qtdecomp`
- `roifilt2`

The MATLAB language has a new data type called the function handle. The function handle captures all the information about a function that MATLAB needs to evaluate it. You can pass a function handle in an argument list to other functions. See “Function Handles” on page 2-22 of these *Release Notes* for further information about function handles.

### Documentation Enhanced

The online *Image Processing Toolbox User's Guide* was enhanced for Release 12 by adding a “Getting Started” section, and by adding glossaries of relevant terms at the beginning of several chapters.

## Major Bug Fixes

The Image Processing Toolbox 2.2.2 includes several important bug fixes that were made in the Image Processing Toolbox 2.2.1 (Release 11.1). This section describes the bugs and how they have been fixed.

### imshow Fixes

You can now display the same image twice using `imshow`, without the previous problem of having the images appear to move slightly the second time.

Also, you can now use the syntax `imshow(I,[])` when all the elements of `I` are the same. Now `imshow` displays I using an intermediate shade of gray. Previously, `imshow` would generate an error for this case. (This fix was introduced in the Image Processing Toolbox 2.2.1 (Release 11.1).)

### bwlabel Segmentation Violation Eliminated

You can now pass a matrix to `bwlabel` that contains values other than 0 or 1. `bwlabel` treats any nonzero element as an object element. Previously, `bwlabel` would cause a segmentation violation for this case. (This fix was introduced in the Image Processing Toolbox 2.2.1 (Release 11.1).)

### dilate And erode Return Correct Answers

The `dilate` and `erode` functions now return the correct answer in all cases. In prior versions of the Image Processing Toolbox, in some cases these functions returned the incorrect answer if you specified the frequency-domain option with a structuring element that contained more than 255 elements.

### freqz2 Fixes

The `freqz2` function now returns correct values for the frequency scaling. Also, freqz2 no longer uses an excessive amount of memory.

### fspecial Function's 'LoG' Option

The `Log` option of the `fspecial` function now returns correctly scaled values.

## Improved Display for imcrop, improfile, and roipoly

The animated lines that the `imcrop`, `improfile`, and `roipoly` functions display on top of images are now displayed clearly.

# Mapping Toolbox 1.2 Release Notes

**New Features** . . . . . . . . . . . . . . . . . . . . . . 24-2
Higher Resolution Atlas Data . . . . . . . . . . . . . . . 24-2
External Data Interface Improved . . . . . . . . . . . . . 24-2
New Interactive Interfaces . . . . . . . . . . . . . . . . . 24-3
New Analysis Functions for Geographic Data . . . . . . . 24-3
Other New Functions . . . . . . . . . . . . . . . . . . 24-4

## New Features

This section introduces the new features and enhancements added in the Mapping Toolbox 1.2 since the Mapping Toolbox 1.1 (Release 11.0).

For information about Mapping Toolbox 1.2 new features that are incorporated from prior releases, see the *Release 11 New Features* (enhancements introduced between Release 9.0 and Release 11.0)

### Higher Resolution Atlas Data

There are now high-resolution country outlines and more city locations available through the `worldhi` database. The `worldmap` command automatically chooses this high-resolution data if the region's area is small enough.

The `worldlo` atlas file has been updated to make it coincide more closely with high-resolution coastlines and boundaries.

The `worldmtxmed` MAT-file provides a medium-resolution political world matrix map.

### External Data Interface Improved

Importing high-resolution atlas data is now much easier with these two visual interfaces:

- Digital Elevation Map Data user interface (invoked with the `demdataui` function)
- Vector Map Level 0 user interface (invoked with the `vmap0ui` function)

Many of the matrix map data interface functions now automatically concatenate data across separate files.

The external interface now supports the GLOBE digital elevation map data, a product similar to GTPO30. Use the `globedem` function for working with that data.

## New Interactive Interfaces

You can now adjust interactively, on a map display, geographic lines such as great circle tracks, small circles and sectors of small circles. Use the `trackg`, `scircleg`, and `sectorg` functions to make these interactive adjustments. While in an edit mode, you can drag the lines around on the map, modify the lines in a control panel, or read measurements.

There is a new visual interface to create colormaps. Use the `cmapui` function to invoke this new interface.

## New Analysis Functions for Geographic Data

You can use the new `elevation` function to find the elevation angle of a geographic point.

The new `gradientm` function performs matrix map data calculations, including gradient, slope, and aspect.

You can use the new `los2` and `viewshed` functions with terrain data to check the line of sight visibility between points or the visibility of entire regions.

Several new functions have been added to support polygon operations.

| Function | Description |
|---|---|
| `bufferm` | Compute polygon buffer regions |
| `polybool` | Perform polygon Boolean operations |
| `polyjoin` | Combine polygon segments into a NaN-clipped polygon |
| `polymerge` | Merge polygon segments with abutting ends |
| `polysplit` | Separate NaN-clipped polygon segments into cell arrays |
| `polyxpoly` | Polygon intersections |

## Other New Functions

Several new functions have been added to support polygon operations.

| Function | Description |
|---|---|
| `contourcmap` | Contour-like color jumps on surface objects |
| `driftcorr` | Compute correction for drift |
| `driftvel` | Compute drift speed and direction |
| `flatearthpoly` | Cut polygons at dateline |
| `lcolorbar` | Labeled colorbar |
| `mapprofile` | Matrix map values along a path |
| `str2angle` | String conversion to angle |

# MATLAB C/C++ Graphics Library 2.1 Release Notes

**New Features** . . . . . . . . . . . . . . . . . . . . . . 25-2
Additions . . . . . . . . . . . . . . . . . . . . . . . . . 25-2
Unsupported MATLAB Features . . . . . . . . . . . . . 25-2

## New Features

This section introduces the new features and enhancements added in the MATLAB C/C++ Graphics Library 2.1 since the MATLAB C/C++ Graphics Library 1.0.1 (Release 11.1).

The MATLAB C/C++ Graphics Library is a collection of MATLAB graphics routines distributed as a single library. The graphics library makes the visualization and GUI-building routines of MATLAB available to stand-alone C and C++ applications. A stand-alone C or C++ application is an executable program that can run independently of the MATLAB interpreted environment. Stand-alone applications are a convenient way to package and distribute a customized MATLAB application. In addition, you may freely distribute applications you develop with the MATLAB C/C++ Graphics Library.

Using this library, in conjunction with the MATLAB Compiler and the MATLAB C/C++ Math Library, you can create stand-alone applications from M-files that use lines, text, meshes, and polygons as well as interactive visual interface components such as menus, push buttons, and dialog boxes.

**Note** You must use the MATLAB Compiler to create C or C++ stand-alone graphics applications. Calling MATLAB C/C++ Graphics Library routines directly from a C or C++ source module is not supported.

### Additions

Version 2.1 of the C/C++ Graphics Library provides the following new features:

- Support for the `eval` function for expressions that do not contain variables
- Support for the `input` function with the same restrictions as `eval`

In addition, the MATLAB C/C++ Graphics Library now includes the MATLAB Math and Graphics Run-Time Library Installer. This installer prepackages all the necessary MATLAB runtime libraries into a single, self-extracting archive file. Developers of MATLAB applications who distribute their applications need to include these libraries in their distribution packages. The installer makes it easier because they only need to include the archive file in the application packages, rather than having to include each math and graphics library individually.

The installer also makes it easier for customers who receive these applications to extract and uncompress the libraries and install them.

## Unsupported MATLAB Features

The C/C++ Graphics Library does *not* support:

- MATLAB objects
- `plotedit` command
- Ghostscript drivers

# MATLAB C/C++ Math Library 2.1 Release Notes

**New Features** . . . . . . . . . . . . . . . . . . . . . . 26-2
New Features . . . . . . . . . . . . . . . . . . . . . . . 26-2
Unsupported MATLAB Features . . . . . . . . . . . . . . 26-2
Documentation Error . . . . . . . . . . . . . . . . . . . 26-3

**Upgrading from an Earlier Release** . . . . . . . . . . 26-4
Recompile Your Programs . . . . . . . . . . . . . . . . 26-4
Changed Features . . . . . . . . . . . . . . . . . . . . 26-4
For More Information . . . . . . . . . . . . . . . . . . 26-5

## New Features

This section introduces the new features and enhancements added in the MATLAB C/C++ Math Library 2.1 since the MATLAB C/C++ Math Library 2.0 (Release 11).

For information about MATLAB C/C++ Math Library 2.1 new features that are incorporated from prior releases, see *Release 11 New Features* (enhancements introduced between Release 9.0 and Release 11.0).

### New Features

Version 2.1 of the MATLAB C/C++ Math Library provides the following new features:

- Support for the `eval` function for expressions that do not contain variables
- Support for the `input` function with the same restrictions as `eval`
- Performance enhancements in the core numerical routines

#### MATLAB Math and Graphics Run-Time Library Installer

The MATLAB C/C++ Math Library now includes the MATLAB Math and Graphics Run-Time Library Installer. This installer prepackages all the necessary MATLAB runtime libraries into a single, self-extracting archive file. Developers of MATLAB applications who distribute their applications need to include these libraries in their distribution packages. The installer makes it easier because they only need to include the archive file in the application packages, rather than having to include each math and graphics library individually. The installer also makes it easier for customers who receive these applications to extract and uncompress the libraries and install them.

### Unsupported MATLAB Features

The C/C++ Math Library does not include any Handle Graphics or Simulink functions. For information about compiling an application that uses graphics functions, see the *MATLAB C/C++ Graphics Library User's Guide*.

In addition, the C/C++ Math Library does *not* support MATLAB objects and MATLAB Java objects.

## Documentation Error

In the section “Creating Structures” in the Using the C Math Library documentation, the examples use a routine named `mlfDestroyArray`. The name of this routine should be `mxDestroyArray`.

## Upgrading from an Earlier Release

### Recompile Your Programs

Existing C/C++ Math Library Version 2.0 hand-written source code is compatible with the Version 2.1 library, but you must recompile your code. Additionally, any M-files which were compiled with Version 2.0 of the MATLAB Compiler must be recompiled with Version 2.1 of the MATLAB Compiler before using them with the Version 2.1 library. If you do not recompile your program, it will produce a run-time error.

### Changed Features

#### Empty Arrays

In Version 2.0 of the C++ Math Library, any empty array could be used as an indexed deletion operator. In Version 2.1, you must use the `empty()` function for indexed deletion.

#### mbuild Function Now Returns Accurate Error Status

The `mbuild` syntax

```
mbuild myprog.c
```

now throws an error when it encounters an error condition.

The `mex` syntax

```
stat = mbuild('myprog.c')
```

now returns a nonzero value to `stat` when it encounters an error condition.

In the past, on Microsoft Windows platforms, `mbuild` always either successfully exited or returned zero (indicating success), regardless of whether an error had actually occurred.

To ensure code from before Release 12 works properly in Release 12, either use `try/catch` logic to deal with error conditions, or use a form of `mbuild` that returns an error status instead of throwing an error. Specifically

```
try
  mbuild something.c
catch
```

```
    disp('something failed');
  end
```

or

```
  status = mbuild('something.c');
  if status ~= 0
    disp('something failed');
  end
```

## For More Information

For information about upgrading from a release earlier than 2.0.1, see *Release 11 New Features*.

# MATLAB Compiler 2.1 Release Notes

**New Features** . . . . . . . . . . . . . . . . . . . . . 27-2
Optimizations . . . . . . . . . . . . . . . . . . . . . . 27-2
Dynamically Linking in MEX-Files in the
Stand-Alone Environment . . . . . . . . . . . . . . 27-3
MATLAB Add-In for Visual Studio® . . . . . . . . . . . 27-4
mlib Files . . . . . . . . . . . . . . . . . . . . . . . 27-4
Additional Datatype Support . . . . . . . . . . . . . . 27-4
Improved Support for load and save . . . . . . . . . . . 27-5
Faster C/C++ Math Library Applications . . . . . . . . . 27-5
Printing from the C/C++ Graphics Library . . . . . . . . 27-5
Additional Language Support . . . . . . . . . . . . . . 27-5

**Third Party Compilers** . . . . . . . . . . . . . . . . . 27-6
Exception Handling . . . . . . . . . . . . . . . . . . . 27-6

**Upgrading from an Earlier Release** . . . . . . . . . . 27-8
Changed Options . . . . . . . . . . . . . . . . . . . . 27-8
Obsolete Options . . . . . . . . . . . . . . . . . . . . 27-8

## New Features

This section introduces the new features added in the MATLAB Compiler 2.1 since the MATLAB Compiler 2.0.1 (Release 11.0).

For information about MATLAB Compiler 2.1 new features that are incorporated from prior releases, see *Release 11 New Features* (enhancements introduced between Release 9.0 and Release 11.0).

**Note** The options for Compiler 2.1 are different from prior versions. Use

```
mcc -?
```

for help on the current set of options.

### Optimizations

The MATLAB Compiler 2.1 supports several types of optimizations. The format of the optimization option is

```
-O <optimization>
```

There are three possibilities.

```
-O <optimization class>:[on|off]
```

```
-O <list>
```

```
-O <optimization level>
```

The first form

```
-O <optimization class>:[on|off]
```

turns the optimization class on or off. This table describes the possibilities.

| Optimization Class | Description |
|---|---|
| `fold_scalar_mxarrays` | Fold scalar valued array constants |
| `fold_non_scalar_mxarrays` | Fold nonscalar valued array constants |

| Optimization Class | Description |
|---|---|
| optimize_integer_for_loops | Optimize for-loops with integer starts and increments |
| optimize_conditionals | Optimize conditional expressions when both operands are integers |
| array_indexing | Optimize simple one- and two-dimensional array index expressions |

The second form of the optimization option

```
-O <list>
```

lists the available optimization classes.

The third form

```
-O <optimization level>
```

uses a bundle file called `opt_bundle_<level>` to determine which optimizations are on or off. For example,

```
-O all
```

looks for a bundle file called `opt_bundle_all` and uses the options in the bundle file. Bundles for `-O all` and `-O none` are provided by default.

## Dynamically Linking in MEX-Files in the Stand-Alone Environment

The MATLAB Compiler 2.1 supports dynamically linking in MEX-files in the stand-alone environment. Specifying `-h` automatically compiles in any referenced MEX-files. Specifying MEX-files on the command line is supported and will work the same way as compiling an M-file.

The only restriction is that you cannot run a Compiler-generated MEX-file from stand-alone code. This restriction is minimal because you can include the M-file source directly, which is preferred by the Compiler.

