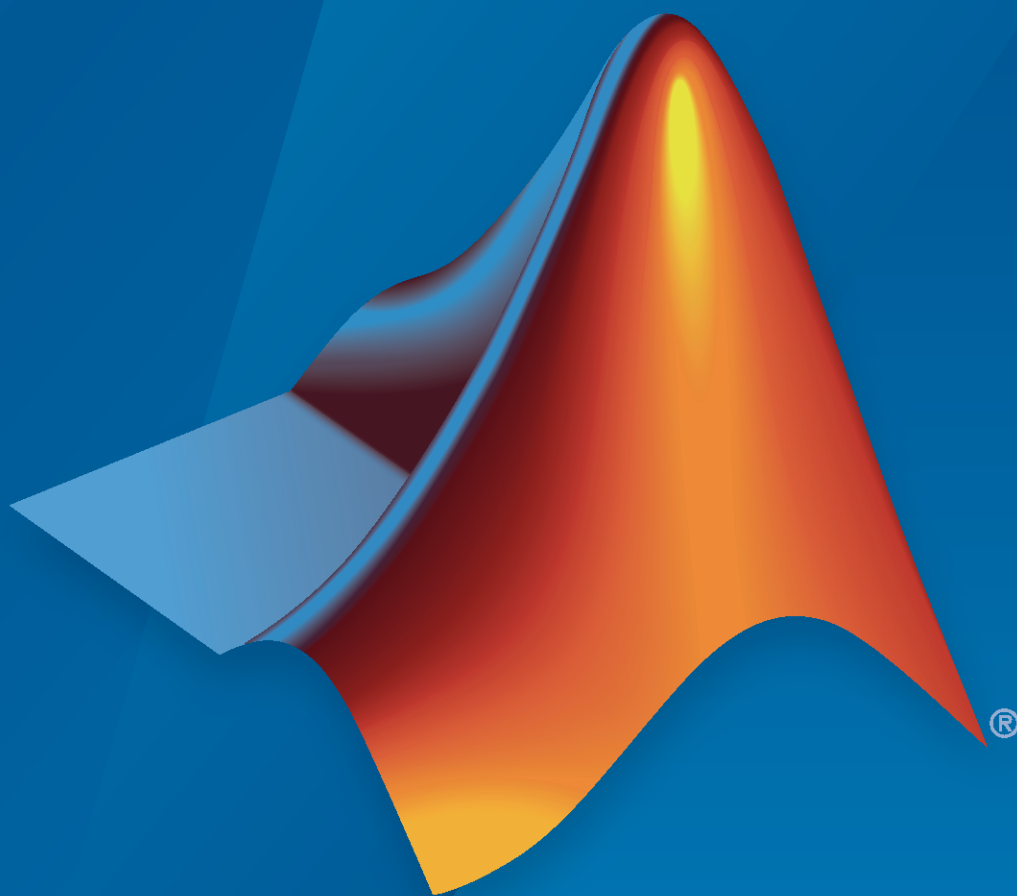


MATLAB® Release Notes



MATLAB®

## How to Contact MathWorks



Latest news: [www.mathworks.com](http://www.mathworks.com)  
Sales and services: [www.mathworks.com/sales\\_and\\_services](http://www.mathworks.com/sales_and_services)  
User community: [www.mathworks.com/matlabcentral](http://www.mathworks.com/matlabcentral)  
Technical support: [www.mathworks.com/support/contact\\_us](http://www.mathworks.com/support/contact_us)



Phone: 508-647-7000



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

### *MATLAB® Release Notes*

© COPYRIGHT 2004-2025 by The MathWorks, Inc.

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

FEDERAL ACQUISITION: This provision applies to all acquisitions of the Program and Documentation by, for, or through the federal government of the United States. By accepting delivery of the Program or Documentation, the government hereby agrees that this software or documentation qualifies as commercial computer software or commercial computer software documentation as such terms are used or defined in FAR 12.212, DFARS Part 227.72, and DFARS 252.227-7014. Accordingly, the terms and conditions of this Agreement and only those rights specified in this Agreement, shall pertain to and govern the use, modification, reproduction, release, performance, display, and disclosure of the Program and Documentation by the federal government (or other entity acquiring for or through the federal government) and shall supersede any conflicting contractual terms or conditions. If this License fails to meet the government's needs or is inconsistent in any respect with federal procurement law, the government agrees to return the Program and Documentation, unused, to The MathWorks, Inc.

### **Trademarks**

MATLAB and Simulink are registered trademarks of The MathWorks, Inc. See [www.mathworks.com/trademarks](http://www.mathworks.com/trademarks) for a list of additional trademarks. Other product or brand names may be trademarks or registered trademarks of their respective holders.

### **Patents**

MathWorks products are protected by one or more U.S. patents. Please see [www.mathworks.com/patents](http://www.mathworks.com/patents) for more information.

R2025b

|                                                                                                                                                    |            |
|----------------------------------------------------------------------------------------------------------------------------------------------------|------------|
| <b>Quality and stability improvements</b>                                                                                                          | <b>1-2</b> |
| <b>getframe Function: Improved performance when capturing frames in loop</b>                                                                       | <b>1-2</b> |
| <b>MATLAB Support Package for Quantum Computing: Solve knapsack and traveling salesperson problems using QUBO (September 2025, Version 25.2.0)</b> | <b>1-2</b> |
| <b>MATLAB Support Package for Quantum Computing: Calculate expectation values using quantum devices (November 2025, Version 25.2.1)</b>            | <b>1-3</b> |

R2025a

|                                                                                                    |            |
|----------------------------------------------------------------------------------------------------|------------|
| <b>Environment</b>                                                                                 | <b>2-2</b> |
| MATLAB Desktop: Access desktop using new layout with customizable sidebars                         | 2-2        |
| Desktop Themes: Change MATLAB desktop colors by selecting dark or light theme                      | 2-3        |
| Desktop Accessibility: Use screen reader and keyboard to interact with different desktop areas     | 2-3        |
| Desktop Search: Access toolbar actions, settings, and Help Center resources using search           | 2-3        |
| Live Editor File Format: Create live scripts using new plain text file format                      | 2-4        |
| Live Editor Videos: Add YouTube videos and video files to your live scripts                        | 2-5        |
| MATLAB Settings: Use Settings window to search for and modify MATLAB settings                      | 2-6        |
| Command Window: Show code suggestions automatically                                                | 2-7        |
| Editor Markdown Support: Preview Markdown files in the Editor                                      | 2-8        |
| Editor Code: Insert code snippets                                                                  | 2-8        |
| Editor Code: Copy code line without selection                                                      | 2-9        |
| Outline Panel: View outline of your code using Outline panel                                       | 2-9        |
| Code Issues Panel: Check code for errors and warnings using Code Issues panel                      | 2-10       |
| Find Files Panel: Search for and replace text in filenames and file content using Find Files panel | 2-10       |

|                                                                                                                          |             |
|--------------------------------------------------------------------------------------------------------------------------|-------------|
| Connections Panel: View and manage your connections using Connections panel .....                                        | 2-11        |
| Add-Ons Panel: Install and manage add-ons using Add-Ons panel .....                                                      | 2-12        |
| Add-Ons Panel: View and install add-ons from MATLAB package repositories .....                                           | 2-12        |
| matlabProcessID Function: Get process identifier for MATLAB code execution process .....                                 | 2-13        |
| Window Managements in MATLAB Online: Use multiple browser tabs for MATLAB Online .....                                   | 2-13        |
| Help Center: Get AI-generated response when searching the web documentation .....                                        | 2-13        |
| Folder Comparison: Compare folders and their subfolders recursively ..                                                   | 2-14        |
| matlab.crashhandling.crashReportFolder Function: Locate crash report folder .....                                        | 2-14        |
| Functionality being removed or changed .....                                                                             | 2-14        |
| <b>Language and Programming .....</b>                                                                                    | <b>2-17</b> |
| Debugging: Manage breakpoints and navigate function call stack using Debugger panel .....                                | 2-17        |
| MATLAB Identifiers: Maximum length of MATLAB identifiers increased to 2048 characters .....                              | 2-17        |
| cell2mat function accepts larger set of data types .....                                                                 | 2-17        |
| onCleanup Object: Cancel cleanup tasks .....                                                                             | 2-18        |
| matlab.lang.Workspace Object: Store workspace variables .....                                                            | 2-18        |
| secretID Object: Keep sensitive information encrypted when setting object properties .....                               | 2-18        |
| Functionality being removed or changed .....                                                                             | 2-19        |
| <b>Data Analysis .....</b>                                                                                               | <b>2-20</b> |
| Variables Editor: View and analyze variable contents using improved interface .....                                      | 2-20        |
| Figure Container: Organize, explore, customize, and generate code for plotted data using an improved figure window ..... | 2-21        |
| allbetween and mustBeBetween Functions: Determine if or validate that all elements are within specified range .....      | 2-21        |
| numunique and allunique Functions: Count and verify unique values ...                                                    | 2-22        |
| unique Function: Specify set order and occurrence arguments together .....                                               | 2-22        |
| table and timetable Data Types: Maximum length of dimension and variable names increased to 2048 characters .....        | 2-22        |
| Moving Statistics Functions: Calculate moving statistics for data in tables and timetables .....                         | 2-23        |
| prctile and quantile Functions: Calculate percentiles and quantiles using inclusive and exclusive methods .....          | 2-23        |
| pivot Function and Pivot Table Live Editor Task: Rearrange data without aggregation .....                                | 2-23        |
| Functionality being removed or changed .....                                                                             | 2-24        |
| <b>Data Import and Export .....</b>                                                                                      | <b>2-25</b> |
| Reading Compressed Data: Read data from compressed and archived files .....                                              | 2-25        |
| File Permissions: View and adjust file permissions .....                                                                 | 2-25        |
| Datastores: Specify multiple remote files using wildcard characters ....                                                 | 2-25        |



|                                                                                                                     |             |
|---------------------------------------------------------------------------------------------------------------------|-------------|
| FTP and SFTP: Specify folders in server hostname .....                                                              | 2-25        |
| xslt Function: Specify XSLT processing engine for transforming an XML file .....                                    | 2-25        |
| HDF5 Interface: Read and write cloud-optimized HDF5 files .....                                                     | 2-25        |
| Import Tool: Import data with timestamps as timetable .....                                                         | 2-26        |
| Image Files: Read and display HEIF and HEIC image files .....                                                       | 2-26        |
| Functionality being removed or changed .....                                                                        | 2-26        |
| <b>Mathematics</b> .....                                                                                            | <b>2-28</b> |
| Sparse Matrix: Create and use single-precision sparse matrices .....                                                | 2-28        |
| ode Object: Solve delay differential equations .....                                                                | 2-28        |
| ode Object: Solve complex-valued functions .....                                                                    | 2-28        |
| Solve ODE Live Editor Task: Detect stiffness and specify minimum step size .....                                    | 2-29        |
| pdepe Function: Specify output function and other odeset options .....                                              | 2-29        |
| pagelsqminnorm Function: Apply Tikhonov regularization to page-wise least-squares solution .....                    | 2-29        |
| Functionality being removed or changed .....                                                                        | 2-29        |
| <b>Graphics</b> .....                                                                                               | <b>2-32</b> |
| Themes: Create graphics using light or dark theme .....                                                             | 2-32        |
| Figure Container: Organize, explore, customize, and generate code for figures using an improved figure window ..... | 2-32        |
| Polar Plots: Plot patches and surfaces in polar coordinates .....                                                   | 2-33        |
| Polar Plots: Specify location of zero degrees as an angle .....                                                     | 2-33        |
| colormaplist Function: List available colormaps .....                                                               | 2-33        |
| nebula Function: Apply blue-to-red colormap to charts .....                                                         | 2-33        |
| Pie Charts and Donut Charts: Choose text interpreter or no interpreter for titles and labels .....                  | 2-34        |
| violinplot Function: Use additional grouping data options and customize color group appearance .....                | 2-34        |
| boxchart Function: Create box charts using table data and customize box charts using additional properties .....    | 2-34        |
| Swarm Charts: Create swarm charts with jitter in different directions ..                                            | 2-34        |
| Geographic Plots: Basemaps have improved appearance .....                                                           | 2-35        |
| tightPosition Function: Get location and size of axes plotting area in specific units .....                         | 2-36        |
| exportgraphics and copygraphics Functions: Specify output dimensions using name-value arguments .....               | 2-36        |
| exportgraphics Function: Export plots as SVG files .....                                                            | 2-36        |
| getframe Function in MATLAB Online: Capture figures in MATLAB Online sessions .....                                 | 2-36        |
| Copying Figures: Select options for copying figures in MATLAB Settings .....                                        | 2-36        |
| Titles and Labels: Use LaTeX to display math and symbols with more consistent results .....                         | 2-37        |
| Text: Larger, easier-to-read text on macOS .....                                                                    | 2-37        |
| Axes Interactions: Specify axes for interaction mode .....                                                          | 2-37        |
| Accessibility in MATLAB: Use screen reader to interact with figures ....                                            | 2-38        |
| Graphics functionality being removed or changed .....                                                               | 2-38        |
| Printing and exporting functionality being removed or changed .....                                                 | 2-43        |
| <b>App Building</b> .....                                                                                           | <b>2-46</b> |

|                                                                                                                                                               |             |
|---------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------|
| Themes: Create apps and custom UI components using light or dark theme                                                                                        | 2-46        |
| uiprintdlg and uiexportdlg Functions: Open dialog boxes for printing and exporting figures                                                                    | 2-46        |
| uislider Function: Specify step size for slider                                                                                                               | 2-46        |
| uitable Function: View multiline content in table cells                                                                                                       | 2-47        |
| UI Components: Interact with UI components using keyboard                                                                                                     | 2-47        |
| UI Components: Replace UIControl objects with UI components                                                                                                   | 2-47        |
| App Designer: Package and share apps more efficiently                                                                                                         | 2-47        |
| App Designer: Access toolstrip actions and Help Center resources using search                                                                                 | 2-48        |
| Text: Larger, easier-to-read text on macOS                                                                                                                    | 2-48        |
| Functionality being removed or changed                                                                                                                        | 2-49        |
| <b>Performance</b>                                                                                                                                            | <b>2-57</b> |
| datetime Format Parsing: Improved performance when parsing common date and time formats                                                                       | 2-57        |
| String Operations: Improved performance with small string arrays                                                                                              | 2-58        |
| writetable, writetimetable, writematrix, and writecell Functions: Improved performance when writing to existing sheets or writing new sheets with sheet names | 2-58        |
| parquetread Function: Improved performance when reading repeated string values                                                                                | 2-59        |
| nufft Function: Improved performance with nonuniform sample points or query points                                                                            | 2-60        |
| mink, maxk, and topkrows Functions: Improved performance when returning a large number of rows for numeric or logical data                                    | 2-61        |
| fillmissing Function: Improved performance when filling entries using linear interpolation                                                                    | 2-62        |
| groupsummary Function: Improved performance when applying multiple computation methods                                                                        | 2-62        |
| uifigure Function: Improved performance when creating UI figure                                                                                               | 2-63        |
| Property Inspector: Improved performance when opening for the first time                                                                                      | 2-63        |
| Axes Toolbar: Improved performance for single Cartesian axes when opening for the first time                                                                  | 2-64        |
| Plot Interactions: Improved responsiveness when interacting with scatter plots that have circle markers                                                       | 2-64        |
| Plot Interactions: Improved responsiveness when panning or zooming within line plots                                                                          | 2-65        |
| Animations in Apps and MATLAB Online: Improved performance for animations in loops                                                                            | 2-66        |
| Plots in Apps: Improved performance when creating multiple legends                                                                                            | 2-67        |
| Operations on Windows 11 Platforms with Multiple Processor Groups: Improved performance                                                                       | 2-67        |
| <b>Software Development Tools</b>                                                                                                                             | <b>2-69</b> |
| Project Panel: Manage project environment and view issues using improved interface                                                                            | 2-69        |
| Project References: Componentize large projects with improved usability                                                                                       | 2-70        |
| Project Comparison: Compare MATLAB projects                                                                                                                   | 2-70        |
| Dependency Analyzer: View project hierarchy and issues                                                                                                        | 2-70        |
| Dependency Analyzer: Analyze dependencies in referenced projects                                                                                              | 2-70        |

|                                                                                                             |             |
|-------------------------------------------------------------------------------------------------------------|-------------|
| Project API: List issues on project startup and shutdown . . . . .                                          | 2-71        |
| Project API: Add all folders and files to project . . . . .                                                 | 2-71        |
| Project API: Analyze file dependencies in referenced projects . . . . .                                     | 2-71        |
| Project API: Process multiple project files simultaneously . . . . .                                        | 2-71        |
| Project API: Improved performance when interacting with project<br>programmatically . . . . .               | 2-71        |
| Code Analyzer: Configure Custom Naming Conventions for MATLAB<br>Identifiers . . . . .                      | 2-72        |
| Source Control Panel: Work with multiple folders and projects under Git and<br>SVN source control . . . . . | 2-72        |
| Source Control: Check for unsaved files before commit . . . . .                                             | 2-73        |
| Git Source Control: View line-by-line revision history for entire MATLAB file<br>. . . . .                  | 2-73        |
| Git Source Control: Create shallow clone of Git repository . . . . .                                        | 2-73        |
| Git Source Control: Share local repository to GitHub . . . . .                                              | 2-74        |
| Git Source Control: Ignore files interactively . . . . .                                                    | 2-74        |
| Git Source Control: Cherry-pick or squash Git commits, and rebase Git<br>branches . . . . .                 | 2-74        |
| Git Source Control: Create, merge, and manage Git branches and remotes<br>with improved usability . . . . . | 2-74        |
| Git Source Control: Work with submodules in MATLAB with improved<br>usability . . . . .                     | 2-74        |
| Git Source Control: Check out Git branch to separate folder using worktrees<br>in MATLAB . . . . .          | 2-74        |
| Git Source Control: Use avatars and verify commit signatures in MATLAB<br>. . . . .                         | 2-74        |
| Git Source Control: MATLAB supports Git LFS process filter . . . . .                                        | 2-75        |
| Git API: List modified files between commits or branches and save file<br>revisions to disk . . . . .       | 2-75        |
| Git API: Add and remove remotes . . . . .                                                                   | 2-75        |
| SVN Source Control: Remember credentials for SVN repositories . . . . .                                     | 2-76        |
| Comparison and Merge Tools: View tag and branch details for files under<br>source control . . . . .         | 2-76        |
| Build Automation: Create custom reusable tasks . . . . .                                                    | 2-76        |
| Build Automation: Run tasks interactively from Editor or MATLAB project<br>. . . . .                        | 2-76        |
| Build Automation: Expanded support for MATLAB projects . . . . .                                            | 2-76        |
| Build Automation: Display why tasks run . . . . .                                                           | 2-77        |
| Build Automation: Create group of MexTask instances using forEachFile<br>static method . . . . .            | 2-77        |
| Build Automation: Filter test suite using selectors . . . . .                                               | 2-77        |
| Unit Testing Framework: Rerun tests from most recent test run using Test<br>Browser . . . . .               | 2-77        |
| Unit Testing Framework: Identify most recent test results in Test Browser<br>. . . . .                      | 2-77        |
| Unit Testing Framework: Control whether to use Test Browser from<br>toolbar . . . . .                       | 2-78        |
| Unit Testing Framework: Create default test runner using<br>withDefaultPlugins static method . . . . .      | 2-78        |
| Toolbox Packaging: Create and package toolboxes using projects . . . . .                                    | 2-78        |
| Extension Points: Extend the MATLAB desktop using extension points . . . . .                                | 2-78        |
| Functionality being removed or changed . . . . .                                                            | 2-79        |
| <b>External Language Interfaces . . . . .</b>                                                               | <b>2-80</b> |

|                                                                                                                  |             |
|------------------------------------------------------------------------------------------------------------------|-------------|
| Call C++ from MATLAB: Pass MATLAB structs to and from C++ functions                                              | 2-80        |
| Call C++ from MATLAB: Pass environment variables in out-of-process mode                                          | 2-80        |
| Call C++ from MATLAB: Stop out-of-process execution using Ctrl+C                                                 | 2-80        |
| Call MATLAB from C++: Create and use single-precision sparse matrices                                            | 2-80        |
| C MAT File API: Maximum length of variable names increased to 2048 characters                                    | 2-81        |
| Call MATLAB from Python: Automatically convert between Python and MATLAB datetime and duration types             | 2-81        |
| Call MATLAB from Python: Automatically convert between MATLAB tables, timetables, and Python Pandas DataFrames   | 2-81        |
| Call Python from MATLAB: Switch execution modes without restarting MATLAB                                        | 2-82        |
| Call .NET from MATLAB: Save .NET objects to MAT files                                                            | 2-82        |
| Call .NET from MATLAB: Select specific numerical version of .NET                                                 | 2-82        |
| Web Services: Keep sensitive information encrypted when using weboptions and matlab.net.http.Credentials objects | 2-82        |
| MATLAB Packages: Java files supported in packages                                                                | 2-82        |
| Functionality being removed or changed                                                                           | 2-82        |
| <b>Hardware Support</b>                                                                                          | <b>2-84</b> |
| Raspberry Pi Hardware: Support for 64-bit Debian 12 Bookworm and 64-bit Debian 11 Bullseye                       | 2-84        |
| Arduino Hardware: Support for Arduino UNO R4 WiFi and Minima boards in MATLAB Online                             | 2-84        |
| Arduino Hardware: Support for Servo Motors and Sensors with Arduino UNO R4 WiFi and Minima Boards                | 2-84        |
| Arduino Hardware: Enhancing voltage data acquisition capabilities with readVoltage function                      | 2-84        |
| Functionality being removed or changed                                                                           | 2-84        |

## R2024b

|                                                                                                                            |            |
|----------------------------------------------------------------------------------------------------------------------------|------------|
| <b>Environment</b>                                                                                                         | <b>3-2</b> |
| Help Center: View documentation in your system web browser                                                                 | 3-2        |
| Live Editor Fonts: Customize font, size, color, and formatting of text and code styles                                     | 3-2        |
| Live Editor Controls: Add range sliders to live scripts                                                                    | 3-3        |
| Live Editor Controls: Populate drop-down list items using additional variable types                                        | 3-3        |
| Live Editor Controls: Restore default values for all controls                                                              | 3-4        |
| help Function: Get updated help text for functions, methods, and classes                                                   | 3-4        |
| Editor in MATLAB Online: View outline of your code using Outline panel                                                     | 3-4        |
| Editor TLC Support in MATLAB Online: View and edit TLC files with syntax highlighting and automatic indenting while typing | 3-5        |

|                                                                                                                                      |             |
|--------------------------------------------------------------------------------------------------------------------------------------|-------------|
| Add-Ons in MATLAB Online: Update add-ons directly from Add-Ons panel .....                                                           | 3-5         |
| Find Files in MATLAB Online: Manage list of search results and export as plain text .....                                            | 3-5         |
| New Desktop for MATLAB (Beta): Try new features including dark theme, an updated layout, and expanded search capabilities .....      | 3-5         |
| Functionality being removed or changed .....                                                                                         | 3-6         |
| <b>Language and Programming .....</b>                                                                                                | <b>3-8</b>  |
| dbstop Function: Find missing semicolons by pausing execution on unsuppressed output .....                                           | 3-8         |
| Error Messages: Error text now indicates specific location in code .....                                                             | 3-8         |
| Debugging in MATLAB Online: View error breakpoint messages in Debugger panel .....                                                   | 3-9         |
| matlab.lang.obfuscateNames Function: Obfuscate names of local variables, local functions, and nested functions .....                 | 3-9         |
| MATLAB Vault: Access and identify existing MATLAB secrets in parallel environments .....                                             | 3-9         |
| Validation Functions: Validate arguments with matrix and vector shapes .....                                                         | 3-10        |
| Handle Objects: Define weak references for handle objects .....                                                                      | 3-10        |
| Object Serialization: Customize how objects are serialized and deserialized .....                                                    | 3-10        |
| matlab.codeanalysis.validateConfiguration Function: View severity of issues in configuration file .....                              | 3-10        |
| typecast Function: Convert data types using "like" syntax, and pass logical, character vector, and complex inputs .....              | 3-11        |
| matlab.indexing.isScalarClass Function: Identify scalar classes .....                                                                | 3-11        |
| Class Diagram Viewer: Import classes from MATLAB packages .....                                                                      | 3-11        |
| Functionality being removed or changed .....                                                                                         | 3-11        |
| <b>Data Analysis .....</b>                                                                                                           | <b>3-14</b> |
| summary Function: Summarize array data and configure summary contents .....                                                          | 3-14        |
| isbetween Function: Specify numeric, nonnumeric, or tabular input data .....                                                         | 3-14        |
| rmmissing and rmoutliers Functions: Define locations of missing entries or outliers using array or table .....                       | 3-14        |
| Clean Outlier Data Live Editor Task and Data Cleaner App: Define outliers using range or workspace variable .....                    | 3-15        |
| Experiment Manager: Improved interface for experiment setup and result analysis .....                                                | 3-15        |
| table and timetable Data Types: Specify the data types of table and timetable variables .....                                        | 3-15        |
| timeseries2timetable Function: Convert events associated with timeseries objects to an event table associated with a timetable ..... | 3-15        |
| Retime Timetable and Synchronize Timetables Live Editor Tasks: Specify custom function as local function or function handle .....    | 3-16        |
| Variables Editor in MATLAB Online: View variables with enhanced readability .....                                                    | 3-16        |
| Functionality being removed or changed .....                                                                                         | 3-16        |
| <b>Data Import and Export .....</b>                                                                                                  | <b>3-18</b> |

|                                                                                                                                               |             |
|-----------------------------------------------------------------------------------------------------------------------------------------------|-------------|
| Reading Online Data: Read remote data over HTTP and HTTPS using file operation, low-level I/O, datastore, video, and HDF5 functions . . . . . | 3-18        |
| JSON: Read and write dictionaries in JSON files . . . . .                                                                                     | 3-18        |
| Parquet: Import Parquet MAP data . . . . .                                                                                                    | 3-18        |
| ZIP Files: Compress and extract password-protected files . . . . .                                                                            | 3-18        |
| Symbolic Links: Create and use symbolic links to manipulate files and folders . . . . .                                                       | 3-18        |
| Spreadsheet Files: Specify how to read merged cells from spreadsheets . . . . .                                                               | 3-19        |
| Image Files: Automatically orient JPEG and TIFF images . . . . .                                                                              | 3-19        |
| sonify Function: Convert numeric data to sound . . . . .                                                                                      | 3-19        |
| VideoReader Object: Read remotely stored video files . . . . .                                                                                | 3-19        |
| FTP and SFTP: Specify time-out values for connections and transfers . . .                                                                     | 3-19        |
| FTP: Enable TLS when connecting to FTP servers . . . . .                                                                                      | 3-19        |
| SFTP: Connect to servers that require passphrases or certificates . . . . .                                                                   | 3-20        |
| HDF5 Interface: Use SZIP filter to read and write compressed datasets . .                                                                     | 3-20        |
| HDF5 Interface: Work with file integrity checks (R2024b Update) . . . . .                                                                     | 3-20        |
| HDF Vgroup Interface: Define longer vgroup names and vgroup class names . . . . .                                                             | 3-20        |
| Scientific File Format Libraries: HDF5 library upgraded to version 1.14.4.3 (R2024b Update) . . . . .                                         | 3-20        |
| Functionality being removed or changed . . . . .                                                                                              | 3-21        |
| <b>Mathematics . . . . .</b>                                                                                                                  | <b>3-23</b> |
| Solve ODE Live Editor Task: Interactively solve systems of ordinary differential equations . . . . .                                          | 3-23        |
| ode Object: Specify ODE as implicit . . . . .                                                                                                 | 3-23        |
| ode Object: Detect stiffness to change solver after creating ode object . .                                                                   | 3-23        |
| ode Options: Set minimum step size for several ODE solvers . . . . .                                                                          | 3-23        |
| lsqminnorm Function: Apply Tikhonov regularization to least-squares solution . . . . .                                                        | 3-23        |
| isapprox Function: Determine approximate equality of elements in two arrays . . . . .                                                         | 3-24        |
| layoutcoords Function: Calculate node and edge coordinates for the layout of a graph or digraph . . . . .                                     | 3-24        |
| polyshape Object: Control whether subtract and xor methods simplify output . . . . .                                                          | 3-24        |
| MATLAB Support Package for Quantum Computing: Solve QUBO problems using QAOA (November 2024, Version 24.2.2) . . . . .                        | 3-24        |
| MATLAB Support Package for Quantum Computing: Calculate expected value and convert QUBO to Ising (February 2025, Version 24.2.4) . . .        | 3-24        |
| Functionality being removed or changed . . . . .                                                                                              | 3-25        |
| <b>Graphics . . . . .</b>                                                                                                                     | <b>3-26</b> |
| violinplot Function: Visualize grouped numeric data by using violin plots . . . . .                                                           | 3-26        |
| compassplot Function: Create one or more compass plots in polar axes .                                                                        | 3-26        |
| constantplane Function: Create infinite planes . . . . .                                                                                      | 3-27        |
| Legends: Control width of icons . . . . .                                                                                                     | 3-28        |
| Pie Charts and Donut Charts: Control slice order and number of slices . .                                                                     | 3-29        |
| Bar Charts: Add labels to ends of bars . . . . .                                                                                              | 3-29        |
| ConstantLine Object: Control text label color . . . . .                                                                                       | 3-30        |
| tilayout Function: Create layouts without specifying dimensions or tile arrangement . . . . .                                                 | 3-31        |

|                                                                                                                                      |             |
|--------------------------------------------------------------------------------------------------------------------------------------|-------------|
| savefig Function: Specify MAT file version .....                                                                                     | 3-31        |
| Name-Value Arguments: Set properties when you call pcolor, sphere,<br>cylinder, and other plotting functions .....                   | 3-31        |
| Graphics in MATLAB Online: Copy images to clipboard .....                                                                            | 3-32        |
| Functionality being removed or changed .....                                                                                         | 3-32        |
| <b>App Building .....</b>                                                                                                            | <b>3-36</b> |
| uibutton and uitogglebutton Functions: Specify HTML markup and display<br>equations in button text .....                             | 3-36        |
| App Designer: Add UI components that span multiple grid cells more easily<br>.....                                                   | 3-36        |
| App Designer: View errors directly in Code View when debugging an app<br>.....                                                       | 3-37        |
| App Designer: Convert between text edit fields and numeric edit fields ..                                                            | 3-37        |
| App Testing Framework: Programmatically interact with alert and<br>confirmation dialog boxes .....                                   | 3-37        |
| App Testing Framework: Specify location of gesture on figures that use<br>nonpixel units .....                                       | 3-37        |
| Functionality being removed or changed .....                                                                                         | 3-37        |
| <b>Performance .....</b>                                                                                                             | <b>3-39</b> |
| datetime Format Parsing: Improved performance when parsing common<br>formats that include localized names or time zone offsets ..... | 3-39        |
| median Function: Improved performance when computing weighted median<br>.....                                                        | 3-40        |
| fillmissing Function: Improved performance when filling numeric entries<br>using nearest neighbor rows .....                         | 3-40        |
| isprime Function: Improved performance when determining which array<br>elements are prime .....                                      | 3-41        |
| simplify Function: Improved performance when simplifying polyshape<br>objects with many boundaries .....                             | 3-41        |
| waitbar Function: Improved performance when updating wait bar dialog<br>box in loop .....                                            | 3-42        |
| Plot Animations: Improved performance when updating charts that have<br>legends in loop .....                                        | 3-42        |
| Plots in Apps: Improved performance when creating surface plots of large<br>data sets .....                                          | 3-43        |
| Create Plot Live Editor Task: Improved loading performance .....                                                                     | 3-44        |
| <b>Software Development Tools .....</b>                                                                                              | <b>3-45</b> |
| Project API: Specify dependency analysis scope .....                                                                                 | 3-45        |
| Project API: Label multiple project files at once .....                                                                              | 3-45        |
| Dependency Analyzer: Find required packages and missing package<br>dependencies .....                                                | 3-45        |
| Source Control in MATLAB Online: Expanded support for Git workflows<br>.....                                                         | 3-46        |
| Project Issues in MATLAB Online: View and fix project startup and<br>shutdown issues .....                                           | 3-46        |
| Code Analyzer: Check code for specified variable names .....                                                                         | 3-46        |
| Code Analyzer in MATLAB Online: Check code for info messages .....                                                                   | 3-46        |
| Build Automation: Create and run group of tasks .....                                                                                | 3-47        |
| Build Automation: Specify task names more flexibly .....                                                                             | 3-47        |

|                                                                                                          |             |
|----------------------------------------------------------------------------------------------------------|-------------|
| Build Automation: Produce code coverage results using coverage formats                                   | 3-47        |
| Build Automation: Specify build file and verbosity level                                                 | 3-48        |
| Build Automation: Load plan from folder into workspace                                                   | 3-48        |
| Build Automation: Delete additional task outputs                                                         | 3-48        |
| Unit Testing Framework: Run function in environment provided by fixtures                                 | 3-48        |
| Unit Testing Framework: Control test runner verbosity using testrunner function                          | 3-48        |
| Unit Testing Framework: Add tests from currently open test file or clear test results using Test Browser | 3-49        |
| Unit Testing Framework: Control whether to use Test Browser from toolstrip in MATLAB Online              | 3-49        |
| App Testing Framework: Programmatically interact with alert and confirmation dialog boxes                | 3-49        |
| App Testing Framework: Specify location of gesture on figures that use nonpixel units                    | 3-49        |
| MATLAB Package Manager: Organize and distribute code using packages                                      | 3-49        |
| Functionality being removed or changed                                                                   | 3-50        |
| <b>External Language Interfaces</b>                                                                      | <b>3-52</b> |
| REST Function Service: Automatic restart after MathWorks Service Host update                             | 3-52        |
| Call C++ from MATLAB: Convert MATLAB structures to C++ struct arrays when passing data from MATLAB       | 3-52        |
| Call MATLAB from C++: Create MATLAB data arrays from user-managed buffers                                | 3-52        |
| Call MATLAB from C++: Create matlab::data::CharArray from UTF-8 input                                    | 3-52        |
| Java: Support for OpenJDK 21 Java                                                                        | 3-52        |
| Python: Support for CPython version 3.12                                                                 | 3-53        |
| Call Python from MATLAB: Follow Python type hints when passing data from MATLAB                          | 3-53        |
| Call MATLAB from Python: Convert between Python and MATLAB dictionaries                                  | 3-53        |
| .NET: Cross-platform support for .NET 6 and higher                                                       | 3-53        |
| .NET: Support for .NET 8                                                                                 | 3-53        |
| Compilers: Support for Intel C++ icx and Fortran ifx                                                     | 3-53        |
| <b>Hardware Support</b>                                                                                  | <b>3-54</b> |
| Arduino Hardware: Support for Arduino UNO R4 WiFi and Minima boards                                      | 3-54        |
| Arduino Hardware: Use MATLAB Support Package for Arduino Hardware in MATLAB Online through web browser   | 3-54        |
| Raspberry Pi Hardware: Support for Debian 12 Bookworm                                                    | 3-54        |
| Raspberry Pi Hardware: libcamera support on camera boards with Bullseye and Bookworm operating systems   | 3-54        |
| Raspberry Pi Hardware: Support for Raspberry Pi 5                                                        | 3-55        |
| Raspberry Pi Hardware: Support for Camera with Raspberry Pi 5 (December 2024 Version 24.2.3)             | 3-55        |
| Functionality being removed or changed                                                                   | 3-55        |



|                                                                                                                                       |            |
|---------------------------------------------------------------------------------------------------------------------------------------|------------|
| <b>Environment</b>                                                                                                                    | <b>4-2</b> |
| Editor Spell Checker: Check spelling in text and comments in MATLAB code files                                                        | 4-2        |
| Live Editor Export: Interactively customize export options when exporting to Markdown files and Jupyter notebooks                     | 4-2        |
| Live Editor Controls: Add date pickers to live scripts                                                                                | 4-2        |
| Live Editor Controls: Replace with similar controls                                                                                   | 4-2        |
| Live Editor Tasks: Create Live Editor task class from template                                                                        | 4-3        |
| Live Editor Output: View table, timetable, structure array, and cell array output with enhanced readability                           | 4-3        |
| Find and Replace Dialog Box: Disable wrap-around search                                                                               | 4-3        |
| Editor Indenting: Configure automatic indenting behavior                                                                              | 4-4        |
| Editor YAML and Markdown Support: View and edit YAML and Markdown files with syntax highlighting and automatic indenting while typing | 4-4        |
| Accessibility in MATLAB Online: Access the Command Window scroll buffer region using the keyboard                                     | 4-4        |
| Accessibility in MATLAB Online: Use a screen reader to create and edit live scripts and functions in the Live Editor                  | 4-5        |
| Add-Ons in MATLAB Online: Install and manage add-ons using Add-Ons panel                                                              | 4-5        |
| Cloud Storage in MATLAB Online: View your connections using Connections panel                                                         | 4-5        |
| Cloud Storage in MATLAB Online: Access your OneDrive files (June 2024)                                                                | 4-5        |
| Editor in MATLAB Online: Resolve conflicts with externally modified files                                                             | 4-5        |
| Editor in MATLAB Online: Reopen last closed file                                                                                      | 4-6        |
| Find Files in MATLAB Online: Search for and replace text in filenames and file content                                                | 4-6        |
| MATLAB Drive: Get recipient suggestions when sharing (April 2024)                                                                     | 4-6        |
| MATLAB Drive: Updated user interface with collapsible sidebar (May 2024)                                                              | 4-6        |
| Product Version Number: All products now use consistent version number based on release                                               | 4-6        |
| Functionality being removed or changed                                                                                                | 4-6        |
| <b>Language and Programming</b>                                                                                                       | <b>4-8</b> |
| Local Functions: Define functions anywhere in scripts and live scripts                                                                | 4-8        |
| createArray Function: Generate arrays of specified class and size                                                                     | 4-8        |
| MATLAB Language Terminology: Packages are now called namespaces                                                                       | 4-8        |
| Metaclass Renaming: Metaclasses and class members have new names                                                                      | 4-8        |
| Class Diagram Viewer: Show associations for classes                                                                                   | 4-9        |
| Class Constructor: <code>?.MyClass</code> syntax accepts immutable properties as name-value arguments                                 | 4-9        |
| MATLAB backgroundPool Function: save function supported in background threads                                                         | 4-10       |
| hasActiveValue Function: Verify that setting has active value                                                                         | 4-10       |
| MATLAB Debugger: View private, protected, and hidden class property values using display and details                                  | 4-10       |