## MATLAB Add-In for Visual Studio®

The MATLAB add-in for the Visual Studio development environment integrates the MATLAB Compiler 2.1 into Visual C/C++ 5 or 6. Running `mbuild -setup` or `mex -setup` automatically adds the Project Wizard into Microsoft Visual Studio so that you can use the Visual Studio environment to compile and run M-files. See the *MATLAB Compiler 2.1 User's Guide* for more information on configuration.

## mlib Files

The introduction of `mlib` files makes it possible to produce a shared library out of a toolbox and then compile M-files that make calls into that toolbox.

When compiling a collection of files (from a toolbox, for example) into a library, the Compiler creates a separate file with the M interface descriptions of the various M-functions that are available in the library. You can then use this file to compile other functions that make calls into the M-files in the library without recompiling the M-files. The extension of these library description files is `.mlib`.

Prior to this release, the Compiler would pass any file that it did not recognize on the command line as an M-file to `mbuild`. In this release, the Compiler uses files with the `.mlib` extension to allow compilation against a library of compiled M-files. See the *MATLAB Compiler 2.1 User's Guide* for more information.

## Additional Datatype Support

**Integer Data Types.** The MATLAB Compiler 2.1 supports the integer datatypes.

| | |
|---|---|
| `int8` | `uint8` |
| `int16` | `uint16` |
| `int32` | `uint32` |

**Function Handles.** A function handle is a new MATLAB data type that captures all the information about a function that MATLAB needs to execute, or evaluate, it. The MATLAB Compiler 2.1 supports function handles. For more information on function handles, see the `function_handle` reference page.

## Improved Support for load and save

The `load` and `save` commands are supported when they do not list the variables to be loaded or saved.

## Faster C/C++ Math Library Applications

The performance of the C/C++ Math Library has been improved considerably. Scalar accelerated versions of many library functions have been added, and improvements have been made to the overall performance of all library applications.

## Printing from the C/C++ Graphics Library

The MATLAB C/C++ Graphics Library now supports printing.

## Additional Language Support

**pause and continue.** The MATLAB Compiler 2.1 supports the MATLAB commands `pause` and `continue`.

**eval and input.** `eval` and `input` are supported for strings that do not contain workspace variables.

---

**Note** The MATLAB Compiler 2.1 does *not* support user-defined classes (MATLAB objects), scripts, or calls to the MATLAB Java interface.

---

## Third Party Compilers

This version of the MATLAB Compiler supports the following PC compilers as they are shipped:

- MSVC C/C++ 5.0, and 6.0
- Borland C++ 5.0, 5.02
- Borland C++Builder 3, 4, 5
- Borland C++ 5.5 (free command line tools)
- LCC (C only compiler, bundled with MATLAB)

---

**Note** When installing MSVC 6.0, if you need to change where this compiler is installed, you must change the location of the `Common` directory (at the appropriate installer dialog.) If you change the location of the `VC98` directory from its default setting, the `MEX` and `MBUILD` scripts will not work properly.

---

### Exception Handling

When using C++, the MATLAB Compiler relies on the availability of exception handling in the C++ language. Several of the supported compilers do not properly support C++ exception handling. Consequently, our support for exception handling is limited on those platforms.

#### GNU C++

GNU C++ 2.7.2 does not support C++ exception handling.

#### Borland

Borland C++ (all versions) has a restriction on support of `goto` statements within `try/catch` blocks. The MATLAB Compiler sometimes generates `goto` statements for complicated `if` conditional statements. The generated code for these will not compile in Borland C++; as a result you will be required to simplify the `if` condition.

**Note** The `-A debugline` option is implemented using `try`/`catch` statements. Therefore, use of this option is also restricted in the above C++ implementation.

## Upgrading from an Earlier Release

This section describes the upgrade issues involved in moving from the MATLAB Compiler 2.0.1 to the MATLAB Compiler 2.1.

For information about upgrading from a release earlier than 2.0.1, see *Release 11 New Features*.

### Changed Options

Some of the options for Compiler 2.1 are different from prior versions. Use

```
mcc -?
```

for help on the current set of options.

### Obsolete Options

As of this release, the MATLAB Compiler 1.2 is no longer available and is not supported due to the evolution of internal data structures in MATLAB. Consequently, the `-V1.2` option is no longer supported, along with any and all options recognized by the MATLAB Compiler 1.2.

# MATLAB Report Generator 1.1 Release Notes

**New Features** . . . . . . . . . . . . . . . . . . . . . . 28-2
New Components . . . . . . . . . . . . . . . . . . . . . 28-2
Enhancements to Components . . . . . . . . . . . . . . . 28-2
Property Table Enhancement . . . . . . . . . . . . . . . . 28-3
Summary Table Enhancement . . . . . . . . . . . . . . . 28-3
Using MATLAB Variables with Components . . . . . . . . 28-3

**Upgrading from an Earlier Release** . . . . . . . . . . 28-4
Use of Existing SGML Files . . . . . . . . . . . . . . . . 28-4

## New Features

This section introduces the new features and enhancements added in the MATLAB Report Generator 1.1 since the MATLAB Report Generator 1.0 (Release 11.0).

**Note** The Simulink Report Generator extends the MATLAB Report Generator.

### New Components

The MATLAB Report Generator 1.1 includes the following three new components.

| Component | Purpose |
|---|---|
| Comment | Inserts a comment into the SGML source file created by the report generation process |
| MATLAB/Toolbox Version Number | Creates and inserts a table that shows the version number, release number, and/or release date of any MathWorks software you are currently using |
| Stop Report Generation | Halts report generation |

### Enhancements to Components

The following components have been enhanced for Release 12:

- You can now put an image on a title page. See the Title Page component reference section for details.
- The Nest Setup File component now has a recursion limit.
- The For Loop component now allows a vector of indices.

## Property Table Enhancement

For all property tables, you can now use a context menu to add rows above or below the current cell and add columns to the left or right of the current cell.

## Summary Table Enhancement

For all summary tables, you can now use embedded looping components for table item generation.

## Using MATLAB Variables with Components

You can enter `%<VariableName>` in any component field where the text appears blue to include the value of a variable from the MATLAB workspace. You cannot enter more than one variable in the `%<>`. If you enter an invalid variable name, then the report will contain `%<VariableName>` instead of the value of the variable.

This `%<VariableName>` syntax is supported for the following components:

- Image
- Link
- Report Options
- Text
- Variable Table

### Example

If you enter the following

```
I have a %<ObjName> and it has %<NumLeaves> leaves. The word
'%<ObjName>' has %<size(ObjName)> letters.
```

and if `ObjName='plant'` and `NumLeaves=3` in the MATLAB workspace,

then the report will contain the following.

```
I have a plant and it has 3 leaves. The word 'plant' has 5 letters.
```

## Upgrading from an Earlier Release

This section describes the upgrade issues involved in moving from the MATLAB Report Generator 1.0 to the MATLAB Report Generator 1.1.

### Use of Existing SGML Files

If you generated a report using the MATLAB Report Generator 1.0 and saved the report file as an SGML file, and then you try to convert that SGML file using the MATLAB Report Generator 1.1 `rptconvert` program, the conversion fails.

In such cases, you need to regenerate the SGML file using the MATLAB Report Generator 1.1.

# MATLAB Runtime Server 6.0 Release Notes

**New Features** . . . . . . . . . . . . . . . . . . . . . 29-2

**Upgrading from an Earlier Release** . . . . . . . . . . 29-3

**Known Software and Documentation Problems** . . . . 29-4

## New Features

This section introduces the new features and enhancements added in the MATLAB Runtime Server 6.0 since the MATLAB Runtime Server 5.0 (Release 11.0).

The MATLAB Runtime Server now includes these new features:

- A new tool, `buildp`, for generating runtime P-code.
- An enhanced implementation of `depfun`, now part of MATLAB. Its output now includes information about Java classes.
- A new example of a MATLAB runtime GUI application. This example has also been adapted for use as a MATLAB runtime engine application on a PC, using ActiveX Automation. For details about this adaptation, see *matlabroot*`\toolbox\runtime\examples\activex\Readme`.

In addition, the `package` script and the other examples have been updated to make them compatible with MATLAB 6.0.

## Upgrading from an Earlier Release

If you are upgrading a runtime application that you built using the MATLAB Runtime Server 5.0 so that it uses the MATLAB Runtime Server 6.0, then you need to repeat the process of compiling, packaging, and testing the application to ensure that it works properly with MATLAB 6.0. Refer to the *MATLAB Runtime Server Application Developer's Guide* for details.

## Known Software and Documentation Problems

This section updates the MATLAB Runtime Server 6.0 documentation set, reflecting known documentation problems. These errors occur in the printed, but not the electronic, version of the *MATLAB Runtime Server Application Developer's Guide*:

- The list of files in “Manually Packaging Files for Shipping (PC)” (pages 4-6 and 4-7) is incorrect. Use the electronic version instead.
- If your application uses Java classes, then the discussion about packaging a runtime application (pages 4-3 through 4-7) is incomplete. Use the electronic version of the instructions for automatically or manually packaging your application.
- Pages 3-10 and 3-11 incorrectly refer to the directories `toolbox\runtime\examples\engine\activex` and `toolbox\runtime\examples\engine`, respectively. Both should instead refer to the directory `toolbox\runtime\examples\activex`, which is the actual location of the files for the ActiveX Automation example.
- The electronic version of the “Testing with the Runtime Server Variant” section for runtime engine applications recommends a more robust method, compared to the corresponding section (page 3-8) of the printed version. Also, see the updated procedure for testing the example runtime engine application, and compare with page 3-28 of the printed version.

# MATLAB Web Server 1.2 Release Notes

**New Features** . . . . . . . . . . . . . . . . . . . . . . . . 30-2
Support for Linux Servers . . . . . . . . . . . . . . . . . . 30-2
New Security Feature . . . . . . . . . . . . . . . . . . . . . 30-2

## New Features

This section introduces the new features and enhancements added in the MATLAB Web Server 1.2 since the MATLAB Web Server 1.0 (Release 11.0).

### Support for Linux Servers

The MATLAB Web Server is available on UNIX (Solaris) workstations and IBM PC compatible computers running Microsoft Windows NT or Linux.

### New Security Feature

The MATLAB Web Server now supports the creation of `hosts.conf`, an optional file providing additional security. If `hosts.conf` is present, only listed machines can connect to the MATLAB Web Server.

# Motorola DSP Developer’s Kit 1.1 Release Notes

**Introduction to the Motorola DSP Developer’s Kit** . . . 31-2
For More Information . . . . . . . . . . . . . . . . . . 31-2

**New Features** . . . . . . . . . . . . . . . . . . . . 31-4
Matrix and Frame Support . . . . . . . . . . . . . . . 31-4

**Known Software and Documentation Problems** . . . . 31-5
Interactive Mode Limitations . . . . . . . . . . . . . . 31-5
Stand-Alone GUI Simulator May Hang
(Windows NT) . . . . . . . . . . . . . . . . . . 31-6
Other Solaris-Specific Limitations . . . . . . . . . . . . 31-6
Documentation Updates . . . . . . . . . . . . . . . . . 31-7

## Introduction to the Motorola DSP Developer’s Kit

**Note** The Motorola DSP Developer’s Kit 1.0 was introduced as a Web-downloadable product after the release of Release 11.1. The Motorola DSP Developer’s Kit 1.1 adds matrix and frame support (see “New Features” on page 31-4).

You can use the Motorola DSP Developer’s Kit 1.0 to develop application software for Motorola DSPs in the MathWorks MATLAB and Simulink environments. The Motorola DSP Developer’s Kit provides an object-oriented interface to program MEX-files or S-functions that call the appropriate Motorola Suite56 DSP Simulator.

With the Motorola DSP Developer’s Kit, you can develop implementation solutions based on the Motorola Suite56 DSP families. You achieve this by implementing algorithms in Motorola DSP assembly language and running the generated object code directly from within MATLAB or Simulink, on the chosen Motorola Suite56 DSP simulator.

The Motorola DSP Developer’s Kit also provides a toolbox (MATLAB MEX-files) and a blockset (Simulink blocks based on S-functions) of commonly used DSP functions. Substituting existing MATLAB functions with the equivalent (at a behavioral level) Motorola DSP functions, you can evaluate the Motorola Suite56 family of DSPs.

In some situations, you can use the toolbox and blockset functions of the Motorola DSP Developer’s Kit unmodified as supplied. However, in the majority of cases, you will want to create your own functions either by modifying the supplied functions or by adding your own functions, basing your new functions on the supplied function templates.

### For More Information

For a complete guide to this product, refer to the online “Motorola DSP Developer’s Kit User’s Guide,” in particular, the “Getting Started” section.

Also, refer to the templates files as a guide to creating DSP MEX/SMEX functions for the Motorola DSP Developer’s Kit. These files are located in the `$MATLAB/toolbox/motdsp/motdspmex/templates` directory.

Type `demos` to run some simple demonstrations of statistical and signal processing functions calculated by the Motorola Suite56 instruction set simulator.

## New Features

This section introduces the new features and enhancements added in the Motorola DSP Developer's Kit 1.1 since the Motorola DSP Developer's Kit 1.0 (released after Release 11.1).

Version 1.1 provides integration with other Release 12 products.

### Matrix and Frame Support

Version 1.1 provides support for true matrices and frames for the blockset in Simulink. For details, see the discussion of matrix support and frames support in the "DSP Release Notes" section.

## Known Software and Documentation Problems

This section updates the Motorola DSP Developer's Kit 1.0 documentation set, reflecting known software and documentation problems.

### Interactive Mode Limitations

#### Interactive Mode Not Currently Supported on Solaris

Interactive mode has been temporarily disabled on Solaris.

#### Limited Support on Microsoft Windows Platforms

The following interactive mode functions have been disabled for Microsoft Windows platforms:

- In the toolbox: `mot56*_dspround`
- In the blockset:
  - `mot56*_sfilter`
  - `mot56*_slog`
  - `mot56*_slog10`
  - `mot56*_smean`
  - `mot56*_ssqrt`
  - `mot56*_ssum`

#### Multiple Users On the Same Machine

Two users cannot run interactive mode on the same machine at the same time.

However, if two users are working on the same machine simultaneously, one user can use interactive mode and the other can use all the functionality except interactive mode. The second user can use interactive mode after the first user is finished.

#### Exiting Interactive Mode

The proper way to exit interactive mode is to exit the Simulator itself. This causes the MATLAB engine to be terminated automatically.

### Suite56 Simulator Abnormal Termination (Solaris)

On Solaris, if the interactive mode Suite56 Simulator GUI terminates abnormally, you may need to clean up the interprocess communications pipes. You can do this by using the following commands.

```
ipcs <hit return>
ipcrm -q <ID><hit return>
```

## Stand-Alone GUI Simulator May Hang (Windows NT)

Calling the simulator `Run()` method immediately after instantiating the simulator object causes the Stand-alone GUI Simulator to hang.