|                                                                                                                            |             |
|----------------------------------------------------------------------------------------------------------------------------|-------------|
| Function Input Arguments: Automatically add input arguments to arguments block .....                                       | 4-11        |
| Validation Functions: mustBeUnderlyingType accepts multiple data types .....                                               | 4-11        |
| Functionality being removed or changed .....                                                                               | 4-11        |
| <b>Data Analysis .....</b>                                                                                                 | <b>4-14</b> |
| islocalmax2 and islocalmin2 Functions: Find local extrema in two dimensions .....                                          | 4-14        |
| clip Function: Clip values to specified range .....                                                                        | 4-14        |
| mean and median Functions: Compute weighted statistics .....                                                               | 4-14        |
| iqr Function: Return first and third quartiles .....                                                                       | 4-14        |
| Experiment Manager: Apply and reset filters for numeric, character array, and string variables in results table .....      | 4-14        |
| Experiment Manager: Add supporting files .....                                                                             | 4-15        |
| Live Editor Output: View sparklines and summary statistics for tables ..                                                   | 4-15        |
| Smooth Data Live Editor Task: Return moving window size .....                                                              | 4-16        |
| fillmissing and filloutliers Functions: Define missing or outlier locations using table .....                              | 4-16        |
| Group-Wise Computations: Apply multiple binning methods to grouping variable .....                                         | 4-16        |
| datetime Data Type: Specify time zone offset using duration value .....                                                    | 4-17        |
| categories Function: Return list of categories as categorical array or string array .....                                  | 4-17        |
| Unstack Table Variables Live Editor Task: Generate template code for custom aggregation function .....                     | 4-17        |
| <b>Data Import and Export .....</b>                                                                                        | <b>4-18</b> |
| Secrets in MATLAB Vault: Remove sensitive information from code .....                                                      | 4-18        |
| SFTP: Specify remote current working folder at login .....                                                                 | 4-18        |
| openedFiles Function: Get file identifiers of all open files .....                                                         | 4-18        |
| HDF5 Interface: Import HDF5 data using Import Tool in MATLAB Online and Import Data Live Editor task .....                 | 4-18        |
| HDF5 Interface in MATLAB Online: Import HDF5 data by double-clicking HDF5 files .....                                      | 4-18        |
| Serial and TCP/IP Connections: Label and Find Connections .....                                                            | 4-19        |
| Image File Format Libraries: LibTIFF library upgraded to version 4.6.0 ..                                                  | 4-19        |
| Scientific File Format Libraries: HDF4 library upgraded to version 4.2.16.2 .....                                          | 4-19        |
| Scientific File Format Libraries: HDF5 library upgraded to version 1.10.11 .....                                           | 4-19        |
| Functionality being removed or changed .....                                                                               | 4-19        |
| <b>Mathematics .....</b>                                                                                                   | <b>4-20</b> |
| ode Object: Solve ODEs and perform sensitivity analysis using SUNDIALS solvers .....                                       | 4-20        |
| pagelsqminnorm Function: Calculate minimum-norm least-squares solutions to systems of linear equations in N-D arrays ..... | 4-20        |
| pagepinv Function: Calculate Moore-Penrose pseudoinverses of pages of N-D array .....                                      | 4-20        |
| scatteredInterpolant Object: Extend interpolation boundary into extrapolation domain .....                                 | 4-20        |
| polydiv Function: Divide polynomials using long division .....                                                             | 4-20        |

|                                                                                                       |             |
|-------------------------------------------------------------------------------------------------------|-------------|
| polyfit Function: Return R-squared value .....                                                        | 4-21        |
| polyshape Object: Control when union and intersect methods simplify<br>output .....                   | 4-21        |
| spdiags Function: Specify diagonal elements as matrix, vector, or scalar<br>.....                     | 4-21        |
| sub2ind Function: Specify subscript inputs as scalars and vectors .....                               | 4-21        |
| Functionality being removed or changed .....                                                          | 4-21        |
| <b>Graphics .....</b>                                                                                 | <b>4-22</b> |
| fpolarplot Function: Plot mathematical expressions in polar coordinates<br>.....                      | 4-22        |
| Polar Plots: Create filled regions in polar coordinates .....                                         | 4-22        |
| linestyleorder Function: Control line styles when plotting multiple data<br>series .....              | 4-23        |
| rgb2hex and hex2rgb Functions: Convert between RGB triplets and<br>hexadecimal color codes .....      | 4-24        |
| Quiver Plots: Access the multiplier used to scale arrows .....                                        | 4-24        |
| Legends and Rectangles: Control transparency of legend boxes and<br>rectangle faces .....             | 4-24        |
| ConstantLine and ConstantRegion Objects: Layer reference lines and filled<br>regions in plots .....   | 4-25        |
| Bar Charts: Control bar group width .....                                                             | 4-26        |
| Donut and Pie Charts: Label charts in different ways .....                                            | 4-26        |
| DatetimeRuler Object: Specify date format for secondary labels when<br>plotting datetime data .....   | 4-27        |
| clf, cla, and clim Functions: Pass array of objects as input .....                                    | 4-27        |
| hold, axis, box, and grid Functions: Pass OnOffSwitchState values as input<br>.....                   | 4-27        |
| scatterhistogram Function: Display smooth marginal histograms using<br>kernel density estimates ..... | 4-28        |
| Graphics in MATLAB Online: Export graphics with specific dimensions and<br>padding .....              | 4-28        |
| Functionality being removed or changed .....                                                          | 4-28        |
| <b>App Building .....</b>                                                                             | <b>4-31</b> |
| uicolorpicker Function: Add and configure color picker components in apps<br>.....                    | 4-31        |
| uislider Function: Specify step size for range slider .....                                           | 4-31        |
| uiradiobutton Function: Specify HTML markup and display equations in<br>radio button text .....       | 4-31        |
| uitable Function: Specify table colors using hexadecimal color codes or<br>color names .....          | 4-32        |
| uitable Function: Visually identify table cells that have drop-down list editor<br>.....              | 4-32        |
| UI Figure Key-Press Callbacks: Access data about pressed key for larger list<br>of keys .....         | 4-32        |
| App Designer: Build apps that interface with Simulink models .....                                    | 4-32        |
| App Designer: Convert between additional similar UI components .....                                  | 4-32        |
| App Designer: Configure grid layout using improved editor interactions<br>.....                       | 4-33        |
| App Designer in MATLAB Online: Access toolbar actions and Help Center<br>resources using search ..... | 4-33        |
| Axes Interactions: Customize behavior of interactions .....                                           | 4-34        |
| Axes Interactions: Interact with axes using the keyboard .....                                        | 4-34        |

|                                                                                                                                  |             |
|----------------------------------------------------------------------------------------------------------------------------------|-------------|
| App Testing Framework: Perform scroll gestures on axes and UI axes . . .                                                         | 4-34        |
| App Testing Framework: Perform gestures on hyperlinks . . . . .                                                                  | 4-35        |
| App Testing Framework: Node expanded callback executes when gestures<br>programmatically expand tree nodes . . . . .             | 4-35        |
| Functionality being removed or changed . . . . .                                                                                 | 4-36        |
| <b>Performance . . . . .</b>                                                                                                     | <b>4-37</b> |
| datetime Format Parsing: Improved performance when parsing common<br>numeric formats . . . . .                                   | 4-37        |
| imread and imwrite Functions: Improved performance with JPEG files . .                                                           | 4-37        |
| smoothdata2 and fillmissing2 Functions: Improved performance for moving<br>average methods . . . . .                             | 4-38        |
| mldivide Function: Improved performance with tridiagonal matrices . . .                                                          | 4-39        |
| audioplayer Function : Improved performance with longer duration audio<br>signals . . . . .                                      | 4-39        |
| Files Panel in MATLAB Online: Improved performance when previewing<br>large MAT-files . . . . .                                  | 4-39        |
| Workspace Panel in MATLAB Online: Improved performance when loading<br>large number of variables . . . . .                       | 4-40        |
| Plots in Apps and MATLAB Online: Ticks and tick labels update faster when<br>zooming . . . . .                                   | 4-40        |
| Plots in Apps and MATLAB Online: Improved responsiveness when<br>interacting with partially transparent 3-D plots . . . . .      | 4-41        |
| <b>Software Development Tools . . . . .</b>                                                                                      | <b>4-43</b> |
| Project API: Find project files by label . . . . .                                                                               | 4-43        |
| Project API: Reanalyze all project dependencies . . . . .                                                                        | 4-43        |
| Project API: Export list of project files to archive . . . . .                                                                   | 4-43        |
| Project Upgrade: Check for compatibility issues and upgrade project with<br>improved usability and appearance . . . . .          | 4-43        |
| Source Control: Support for signing Git commits . . . . .                                                                        | 4-44        |
| Source Control: Support for Git hooks . . . . .                                                                                  | 4-44        |
| Source Control API: Discard changes in Git repositories programmatically<br>. . . . .                                            | 4-44        |
| Source Control in MATLAB Online: Expanded support for Git workflows<br>. . . . .                                                 | 4-44        |
| Build Automation: Perform additional test and coverage analyses using<br>TestTask class . . . . .                                | 4-45        |
| Build Automation: Obfuscate source code using PcodeTask class . . . . .                                                          | 4-45        |
| Build Automation: Build MEX file using MexTask class . . . . .                                                                   | 4-45        |
| Build Automation: Programmatically access code analysis results using<br>CodeIssuesTask class . . . . .                          | 4-45        |
| Build Automation: Run tests in parallel . . . . .                                                                                | 4-45        |
| Build Automation: Create initial build file . . . . .                                                                            | 4-45        |
| Build Automation: Disable incremental builds for task . . . . .                                                                  | 4-46        |
| Unit Testing Framework: Use renamed methods and name-value arguments<br>when working with namespaces . . . . .                   | 4-46        |
| Unit Testing Framework: Run tests using Test Browser in MATLAB Online<br>. . . . .                                               | 4-46        |
| Unit Testing Framework: Generate single-file HTML test report . . . . .                                                          | 4-46        |
| Unit Testing Framework: Control highlighting for covered or missed<br>executables in interactive code coverage reports . . . . . | 4-47        |
| Unit Testing Framework: Modify HTML document title of code coverage<br>report . . . . .                                          | 4-47        |

|                                                                                                                                                       |             |
|-------------------------------------------------------------------------------------------------------------------------------------------------------|-------------|
| Unit Testing Framework: Enable parallelizable plugins to work with default data if communication buffer is empty . . . . .                            | 4-47        |
| Unit Testing Framework: Use shared test fixtures in interactive tests . . .                                                                           | 4-47        |
| Unit Testing Framework: Respect element count when comparing sets . .                                                                                 | 4-48        |
| App Testing Framework: Perform scroll gestures on axes and UI axes . . .                                                                              | 4-48        |
| App Testing Framework: Perform gestures on hyperlinks . . . . .                                                                                       | 4-48        |
| App Testing Framework: Node expanded callback executes when gestures programmatically expand tree nodes . . . . .                                     | 4-48        |
| Functionality being removed or changed . . . . .                                                                                                      | 4-49        |
| <b>External Language Interfaces . . . . .</b>                                                                                                         | <b>4-50</b> |
| REST Function Service: Call user-defined MATLAB functions from any local or remote client program using REST . . . . .                                | 4-50        |
| Python Interface: Convert between MATLAB tables and Python Pandas DataFrames . . . . .                                                                | 4-50        |
| Python Interface: Interactively run Python code with Run Python Live Editor Task . . . . .                                                            | 4-50        |
| Python Interface: Convert between MATLAB and Python dictionaries . . .                                                                                | 4-50        |
| Publish C++ Interface: Share library definition file with publisher . . . .                                                                           | 4-51        |
| Publish C++ Interface: Use InterfaceName name-value argument, renamed from PackageName, to identify MATLAB interface to C++ library . . . .           | 4-51        |
| Call C++ from MATLAB: Use string for C++ enum parameter . . . . .                                                                                     | 4-51        |
| Call MATLAB from C++: Support for data types in matlab::engine::MATLABEngine functions feval and fevalAsync . . . . .                                 | 4-52        |
| Call MATLAB from C++: Support for handle classes in strongly typed C++ interface . . . . .                                                            | 4-52        |
| Call MATLAB from C++: Support for comparison operators . . . . .                                                                                      | 4-52        |
| Call MATLAB from C++: Use Namespaces name-value argument, renamed from Packages, with matlab.engine.typedinterface.generateCPP function . . . . .     | 4-52        |
| Java: MATLAB support for OpenJDK™ 17 . . . . .                                                                                                        | 4-52        |
| .NET Interface: Support for .NET 6 . . . . .                                                                                                          | 4-52        |
| Call MATLAB from .NET: Support for dictionary objects . . . . .                                                                                       | 4-53        |
| Call MATLAB from .NET: Support for handle classes in strongly typed C# interface . . . . .                                                            | 4-53        |
| Call MATLAB from .NET: Support for comparison operators . . . . .                                                                                     | 4-53        |
| Call MATLAB from .NET: Support for IConvertible interface . . . . .                                                                                   | 4-53        |
| Call MATLAB from .NET: Use Namespaces name-value argument, renamed from Packages, with matlab.engine.typedinterface.generateCSharp function . . . . . | 4-54        |
| Compilers: Support for MinGW64 Compiler (Fortran) . . . . .                                                                                           | 4-54        |
| Functionality being removed or changed . . . . .                                                                                                      | 4-54        |
| <b>Hardware Support . . . . .</b>                                                                                                                     | <b>4-56</b> |
| Hardware Boards, Cameras, and Drones: Documentation for hardware support packages moved to MATLAB documentation . . . . .                             | 4-56        |
| Arduino Hardware: Support on macOS with Apple silicon . . . . .                                                                                       | 4-56        |
| Arduino Hardware: Read data from ADIS16505 sensor . . . . .                                                                                           | 4-56        |
| Arduino Compatible Boards: ESP32 core version update . . . . .                                                                                        | 4-56        |
| Raspberry Pi Hardware: Support for Debian Bullseye with kernel 6.1.21 and above . . . . .                                                             | 4-57        |
| Raspberry Pi Hardware: Support on macOS with Apple silicon . . . . .                                                                                  | 4-57        |
| USB Webcams: Retrieve property metadata for webcam object using propertyInfo function . . . . .                                                       | 4-57        |

|                                                                         |      |
|-------------------------------------------------------------------------|------|
| USB Webcams: Use webcam objects as dictionary keys and values . . . . . | 4-57 |
| Functionality being removed or changed . . . . .                        | 4-57 |

## R2023b

|                                                                                                                                               |             |
|-----------------------------------------------------------------------------------------------------------------------------------------------|-------------|
| <b>Environment . . . . .</b>                                                                                                                  | <b>5-2</b>  |
| Live Editor Tasks: Create Live Editor task from selected code with interactive controls . . . . .                                             | 5-2         |
| Live Editor Tables: Add tables containing text and images to live scripts and functions . . . . .                                             | 5-2         |
| Live Editor Controls: Add color pickers and state buttons to live scripts . .                                                                 | 5-2         |
| Live Editor Controls: Select folders interactively in live scripts using file browser . . . . .                                               | 5-3         |
| Live Editor Accessibility: Interact with inline output using the keyboard . . . . .                                                           | 5-3         |
| Live Editor Export: Interactively customize export options, such as document paper size, orientation, margins, and figure format . . . . .    | 5-3         |
| export Function: Convert live scripts and functions to Markdown files and Jupyter notebooks . . . . .                                         | 5-4         |
| Editor Execution: Run sections even if other sections contain syntax errors . . . . .                                                         | 5-4         |
| Editor Variable Renaming: Automatically rename next references to variables or functions in file . . . . .                                    | 5-4         |
| Find and Replace Dialog Box: Use capture groups in regular expressions to search for and replace groups of characters . . . . .               | 5-5         |
| Debugging in MATLAB Online: Manage breakpoints and navigate the function call stack using Debugger panel . . . . .                            | 5-5         |
| Desktop Layout in MATLAB Online: Open additional panels directly from sidebars . . . . .                                                      | 5-6         |
| Themes in MATLAB Online: Save color customizations for individual themes . . . . .                                                            | 5-7         |
| Window Management in MATLAB Online: Automatically collapse windows . . . . .                                                                  | 5-7         |
| MATLAB on Macs: Better performance and improved battery life on MacBooks with MATLAB and Simulink running natively on Apple silicon . . . . . | 5-8         |
| MATLAB on Macs: Java Runtime Environment required for MATLAB and MATLAB Runtime on Apple silicon . . . . .                                    | 5-8         |
| MATLAB Drive: New URL for MATLAB Drive Online (January 2024) . . . . .                                                                        | 5-8         |
| Functionality being removed or changed . . . . .                                                                                              | 5-8         |
| <b>Language and Programming . . . . .</b>                                                                                                     | <b>5-10</b> |
| configureDictionary Function: Create dictionaries with configured keys and values . . . . .                                                   | 5-10        |
| lookup, insert, and remove Functions: Find, add, and remove dictionary entries . . . . .                                                      | 5-10        |
| codeIssues Object: Export issues identified in code . . . . .                                                                                 | 5-10        |
| Code Compatibility Analyzer App: Group and filter code issues with improved interface . . . . .                                               | 5-10        |
| Class Diagram Viewer: Add known subclasses to diagrams . . . . .                                                                              | 5-11        |

|                                                                                                                                                             |             |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------|
| mustBeSparse Function: Validate sparsity of arguments . . . . .                                                                                             | 5-11        |
| MATLAB backgroundPool Function: load function supported in background threads . . . . .                                                                     | 5-12        |
| Functionality being removed or changed . . . . .                                                                                                            | 5-12        |
| <b>Data Analysis . . . . .</b>                                                                                                                              | <b>5-14</b> |
| Experiment Manager App: Design experiments to run MATLAB code, and visualize, filter, and compare results . . . . .                                         | 5-14        |
| Pivot Table Live Editor Task: Interactively summarize tabular data in pivoted table . . . . .                                                               | 5-14        |
| resize, paddata, and trimdata Functions: Change the size of data by adding or removing elements . . . . .                                                   | 5-14        |
| smoothdata2 Function: Smooth noisy data in two dimensions . . . . .                                                                                         | 5-15        |
| kde Function: Estimate a probability function by using kernel density estimation for univariate data . . . . .                                              | 5-15        |
| Join Tables Live Editor Task: Sort output timetable by row times when row times are not key values . . . . .                                                | 5-15        |
| Data Cleaning: Interactively fill missing data with values from nearest neighbor rows . . . . .                                                             | 5-15        |
| Live Editor: Display the contents of nested tables as variables . . . . .                                                                                   | 5-15        |
| Live Editor: Copy output data as tab-delimited text . . . . .                                                                                               | 5-16        |
| pivot Function: Display empty groups in pivoted table . . . . .                                                                                             | 5-16        |
| pivot Function: Specify row names for pivoted table as row group names . . . . .                                                                            | 5-16        |
| stackedplot Function: Plot events associated with timetables . . . . .                                                                                      | 5-16        |
| Timetable Events: Perform calculations directly on event tables without extracting their data . . . . .                                                     | 5-17        |
| containsrange, overlapsrange, and withinrange Functions: Specify time ranges by using event filters . . . . .                                               | 5-17        |
| Variables Editor in MATLAB Online: Interactively change display format for datetime or duration data . . . . .                                              | 5-17        |
| Variables Editor in MATLAB Online: View and interactively edit scalar structures, object arrays, and tables with grouped variables . . . . .                | 5-18        |
| Accessibility in MATLAB Online: Use a screen reader to interact with the Workspace browser, Variables editor, Property Inspector, and Import Tool . . . . . | 5-19        |
| Histogram Distribution Charts: Calculate values and create histograms using percentages . . . . .                                                           | 5-19        |
| tallrng Function: Specify random number algorithm without specifying seed . . . . .                                                                         | 5-19        |
| Functionality being removed or changed . . . . .                                                                                                            | 5-19        |
| <b>Data Import and Export . . . . .</b>                                                                                                                     | <b>5-20</b> |
| readstruct Function: Read data from JSON files . . . . .                                                                                                    | 5-20        |
| writestruct Function: Write structure arrays to JSON files . . . . .                                                                                        | 5-20        |
| Parquet: Create ParquetDatastore objects more efficiently with partition control in parallel environments . . . . .                                         | 5-20        |
| cmdsep Function: Return command separator for current platform . . . . .                                                                                    | 5-20        |
| NetCDF Interface: Import netCDF data using Import Tool in MATLAB Online and Import Data Live Editor Task . . . . .                                          | 5-20        |
| NetCDF Interface in MATLAB Online: Import netCDF data by double-clicking netCDF files . . . . .                                                             | 5-20        |
| Image File Format Libraries: LibTIFF library upgraded to version 4.5.0 . . . . .                                                                            | 5-20        |

|                                                                                                                                                    |             |
|----------------------------------------------------------------------------------------------------------------------------------------------------|-------------|
| Scientific File Format Libraries: HDF4 library upgraded to version 4.2.16                                                                          | 5-21        |
| Scientific File Format Libraries: HDF5 library upgraded to version 1.10.10                                                                         | 5-21        |
| Scientific File Format Libraries: NetCDF library upgraded to version 4.9.1                                                                         | 5-21        |
| serialbreak Function: Send break to device connected to serial port . . .                                                                          | 5-21        |
| Functionality being removed or changed . . . . .                                                                                                   | 5-21        |
| <b>Mathematics</b> . . . . .                                                                                                                       | <b>5-23</b> |
| Ordinary Differential Equations: Use object-oriented solution framework for ODEs . . . . .                                                         | 5-23        |
| rng Function: Change default algorithm and seed for random number generator, and specify random number algorithm without specifying seed . . . . . | 5-23        |
| MATLAB Support Package for Quantum Computing: Solve Quadratic Unconstrained Binary Optimization (QUBO) problems . . . . .                          | 5-24        |
| MATLAB Support Package for Quantum Computing: Control circuit plot appearance and behavior with QuantumCircuitChart objects . . . . .              | 5-24        |
| MATLAB Support Package for Quantum Computing: Connect to quantum hardware through IBM (October 2023, Version 23.2.1) . . . . .                     | 5-24        |
| deconv Function: Perform least-squares deconvolution with different convolved subsections . . . . .                                                | 5-25        |
| svdappend Function: Calculate revised singular value decompositions . .                                                                            | 5-25        |
| expmv Function: Calculate matrix exponential multiplied by vector . . . .                                                                          | 5-25        |
| expm Function: Improved algorithm for single-precision matrices . . . . .                                                                          | 5-25        |
| scatteredInterpolant Object: Use multivalued interpolation to interpolate multiple data sets simultaneously . . . . .                              | 5-25        |
| <b>Graphics</b> . . . . .                                                                                                                          | <b>5-27</b> |
| piechart and donutchart Functions: Create configurable, interactive pie and donut charts . . . . .                                                 | 5-27        |
| Plot Colors: Choose from nine color palettes . . . . .                                                                                             | 5-27        |
| abyss Function: Apply blue-to-black colormap to charts and images . . . .                                                                          | 5-28        |
| Legend: Control the order of legend entries . . . . .                                                                                              | 5-28        |
| xscale, yscale, and zscale Functions: Quickly change between logarithmic and linear axes scales after plotting . . . . .                           | 5-29        |
| xsecondarylabel, ysecondarylabel, and zsecondarylabel Functions: Create, delete, or modify secondary axis labels . . . . .                         | 5-29        |
| Bar Charts: Specify bar labels as string vectors . . . . .                                                                                         | 5-30        |
| xregion and yregion Functions: Create unbounded regions and specify multiple regions as matrix . . . . .                                           | 5-30        |
| Text Objects: Control whether text labels are included in axes limits . . . .                                                                      | 5-31        |
| Contour Plots: Customize colors of contour labels . . . . .                                                                                        | 5-32        |
| Colors and Line Styles: Control automatic color and line style selection for additional chart objects . . . . .                                    | 5-32        |
| Heatmap Charts: Control text interpreter for text labels in heatmap charts . . . . .                                                               | 5-33        |
| Images: Display images with datetime, duration, or categorical pixel coordinates . . . . .                                                         | 5-34        |
| swarmchart Function: Create horizontal swarm charts . . . . .                                                                                      | 5-34        |
| Geographic Plots: Some basemaps have improved appearance at high zoom levels . . . . .                                                             | 5-35        |
| Figure Code in MATLAB Online: Generate code from Camera tab . . . . .                                                                              | 5-36        |



|                                                                                                                        |             |
|------------------------------------------------------------------------------------------------------------------------|-------------|
| Figure Code in MATLAB Online: Generate code from Property Inspector .....                                              | 5-36        |
| Functionality being removed or changed .....                                                                           | 5-36        |
| <b>App Building .....</b>                                                                                              | <b>5-42</b> |
| uislider Function: Create slider to specify range of values in apps and on App Designer canvas .....                   | 5-42        |
| uicontextmenu and uimenu Functions: Access information about action used to open context menu .....                    | 5-42        |
| uispinner and uieditfield Functions: Create spinners and numeric edit fields with placeholder text .....               | 5-42        |
| uilibox and uidropdown Functions: Access index of component value in list of items .....                               | 5-43        |
| uitable Function: Query selected table data in current table display ....                                              | 5-43        |
| uicontextmenu Function: Context menus on trees appear when you right-click anywhere in the tree .....                  | 5-43        |
| appmigration.migrateGUIDEApp Function: Migrating multiple apps from GUIDE to App Designer updates app references ..... | 5-44        |
| App Designer: Interactively rearrange tabs, menus, tree nodes, and toolbar tools .....                                 | 5-44        |
| App Designer: Write unit tests for custom UI components .....                                                          | 5-45        |
| App Designer: View changes to HTML UI component source while working in Design View .....                              | 5-46        |
| Comparison Tool: Save app comparison results as PDF or DOCX reports programmatically and interactively .....           | 5-46        |
| Axes Interactions: Interpolate data tip location .....                                                                 | 5-46        |
| Axes Interactions: Indicate azimuth and elevation angles .....                                                         | 5-46        |
| Axes Interactions: Zoom into rectangular region of 2-D view of 3-D chart .....                                         | 5-46        |
| App Testing Framework: Automatically scroll to components outside viewable area .....                                  | 5-47        |
| Web Apps and Standalone Applications: Data brushing supported in graphics .....                                        | 5-47        |
| <b>Performance .....</b>                                                                                               | <b>5-48</b> |
| Language and Programming: Improved performance for reading and writing class property values .....                     | 5-48        |
| dde23 Function: Improved performance integrating high-dimensional systems .....                                        | 5-48        |
| nufftn Function: Improved performance with nonuniform sample points or query points .....                              | 5-49        |
| fillmissing Function: Improved performance when filling entries with previous or next numeric value .....              | 5-50        |
| ismember Function: Improved performance with small unsorted set array .....                                            | 5-50        |
| islocalmax and islocalmin Functions: Improved performance when finding local minima and maxima .....                   | 5-51        |
| Data Cleaner App: Improved rendering performance of Cleaning Parameters panel .....                                    | 5-51        |
| Tiled Chart Layouts: Improved performance for flow layouts with spanned axes .....                                     | 5-52        |
| Plots in Apps: Improved performance when scrolling to zoom .....                                                       | 5-53        |
| Plots in Apps: Improved responsiveness for scatter plot interactions in apps .....                                     | 5-53        |

|                                                                                                                 |             |
|-----------------------------------------------------------------------------------------------------------------|-------------|
| Plots in Apps: Constant lines and constant regions update continuously as you pan .....                         | 5-54        |
| App Building: Improved performance when resizing apps with nested containers .....                              | 5-55        |
| App Building: Improved startup performance for apps with panels in multiple tabs .....                          | 5-56        |
| App Designer: Improved app loading and running performance .....                                                | 5-57        |
| App Designer: Improved performance when adding custom UI components to an app .....                             | 5-58        |
| Property Inspector in MATLAB Online: Improved performance when opening for the first time .....                 | 5-58        |
| Graphics in MATLAB Online: Images and 3-D plots in apps have significant performance gains .....                | 5-58        |
| <b>Software Development Tools .....</b>                                                                         | <b>5-60</b> |
| Source Control API: Interact with Git source control programmatically ..                                        | 5-60        |
| Project API: Determine whether file belongs to a project .....                                                  | 5-60        |
| Source Control in MATLAB Online: Perform source control operations using unified panel .....                    | 5-60        |
| Source Control in MATLAB Online: Expanded support for Git workflows .....                                       | 5-62        |
| Projects in MATLAB Online: Added support for team collaboration workflows .....                                 | 5-62        |
| Project Examples: Identify and run tests in projects .....                                                      | 5-63        |
| Build Automation: Use built-in tasks to define common build actions ....                                        | 5-63        |
| Build Automation: Automatically open MATLAB project when running builds .....                                   | 5-63        |
| Build Automation: Invoke build tool from project subfolders .....                                               | 5-63        |
| Build Automation: Operate on file collections .....                                                             | 5-63        |
| Build Automation: Name the inputs and outputs of tasks for incremental builds .....                             | 5-63        |
| Build Automation: Run tasks using build options .....                                                           | 5-64        |
| Unit Testing Framework: Programmatically access code coverage results using the runtests function .....         | 5-64        |
| Unit Testing Framework: Test for missing elements .....                                                         | 5-64        |
| Unit Testing Framework: Modify test report title .....                                                          | 5-64        |
| App Testing Framework: Automatically scroll to components outside viewable area .....                           | 5-64        |
| matlab.test.behavior.Missing Class: Test against constructor arguments that result in incompatible values ..... | 5-65        |
| Functionality being removed or changed .....                                                                    | 5-65        |
| <b>External Language Interfaces .....</b>                                                                       | <b>5-66</b> |
| Call MATLAB from C++: Generate output types for strongly typed methods and functions .....                      | 5-66        |
| Publish C++ Interface: Support for pointer return of string and fundamental types .....                         | 5-66        |
| Call MATLAB from .NET: Generate C# code interface for MATLAB packages, classes, and functions .....             | 5-66        |
| Call .NET from MATLAB: Create explicit interface view of .NET object ..                                         | 5-66        |
| Python: Support for version 3.11 .....                                                                          | 5-66        |
| Python Interface: Use of environment variable PYTHONHOME to load interpreter .....                              | 5-66        |

|                                                                                                         |             |
|---------------------------------------------------------------------------------------------------------|-------------|
| Python Interface: Convert list and tuple of Python datetime or timedelta objects to MATLAB arrays ..... | 5-67        |
| Python Interface: Resolve Python configuration issues .....                                             | 5-67        |
| Compilers: Download MinGW-w64 8.1 from Add-On Explorer .....                                            | 5-67        |
| Compilers: Support for NAG Fortran on Apple silicon .....                                               | 5-67        |
| Compilers: Support for Microsoft Visual Studio Build Tools .....                                        | 5-67        |
| Functionality being removed or changed .....                                                            | 5-68        |
| <b>Hardware Support .....</b>                                                                           | <b>5-69</b> |
| Support for ArduCam Multi Camera Adapter Module on Raspberry Pi Hardware .....                          | 5-69        |

## R2023a

|                                                                                                               |            |
|---------------------------------------------------------------------------------------------------------------|------------|
| <b>Environment .....</b>                                                                                      | <b>6-2</b> |
| MATLAB Install: Reduce footprint using default installation without local documentation .....                 | 6-2        |
| Editor: Interactively increment numeric values within section and run section after every change .....        | 6-2        |
| Live Editor Controls: Add file browser to select file interactively in live script .....                      | 6-2        |
| Live Editor Controls: Align controls within a section when code is hidden .....                               | 6-3        |
| Desktop Layout in MATLAB Online: Access desktop tools and change the desktop layout using sidebars .....      | 6-3        |
| Code Issues Tool in MATLAB Online: Check code for errors and warnings using Code Issues tool .....            | 6-4        |
| Find Files Tool in MATLAB Online: Search for files with improved Find Files tool .....                        | 6-5        |
| Editor in MATLAB Online: Edit read-only files .....                                                           | 6-6        |
| Search in MATLAB Online: Access videos using Search box .....                                                 | 6-6        |
| System Command Functions: Set and get multiple variables using operating system commands .....                | 6-6        |
| loadenv Function: Load environment variables from .env and plain text files .....                             | 6-6        |
| Comparison Tool: Save text comparison results as PDF or DOCX reports programmatically and interactively ..... | 6-6        |
| Toolbox Packaging: Specify packaging options when creating custom toolbox programmatically .....              | 6-7        |
| Functionality being removed or changed .....                                                                  | 6-7        |
| <b>Language and Programming .....</b>                                                                         | <b>6-8</b> |
| Code Analyzer: Enable custom checks and configure existing checks ....                                        | 6-8        |
| fix Function: Fix code issues from the command line .....                                                     | 6-8        |
| Code Analyzer App: Apply fixes to code issues interactively .....                                             | 6-9        |
| dictionary Object: Access and assign dictionary cell values with curly braces .....                           | 6-9        |
| Output Argument Validation: Debug within output argument blocks ....                                          | 6-10       |
| Functionality being removed or changed .....                                                                  | 6-10       |

|                                                                                                                               |             |
|-------------------------------------------------------------------------------------------------------------------------------|-------------|
| <b>Data Analysis</b> .....                                                                                                    | <b>6-12</b> |
| pivot Function: Summarize tabular data using pivot table .....                                                                | <b>6-12</b> |
| table and timetable Data Types: Perform calculations directly on tables and<br>timetables without extracting their data ..... | <b>6-12</b> |
| Timetable Events: Find and label events in timetables using attached event<br>tables .....                                    | <b>6-12</b> |
| combinations Function: Generate all element combinations of arrays ...                                                        | <b>6-12</b> |
| fillmissing2 Function: Fill missing entries in two-dimensional data .....                                                     | <b>6-13</b> |
| fillmissing Function: Use values from nearest neighbors to fill missing data<br>.....                                         | <b>6-13</b> |
| Descriptive Statistics and Arithmetic: Omit or include missing data of<br>multiple data types .....                           | <b>6-13</b> |
| Find and Remove Trends Live Editor Task: Interactively find and remove<br>periodic and polynomial trends .....                | <b>6-13</b> |
| Data Cleaner App: Save session as MAT-file and reload session .....                                                           | <b>6-14</b> |
| head and tail Functions: Get top or bottom rows of array .....                                                                | <b>6-14</b> |
| groupsummary Function: Compute number of unique elements .....                                                                | <b>6-14</b> |
| movevars Function: Move table variables after last variable without After<br>name-value argument .....                        | <b>6-14</b> |
| rowfun and varfun Functions: OutputFormat name-value argument can take<br>"auto" as value .....                               | <b>6-15</b> |
| day Function: Return ISO day of week .....                                                                                    | <b>6-15</b> |
| week Function: Return ISO week of year or week of month .....                                                                 | <b>6-15</b> |
| Variables Editor: Edit categories of categorical table variable in MATLAB<br>Online .....                                     | <b>6-16</b> |
| <b>Data Import and Export</b> .....                                                                                           | <b>6-17</b> |
| Import Data Live Editor Task: Import data in live scripts .....                                                               | <b>6-17</b> |
| audiowrite Function: Write MP3 audio files .....                                                                              | <b>6-17</b> |
| imfinfo Function: Get information about XMP metadata embedded in JPEG<br>files .....                                          | <b>6-17</b> |
| Parallel Processing: Use readtable in thread-based environments .....                                                         | <b>6-17</b> |
| Scientific File Format Libraries: CDF library upgraded to version 3.8.1 ..                                                    | <b>6-17</b> |
| Scientific File Format Libraries: CFITSIO library upgraded to version 4.1.0<br>.....                                          | <b>6-17</b> |
| Functionality being removed or changed .....                                                                                  | <b>6-18</b> |
| <b>Mathematics</b> .....                                                                                                      | <b>6-19</b> |
| MATLAB Support Package for Quantum Computing: Build, simulate, and<br>run quantum algorithms .....                            | <b>6-19</b> |
| pageeig Function: Perform eigenvalue decomposition on pages of N-D<br>arrays .....                                            | <b>6-19</b> |
| randi Function: Create random logical array .....                                                                             | <b>6-19</b> |
| Functionality being removed or changed .....                                                                                  | <b>6-20</b> |
| <b>Graphics</b> .....                                                                                                         | <b>6-21</b> |
| xregion and yregion Functions: Highlight horizontal or vertical regions of<br>plots .....                                     | <b>6-21</b> |
| sky Function: Apply monochromatic colormap to charts .....                                                                    | <b>6-21</b> |
| tiledlayout Function: Create horizontal or vertical layouts .....                                                             | <b>6-22</b> |
| animatedline Function: Create animated lines using numeric, datetime, or<br>duration data .....                               | <b>6-22</b> |

|                                                                                                                        |             |
|------------------------------------------------------------------------------------------------------------------------|-------------|
| Grid Lines: Customize grid line thickness . . . . .                                                                    | 6-22        |
| Axes Labels: Rotate x- and y-axes labels without overlapping the axes . .                                              | 6-23        |
| Plotting Series of Lines: Control cycling order of line styles . . . . .                                               | 6-24        |
| Plotting Series of Lines: Control whether the data range of a line affects<br>automatic axes limits . . . . .          | 6-24        |
| fontsize and fontname Functions: Optionally specify the object containing<br>the text . . . . .                        | 6-25        |
| DatetimeRuler Object: Set or get the reference date for plotting datetime<br>values . . . . .                          | 6-25        |
| Image Display Preferences: Access and update imshow preferences in<br>MATLAB Online . . . . .                          | 6-25        |
| Functionality being removed or changed . . . . .                                                                       | 6-26        |
| <b>App Building . . . . .</b>                                                                                          | <b>6-27</b> |
| addStyle Function: Add styles to items in list box or drop-down UI<br>component . . . . .                              | 6-27        |
| uistack Function: Change stacking order of UI components in UI figure .                                                | 6-27        |
| uipanel and uibuttongroup Functions: Specify container border color . . .                                              | 6-27        |
| uihtml Function: Send events between MATLAB and HTML . . . . .                                                         | 6-27        |
| uiimage Function: Specify image alt text for use with screen readers . . .                                             | 6-28        |
| appmigration.migrateGUIDEApp Function: Programmatically migrate<br>existing GUIDE apps to App Designer . . . . .       | 6-28        |
| App Designer: View progress when loading an app . . . . .                                                              | 6-28        |
| App Designer: Add label to unlabeled UI component . . . . .                                                            | 6-28        |
| App Designer: Replace assigned callback with new callback . . . . .                                                    | 6-28        |
| App Testing Framework: Test context menus within labels . . . . .                                                      | 6-29        |
| Axes Interactions: Customize behavior of interactions with axes view . . .                                             | 6-29        |
| Plots in Apps: Enable data cursor mode . . . . .                                                                       | 6-29        |
| Plots in Apps: Specify axes for interaction mode . . . . .                                                             | 6-30        |
| Functionality being removed or changed . . . . .                                                                       | 6-30        |
| <b>Performance . . . . .</b>                                                                                           | <b>6-32</b> |
| Language and Programming: Improved performance when calling functions<br>and methods . . . . .                         | 6-32        |
| Function Handles: Improved performance when invoking handles to named<br>functions . . . . .                           | 6-32        |
| varargin Argument: Improved performance when specifying zero or more<br>inputs . . . . .                               | 6-33        |
| timetable Data Type Indexing: Improved performance when subscripting<br>with times or with withtol subscript . . . . . | 6-34        |
| Complex Matrices: Improved performance when using colon indexing to<br>copy complex matrices . . . . .                 | 6-35        |
| mean, std, var, and rmse Functions: Improved performance when computing<br>along default vector dimension . . . . .    | 6-37        |
| Moving Statistics Functions: Improved performance when computing over<br>matrix with sample points . . . . .           | 6-37        |
| histcounts Function: Improved performance with small numeric and logical<br>input data . . . . .                       | 6-38        |
| fzero function: Improved performance . . . . .                                                                         | 6-38        |
| Plots in Apps: Improved performance when rerendering axes . . . . .                                                    | 6-39        |
| Plots in Apps: Improved performance when creating axes . . . . .                                                       | 6-39        |
| Plots in Apps: Improved responsiveness of ruler-pan interaction . . . . .                                              | 6-40        |
| Live Editor: Improved performance when filtering numeric table variables<br>. . . . .                                  | 6-41        |

|                                                                                                          |             |
|----------------------------------------------------------------------------------------------------------|-------------|
| Property Inspector: Improved performance when opening for the first time                                 | 6-41        |
| Property Inspector: Improved performance when switching between objects                                  | 6-42        |
| Variables Editor: Improved performance of cell editing in MATLAB Online                                  | 6-42        |
| Variables Editor: Improved speed of data display when scrolling in MATLAB Online                         | 6-43        |
| App Building: Improved app startup performance                                                           | 6-43        |
| App Building: Improved startup performance for apps with multiple tabs                                   | 6-44        |
| App Building: Improved performance when resizing some apps                                               | 6-45        |
| <b>Software Development Tools</b>                                                                        | <b>6-47</b> |
| Build Automation: Improve build speed and efficiency with incremental builds                             | 6-47        |
| Build Automation: Create and run tasks that accept arguments                                             | 6-47        |
| Dependency Analyzer: Analyze files and folders with or without a project                                 | 6-47        |
| Project Preferences: Recreate empty project folders in Git repositories                                  | 6-47        |
| Project API: Determine if file is under project root folder                                              | 6-47        |
| Project API: Export subset of project files to archive                                                   | 6-47        |
| Project Sharing: Include only specific files in project archive using export profile                     | 6-48        |
| Comparison Tool: Automate comparison report generation for continuous integration (CI) workflows         | 6-48        |
| Source Control in MATLAB Online: Save uncommitted changes by creating a Git stash                        | 6-48        |
| Source Control in MATLAB Online: Manage Git remote repositories locally using Branch Manager             | 6-48        |
| Source Control in MATLAB Online: Detect and extract conflict markers from text and binary files          | 6-48        |
| Comparison Tool in MATLAB Online: Compare project definition files                                       | 6-48        |
| Dependency Analyzer in MATLAB Online: Investigate circular dependencies using the Project Hierarchy view | 6-49        |
| Unit Testing Framework: Run tests interactively by using Test Browser                                    | 6-49        |
| Unit Testing Framework: Programmatically access code coverage results                                    | 6-50        |
| Unit Testing Framework: Temporarily set environment variables                                            | 6-50        |
| Unit Testing Framework: Test for handle validity                                                         | 6-50        |
| Unit Testing Framework: Write text to files in thread-based environment                                  | 6-50        |
| Unit Testing Framework: Use renamed classes in testing and other automated workflows                     | 6-50        |
| App Testing Framework: Test context menus within labels                                                  | 6-51        |
| Performance Testing Framework: Use fewer samples to meet the objective margin of error                   | 6-51        |
| Functionality being removed or changed                                                                   | 6-52        |
| <b>External Language Interfaces</b>                                                                      | <b>6-53</b> |
| Publish C++ Interface: Publish interface for C++ library in Live Editor                                  | 6-53        |
| Interface to C++ Library: Execute C++ library functions out-of-process                                   | 6-53        |
| Interface to C++ Library: Support for default arguments                                                  | 6-53        |

|                                                                                                                                            |             |
|--------------------------------------------------------------------------------------------------------------------------------------------|-------------|
| Interface to C++ Library: Support for comments in function templates . .                                                                   | 6-53        |
| Publish C++ Interface: Put libraries on run-time path . . . . .                                                                            | 6-53        |
| Publish C++ Interface: Resolve multiple redefinition and unresolved<br>external symbol errors . . . . .                                    | 6-54        |
| Publish C++ Interface: Information to debug C++ library functions . . .                                                                    | 6-54        |
| Publish C++ Interface: Support for MATLAB operators for C++ methods<br>. . . . .                                                           | 6-54        |
| Java Interface: Support for Java 11 JDK and JRE . . . . .                                                                                  | 6-54        |
| Java Interface: jenv and matlab_jenv provide environment information . .                                                                   | 6-54        |
| Python Interface: Convert between MATLAB datetime and Python datetime,<br>NumPy datetime64 types . . . . .                                 | 6-54        |
| Python Interface: Convert between MATLAB duration and Python timedelta,<br>NumPy timedelta64 types . . . . .                               | 6-55        |
| Python Objects: Use Python objects as keys in dictionary . . . . .                                                                         | 6-55        |
| .NET Interface: Convert between MATLAB dictionary and .NET<br>System.Collections.Generic.Dictionary objects . . . . .                      | 6-55        |
| .NET Objects: Use .NET objects as keys in dictionary . . . . .                                                                             | 6-55        |
| .NET Engine: Support for MATLAB structs . . . . .                                                                                          | 6-55        |
| Perl 5.36.0: MATLAB support on Windows . . . . .                                                                                           | 6-55        |
| Compiler support changed for building C and C++ interfaces, MEX files,<br>and standalone MATLAB engine and MAT-file applications . . . . . | 6-56        |
| Functionality being removed or changed . . . . .                                                                                           | 6-56        |
| <b>Hardware Support . . . . .</b>                                                                                                          | <b>6-58</b> |
| Support for MJPEG format in USB Webcams on Windows . . . . .                                                                               | 6-58        |
| Support for 32-bit Debian Bullseye on Raspberry Pi . . . . .                                                                               | 6-58        |
| Support for Raspberry Pi Zero 2 W and Raspberry Pi Compute Module 4<br>. . . . .                                                           | 6-58        |

## R2022b

|                                                                                                                     |            |
|---------------------------------------------------------------------------------------------------------------------|------------|
| <b>Environment . . . . .</b>                                                                                        | <b>7-2</b> |
| Live Editor Controls: Add numeric spinners to increment and decrement<br>variable values in live scripts . . . . .  | 7-2        |
| Live Editor Tasks: Specify what code to run when control value changes<br>. . . . .                                 | 7-2        |
| Suggestions and Completions: Change when to show suggestions and how<br>to accept them . . . . .                    | 7-2        |
| Search in MATLAB Online: Access toolstrip actions, preferences, and Help<br>Center resources using search . . . . . | 7-3        |
| System Theme in MATLAB Online: Change colors of MATLAB desktop to<br>match operating system color scheme . . . . .  | 7-5        |
| Help Center: View documentation for all products and determine which<br>products are installed . . . . .            | 7-5        |
| unsetenv Function: Remove environment variable . . . . .                                                            | 7-6        |
| isenv Function: Determine if environment variable exists . . . . .                                                  | 7-6        |
| Comparison Tool: Compare MAT files, FIG files, live scripts, and live<br>functions in MATLAB Online . . . . .       | 7-6        |
| Functionality being removed or changed . . . . .                                                                    | 7-6        |

|                                                                                                                                                 |             |
|-------------------------------------------------------------------------------------------------------------------------------------------------|-------------|
| <b>Language and Programming</b> .....                                                                                                           | <b>7-8</b>  |
| dictionary Object: Map unique keys to values for fast lookup .....                                                                              | <b>7-8</b>  |
| Output Argument Validation: Validate output arguments in functions and<br>class methods .....                                                   | <b>7-8</b>  |
| Code Analyzer App: Identify code issues .....                                                                                                   | <b>7-9</b>  |
| codeIssues Object: Identify and store code issues from one or more files<br>.....                                                               | <b>7-9</b>  |
| Custom Compact Display: Live Editor, Variables editor, and Workspace<br>browser follow CustomCompactDisplayProvider implementations .....       | <b>7-9</b>  |
| Customized Dot Indexing: New methods handle parentheses-dot<br>combinations .....                                                               | <b>7-9</b>  |
| Functionality being removed or changed .....                                                                                                    | <b>7-9</b>  |
| <b>Data Analysis</b> .....                                                                                                                      | <b>7-11</b> |
| Data Cleaner App: Clean data in table .....                                                                                                     | <b>7-11</b> |
| Data Cleaner App: View sparklines and summary statistics .....                                                                                  | <b>7-11</b> |
| stackedplot Function: Plot data from multiple tables or timetables .....                                                                        | <b>7-11</b> |
| detrend Function: Detrend tabular data .....                                                                                                    | <b>7-12</b> |
| rmoutliers Function: Define outlier locations, and optionally return outlier<br>indicator, thresholds, and center value .....                   | <b>7-12</b> |
| mape and rmse Functions: Calculate error between forecast and actual data<br>.....                                                              | <b>7-12</b> |
| isuniform Function: Determine if vector is uniformly spaced .....                                                                               | <b>7-12</b> |
| Serial date numbers and date strings are not recommended .....                                                                                  | <b>7-12</b> |
| datetime and convertTo Functions: Convert between CDF TT2000 times and<br>datetime arrays .....                                                 | <b>7-13</b> |
| Data Preprocessing Live Editor Tasks: Plot multiple table variables .....                                                                       | <b>7-13</b> |
| Data Preprocessing Live Editor Tasks: Append cleaned table variables and<br>specify logical tabular output .....                                | <b>7-14</b> |
| Clean Missing Data Live Editor Task: Plot nonnumeric table data, specify<br>minimum number of missing entries, and use custom fill method ..... | <b>7-14</b> |
| Clean Outlier Data Live Editor Task: Convert outliers to missing .....                                                                          | <b>7-14</b> |
| Live Editor and Variables Editor: Control placement of missing values when<br>sorting numeric data in MATLAB Online .....                       | <b>7-15</b> |
| Variables Editor: View sparklines and summary statistics for tabular data in<br>MATLAB Online .....                                             | <b>7-15</b> |
| Variables Editor: Navigate variable with Find and Go To dialog boxes in<br>MATLAB Online .....                                                  | <b>7-15</b> |
| Variables Editor: Interactively save logical indices of data selection in<br>MATLAB Online .....                                                | <b>7-16</b> |
| Variables Editor: Replace tabular data with empty arrays in MATLAB Online<br>.....                                                              | <b>7-16</b> |
| Functionality being removed or changed .....                                                                                                    | <b>7-16</b> |
| <b>Data Import and Export</b> .....                                                                                                             | <b>7-22</b> |
| Parquet: Use Parquet files containing nested structured data .....                                                                              | <b>7-22</b> |
| SequentialDatastore Object: Sequentially read data from multiple<br>datastores .....                                                            | <b>7-22</b> |
| Datastores: Create subsets of arbitrarily nested transformations and<br>combinations of datastores .....                                        | <b>7-22</b> |
| Parallel Processing: Use low-level file I/O operations in thread-based<br>environments .....                                                    | <b>7-22</b> |
| CDF Interface: Read CDF_INT8 and CDF_TIME_TT2000 data .....                                                                                     | <b>7-22</b> |



|                                                                                                                                                                     |             |
|---------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------|
| imwrite Function: Use datetime values to write metadata for PNG images                                                                                              | 7-23        |
| Image File Format Libraries: LibTIFF library upgraded to version 4.4.0                                                                                              | 7-23        |
| Scientific File Format Libraries: HDF4 and HDF-EOS2 libraries are upgraded                                                                                          | 7-23        |
| Functionality being removed or changed                                                                                                                              | 7-23        |
| <b>Mathematics</b>                                                                                                                                                  | <b>7-24</b> |
| pagenorm Function: Calculate norms using pages of N-D arrays                                                                                                        | 7-24        |
| Functionality being removed or changed                                                                                                                              | 7-24        |
| <b>Graphics</b>                                                                                                                                                     | <b>7-25</b> |
| Plotting Table Data: Create stem, staircase, and geographic plots by passing tables directly to plotting functions                                                  | 7-25        |
| errorbar Function: Plot multiple lines with error bars at once                                                                                                      | 7-25        |
| tightPosition Function: Get the location and size of axes plotting area                                                                                             | 7-26        |
| tilenum and tilerowcol Functions: Convert between rows and columns and tile numbers in tiled chart layouts                                                          | 7-26        |
| Tiled Chart Layout: Define GridSizeChangedFcn callback that executes when the grid size changes                                                                     | 7-26        |
| Contour Plots: Customize contour labels, colors, and transparency                                                                                                   | 7-26        |
| boxchart Function: Specify the box edge color and box median line color                                                                                             | 7-27        |
| 3-D Stem and Bar Plots: Expanded data type support                                                                                                                  | 7-28        |
| Image Objects: Control the maximum resolution for displaying images                                                                                                 | 7-28        |
| Figure Code: Generate code for figure formatting in MATLAB Online                                                                                                   | 7-28        |
| Plot Options: Customize figure creation, data linking, and labeling in MATLAB Online                                                                                | 7-29        |
| Functionality being removed or changed                                                                                                                              | 7-29        |
| <b>App Building</b>                                                                                                                                                 | <b>7-34</b> |
| ClickedFcn and DoubleClickedFcn Callbacks: Program a response to a user clicking or double-clicking a tree, check box tree, list box, table, or drop-down component | 7-34        |
| uieditfield Function: Specify valid length and input type for edit field text                                                                                       | 7-34        |
| uiimage Function: Create an image hyperlink                                                                                                                         | 7-34        |
| uipanel and uibuttongroup Functions: Specify container border width                                                                                                 | 7-34        |
| uigridlayout Function: Query grid layout manager size and location                                                                                                  | 7-35        |
| uibutton and uitogglebutton Functions: Specify additional icon and icon alignment options                                                                           | 7-35        |
| uitoolbar Function: Specify background color of toolbar                                                                                                             | 7-35        |
| scroll Function: Programmatically scroll to specified component in container                                                                                        | 7-35        |
| isInScrollView Function: Determine if any component in a container is visible, not only direct children                                                             | 7-35        |
| App Designer: Add and delete callbacks more efficiently                                                                                                             | 7-35        |
| App Designer: Move components between tabs                                                                                                                          | 7-36        |
| Graphics Support: Restore axes view and display context menu for interaction modes                                                                                  | 7-36        |
| Graphics Support: Use visual camera toolbar interface                                                                                                               | 7-36        |
| Comparison Tool: Toggle the ability to navigate only mergeable changes                                                                                              | 7-36        |

|                                                                                                                  |             |
|------------------------------------------------------------------------------------------------------------------|-------------|
| Functionality being removed or changed .....                                                                     | 7-36        |
| <b>Performance .....</b>                                                                                         | <b>7-40</b> |
| prctile, quantile, and iqr Functions: Improved performance with small input data .....                           | 7-40        |
| mldivide and pagemldivide Functions: Improved performance with small matrices .....                              | 7-40        |
| conv, conv2, and convn Functions: Improved performance when convolving two vectors, matrices, and arrays .....   | 7-41        |
| Tall Arrays: Improved performance when indexing tall arrays backed by parquetDatastore .....                     | 7-42        |
| App Building: Improved app startup performance .....                                                             | 7-43        |
| App Building: Improved startup performance for apps with multiple tabs .....                                     | 7-43        |
| Plots in Apps: Improved responsiveness when interacting with large images .....                                  | 7-44        |
| Plots in Apps: Ticks and grid lines update as you pan .....                                                      | 7-45        |
| Plots in Apps: Improved performance and smaller PDF files when exporting complex plots .....                     | 7-46        |
| Tiled Chart Layouts: Improved performance for flow layouts with spanned axes .....                               | 7-47        |
| ArrayDatastore Object: Improved performance when reading data using readall .....                                | 7-48        |
| ArrayDatastore Object: Improved performance when specifying number of rows to read .....                         | 7-49        |
| parquetread Function: Improved performance reading string data from Parquet files .....                          | 7-49        |
| parquetDatastore Function: Improved performance when creating datastore with large number of Parquet files ..... | 7-50        |
| Workspace Browser: Improved performance of variable-deletion dialog boxes in MATLAB Online .....                 | 7-50        |
| Variables Editor and Live Editor: Improved speed of data display when scrolling in MATLAB Online .....           | 7-50        |
| lookfor Function: Improved performance when searching .....                                                      | 7-51        |
| <b>Software Development Tools .....</b>                                                                          | <b>7-52</b> |
| Build Tool: Create and run software-build tasks .....                                                            | 7-52        |
| Dependency Analyzer: New warnings to identify problems .....                                                     | 7-52        |
| Project API: Extract project from archive .....                                                                  | 7-52        |
| Source Control in MATLAB Online: Manage Git branches and repositories .....                                      | 7-52        |
| Unit Testing Framework: Add methods and properties to test classes interactively .....                           | 7-52        |
| Unit Testing Framework: Specify action to take against invalid test files .....                                  | 7-52        |
| Unit Testing Framework: Generate descriptive test parameter names using cell arrays .....                        | 7-53        |
| Unit Testing Framework: Compare dictionaries in tests .....                                                      | 7-54        |
| Functionality being removed or changed .....                                                                     | 7-54        |
| <b>External Language Interfaces .....</b>                                                                        | <b>7-56</b> |
| .NET Interface: Support for .NET 5 and .NET Core .....                                                           | 7-56        |
| .NET Engine API: Call MATLAB from .NET applications .....                                                        | 7-56        |

|                                                                                                           |             |
|-----------------------------------------------------------------------------------------------------------|-------------|
| Publish C++ Interface: Support for char* output as nullTerminated string                                  | 7-56        |
| Call MATLAB from C++: Use MATLAB enumeration classes in strongly-typed interface                          | 7-56        |
| Python Interface: Display keyword (key-value pair) arguments created by pyargs                            | 7-56        |
| Python Interface: Convert scalar logical and numeric Python types to MATLAB types                         | 7-57        |
| Python Engine: Pass NumPy arrays directly to MATLAB functions                                             | 7-57        |
| Python Engine: Install MATLAB Engine API with pip Command                                                 | 7-57        |
| Python: Support for Version 3.10                                                                          | 7-58        |
| Functionality being removed or changed                                                                    | 7-58        |
| <b>Hardware Support</b>                                                                                   | <b>7-59</b> |
| Upload Libraries and Configure Pins for I2C, SPI, and Serial Communication Using the Arduino Explorer App | 7-59        |
| Support for servo and rotary encoder peripherals connected to Arduino-compatible ESP32 boards             | 7-59        |

## R2022a

|                                                                                                                    |            |
|--------------------------------------------------------------------------------------------------------------------|------------|
| <b>Environment</b>                                                                                                 | <b>8-2</b> |
| Themes in MATLAB Online: Change the colors of the MATLAB desktop by selecting a dark or light theme                | 8-2        |
| Live Editor Colors: Change the text and background colors of live scripts and functions                            | 8-2        |
| Live Editor Hyperlinks: Insert hyperlinks to specific locations in separate live scripts or live functions         | 8-3        |
| Live Editor Export: Export live scripts and functions programmatically using the export function                   | 8-3        |
| Live Editor Accessibility: Interact with output in live scripts using the keyboard                                 | 8-3        |
| Live Editor Tasks: View and interact with tasks when code is hidden                                                | 8-3        |
| Component Browser: Reorder children in App Designer or the Property Inspector                                      | 8-4        |
| Editor Python Support: View and edit Python files with syntax highlighting, auto-indenting, and delimiter matching | 8-4        |
| Find and Replace Dialog Box: Search text in the Editor and Live Editor using regular expressions                   | 8-4        |
| Profiler: Access the Profiler from the Apps tab                                                                    | 8-4        |
| Internationalization: UTF-8 system encoding on Windows platforms                                                   | 8-5        |
| Installation Settings: Configure persistent settings for MATLAB installations                                      | 8-5        |
| Comparison Tool: Save results as HTML report                                                                       | 8-5        |
| Comparison Tool: Compare folders in MATLAB Online                                                                  | 8-5        |
| MATLAB Drive: macOS 10.15 Catalina will no longer receive updates to MATLAB Drive Connector (April 2022)           | 8-5        |
| Functionality being removed or changed                                                                             | 8-6        |
| <b>Language and Programming</b>                                                                                    | <b>8-7</b> |

|                                                                                                                                                                  |             |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------|
| Class Introspection: Description and DetailedDescription properties of metaclasses contain text from code comments . . . . .                                     | 8-7         |
| Class Introspection: Access class aliases from meta.class instance . . . . .                                                                                     | 8-7         |
| Background Pool: See futures in the background . . . . .                                                                                                         | 8-7         |
| cancelAll Method: Cancel currently queued and running futures in the background pool . . . . .                                                                   | 8-7         |
| Background Pool: Check the status of the background pool . . . . .                                                                                               | 8-7         |
| pcode Function: Create P-code files with enhanced obfuscation . . . . .                                                                                          | 8-7         |
| str2num Function: Restrict evaluation to basic math expressions . . . . .                                                                                        | 8-8         |
| assert Function: Output displays which assertion threw an error and the location in the code . . . . .                                                           | 8-8         |
| Functionality being removed or changed . . . . .                                                                                                                 | 8-8         |
| <b>Data Analysis . . . . .</b>                                                                                                                                   | <b>8-10</b> |
| Data Cleaner App: Interactively preprocess and organize column-oriented data . . . . .                                                                           | 8-10        |
| allfinite, anynan, and anymissing Functions: Determine if all array elements are finite, any element is NaN, and any element is missing . . . . .                | 8-10        |
| quantile, prctile, and iqr Functions: Calculate quantiles, percentiles, and interquartile range . . . . .                                                        | 8-10        |
| rms Function: Calculate root-mean-square value . . . . .                                                                                                         | 8-10        |
| std and var Functions: Optionally return mean as a second output . . . . .                                                                                       | 8-11        |
| Date and Time Functions: Some Financial Toolbox functions combined with MATLAB functions . . . . .                                                               | 8-11        |
| Date and Time Functions: Some Financial Toolbox functions moved to MATLAB . . . . .                                                                              | 8-12        |
| matlab.datetime.compatibility.convertDatenum Function: Convert text timestamps and serial date numbers to datetime values in a backward-compatible way . . . . . | 8-13        |
| categorical Data Type: Use a pattern object to specify category names that match a pattern . . . . .                                                             | 8-13        |
| table and timetable Data Types: Use a pattern object to specify row, variable, and property names that match a pattern . . . . .                                 | 8-14        |
| Data Preprocessing Functions: Append transformed variables to input data using the ReplaceValues name-value argument . . . . .                                   | 8-15        |
| Data Preprocessing Functions: Return table with logical values using the OutputFormat name-value argument . . . . .                                              | 8-15        |
| ismissing, rmmissing, and groupsummary Functions: Accept data types with no standard missing value . . . . .                                                     | 8-16        |
| Functionality being removed or changed . . . . .                                                                                                                 | 8-16        |
| <b>Data Import and Export . . . . .</b>                                                                                                                          | <b>8-18</b> |
| Parquet: Read Parquet file data more efficiently using rowfilter to conditionally filter rows . . . . .                                                          | 8-18        |
| Parquet: Determine and define row groups in Parquet file data . . . . .                                                                                          | 8-18        |
| Parquet: Convert, import, and export nested data structures . . . . .                                                                                            | 8-18        |
| writelines Function: Write plain text to a file . . . . .                                                                                                        | 8-18        |
| Reading Online Data: Use web options when reading files over HTTP and HTTPS . . . . .                                                                            | 8-18        |
| Opus Files: Work with Opus (.opus) audio files. . . . .                                                                                                          | 8-19        |
| HDF5 Interface: Write datasets using dynamically loaded filters . . . . .                                                                                        | 8-19        |
| NetCDF Interface: Enable byte-range reading of remote datasets . . . . .                                                                                         | 8-19        |
| NetCDF Interface: Read and write variable length array data types (NC_VLEN) . . . . .                                                                            | 8-19        |

|                                                                                                                                                |             |
|------------------------------------------------------------------------------------------------------------------------------------------------|-------------|
| Scientific File Format Libraries: NetCDF library is upgraded . . . . .                                                                         | 8-19        |
| Hardware Manager App: Discover and connect to your hardware from<br>MATLAB . . . . .                                                           | 8-19        |
| TCP/IP Client Interface: Specify transfer delay options . . . . .                                                                              | 8-19        |
| Functionality being removed or changed . . . . .                                                                                               | 8-20        |
| <b>Mathematics</b> . . . . .                                                                                                                   | <b>8-21</b> |
| pagemldivide, pagemrdivide, and pageinv Functions: Solve linear equations<br>and calculate matrix inverses using pages of N-D arrays . . . . . | 8-21        |
| tensorprod Function: Calculate tensor products between two arrays . . . .                                                                      | 8-21        |
| round Function: Control tiebreak behavior . . . . .                                                                                            | 8-21        |
| null and orth Functions: Specify tolerance to treat singular values below a<br>threshold as zero . . . . .                                     | 8-21        |
| norm Function: Frobenius norm calculations support N-D arrays . . . . .                                                                        | 8-21        |
| equilibrate Function: Specify output format of factorization . . . . .                                                                         | 8-21        |
| rand, randi, and randn Functions: Support for complex input and<br>RandStream object with the "like" syntax . . . . .                          | 8-22        |
| eps, flintmax, intmax, intmin, realmax, and realmin Functions: Use "like"<br>syntax to return scalars based on prototype object . . . . .      | 8-22        |
| qr and gsvd Functions: Option for economy-size decompositions . . . . .                                                                        | 8-22        |
| Functionality being removed or changed . . . . .                                                                                               | 8-22        |
| <b>Graphics</b> . . . . .                                                                                                                      | <b>8-23</b> |
| Plotting Table Data: Create line plots by passing tables directly to plotting<br>functions . . . . .                                           | 8-23        |
| Data Tips: View table variable names as row labels . . . . .                                                                                   | 8-23        |
| Data Tips: View visual property values for scatter plots and bubble charts<br>. . . . .                                                        | 8-24        |
| Bubble Charts and 3-D Scatter Plots: Plot multiple data sets at once . . . .                                                                   | 8-25        |
| fontname and fontsize Functions: Specify the font and font size for graphics<br>objects . . . . .                                              | 8-26        |
| exportgraphics Function: Create animated GIF files . . . . .                                                                                   | 8-26        |
| Annotation Graphics Objects: Change the annotation rotation angle with the<br>Rotation property . . . . .                                      | 8-26        |
| Quiver Plots: Align the heads, centers, or tails of arrows with data points<br>. . . . .                                                       | 8-26        |
| xlim, ylim, and zlim Functions: Query the axis limit method . . . . .                                                                          | 8-27        |
| view Function: Change the view on multiple axes simultaneously . . . . .                                                                       | 8-27        |
| rendererinfo Function: Get renderer information without specifying the axes<br>. . . . .                                                       | 8-27        |
| linkaxes Function: Synchronize axes in all dimensions by default . . . . .                                                                     | 8-27        |
| cameratoolbar Function: Syntax support for figures created with the<br>uifigure function . . . . .                                             | 8-28        |
| Callbacks in Live Editor: Create callbacks for figures in the Live Editor .                                                                    | 8-28        |
| Figure Code: Generate code for figure interactions in MATLAB Online . .                                                                        | 8-28        |
| Functionality being removed or changed . . . . .                                                                                               | 8-29        |
| <b>App Building</b> . . . . .                                                                                                                  | <b>8-32</b> |
| uistyle Function: Add icons and format text in table cells and tree nodes<br>. . . . .                                                         | 8-32        |
| uitable Function: Rearrange columns of table UI components interactively<br>. . . . .                                                          | 8-32        |

|                                                                                                                               |             |
|-------------------------------------------------------------------------------------------------------------------------------|-------------|
| focus Function: Give keyboard focus to UI components programmatically                                                         | 8-33        |
| isInScrollView Function: Determine if a component is visible in a scrollable container                                        | 8-33        |
| uigridlayout Function: Resize table, list box, and image UI components to fit content                                         | 8-33        |
| Live Editor Tasks: Develop your own Live Editor tasks for use in live scripts and functions                                   | 8-34        |
| Custom UI Components: Interactively create custom UI components in App Designer                                               | 8-34        |
| App Designer: Modify tab focus order of components                                                                            | 8-34        |
| App Designer: Specify error handling options and navigate from error messages when debugging an app                           | 8-35        |
| App Designer: Manage image files in your app with an improved workflow                                                        | 8-35        |
| App Designer: Convert components in a grid layout manager to use pixel-based positioning                                      | 8-35        |
| App Designer: Use App Designer in most modern web browsers in MATLAB Online                                                   | 8-35        |
| App Designer: Customize design environment layout                                                                             | 8-35        |
| Comparison Tool: Compare and merge app files in MATLAB Online                                                                 | 8-36        |
| Functionality being removed or changed                                                                                        | 8-36        |
| <b>Performance</b>                                                                                                            | <b>8-38</b> |
| table Data Type Indexing: Improved performance when subscripting with dot notation or multiple levels of indexing             | 8-38        |
| Classes: Improved performance for static methods, constant property access, and package functions in scripts                  | 8-39        |
| try Block: Improved performance when statements run error-free                                                                | 8-40        |
| Python Data Type Conversion: Improved performance when converting between Python and MATLAB data types in out-of-process mode | 8-41        |
| MATLAB Engine API for Python: Improved performance with large multidimensional arrays in Python                               | 8-42        |
| Matrix multiplication: Improved performance when multiplying sparse and full matrices                                         | 8-43        |
| inv Function: Improved performance when inverting large triangular matrices                                                   | 8-44        |
| sprand and sprandn Functions: Improved performance when generating random sparse matrices                                     | 8-44        |
| fzero Function: Improved performance                                                                                          | 8-45        |
| diff Function: Improved performance with large number of elements                                                             | 8-45        |
| groupsummary, groupfilter, and grouptransform Functions: Improved performance with small group size                           | 8-46        |
| nufftn Function: Improved performance with nonuniform sample points or query points                                           | 8-46        |
| Variables Editor and Live Editor: Improved speed of data display when scrolling                                               | 8-47        |
| App Building: Improved performance when creating UI components                                                                | 8-47        |
| uitable Function and UI Containers: Improved performance when updating properties successively                                | 8-48        |
| UI Components: Improved performance when setting a property with an unchanged value                                           | 8-49        |
| App Designer: Improved performance when loading apps with UIAxes components off the canvas                                    | 8-49        |

|                                                                                                                       |             |
|-----------------------------------------------------------------------------------------------------------------------|-------------|
| Plots in Apps: Improved responsiveness for event-driven updates in apps<br>.....                                      | 8-50        |
| Plots in Apps: Improved responsiveness of axes interactions within apps<br>.....                                      | 8-51        |
| Plots in Apps: Improved responsiveness of axes interactions in plots with<br>two y-axes .....                         | 8-52        |
| Plots in Apps: Faster animations in apps when multiple figures are open<br>.....                                      | 8-53        |
| Property Inspector: Improved performance when opening for the first time<br>.....                                     | 8-54        |
| <b>Software Development Tools .....</b>                                                                               | <b>8-55</b> |
| Projects: Reduce test runtime in continuous integration workflows using the<br>dependency cache .....                 | 8-55        |
| Dependency Analyzer: Save dependency graph as image .....                                                             | 8-55        |
| Code Compatibility Analyzer App: Identify and address compatibility issues<br>against current version of MATLAB ..... | 8-55        |
| Unit Testing Framework: Create test classes interactively using the Current<br>Folder browser .....                   | 8-55        |
| Unit Testing Framework: Create temporary folders that are automatically<br>removed .....                              | 8-55        |
| Unit Testing Framework: Generate DOCX, HTML, and PDF reports after<br>test execution .....                            | 8-56        |
| Unit Testing Framework: Debug uncaught errors in tests .....                                                          | 8-56        |
| Unit Testing Framework: Collect statement and function coverage metrics<br>for your source code .....                 | 8-56        |
| Functionality being removed or changed .....                                                                          | 8-56        |
| <b>External Language Interfaces .....</b>                                                                             | <b>8-58</b> |
| C++ Interface: Array size help text for functions and methods .....                                                   | 8-58        |
| C Interface: Build third-party C library interface using<br>clibgen.generateLibraryDefinition .....                   | 8-58        |
| C++ Interface: Support for C++ language features .....                                                                | 8-58        |
| C++ Interface: Publisher options .....                                                                                | 8-58        |
| Call MATLAB from C++: Generate C++ code Interface for MATLAB<br>Packages, Classes, and Functions .....                | 8-59        |
| MATLAB Data Array API: matlab::data::Array support for row-major order<br>.....                                       | 8-59        |
| MEX Functions: UTF-8 system encoding on Windows platforms .....                                                       | 8-59        |
| Python: Use Name=Value syntax to pass keyword arguments to Python<br>functions .....                                  | 8-59        |
| Python: Convert Python list and tuple types to MATLAB types .....                                                     | 8-59        |
| Perl 5.34.0: MATLAB support on Windows .....                                                                          | 8-60        |
| Compilers: Support for Microsoft Visual Studio 2022 .....                                                             | 8-60        |
| Functionality being removed or changed .....                                                                          | 8-60        |