To avoid this problem, call `WriteToMem/Reg()` between instantiating the simulator object and calling `Run()`.

## Other Solaris-Specific Limitations

### Limited Compiler Support

MEX-file builds currently only support Version 5.0 of Sun's native CC compiler.

### Use Full Pathnames

Using partial (relative) pathnames on Solaris machines may cause problems. Use full (absolute) pathnames for files where appropriate, or use `./` to indicate the current working directory.

## Documentation Updates

The following information supplements the information found in the online Motorola DSP Developer's Kit User's Guide.

### Setting the System Path

If you experience problems related to the system path, make sure that the following system global variables are set for your platform.

**For Microsoft Windows 95 and Windows 98 .** Make sure your `autoexec.bat` file includes the `<MATLAB_INSTALL_DIR>\bin` directory in the path variable.

Use the following command if this path is not already set, and then reboot for the changes to take effect.

```
SET PATH=%PATH%;<MATLAB_INSTALL_DIR>\bin
```

**For Solaris.** To enable Motorola MEX-files to run from any working directory, add the following directory to your `LD_LIBRARY_PATH` environment variable, if it does not already exist.

```
<MATLAB_INSTALL_DIR>/bin/
```

### Rebuilding the Motorola DSP Developer's Kit

You can conveniently rebuild the entire Motorola DSP Developer's Kit by using the supplied `build_all_xxx.m` scripts to build from the `<matlab>/toolbox/motdsp/motdspmex` directory. These scripts take between 20 and 90 minutes to run, depending on the platform and hardware specifications.

### Building MEX-Files

Use `motdsp_mexopts3xx.sh` or `motdsp_mexopts6xx.sh` to build MEX-files. Specify the MEX options by using the `-f` flag. On Solaris, these files are supplied in the `motdspmex` directory. On Windows, you must execute the `motdsp_build_mexopts.m` script to generate the `motdsp_mexopts*.sh` files.

### Length Function and Block are Obsolete

The toolbox functions `mot563_length` and `mot566_length`, mentioned in the documentation, have been removed. The corresponding blocks have also been removed from the blockset. Use the MATLAB built-in `length` function to calculate lengths.

# Neural Network Toolbox 4.0 Release Notes

**New Features** . . . . . . . . . . . . . . . . . . . . . . . 32-2
Control System Applications . . . . . . . . . . . . . . . . 32-2
Visual Interface . . . . . . . . . . . . . . . . . . . . . . . 32-2
New Training Functions . . . . . . . . . . . . . . . . . . . 32-3
Design of General Linear Networks . . . . . . . . . . . . 32-3
Improved Early Stopping . . . . . . . . . . . . . . . . . . 32-3
Generalization and Speed Benchmarks . . . . . . . . . . 32-4
Demonstration of a Sample Training Session . . . . . . . 32-4

**Platform Limitations for HP and IBM** . . . . . . . . . 32-5

**Upgrading from an Earlier Release** . . . . . . . . . . 32-6
Use of trainwb and trainwb1 Not Recommended . . . . . . 32-6

## New Features

This section introduces the new features and enhancements added in the Neural Network Toolbox 4.0 since the Neural Network Toolbox 3.0.1 (Release 11.0).

### Control System Applications

A new “Control Systems” chapter of the *Neural Network Toolbox User's Guide* presents three practical control systems applications:

- Network model predictive control
- Model reference adaptive control
- Feedback linearization controller

### Visual Interface

A visual interface has been added to the toolbox. This interface allows you to:

- Create networks
- Enter data into the visual interface
- Initialize, train, and simulate networks
- Export the training results from the visual interface to the command line workspace
- Import data from the command line workspace to the visual interface

To open the Network/Data Manager window, type `nntool`.

---

**Note** See “Platform Limitations for HP and IBM” for information about platform-specific limitations for features described in this section.

---

## New Training Functions

The Neural Network Toolbox now has four training algorithms that apply weight and bias learning rules. One algorithm applies the learning rules in batch mode. Three algorithms apply learning rules in three different incremental modes:

- `trainb` – Batch training function
- `trainc` – Cyclical order incremental training function
- `trainr` – Random order incremental training function
- `trains` – Sequential order incremental training function

All four functions present the whole training set in each epoch (pass through the entire input set).

---

**Note** We no longer recommend using `trainwb` and `trainwb1`, which have been replaced by `trainb` and `trainr`. The function `trainr` differs from `trainwb1` in that `trainwb1` only presented a single vector each epoch instead of going through all vectors, as is done by `trainr`.

---

These new training functions are relatively fast because they generate M-code. The functions `trainb`, `trainc`, `trainr`, and `trains` all generate a temporary M-file consisting of specialized code for training the current network in question.

## Design of General Linear Networks

The function `newlind` now allows you to design linear networks with multiple inputs, outputs, and input delays.

## Improved Early Stopping

Early stopping can now be used in combination with Bayesian regularization. In some cases this can improve the generalization capability of the trained network.

## Generalization and Speed Benchmarks

Generalization benchmarks comparing the performance of Bayesian regularization and early stopping are provided. Also included are speed benchmarks, which compare the speed of convergence of the various training algorithms on a variety of problems in pattern recognition and function approximation. These benchmarks can aid you in selecting the appropriate algorithm for your problem.

## Demonstration of a Sample Training Session

A new demonstration that illustrates a sample training session is included in the "Backpropagation" chapter of the *Neural Network Toolbox User's Guide*. A sample training session script is also provided. You can modify this script to fit your problem.

## Platform Limitations for HP and IBM

The Neural Network Tool, a graphical user interface for the Neural Network Toolbox, is not supported on the HP and IBM platforms. You can achieve the same functionality by using the MATLAB command window. For background information, see "Platform Limitations" on page 1-17.

## Upgrading from an Earlier Release

This section describes the upgrade issues involved in moving from the Neural Network Toolbox 3.0 to the Neural Network Toolbox 4.0.

### Use of trainwb and trainwb1 Not Recommended

We no longer recommend using `trainwb` and `trainwb1`, which have been replaced by `trainb` and `trainr`. The function `trainr` differs from `trainwb1` in that `trainwb1` only presented a single vector each epoch instead of going through all vectors, as is done by `trainr`.

# Optimization Toolbox 2.1 Release Notes

**New Features** . . . . . . . . . . . . . . . . . . . . . . 19-2
Toolbox Speed . . . . . . . . . . . . . . . . . . . . . . . 19-2
Function Handles . . . . . . . . . . . . . . . . . . . . . 19-2
Large Structured Problems . . . . . . . . . . . . . . . . . 19-2
Functions with New or Changed Capabilities . . . . . . . . 19-3

**Major Bug Fixes** . . . . . . . . . . . . . . . . . . . . . 19-4

**Upgrading from an Earlier Release** . . . . . . . . . . 19-5

## New Features

This section introduces the new features and enhancements added in the Optimization Toolbox 2.1 since the Optimization Toolbox 2.0 (Release 11.0):

- Overall toolbox speed is improved.
- Functions that take a function as an argument now support the use of function handles.
- Large structured problems are supported.
- Several other existing functions have been enhanced.

For information about Optimization Toolbox new features that are incorporated from prior releases, see *Release 11 New Features* (for enhancements introduced between Release 10.0 and Release 11.0).

### Toolbox Speed

By improving the speed of `optimset` and `optimget`, the overall speed of the toolbox is improved. Applications that call these functions repeatedly should exhibit improved execution time.

### Function Handles

For any optimization function that expects a function as an argument, you can now specify that argument as a function handle. These optimization functions also accept additional parameters, which they pass to the passed-in function.

For information about function handles, see the `function_handle` (@), `func2str`, and `str2func` reference pages, and the “Function Handles” section of “Programming and Data Types” in the MATLAB documentation.

### Large Structured Problems

The functions `fmincon`, `fminunc`, `fsolve`, `lsqcurvefit`, `lsqlin`, `lsqnonlin`, and `quadprog` now support solving large structured problems, i.e., problems that have large dense Hessian or Jacobian matrices that you do not want to form explicitly, but for which Hessian-matrix (or Jacobian-matrix) products are efficient to compute.

Two new `options` parameters, `HessMult` and `JacobMult`, provide access to this new feature.

See “Large Scale Examples” in the *Optimization Toolbox User’s Guide*, and the respective function reference pages for more information.

## Functions with New or Changed Capabilities

| **Function** | **New or Changed Capability** |
|---|---|
| `fminbnd`, `fminsearch`, `fzero`, `lsqnonneg` | A new `Display` options parameter value, `'notify'`, displays output only if the function does not converge. For these functions, `'notify'` is the new default. |
| `fmincon`, `fminunc`, `quadprog` | A new options parameter, `HessMult`, enables you to provide a function that computes the Hessian-matrix product for large structured problems. |
| `fsolve`, `lsqcurvefit`, `lsqlin`, `lsqnonlin` | A new options parameter, `JacobMult`, enables you to provide a function that computes the Jacobian-matrix product for large structured problems. |

## Major Bug Fixes

The Optimization Toolbox includes several bug fixes, including the following descriptions (online only) of particularly important bug fixes.

## Upgrading from an Earlier Release

The Optimization Toolbox uses an options structure to access various algorithm options. The Version 2.0 (R11) version of the options structure may be incompatible with the 2.1 (R12) version in some cases. In particular, if you have saved an options structure in a MAT-file from R11 and load it into R12 you may get an error similar to

```
??? Error using ==> subsref
Reference to non-existent field 'MaxSQPIter'.
```

To avoid this error, wrap a call to `optimset` around the old options structure to update it. For example,

```
optionsnew = optimset(optionsold);
```

and then use the `optionsnew` options structure.

If you were using `optimset` to create your options structure, as opposed to loading it from an older MAT-file, you should not need to make any changes.

# Power System Blockset 2.1 Release Notes

**New Features** . . . . . . . . . . . . . . . . . . . . . . 34-2
Faster Circuit Topology Analysis . . . . . . . . . . . . . . 34-2
Simulation Enhancements . . . . . . . . . . . . . . . . . . 34-2
Measurement Enhancements . . . . . . . . . . . . . . . . . 34-2
Machines Enhancements . . . . . . . . . . . . . . . . . . 34-3
Three-Phase Transformer Blocks Added . . . . . . . . . . 34-3
Power Electronics Enhancements . . . . . . . . . . . . . 34-4
Three-Phase Library Enhancements . . . . . . . . . . . . 34-4
Elements Library Enhancements . . . . . . . . . . . . . . 34-4
New Control Blocks . . . . . . . . . . . . . . . . . . . . 34-5
Powergui . . . . . . . . . . . . . . . . . . . . . . . . . 34-5
New Demos and Improved Documentation . . . . . . . . . 34-5

## New Features

This section introduces the new features and enhancements added in the Power System Blockset 2.1 since the Power System Blockset 1.1 (Release 11.0).

### Faster Circuit Topology Analysis

The Power System Blockset 2.1 dramatically speeds up circuit topology analysis. This allows your power system simulations to start much more quickly. The Power System Blockset 2.1 can handle larger, more complex models with reduced simulation overhead.

### Simulation Enhancements

In addition to the continuous simulation of power systems available in the Power System Blockset 1.1, you can now also discretize your Power System Blockset model, using the fixed-step trapezoidal method (illustrated in the Current Transformer Saturation demo). See the Discrete System block in the Power System Blockset library for more information.

This feature also can dramatically speed up your simulations – especially those with power electronics devices.

Finally, with discretized models, you now have the ability to generate code through the Real-Time Workshop for even more simulation speed.

The forward voltage of discretized power electronic devices (e.g., diode, thyristor, IGBT, and GTO) is now simulated in all cases, including discrete and continuous (with `Lon=0`).

### Measurement Enhancements

The Power System Blockset 2.1 introduces the following measurement enhancements in the powerlib Extras Measurements library:

- You can now use the multimeter to measure the voltage, current, and flux of transformers without adding voltage or current measurement blocks (see the Three-Phase Saturable Transformer demo)
- You can now measure impedance (see the Linear Filter and the Single-Phase Line demos)

- The Power System Blockset 2.1 adds the following new measurement blocks, in the powerlib Elements library:
  - 3-Phase Sequence Analyzer
  - Total Harmonic Distinction Measurement
  - abc_to_dq0 Transformation
  - dq_to_abc0 Transformation

## Machines Enhancements

The Power System Blockset 2.1 introduces the following enhancements to machines, found in the powerlib Machines library:

- The Steam Turbine and Governor block has been added.
- The DC Machine block has been added.
- For the Synchronous Machine block, a choice for salient pole or round rotor has been added.
- For the Simplified Synchronous Machine block, a choice for two, three, or four wires has been added. Use the three wire option to set the initial conditions with the Powergui load flow, then come back to the four wire option to simulate transients.
- For the Asynchronous Machine block, you can choose whether or not to have a reference frame used for dq transformation.

## Three-Phase Transformer Blocks Added

Two new Three-Phase Transformer blocks have been added for Release 12. These functions implement a three-phase transformer with a configurable phase connection.

- Three-Phase Transformer (Two Windings)
- Three-Phase Transformer (Three Windings)

## Power Electronics Enhancements

The Power System Blockset 2.1 enhances power electronics modeling by:

- Adding a universal bridge that allows continuous or discrete simulation of the three-phase bridges of diodes, thyristors, IGBT diodes, GTO diodes, or MOSFET diodes
- Adding a new IGBT model

These enhancements are found in the powerlib Power Electronics library.

## Three-Phase Library Enhancements

The Power System Blockset 2.1 adds a 3-Phase Fault block. Also, the 3-Phase Breaker block now includes internal and external control.

These enhancements are found in the powerlib Extras Three-Phase library.

## Elements Library Enhancements

The Power System Blockset 2.1 enhances the powerlib Elements library by:

- Adding the option to specify two and three windings for the Universal Three-Phase Transformer block
- Adding the option to specify internal or external breaker timing for the Circuit Breaker block

## New Control Blocks

The Power System Blockset 2.1 adds the following new control blocks to the powerlib Extras Control Blocks library:

- PWMPulse Generator
- PLL
- Synchronized 6 Pulse Generator
- Synchronized 6 Pulse Generator
- Discrete Synchronized 6 Pulse Generator
- Synchronized 12 Pulse Generator
- Discrete Synchronized 12 Pulse Generator
- HVDC Control
- 3-Phase Signal Generator

## Powergui

The Power System Blockset 2.1 includes a redesigned Powergui tool. You can now:

- Resize the Powergui windows.
- Display phasor magnitude in peak or rms values.