**R2021b**

|                          |            |
|--------------------------|------------|
| <b>Environment .....</b> | <b>9-2</b> |
|--------------------------|------------|

|                                                                                                                                                   |             |
|---------------------------------------------------------------------------------------------------------------------------------------------------|-------------|
| Editor Selection: Select and edit a rectangular area of code . . . . .                                                                            | 9-2         |
| Editor Display: Zoom in and out in the Editor . . . . .                                                                                           | 9-2         |
| Editor Code: Show code suggestions and completions automatically . . . . .                                                                        | 9-2         |
| Editor Debugging: Diagnose problems in scripts and functions using inline debugging controls and a breadcrumb-style function call stack . . . . . | 9-3         |
| Editor Refactoring: Automatically convert selected code to a function . . . . .                                                                   | 9-4         |
| Editor Code: Automatically complete block endings, match delimiters, and wrap comments while editing code . . . . .                               | 9-4         |
| Editor Sections: Create sections with an improved appearance . . . . .                                                                            | 9-4         |
| Editor Code: Change the case of text and code . . . . .                                                                                           | 9-5         |
| Editor Bookmarks: Maintain bookmarks after closing a file . . . . .                                                                               | 9-5         |
| Live Editor Controls: Set default values for sliders, drop-down lists, check boxes, and edit fields . . . . .                                     | 9-5         |
| Live Editor Animations: Export animations to movies or animated GIFs . . . . .                                                                    | 9-6         |
| Live Editor Figures: Interact with real MATLAB figures and resize them with improved layouts . . . . .                                            | 9-6         |
| Live Editor: Improved performance when saving live scripts or functions . . . . .                                                                 | 9-7         |
| Comparison Tool: Compare and merge text files with improved usability, appearance, and syntax highlighting . . . . .                              | 9-7         |
| Importing Preferences from Previous Releases: MATLAB checks for preferences from R2019b or newer . . . . .                                        | 9-8         |
| Display language: MATLAB uses Windows display language settings for selecting desktop language . . . . .                                          | 9-8         |
| Functionality being removed or changed . . . . .                                                                                                  | 9-8         |
| <b>Language and Programming . . . . .</b>                                                                                                         | <b>9-10</b> |
| cast Function: Consistent output for all syntaxes with the same data type conversion . . . . .                                                    | 9-10        |
| Run Code in the Background: Use parallel language to run code asynchronously . . . . .                                                            | 9-10        |
| Portable Parallel Code: Share parallel code and seamlessly run in parallel . . . . .                                                              | 9-10        |
| Compact Display for Classes: Customize display of information about classes when space is limited . . . . .                                       | 9-11        |
| Class Aliasing: Create aliases for renamed classes to maintain backward compatibility . . . . .                                                   | 9-11        |
| Modular Indexing: Customize class indexing operations individually using new superclasses . . . . .                                               | 9-11        |
| Scalar Classes: Inherit from the matlab.mixin.Scalar superclass to ensure instances behave as scalars . . . . .                                   | 9-12        |
| startat Function: Time zone information in datetime objects now supported . . . . .                                                               | 9-12        |
| Functionality being removed or changed . . . . .                                                                                                  | 9-12        |
| <b>Data Analysis . . . . .</b>                                                                                                                    | <b>9-14</b> |
| Compute by Group Live Editor Task: Interactively summarize, transform, or filter groups of data . . . . .                                         | 9-14        |
| Normalize Data Live Editor Task: Interactively center and scale data . . . . .                                                                    | 9-14        |
| Clean Missing Data Live Editor Task: Define missing values . . . . .                                                                              | 9-14        |
| trenddecomp Function: Find trends in data . . . . .                                                                                               | 9-14        |
| min and max Functions: Specify the comparison method for determining minimum and maximum values . . . . .                                         | 9-14        |



|                                                                                                                                       |             |
|---------------------------------------------------------------------------------------------------------------------------------------|-------------|
| uniquetol Function: Options to control element selection and preserve range of data . . . . .                                         | 9-14        |
| Data Preprocessing Functions: Specify table variable as sample points vector . . . . .                                                | 9-14        |
| dateshift Function: Shift to next occurrence of weekday or weekend day . . . . .                                                      | 9-15        |
| isbetween Function: Support for open, closed, and half open intervals . .                                                             | 9-15        |
| isregular Function: Support for datetime and duration data types . . . . .                                                            | 9-15        |
| istabular Function: Determine if input is a table or timetable . . . . .                                                              | 9-15        |
| retime and synchronize Functions: Median and mode methods supported . . . . .                                                         | 9-15        |
| timeofday Function: Return the date as the second output argument . . .                                                               | 9-15        |
| timeseries2timetable Function: Convert timeseries objects to timetables . . . . .                                                     | 9-16        |
| Functionality being removed or changed . . . . .                                                                                      | 9-16        |
| <b>Data Import and Export . . . . .</b>                                                                                               | <b>9-17</b> |
| sftp Function: Connect to SFTP servers . . . . .                                                                                      | 9-17        |
| Datastores: Specify FileSet objects as data locations for some datastores . . . . .                                                   | 9-17        |
| Table Import: Read tables from HTML and Microsoft Word documents . .                                                                  | 9-17        |
| HDF5 Interface: Use new functionality in support of HDF5 1.10.7 . . . . .                                                             | 9-17        |
| NetCDF Interface: Read and write NC_STRING data . . . . .                                                                             | 9-18        |
| Scientific File Format Libraries: HDF5 and NetCDF libraries are upgraded . . . . .                                                    | 9-18        |
| Audio, Video, and Image I/O Functions: Run functions in a thread-based environment . . . . .                                          | 9-18        |
| Image File Format Libraries: LibTIFF library upgraded to version 4.2.0 .                                                              | 9-18        |
| New Serial Explorer and TCP/IP Explorer apps . . . . .                                                                                | 9-18        |
| Functionality being removed or changed . . . . .                                                                                      | 9-19        |
| <b>Mathematics . . . . .</b>                                                                                                          | <b>9-20</b> |
| ode78 and ode89 Functions: High-order Runge-Kutta solvers for ordinary differential equations . . . . .                               | 9-20        |
| pagesvd Function: Perform singular value decomposition on pages of N-D arrays . . . . .                                               | 9-20        |
| svd Function: Option to control output format of singular values . . . . .                                                            | 9-20        |
| mpower Function: Improved algorithm for defective matrices . . . . .                                                                  | 9-20        |
| Functionality being removed or changed . . . . .                                                                                      | 9-20        |
| <b>Graphics . . . . .</b>                                                                                                             | <b>9-21</b> |
| Plotting Table Data: Create scatter plots, bubble charts, and swarm charts by passing tables directly to plotting functions . . . . . | 9-21        |
| Axes Ticks and Colors: Control the appearance of axis tick marks and tick label colors . . . . .                                      | 9-22        |
| Create Plot Live Task: Add additional visualizations to generated plots . .                                                           | 9-23        |
| Create Plot Live Task: Control chart input syntax using configuration drop-down . . . . .                                             | 9-24        |
| exportgraphics Function: Capture and append graphics to existing PDFs . . . . .                                                       | 9-24        |
| stackedplot Function: Support for semilog y-axes . . . . .                                                                            | 9-24        |
| Text Objects: Use editInteractions in the Interactions property to click or tap on text to edit . . . . .                             | 9-24        |

|                                                                                                                     |             |
|---------------------------------------------------------------------------------------------------------------------|-------------|
| dataTipTextRow Function: Customize data tip content using data properties, such as UserData .....                   | 9-24        |
| MATLAB Online™ Accessibility: Use a screen reader to interact with figures .....                                    | 9-25        |
| Functionality being removed or changed .....                                                                        | 9-25        |
| <b>App Building .....</b>                                                                                           | <b>9-26</b> |
| UIAlert, uiConfirm, and uiProgressDlg Functions: Mark up text and display equations in dialog boxes .....           | 9-26        |
| addStyle Function: Add styles to nodes and levels in a tree UI component .....                                      | 9-26        |
| uitable Function: Set and query table selections programmatically and control table selection options .....         | 9-26        |
| uitextarea Function: Program apps to respond while a user is typing in a text area component .....                  | 9-26        |
| Run Code in the Background: Use parallel language to create more responsive apps .....                              | 9-26        |
| App Designer: Debug code in Code View .....                                                                         | 9-27        |
| App Designer: Efficiently manage your app code with tools and shortcuts from Live Editor .....                      | 9-27        |
| App Designer: Interactively modify canvas zoom level and fit canvas to view .....                                   | 9-29        |
| App Designer: Convert between similar UI components .....                                                           | 9-29        |
| App Designer: Add help text for your app .....                                                                      | 9-30        |
| App Designer: Remove auto-reflow behavior from an app with auto-reflow .....                                        | 9-30        |
| Deployed Web Apps: Deploy web apps directly to the MATLAB Web App Server from within App Designer .....             | 9-30        |
| App Testing Framework: Perform press gestures on axes and UI axes with different selection types .....              | 9-30        |
| App Testing Framework: Perform drag gestures on axes and figures with different selection types .....               | 9-31        |
| App Testing Framework: Use any units of measurement in gestures at the center of components .....                   | 9-31        |
| Functionality being removed or changed .....                                                                        | 9-31        |
| <b>Performance .....</b>                                                                                            | <b>9-33</b> |
| table Data Type Indexing: Improved performance when assigning elements by subscripting with curly braces .....      | 9-33        |
| qrinsert and qrdelete Functions: Improved performance modifying QR factorizations .....                             | 9-33        |
| Titles and Labels in Plots: Improved performance when creating and querying titles or labels in a loop .....        | 9-34        |
| Plot Interactions: Improved performance for rendering data tips and rotating scatter plots of large data sets ..... | 9-35        |
| Plots in Apps: Improved performance for creating plots .....                                                        | 9-35        |
| App Designer: Improved performance when opening Start Page and loading apps .....                                   | 9-36        |
| App Designer: Improved performance when saving apps .....                                                           | 9-36        |
| Comparison Tool: Improved performance when loading and saving MLAPP files .....                                     | 9-37        |
| uigridlayout Function: Improved performance when adding components spanning multiple columns with 'fit' width ..... | 9-37        |

|                                                                                                                  |             |
|------------------------------------------------------------------------------------------------------------------|-------------|
| uigridlayout Function: Improved resizing performance when wrapping text<br>in resizable columns . . . . .        | 9-38        |
| Live Editor: Improved performance when saving live scripts or functions<br>. . . . .                             | 9-39        |
| Data Processing Dialog Boxes: Improved resizing performance . . . . .                                            | 9-39        |
| Figure Interactions: Improved performance when using built-in axes<br>interactions . . . . .                     | 9-39        |
| UI Figures: Improved performance when displaying axes toolbar . . . . .                                          | 9-40        |
| UI Figures: Improved performance when interacting with linked axes . .                                           | 9-40        |
| <b>Software Development Tools . . . . .</b>                                                                      | <b>9-41</b> |
| Projects: Collaborate using projects in MATLAB Online . . . . .                                                  | 9-41        |
| Source Control: Work with files under Git in MATLAB Online . . . . .                                             | 9-41        |
| Unit Testing Framework: Use the TestCase class template to create tests<br>more quickly and accurately . . . . . | 9-41        |
| Unit Testing Framework: Run live-function-based tests interactively in<br>MATLAB Online . . . . .                | 9-41        |
| App Testing Framework: Perform press gestures on axes and UI axes with<br>different selection types . . . . .    | 9-41        |
| App Testing Framework: Perform drag gestures on axes and figures with<br>different selection types . . . . .     | 9-42        |
| App Testing Framework: Use any units of measurement in gestures at the<br>center of components . . . . .         | 9-42        |
| Functionality being removed or changed . . . . .                                                                 | 9-42        |
| <b>External Language Interfaces . . . . .</b>                                                                    | <b>9-44</b> |
| C++ interface: Support for C++ language features . . . . .                                                       | 9-44        |
| C++ interface: Publisher options . . . . .                                                                       | 9-44        |
| Java interface: Specify JRE path for MATLAB . . . . .                                                            | 9-45        |
| Java: Call into MATLAB from a Java program called by MATLAB . . . . .                                            | 9-45        |
| Python interface: Run Python commands and scripts from MATLAB . . . .                                            | 9-45        |
| Python: Support for complex multidimensional arrays . . . . .                                                    | 9-45        |
| Python: Version 3.9 support . . . . .                                                                            | 9-46        |
| WSDL Web Services Documents: Apache CXF version 3.4.2 support . . . .                                            | 9-46        |
| Perl 5.32.1: MATLAB support on Windows . . . . .                                                                 | 9-46        |
| Functionality being removed or changed . . . . .                                                                 | 9-47        |
| <b>Hardware Support . . . . .</b>                                                                                | <b>9-49</b> |
| Connect and Control Arduino board using the Arduino Explorer App . . .                                           | 9-49        |
| Read data from APDS9960 sensor connected to the Arduino hardware . .                                             | 9-49        |
| Support for CAN shields on Raspberry Pi Hardware . . . . .                                                       | 9-49        |

## R2021a

|                                                                                                                                      |             |
|--------------------------------------------------------------------------------------------------------------------------------------|-------------|
| <b>Environment . . . . .</b>                                                                                                         | <b>10-2</b> |
| Live Editor Controls: Create dynamic controls in live scripts by linking<br>variables to drop-down items and slider values . . . . . | 10-2        |

|                                                                                                                                        |              |
|----------------------------------------------------------------------------------------------------------------------------------------|--------------|
| Live Editor Fonts: Change the name, style, size, and color of fonts programmatically using settings . . . . .                          | 10-3         |
| Live Editor Display: Specify where to display output by default . . . . .                                                              | 10-3         |
| Live Editor Functions: Run live functions interactively using the Run button in MATLAB Online . . . . .                                | 10-4         |
| Live Editor Bookmarks: Navigate quickly between lines . . . . .                                                                        | 10-4         |
| Live Editor Animation Playback Controls: Interactive interface to control animations . . . . .                                         | 10-4         |
| Live Editor Performance: Improved performance when saving large live scripts or functions . . . . .                                    | 10-4         |
| Help Browser: View web documentation by default . . . . .                                                                              | 10-5         |
| Documentation: View MATLAB documentation in French, Italian, and German . . . . .                                                      | 10-5         |
| MATLAB Drive: Get the location of your MATLAB Drive root folder programmatically . . . . .                                             | 10-5         |
| MATLAB Drive: Pause and resume syncing in MATLAB Drive Connector . . . . .                                                             | 10-5         |
| MATLAB Drive: MATLAB Drive Connector now available in Chinese and Korean . . . . .                                                     | 10-5         |
| MATLAB Drive: Change the folder permissions for an invited member of a shared folder in MATLAB Drive Online (May 2021) . . . . .       | 10-5         |
| MATLAB Drive: Share folder by invitation to others who already have access to the folder through a view-only link (May 2021) . . . . . | 10-6         |
| Functionality being removed or changed . . . . .                                                                                       | 10-6         |
| <b>Language and Programming . . . . .</b>                                                                                              | <b>10-7</b>  |
| Name=Value Syntax: Use name=value syntax for passing name-value arguments . . . . .                                                    | 10-7         |
| Retrieving Display Format: format function can get and set display format . . . . .                                                    | 10-7         |
| Capturing disp Output: Use the formattedDisplayText function to store disp output as a string . . . . .                                | 10-8         |
| Virtual File Storage: mkdir and rmdir will now be able to create and remove files from VFS directories . . . . .                       | 10-8         |
| Function Argument Validation: Debugger and profiler is now supported . . . . .                                                         | 10-8         |
| Class Diagram Viewer: Create graphical class diagrams to explore class details and share designs . . . . .                             | 10-8         |
| Enumeration Comparisons: Use isequal to compare enumeration members with text data types . . . . .                                     | 10-8         |
| eval function: Context checking to resolve identifiers . . . . .                                                                       | 10-9         |
| Functionality being removed or changed . . . . .                                                                                       | 10-10        |
| <b>Data Analysis . . . . .</b>                                                                                                         | <b>10-12</b> |
| Data Preprocessing Live Editor Tasks: Operate on multiple table variables and specify output format for table input . . . . .          | 10-12        |
| Clean Outlier Data Live Editor Task: Visualize results with a histogram . . . . .                                                      | 10-12        |
| fillmissing Function: Specify custom fill method . . . . .                                                                             | 10-12        |
| normalize Function: Normalize multiple data sets with same parameters . . . . .                                                        | 10-12        |
| groupcounts Function: Display percentages of group counts . . . . .                                                                    | 10-13        |
| ts2timetable Function: Convert timeseries objects to timetables . . . . .                                                              | 10-13        |

|                                                                                                                                      |              |
|--------------------------------------------------------------------------------------------------------------------------------------|--------------|
| table and timetable Functions: Specify dimension names using the<br>'DimensionNames' name-value argument .....                       | 10-13        |
| Functionality being removed or changed .....                                                                                         | 10-13        |
| <b>Data Import and Export .....</b>                                                                                                  | <b>10-15</b> |
| XML Files: Read, write, and import XML files using readtable,<br>readtimetable, and other functions .....                            | 10-15        |
| MATLAB API for Advanced XML Processing: Create, read, write, transform,<br>and query XML .....                                       | 10-15        |
| XML Files: Register XML namespace prefixes for evaluating XPath<br>expressions using readtable,readstruct, and other functions ..... | 10-15        |
| Low-level file I/O functions and remote data: Perform read and write<br>operations on remotely stored files .....                    | 10-16        |
| save and load functions and remote data: Save, load, and append data to<br>remotely stored v7.3 MAT-files .....                      | 10-16        |
| Reading Online Data: Read files over HTTP and HTTPS using readtable,<br>audioread, and other reading functions .....                 | 10-16        |
| Parquet Data Format: Use categorical data in parquet data format ....                                                                | 10-17        |
| Datastores: Read all data from a datastore using parallel processing ..                                                              | 10-17        |
| Data Compression Functions: Improved functionality in zip/unzip and tar/<br>untar .....                                              | 10-17        |
| imfinfo function: Get information about all Adobe Digital Negative (DNG)<br>file tags .....                                          | 10-17        |
| jsonencode: Add indentation to JSON text .....                                                                                       | 10-17        |
| Functionality being removed or changed .....                                                                                         | 10-18        |
| <b>Mathematics .....</b>                                                                                                             | <b>10-20</b> |
| Graph Algorithms: Compute all paths, all cycles, and cycle basis .....                                                               | 10-20        |
| griddedInterpolant Object: Use multivalued interpolation to interpolate<br>multiple data sets simultaneously .....                   | 10-20        |
| eig Function: Improved algorithm for skew-Hermitian matrices .....                                                                   | 10-20        |
| cdf2rdf Function: Improved algorithm for all inputs .....                                                                            | 10-20        |
| Functionality being removed or changed .....                                                                                         | 10-20        |
| <b>Graphics .....</b>                                                                                                                | <b>10-22</b> |
| Create Plot Live Editor Task: Create plots interactively and generate code<br>.....                                                  | 10-22        |
| bubblecloud Function: Visualize part-to-whole relationships .....                                                                    | 10-22        |
| tiledlayout Function: Control the tile indexing scheme .....                                                                         | 10-22        |
| PolarAxes Objects: Use the CurrentPoint property or call ginput to get the<br>cursor location within polar axes .....                | 10-23        |
| Scatter Plots and Constant Lines: Create multiple scatter plots or constant<br>lines at once .....                                   | 10-23        |
| Axis Limits: Define LimitsChangedFcn callback that executes when the<br>limits of an axis change .....                               | 10-23        |
| Axis Limits: Control axis limits .....                                                                                               | 10-23        |
| exportgraphics and copygraphics Functions: Specify RGB, CMYK, or<br>grayscale output .....                                           | 10-24        |
| colororder Function: Control colors in stacked plots .....                                                                           | 10-24        |
| Tick Labels: Automatically rotate tick labels .....                                                                                  | 10-24        |
| patch and errorbar Functions: Expanded data type support .....                                                                       | 10-25        |
| Geographic Plots: Access basemaps using additional proxy server<br>authentication types .....                                        | 10-25        |

|                                                                                                                                       |              |
|---------------------------------------------------------------------------------------------------------------------------------------|--------------|
| Functionality being removed or changed . . . . .                                                                                      | 10-25        |
| <b>App Building . . . . .</b>                                                                                                         | <b>10-31</b> |
| uihyperlink Function: Add and configure clickable links in apps and on the App Designer canvas . . . . .                              | 10-31        |
| uitree Function: Add and configure check box trees in apps and on the App Designer canvas . . . . .                                   | 10-31        |
| Interpreter Property: Style text and display equations in labels with HTML and LaTeX markup . . . . .                                 | 10-31        |
| WindowStyle Property: Create UI figures that remain in the foreground . . . . .                                                       | 10-31        |
| scroll Function: Scroll to a location within a table UI component programmatically . . . . .                                          | 10-31        |
| UI Component Accessibility: Select ListBox items, Table cells, ColorPicker colors, and DatePicker menus using the keyboard . . . . .  | 10-32        |
| App Designer: Use custom UI components in App Designer . . . . .                                                                      | 10-32        |
| App Designer: Zoom and pan in the canvas, and zoom in the Code View editor . . . . .                                                  | 10-33        |
| App Designer: Control color and tab settings in Code View using MATLAB preferences . . . . .                                          | 10-33        |
| App Designer: Customize split-screen layouts in the App Designer editor . . . . .                                                     | 10-33        |
| App Testing Framework: Perform gestures on panels and tables . . . . .                                                                | 10-33        |
| App Testing Framework: Close alert dialog box in front of figure window . . . . .                                                     | 10-34        |
| Web Apps and Standalone Applications: Datatips supported in graphics . . . . .                                                        | 10-34        |
| Functionality Being Removed or Changed . . . . .                                                                                      | 10-34        |
| <b>Performance . . . . .</b>                                                                                                          | <b>10-36</b> |
| Sparse Matrix Multiplication: Improved performance multiplying large sparse matrices . . . . .                                        | 10-36        |
| Sparse Linear Systems: Improved performance solving sparse linear systems $A \cdot X = B$ with multicolumn B . . . . .                | 10-36        |
| vecnorm Function: Improved performance operating on data with multiple columns . . . . .                                              | 10-37        |
| ismember Function: Improved performance for cell inputs . . . . .                                                                     | 10-37        |
| unique Function: Improved performance for numeric, logical, char, and cell inputs . . . . .                                           | 10-38        |
| Graph Functions: Improved performance modifying node and edge lists . . . . .                                                         | 10-39        |
| Axes Toolbar: Appears without delay when axes are ready . . . . .                                                                     | 10-39        |
| Rearranging UI Components: Improved performance when rearranging UI components in a UI figure . . . . .                               | 10-39        |
| UI Figure Interactions: Faster responses to scroll, pointer movement, and resize interactions in UI figures . . . . .                 | 10-40        |
| Plots in Apps: Improved performance for polar plots, volume visualizations, plots with more than 16 axes, and older systems . . . . . | 10-41        |
| Plots in Apps: Improved performance for plots with large numbers of markers . . . . .                                                 | 10-42        |
| Live Editor: Improved performance when saving large live scripts or functions . . . . .                                               | 10-42        |
| <b>Software Development Tools . . . . .</b>                                                                                           | <b>10-44</b> |

|                                                                                                 |              |
|-------------------------------------------------------------------------------------------------|--------------|
| Projects: List all referenced projects of the current project . . . . .                         | 10-44        |
| Projects: List impacted project files . . . . .                                                 | 10-44        |
| Dependency Analyzer: Find required add-ons . . . . .                                            | 10-44        |
| Unit Testing Framework: Create test runners using alternative syntax . . . . .                  | 10-44        |
| Unit Testing Framework: Initialize parameterization properties at suite creation time . . . . . | 10-44        |
| Unit Testing Framework: Run tests in parallel on thread-based pool . . . . .                    | 10-44        |
| Unit Testing Framework: Run tests in MATLAB Online interactively . . . . .                      | 10-45        |
| App Testing Framework: Perform gestures on panels and tables . . . . .                          | 10-45        |
| App Testing Framework: Close alert dialog box in front of figure window . . . . .               | 10-45        |
| Functionality being removed or changed . . . . .                                                | 10-45        |
| <b>External Language Interfaces . . . . .</b>                                                   | <b>10-46</b> |
| C++ Interface: Support for C++ language features . . . . .                                      | 10-46        |
| C++ Interface: Publisher options and analysis . . . . .                                         | 10-46        |
| Java Packages to be removed . . . . .                                                           | 10-46        |
| Java Engine: MATLAB value object support . . . . .                                              | 10-46        |
| Python Interface and Engine: Version 3.6 support discontinued . . . . .                         | 10-46        |
| Perl 5.32.0: MATLAB support on Windows . . . . .                                                | 10-47        |
| <b>Hardware Support . . . . .</b>                                                               | <b>10-48</b> |
| Support added for IMU sensors . . . . .                                                         | 10-48        |
| New functionalities added to Raspberry Pi Resource Monitor app . . . . .                        | 10-48        |

## R2020b

|                                                                                                                                 |             |
|---------------------------------------------------------------------------------------------------------------------------------|-------------|
| <b>Environment . . . . .</b>                                                                                                    | <b>11-2</b> |
| MATLAB Online Accessibility: Use a screen reader to interact with the Command Window and create scripts and functions . . . . . | 11-2        |
| Live Editor Images: Add alternative text to images . . . . .                                                                    | 11-2        |
| Live Editor Images: Change the size of images . . . . .                                                                         | 11-2        |
| Live Editor Hyperlinks: Navigate to existing files from a live script or live function using links . . . . .                    | 11-2        |
| Live Editor Export: Export all live scripts and live functions in a folder to a standard format . . . . .                       | 11-2        |
| matlabRelease Object: Query MATLAB Release Information . . . . .                                                                | 11-3        |
| Query Parallel Functionality: Determine if support for Parallel Computing Toolbox functionality is available . . . . .          | 11-3        |
| Comparison Tool: Compare text files in MATLAB Online . . . . .                                                                  | 11-3        |
| MATLAB Drive: Decreased password confirmation requests in MATLAB Drive Connector . . . . .                                      | 11-3        |
| MATLAB Drive: Faster shut down of MATLAB Drive Connector when syncing is in progress . . . . .                                  | 11-4        |
| MATLAB Drive: Integration with OS-specific password management systems . . . . .                                                | 11-4        |
| MATLAB Drive: MATLAB Drive Connector available in Japanese . . . . .                                                            | 11-4        |
| MATLAB Drive: Download folders from MATLAB Drive online to your desktop (December 2020) . . . . .                               | 11-4        |

|                                                                                                                                                                                                       |              |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------|
| MATLAB Drive: Preview the contents of a shared folder in MATLAB Drive online before accepting invitation (December 2020) . . . . .                                                                    | 11-4         |
| <b>Language and Programming . . . . .</b>                                                                                                                                                             | <b>11-5</b>  |
| pattern Object and Functions: Match patterns in text functions . . . . .                                                                                                                              | 11-5         |
| extract Function: Extract substrings from strings . . . . .                                                                                                                                           | 11-5         |
| Functions: New validation functions for arguments and properties . . . . .                                                                                                                            | 11-5         |
| underlyingType, isUnderlyingType, and mustBeUnderlyingType Functions:<br>Query the underlying data type of classes . . . . .                                                                          | 11-6         |
| height and width Functions: Return number of rows or columns in an array<br>. . . . .                                                                                                                 | 11-6         |
| Class conversions: Assignment operations convert more classes into built-in<br>data types . . . . .                                                                                                   | 11-6         |
| Functionality being removed or changed . . . . .                                                                                                                                                      | 11-7         |
| <b>Data Analysis . . . . .</b>                                                                                                                                                                        | <b>11-8</b>  |
| Implicit Expansion: For calendarDuration, categorical, datetime, and<br>duration arrays, automatically expand dimensions of length 1 when<br>applying element-wise operations and functions . . . . . | 11-8         |
| normalize Function: Scale data by interquartile range . . . . .                                                                                                                                       | 11-9         |
| groupsummary Function: Summarize data using functions that require<br>multiple input arguments . . . . .                                                                                              | 11-9         |
| fillmissing Function and Clean Missing Data Live Editor Task: Specify<br>maximum gap size to fill . . . . .                                                                                           | 11-9         |
| Clean Outlier Data Live Editor Task: Define outliers based on percentile<br>thresholds . . . . .                                                                                                      | 11-9         |
| Functionality being removed or changed . . . . .                                                                                                                                                      | 11-9         |
| <b>Data Import and Export . . . . .</b>                                                                                                                                                               | <b>11-11</b> |
| readstruct and writestruct functions: Read and write structured data in<br>XML files . . . . .                                                                                                        | 11-11        |
| readlines function: Read the lines in a text file as a string array . . . . .                                                                                                                         | 11-11        |
| Spreadsheet files: Customize formatting when writing data to spreadsheet<br>files with PreserveFormat and AutoFitWidth . . . . .                                                                      | 11-11        |
| imread function and Tiff object: Read images from Aperio SVS and TIFF<br>files containing JPEG2000 compression . . . . .                                                                              | 11-11        |
| ArrayDatastore object: Create datastores from in-memory data . . . . .                                                                                                                                | 11-11        |
| Datastore: Transform multiple datastores using the transform function<br>. . . . .                                                                                                                    | 11-11        |
| FileDatastore object: Shuffle and create subsets of a FileDatastore . . . . .                                                                                                                         | 11-12        |
| writeall function: Write data from text and spreadsheet files to different row<br>groups in Parquet files . . . . .                                                                                   | 11-12        |
| fileparts function: Parse file names specified as cell arrays of character<br>vectors and string arrays . . . . .                                                                                     | 11-12        |
| Audio devices: Refresh the available audio devices using the audiodevreset<br>function . . . . .                                                                                                      | 11-12        |
| Audio files and web-based data: Read and write remotely stored audio files<br>using audioread, audiowrite, and audioinfo . . . . .                                                                    | 11-12        |
| HDF5 files and web-based data: Read and write remotely-stored HDF5 files<br>using existing HDF5 functions . . . . .                                                                                   | 11-13        |
| HDF5 files: Read and write file names encoded using Unicode characters<br>. . . . .                                                                                                                   | 11-13        |
| Scientific File Format Libraries: NetCDF library upgraded to 4.7.3 . . . . .                                                                                                                          | 11-13        |



|                                                                                                                       |       |
|-----------------------------------------------------------------------------------------------------------------------|-------|
| Image File Format Libraries: LibTIFF library upgraded to version 4.1.0                                                | 11-13 |
| Bluetooth Interface: Support for communicating with Bluetooth devices                                                 | 11-13 |
| TCP/IP Client Interface: New functions and properties                                                                 | 11-14 |
| Serial Port Interface: Improved performance                                                                           | 11-14 |
| Functionality being removed or changed                                                                                | 11-15 |
| <b>Mathematics</b>                                                                                                    | 11-17 |
| Optimize Live Editor Task: Solve optimization problems interactively                                                  | 11-17 |
| pagetimes Function: Perform matrix multiplication on pages of N-D arrays                                              | 11-17 |
| pagetranspose and pagectranspose Functions: Transpose pages of N-D arrays                                             | 11-17 |
| svdsketch Function: Compute SVD factors of low-rank matrix sketch                                                     | 11-17 |
| Functionality being removed or changed                                                                                | 11-17 |
| <b>Graphics</b>                                                                                                       | 11-18 |
| bubblechart, bubblechart3, and polarbubblechart Functions: Create bubble charts in 2-D, 3-D, and in polar coordinates | 11-18 |
| Swarm charts and Scatter objects: Visualize distributions of discrete data                                            | 11-18 |
| scatter Function: Vary the transparency across all points                                                             | 11-19 |
| tiledlayout and nexttile Functions: Improved placement of legends, and colorbars, and shared decorations              | 11-19 |
| axis Function: Pad axis limits to show plotted data near the limits more clearly                                      | 11-20 |
| Titles, Subtitles, and Axis Labels: Add subtitles to plots, and align titles and axis labels with the plot box        | 11-21 |
| Data Tips: Customize data tip content on standalone visualizations                                                    | 11-22 |
| turbo Colormap: jet colormap alternative with more perceptually uniform transitions                                   | 11-23 |
| Colormap Editor: Customize colormaps using modernized interface                                                       | 11-23 |
| boxchart Function: Use color to differentiate between box charts                                                      | 11-24 |
| im2gray and cmap2gray: Convert images and colormaps to grayscale                                                      | 11-24 |
| validatecolor Function: Calculate normalized RGB triplets for color names, hexadecimal color codes, or integer values | 11-24 |
| Markers: Specify horizontal or vertical line markers for plots                                                        | 11-25 |
| surf and meshc Functions: Specify Z-level for contours on surface and mesh plots                                      | 11-25 |
| animatedline Function: Create animated lines in polar plots                                                           | 11-26 |
| colororder Function: Control colors in geographic bubble charts                                                       | 11-26 |
| Functionality being removed or changed                                                                                | 11-26 |
| <b>App Building</b>                                                                                                   | 11-29 |
| uitable Function: Configure column widths to use weighted variable or to automatically adjust to fit data             | 11-29 |
| scroll Function: Scroll to the top or bottom of a text area programmatically                                          | 11-29 |
| WindowStyle Property: Create modal UI figures                                                                         | 11-29 |
| Icon Property: Specify custom icons for UI figure windows and toolbar push and toggle tools                           | 11-29 |

|                                                                                                                                                        |              |
|--------------------------------------------------------------------------------------------------------------------------------------------------------|--------------|
| WordWrap Property: Wrap long text to fit the width of certain UI components .....                                                                      | 11-29        |
| Enable Property: Turn interaction off and on for buttons and panel groups .....                                                                        | 11-30        |
| BackgroundColor Property: Set the background color for grid layouts ..                                                                                 | 11-30        |
| Custom Components: Develop your own class of UI components .....                                                                                       | 11-30        |
| App Designer: Allow only one running instance of your app at a time ..                                                                                 | 11-30        |
| App Designer: Change the stacking order of UI components .....                                                                                         | 11-31        |
| App Designer: Add and configure toolbar components on the App Designer canvas .....                                                                    | 11-31        |
| App Designer: Draw UI components on the App Designer canvas .....                                                                                      | 11-31        |
| App Designer: Find differences and merge apps .....                                                                                                    | 11-31        |
| Graphics Support: Create more plots in apps with full support for any type of axes .....                                                               | 11-32        |
| Graphics Support: Identify coordinates and display text by clicking or tapping .....                                                                   | 11-32        |
| App Capture: Capture user interfaces using exportapp and getframe ..                                                                                   | 11-32        |
| App Testing Framework: Perform choose gestures on context menu items .....                                                                             | 11-33        |
| App Testing Framework: Perform drag gestures on axes and UI axes ..                                                                                    | 11-33        |
| App Testing Framework: Perform gestures on push tools and toggle tools .....                                                                           | 11-33        |
| Functionality Being Removed or Changed .....                                                                                                           | 11-33        |
| <b>Performance .....</b>                                                                                                                               | <b>11-37</b> |
| sum Function: Improved performance summing the first dimension of numeric arrays .....                                                                 | 11-37        |
| polyfit Function: Improved performance fitting data .....                                                                                              | 11-37        |
| accumarray Function: Improved performance with fill values and certain function handles .....                                                          | 11-38        |
| spdiags Function: Improved performance constructing sparse banded matrices .....                                                                       | 11-39        |
| uilibox: Improved performance when setting multiple items in a list box .....                                                                          | 11-40        |
| uitree: Improved performance when creating many nodes in a tree ...                                                                                    | 11-40        |
| Data Tip Markers: Improved rendering performance of data tip markers in surface plots of large data sets created in UI figures and MATLAB Online ..... | 11-41        |
| <b>Software Development Tools .....</b>                                                                                                                | <b>11-42</b> |
| Code Compatibility Report: Unsupported Functionality Will Now Issue Warning .....                                                                      | 11-42        |
| Dependency Analyzer: Export to archive and generate a dependency report .....                                                                          | 11-42        |
| Source Control: Improved workflow to set up Git source control .....                                                                                   | 11-42        |
| Projects: Change project definition file type and preserve source control history .....                                                                | 11-42        |
| Unit Testing Framework: Run tests in parallel on clusters and clouds ..                                                                                | 11-42        |
| Unit Testing Framework: Run tests in parallel with standalone applications .....                                                                       | 11-43        |
| Unit Testing Framework: Report the validity of shared test fixtures ...                                                                                | 11-43        |
| App Testing Framework: Perform choose gestures on context menu items .....                                                                             | 11-43        |
| App Testing Framework: Perform drag gestures on axes and UI axes ..                                                                                    | 11-43        |

|                                                                                                                               |              |
|-------------------------------------------------------------------------------------------------------------------------------|--------------|
| App Testing Framework: Perform gestures on push tools and toggle tools .....                                                  | <b>11-43</b> |
| Functionality being removed or changed .....                                                                                  | <b>11-44</b> |
| <b>External Language Interfaces .....</b>                                                                                     | <b>11-46</b> |
| C++ Interface: Support for nullptr .....                                                                                      | <b>11-46</b> |
| C++ Interface: Create interface with C++ source files .....                                                                   | <b>11-46</b> |
| Python: Version 3.8 support .....                                                                                             | <b>11-46</b> |
| Python: Terminate Python interpreter and start new one in same MATLAB session .....                                           | <b>11-46</b> |
| mxCreateString C Matrix API functions: UTF-8 support .....                                                                    | <b>11-46</b> |
| MATLAB Data API: Create matlab::data::Object arrays .....                                                                     | <b>11-46</b> |
| Compiler support changed for building C++ interfaces, MEX files, and standalone MATLAB engine and MAT-file applications ..... | <b>11-47</b> |
| Functionality being removed or changed .....                                                                                  | <b>11-47</b> |
| <b>Hardware Support .....</b>                                                                                                 | <b>11-48</b> |
| Live Editor Task: Interactively capture images from USB Webcam interactively and generate MATLAB code in a live script. ....  | <b>11-48</b> |
| Functionality being removed or changed .....                                                                                  | <b>11-48</b> |



# R2025b

---

**Version: 25.2**

**New Features**

**Bug Fixes**

## Quality and stability improvements

R2025b delivers quality and stability improvements, building on the new features introduced in R2025a.

### getframe Function: Improved performance when capturing frames in loop

Capturing frames in a loop using the `getframe` function shows improved performance. The improvement becomes more noticeable as the size of the figure and number of frames increase.

For example, create a figure that is 1074-by-647 pixels in size and plot a surface. Change the shape of the surface 100 times in a loop, and capture a frame in each iteration. This code is about 1.5x faster than in the previous release.

```
function timingCaptureFrames
f = figure(Position=[0 0 1074 647]);
Z = peaks;
s = surf(Z);
loops = 100;
M(loops) = struct(cdata=[],colormap=[]);
axis tight manual
tic
for k = 1:loops
    Zframe = sin(k*pi/10)*Z;
    s.ZData = Zframe;
    M(k) = getframe(f);
end
toc
end
```

The approximate execution times are:

**R2025a:** 7.79 s

**R2025b:** 5.18 s

The code was timed on a Windows® 11, Intel Core® i7-8665U CPU 8-Core Processor @ 1.90 GHz test system with a Microsoft® Remote Display Adapter and Intel® UHD Graphics 620 by calling the `timingCaptureFrames` function.

## MATLAB Support Package for Quantum Computing: Solve knapsack and traveling salesperson problems using QUBO (September 2025, Version 25.2.0)

You can solve knapsack and traveling salesperson problems as QUBO problems. Convert the knapsack or traveling salesperson problem to an equivalent QUBO formulation using the `knapsack2qubo` or `tsp2qubo` function, respectively, which returns a `qubo` object. Solve the QUBO problem by using the `solve` function, which returns a `quboResult` object. You can then convert the QUBO result back to the knapsack or traveling salesperson formulation using the `quboResult2knapsack` or `quboResult2tsp` function, respectively.

---

## **MATLAB Support Package for Quantum Computing: Calculate expectation values using quantum devices (November 2025, Version 25.2.1)**

When you run a quantum circuit using the `run` function, you can specify the possible Pauli basis measurement results by using the new `Observable` name-value argument. If you specify `Observable`, then the `run` function returns a `QuantumTaskAWS` or `QuantumTaskIBM` object whose result is the expectation value of measuring the quantum circuit in the basis specified by the observable. You can then retrieve the result of the quantum task object as a numeric expectation value using the `fetchOutput` function.





# R2025a

---

**Version: 25.1**

**New Features**

**Bug Fixes**

**Compatibility Considerations**

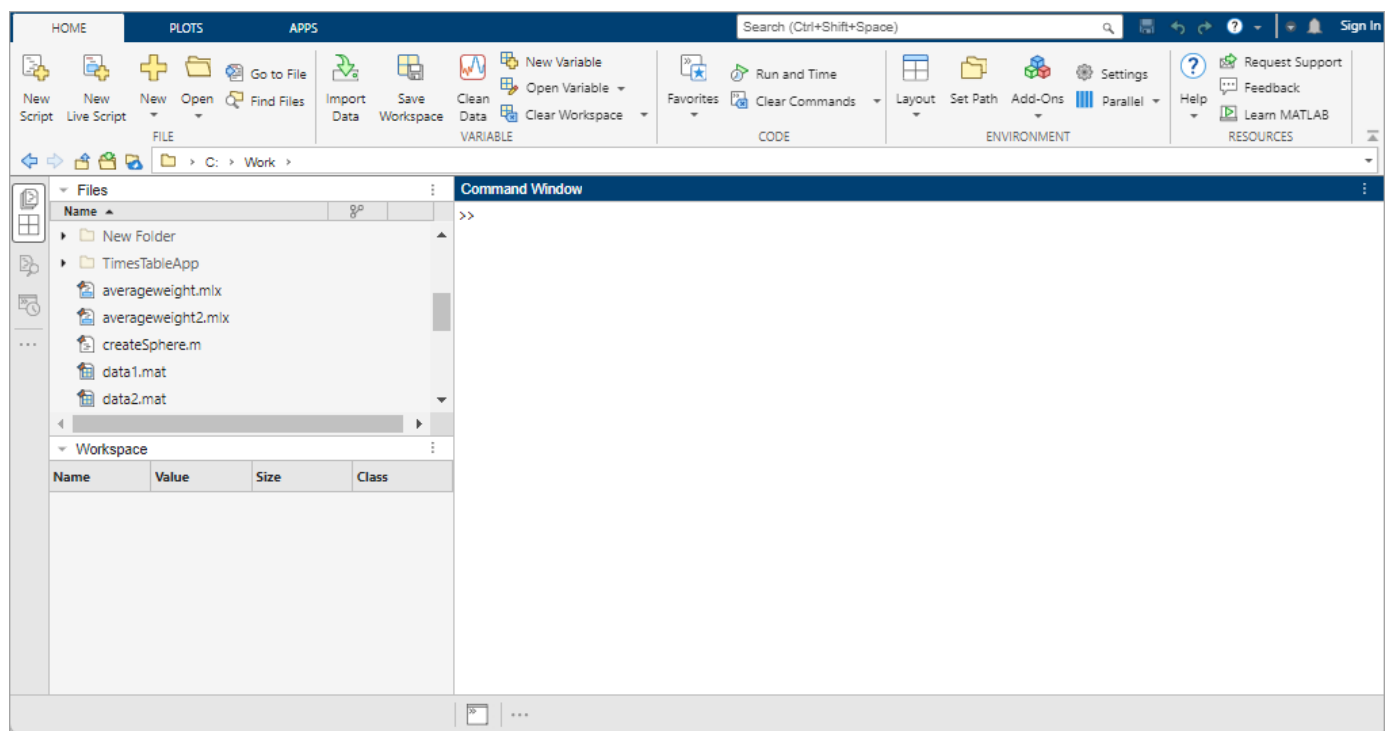
## Environment

### ★ MATLAB Desktop: Access desktop using new layout with customizable sidebars

In R2025a, the MATLAB desktop has a new layout designed to improve productivity and streamline workflows. The new layout includes customizable sidebars on the left, right, and bottom of the desktop. The sidebars show the tools, such as the Files panel (previously referred to as the Current Folder browser), the Workspace panel, and the Command Window, that are docked in the desktop.

You can use the sidebars to access the desktop tools in panels and change the desktop layout. For example, to show or hide a panel, click its icon on the sidebar. To move a panel, drag the panel's title bar or sidebar icon to the left, right, or bottom of the desktop. To open a panel that is not already open, click the Open more panels button \*\*\* in any sidebar. Then, in the Open Panel dialog box, select from the available panels.

For more information about how to configure your desktop, see [Configure the Desktop](#).

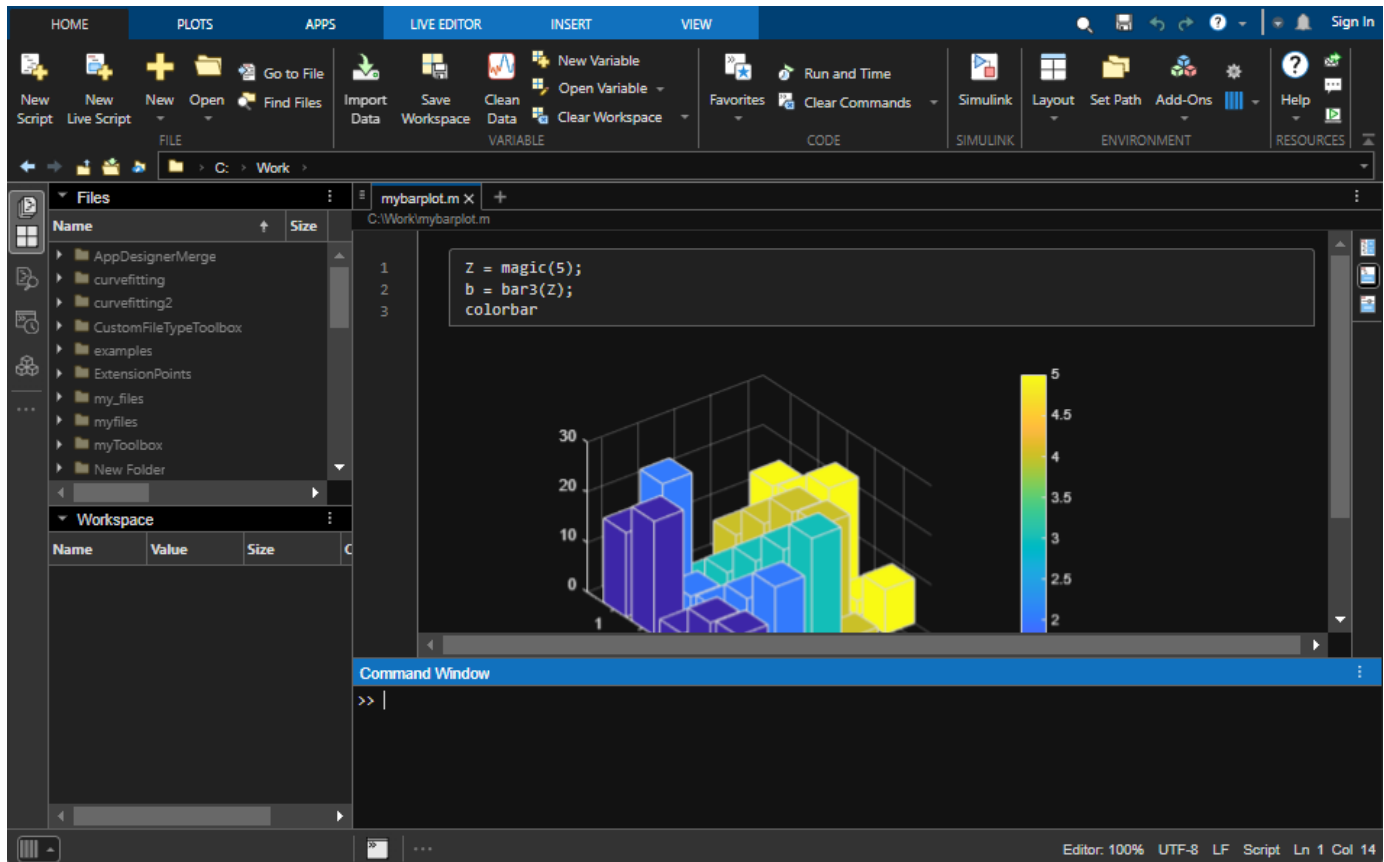


As part of this change, undocking some tools, such as the Files panel and Workspace panel, is no longer supported. Undocking the Command Window, Find Files panel, and Variables editor is still supported. In addition, customized layouts saved in a previous release of MATLAB are not supported.

## ★ Desktop Themes: Change MATLAB desktop colors by selecting dark or light theme

You can change the colors of the MATLAB desktop using themes. For example, to select the dark theme, on the **Home** tab, in the **Environment** section, click  **Settings**. Select **MATLAB > Appearance** and set **Desktop Theme** to Dark.

For more information, see [Change Desktop Theme and Colors](#).



## ★ Desktop Accessibility: Use screen reader and keyboard to interact with different desktop areas

You can use a screen reader and keyboard to interact with the Command Window, navigate through the MATLAB desktop tools, and create and edit files in the Editor and Live Editor.

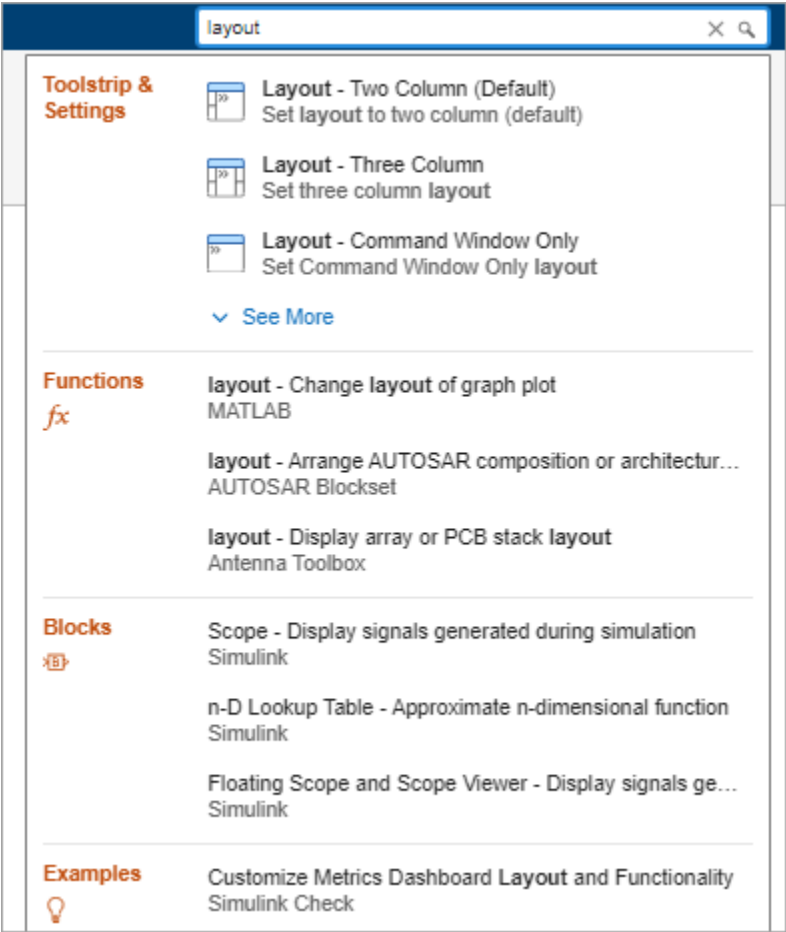
For more information, see [Accessibility in MATLAB](#).

## ★ Desktop Search: Access toolstrip actions, settings, and Help Center resources using search

You can now use the search box in the top-right corner of the desktop to easily access toolstrip actions, settings, and Help Center resources such as functions, blocks, examples, answers, and

videos. To navigate to the search box using a keyboard, press **Ctrl+Shift+Space** (or **Command+Shift+Space** on macOS systems).

For example, to restore the MATLAB desktop to its default layout, type the word `layout` in the search box. In the list of results, select the **Layout - Two Column (Default)** toolbar action.

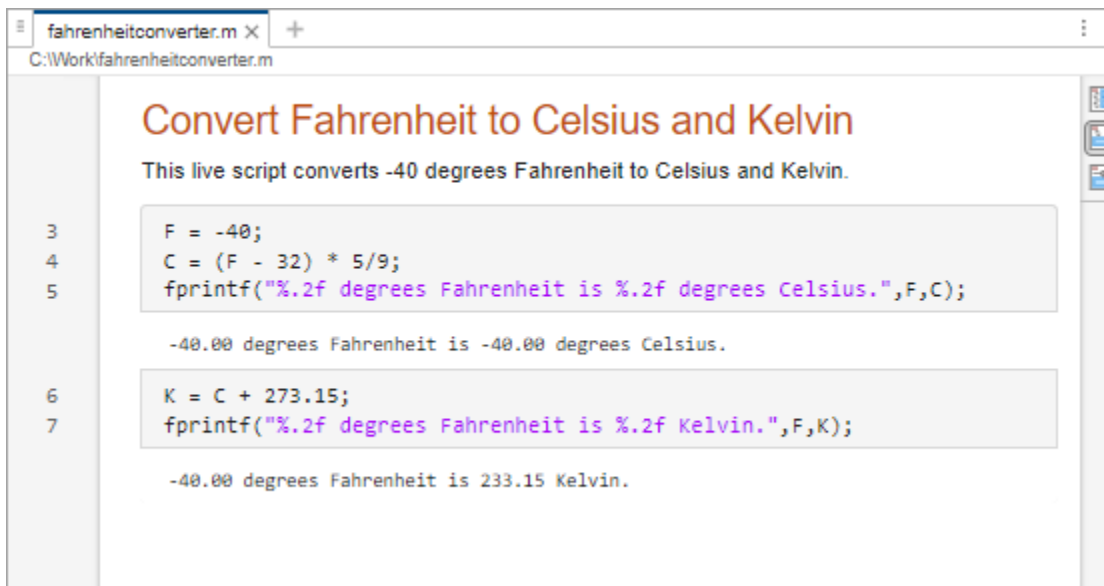


★ **Live Editor File Format: Create live scripts using new plain text file format**

You can now create and save live scripts and functions using a plain text file format (.m). This new file format for live scripts improves integration with source control and allows you to open live scripts in external text editors.

Live scripts saved using the plain text file format behave just like other live scripts. They open in the Live Editor and can contain code, output, and formatted text. They also can include interactive controls and tasks.

For more information, see Live Code File Format (.m).



## Live Editor Videos: Add YouTube videos and video files to your live scripts

You can add YouTube® videos and video files to your live scripts. Online videos are subject to the terms of service and privacy policy of the video provider.

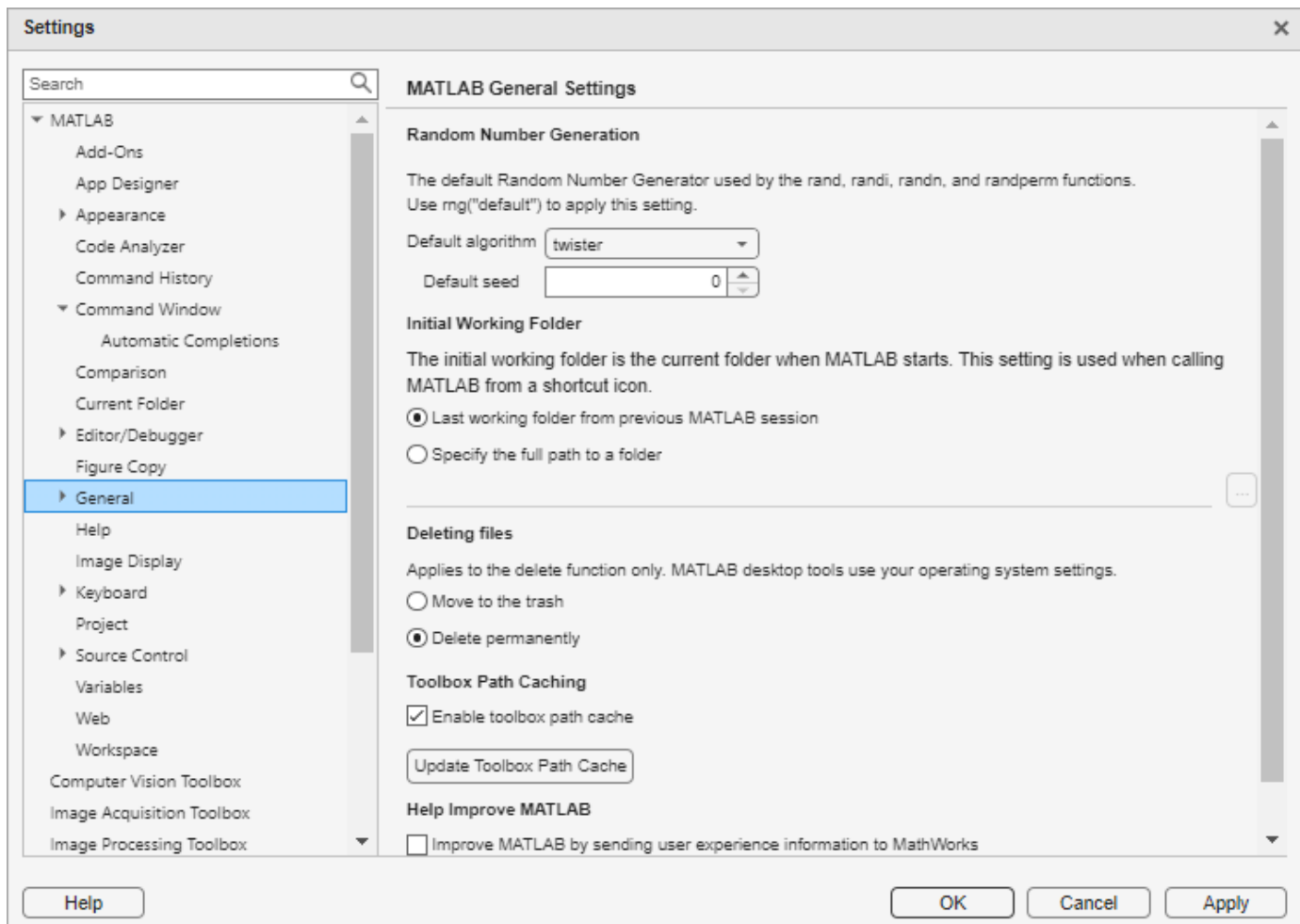
To add a video to a live script, go to the **Insert** tab, and select **Video > Online Video** or **Video > Video File**. You can add videos only in text lines. If you insert a video in a code line, the Live Editor places it in a text line below the current code line.



## MATLAB Settings: Use Settings window to search for and modify MATLAB settings

MATLAB preferences are now renamed as settings, and you can search for and modify these settings using the updated Settings window. To open the Settings window, go to the **Home** tab, and in the **Environment** section, click  **Settings**.

For more information, see Settings Window.



## Command Window: Show code suggestions automatically

Starting in R2025a, when you write code in the Command Window, MATLAB automatically provides real-time suggestions as you type. These suggestions include the names of functions, models, MATLAB objects, files, folders, variables, structures, and graphics properties. In addition, when you type a function name, MATLAB shows basic syntax information for the function and offers suggestions for parameters and options.

To disable or change the default behavior of code suggestions in the Command Window, go to the **Home** tab, and in the **Environment** section, click  **Settings**. Select **MATLAB > Command Window > Suggestions** and in the **Suggestions** section, configure the settings based on your preferences.

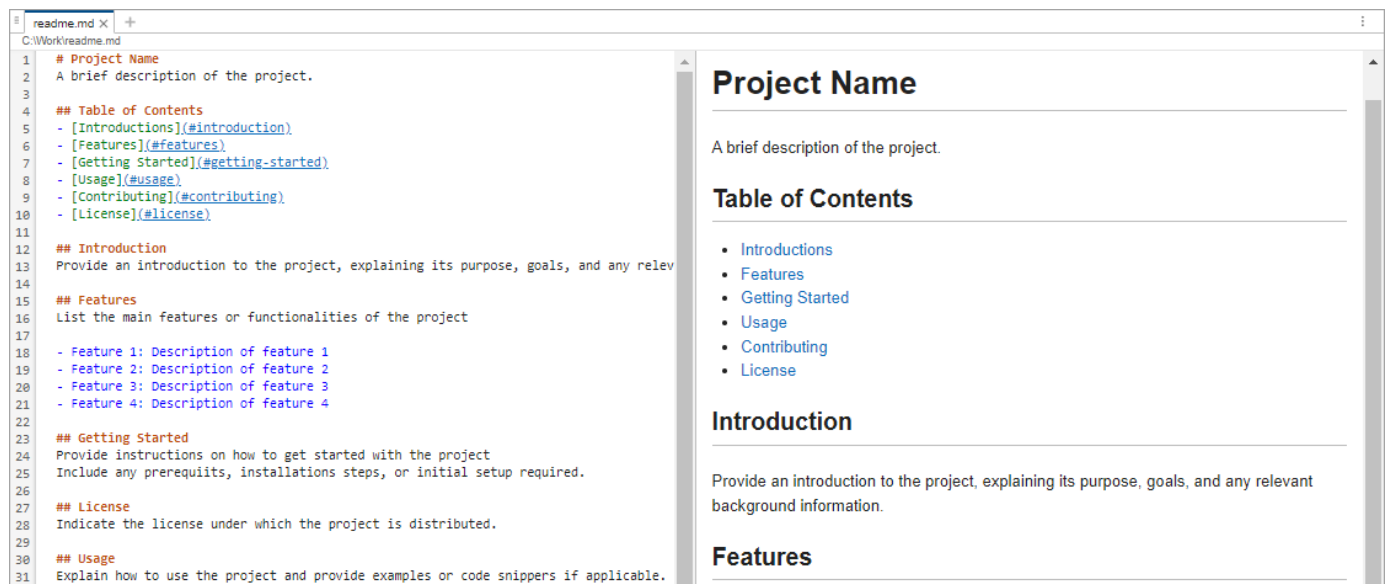
For more information, see Code Suggestions.



As part of this change, the Function browser and the Browse for functions button  have been removed. To find functions, use code suggestions or the search box in the top-right corner of the MATLAB desktop.

## Editor Markdown Support: Preview Markdown files in the Editor

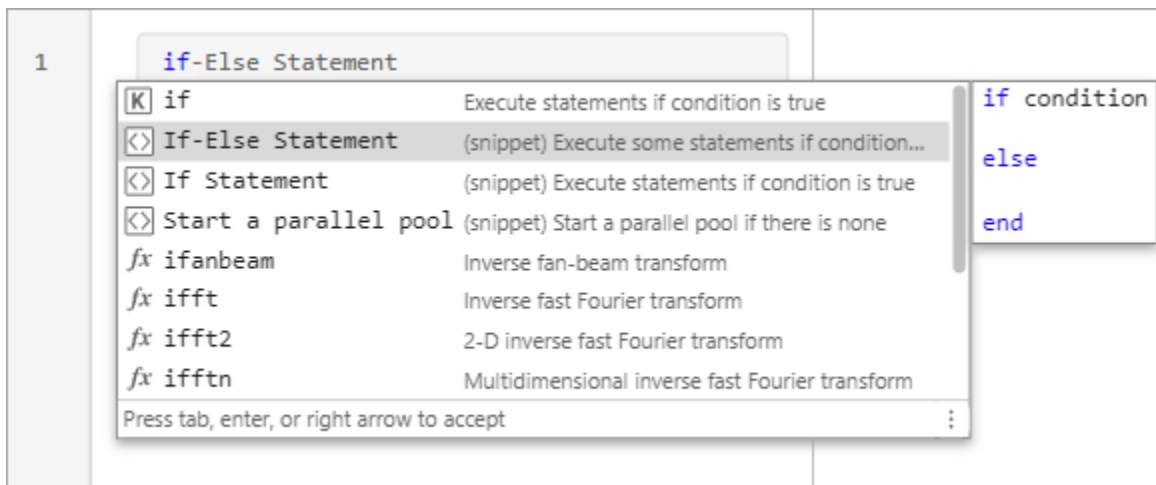
When you open a Markdown file in MATLAB, the Editor shows a preview of the file to the right of the Markdown file. As you make edits to the file, the Editor updates the preview. To close the preview, go to the **View** tab and in the **Preview** section, click the **Markdown** button off.



## Editor Code: Insert code snippets

Code snippets are templates designed to simplify adding recurring code structures, such as loops or conditional statements, to your code. As you type in the Editor or Live Editor, MATLAB shows relevant code snippets along with other name suggestions. For example, if you type the word `if`, MATLAB suggests the `If-Else Statement` and `If Statement` snippets in addition to functions whose name starts with `if`.



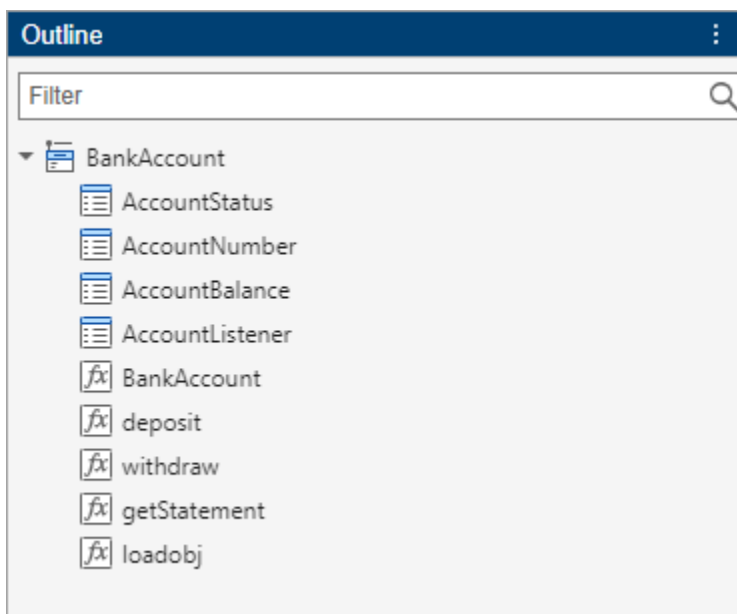


## Editor Code: Copy code line without selection

You can copy a line of code in the Editor, Live Editor, and App Designer without selecting the code. To copy a code line, use the **Ctrl+L** keyboard shortcut. To cut a code line, use the **Ctrl+X** keyboard shortcut. On macOS, use the **Command+C** and **Command+X** keyboard shortcuts instead.

## Outline Panel: View outline of your code using Outline panel

The Outline panel provides a high-level outline of scripts, live scripts, functions, and classes. To view an outline of a file open in the Editor or Live Editor, go to the **View** tab, and in the **Tools** section, click **Outline**. To navigate to an area within your code file, double-click the related entry in the Outline panel.

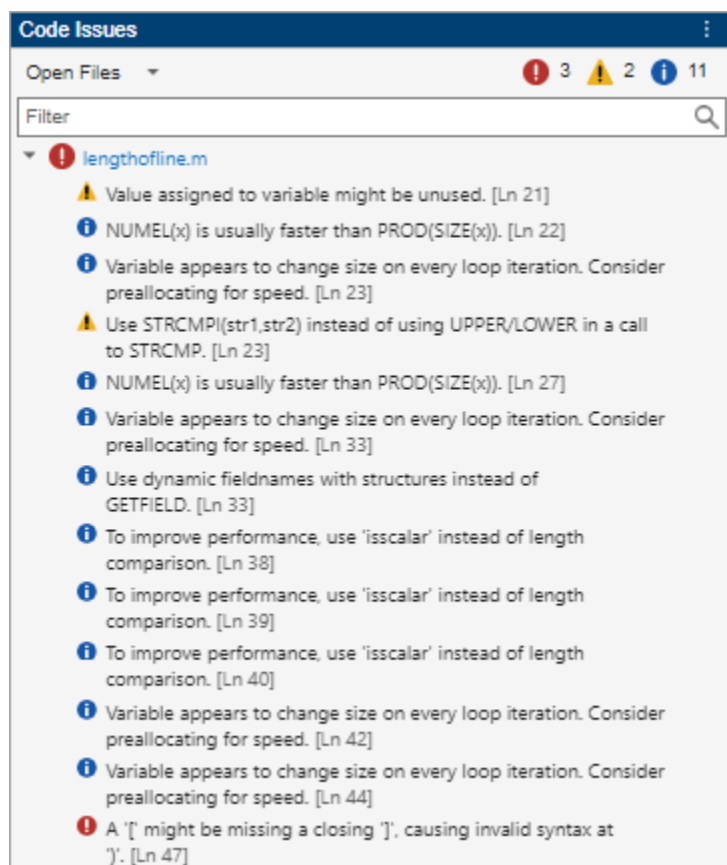


## Code Issues Panel: Check code for errors and warnings using Code Issues panel

You can use the Code Issues panel to view error and warning messages about your code. The Code Issues panel displays potential coding problems that the MATLAB Code Analyzer finds as it automatically checks your code. Using the Code Issues panel, you can choose to view the errors and warnings for the current file or for all open files. You also can filter the list of messages by type (error, warning, or info) as well as by message text.

To open the Code Issues panel, go to the **Editor** or **Live Editor** tab, and in the **Analyze** section, click **Code Issues**. By default, the Code Issues panel opens on the right side of the desktop. To hide the

Code Issues panel, click the Code Issues icon  on the sidebar.



## Find Files Panel: Search for and replace text in filenames and file content using Find Files panel

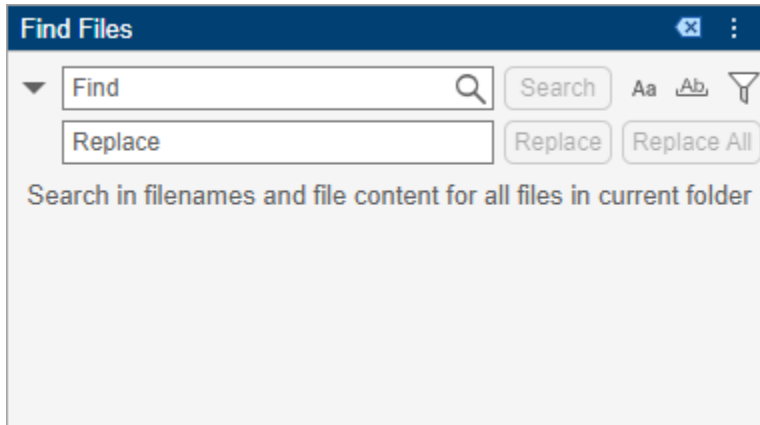
You can use the Find Files panel to search for text in filenames and file content. To search, click the

Find Files icon  on the left sidebar. When searching, you can choose whether to match the case of your search text as well as whether to match the whole word. You also can select what folder to search in and filter results by file extension.

To replace text in filenames and file content, click the expand button  to the left of the search box to open the replace options. Replacing text is supported in text files (.txt) and MATLAB plain text code files (.m).

To open the Find Files panel in a separate window, click the Find Files actions button  at the top of the Find Files panel and select **Undock Find Files**.

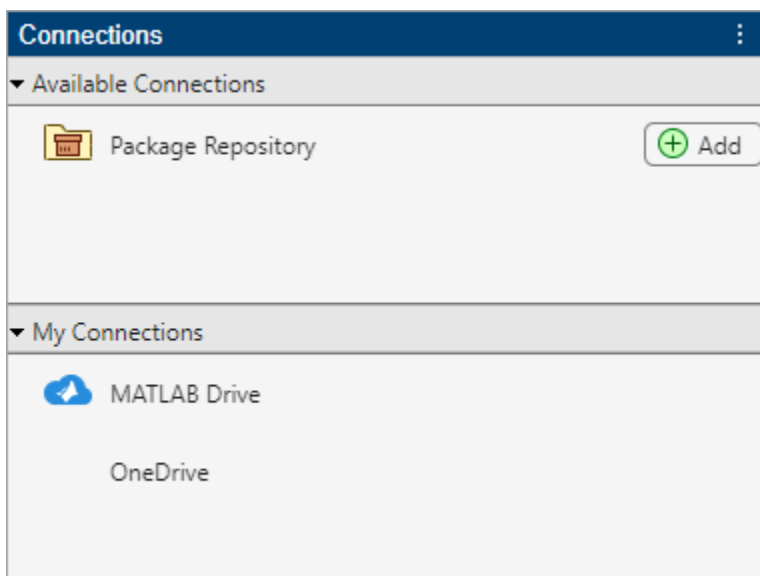
For more information, see Find Files.



As part of this change, you can no longer exclude files from the search based on their file type or size.

## Connections Panel: View and manage your connections using Connections panel

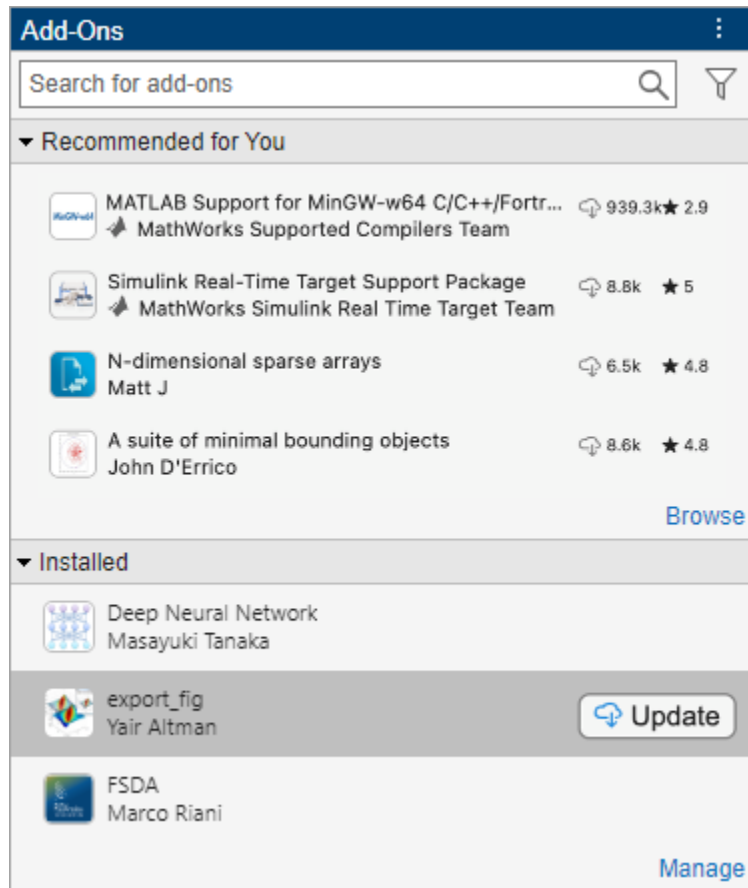
Use the Connections panel to view and manage your available connections, including OneDrive™ and MATLAB Drive™. To open the Connections panel, click the Open more panels button \*\*\* on any sidebar and select the Connections panel.



## Add-Ons Panel: Install and manage add-ons using Add-Ons panel

You can use the Add-Ons panel to find, install, and manage add-ons. You also can update an installed add-on, such as a support package, directly from the Add-Ons panel. To open the Add-Ons panel, click the Add-Ons icon  on the left sidebar. If the Add-Ons icon is not on the sidebar, click the Open more panels button  and select the Add-Ons panel.

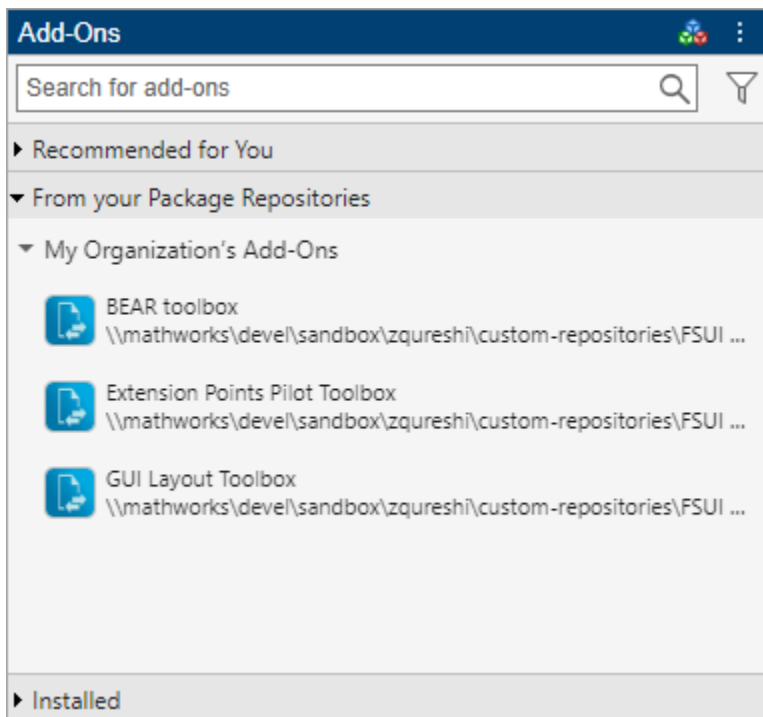
For more information, see [Get and Manage Add-Ons](#).



## Add-Ons Panel: View and install add-ons from MATLAB package repositories

You can view and install add-ons from MATLAB package repositories directly from the Add-Ons panel. To view and install the add-ons, click the Add-Ons actions button  at the top of the Add-Ons panel and select **Manage Package Repositories** to open the Connections panel. In the Connections panel, add a connection to the MATLAB package repository. Then in the **My Connections** section, click the  button to the right of the package repository connection and select **Browse for Repository Add-Ons** to return to the Add-Ons panel. In the Add-Ons panel, go to the **From your Package Repositories** section and select an add-on to install. You also can search for an add-on using the search box at the top of the panel.