## New Demos and Improved Documentation

The Power System Blockset 2.1 adds several new demos to its library. The new demos illustrate a multimeter, discretization, and several new blocks. Examples of new demos include the Complete 12-Pulse HVDC Transmission System and the Three-Phase Series Compensated Network demos.

The tutorial section of the *Power System Blockset User's Guide* has been expanded, including new test cases and an explanation of how to customize your blocks.

# Real-Time Windows Target 2.0 Release Notes

**New Features** . . . . . . . . . . . . . . . . . . . . . . . 35-2
I/O Device Driver Architecture . . . . . . . . . . . . . . . 35-2
External Mode Interface . . . . . . . . . . . . . . . . . . . . 35-3
C Compiler Support . . . . . . . . . . . . . . . . . . . . . 35-3
I/O Device Driver Support . . . . . . . . . . . . . . . . . 35-3

**Upgrading from an Earlier Release** . . . . . . . . . . 35-5
Reinstall the Real-Time Windows Kernel . . . . . . . . . 35-5
Enter New External Mode Interface Filename . . . . . . . 35-5
Remove I/O Adapter Blocks from Your Simulink Model . . . 35-6
Replace I/O Driver Blocks in Your Simulink Model . . . . . 35-6

**Known Software and Documentation Problems** . . . . 35-7
Windows 2000 Not Currently Supported . . . . . . . . . 35-7
Undocumented Procedure for Configuring I/O Drivers . . . 35-7

## New Features

This section introduces the new features and enhancements added to the Real-Time Windows Target 2.0 since the Real-Time Windows Target 1.0 (Release 11.0).

**Note** The Real-Time Windows Target 2.0 extends the functionality of Real-Time Workshop.

### I/O Device Driver Architecture

The Real-Time Window Target 2.0 includes a new I/O device driver architecture that simplifies the use of I/O blocks in Simulink block diagrams. After installing a new I/O board in your PC, you place a device driver block in your model and specify board-specific settings (e.g., board address) for this board.

Once you have added a new I/O board to your PC and you have used the device driver dialog box to specify board information, the board information is retained for future sessions. When you add additional blocks, you select an I/O board from a list of installed boards. You do not have to re-enter the same board information for each driver block.

You can use multiple I/O boards in your computer. When doing so, the driver blocks show a list of installed boards for you to choose. This reduces the need to search through a growing list of device drivers.

New I/O drivers allow you to use either normalized units for analog inputs and outputs, or allow you to directly specify units as voltage.

I/O driver blocks now provide a separate I/O block for digital inputs, digital outputs, analog inputs, and analog outputs. This avoids any confusing signal indexing required to specify a combination of analog and digital I/O.

The Adapter block has been eliminated. See “Remove I/O Adapter Blocks from Your Simulink Model” on page 35-6 for upgrade issues.

## External Mode Interface

Changes have been made to the external mode interface. These include minor changes and bug fixes that ensure data is displayed at the end of a simulation even when the number of points collected differs from the specified buffer size.

## C Compiler Support

The Real-Time Windows Target 2.0 supports Microsoft Visual C/C++ version 5.0 and 6.0. This support was initially introduced in the Real-Time Windows Target 1.5, which was available as a download from the Web.

## I/O Device Driver Support

New I/O device drivers have been added to the Real-Time Windows Target 2.0 for supporting additional boards.

**Counter support** - Real-Time Windows Target supports the counter boards CIO-QUAD02 and CIO-QUAD04 from ComputerBaords.

Following is a list of the new supported boards:

- ComputerBoards PCI-DAS1000
- ComputerBoards PCI-DAS1001
- ComputerBoards PCI-DAS1002
- ComputerBoards PCI-DAS1200
- ComputerBoards PCI-DAS1200/JR
- ComputerBoards PCI-DAS1602/12
- ComputerBoards PCI-DAS1602/16
- ComputerBoards PCI-DAS1602/16-JR
- ComputerBoards PCM-DAC08
- ComputerBoards PCM-DAS16S/330
- ComputerBoards PCM-DAS16D/12
- ComputerBoards PCM-DAS16S/12
- ComputerBoards PCM-DAS16D/16
- ComputerBoards PCM-DAS16S/16
- ComputerBoards CIO-QUAD02
- ComputerBoards CIO-QUAD04

- National Instruments Lab-PC-1200
- National Instruments Lab-PC-1200AI
- National Instruments PCI-1200
- National Instruments PXI-6025E
- National Instruments PXI-6031E
- National Instruments PXI-6040E
- National Instruments PXI-6052E
- National Instruments PXI-6070E
- National Instruments PXI-6071E
- Technology 80 5312B

## Upgrading from an Earlier Release

This section describes the upgrade issues involved in moving from the Real-Time Windows Target 1.0 and 1.5 to the Real-Time Windows Target 2.0.

### Reinstall the Real-Time Windows Kernel

When upgrading from an earlier release, first uninstall the Real-Time Windows Target kernel. In the MATLAB window, type

```
rtwintgt -uninstall
```

After you install MATLAB R12 products, including the Real-Time Windows Target 2.0, install the new Real-Time Windows Target kernel. In the MATLAB window, type

```
rtwintgt -install
```

Depending on which version of Windows you are using on your PC, you may need to reboot your PC before using the Real-Time Windows Target. If a reboot is required, a message appears indicating a reboot is needed.

### Enter New External Mode Interface Filename

The name of the external mode interface file was change from `win_tgt` to `rtwinext`. If you create a new Simulink model, the new filename is entered correctly. If you have Simulink models where you used Real-Time Windows Target 1.0 or 1.5, you need to change the filename using the following procedure:

1. In the Simulink window, and from the **Tools** menu, click **External mode control panel**.

2. On the External Mode Control Panel dialog box, click the **Target interface** button.

3. In the External Target Interface dialog box, and in the **MEX-file for external mode** box, enter

   ```
   rtwinext
   ```

4. Click **Ok**.

## Remove I/O Adapter Blocks from Your Simulink Model

I/O Adapter blocks have been eliminated from the Real-Time Windows Target 2.0. You need to remove all I/O Adapter blocks from your Simulink model. Before removing these blocks, we recommend that you record board and connectivity information.

You now enter specific board information in the individual driver blocks. However, you only have to do this once. After entering information in the first block, you can select your board from a pull-down list in the following blocks.

## Replace I/O Driver Blocks in Your Simulink Model

the Real-Time Windows Target 1.0 and 1.5 had only two driver blocks – RT Input and RT Output. In the Real-Time Windows Target 2.0, the drivers are now divided into Analog Input (A/D), Analog Output (D/A), Digital Input, Digital Output, and others.

You need to remove all the Real-Time Windows Target 1.0 and 1.5 blocks from your Simulink model and replace them with the new Version 2.0 blocks. Remember to replace RT Input with the appropriate Analog Input and Digital Input blocks. And similarity, you must replace RT Output with Analog Output and Digital Output blocks. For example, if you had a block with both analog input and digital input, you must replace this block by adding two new blocks.

# Known Software and Documentation Problems

This section updates the Real-Time Windows Target 2.0 documentation set, reflecting known the Real-Time Windows Target 2.0 software and documentation problems.

## Windows 2000 Not Currently Supported

Real-Time Windows Target does not currently support Windows 2000. Please see The MathWorks Web site for the latest news on support for Windows 2000.

## Undocumented Procedure for Configuring I/O Drivers

Currently, the documentation does not describe the use of the new I/O blocks. However, the graphical user interfaces (GUIs) have been designed in a manner that is far more intuitive and easy to use than the driver blocks in Version 1.0 and 1.5.

# Real-Time Workshop Ada Coder 4.0 Release Notes

**New Features** . . . . . . . . . . . . . . . . . . . . . . . 36-2
Data-Logging Format Support . . . . . . . . . . . . . . . 36-2
Ada83 Support . . . . . . . . . . . . . . . . . . . . . . . . 36-2
S-Function Support . . . . . . . . . . . . . . . . . . . . . 36-2
Block Comment Generation . . . . . . . . . . . . . . . . 36-3

## New Features

This section introduces the new features and enhancements added in the Real-Time Workshop Ada Coder 4.0 since the Real-Time Workshop Ada Coder 3.0.1 (Release 11.0).

---

**Note** The Real-Time Workshop Ada Coder 4.0 requires the Real-Time Workshop.

---

### Data-Logging Format Support

The Real-Time Workshop Ada Coder (Ada95 targets) now supports all data-logging formats (matrix, structure, and structure /time).

See "Data Logging" in the *Real-Time Workshop User's Guide* for further information.

### Ada83 Support

The Ada Coder now generates Ada83 code, in addition to Ada95 code.

The Ada83 code does not support data logging, but is suitable for generating embedded code for use with legacy Ada83 compilers.

The Ada83 target is available from the System Target File Browser by selecting the system target file `rt_ada83.tlc` (Ada83 Target for GNAT). The accompanying template makefile `gnat83.tmf` uses the `-gnat83` switch in the GNAT Ada95 compiler.

See "Supported Targets" in the *Real-Time Workshop User's Guide* for further information.

### S-Function Support

Simulink now supports noninlined Ada S-functions. For Real-Time Workshop code generation purposes, you can create a wrapper S-function that calls an Ada S-function.

## Block Comment Generation

The new **Insert block descriptions in code** option lets you insert text from the **Description** field of blocks in your model as comments in generated code. When this option is selected, comments are inserted into the code generated for any blocks that have text in their **Description** fields. See “Generating Block Comments” in the *Real-Time Workshop User's Guide* for further information.

# Real-Time Workshop Embedded Coder 1.0

**Introduction to the Real-Time Workshop Embedded Coder** . . . . . . . . . . . . . . . . . 37-2
Real-Time Workshop Embedded Coder Demos . . . . . . . . 37-2

## Introduction to the Real-Time Workshop Embedded Coder

The Real-Time Workshop Embedded Coder is an add-on product for use with the Real-Time Workshop. The Real-Time Workshop Embedded Coder is designed for generation of highly optimized code for embedded systems.

The Real-Time Workshop Embedded Coder replaces and enhances the Embedded Real-Time (ERT) target. It is 100% compatible with the ERT target.

In addition to supporting all previous functions of the ERT target, the Real-Time Workshop Embedded Coder includes the following enhancements:

- Support for singletasking multirate models.
- The **Create Simulink (S-Function) block** option lets you build a noninlined C MEX S-function wrapper that calls the generated code in one automated step. The option also builds a model containing the generated S-Function block.
- The **Generate HTML report** option writes a Web-viewable code generation report that describes code modules and helps to identify code generation optimizations relevant to your program. The report is automatically opened into the MATLAB Help Browser.
- The **Insert block descriptions in code** option lets you insert text from the **Description** field of blocks in your model as comments in generated code.
- The **Generate ASAP2 file** code generation option lets you export an ASAP2 file containing information about your model during the code generation process. See “Generating ASAP2 Files” in the Real-Time Workshop online documentation for information.

See “The Real-Time Workshop Embedded Coder” in the *Real-Time Workshop User's Guide* for full documentation of the Real-Time Workshop Embedded Coder.

## Real-Time Workshop Embedded Coder Demos

We have provided a number of demos to help you become familiar with features of the Real-Time Workshop Embedded Coder and to inspect generated code. To run the demos:

1. From the MATLAB **Help** menu, select **Demos**. The MATLAB Demo Window opens.

   Alternatively, you can open the MATLAB Demo Window by typing the following command at the MATLAB prompt.

   ```
   demos
   ```

2. In the topic list on the left of the MATLAB Demo Window, double-click **Real-Time Workshop**.
3. A subtopic list opens below **Real-Time Workshop**. Select **Embedded Coder** from the subtopic list.
4. Click the **Run Demos** button. The Real-Time Workshop Embedded Coder demo window opens.
5. Double-click on the blue **Open** box to run the desired demo.
6. A Simulink window, containing a demonstration model and instructions, opens. Follow the instructions for the demo.

# Requirements Management Interface 1.0 Release Notes

**Introducing the Requirements Management Interface** . 38-2

**Platform Limitations for HP and IBM** . . . . . . . . . 38-3

## Introducing the Requirements Management Interface

The Requirements Management Interface 1.0 was introduced as a Web-downloadable product after Release 11.1. There are no new features or changes for the Requirements Management Interface for Release 12.

The Requirements Management Interface allows you to associate requirements with Simulink models, Stateflow diagrams, and MATLAB M-files.

There is a standard version of the Requirements Browser and a version that works with the DOORS requirements management system, a Quality Software and Systems, Inc. (QSS) product. For the standard version, the requirements are in Microsoft Word, Microsoft Excel, and HTML documents. For the DOORS version, the requirements are managed by DOORS.

When you use the DOORS version of the Requirements Management Interface for a given model, diagram, or M-file, you can only associate it with DOORS requirements (and not Word, Excel, or HTML requirements documents). Similarly, when you use the standard version of the Requirements Management Interface for a given model, diagram, or M-file, you can only associate Word, Excel, and HTML requirements documents with it (and not DOORS requirements).

Use the Requirements Management Interface to:

- Associate requirements with the following objects:
  - Simulink subsystems and blocks
  - Stateflow charts, states, and transitions
  - MATLAB M-files
- See which objects have requirements associated with them.
- Go from the Requirements Management Interface Navigator to a requirement.
- Go from a requirement in DOORS to a MATLAB object (for DOORS version only).
- Run a MATLAB script that executes a Simulink simulation or M-file from DOORS (for DOORS version only).

For instructions to use the Requirement Management Interface, see the documentation in the Help browser.

## Platform Limitations for HP and IBM

The Requirements Management Interface is not supported on HP and IBM platforms. For background information, see “Platform Limitations” on page 1-17.

# Signal Processing Toolbox 5.0 Release Notes

**New Features** . . . . . . . . . . . . . . . . . . . . . . . 39-2
Filter Design & Analysis Tool (FDATool) . . . . . . . . . . 39-3
SPTool . . . . . . . . . . . . . . . . . . . . . . . . . . . 39-4
Spectrogram Demo . . . . . . . . . . . . . . . . . . . . . 39-5
Automatic Order Adjustment in FIR Filter Functions . . . . 39-5
grpdelay . . . . . . . . . . . . . . . . . . . . . . . . . 39-6
hilbert . . . . . . . . . . . . . . . . . . . . . . . . . . 39-6
lpc . . . . . . . . . . . . . . . . . . . . . . . . . . . . 39-6
xcorr . . . . . . . . . . . . . . . . . . . . . . . . . . . 39-6

**Major Bug Fixes** . . . . . . . . . . . . . . . . . . . . 39-7

**Upgrading from an Earlier Release** . . . . . . . . . . 39-8
Changes to remez . . . . . . . . . . . . . . . . . . . . 39-8

## New Features

This section introduces the new features and enhancements added in the Signal Processing Toolbox 5.0 since the Signal Processing Toolbox 4.3 (Release 11.1).