For more information, see [Install Add-Ons from MATLAB Package Repositories](#).



## matlabProcessID Function: Get process identifier for MATLAB code execution process

Call the `matlabProcessID` function to get the process identifier for an active MATLAB process.

## Window Managements in MATLAB Online: Use multiple browser tabs for MATLAB Online

In MATLAB Online™, you can now open some files, apps, and tools in new browser tabs.

Using multiple browser tabs for MATLAB Online lets you:

- Display MATLAB Online content on multiple monitors.
- Seamlessly switch between multiple full-size MATLAB Online windows.

For more information, see [Manage Windows in MATLAB Online](#).

## Help Center: Get AI-generated response when searching the web documentation

When you search the web documentation, Help Center displays an AI-generated response along with the search results. The response includes:

- A concise answer to your query
- A list of sources used to generate the answer
- A list of related searches for further exploration

AI-generated responses are based on relevant MathWorks® documentation pages.

## Folder Comparison: Compare folders and their subfolders recursively

When you select folders to compare, the Comparison Tool now includes the contents of all subfolders recursively in the comparison. To include only the folders you select and the first-level subfolders in the comparison report, in MATLAB, set the `IncludeSubfolders` comparison setting to false.

```
s = settings;  
s.comparisons.folder.IncludeSubfolders.TemporaryValue = false;
```

For more information, see [Compare Files and Folders](#) and [Merge Files](#).

## matlab.crashhandling.crashReportFolder Function: Locate crash report folder

The new `matlab.crashhandling.crashReportFolder` function returns a path to the MATLAB crash report folder. Some crash files might be stored in other locations. For more information, see “When MATLAB Terminates Unexpectedly”.

## ⚠️ Functionality being removed or changed

### matlab function -nojvm startup option supports desktop tools and graphics

*Behavior change*

If you start MATLAB using the `matlab` function with the `-nojvm` startup option, desktop tools and graphics appear as expected. In previous releases, desktop tools and graphics were not supported when MATLAB was started using the `-nojvm` startup option.

To prevent desktop tools and graphics from appearing, use the `-nodisplay` or `-nodesktop` options in addition to the `-nojvm` option. For example, this command starts MATLAB without the JVM software and without the desktop.

```
matlab -nojvm -nodisplay
```

### matlab function no longer supports -nosplash startup option

*Behavior change*

The `matlab` function `-nosplash` startup option is no longer supported. If you specify the `-nosplash` option at startup, MATLAB ignores the option. Previously, you could use the `-nosplash` startup option to suppress the display of the splash screen.

### Use monospaced font when running MATLAB using -nodesktop startup option

*Behavior change*

To ensure that output displays as intended when running MATLAB using the `-nodesktop` startup option on macOS or Linux®, make sure that your macOS Terminal or Linux system prompt uses a monospaced font.

### publish function no longer supports "entireFigureWindow" value for figureSnapMethod name-value argument

*Errors*

For the `publish` function, specifying the `figureSnapMethod` name-value argument as "entireFigureWindow" is no longer supported. Use the "entireGUIWindow", "print", or "getframe" values instead.

### **publish function no longer supports "bmp" value for imageFormat name-value argument with "pdf" format or "print" figureSnapMethod name-value argument**

#### *Errors*

For the `publish` function, specifying the `imageFormat` name-value argument as "bmp" is no longer supported when the value of the `format` name-value argument is "pdf". With this change, you can instead specify `imageFormat` as either "png" or "jpg" for PDFs, with "png" being the default value. Specifying the `imageFormat` name-value argument as "bmp" is also no longer supported when the value of the `figureSnapMethod` name-value argument is "print".

### **Figures appear larger and as tabs by default in published documents**

#### *Behavior change*

By default, figures created when publishing a document appear larger than in previous releases. In addition, these figures appear as tabs in a figure container.

To publish documents using the previous default figure size, set the `Position` property of each figure using a position vector of the form [left bottom width height]. Setting the `Position` property undocks the figure from the figure container.

```
f = figure(Position=[100 100 560 420]);
```

To use the previous default initial figure size for all figures, set the default figure position of the graphics root object.

```
set(groot,DefaultFigurePosition=[100 100 560 420])
```

For more information, see [Figure](#).

### **Customizing quick access toolbar and Current Folder toolbar using Settings window is no longer supported**

#### *Behavior change*

Customizing the quick access toolbar and Current Folder toolbar using the Settings window is no longer supported. In addition, moving the quick access toolbar is no longer supported. Previously, you could rearrange, add, and remove controls from both toolbars using the Settings window, and you could move the quick access toolbar below the MATLAB Toolstrip.

To rearrange controls in the quick access toolbar, drag them to the desired locations instead. To add a toolstrip control to the quick access toolbar, right-click the control and select **Add to Quick Access Toolbar**. You also can select from a list of common controls by going to the quick access toolbar and clicking the Common Controls button . To remove a control from the quick access toolbar, right-click the control and select **Remove from Quick Access Toolbar**.

There is no alternative way to customize the Current Folder toolbar.

### **Selecting and comparing files and folders from the Home tab has been removed**

#### *Behavior change*

The **Compare** button on the **Home** tab that opens the Select Files or Folders for Comparison dialog box has been removed. To select files or folders on disk for comparison, use one of these methods instead:

- MATLAB editors — Open a file in the MATLAB Editor or Live Editor. On the **Editor** or **Live Editor** tab, select **Compare > Compare To**. Then, select the file you want to compare to.

You can also compare a file with the autosaved version or the saved version on disk. In the **Editor** tab, select **Compare > Compare to Version on Disk** or **Compare to Backup**. If you modified your file, the Editor saves the file before comparing. The **Compare to Version on Disk** and **Compare to Backup** options are available only if the option for automatically saving changes to a file is disabled. To disable the option, go to the **Home** tab, and in the **Environment** section, click **Settings**. Then, select **Editor/Debugger > Saving** and clear **Save changes upon clicking away from a file**. This option is not available in the Live Editor.

- Files or Project panel — To compare two files on disk, select both files. Then, right-click and select **Compare Selected Files/Folders**. The Comparison Tool opens the file you select first on the left.

Alternatively, right-click a file and select **Compare To**. Then, select the file you want to compare to.

- Command Window — Use the `visdiff` function.

For more information, see [Compare Files and Folders](#) and [Merge Files](#).

### **XML comparison report has been removed**

#### *Behavior change*

When you compare two XML files, the Comparison Tool opens a text comparison report. For more information, see [Compare Text Files](#).

Functions related to XML comparison, such as `xmlcomp.compare`, `xmlcomp.zipTempFiles`, and `xmlcomp.showLogFile`, will be removed in a future release. Starting in R2025a, these functions issue a warning.

### **XML comparison type for visdiff function has been removed**

#### *Errors*

The XML comparison type for the `visdiff` function has been removed. Overriding the default comparison type by specifying "xml" is no longer supported. Starting in R2025a, the syntax `visdiff(filename1,filename2,"xml")` issues an error.

### **Command History saved to settings instead of history file**

#### *Behavior change*

MATLAB now saves the Command History to settings. Previously, MATLAB saved the Command History to the history file `History.xml` in the settings folder. For more information, see [Command History](#).

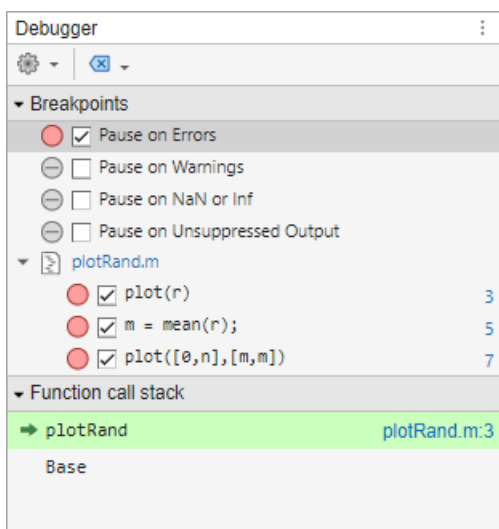


## Language and Programming

### ★ Debugging: Manage breakpoints and navigate function call stack using Debugger panel

Use the Debugger panel to manage breakpoints and navigate the function call stack while debugging. To open the Debugger panel, go to the **Editor** or **Live Editor** tab, and in the **Analyze** section, click **Debugger**. You also can open the panel using the Open more panels button \*\*\* on the sidebar.

For more information, see [Debug MATLAB Code Files](#).



### ⚠ MATLAB Identifiers: Maximum length of MATLAB identifiers increased to 2048 characters

The maximum allowed identifier length has increased to 2048 characters. This change allows variables, functions, classes, and many other entities to have names of up to 2048 characters. You can query the maximum identifier length by using the `namelengthmax` function.

Using longer identifiers might negatively impact performance and memory. Operating systems might have maximum filename lengths that are shorter than this limit.

### ⚠ Compatibility Considerations

Previously, the maximum length of identifiers was 63 characters. Code that depends on identifiers being truncated to 63 characters might no longer run as expected.

### cell2mat function accepts larger set of data types

Before R2025a, `cell2mat` accepted numeric, logical, and character types, as well as structures with the same field names. All cells of the input array had to have the same data type.

`cell2mat` can now accept cell arrays containing additional data types, including strings, categoricals, `gpuArray` objects, and class-based objects.

The function can also convert a cell array that contains different data types if MATLAB can convert those data types to one output type. For examples of basic data type combinations and their resulting types, see [Valid Combinations of Unlike Classes](#).

If the cell array contains class-based objects of different classes, MATLAB attempts to concatenate them. For more information, see [Concatenating Objects of Different Classes](#).

## **onCleanup Object: Cancel cleanup tasks**

The `cancel` object function cancels any cleanup tasks specified by the `onCleanup` object and prevents them from being executed when the object is destroyed.

## **matlab.lang.Workspace Object: Store workspace variables**

Store variables within a workspace as a `matlab.lang.Workspace` object. You can pass this object and access it from other workspaces in the same way as any other variable. To create a workspace object, use one of these functions:

- `matlab.lang.Workspace.baseWorkspace` — store variables from the base workspace. The base workspace contains variables that you create at the command line. The base workspace includes any variables that scripts create, if you run the script from the command line or from the Editor.
- `matlab.lang.Workspace.currentWorkspace` — store variables from the current workspace. For instance, call `matlab.lang.Workspace.currentWorkspace` inside a function to save variables in the function workspace.
- `matlab.lang.Workspace.globalWorkspace` — store global variables.

After you create a workspace object, you can use these object functions:

- `variables` — Access information about workspace variables stored in a workspace object.
- `variableNames` — Return the names of variables stored in a workspace object.
- `evaluateAndCapture` — Evaluate code in the workspace represented by a workspace object.

## **secretID Object: Keep sensitive information encrypted when setting object properties**

You can avoid saving credentials and other sensitive information as plain text in properties of objects such as `weboptions` by using a `secretID` object. First, save sensitive information to your MATLAB vault as a secret using `setSecret`. Then when you set an object property, such as the `Password` property of a `weboptions` object, use a `secretID` object to keep your sensitive information encrypted.

For example, if you save your password as a secret named "myPass", use a `secretID` object to set a property to your associated secret.

```
opts = weboptions;  
opts.Password = secretID("myPass");
```

## ▲ Functionality being removed or changed

### Defining classes: Using function syntax to define classes will not be supported in a future release

*Still runs*

Support for defining classes using `function` syntax will be removed in a future release. With appropriate code changes, replace existing function-based classes with classes defined using the `classdef` keyword.

### Defining classes: Using `schema.m` will not be supported in a future release

*Warns*

Support for classes defined using `schema.m` files will be removed in a future release. Replace existing schema-based classes with classes defined using the `classdef` keyword.

### Change in precedence for functions and classes in @-folders with the same name

*Warns*

In a future release, the rule giving precedence to a class defined in a class folder (@-folder) over a function of the same name, regardless of position on the path, will be removed. MATLAB now issues warnings when these three conditions are met:

- You have a class defined in a class folder and a function with the same name on your path.
- The function appears earlier in the path order than the class.
- You use the name of the function and class.

For more information, see [Precedence of Classes in @-folders over Functions of the Same Name Will Be Removed in a Future Release](#).

## Data Analysis

### ★ Variables Editor: View and analyze variable contents using improved interface

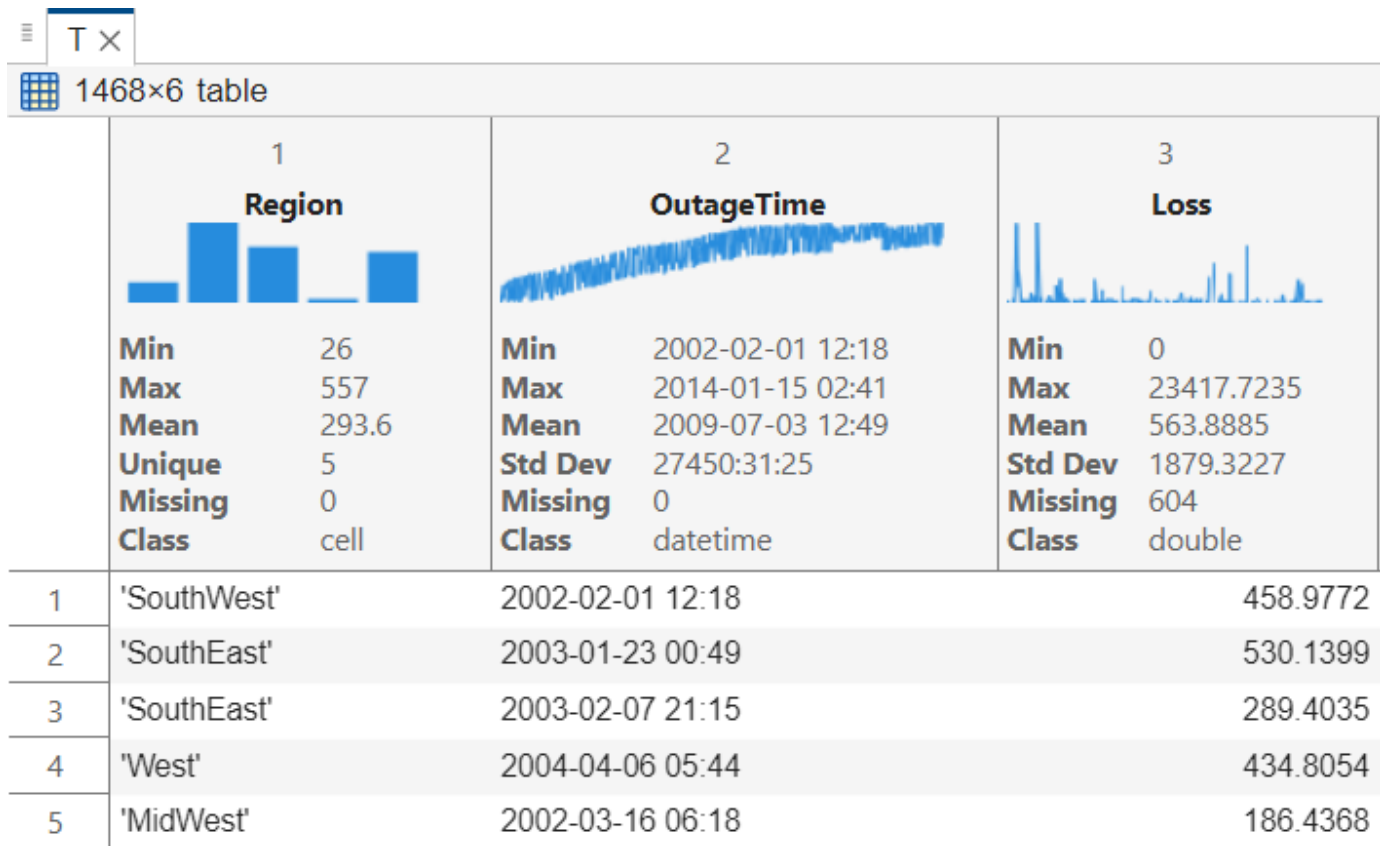
The Variables editor has new tools for navigating and analyzing variable contents. For example, you can:

- View edits to variables of all types as MATLAB code in the Command Window.
- View sparklines and summary statistics for data in a table or timetable.
- Create a logical variable from selected data indices.
- Navigate variable contents using the **Find** and **Go To** buttons on the toolbar.
- Create a linked plot that is synchronized with workspace variables.
- Edit categories of a categorical table variable.

The Variables editor display has enhanced readability for some variable types. For example, you can:

- Review and analyze data in a table, timetable, structure array, or cell array using a simplified, borderless design and row striping.
- Edit the contents of and view useful information for data in a scalar structure, object array, or cell array.
- Interactively edit each page of data in a multidimensional array.
- View the event labels for data in a timetable.

You can open the Variables editor by double-clicking a variable in the Workspace panel or using the `openvar` function.



### Figure Container: Organize, explore, customize, and generate code for plotted data using an improved figure window

Figures now open as tabs in a docked figure container in the MATLAB desktop by default. You can use the figure container to organize, explore, customize, and generate code for plotted data more efficiently:

- Quickly customize and annotate figures using the toolstrip of the figure container.
- Automatically generate code to reproduce figure interactions and customizations.
- View figures more easily with their larger default size and wider aspect ratio.
- Customize the layout of open figures by rearranging their tabs, or view multiple figures simultaneously by arranging them in tiles.

For more information, see [Interactively Explore and Analyze Plotted Data](#).

For compatibility considerations, see “Figure Container: Figures appear larger and as tabs by default” on page 2-38.

### allbetween and mustBeBetween Functions: Determine if or validate that all elements are within specified range

You can check that elements in input data are within a specified range by using these functions:

- `allbetween` — Return logical 1 (`true`) if all elements are within the specified range and return logical 0 (`false`) otherwise.
- `mustBeBetween` — Validate that all elements are within the specified range and throw an error if any element is outside of the range.

For both functions, you can specify whether the upper and lower bounds are inclusive or exclusive. Additionally, for an input table or timetable, you can specify which variables to operate on using the `DataVariables` name-value argument.

## **numunique and allunique Functions: Count and verify unique values**

You can count the number of unique values in input data and verify the uniqueness of all values in input data by using these functions:

- `numunique` — Return the number of unique values.
- `allunique` — Return logical 1 (`true`) if all values are unique and return logical 0 (`false`) otherwise.

## **unique Function: Specify set order and occurrence arguments together**

For the `unique` function, you can specify in the same function call both the order of values in the returned set and whether the first or last index of a repeated value is returned. For example, in the function call `[C,ia,ic] = unique(A,"stable","last")`:

- `C` contains the unique elements, ordered as in the input data `A`.
- `ia` contains the indices of the last occurrence of these elements in `A`.
- `ic` is a mapping from elements in `C` to elements in `A`.

Previously, specifying both the set order and occurrence arguments was not supported.

## **⚠table and timetable Data Types: Maximum length of dimension and variable names increased to 2048 characters**

In tables and timetables, the maximum allowed length of dimension and variable names has increased to 2048 characters. This length matches the updated maximum identifier length, which you can query by using the `namelengthmax` function.

Using longer dimension and variable names in tables and timetables might negatively impact performance and memory.

## **⚠Compatibility Considerations**

Previously, the maximum length of dimension and variable names in tables and timetables was 63 characters. Code that depends on dimension and variable names being no longer than 63 characters might not run as expected.

If you save a table or timetable with long dimension or variable names to a MAT file, then you can load it into a MATLAB release before R2025a. However, the earlier MATLAB release truncates the long dimension and variable names in the table or timetable to 63 characters.

## Moving Statistics Functions: Calculate moving statistics for data in tables and timetables

You can calculate moving statistics for numeric data in tables and timetables. By default, moving statistics functions calculate statistics for all variables in a table or timetable. However, you can specify the variables to use for these calculations. Also, you can append results as new variables in the output.

These functions support moving statistics for table and timetable data:

- `movmad`
- `movmax`
- `movmean`
- `movmedian`
- `movmin`
- `movprod`
- `movstd`
- `movsum`
- `movvar`

## prctile and quantile Functions: Calculate percentiles and quantiles using inclusive and exclusive methods

You can calculate percentiles and quantiles using an inclusive or exclusive method with the `prctile` and `quantile` functions. Specify the method name as "inclusive" or "exclusive", respectively. The "inclusive" method includes the 0th and 100th percentiles (or 0th and 1st quantiles) within the bounds of the data; the "exclusive" method excludes them.

Additionally, the name of the default method has changed from "exact" to "midpoint". The `prctile` and `quantile` functions continue to support `Method="exact"` for backward compatibility.

These examples illustrate the differences between the default method and the two new methods.

|                                                                                                       |                                                                                                   |                                                                                                        |
|-------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------|
| <code>x = 1:5;</code><br><code>P = prctile(x,25,Method="midpoint");</code><br><code>P = 1.7500</code> | <code>x = 1:5;</code><br><code>P = prctile(x,25,Method="inclusive");</code><br><code>P = 2</code> | <code>x = 1:5;</code><br><code>P = prctile(x,25,Method="exclusive");</code><br><code>P = 1.5000</code> |
|-------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------|

## pivot Function and Pivot Table Live Editor Task: Rearrange data without aggregation

You can rearrange data that does not require aggregation, including nonnumeric data, by specifying the `Method` name-value argument of the `pivot` function as "none". When using the Pivot Table Live Editor task, select the No aggregation computation method in the **Values** pane of the **Select pivot variables** section.

For an example, see [Rearrange Data Without Aggregation](#).

## **Functionality being removed or changed**

### **Specify page orientation when printing variable contents by using MATLAB Print dialog box**

#### *Behavior change*

To specify the page orientation when printing the contents of a variable, use the options in the MATLAB Print dialog box. Page numbers are no longer printed in the page header, and all rows print before wrapped columns.

Previously, the Page Setup dialog box controlled the page orientation.

### **Plot Catalog tool has been removed**

#### *Behavior change*

The Plot Catalog tool has been removed. To interactively create and explore visualizations for your data, use the **Plots** tab of the MATLAB Toolstrip or the Create Plot task in the Live Editor.

For more information about visualizations, see [Types of MATLAB Plots](#) or toolbox-specific documentation.

### **Variables editor displays properties of timeseries objects**

#### *Behavior change*

When you open a `timeseries` object in the Variables editor, the editor now displays the Property, Value, Size, and Class columns directly, rather than displaying a specialized editor for the `timeseries` data. You can still access and view all the properties of the `timeseries` object and edit some of its property values using this updated display.

### **Variables editor no longer supports opening a variable that does not exist**

#### *Errors*

The Variables editor no longer supports opening a variable that does not exist. For example, if the variable `a` does not exist, then `openvar a` returns an error.



## Data Import and Export

### Reading Compressed Data: Read data from compressed and archived files

You can read data from compressed and archived files as a table, cell array, matrix, struct, timetable, dictionary, string array, or variable. Specify ZIP, GZ, and TAR files when using these functions:

- `readtable`
- `readcell`
- `readmatrix`
- `readstruct`
- `readtimetable`
- `readdictionary`
- `readlines`
- `readvars`

### File Permissions: View and adjust file permissions

You can view the permissions of files, folders, and symbolic links using the `filePermissions` function. The data type of the returned permissions information depends on the system where the file, folder, or symbolic link is located. You can then get or set individual permissions by using the `getPermissions` and `setPermissions` functions, respectively.

### Datastores: Specify multiple remote files using wildcard characters

You can create a datastore from multiple remote files. Specify multiple remote files using wildcard characters anywhere in the file path when creating `tabularTextDatastore`, `spreadsheetDatastore`, `imageDatastore`, `parquetDatastore`, and `fileDatastore` objects.

### FTP and SFTP: Specify folders in server hostname

When connecting to an FTP or SFTP server using the `ftp` or `sftp` function, you can specify folder paths in the server hostname. For example, connect to a sample FTP server.

```
ftpobj = ftp("ftp.example.com/myfolder");
```

### xslt Function: Specify XSLT processing engine for transforming an XML file

When transforming an XML file using the `xslt` function, you can specify the XSLT processing engine as either the MATLAB API for XML Processing (MAXP) or Java API for XML Processing (JAXP). In the `xslt(__,XMLEngine=xmlengine)` syntax, specify `xmlengine` as `"maxp"` or `"jaxp"`, respectively.

### HDF5 Interface: Read and write cloud-optimized HDF5 files

You can create HDF5 files that are optimized for the cloud by specifying file-space handling strategies, file-space page sizes, and page buffer sizes. A cloud-optimized HDF5 file has an internal

structure that is arranged for more efficient data access when the file is in cloud object stores. Use these functions in the low-level HDF5 interface:

- `H5P.set_file_space_strategy`
- `H5P.set_file_space_page_size`
- `H5P.set_page_buffer_size`

You can retrieve the file-space handling strategy, file-space page size, page buffer size, and minimum percentage for metadata and raw data pages for an HDF5 file. Use these functions in the low-level HDF5 interface:

- `H5P.get_file_space_strategy`
- `H5P.get_file_space_page_size`
- `H5P.get_page_buffer_size`

To support cloud-optimized HDF5 files, the `H5P.set_chunk_cache` function accepts new values for three of its input arguments:

- Specify `nslots` as `"H5D_CHUNK_CACHE_NSLOTS_DEFAULT"` to use the number of elements specified in the file access property list to open the file instead of setting the property on the dataset.
- Specify `nbytes` as `"H5D_CHUNK_CACHE_NBYTES_DEFAULT"` to use the number of bytes specified in the file access property list to open the file instead of setting the property on the dataset.
- Specify `w0` as `"H5D_CHUNK_CACHE_W0_DEFAULT"` to use the preemption policy value specified in the file access property list to open the file instead of setting the property on the dataset.

## Import Tool: Import data with timestamps as timetable

You can import timestamped data from a file as a timetable using the Import Tool. If the tool detects timestamps in the first variable of the data to import, the tool uses the `timetable` data type by default.

## Image Files: Read and display HEIF and HEIC image files

You can now read HEIF and HEIC image files using the `imread` function. You can also use `imfinfo` to get information about HEIF and HEIC image files and `imshow` to display the HEIF and HEIC images.

This functionality requires MATLAB Support for HEIF/HEIC Image Format, which is available only on MATLAB desktop environment.

To install the support package:

- On the MATLAB **Home** tab, in the **Environment** section, select **Add-Ons > Explore Add-Ons**.
- In the Add-On Explorer search bar, search for MATLAB Support for HEIF/HEIC Image Format, and then click the MATLAB Support for HEIF/HEIC Image Format add-on.

## Functionality being removed or changed

**`fopen` function no longer supports `fopen("all")` syntax**  
*Errors*

The `fopen` function returns an error if you use the `fopen("all")` syntax. To get the file identifiers of all open files, use the `openedFiles` function instead.

### **web function opens local pages and files in HTML Viewer instead of MATLAB web browser**

#### *Behavior change*

When you use the `web` function to open a local URL or file, the page now opens in the HTML Viewer. Previously, the page opened in the MATLAB web browser. As part of this change, the MATLAB web browser has been removed.

### **web function returns handle to most recent MATLAB web browser as `matlab.htmlviewer.HTMLViewer` class**

#### *Behavior change*

The `web` function now returns a handle to the most recent MATLAB web browser as a `matlab.htmlviewer.HTMLViewer` object. In previous releases, the `web` function returns the handle as an instance of a Java<sup>®</sup> class. With this change, some methods that were previously supported in the returned handle are no longer supported.

In most cases, you do not need to make any changes to your code. However, if your code uses methods that are not supported in the returned `matlab.htmlviewer.HTMLViewer` class, you must update your code.

### **Import Tool no longer supports dragging a file to import**

#### *Behavior change*

You can no longer drag a file into the Import Tool to import its data. To import data from a file using the Import Tool, right-click the file in the **Files** panel and select **Import Data**.

### **Import Tool imports text as string array**

#### *Behavior change*

The Import Tool imports all text as string arrays, except for in cell arrays. Previously, you could choose to import text as either a string array or a cell array of character vectors.

## Mathematics

### ▲ Sparse Matrix: Create and use single-precision sparse matrices

Single-precision sparse matrices are now supported in all functions that support double-precision sparse matrices, including elementary math, linear algebra, and sparse matrix functions. You can create a single-precision sparse matrix using these functions:

- `sparse`
- `speye`
- `spalloc`
- `sprand`
- `sprandn`
- `sprandsym`

By default, these functions create a double-precision sparse matrix. To create single-precision sparse data, specify the new `typename` argument as `"single"`. For example, create a 50-by-100 single-precision sparse matrix with density 0.1 by calling `sprand(50,100,0.1,"single")`.

You can also create single-precision sparse matrices by providing single-precision input data to `sparse`. For example, create a 4-by-4 single-precision sparse matrix by calling `sparse(1:4,1:4,single([3 4 5 6]))`.

### ▲ Compatibility Considerations

If you save a single-precision sparse matrix to a MAT file, you can load it into an earlier release of MATLAB. However, how the data is loaded depends on the release of MATLAB:

- MATLAB R2021b through R2024b — MATLAB warns that the data has an unsupported data type and loads it as a double-precision sparse matrix.
- MATLAB R2021a and earlier — MATLAB loads the data as a single-precision sparse matrix, but you cannot use the matrix because its data type is unsupported. Attempting to use the matrix can result in undefined behavior.

### ode Object: Solve delay differential equations

You can solve delay differential equations (DDEs) using an ode object by specifying the `Solver` property as `"dde23"`, `"ddesd"`, or `"ddensd"` and specifying the `DelayDefinition` property as an `odeDelay` object. If you specify only `DelayDefinition` and not `Solver`, the ode object automatically selects a solver.

### ode Object: Solve complex-valued functions

You can solve ordinary differential equations (ODEs) with complex solution values using an ode object by specifying the `SeparateComplexParts` property as `"on"`. Specify `SeparateComplexParts` as `"on"` when the ODE is complex and the selected solver fails or takes a long time to converge. This is especially helpful for solving implicit and stiff ODEs where the Jacobian is not specified.

## Solve ODE Live Editor Task: Detect stiffness and specify minimum step size

In the Solve ODE task, you can enable stiffness detection for problems where the task automatically selects the solver. In the **Specify solver options** section, select **Detect stiffness** when **Solver** is **Automatic**.

You also can specify a minimum step size when the selected solver includes the **MinStep** option. In the **Specify solver options** section, enter a value in the **Min step** box. For a summary of the available options for each solver, see the **SolverOptions** property of the ode object.

## pdepe Function: Specify output function and other odeset options

You can now specify all **odeset** options, including the output function, when using the **pdepe** function. However, **pdepe** overrides values for the **Mass**, **MStateDependence**, **MvPattern**, **MassSingular**, **Jacobian**, **JPattern**, and **Vectorized** options when defining the PDE to solve.

## pagelsqminnorm Function: Apply Tikhonov regularization to page-wise least-squares solution

You can apply Tikhonov regularization to the page-wise least-squares solution of the **pagelsqminnorm** function by specifying the regularization factor **alpha** as a real number or a multidimensional array. Specify **alpha** to return a solution **X** that minimizes  $\text{pagenorm}(A.*X-B).^2 + \text{alpha}.^2.*\text{pagenorm}(X).^2$  for each column of each page of **X**, given coefficient and input arrays **A** and **B**.

## ▲ Functionality being removed or changed

### Two-input syntax of diff function calculates differences along only the first dimension whose size is greater than 1

*Behavior change*

The two-input syntax **diff(X,n)** of the **diff** function now operates along only the first dimension of **X** whose size is greater than 1 for all values of **n**. If **n** is larger than that dimension, **diff** returns an empty array. For example, this code now returns a 0-by-3 empty double matrix.

```
X = [1 1 1; 5 5 5];
Y = diff(X,3);
```

Previously, when **n** was larger than the first dimension of **X** whose size is greater than 1, **diff(X,n)** reduced the size of that dimension to 1 and then calculated differences along the next dimension whose size is greater than 1. In previous releases, this code returned 0.

### Three- and four-input syntaxes of spdiags function return output with data type that is more consistent with inputs

*Behavior change*

The output of the three-input syntax and the four-input syntax of the **spdiags** function is more consistent with the inputs, as shown in this table.

| Syntax                              | Output Type in R2024b and Earlier                                                                                                | Output Type Starting in R2025a |
|-------------------------------------|----------------------------------------------------------------------------------------------------------------------------------|--------------------------------|
| <code>S = spdiags(Bin,d,A)</code>   | <ul style="list-style-type: none"><li>• sparse logical if either Bin or A is logical</li><li>• sparse double otherwise</li></ul> | Same as A                      |
| <code>S = spdiags(Bin,d,m,n)</code> | <ul style="list-style-type: none"><li>• sparse logical if Bin is logical</li><li>• sparse double if Bin is not logical</li></ul> | Same as Bin                    |

For example, this code now returns a 5-by-5 sparse single matrix.

```
S = spdiags(single([-1 1]),[-1 1],5,5);
```

In previous releases, this code returned a 5-by-5 sparse double matrix.

**colon operator no longer supports nonscalar operands**

*Errors*

`colon` now returns an error when creating vectors if one or more operands are not scalar. For example, expressions like `[1 2 3]:2:10` now error. Previously, `colon` used the first element of any nonscalar operands to evaluate the expression.

To preserve the behavior of previous releases, update your code to specify the first element of any nonscalar operands in place of those nonscalar operands. For example, this code now returns an error because `size(A)` is nonscalar.

```
A = randn(3,4);
v = 1:size(A);
```

In previous releases, this code was equivalent to `v = 1:3` because `size(A)` returned the vector `[3 4]` and `colon` used only the size of the first dimension of `A` to evaluate the expression. To achieve the same result, explicitly specify the size of the first dimension of `A` as the operand.

```
v = 1:size(A,1);
```

As another example, this code also returns an error because `id` is nonscalar.

```
id = find([1 2 3 1 2 3] == 3);
v = 1:id;
```

In previous releases, this code was equivalent to `v = 1:3` because `find` returned the vector `[3 6]` and `colon` used only the first element of this returned vector to evaluate the expression. To achieve the same result, explicitly specify the first element of the vector as the operand.

```
v = 1:id(1);
```

Alternatively, you can directly specify that `find` return only the first index.

```
id = find([1 2 3 1 2 3] == 3,1,"first");
v = 1:id;
```

If you have a vector of sorted values, you can also use the `min` function to access the first element of the vector. For example, this code now returns an error because `strfind` returns a vector.

```
path = 'matlab/users/username/examples';  
topDirname = path(1:strfind(path, '/')-1);
```

To achieve the same result as in previous releases, use the `min` function on the sorted vector returned by `strfind`.

```
topDirname = path(1:min(strfind(path, '/'))-1);
```

**pow2 and log2 functions will no longer accept complex inputs when you specify two inputs or two outputs**

*Warns*

The `pow2` and `log2` functions issue a warning if you specify complex inputs when using the two-input or two-output syntax. Support for complex inputs in these syntaxes will be removed in a future release. In previous releases, the two-input syntax of `pow2` and the two-output syntax of `log2` ignored the imaginary parts of complex inputs and processed only the real parts.

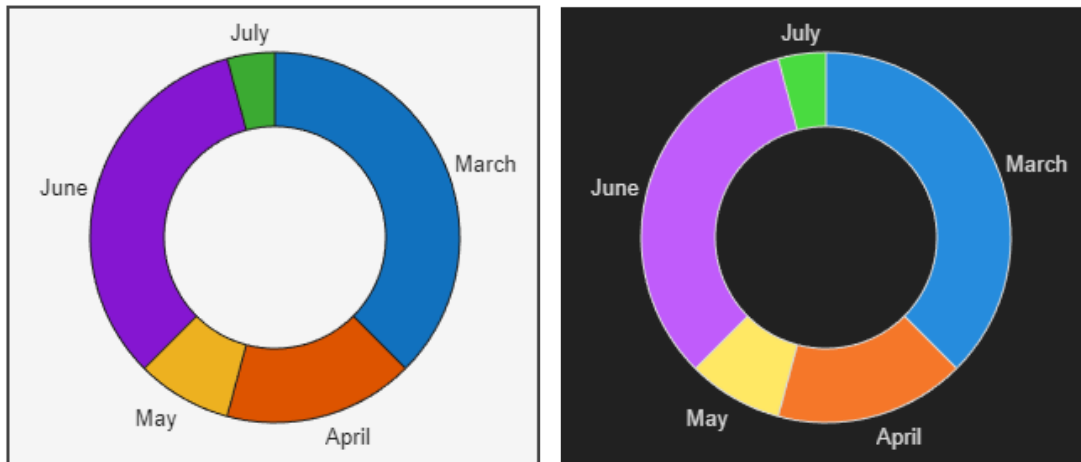
To preserve the behavior of previous releases, use the `real` function to extract the real parts of complex inputs, as shown in this table.

| Not Recommended (Warns)                  | Recommended                                          |
|------------------------------------------|------------------------------------------------------|
| X = 2 + 1i;<br>E = 2i;<br>Y = pow2(X,E); | X = 2 + 1i;<br>E = 2i;<br>Y = pow2(real(X),real(E)); |
| X = 2 - 1i;<br>[F,E] = log2(X);          | X = 2 - 1i;<br>[F,E] = log2(real(X));                |

## Graphics

### ★ Themes: Create graphics using light or dark theme

You can view and create figures in light or dark theme. When you create plots, MATLAB automatically selects colors that are visually appealing for the theme. By default, figures use the theme of the MATLAB desktop, but you can choose a different graphics theme.



You can manage the colors of a chart and the theme of a figure in these ways:

- `theme` function or the Theme property of the figure – Set and get the theme of a figure.
- MATLAB Settings window – Configure themes for all figures.
- `fliplightness` function – Flip the lightness of color values.
- `getTheme` method – Get the theme of the figure for a `ChartContainer` subclass.

For more information, see Graphics and App Themes.

### ★ Figure Container: Organize, explore, customize, and generate code for figures using an improved figure window

Figures now open as tabs docked in a figure container by default. You can use the figure container to organize, explore, customize, and generate code for figures more efficiently:

- View figures more easily with their larger default size and wider aspect ratio.
- Customize the layout of open figures by rearranging their tabs, or view multiple figures simultaneously by arranging them in tiles.
- Quickly customize and annotate figures using the toolstrip of the figure container.
- Automatically generate code to reproduce figure interactions and customizations.

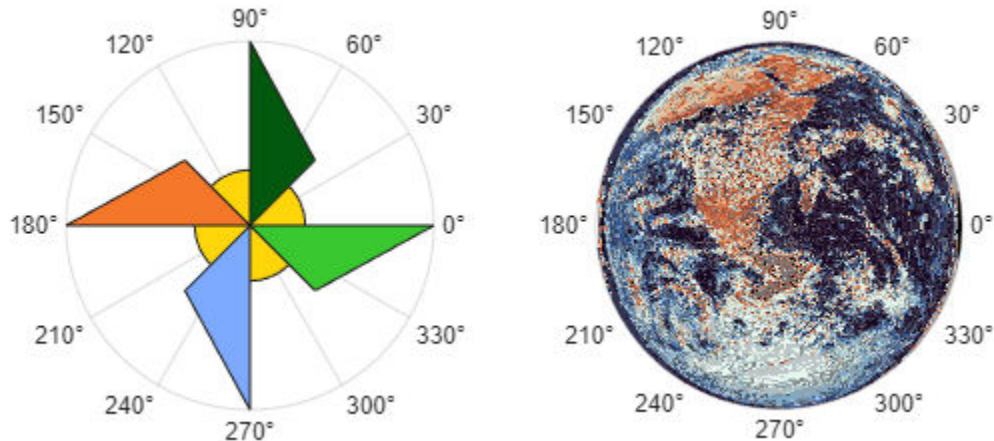
For more information, see Interactively Customize Plot Appearance.

For compatibility considerations, see “Figure Container: Figures appear larger and as tabs by default” on page 2-38.



## Polar Plots: Plot patches and surfaces in polar coordinates

Create patches and surfaces in polar coordinates by passing a `PolarAxes` object to the `patch`, `pcolor`, `surf`, `surface`, `mesh`, `fill`, and `fill3` functions.



## Polar Plots: Specify location of zero degrees as an angle

Specify the location of zero degrees on the *theta*-axis as an angle value by setting the `ThetaZeroLocation` property of the `PolarAxes` object. You can assign a value such as 45 (when the `ThetaAxisUnits` property is "degrees") or  $\pi/4$  (when the `ThetaAxisUnits` property is "radians").

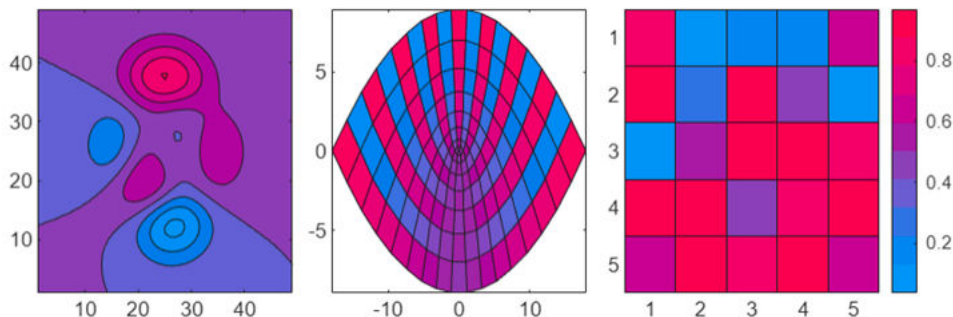
The `ThetaZeroLocation` property also supports its original options: "right", "left", "top", and "bottom".

## colormaplist Function: List available colormaps

Return a list of predefined colormaps as a string array by using the `colormaplist` function.

## nebula Function: Apply blue-to-red colormap to charts

Use the `nebula` function to get a blue-to-red colormap for coloring your charts. Like for all predefined colormaps, you can optionally specify the number of colors for the `nebula` colormap.



## **Pie Charts and Donut Charts: Choose text interpreter or no interpreter for titles and labels**

Use LaTeX or TeX markup in the titles and labels of pie charts and donut charts by setting the `Interpreter` property of a `PieChart` or `DonutChart` object to `"latex"` or `"tex"`. To display text without an interpreter, set this property to `"none"`.

## **violinplot Function: Use additional grouping data options and customize color group appearance**

You can now create violin plots from table and timetable data without specifying grouping variables. Use the new syntax `violinplot(tbl,yvar)` of the `violinplot` function to create violin plots from the `yvar` variables in table `tbl`.

You can also specify one shared set of sample data for multiple sets of positional grouping data. To share sample data coordinates, specify `ydata` as a vector and `xgroupdata` as a matrix in the syntax `violinplot(xgroupdata,ydata)`.

In addition, you can customize the width and layout of color groups using these name-value arguments:

- `ColorGroupWidth` — Adjust the space between each violin plot color grouping.
- `ColorGroupLayout` — Specify whether to group or overlay the violin plots in each color grouping.
- `DensityWidth` — Adjust the space between each overlaid color grouping.

## **boxchart Function: Create box charts using table data and customize box charts using additional properties**

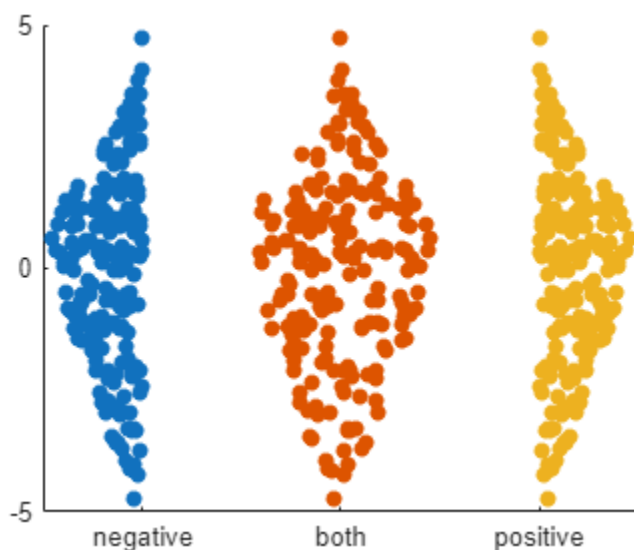
You can now create box charts from table and timetable data. Use the new syntax `boxchart(tbl,yvar)` of the `boxchart` function to create box charts from the `yvar` variables in table `tbl`. To create grouped box charts from a table, use `boxchart(tbl,xvar,yvar)`, where the data in `yvar` is grouped by the data in `xvar`.

You can further customize the appearance of box chart whiskers as well as color groups using these name-value arguments:

- `CapWidth` — Set the width of the whisker end caps.
- `ColorGroupWidth` — Adjust the space between each box chart color grouping.
- `ColorGroupLayout` — Specify whether to group or overlay the box charts in each color grouping.
- `BoxWidth` — Adjust the space between each overlaid color grouping.

## **Swarm Charts: Create swarm charts with jitter in different directions**

Set the `XJitterDirection`, `YJitterDirection`, and `ZJitterDirection` properties to control the direction of the jitter in a swarm chart. Each of these properties can have a value of `"negative"`, `"positive"`, or `"both"`.



## Geographic Plots: Basemaps have improved appearance

The "streets-light", "streets-dark", "streets", and "topographic" basemaps hosted by Esri® have an improved appearance when the basemap tiles contain text. For example, this image compares a basemap in R2024b with the same basemap in R2025a. In R2025a, the labels for Wisconsin and St. Louis are fully visible.



In addition, all cached basemap tiles have a crisper appearance. For more information about cached basemap tiles, see [Access Basemaps for Geographic Axes and Charts](#).

## **tightPosition Function: Get location and size of axes plotting area in specific units**

Get the location and size of the axes plotting area in specific units, such as pixel, inch, or centimeter units, by calling the `tightPosition` function with the `Units` name-value argument. For example, get the plotting area in inches.

```
ax = gca;  
pos = tightPosition(ax,Units="inches");
```

## **exportgraphics and copygraphics Functions: Specify output dimensions using name-value arguments**

Specify the width, height, padding, and whether to preserve the aspect ratio of the original graphic when you call the `exportgraphics` and `copygraphics` functions.

Use these name-value arguments to control these aspects of the content:

- `Width` and `Height` — Specify the width and height of the output.
- `Padding` — Specify the amount of padding around the perimeter of the graphic.
- `Units` — Specify the units for the width, height, and padding values.
- `PreserveAspectRatio` — Specify whether to automatically add padding to preserve the original aspect ratio if the `Width` and `Height` values conflict with the original aspect ratio of the graphic.

## **exportgraphics Function: Export plots as SVG files**

Using the `exportgraphics` function, you can export a plot as an SVG file by specifying an ".svg" file extension in the filename. For example, export the contents of the current figure as an SVG file.

```
exportgraphics(gcf,"myfigure.svg")
```

## **getframe Function in MATLAB Online: Capture figures in MATLAB Online sessions**

The `getframe` function is now supported in MATLAB Online. You can also use `getframe` to capture figures created with the `uifigure` function in any environment.

## **Copying Figures: Select options for copying figures in MATLAB Settings**

Use the new **Figure Copy** page in the MATLAB Settings window to select options for copying figures to the clipboard. MATLAB uses these settings when you copy a figure using the figure toolbar.

The new settings page includes these options:

- Width and height in specified units
- Image (pixels) or vector graphics
- Resolution (for images)

- Background color
- Grayscale content

## Titles and Labels: Use LaTeX to display math and symbols with more consistent results

As in previous releases, you can use the LaTeX interpreter to display text and labels in your graphics. Now, when you use markup in your graphics, the results are consistent across MATLAB. For more information, see Supported LaTeX Commands.

## Text: Larger, easier-to-read text on macOS

Text in graphics has a larger, more readable default font size on macOS systems. The size is now consistent across Windows and macOS systems. Specifically, graphics objects that have a `Units` or `FontUnits` property set to "points", "inches", or "centimeters" display about 1.33x larger than their previous size on macOS systems.

For compatibility considerations, see "Text: Text and other objects specified in "points", "inches", or "centimeters" are larger on macOS" on page 2-41.

## ⚠ Axes Interactions: Specify axes for interaction mode

In figures created using the `figure` function, you can set the interaction mode for axes. Specify the `Axes` object as the first argument for these functions:

- `pan`
- `zoom`
- `rotate3d`
- `datacursormode`
- `brush`

For example, in a figure with two axes, enable pan mode for only the first axes.

```
f = figure;
t = tiledlayout(f,1,2);
ax1 = nexttile(t);
ax2 = nexttile(t);
pan(ax1,"on")
```

When setting the interaction mode for axes, these functions do not return `pan`, `zoom`, `rotate3d`, `DataCursorManager`, or `brush` objects. Previously, these functions set the interaction mode for all `Axes` children of the current or target figure.

## ⚠ Compatibility Considerations

If you change an interaction mode for an `Axes` object in a figure, the interaction modes of other axes in that figure are not affected.

If you create a new plot in existing axes, interaction modes for the axes are now reset to their default values. Previously, creating a new plot in existing axes did not change the interaction mode for the figure.

## Accessibility in MATLAB: Use screen reader to interact with figures

You can use a screen reader and keyboard commands to pan, zoom, and rotate when you work with plotted data. Using a screen reader to interact with figures is not supported in the Live Editor.

For more details on interacting with MATLAB figures, see [Control Chart Interactivity](#).

## ⚠ Graphics functionality being removed or changed

### Figure Container: Figures appear larger and as tabs by default

*Behavior change*

By default, figures created using the `figure` function appear as tabs in a figure container. This change impacts the figures in these ways:

- By default, the `WindowState` property of Figure objects is "docked" because the figures are docked in the figure container. In previous releases, the `WindowState` property was "normal" because figures were not docked by default.
- By default, the figure container is undocked from the MATLAB desktop, and the figure container docked state does not persist between MATLAB sessions. In previous releases, the figure container was docked in the desktop by default, and the figure container docked state persisted between MATLAB sessions.

To create a figure window that is independent of other windows, specify the window style as "normal".

```
f = figure(WindowStyle="normal");
```

To dock the figure container into the desktop, click the Dock button  in the top-right corner of the figure container.

In addition, figures have a larger default size that changes based on the size of your monitor. To use the previous default figure size for a single figure, set its `Position` property using a position vector of the form `[left bottom width height]`. Setting the `Position` property undocks the figure from the figure container.

```
f = figure(Position=[100 100 560 420]);
```

To use the previous default initial figure size for all figures, including those in the figure container, set the default figure position of the graphics root object.

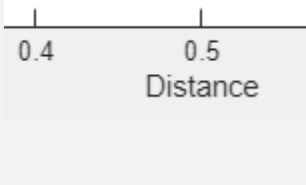
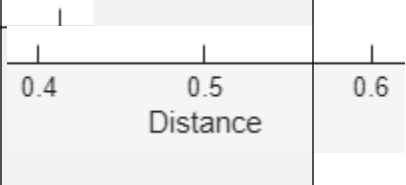
```
set(groot,DefaultFigurePosition=[100 100 560 420])
```

### Plot Appearance: Default colors for charts have changed slightly

*Behavior change*

Some of the default colors of graphics objects have changed. These changes affect the light theme colors. This table describes the changes and provides the RGB triplets and color samples to compare R2025a with the previous release.

| Affected Graphics           | Affected Object Properties                                                                                                          | R2024b Color                 |                                                                                       | R2025a Color                 |                                                                                       |
|-----------------------------|-------------------------------------------------------------------------------------------------------------------------------------|------------------------------|---------------------------------------------------------------------------------------|------------------------------|---------------------------------------------------------------------------------------|
| Figure background color     | Color property of Figure object                                                                                                     | [0.94 0.94 0.94]             |                                                                                       | [0.9608 0.9608 0.9608]       |                                                                                       |
|                             |                                                                                                                                     |                              |                                                                                       |                              |                                                                                       |
| Plot colors                 | ColorOrder property of Axes object<br><br>(This property controls the default color of Line, Scatter, Bar, and other plot objects.) | <b>RGB Triplet</b>           | <b>Sample</b>                                                                         | <b>RGB Triplet</b>           | <b>Sample</b>                                                                         |
|                             |                                                                                                                                     | [0.0000<br>0.4470<br>0.7410] |    | [0.0660<br>0.4430<br>0.7450] |    |
|                             |                                                                                                                                     | [0.8500<br>0.3250<br>0.0980] |    | [0.8660<br>0.3290<br>0.0000] |    |
|                             |                                                                                                                                     | [0.9290<br>0.6940<br>0.1250] |    | [0.9290<br>0.6940<br>0.1250] |    |
|                             |                                                                                                                                     | [0.4940<br>0.1840<br>0.5560] |   | [0.5210<br>0.0860<br>0.8190] |   |
|                             |                                                                                                                                     | [0.4660<br>0.6740<br>0.1880] |  | [0.2310<br>0.6660<br>0.1960] |  |
|                             |                                                                                                                                     | [0.3010<br>0.7450<br>0.9330] |  | [0.1840<br>0.7450<br>0.9370] |  |
|                             |                                                                                                                                     | [0.6350<br>0.0780<br>0.1840] |  | [0.8190<br>0.0150<br>0.5450] |  |
| Plot titles and annotations | Color property of Text or annotation object                                                                                         | [0 0 0]                      |                                                                                       | [0.1294 0.1294 0.1294]       |                                                                                       |
|                             |                                                                                                                                     | My Plot Title                |                                                                                       | My Plot Title                |                                                                                       |

| Affected Graphics                    | Affected Object Properties                                                        | R2024b Color                                                                                      | R2025a Color                                                                                          |
|--------------------------------------|-----------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------|
| Axes labels, tick labels, and rulers | Axes labels: Color property of Text object                                        | [0.15 0.15 0.15]                                                                                  | [0.1294 0.1294 0.1294]                                                                                |
|                                      | Axes tick labels and rulers: XColor, YColor, and ZColor properties of Axes object |  0.40.5Distance |  0.40.50.6Distance |

**Rendering: opengl function, Renderer property, and renderer startup options have no effect**

*Behavior change*

The following options for querying or setting the graphics renderer have no effect. MATLAB no longer uses OpenGL® technology to render graphics, and you no longer need to set the renderer in your graphics workflows.

- Calling the `opengl` function has no effect. To query the graphics renderer, use the `rendererinfo` function instead.
- The `Renderer` property of a figure has no effect.
- The MATLAB startup options `-softwareopengl`, `-nosoftwareopengl`, `-softwareopenglmesa`, and `-noopengl` have no effect.

The `opengl` function, `Renderer` property, and startup options will be removed in a future release.

**Rendering: GraphicsSmoothing and FontSmoothing properties have no effect**

*Behavior change*

The `GraphicsSmoothing` and `FontSmoothing` properties have no effect. All graphics and text are smooth regardless of the value of these properties.

- The `GraphicsSmoothing` property of figures has no effect.
- The `FontSmoothing` property of axes, rulers, geographic scales, and text objects has no effect.

These properties will be removed in a future release.

**TeX and LaTeX Equations: Fonts and text spacing of equations have changed**

*Behavior change*

The fonts and text spacing of some TeX and LaTeX equations and symbols have changed. For example, here are two paragraphs containing LaTeX equations and a fraction. The left paragraph shows the appearance in R2024b, and the right paragraph shows the same text in R2025a.

Content exported from a figure as vector graphics (SVG, PDF, EPS, or EMF files) continue to use the R2024b format in R2025a.



| R2024b                                                                                                                                                                                                                                                                               | R2025a                                                                                                                                                                                                                                                                               |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <p>I can write equations like<br/> <math>c = \sqrt{a^2 + b^2}</math> and <math>\omega^2 = \gamma^2 + \theta^2</math>.</p> <p>I can also calculate the number of combinations of<br/> <math>n</math> items taken <math>k</math> at a time using <math>\frac{n!}{k!(n-k)!}</math>.</p> | <p>I can write equations like<br/> <math>c = \sqrt{a^2 + b^2}</math> and <math>\omega^2 = \gamma^2 + \theta^2</math>.</p> <p>I can also calculate the number of combinations of<br/> <math>n</math> items taken <math>k</math> at a time using <math>\frac{n!}{k!(n-k)!}</math>.</p> |

### **Text: Text and other objects specified in "points", "inches", or "centimeters" are larger on macOS**

#### *Behavior change*

Graphics objects that have a `Units` or `FontUnits` property set to "points", "inches", or "centimeters" now use a conversion value of 1 pixel = 1/96 inch on macOS systems. Previously, the conversion value was 1 pixel = 1/72 inch. As a result, these objects have increased in size by a factor of 96/72 (about 1.33x). The new size provides consistency between macOS and Windows systems.

If you have code that uses the default ("normalized") units for positioning objects such as axes and annotations, those objects are unaffected by this change. However, text objects, such as titles and labels, appear larger by default on macOS systems. To preserve the sizes of text and other affected objects so that they match the sizes on macOS systems in previous releases, make these changes in your code:

- Objects with `Units` or `FontUnits` set to "points" — Update the value of the property from "points" to "pixels".
- Objects with `Units` or `FontUnits` set to "inches" — Update the value of the property from "inches" to "pixels" and multiply all `Position` values by 72.
- Objects with `Units` or `FontUnits` set to "centimeters" — Update the value of the property from "centimeters" to "pixels" and multiply all `Position` values by 72/2.54.

### **Box Charts: Box widths are calculated based on input data**

#### *Behavior change*

If you call the `boxchart` function without specifying the box width, MATLAB calculates the box width according to your data. In previous releases, the default box width was 0.5. To preserve the appearance from previous releases, specify the `BoxWidth` name-value argument as 0.5 when calling the `boxchart` function.

### **Color Utility Functions: The `colordef`, `graymon`, and `whitebg` functions have been removed**

#### *Errors*

The `colordef`, `graymon`, and `whitebg` functions have been removed. To control the appearance of graphics in a figure, select a graphics theme. For more information, see [Graphics and App Themes](#).

### **Modifying Plots: Plot tools are no longer supported**

#### *Errors*

The Figure Palette, Plot Browser, and Property Editor tools have been removed or incorporated into the toolbar of the figure container. As a result, these functions now return an error:

- `figurepalette`
- `plotbrowser`
- `propertyeditor`
- `propedit`
- `plottools`

In addition, the `"showtoolsmenu"` and `"hidetoolsmenu"` options for the `plotedit` function return an error.

Instead, to interactively select graphics objects and modify their properties, you can enable plot edit mode and use the Property Inspector. To create a grid of axes in a figure, use the  `tiledlayout`  function. To create new charts within existing axes, use the **Plots** tab in the MATLAB Toolstrip and select the **Reuse Figure** option.

### Figure Menu Bar: Menu bar no longer contains some menu items

*Behavior change*

For figures that display the menu bar instead of the toolstrip (for example, from setting the `MenuBar` property of the `Figure` object), the menu bar no longer contains the items described in this table.

| Menu         | Removed Item                                                                                                                                                                                                         | Recommendation                                                                                                                                                                                                                                                                                                                                                                                                                 |
|--------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>File</b>  | <ul style="list-style-type: none"> <li>• Import Data</li> <li>• Save Workspace</li> </ul>                                                                                                                            | Use the <b>Variable</b> section on the <b>Home</b> tab of the MATLAB Toolstrip.                                                                                                                                                                                                                                                                                                                                                |
| <b>Edit</b>  | <ul style="list-style-type: none"> <li>• Find Files</li> <li>• Clear Figure</li> <li>• Clear Command Window</li> <li>• Clear Command History</li> <li>• Clear Workspace</li> </ul>                                   | Use the <b>File</b> , <b>Variable</b> , and <b>Code</b> sections on the <b>Home</b> tab of the MATLAB Toolstrip.                                                                                                                                                                                                                                                                                                               |
| <b>View</b>  | <ul style="list-style-type: none"> <li>• Figure Palette</li> <li>• Plot Browser</li> <li>• Property Editor</li> </ul>                                                                                                | See “Modifying Plots: Plot tools are no longer supported” on page 2-41.                                                                                                                                                                                                                                                                                                                                                        |
| <b>Tools</b> | <ul style="list-style-type: none"> <li>• Zoom In</li> <li>• Zoom Out</li> <li>• Pan</li> <li>• Rotate 3D</li> <li>• Data Tips</li> <li>• Restore View</li> <li>• Options</li> <li>• Align Distribute Tool</li> </ul> | <p>Interact with axes by using built-in axes interactions, or enable interaction modes by using the axes toolbar. Customize an interaction by right-clicking the chart when an interaction mode is enabled or by using the <code>InteractionOptions</code> property of the axes.</p> <p>Control the arrangement of multiple axes in a figure by creating a tiled chart layout using the <code>tiledlayout</code> function.</p> |

**Axes Interactions: Plot-box outline rotation is no longer supported***Behavior change*

The plot-box outline rotation style for the `rotate3d` object is no longer supported. The rotation style rotates the entire axes, regardless of the value of the `RotateStyle` property.

**Axes Interactions: Azimuth and elevation angles are not displayed by default***Behavior change*

When you interactively rotate axes, the axes no longer display the azimuth and elevation angles by default. If you want to display these angles, in the `InteractionOptions` property of the axes, specify the `RotateIndicator` property as `"azimuthelevation"`.

For example, create a 3-D plot and set its axes to display the azimuth and elevation angles during interactive rotation.

```
ax = axes;  
surf(ax,peaks)  
ax.InteractionOptions.RotateIndicator = "azimuthelevation";
```

**Axes Interaction Modes: Enable property of interaction objects might be inaccurate***Behavior change*

The `Enable` property of zoom, pan, `rotate3d`, and brush interaction objects might have a value of `off` when the interaction is enabled. If you want to determine whether the interaction mode is enabled, check the icons in the axes toolbar. If an interaction mode is enabled, its corresponding icon appears blue in the toolbar.

**▲Printing and exporting functionality being removed or changed****Printing: The `print` function no longer executes `SizeChangedFcn` callbacks***Behavior change*

If you write a `SizeChangedFcn` callback for a figure, the callback no longer executes when you call the `print` function.

**Exporting: The `hgsave` function and "compact" option of `savefig` will be removed***Warns*

The `hgsave` function issues a warning that it will be removed in a future release. To save a FIG file, use the `savefig` function instead.

If you use the `savefig` function with the `"compact"` argument, the function issues a warning that the option will be removed in a future release. `savefig` saves compact FIG files regardless of whether you specify the `"compact"` option. These compact files are not compatible with releases prior to R2014b.

**Exporting: The `getframe` function no longer captures figure borders or decorations***Errors*

The `getframe` function returns an error if you use the `rect` argument to capture a region that is not fully contained within the drawable area of the figure. The drawable area is the area inside the borders of the figure and does not include the figure toolstrip.

**Exporting: exportgraphics and copygraphics do not support adjacent containers***Behavior change*

If you pass a container (such as a figure or panel) to the `exportgraphics` or `copygraphics` function, and that container has child containers, the function returns an error in some cases:

- If there are multiple child containers, and at least one of the child containers has axes objects or standalone visualization objects in it
- If the parent container has children that are axes objects (or standalone visualization objects) and those children are next to containers at the same level in the graphics hierarchy

To capture the contents of a child container, pass that container to the `exportgraphics` or `copygraphics` function. To capture all of the content of a figure, use the `exportapp` function.

**Printing and Exporting: Background color of output no longer depends on InvertHardCopy property***Behavior change*

The `InvertHardCopy` property of a figure has no effect. If you call the `print` or `saveas` function to save a figure to a file or copy it to the clipboard, the default background color corresponds to the theme of the figure. The background is white if the figure has the light theme. The background is dark gray (almost black) if the figure has the dark theme. If you set the `Color` property of the figure before calling `print` or `saveas`, the output uses the color you specify regardless of the theme.

If you call the `print` function to print a hard copy, the background color is white (no ink) by default. If you set the `Color` property of the figure, the output uses that color.

**Printing and Exporting: The hgexport and printopt functions have been removed***Errors*

The `hgexport` function has been removed. To save the contents of a plot as an image or vector graphics file, use the `exportgraphics` function. To copy the contents of a plot to the clipboard, use the `copygraphics` function. To save the contents of an app as an image file, use the `exportapp` function.

The `printopt` function has been removed. To configure printer defaults, use the options provided by your operating system.

**Printing and Exporting: The print and saveas functions no longer support UI components and certain file types***Errors*

The `print` and `saveas` functions no longer support capturing UI components or containers such as panels and tabs. The functions also no longer support BMP, HDF, PBM, PCX, PGM, PPM, or full-page PostScript® (.ps) files.

To export a figure containing UI components or containers, use the `exportapp` function instead. For example, this code creates a simple app and exports the figure as a PDF file.

```
% Create figure with three UI components
f = uifigure;
button1 = uibutton(f,Position=[150 300 100 50]);
button2 = uibutton(f,Position=[300 300 100 50]);
slider1 = uislider(f,Position=[150 250 250 3]);
```

```
% Export the contents of the figure
exportapp(f, "myapp.pdf")
```

To export a graphic as a BMP, HDF, PBM, PCX, PGM, or PPM file, use the `getframe` and `imwrite` functions instead. For example, this code exports a figure as a BMP file.

```
plot([0 3 2 4 1]);
F = getframe(gcf);
imwrite(F.cdata, "myplot.bmp");
```

To export vector graphics files, use the `exportgraphics` function. Specify an `.svg`, `.eps`, `.pdf`, or `.emf` file extension and set the `ContentType` name-value argument to `"vector"`. For example, this code exports a figure as a PDF file.

```
plot([0 3 2 4 1]);
exportgraphics(gcf, "myplot.pdf", ContentType="vector")
```

### **Printing and Exporting: Figures shrink to fit page size and `Renderer` property has no effect on output**

#### *Behavior change*

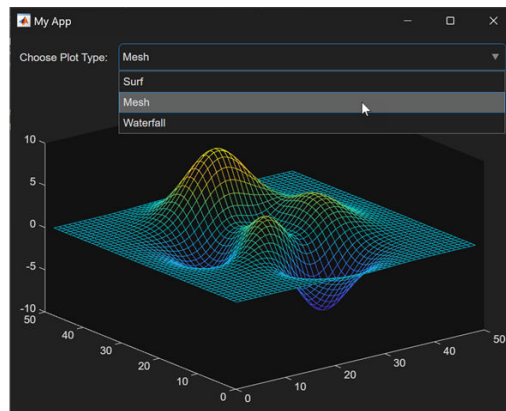
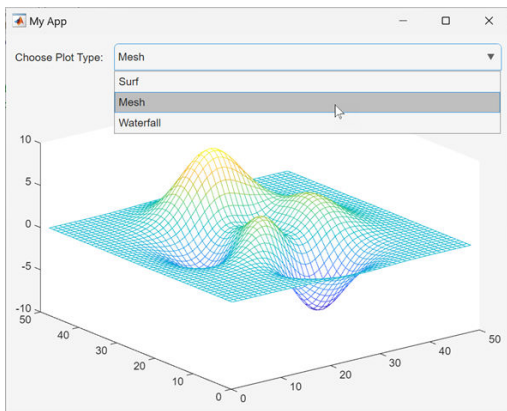
The `print` and `saveas` functions no longer clip the contents of a large figure when the page is smaller than the figure. Now, these functions shrink the content to fit on the page and preserve the aspect ratio of the figure.

The `Renderer` property of a figure has no effect on the output produced by the `print` and `saveas` functions.

## App Building

### ★ Themes: Create apps and custom UI components using light or dark theme

You can view and create figures in light or dark theme. When you create UI components and plots, MATLAB automatically selects colors that are visually appealing for the theme. By default, figures use the theme of the MATLAB desktop, but you can choose a different graphics theme.



You can manage the colors and themes of your apps in these ways:

- `theme` function or Theme property of the figure — Set and get the theme of a figure.
- MATLAB Settings window — Configure themes for all figures.
- ThemeChangedFcn property of the figure — Program response to figure theme change.
- `fliplightness` function — Flip the lightness of color values.
- `getTheme` method — Get the theme of the figure for a ComponentContainer class.

For more information, see Graphics and App Themes.

### uiprintdlg and uiexportdlg Functions: Open dialog boxes for printing and exporting figures

You can open dialog boxes for printing and exporting figures by using the `uiprintdlg` and `uiexportdlg` functions, respectively. `uiprintdlg` enables printing a figure to an external printer or a PDF, and `uiexportdlg` enables saving a figure in file formats such as PNG, TIFF, and JPG.

### uislider Function: Specify step size for slider

When you create a slider component using the `uislider` function, you can specify the amount that the value changes when a user interacts with the slider thumbs by setting the Step property.

For more information, see Slider.

## uitable Function: View multiline content in table cells

When you create a table UI component using the `uitable` function, the table adjusts row heights to display multiline cell content. For example, this code creates a table that includes two lines of text in one cell. The height of the row with multiline content adjusts to display both lines of text.

```
fig = uifigure;
t = uitable(uifigure, ...
    Data=["Omelette", "Eggs"; "Hamburger", "Beef"+newline+"Bun"]);
```



|   | 1         | 2           |
|---|-----------|-------------|
| 1 | Omelette  | Eggs        |
| 2 | Hamburger | Beef<br>Bun |

## UI Components: Interact with UI components using keyboard

You can use keyboard shortcuts to interact with slider, knob, and switch UI components. To move focus to a slider, knob, or switch in an app, use the **Tab** key. Once a component is in focus, use the arrow keys to change the value of a slider or knob or the **Space** or **Enter** keys to toggle the value of a switch.

Additionally, interacting with table UI components in figure-based apps now uses different keyboard shortcuts. To move focus to a table in an app, use the **Tab** key. Once a table is in focus, use the arrow keys to navigate between different cells in the table. In previous releases, the **Tab** key navigated between different cells in the table.

## UI Components: Replace UIControl objects with UI components

If you have an existing app created using the `figure` function, you can now add UI components to the app. Replacing `UIControl` objects with UI components is a first step for updating figure-based apps to use `uifigure`, which is the recommended functionality for building apps.

In addition, UI components have a larger set of configuration options than `UIControl` objects. For example, you can replace a `UIControl` push button with a `Button` object created using the `uibutton` function and then add HTML markup to the button text by setting the `Interpreter` property to "html".

For more information about updating figure-based apps to use UI components, see [Update UIControl Objects and Callbacks](#).

## App Designer: Package and share apps more efficiently

When you package apps from App Designer to share with others, the packaging steps are simpler and use a new interface. In addition, apps are now packaged as MLTBX files instead of MLAPPINSTALL files.

To package an App Designer app, use these steps:

- 1 With the app open in App Designer, on the **Designer** tab, select **Share > MATLAB App**.
- 2 Review the app information by following the steps in the Configure MATLAB App for Sharing dialog box, and then click **Package**.

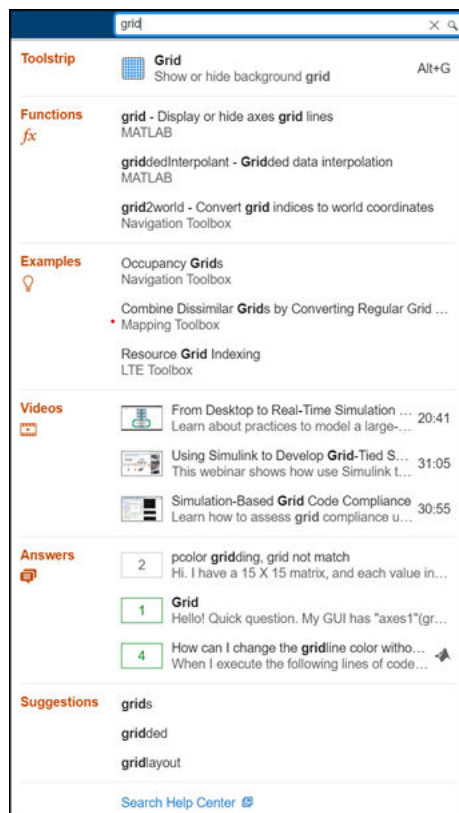
When the packaging process completes successfully, MATLAB creates an MLTBX file with the packaged app. You can install the app by opening the MLTBX file and then access the app from the **Apps** tab of the MATLAB Toolstrip. Or, you can share the app with others by sharing the MLTBX file directly or uploading it to MATLAB File Exchange.

For more information, see Package Apps in App Designer.

## App Designer: Access toolstrip actions and Help Center resources using search

You can now use the search box in the top-right corner of App Designer to easily access toolstrip actions and Help Center resources, such as functions, examples, and answers.

For example, to show the background grid on the App Designer canvas, type the word `grid` in the search box. In the list of results, select the **Grid** toolstrip action.



## Text: Larger, easier-to-read text on macOS

Text in apps has a larger, more readable default font size on macOS systems. The size is now consistent across Windows and macOS systems. Specifically, objects that have a `Units` or



FontUnits property set to "points", "inches", or "centimeters" display about 1.33x larger than their previous size on macOS systems.

For compatibility considerations, see “Text: Text and other objects specified in "points", "inches", or "centimeters" are larger on macOS” on page 2-56.

## ▲ **Functionality being removed or changed**

### **GUIDE design environment and guide function have been removed**

#### *Errors*

The GUIDE design environment and guide function have been removed. You can still run existing GUIDE apps in MATLAB, but you can no longer edit apps using the drag-and-drop GUIDE environment.

To continue working with an existing GUIDE app and to maintain its compatibility with future MATLAB releases, use one of the recommendations listed in the table.

| <b>App Development Needs</b> | <b>Strategy</b>                                                                             |
|------------------------------|---------------------------------------------------------------------------------------------|
| Ongoing development          | Migrate your app to App Designer using the GUIDE to App Designer Migration Tool for MATLAB. |
| Occasional editing           | Edit your app using the existing app code file that GUIDE created.                          |
| Only running                 | Continue to run your GUIDE app.                                                             |

For more information about maintaining and migrating apps, see GUIDE Migration Strategies.

To create new apps, use App Designer.

### **actxcontrol, actxcontrollist, and actxcontrolselect functions have been removed**

#### *Errors*

The actxcontrol, actxcontrollist, and actxcontrolselect functions have been removed. If you call the functions, MATLAB returns an error. For app development using App Designer, update your code to use alternative functionality. For a list of functionality that you can use instead, see Recommendations for MATLAB Apps Using Java & ActiveX.

### **pagesetupdlg function has been removed**

#### *Errors*

The pagesetupdlg function has been removed. If you call pagesetupdlg, MATLAB returns an error. Use uiprintdlg instead.

### **printpreview, printdlg, and exportsetupdlg functions will be removed**

#### *Warns*

The printpreview, printdlg, and exportsetupdlg functions will be removed in a future release. If you call printpreview or printdlg, MATLAB issues a warning and runs the new uiprintdlg function instead. If you call exportsetupdlg, MATLAB issues a warning and runs the new uiexportdlg function instead.

To simplify the interfaces, the new dialog boxes do not include options for customizing fonts, line width, or tick values, or for setting the renderer. To control the appearance of a figure that you want to print or export, set properties of the figure and the objects in the figure before opening the dialog box.

**matlab.apputil.create function will be removed**  
*Still runs*

The `matlab.apputil.create` function will be removed in a future release. If you call `matlab.apputil.create`, the function now prompts you to package your app as a toolbox. Calling the function is equivalent to selecting **Add-Ons > Package Toolbox** from the **Home** tab of the MATLAB Toolstrip.