For information about the Signal Processing Toolbox features incorporated in prior releases, see:

- *Release 11.1 New Features*
  (enhancements introduced between Release 11.0 and Release 11.1)
- *Release 11 New Features*
  (enhancements introduced between Release 9.0 and Release 11.0)

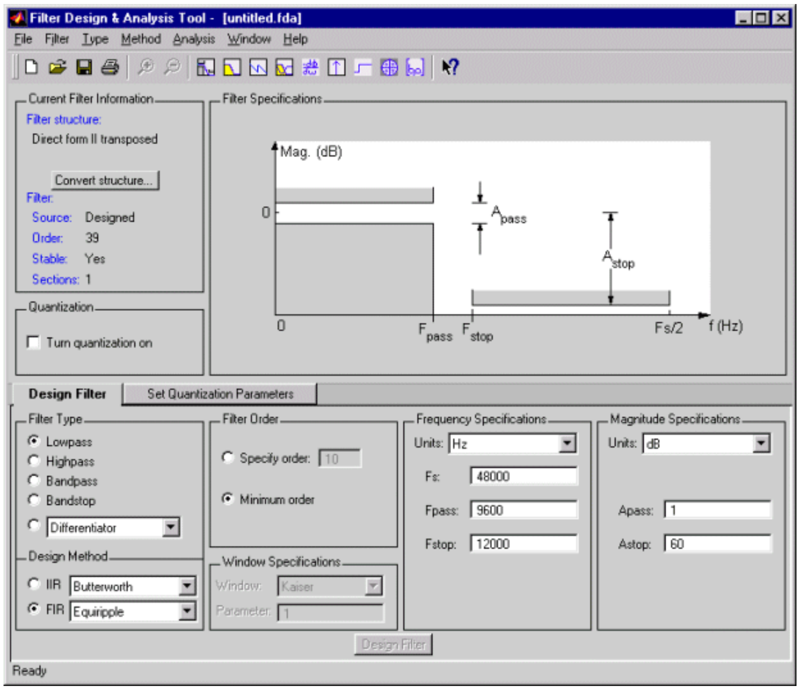
The Signal Processing Toolbox 5.0 provides a number of important enhancements, and a completely new facility for designing and analyzing filters, the Filter Design & Analysis Tool (FDATool).

This section is organized into the following subsections:

- "Filter Design & Analysis Tool (FDATool)" on page 39-3
- "SPTool" on page 39-4
- "Spectrogram Demo" on page 39-5
- "Automatic Order Adjustment in FIR Filter Functions" on page 39-5
- "grpdelay" on page 39-6
- "hilbert" on page 39-6
- "lpc" on page 39-6
- "xcorr" on page 39-6

## Filter Design & Analysis Tool (FDATool)

The Filter Design & Analysis Tool provides a simple and intuitive graphical interface to most of the Signal Processing Toolbox's digital filter design functions.



Filter structure: Direct form II transposed

Filter:
Source: Designed
Order: 39
Stable: Yes
Sections: 1

Turn quantization on [ ]

Filter Type: [✓ Lowpass] Highpass Bandpass Bandstop Differentiator

Filter Order: Specify order: 10 [✓ Minimum order]

Design Method: IIR Butterworth [✓ FIR Equiripple]

Window Specifications: Window: Kaiser Parameter: 1

Frequency Specifications: Units: Hz Fs: 48000 Fpass: 9600 Fstop: 12000

Magnitude Specifications: Units: dB Apass: 1 Astop: 60

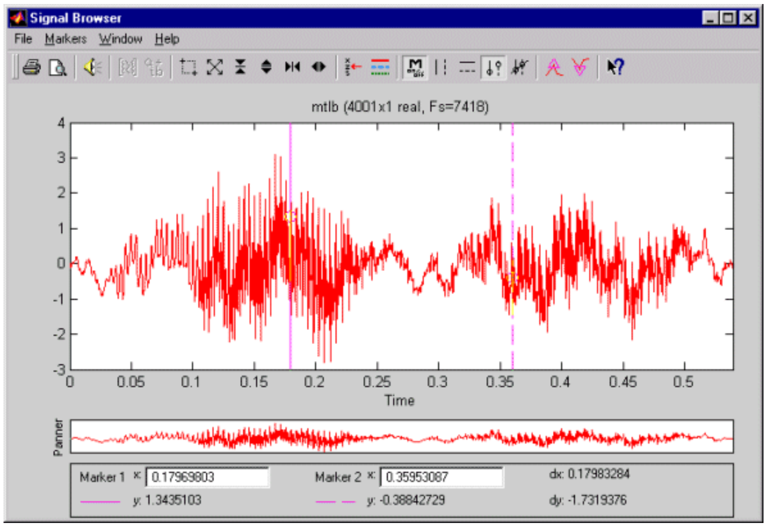
Ready

To start the tool, type `fdatool` at the MATLAB command line.

## SPTool

SPTool has received a number of enhancements:

- The Signal Browser can now play a selected portion of a signal.
- The markers (rulers) have been enhanced in all component tools (Signal Browser, Filter Viewer, Filter Designer, and Spectrum Viewer).
- The toolbars have been enhanced in all component tools. (The new Signal Browser interface is shown below.)

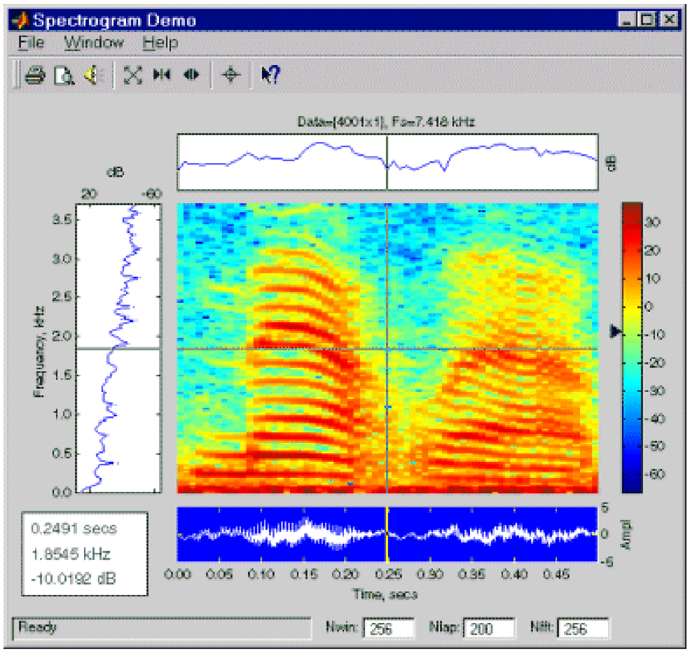


Marker 1 x: 0.17969803 y: 1.3435103
Marker 2 x: 0.35953087 y: -0.38842729
dx: 0.17983284 dy: -1.7319376

To start the tool, type `sptool` at the MATLAB command line.

## Spectrogram Demo

The Spectrogram Demo (`specgramdemo(y,Fs)`) is a new demo that displays a spectrogram, a time plot, and a frequency slice of an input signal, *y*, assuming a sample rate of *Fs* Hz.



## Automatic Order Adjustment in FIR Filter Functions

Most of the FIR filter functions (`fir1`, `fir2`, `firls`, `remez`, `fircls`, `fircls1`) now automatically increase the order of linear phase filters by 1 when necessary to meet the design specifications. For example, a Type II (odd order, symmetric) linear phase filter must have a zero at $\pi$ radians. If a gain other than zero is instead specified at $\pi$ radians in any of the above mentioned functions, the function adds 1 to the filter order and proceeds to design the higher order filter.

A warning message is produced in the command window whenever a function automatically changes the filter order.

### grpdelay

The `grpdelay` function now returns more accurate results for IIR filters.

### hilbert

The `hilbert` function no longer zero-pads sequences that have a non-power-of-two length, so the transforms are now exact.

### lpc

The `lpc` function can now return an estimate of the prediction error variance (power) as a second output.

### xcorr

The `xcorr` function can now efficiently compute the cross-correlation and autocorrelation of large vectors.

## Major Bug Fixes

The Signal Processing Toolbox includes several bug fixes, including the following descriptions (online only) of particularly important bug fixes.

## Upgrading from an Earlier Release

This section describes the upgrade issues involved in moving from the Signal Processing Toolbox 4.3 (Release 11.1) to the Signal Processing Toolbox 5.0.

### Changes to remez

In the Signal Processing Toolbox 5.0, the `remez` function no longer supports the `'m'` option that previously executed an M-file variant of the function.

# Simulink Performance Tools 4.0 Release Notes

**Introduction to the Simulink Performance Tools** . . . 40-2
The Simulink Accelerator . . . . . . . . . . . . . . . . . 40-2
The Test Coverage Tool . . . . . . . . . . . . . . . . . 40-2
The Model Differences Tool . . . . . . . . . . . . . . . 40-2
The Model Profiler . . . . . . . . . . . . . . . . . . . 40-2

**Platform Limitations for HP and IBM** . . . . . . . . 40-3

**Known Software and Documentation Problems** . . . . 40-4
Simulink Model Differences Tool . . . . . . . . . . . . 40-4

## Introduction to the Simulink Performance Tools

The Simulink Performance Tools are an optional, separately priced, set of tools that enhance your Simulink development environment. The Simulink Performance Tools are new with Release 12. The Simulink Performance Tools consist of:

- The Simulink Accelerator
- The Model Coverage Tool
- The Model Differences Tool
- The Model Profiler

### The Simulink Accelerator

The Simulink Accelerator speeds simulation of a Simulink model. It does this by first compiling a model into executable code, and then running the compiled version.

### The Model Coverage Tool

The Model Coverage Tool reports the extent to which simulation of a model exercises possible execution pathways through the model. See “Performance Tools” in the online help for Simulink.

### The Model Differences Tool

The Model Differences Tool displays differences between two Simulink models. For example, the tool displays all blocks that exist in both models but have different attributes and all blocks that are present in only one of the two models. The tool documentation does not document the following feature. You can use the `mdldiff` command to compare two libraries. Type `help mdldiff` at the MATLAB command prompt for more information.

### The Model Profiler

The Model Profiler generates and displays a profile of a simulation run. The profile shows how much time Simulink spent in each function required to simulate the model.

## Platform Limitations for HP and IBM

The Model Differences Tool is not supported on the HP and IBM platforms. For background information, see “Platform Limitations” on page 1-17.

## Known Software and Documentation Problems

This section updates the Simulink Performance Tools 4.0 documentation, reflecting known Simulink Performance Tools 4.0 software and documentation problems.

### Simulink Model Differences Tool

The Prerelease version of the Simulink Model Differences Tool has these problems:

- Comparing models with the same name will not work. Rename one of the model before doing the comparison.
- The contents of masked subsystems are not shown.
- Help is not available.

# Simulink Report Generator 1.1 Release Notes

**New Features** . . . . . . . . . . . . . . . . . . . . . . . . 41-2
New Simulink Components . . . . . . . . . . . . . . . . . 41-2
New Stateflow Components . . . . . . . . . . . . . . . . . 41-3
New Fixed-Point Components . . . . . . . . . . . . . . . . 41-3
Enhancements to Components . . . . . . . . . . . . . . . 41-5

**Upgrading from an Earlier Release** . . . . . . . . . . 41-6
Use of Existing SGML Files . . . . . . . . . . . . . . . . 41-6

## New Features

This section introduces the new features and enhancements added in the Simulink Report Generator 1.1 since the Simulink Report Generator 1.0 (Release 11.0).

---

**Note** The Simulink Report Generator extends the functionality of the MATLAB Report Generator.

---

### New Simulink Components

The Simulink Report Generator 1.1 includes the following new Simulink components.

| Component | Purpose |
|---|---|
| Block Automatic Property Table | Creates a two-column property name/ property value table |
| Block Type: Bus | Creates a list of all signals exiting from a Bus Selector block |
| Block Type: Look-Up Table | Reports on one-dimensional and two-dimensional look-up table blocks |
| Model Change Log | Uses a reported model's `ModifiedHistory` parameter to construct a model history table that displays information about each logged revision to the model |

## New Stateflow Components

The Simulink Report Generator 1.1 includes the following new Stateflow components.

| Component | Purpose |
|---|---|
| Chart Loop | Runs its children for each Stateflow chart that you choose |
| Stateflow Linking Anchor | Designates a location to which other links point (for use with Stateflow) |
| Stateflow Name (Chart) | Inserts the name of the Stateflow object specified by its parent component into the report |

## New Fixed-Point Components

The following components have been added to the Simulink Report Generator 1.1 to support the Fixed-Point Blockset 3.0.

| Component | Purpose |
|---|---|
| Fixed-Point Block Loop | Runs its children for the Simulink model, system, or signal defined by its parent component; for example, if the Fixed-Point Block Loop is parented by the Simulink Model Loop, it will run its children for all fixed-point blocks in the Simulink model. |
| Fixed-Point Logging Options | Sets fixed-point options similar to those set in the Fixed-Point Blockset Interface GUI |

| Component | Purpose |
|---|---|
| Fixed-Point Property Table | Inserts a property name/property value table for a fixed-point block |
| Fixed-Point Summary Table | Shows properties or parameters of the selected fixed-point blocks in a table |

## Enhancements to Components

The following components have been enhanced for Release 12.

| Component | Enhancement |
|---|---|
| Block Loop | You can create a loop list manually. |
| Object Report | New box object added |
| Block Type: Scope Snapshot | You can add captions. |
| Signal Loop | You can remove input/output/internal signals from a loop. |
| Stateflow Snapshot (chart) | You can turn off callouts. |
| System Hierarchy | New list style (bulleted or numbered) control added |
| System Loop | You can loop on a manually created list of systems |
| System Snapshot | You can add captions. |

## Upgrading from an Earlier Release

This section describes the upgrade issues involved in moving from the MATLAB Report Generator 1.0 to the MATLAB Report Generator 1.1.

### Use of Existing SGML Files

If you generated a report using the Simulink Report Generator 1.0 and saved the report file as an SGML file, and then try to convert that SGML file using the Simulink Report Generator 1.1 `rptconvert` program, the conversion fails.

In such cases, you need to regenerate the SGML file using the Simulink Report Generator 1.1.

# Spline Toolbox 3.0 Release Notes

**New Features** . . . . . . . . . . . . . . . . . . . . . . 42-2
Spline Tool Provides a Visual Interface to the Spline Toolbox . . . . . . . . . . . . . . . . . . . 42-2
Automatic Knot Choice Simplifies Use of spapi and spap2 . . 42-2
Automatic Smoothing Parameter Choice Simplifies Use of csaps . . . . . . . . . . . . . . . . . . . . 42-2
Use of Rational Splines . . . . . . . . . . . . . . . . . 42-3
B-Spline Visual Interface . . . . . . . . . . . . . . . . 42-3
Other New Functions . . . . . . . . . . . . . . . . . . 42-3
Other Enhancements . . . . . . . . . . . . . . . . . . 42-3

**Major Bug Fix** . . . . . . . . . . . . . . . . . . . . . . 42-5
spmak and ppmak Use Size Argument to Correctly Construct Multivariate Splines . . . . . . . . . . . . . . . 42-5

**Upgrading from an Earlier Release** . . . . . . . . . 42-7
optknt and newknt Output No Longer Needs To Be Run Through augknt . . . . . . . . . . . . . . . . . 42-7

**Known Software and Documentation Problems** . . . . 42-8
Correction for splinetool Reference Page . . . . . . . . . 42-8

## New Features

This section introduces the new features and enhancements added in the Spline Toolbox 3.0 since the Spline Toolbox 2.0.1 (Release 11.0).

### Spline Tool Provides a Visual Interface to the Spline Toolbox

The `splinetool` function invokes a new visual interface that allows you to:

- Import data in various ways
- Try out and compare all the different spline fits available
- Vary the parameters, including the data
- Look at the error
- Look at the derivatives
- View the toolbox functions that produce specific spline fits
- Print the graphs, and save approximations for subsequent use

### Automatic Knot Choice Simplifies Use of spapi and spap2

Until this release, if you wanted to construct a spline interpolant to given data, you had to specify the spline space from which this spline was to be chosen, by providing an appropriate knot sequence. Starting with this release, if you are not so certain about how to choose knots, you can simply specify the order of the spline to be used instead, and `spapi` will provide a suitable knot sequence.

The same difficulty of having to choose a knot sequence occurred in the construction of a least-squares spline approximation to given data, and here, too, you can instead merely specify the number of polynomial pieces of the given order to be used in the approximating spline in `spap2`.

### Automatic Smoothing Parameter Choice Simplifies Use of csaps

You can now use `csaps` without specifying the smoothing parameter to be used. If none is specified, `csaps` will optionally return the one it chose for the given data, for further experimentation.

## Use of Rational Splines

The relevant function functions (e.g., fnval, fnplt, fntlr, fnbrk, fnrfn, fn2fm, etc.) can now operate on rational splines (NURBS). Specific examples of a rational spline are provided by rsmak. Both rsmak and rpmak are available to generate arbitrary rational splines in B-form and ppform, respectively.

## B-Spline Visual Interface

Splines in the Spline Toolbox are constructed as a linear combination of B-splines. Run bspligui to show how such a B-spline varies as you vary its knots. You can:

- Add knots
- Delete knots
- Move knots or breaks
- Increase/decrease the multiplicity of a knot

## Other New Functions

The following functions have been added in the Spline Toolbox 3.0:

- aptknt(x,k) provides a good knot sequence for interpolation by splines of order k to data at x.
- fndir is available for the construction of directional derivatives, and hence of Jacobians, gradients, and surface normals.
- fntlr is available for the calculation of derivative values; this is particularly useful for rational splines for which formal differentiation is inefficient.
- chbpnt(knots,k) provides a good data site sequence for interpolation by splines of order k with knot sequence knots.

## Other Enhancements

- Both csaps and spaps can now work with a nonconstant weight in the roughness measure.
- Both csaps and spaps can also now deal better with near-zero error weights.
- You can now give spaps a smoothing parameter rather than a tolerance.
- fnbrk can now change the basic interval of any form.

- You can now make fnval treat splines as continuous from the left.
- You can also now use fnval in the form fnval(x,f) as needed for fzero, fminbnd, etc.
- fnplt can now be made not to break the graph of a function at a jump.
- The second argument of newknt(fn,newl) has become optional.
- aveknt(x,k) can now handle an x of length k (of use in aptknt).
- optknt can now handle much more nonuniformly spaced data sites, particularly by using optknt(tau,k,maxiter) to increase the maximum number of steps used to iteratively solve for the optimal knots.

## Major Bug Fix

The Spline Toolbox 3.0 includes several bug fixes, including the following particularly important bug fix.

### spmak and ppmak Use Size Argument to Correctly Construct Multivariate Splines

If the spline to be constructed by `spmak (knots, coefs)` is multivariate, but is meant to be a constant function without any interior knots in its last variable, then the last dimension of `coefs` is necessarily 1.

For example, a one-piece trivariate constant function would have knots stored in a length 3 cell array and a 3-D coefficient array.

```
knots = {[0 1],[0 1],[0 1]};
coefs = ones([1,1,1]);
```

However, this `coefs` array would be truncated to 2-D because MATLAB suppresses all trailing singleton dimensions (dimensions greater than 2 whose size is 1). Thus in earlier versions of the toolbox, this would fail.

```
spmak(knots, coefs)
??? Error using ==> spmak
coefs must be a ([1+]length(KNOTS))-dimensional array
```

Even if you managed to construct the B-form of this constant function, the various `fn`... functions could not work with it.

For the Spline Toolbox 3.0, all these `fn`... functions now handle splines with coefficient arrays of this kind correctly. You can now specify the intended size of the coefficient array by

```
spmak(knots,coefs,sizec)
```

where `sizec` has the intended dimensionality of `coefs`. Now

```
spmak({[0 1],[0 1],[0 1]},ones([1,1,1]),[1 1 1])
```

will correctly construct a one-piece trivariate constant function, as will this.

```
spmak({[0 1],[0 1],[0 1]},1,[1 1 1]
```

Analogously, you can now use the optional third input argument, `d`, in

```
ppmak(breaks, coefs, d)
```

to specify the intended dimensions of the coefficient array.

## Upgrading from an Earlier Release

This section describes the upgrade issues involved in moving from the Spline Toolbox 2.0 (Release 11) to the Spline Toolbox 3.0.

### optknt and newknt Output No Longer Needs To Be Run Through augknt

In the Spline Toolbox 3.0, you no longer need to run the output from `optknt` and `newknt` through `augknt` to get a complete knot sequence. Running that output through `augknt` now does not cause any problems, but is unnecessary.

## Known Software and Documentation Problems

### Correction for splinetool Reference Page

The example on the reference page for the `splinetool` function incorrectly mentions a **newknt** button as being part of the visual interface. In fact, the name of that button is the **adjust** button.

# Stateflow Coder 4.0 Release Notes

**New Features** . . . . . . . . . . . . . . . . . . . . . 43-2
Improved Code Generation . . . . . . . . . . . . . . . . . 43-2
Temporal Logic . . . . . . . . . . . . . . . . . . . . . 43-2

## New Features

This section introduces the new features and enhancements added in Stateflow Coder 4.0 since Stateflow Coder 2.0.1.

For information about Stateflow Coder new features that are incorporated from prior releases, see:

- *Release 11.1 New Features*
  (enhancements introduced between Release 11.0 and Release 11.1)
- *Release 11 New Features*
  (enhancements introduced between Release 9.0 and Release 11.0)

**Note** The Stateflow Coder extends the functionality of Stateflow.

### Improved Code Generation

The Stateflow Coder 4.0 code generation has been significantly improved:

- The code looks hand-written.
- ROM and RAM size rivals hand-written code.
- Code generation is faster.

### Temporal Logic

**Note** The temporal logic feature was introduced in Stateflow Coder 3.0.

You can now use temporal conditions (before, after, at, every time) to determine the activation of transitions and duration of state activation. Temporal logic provides a simple paradigm for event scheduling.

Temporal conditions allow your Stateflow model to express clearly and simply the time-dependent behavior of a system.

Stateflow Coder 4.0 supports the new temporal logic feature in Stateflow, which allows it to be extremely efficient with memory usage in the generated code. When you use temporal operators, Stateflow manages the various counters needed for event scheduling. Then, with behind-the-scenes optimizations, Stateflow Coder 4.0 can allocate memory more economically.

# Statistics Toolbox 3.0 Release Notes

**New Features** . . . . . . . . . . . . . . . . . . . . . . . 44-2
Summary of Enhancements . . . . . . . . . . . . . . . . . 44-2
New Functions . . . . . . . . . . . . . . . . . . . . . . . 44-3
New Demos . . . . . . . . . . . . . . . . . . . . . . . . . 44-4
New Sample Data Files . . . . . . . . . . . . . . . . . . 44-5
Updated Functions for ANOVA-Type Tables . . . . . . . 44-5
Other Updated Functions . . . . . . . . . . . . . . . . . 44-6

## New Features

This section introduces the new features and enhancements added in the Statistics Toolbox 3.0 since the Statistics Toolbox 2.2 (Release 11.0).

For information about Statistics Toolbox 3.0 new features that are incorporated from prior releases, see *Release 11 New Features* (for enhancements introduced between Release 9.0 and Release 11.0).

### Summary of Enhancements

#### Expanded Support for Linear Models

Version 3.0 expands the Statistics Toolbox support for linear models in general, and analysis of variance in particular. The following are the major changes for the Statistics Toolbox version 3.0:

- Improvements in one-way analysis of variance (`anova1`)
- Higher-way analysis of variance (`anovan`)
- Analysis of covariance (`aoctool`)
- Multiple comparisons of means or other estimates (`multcompare`)
- Multivariate analysis of variance (`manova1`, `manovacluster`)
- Graphics functions useful for examining data used for multivariate analysis of variance (`gscatter`, `gplotmatrix`, `gname`)
- Response surface fitting with multiple responses (`rstool`)
- Nonparametric analysis of variance (`friedman`, `kruskalwallis`)
- More flexible calculation of confidence bounds (`polytool`, `nlintool`, `nlpredci`)

#### Other Enhancements

In addition, the following changes do not involve linear models:

- Generalized linear models (`glmfit`, `glmval`)
- Robust regression (`robustfit`, `polytool`)
- Distribution testing and plotting (`cdfplot`, `lillietest`, `kstest`, `kstest2`)
- Fractional factorial design generation (`fracfact`)
- Importing numeric and text data from tab-delimited files (`tdfread`)

- More flexible handling of grouping variables (boxplot, grpstats)
- Multivariate t random number generation (mvtrnd) and improvements to other t distribution functions

Numerous other functions received enhancements, as described in the following sections:

- "New Functions" on page 44-3
- "New Demos" on page 44-4
- "New Sample Data Files" on page 44-5
- "Updated Functions for ANOVA-Type Tables" on page 44-5
- "Other Updated Functions" on page 44-6

## New Functions

The following functions have been added to the Statistics Toolbox 3.0.

| Function | Description |
|---|---|
| anovan | N-way Analysis of Variance (ANOVA) |
| aoctool | Interactive plot for fitting and predicting analysis of covariance models |
| cdfplot | Plot of empirical cumulative distribution function |
| fracfact | Generate fractional factorial design from generators |
| friedman | Friedman's nonparametric two-way Analysis of Variance (ANOVA) |
| glmfit | Generalized linear model fitting |
| glmval | Compute predictions for generalized linear model |
| gplotmatrix | Plot matrix of scatter plots by group |
| gscatter | Scatter plot by group |

| Function | Description |
|---|---|
| `jbtest` | Jarque-Bera test for goodness-of-fit to a normal distribution |
| `kruskalwallis` | Kruskal-Wallis nonparametric one-way Analysis of Variance (ANOVA) |
| `kstest` | Kolmogorov-Smirnov test of the distribution of one sample |
| `kstest2` | Kolmogorov-Smirnov test to compare the distribution of two samples |
| `lillietest` | Lilliefors test for goodness-of-fit to a normal distribution |
| `manova1` | One-way Multivariate Analysis of Variance (MANOVA) |
| `manovacluster` | Plot dendrogram showing group mean clusters after MANOVA |
| `multcompare` | Multiple comparison test of means or other estimates |
| `mvtrnd` | Random matrices from the multivariate t distribution |
| `robustfit` | Robust regression |
| `tdfread` | Read file containing tab-delimited numeric and text values |

## New Demos

### glmdemo

The `glmdemo` function is a slideshow-style demo of generalized linear model fitting.

### robustdemo

The `robustdemo` function demonstrates robust fitting. The function graphs (*x*,*y*) data with an outlier, and shows how the least squares and robust fits differ. You can move points with the mouse, and see how the two fits change. You can also display the least squares leverage and the robust weight for each point. You can also provide input data instead of using the built-in example.

## New Sample Data Files

### carbig

The `carbig` data file is a large dataset on cars from the 70s and 80s.

### carsmall

The `carsmall` data file is a subset of `carbig`, containing cars from just three model years.

## Updated Functions for ANOVA-Type Tables

Several functions display tables, such as ANOVA tables, in a figure window. These figure windows now have a new **Copy Text** option on the **Edit** menu. You can use this option to copy the table as tab-delimited text into Microsoft Excel, Microsoft Word, or other applications. These two functions that produce such tables have been updated:

- `anova1`
- `anova2`

The changes to each of these functions are described below.

---

**Note** The `aoctool`, `friedman`, and `kruskalwallis` functions are new functions added in the Statistics Toolbox 3.0; these functions also display ANOVA-type tables.

---

### anova1

```
[p,table,stats] = anova1(x,group,'displayopt')
```

- New output `table` is a cell array of the ANOVA table values, including row and column labels.
- Now returns a `stats` output structure useful for performing multiple comparisons (see `multcompare` for more information).
- New input `'displayopt'` is `'off'` to omit the table and boxplot display, or `'on'` (the default) to display the table and boxplot.
- If `x` is a matrix, `group` can now be a character array or cell array of strings with one row for each column of `x`. The boxes in the boxplot are then labeled using the rows of `group`.
- If `x` is a vector, `group` can be a vector of integers or a character array or cell array of strings with one row for each element of `x`. The boxes in the boxplot are labeled with values from `group`.
- P-value added to both `table` and to the table display.
- Now accepts group numbers that are not of the form 1, …, *g*.

### anova2

```
[p,table,stats] = anova2(x,reps,'displayopt')
```

- Now returns `table` as a second output.
- The additional input `'displayopt'` can be used to suppress the ANOVA table display.
- The additional output `stats` can be used as input to `multcompare` to perform multiple comparisons of row or column means.

## Other Updated Functions

### Linear Model Functions

Linear model functions (e.g., `anova1`, `polyval`, etc.) ignore observations with `NaN` value in the `X` or `Y` input.

## betafit

```
[phat,pci] = betafit(x,alpha)
```

The `betafit` function now:

- Removes `NaN` data before fitting
- Issues an error message if there are any `0` or `1` values
- Issues an error message if `x` is constant

## boxplot

```
boxplot(x,notch,sym,vert,whis)
boxplot(x,g,notch,sym,vert,whis)
```

The second syntax for `boxplot` above is new. The first syntax displays a box for each column of the `x` matrix. The second syntax displays a box for each level of the grouping variable `g`. In addition, `g` can be a cell array of grouping variables to produce a separate box for each unique combination of grouping variable levels. See `grpstats`.

## cluster

```
T = cluster(Z,cutoff,depth,flag)
```

The `cluster` function adds a flag argument which overrides the default meaning of the `cutoff` argument. If `flag` is `'inconsistent'`, then `cutoff` is interpreted as a threshold for the inconsistency coefficient. If `flag` is `'clusters'`, then `cutoff` is the maximum number of clusters.

## crosstab

```
[table,chi2,p,labels] = crosstab(col1,col2,...)
```

The `crosstab` function now accepts any number of inputs, not just two. Each input can be a numeric vector, a string array, or a cell array of strings. (In the previous release each input had to be a vector of positive integers taking values 1, ..., *g* for some *g*.) If there are *v* input variables, the output `table` is a *v*-dimensional array, with `table(i,j,k,...)` counting the number of times that the first argument takes its *i*th value, that the second argument takes its *j*th value, that the third argument takes its *k*th value, and so on.

For the case of two positive integer input arguments, the function yields the same results as the previous release unless there are missing integers (i.e., not all of 1, ..., *g* appear in the input). In that case, the previous release would have produced a divide-by-zero warning and would have generated a row or column of zeros in `table`. The new version simply does not consider that category, so it does not reserve zeros for it.

As in the previous release, `chi2` is a chi-square statistic for testing independence, and `p` is its p-value. In this release, `table` can be other than a two-dimensional table, and the test is that all dimensions are independent.

The `labels` output is a cell array with one column for each input argument. The column lists the values of that input. Revisiting the example above, `table(i,j,k,...)` counts the number of times that the first argument takes the value `labels{i,1}`, that the second argument takes the value `labels{j,2}`, that the third argument takes the value `labels{k,3}`, and so on.

### ewmaplot

```
h = ewmaplot(data,lambda,alpha,specs)
```

The `ewmaplot` default for `alpha` changed to 0.27% to conform to the standard ewma chart definition.

### grpstats

```
[means,sem,counts,gname] = grpstats(x,group)
```

The `grpstats` argument `group` is no longer restricted to be a vector of integers. It can be a grouping variable that is a numeric vector, a string matrix, or a cell array of strings. In addition it can be a cell array containing multiple group vectors. The function computes statistics on groups defined by unique combinations of levels of the grouping variables. The new output `gname` is a cell array with one row per group and one column per grouping variable. Elements of `means`, `sem`, and `counts` are statistics calculated for the group defined by values in the corresponding row of `gname`. Examples include

```
[m,s,c] = grpstats(x,g1);
[m,s,c,gnames] = grpstats(x,{g1 g2});
```

### nlinfit

```
[beta,r,J] = nlinfit(X,y,fun,beta0)
```

The `nlinfit` function now accepts inline functions and function handles (`@FF`) in addition to the text strings (`'FF'`) accepted in the past for input `fun`.

### nlintool

```
nlintool(x,y,fun,beta0,alpha,'xname','yname')
```

The interface invoked with the `nlintool` function now:

- Adds a new menu option to compute different types of confidence intervals. Intervals can be simultaneous (provide a specified confidence level over all `x` values simultaneously) or nonsimultaneous (provide that level for a single predetermined `x` value). They can apply to the estimated regression function only (not taking account any variability from a new observation) or to a prediction for a new observation (taking its variability into account).
- Accepts inline functions and function handles (`@FF`) in addition to the text strings (`'FF'`) accepted in the past for input `fun`.

### nlpredci

```
ypred = nlpredci(fun,inputs,beta,r,J,alpha,'simopt','predopt')
```

The `nlpredci` function has new arguments that allow the same types of confidence intervals produced by `nlintool`.

### norminv

```
x = norminv(p,mu,sigma)
```

The `norminv` function now returns `NaN` for each element of `p` that is `NaN`.

### normplot

```
h = normplot(x)
```

The `normplot` function now strips `NaN` values individually from each column of `x`.

### normrnd

```
r = normrnd(mu,sigma,m,n)
```

The `normrnd` function now returns the mean if `sigma` is `0`.

### polytool

```
h = polytool(x,y,n,alpha,xname,yname)
```

The interface invoked by the `polytool` function has the following enhancements:

- Removes `x(j)` and `y(j)` if either is `NaN`, and display a warning when doing so.
- The **Method** menu provides the option of using robust (bisquare) fitting in place of least squares.
- The new **Bounds** menu option computes different types of confidence intervals. Intervals can be simultaneous (provide a specified confidence level over all `x` values simultaneously) or nonsimultaneous (provide that level for a single predetermined `x` value). They can apply to the estimated regression function only (not taking account any variability from a new observation) or to a prediction for a new observation (taking its variability into account).

## prctile

```
 y = prctile(x,p)
```

The `prctile` function now strips `NaN` values individually from each column of `x`.

## qqplot

```
h = qqplot(x,y,pvec)
```

The `qqplot` function now:

- Strips `NaN` values individually from each column of `x` and `y`.
- If `y` is omitted, uses standard normal quantiles.

## ranksum

```
[p,h,stats] = ranksum(x,y,alpha)
```

New `ranksum` output `stats` is a structure that always contains a field named `ranksum` whose value is the value of the rank sum statistic, and that for large samples contains a field named `zval` that is the value of the normal (Z) statistic used to compute the p-value `p`.

## schart

```
[outliers,h] = schart(data,conf,specs)
```

The `schart` default for `conf` changed to 99.73% to conform to the standard s-chart definition.

### signrank

```
[p,h] = signrank(x,y,alpha)
```

The `signrank` function has the following enhancements:

- If `y` is a scalar, extend it to the same length as `x`. This facilitates comparison of the median of one sample to a constant value.
- If `x` and `y` are the same, return `p=1` and `h=0`.
- If `p=alpha`, now `h=1` rather than `h=0` (rejects hypothesis).
- New return value `stats` is a structure that always contains a field named `signed rank` whose value is the value of the signed rank statistic, and that for large samples contains a field named `zval` that is the value of the normal (Z) statistic used to compute the p-value `p`.

### signtest

```
[p,h,stats] = signtest(x,y,alpha)
```

The `signtest` function has the following enhancements:

- If `p=alpha`, now `h=1` rather than `h=0` (rejects hypothesis)
- New return value `stats` is a structure that always contains a field named `sign` whose value is the value of the sign statistic, and that for large samples contains a field named `zval` that is the value of the normal (Z) statistic used to compute the p-value `p`. Z is signed, not an absolute value.

### tcdf, tinv, tpdf, trnd, tstat

The `tcdf`, `tinv`, `tpdf`, `trnd`, and `tstat` functions now accept noninteger degrees of freedom.

### ttest

```
[h,sig,ci,stats] = ttest(x,m,alpha,tail)
```

The `ttest` function has these enhancements:

- Added new output `stats`, the value of the *t* statistic and its degrees of freedom.
- Removes `NaN` from `x` before starting test.

### ttest2

```
[h,sig,ci,stats] = ttest2(x,y,alpha,tail)
```

The `ttest2` function has these enhancements:

- Adds new output `stats`, the value of the *t* statistic and its degrees of freedom.
- If `tail` is `1` or `-1`, now `ci` has one endpoint set to `Inf` or `-Inf`.
- Removes `NaN` from `x` and `y` before starting test.

### weibplot

```
h = weibplot(x)
```

The `weibplot` function strips `NaN` values individually from each column of `x`.

### xbarplot

```
[outliers,h] = xbarplot(data,conf,specs,'sigmaest')
```

The `xbarplot` function has these enhancements:

- Changed default `conf` to 99.73%, to conform to the standard x-bar chart definition.
- Corrected calculation of control limits. With the new default `conf` the control limits are three-sigma limits.
- New input `sigmaest` specifies how to estimate sigma in the control limit calculation. The default is `'std'`, meaning estimate using the average of the subgroup standard deviations. The value `'variance'` uses a pooled variance estimate; this was the value used in previous releases. The value `'range'` uses the average of the subgroup ranges, and requires subgroups with no more than 25 observations.

### ztest

```
[h,sig,ci,zval] = ztest(x,m,sigma,alpha,tail)
```

The `ztest` function has these enhancements:

- Adds new output `zval`, the value of the test statistic.
- If `tail` is `1` or `-1`, now `ci` has one endpoint set to `Inf` or `-Inf`.
- Removes `NaN` from `x` before starting test.

# Symbolic Math Toolbox 2.1.2 Release Notes

**Major Bug Fixes** . . . . . . . . . . . . . . . . . . 45-2

## Major Bug Fixes

The Symbolic Math Toolbox includes several bug fixes. The particularly important bug fixes are described in the online documentation.

# System Identification Toolbox 5.0 Release Notes

**New Features** . . . . . . . . . . . . . . . . . . . . . . 46-2
Object-Based Design . . . . . . . . . . . . . . . . . . . 46-2
Advanced Feature Enhancements . . . . . . . . . . . . . 46-2

**Upgrading from an Earlier Release** . . . . . . . . . 46-4
Theta Models No Longer Supported . . . . . . . . . . . 46-4

## New Features

This section introduces the new features and enhancements added in the System Identification Toolbox 5.0 since the System Identification Toolbox 4.0.5 (Release 11.0).

### Object-Based Design

Based on MATLAB's object technology, the System Identification Toolbox 5.0 provides functions for creating objects directly associated with your models and data. Some quick examples illustrating this feature are

```
z = iddata(y,u,Ts);
sys = pem(z);
```

The new object-based syntax makes it much easier to perform analysis activities beyond what the System Identification Toolbox visual interfaces support. The use of an object-based design by the System Identification Toolbox 5.0 makes it much easier to work with Control System Toolbox objects seamlessly, including converting back and forth between the two toolbox's objects and applying the relevant analysis tools to objects from both toolboxes.

For an overview of the features included in this new object-based approach, type

```
help idhelp
```

You do not need to rewrite any code you wrote using an earlier version of the System Identification Toolbox; the earlier command-line syntax is still supported in the System Identification Toolbox 5.0.

### Advanced Feature Enhancements

The System Identification Toolbox 5.0 includes several enhancements to some of the toolbox's more advanced features:

- Free parameterization for state-space models is now supported. For example, you can simply use

  ```
  m = pem(z,4)
  ```

  to obtain a fourth order state-space model with a well-conditioned parameterization.

- You can now add initial filter conditions. This yields much better performance for slow dynamics. See the `'InitialState'` property of `idmodel` objects for further information.
- You can now use the `SearchDirection` and `Advanced` properties of `idmodel` objects to access several variants of iterative search algorithms. For more information, type

  ```
  idprops algorithm
  ```

- You can now focus the model approximation inherent in system identification to various frequency regions, by using the `Focus` property. The values for the `Focus` property include `'Prediction'`, `'Simulation'`, or any idmodel or LTI object that uses the frequency weighting of that system.

## Upgrading from an Earlier Release

This section describes the upgrade issues involved in moving from the System Identification Toolbox 4.0.5 to the System Identification Toolbox 5.0.

### Theta Models No Longer Supported

Theta models (matrices) are no longer supported in the System Identification Toolbox 5.0. Existing code that uses functions such as `th2par` and `th2ss` to access the theta model data will continue to work in the System Identification Toolbox 5.0. However, if you have code that directly indexes into the theta matrix (e.g., `th(1,3)`), that code will no longer work.

# Wavelet Toolbox 2.0 Release Notes

**New Features** . . . . . . . . . . . . . . . . . . . . . . . 47-2
New Visual Interface Tools . . . . . . . . . . . . . . . . . 47-2
New Wavelets . . . . . . . . . . . . . . . . . . . . . . . 47-5
New Functions . . . . . . . . . . . . . . . . . . . . . . . 47-7
Other Enhancements . . . . . . . . . . . . . . . . . . . . 47-9

**Major Bug Fixes** . . . . . . . . . . . . . . . . . . . . 47-11

**Upgrading from an Earlier Release** . . . . . . . . . 47-12
Default Extension Mode . . . . . . . . . . . . . . . . . 47-12
Wavelet Packet Structures and Obsolete Functions . . . . 47-12

**Known Software and Documentation Problems** . . . 47-13
Graphics and Equations in the HTML Documentation . . 47-13

## New Features

This section introduces the new features and enhancements added in the Wavelet Toolbox 2.0 since the Wavelet Toolbox 1.2 (Release 11.0).

This section is organized into the following subsections:

- “New Visual Interface Tools” on page 47-2
- “New Wavelets” on page 47-5
- “New Functions” on page 47-6
- “Other Enhancements” on page 47-8

### New Visual Interface Tools

The Wavelet Toolbox 2.0 introduces several new visual interfaces to facilitate wavelet analysis.

#### Continuous Wavelet 1-D

The Continuous Wavelet 1-D tool has two new axes.

The first new axis, **Coefficients Line**, displays the coefficient line plot corresponding to a scale selected using the mouse. This functionality is useful for exploring the continuous wavelet coefficients.

The second new axis, **Local Maxima Lines**, displays the coefficient lines corresponding to the chaining across scales of the coefficients local maxima. This functionality gives an interesting skeleton of the continuous wavelet coefficients. An automatic scale-to-frequency translator is now available using the mouse.

#### Complex Continuous Wavelet 1-D

The **Complex Continuous Wavelet 1-D** tool performs continuous wavelet analysis of real signals using complex wavelets. For both the modulus and the angle of the wavelet coefficients, three axes are displayed:

- The coefficients (modulus or angle)
- A coefficient line plot corresponding to a scale selected using the mouse
- The coefficient lines corresponding to the chaining across scales of the local maxima

An automatic scale-to-frequency translator is now available using the mouse. This new tool is useful when the data processing requires careful attention to phase information.

### Signal De-Noising Using SWT 1-D

The **Stationary Wavelet Transform De-noising 1-D** tool performs translation invariant de-noising of signals. This new tool is useful when the signal to be recovered contains some isolated discontinuities. The basic idea is to average many slightly different discrete wavelet de-noising procedures.

### Image De-Noising Using SWT 2-D

The **Stationary Wavelet Transform De-noising 2-D** tool performs translation invariant de-noising of images. This new tool is useful for image de-noising and dramatically improves the performance obtained using simple de-noising strategies. The basic idea is to average many slightly different discrete wavelet de-noising procedures.

### Local Variance Adaptive Thresholding 1-D

This tool allows you to define time-dependent thresholds level by level, thereby increasing the capabilities of the de-noising strategies. A graphical editor helps you define a convenient signal segmentation.

This tool is added to the graphical interfaces involving wavelet or stationary wavelet coefficient thresholding (DWT 1-D De-noising and Compression, SWT 1-D De-noising, Density, and Regression estimation).

This new tool is useful for de-noising signals generated according to the classical noisy model with some kind of nonstationary variance noise.

### Density Estimation

The **Density Estimation 1-D** tool performs wavelet based density estimation. This new tool is useful for data coming from an irregular underlying density function. For this kind of data, wavelet methods overcome the traditional histogram or kernel methods.

### Regression Estimation

The **Regression Estimation 1-D** tool performs wavelet based regression estimation for two different underlying models: the fixed design model, and the random design model. This tool provides a visual interface for exploring some de-noising schemes for equally or unequally sampled data. It is useful for de-noising data observed at unequally spaced time instants or for estimating the nonlinear relationships between two variables. It can also be used to smooth data.

### Wavelet Coefficients Selection 1-D

The **Wavelet Coefficients Selection 1-D** tool performs wavelet reconstruction schemes based on various wavelet coefficient selection strategies:

- *Global* selection of biggest coefficients (in absolute value)
- *By level* selection of biggest coefficients
- *Automatic* selection of biggest coefficients
- *Manual* selection of coefficients

This new tool is useful for stepwise selection of coefficients for manual signal compression.

### Wavelet Coefficients Selection 2-D

The **Wavelet Coefficients Selection 2-D** tool performs wavelet reconstruction schemes based on various wavelet coefficient selection strategies:

- *Global* selection of biggest coefficients (in absolute value)
- *By level* selection of biggest coefficients
- *Automatic* selection of biggest coefficients

This new tool is useful for stepwise selection of coefficients for manual image compression.

### Signal Extension/Truncation

The **Signal Extension** tool performs one-dimensional signal extension using various available extension methods: periodic, symmetric, smooth, and zero-padding. This tool also allows the truncation of a signal.

### Image Extension/Truncation

The **Image Extension** tool performs two-dimensional image extension using various available extension methods: periodic, symmetric, smooth, and zero-padding. This tool also allows the truncation of an image.

### Residuals Display

The **Residuals** tool displays the residuals obtained after a de-noising or compression procedure with some usual frequency diagrams and characteristics, as well as time-series diagrams. This new tool is added to the visual interface to be used with wavelet, wavelet packet, stationary wavelet coefficients, and thresholding.

### Dynamic Visualization Tool

The Dynamic Visualization Tool includes a new toggle button, **View Axes**, that launches a palette for viewing an axis in "full size." Once selected a figure appears and the push buttons assume the position (and relative size) of the axes on the current figure. This functionality is useful when the axes dimensions are small, which happens often in compression and de-noising.

## New Wavelets

**Note** See Chapter 6, "Advanced Topics," in the *Wavelet Toolbox User's Guide* for details about these wavelets.

| **Wavelet (Short Name)** | **Description** |
|---|---|
| `cgau` | Complex Gaussian wavelets (derivatives of the complex Gaussian function) |
| `cmor` | Complex Morlet wavelets |
| `dmey` | Discrete Meyer wavelet (FIR based approximation of the Meyer wavelet) |
| `fbsp` | Complex frequency B-spline wavelets |

| Wavelet (Short Name) | Description |
|---|---|
| gaus | Gaussian wavelets (derivatives of the Gaussian probability density function) |
| rbio | Reverse Biorthogonal wavelets (Reverse Biorthogonal wavelets based on bior) |
| shan | Complex Shannon wavelets |
| sym | Symlets (now available from sym2 to sym45) |

## New Functions

### Stationary Wavelet Transforms

| Function | Description |
|---|---|
| iswt | Inverse discrete stationary wavelet transform 1-D |
| iswt2 | Inverse discrete stationary wavelet transform 2-D |
| swt | Discrete stationary wavelet transform 1-D |
| swt2 | Discrete stationary wavelet transform 2-D |

### Thresholds for 1-D and 2-D De-noising and Compression

| Function | Description |
|---|---|
| wbmpen | Compute threshold for wavelet, 1-D or 2-D, de-noising using penalization strategy |
| wdcbm | Compute thresholds for wavelet 1-D de-noising or compression using Birgé-Massart strategy |
| wdcbm2 | Compute thresholds for wavelet 2-D de-noising or compression using Birgé-Massart strategy |

| Function | Description |
|---|---|
| `wpbmpen` | Compute threshold for wavelet packet, 1-D or 2-D, de-noising using penalization strategy |
| `wthrmngr` | Threshold settings manager |

## General Tree Object Functions

General tree object functions are used for Wavelet Packets 1-D and 2-D. They can also be used in a more general context involving tree objects.

| Function | Description |
|---|---|
| `dtree` | Constructor for the class DTREE (Data Tree) |
| `get` | Get tree objects field contents |
| `ntree` | Constructor for the class NTREE (N-ary Tree) |
| `plot` | Graphical tree-management utility |
| `read` | Read values in tree object fields |
| `set` | Set tree objects field contents |
| `wptree` | Constructor for the class WPTREE (Wavelet Packet Tree) |
| `write` | Write values in tree object fields |

The `plot` function opens a window and displays the tree. This window is a visual tool for splitting, merging, and visualizing each node of the tree.

## Wavelet Utilities

| Function | Description |
|---|---|
| `centfrq` | Compute wavelet center frequency |
| `scal2frq` | Convert scale to frequency |

### Signal Processing Utility

| Function | Description |
|---|---|
| `wvarchg` | Find variance change points |

## Other Enhancements

### Complex CWT

The `cwt` function performs continuous wavelet analysis of a real signal using a real or complex wavelet.

### Threshold Settings Using the Visual Interface

The threshold lines in the de-noising and compression tools can be dragged using the mouse. The corresponding control buttons (edits, sliders, etc.) are automatically updated.

### New De-noising and Compression Methods

New methods for de-noising and compression tools using the visual interface are available in Wavelet 1-D, Wavelet 2-D, Wavelet Packet 1-D, and Wavelet Packet 2-D.

### Saving De-noising and Compression Parameters

In most of the menus, **Save** options are available. When signals, coefficients, or decompositions are obtained using de-noising or compression, the related parameters (wavelet name and thresholds) are now saved when the save operation is performed.

### Length or Size Information

In most windows, a frame containing the analysis parameters displays in the upper right corner. In addition to the signal or image name, the length of the signal or the size of the image now displays.

### Menus and Toolbar

The MATLAB default toolbar and menus displayed in figures are now available in most windows updated with the specific wavelet-oriented features.

## New Extension Modes for DWT

| Command to Set the Mode | Description |
|---|---|
| `dwtmode('sym')` | Set the DWT mode to symmetrization (new default mode, boundary value replication) |
| `dwtmode('zpd')` | Set the DWT mode to zero padding |
| `dwtmode('spd')` or `dwtmode('sp1')` | Set the DWT mode to smooth padding of order 1 (first derivative interpolation at the edges) |
| `dwtmode('sp0')` | Set the DWT mode to smooth padding of order 0 (constant extension at the edges) |
| `dwtmode('ppd')` | Set the DWT mode to periodic padding (periodic extension at the edges) |

The five modes above are applicable for both 1-D and 2-D extensions. The associated DWT is slightly redundant for an arbitrary length signal. The following mode sets the DWT to periodization.

```
dwtmode('per')
```

This option produces the smallest wavelet decomposition. All functions that use the DWT (i.e., Discrete Wavelet (1-D & 2-D) or Wavelet Packet (1-D & 2-D)) use the specified DWT extension mode.

**Changing the default extension mode.** The default extension mode is `'sym'` (boundary value replication). This default can be changed. The default mode is loaded from the file `DWTMODE.DEF` if it exists.

If not, the file `DWTMODE.CFG` (in the `toolbox/wavelet/wavelet` directory) is used.

`dwtmode('save',mode)` saves `mode` as the new default mode in the file `DWTMODE.DEF` (all the files named `DWTMODE.DEF` are deleted before saving).

`dwtmode('save')` is equivalent to `dwtmode('save',currentMode)`.

## Major Bug Fixes

The Wavelet Toolbox includes several bug fixes, including the following descriptions (online only) of particularly important bug fixes.

## Upgrading from an Earlier Release

This section describes the upgrade issues involved in moving from the Wavelet Toolbox 1.2 to the Wavelet Toolbox 2.0.

### Default Extension Mode

The new default extension mode for DWT is `'sym'` (see “New Extension Modes for DWT”). If you want to restore the default mode used in earlier versions, you have to execute the `dwtmode` function. `dwtmode('save','zpd')` saves `'zpd'` extension mode as the default mode in the file `DWTMODE.DEF`.

### Wavelet Packet Structures and Obsolete Functions

`wtbxmngr('V1')` lets you use earlier versions of wavelet packets data structures and functionality, as well as the following obsolete functions: `dwtper`, `idwtper`, `maketree`, `wdatamgr`, `dwtper2`, `idwtper2`, and `plottree`.

The obsolete functions are located in the `$MATLAB/toolbox/wavelet/waveobsolete` directory, where `$MATLAB` is the root installation directory of MATLAB.

`wtbxmngr('V2')` restores the new version of wavelet packet data structures and functionality, and disables the obsolete functions.

---

**Note** Although the Wavelet Toolbox 2.0 maintains some compatibility with earlier versions, you should avoid using `wtbxmngr('V1')`, because future releases may not include it.

---

## Known Software and Documentation Problems

### Graphics and Equations in the HTML Documentation

The Wavelet Toolbox documentation contains some graphics and equations that appear too large in the HTML version of the documentation. Also, some graphics are displayed poorly.

The equations and graphics in the PDF version of these documents display properly.

# xPC Target 1.1 Release Notes

**Introduction to xPC Target and xPC Target Embedded Option** . . . . . . . . . . 48-2
xPC Target . . . . . . . . . . . . . . . . . . . . . . 48-2
xPC Target Embedded Option . . . . . . . . . . . . . . 48-2

**New Features** . . . . . . . . . . . . . . . . . . . . 48-3
Web Browser Interface . . . . . . . . . . . . . . . . 48-3
Target PC Command Line Interface . . . . . . . . . . . 48-3
MATLAB Command Line Interface . . . . . . . . . . . 48-3
Signal Monitoring . . . . . . . . . . . . . . . . . . 48-3
I/O Device Driver Support . . . . . . . . . . . . . . . 48-4

**Known Software and Documentation Problems** . . . . 48-5
Uploading Data with External Mode . . . . . . . . . . . 48-5
Target Command Line Interface . . . . . . . . . . . . 48-5
Memory-Mapped Devices . . . . . . . . . . . . . . . . 48-5

## Introduction to xPC Target and xPC Target Embedded Option

The xPC Target and xPC Target Embedded Option products were introduced as MathWorks products in Release 11.1.

These products have been updated between Release 11.1 and Release 12, as described in the "New Features" section that follows.

**Note** xPC Target 1.1 extends the functionality of the Real-Time Workshop. The xPC Target Embedded Option 1.1 in turn extends the functionality of the xPC Target and requires an additional licence from The MathWorks.

### xPC Target

xPC Target is a host-target PC solution for prototyping, testing, and deploying real-time systems. It is an environment where the host and target computers are different computers.

In this environment you use your desktop PC as a host computer with MATLAB, Simulink, and Stateflow (optional) to create models using Simulink blocks and Stateflow diagrams. After creating a model, you can run simulations in nonreal-time.

You can than use your host computer with Real-Time Workshop®, Stateflow Coder (optional) and a C compiler to create executable code. After creating the executable code, you can run your target application in real time on a second PC compatible system.

### xPC Target Embedded Option

The xPC Target Embedded Option requires an additional license from The MathWorks.

This option allows you to boot the target PC from a device other than a floppy disk drive such as a hard disk drive or flash memory. It also allows you to create stand-alone applications on the target PC independent from the host PC.

# New Features

This section introduces the new features and enhancements added to xPC Target 1.1 and xPC Target Embedded Option 1.1 since xPC Target 1.0 (Release 11.1).

## Web Browser Interface

If the target PC is connected to a network, you can use a Web browser to interact with the target application from any computer connected to a network. Currently, this feature is limited to one browser connection at a time. xPC Target supports Netscape Navigator 4.5, Microsoft Internet Explorer 4.0 and later versions.

## Target PC Command Line Interface

You can interact with the xPC Target environment through the target PC command window. Enter most of the xPC Target commands in the command window on the target PC. This interface is useful with stand-alone applications that are not connected to your host PC.

## MATLAB Command Line Interface

The behavior for the commands `getparamid` and `getsignalid` has been changed. The default behavior is to return the parameter or signal name (P0, P1 . . . or S0, S1, . . .), and you can also return the parameter or signal index (0, 1, 2 . . .) by setting an argument flag.

The behavior for the command `getsignalid` in xPC Target 1.1 is different from xPC Target 1.0 and may break scripts that use this command. The default behavior for the command `getsignalid` with Version 1.0 was to return the signal index.

## Signal Monitoring

This is the process for acquiring signal data without time information. The advantage of this process is that there is minimal CPU overhead for collecting the data.

## I/O Device Driver Support

xPC Target 1.1 includes new device drivers blocks:

- New blocks to support additional I/O boards
- Blocks to support a GPIB fieldbus using a National Instruments GPIB controller and an RS232 connection to the target PC

## Known Software and Documentation Problems

This section updates the xPC Target 1.1 documentation set, reflecting known xPC Target 1.1 software and documentation problems.

### Uploading Data with External Mode

xPC Target does not support uploading data to normal Simulink scope blocks when using Simulink external mode. Instead, use xPC Target scope blocks or the xPC Target graphical user interface (GUI).

### Target Command Line Interface

Visualizing or changing parameter values is limited to scalar parameters.

### Memory-Mapped Devices

Some supported boards in the xPC Target I/O library are memory-mapped devices. These memory-mapped boards are accessed in the address space between 640K and 1M in the lower memory area. xPC Target reserves a 112 kB memory space for memory mapped devices in the address range

```
C0000 - DBFFF
```

Some drivers for memory-mapped devices allow you to select an address range supported by the device, but not supported by xPC Target. For example, the CAN drivers for Softing, allow you to select memory ranges above DBFFF. Select a memory range supported by both the device and xPC Target.