**UI Components: Some default colors for UI components have changed slightly**  
*Behavior change*

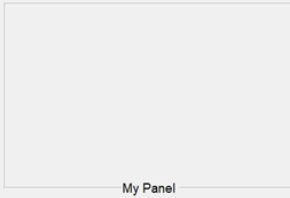
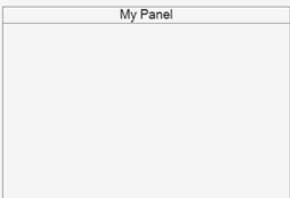
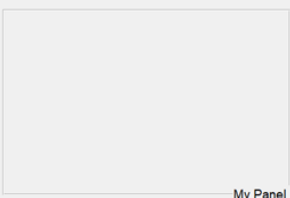
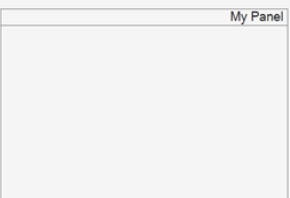
Some of the default colors of UI components and containers have changed. These changes affect the light theme colors. This table describes the changes and provides the RGB triplets and color samples to compare R2025a with the previous release.

| Affected App Area     | Affected Component Properties                                                                                                                          | R2024b Color                                   | R2025a Color                                        |
|-----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------|-----------------------------------------------------|
| Container backgrounds | <ul style="list-style-type: none"><li>Color property of Figure object</li><li>BackgroundColor property of UI container objects such as Panel</li></ul> | [0.94 0.94 0.94]                               | [0.9608 0.9608 0.9608]                              |
| App text              | FontColor property of UI component objects                                                                                                             | [0 0 0]<br>Label                               | [0.1294 0.1294 0.1294]<br>Label                     |
| Unvisited hyperlinks  | FontColor property of Hyperlink object                                                                                                                 | [0 0.3984 0.7969]<br><a href="#">Hyperlink</a> | [0.0667 0.4431 0.7451]<br><a href="#">Hyperlink</a> |
| Visited hyperlinks    | VisitedColor property of Hyperlink object                                                                                                              | [0.5 0 0.5]<br><a href="#">Hyperlink</a>       | [0.5216 0.0863 0.8196]<br><a href="#">Hyperlink</a> |

**UI Components: Updated look for panels and button groups in figure-based apps**  
*Behavior change*

Panels and button groups created in figure-based apps have an updated appearance. Because of this update, certain title and border options have changed.

Panel and button group titles can appear only at the top of the container. As a result, these `TitlePosition` values have changes in behavior.

| Value          | R2024b and Earlier                                                                | Starting in R2025a                                                                 | Recommended Value                                                                            |
|----------------|-----------------------------------------------------------------------------------|------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------|
| "leftbottom"   |  |  | Update your code to use "lefttop" instead of "leftbottom" to reflect the title position.     |
| "centerbottom" |  |  | Update your code to use "centertop" instead of "centerbottom" to reflect the title position. |
| "rightbottom"  |  |  | Update your code to use "righttop" instead of "rightbottom" to reflect the title position.   |

Panel and button group borders always appear as a line. As a result, some **BorderType** values have changes in behavior. Specifying **BorderType** as any of the options in the table causes a warning. Additionally, the default **BorderType** value has changed from 'etchedin' to 'line'.

| Value       | R2024b and Earlier                                                                  | Starting in R2025a                                                                   | Recommended Value                                                                 |
|-------------|-------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------|
| "etchedin"  |  |  | Update your code to use "line" instead of "etchedin" to reflect the border type.  |
| "etchedout" |  |  | Update your code to use "line" instead of "etchedout" to reflect the border type. |
| "beveledin" |  |  | Update your code to use "line" instead of "beveledin" to reflect the border type. |

| Value        | R2024b and Earlier                                                                | Starting in R2025a                                                                 | Recommended Value                                                                  |
|--------------|-----------------------------------------------------------------------------------|------------------------------------------------------------------------------------|------------------------------------------------------------------------------------|
| "beveledout" |  |  | Update your code to use "line" instead of "beveledout" to reflect the border type. |

As a result of the changes to the `BorderType` property, the `ShadowColor` property has no effect and warns if you set it. To specify the border color of a panel or a button group, use the `BorderColor` property instead.

The `ShadowColor` property no longer appears in the list returned by calling the `get` function on a `Panel` or `ButtonGroup` object. For more information about container properties, see `Panel` or `ButtonGroup`.

**UI Components: Table Extent property returns default table UI component width and height and is not recommended**

*Behavior change*

When you query the `Extent` property of a table UI component created using the `uitable` function, the property now always returns `[0 0 300 300]` in pixel units, or the equivalent value if the table is using non-pixel units. The third and fourth elements of the vector represent the default table width and height. Previously, querying `Extent` in a figure-based app returned a vector where the third and fourth elements represent the width and height of the rectangle outlining the table area. If you use the `Extent` property to specify the position of UI components in your app, the table UI component might have a different size or the layout of your app might appear shifted.

Update your code to remove references to `Extent` using one of these alternatives:

- Manage your app layout using a grid layout manager created using the `uigridlayout` function. Specify a row height or column width of "fit" to automatically scale your app layout based on the size of the table with its data.
- Lay out the table and other UI components by manually specifying the `Position` property value.

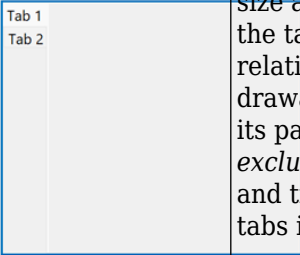
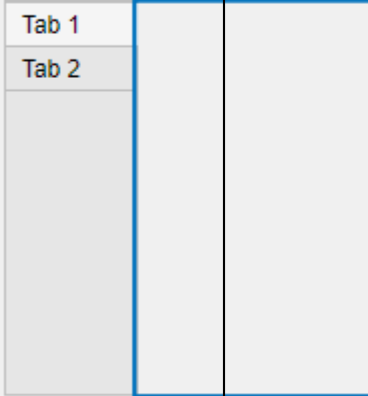
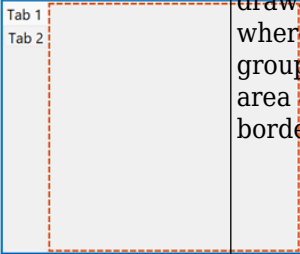
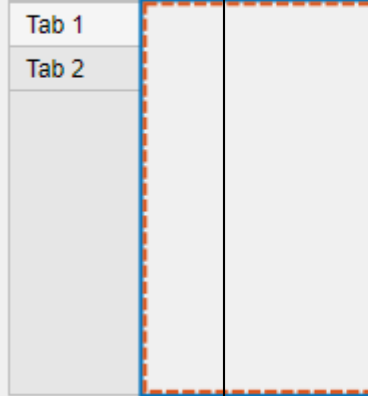
The `Extent` property is not recommended and no longer appears in the list returned by calling the `get` function on a `Table` object.

**UI Components: InnerPosition property of tab groups returns the position excluding borders and titles**

*Behavior change*

For apps created using the `figure` function, when you query the `InnerPosition` property of a tab group, the property returns a different value than in the previous release. In addition, if the `TabLocation` property is 'left' or 'bottom', when you query the `Position`, `InnerPosition`, and `OuterPosition` properties of a tab in the tab group, the properties might return different values than in the previous release. These changes make the property values and behaviors consistent with the properties in `uifigure`-based apps.

This table describes the differences in the property values between R2025a and previous releases.

| Property                                            | R2024b and Earlier                                                                                                                                                                                                           |                                                                                                                                                                                 | Starting in R2025a                                                                                                                                                                                                           |                                                                                                                                                                                  |
|-----------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                                                     | Description                                                                                                                                                                                                                  | Image                                                                                                                                                                           | Description                                                                                                                                                                                                                  | Image                                                                                                                                                                            |
| InnerPosition of a tab group                        | A four-element vector of the form [left bottom width height] that represents the size and location of the tab group relative to the drawable area of its parent, <i>including</i> borders and titles of the tabs it contains | Tab group InnerPosition (blue solid line)<br>                                                 | A four-element vector of the form [left bottom width height] that represents the size and location of the tab group relative to the drawable area of its parent, <i>excluding</i> borders and titles of the tabs it contains | Tab group InnerPosition (blue solid line)<br>                                                 |
| Position, InnerPosition, and OuterPosition of a tab | A four-element vector of the form [left bottom width height] that represents the size and location of the tab relative to the tab group drawable area, where the tab group drawable area <i>includes</i> borders and titles  | Tab Position (orange dashed line) relative to tab group drawable area (blue solid line)<br> | A four-element vector of the form [left bottom width height] that represents the size and location of the tab relative to the tab group drawable area, where the tab group drawable area <i>excludes</i> borders and titles  | Tab Position (orange dashed line) relative to tab group drawable area (blue solid line)<br> |

If you notice changes in your app layout, consider updating your code in this way:

- **Tab group InnerPosition** — Replace references to the tab group `InnerPosition` property with the tab group `Position` property to use the size and location of the tab group including the borders and tab titles.
- **Tab Position, InnerPosition, and OuterPosition** — Replace references to the first two elements of the position vector to instead use the space taken up on the left or bottom by the tab titles. For example, if your app contains a tab group with a `TabLocation` value of 'left' and you reference the `Position` property of a tab in that tab group, replace that reference with the horizontal distance between the tab group border and the tab drawable area.

For more information about tab group and tab properties, see `TabGroup` and `Tab`.

### **UI Controls: Some control font size and text content might appear differently in figure-based apps**

*Behavior change*

Controls created using the `uicontrol` function have these differences:

- The default font used by the `uicontrol` function might have a different appearance and require a different amount of space than in previous releases. To update your app, use the `Position` property of the `UIControl` object to account for the space the text needs.
- `UIControl` check boxes and tooltips do not render HTML content.
- Radio button and toggle button callback functions are executed only when selecting a button that is not already selected. Previously, the callbacks were executed even if an app user clicked a button that was already selected.

### **Figures: Figure windows display a toolstrip instead of a menu bar and toolbar**

*Behavior change*

Starting in R2025a, when you create a figure using the `figure` function or load a saved figure, the figure appears with a toolstrip instead of a menu bar and toolbar. The figure toolstrip provides improved options for interacting with plots and data in a figure.

However, apps and saved figures that modify the figure menu bar or toolbar (for example, by setting the `ToolBar` property of the `Figure` object) continue to display the menu bar and toolbar instead of the toolstrip. If your app relies on menu bar or toolbar functionality, you can specify that the app display the menu bar or toolbar instead of the toolstrip by setting the figure `MenuBar` or `ToolBar` properties when you create the figure for your app.

```
f = figure;  
f.MenuBar = "figure";  
f.ToolBar = "figure";
```

In addition, figures created from a packaged standalone desktop application and figures created when running code noninteractively using the `-batch` startup option continue to display the menu bar and toolbar instead of the toolstrip.

### **Figures: Scroll event data for `WindowScrollWheelFcn` figure callback returns updated scroll count values**

*Behavior change*

When you specify a `WindowScrollWheelFcn` callback function for a figure created using the `figure` function, the callback event data returns different values in some cases. This table describes the differences.

| Event Data Property  | R2024b and Earlier                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         | Starting in R2025a                                                                                                          |
|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------|
| VerticalScrollCount  | <p>A positive or negative number that indicates the direction and number of scroll wheel clicks. Scrolling down returns a positive value. Scrolling up returns a negative value.</p> <p>The vertical scroll count is the sum of all scroll wheel clicks that occurred since the last time the callback executed. Typically, the value is 1 or -1. However, the scroll count can have a larger magnitude if the <code>WindowScrollWheelFcn</code> callback takes a long time to return or if the user spins the scroll wheel very fast.</p> | A positive or negative number that indicates the direction of scrolling. Scrolling down returns 1. Scrolling up returns -1. |
| VerticalScrollAmount | The number of lines scrolled for each click of the scroll wheel. Typically, the value is 3, but some systems might return a different value depending on how the mouse is configured.                                                                                                                                                                                                                                                                                                                                                      | 3                                                                                                                           |

### Figures: Modal figures always block all MATLAB windows

#### *Behavior change*

When you create a modal figure by specifying the figure `WindowStyle` property as "modal", the figure window blocks interaction with all MATLAB windows, including other figure windows and the MATLAB desktop. In previous releases, modal figures created using the `uifigure` function did not block the MATLAB desktop.

If you need to interact with the MATLAB desktop while a modal figure is in focus, press **Ctrl+C**. This action causes the figure `WindowStyle` property to change from 'modal' to 'normal'.

### Figures: Axes and containers laid out using nondefault units might appear in different positions

#### *Behavior change*

If you create a figure using the `figure` function and add axes or containers to the figure using a nondefault value for the `Units` property, the axes or containers might appear with a different location or size than in previous releases. The reason for the behavior change is that querying the `Position` property of a `Figure` object might not return the final size of the figure immediately after the figure is created.

For example, this code might create the axes at a larger size than in previous releases, and the axes width and height might be larger than half the final figure width and height.

```
f = figure;
a = axes(f, ...
```

```
Units="pixels", ...  
Position=[1 1 f.Position(3)/2 f.Position(4)/2]);
```

To lay out graphics objects or UI components relative to the size of the figure they are in, use one of these strategies:

- Use the default units to lay out the object (for example, "normalized" for Axes objects).
- Specify the figure position using the `Position` property.
- Use a layout manager by calling `tiledlayout` for plots or `uigridlayout` for UI components and containers.

### **Fonts: MATLAB supports slightly different set of fonts**

*Behavior change*

The set of fonts that MATLAB can display is slightly different than in previous releases. If you specify a font that MATLAB cannot display, MATLAB instead displays the text using a common font available on the machine. If MATLAB cannot display a specified font, you can select a different font instead. To see a list of fonts installed on your system, use the `listfonts` function.

### **Text: Text and other objects specified in "points", "inches", or "centimeters" are larger on macOS**

*Behavior change*

Objects that have a `Units` or `FontUnits` property set to "points", "inches", or "centimeters" now use a conversion value of 1 pixel = 1/96 inch on macOS systems. Previously, the conversion value was 1 pixel = 1/72 inch. As a result, these objects have increased in size by a factor of 96/72 (about 1.33x). The new size provides consistency between macOS and Windows systems.

If you have code that uses pixel units for positioning objects such as UI components in App Designer or in apps created using the `uifigure` function, those objects are unaffected by this change. However, text objects such as titles and labels, text in axes, and text in `UIControl` objects appear larger by default on macOS systems. To preserve the sizes of text and other affected objects so that they match the sizes on macOS systems in previous releases, make these changes in your code:

- Objects with `Units` or `FontUnits` set to "points" — Update the value of the property from "points" to "pixels".
- Objects with `Units` or `FontUnits` set to "inches" — Update the value of the property from "inches" to "pixels" and multiply all `Position` values by 72.
- Objects with `Units` or `FontUnits` set to "centimeters" — Update the value of the property from "centimeters" to "pixels" and multiply all `Position` values by 72/2.54.

### **Some javacomponent functionality has been removed**

*Behavior change*

The undocumented `javacomponent` function has some changes in behavior and some functionality has been removed. Replace calls to the `javacomponent` function with documented UI component alternatives.

For more details and a list of documented functionality that you can use instead, see [Recommendations for MATLAB Apps Using Java & ActiveX](#).



## Performance

### datetime Format Parsing: Improved performance when parsing common date and time formats

datetime format parsing performance is significantly faster for common date and time formats. Most date and time formats that are composed of letter identifiers separated by common delimiters see improved performance.

- Letter identifiers specify format components such as date fields, time fields, and time zone offsets. For a list of identifiers, see the `datetime` Format property.
- Common delimiters are ":", ";", ",", ".", "-", "/", and the space character.

Examples of common input formats are:

- "dd.MM.uuuu"
- "MM/dd/uuuu"
- "uuuu MM dd"
- "HH:mm, dd MMMM uuuu Z"
- "uuuu MMM dd HH:mm:ss.SSS a ZZZZZ"
- "uuuu-MM-dd 'T' HH:mm"
- "QQQ/uuuu"

In R2025a, there is no improvement for input formats that include D, G, Q, QQ, W, y, yy, and z. (Only the identifiers listed in the previous sentence are affected. There can still be a performance improvement when you use identifiers such as QQQ or yyyy.)

Code that uses the `datetime` function to parse and convert text is generally about 4–17x faster, depending on the size and format of the input array. For example, this code shows an example of the parsing performance improvement. It creates a string array of dates that use periods as delimiters. The dates in the string array use the `dd.MM.uuuu` format. Then the code uses the `datetime` function to parse and convert the string array. The code is about 9.6x faster than in the previous release.

```
function timingTest
s = randi([1,31],1e6,1) + ".10.2024";
d = datetime(s,InputFormat="dd.MM.uuuu");
end
```

The approximate execution times are:

**R2024b:** 3.27 s

**R2025a:** 0.34 s

The code was timed on a Windows 11, AMD EPYC™ 74F3 24-Core Processor @ 3.19 GHz test system using the `timeit` function.

```
timeit(@timingTest)
```

## String Operations: Improved performance with small string arrays

String operations, such as string construction and indexing, show improved performance when working on small string arrays. For example, this code constructs a string from a character vector. The code is about 2x faster than in previous release.

```
function timingTest
for i = 1:1e7
    str = string('MATLAB');
end
end
```

The approximate execution times are:

**R2024b:** 2.4 s

**R2025a:** 1.2 s

As another example, this code assigns a single element of a string array. The code is about 16x faster than in the previous release.

```
function timingTest
str = ["MATLAB"; "Simulink"];
for i = 1:1e7
    str(1) = "MATLAB";
end
end
```

The approximate execution times are:

**R2024b:** 4.5 s

**R2025a:** 0.28 s

The code was timed on a Windows 11, AMD EPYC 74F3 24-Core Processor @ 3.19 GHz test system using the `timeit` function.

```
timeit(@timingTest)
```

## writetable, writetimetable, writematrix, and writecell Functions: Improved performance when writing to existing sheets or writing new sheets with sheet names

The `writetable`, `writetimetable`, `writematrix`, and `writecell` functions show improved performance when writing data to existing sheets in XLSX, XLSM, XLTX, and XLTM files. These functions also show improved performance when writing data to new sheets with sheet names in those file types. The performance improvement increases with the number of sheets written.

For example, create an XLSX file with four sheets, where only the last sheet is nonempty.

```
function setupData
t1 = table(0);
filename = "test4.xlsx";
writetable(t1,filename,Sheet=4);
end
```

Then write data to the four individual sheets. This code is about 1.9x faster than in the previous release.

```
function timingTest
for ii = 1:4
    t2 = table(magic(20));
    writetable(t2,"test4.xlsx",Sheet=ii);
end
end
```

The approximate execution times are:

**R2024b:** 0.063 s

**R2025a:** 0.033 s

The code was timed using the `timeit` function.

```
setupData;
timeit(@timingTest)
```

As another example, write data to new sheets of an XLSX file, including sheet names. This code is about 1.8x faster than in the previous release.

```
data = rand(1000,100);
t = array2table(data);
filename = "test10.xlsx";
tic
for idx = 1:10
    sheetname = sprintf('Sheet_%02d',idx);
    writetable(t,filename,Sheet=sheetname);
end
toc
```

The approximate execution times are:

**R2024b:** 34.64 s

**R2025a:** 19.05 s

The code was timed on a Windows 11, AMD EPYC 74F3 24-Core Processor @ 3.19 GHz test system.

## parquetread Function: Improved performance when reading repeated string values

The `parquetread` function shows improved performance when reading data containing repeated string values. For example, create a table with repeated string values, and then export the table to a Parquet file.

```
function setupData
rng(1)
causes = ["attack" "earthquake" "energy emergency" ...
          "equipment fault" "fire" ...
          "severe storm" "thunderstorm" "unknown" ...
          "wind" "winter storm"];
idx = randi(numel(causes),[1e7 1]);
```

```
causes = causes(idx)';  
T = table(causes,VariableNames="OutageCause");  
parquetwrite("causes.parquet",T);  
end
```

Read the Parquet file. This code is about 4.8x faster than in the previous release.

```
function timingTest  
parquetread("causes.parquet");  
end
```

The approximate execution times are:

**R2024b:** 3.71 s

**R2025a:** 0.76 s

The code was timed on a Windows 11, AMD EPYC 74F3 24-Core Processor @ 3.19 GHz test system using the `timeit` function.

```
setupData;  
timeit(@timingTest)
```

## nufft Function: Improved performance with nonuniform sample points or query points

The `nufft` function shows improved performance when operating on either nonuniformly spaced sample points or nonuniformly spaced query points.

For example, this code computes the nonuniform discrete Fourier transform of a signal using 1,000,000 nonuniform sample points and 10,000 uniform query points. The code is about 13x faster than in the previous release.

```
function t = timingNonuniformSamplePoints  
t = sort(randn(1,1e6));  
f = 1:1e4;  
x = randn(size(t));  
  
y = @() nufft(x,t,f);  
t = timeit(y);  
end
```

The approximate execution times are:

**R2024b:** 1.59 s

**R2025a:** 0.12 s

As another example, this code computes the nonuniform discrete Fourier transform of a signal using 1,000,000 uniform sample points and 10,000 nonuniform query points. The code is about 14x faster than in the previous release.

```
function t = timingNonuniformQueryPoints  
t = 1:1e6;  
f = sort(randn(1,1e4));  
x = randn(size(t));
```

```
y = @() nufft(x,t,f);
t = timeit(y);
end
```

The approximate execution times are:

**R2024b:** 0.97 s

**R2025a:** 0.07 s

The code was timed on a Windows 11, AMD EPYC 74F3 24-Core Processor @ 3.19 GHz test system by calling the `timingNonuniformSamplePoints` and `timingNonuniformQueryPoints` functions.

## **mink, maxk, and topkrows Functions: Improved performance when returning a large number of rows for numeric or logical data**

The `mink`, `maxk`, and `topkrows` functions show improved performance for numeric data, including complex data, and logical data when the number of rows to return is large.

- The `mink` and `maxk` functions show improved performance when  $k$  (the number of minima or maxima to return) is at least 512 and is at least 8% of the operating dimension length.
- The `topkrows` function shows improved performance when  $k$  (the number of rows to return) is at least 2048 and is at least 20% of the total number of rows.

For example, this code returns the smallest 5000 elements of each column in a numeric matrix containing 10,000 rows by using the `mink` function. The code is about 3.75x faster than in the previous release.

```
function timingTestMink
A = rand(1e4,10);
for i = 1:120
    M = mink(A,5000);
end
end
```

The approximate execution times are:

**R2024b:** 1.05 s

**R2025a:** 0.28 s

Similarly, this code returns the top 5000 rows from a numeric matrix containing 10,000 rows in descending order, primarily based on the first column, by using the `topkrows` function. The code is about 2x faster than in the previous release.

```
function timingTestTopkrows
A = rand(1e4,10);
for i = 1:120
    T = topkrows(A,5000);
end
end
```

The approximate execution times are:

**R2024b:** 0.14 s

**R2025a:** 0.07 s

In both cases, the code was timed on a Windows 11, AMD EPYC 74F3 24-Core Processor @ 3.19 GHz test system using the `timeit` function.

```
timeit(@timingTestMink)
timeit(@timingTestTopkrows)
```

## **fillmissing Function: Improved performance when filling entries using linear interpolation**

The `fillmissing` function shows improved performance when you specify the fill method as `"linear"`.

For example, this code uses linear interpolation to replace the NaN values in a 1000-element numeric vector. The code is about 2.2x faster than in the previous release.

```
function timingTest
A = rand(1000,1);
idx = randperm(1000,50);
A(idx) = NaN;
for i = 1:2e4
    F = fillmissing(A,"linear");
end
end
```

The approximate execution times are:

**R2024b:** 0.84 s

**R2025a:** 0.38 s

The code was timed on a Windows 11, AMD EPYC 74F3 24-Core Processor @ 3.19 GHz test system using the `timeit` function.

```
timeit(@timingTest)
```

## **groupsummary Function: Improved performance when applying multiple computation methods**

The `groupsummary` function shows improved performance when you specify at least two of the `"sum"`, `"mean"`, `"min"`, or `"max"` computation methods, especially when the number of data points per group is small.

For example, this code computes the sum and maximum for a table with one data variable and one grouping variable. The code is about 6.6x faster than in the previous release.

```
function timingTest
data = rand(1000,1);
groups = categorical(randi(500,[1000 1]));
for i = 1:1200
    groupsummary(data,groups,["sum" "max"]);
end
end
```

The approximate execution times are:

**R2024b:** 2.11 s

**R2025a:** 0.32 s

The code was timed on a Windows 11, AMD EPYC 74F3 24-Core Processor @ 3.19 GHz test system using the `timeit` function.

```
timeit(@timingTest)
```

## uifigure Function: Improved performance when creating UI figure

Creating a figure using the `uifigure` function has improved performance. This improvement affects App Designer apps because App Designer uses the `uifigure` function to create the figure for an app. The performance improvement is most noticeable when you create a single figure outside of a loop.

For example, create a single UI figure. This code is about 5.7x faster than in the previous release.

```
function timingFigure
tic
f = uifigure;
drawnow
toc
end
```

The approximate execution times are:

**R2024b:** 2.0 s

**R2025a:** 0.35 s

The code was timed on a Windows 11, Intel Xeon® CPU W-2133 @ 3.60 GHz test system by calling the `timingFigure` function.

## Property Inspector: Improved performance when opening for the first time

The Property Inspector shows improved performance when opening for the first time in a MATLAB session. The delay between clicking **More Properties** or calling `inspect` and the Property Inspector being ready is reduced.

For example, open the Property Inspector for the first time in a MATLAB session. You can use the Property Inspector about 2.3x sooner than in the previous release.

```
axes
tic
inspect
toc
```

The approximate execution times are:

**R2024b:** 4.65 s

**R2025a:** 2.00 s

The code was timed on a Windows 11, AMD EPYC 74F3 24-Core Processor @ 3.19 GHz test system.

## **Axes Toolbar: Improved performance for single Cartesian axes when opening for the first time**

The axes toolbar shows improved performance for a single set of Cartesian axes when opening for the first time in a MATLAB session. The delay between pausing on the axes and the axes toolbar being visible is reduced.

For example, create a set of Cartesian axes by using the `axes` command. When you pause on the set of axes for the first time in a MATLAB session, the axes toolbar becomes visible about 2x faster than in the previous release.

The approximate rendering times are:

**R2024b:** 1.8 s

**R2025a:** 0.9 s

The interaction was timed on a Windows 11, AMD EPYC 74F3 24-Core Processor @ 3.19 GHz test system by pausing on a set of Cartesian axes.

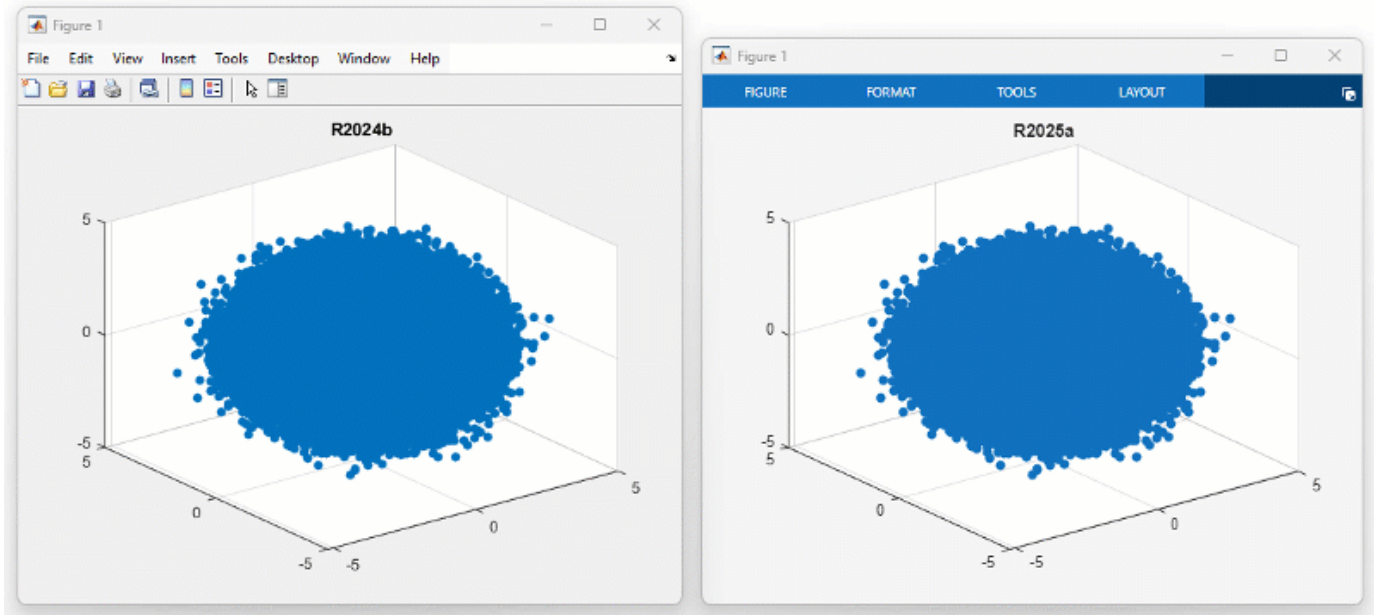
## **Plot Interactions: Improved responsiveness when interacting with scatter plots that have circle markers**

Scatter plots that display circle markers show improved responsiveness when you perform interactions, such as panning, zooming, and rotating. The improvement is more noticeable when you plot more than 25,000 points and your system has a GPU.

For example, on a Windows 11, Intel Xeon CPU W-2133 @ 3.60 GHz test system that has an NVIDIA Quadro® P600 GPU, if you run this code and then drag to rotate the plot, the rotation is smoother and follows your cursor more closely in R2025a than in the previous release.

```
f = figure;  
ax = axes(f);  
x = randn(1,2000000);  
y = randn(1,2000000);  
z = randn(1,2000000);  
scatter3(ax,x,y,z,"filled")
```



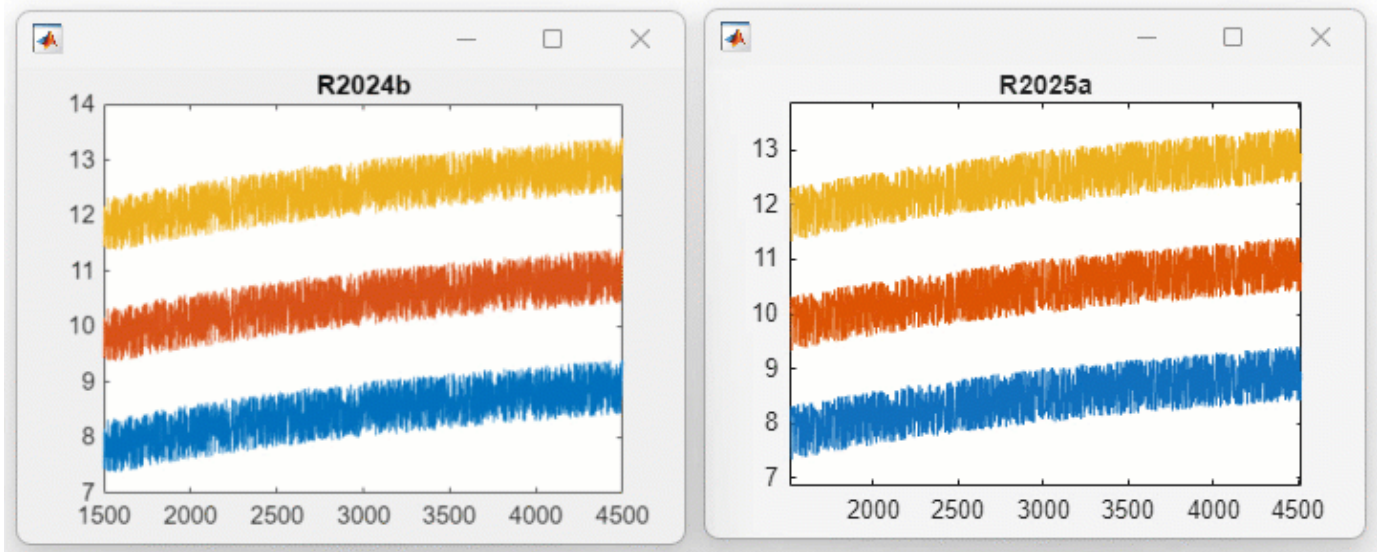


## Plot Interactions: Improved responsiveness when panning or zooming within line plots

Line plots in apps or in figures created with the `uifigure` function show improved responsiveness when you pan or zoom within the axes. If you plot a large number of points and then pan within the axes, the content that was previously outside the boundaries of the axes comes into view more quickly in R2025a than in the previous release. Before R2025a, gaps in the content appeared as you panned or zoomed.

For example, on a Windows 11, Intel Xeon CPU W-2133 @ 3.60 GHz test system, if you run this code and then pan within the axes, the content updates immediately without any gaps.

```
f = uifigure;
ax = axes(f);
x = 1:250000;
y = [log(x') log(x')+2 log(x')+4] + rand(250000,1);
plot(ax,x,y)
xlim(ax,[1500 4500])
```



If you create the figure with the `figure` function, the content that was previously outside the boundaries of the axes comes into view quickly (as in the previous release), but the interaction follows the cursor more closely.

## Animations in Apps and MATLAB Online: Improved performance for animations in loops

Animations that involve updating plot data in a loop and using the `drawnow` command show improved performance. You can observe the improvement if you run the animations in any of these contexts:

- In an app created in App Designer
- In a figure created with the `uifigure` function
- In any type of figure created in MATLAB Online

For example, create a figure using the `uifigure` function and display a line animation. This code is about 5.2x faster than in the previous release.

```
f = uifigure;
ax = axes(f);
p = animatedline(ax);
xlim(ax,[0 10])
ylim(ax,[-1 1])
x = linspace(0,10,500);
drawnow
tic
for i = 1:500
    y = sin(x);
    addpoints(p,x(i),y(i))
    % Redraw figure after adding a point
    drawnow
end
toc
```

The approximate execution times are:

**R2024b:** 15.61 s

**R2025a:** 2.98 s

The code was timed on a Windows 11, Intel Xeon CPU W-2133 @ 3.60 GHz test system.

## Plots in Apps: Improved performance when creating multiple legends

Creating multiple legends in an app created in App Designer or using the `uifigure` function shows improved performance. For example, create 16 plots in a 4-by-4 tiled chart layout and add a legend to each plot in a separate loop. This code is about 1.7x faster than in the previous release.

```
f = uifigure;
% Create 16 plots in a 4-by-4 tiled chart layout
t = tiledlayout(f,4,4);
for i = 1:16
    ax = nexttile(t);
    plot(ax,magic(10));
end
drawnow
tic
for i = 1:16
    % Create legend in the ith axes
    legend(nexttile(t,i))
    drawnow
end
toc
```

The approximate execution times are:

**R2024b:** 9.45 s

**R2025a:** 5.66 s

The code was timed on a Windows 11, Intel Xeon CPU W-2133 @ 3.60 GHz test system.

## Operations on Windows 11 Platforms with Multiple Processor Groups: Improved performance

On machines running Windows 11 (or later) or Windows Server 2022 (or later) with multiple processor groups, MATLAB can now use cores across all existing processor groups. Previously, MATLAB used only the cores in the processor group running MATLAB. Because of this change, computationally expensive operations can show improved performance.

For example, this code performs matrix multiplication with two 15,000-by-15,000 matrices. The code is about 1.6x faster than in the previous release.

```
function t = timingTest
A = ones(15000);
B = ones(15000);

f = @( ) A*B;
t = timeit(f);
end
```

The approximate execution times are:

**R2024b:** 5.40 s

**R2025a:** 3.42 s

The code was timed on a Windows 11, AMD Ryzen Threadripper PRO 7995WX 96-Core Processor @ 2.5 GHz test system by calling the `timingTest` function. On this test system, the default maximum number of computational threads in R2024b is 64, while the default maximum number of computational threads in R2025a is 96.

## Software Development Tools

### ★ Project Panel: Manage project environment and view issues using improved interface

Starting in R2025a, the project user interface is located in the Project panel, where you can access simplified workflows and new features.

- Project settings – Use a single interface to manage the project environment, path setup, labels, and custom tasks. For more information, see [Manage Project Settings](#), [Path](#), [Labels](#), and [Startup and Shutdown Tasks](#).
- Project Issues panel – Easily investigate and fix issues detected by project startup, shutdown, and integrity checks. For more information, see [Run Project Checks](#).
- Interactive tutorial – Step through a minimal project setup. For more information, see [Set Up Project](#).

This table describes how to perform common project tasks.

| Task                                                                             | R2024b and Earlier                                                                          | Starting in R2025a                                                                                                                                                                                                                                                                                                          |
|----------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Create a project from an existing folder.                                        | On the <b>Home</b> tab, select <b>New &gt; Project &gt; From Folder</b> .                   | On the <b>Home</b> tab, select <b>New &gt; Project</b> . Then, select an existing folder. If the folder you select is empty, the software creates a blank project.                                                                                                                                                          |
| Create an empty project.                                                         | On the <b>Home</b> tab, select <b>New &gt; Project &gt; Blank Project</b> .                 |                                                                                                                                                                                                                                                                                                                             |
| Clone or check out a remote repository and create a project in the working copy. | On the <b>Home</b> tab, select <b>New &gt; Project &gt; From Git</b> or <b>From SVN</b> .   | To create a project from a non-project source control repository: <ol style="list-style-type: none"> <li>1 On the <b>Home</b> tab, select <b>New &gt; Git Clone</b> or <b>SVN Checkout</b>.</li> <li>2 Create a project from an existing working folder. On the <b>Home</b> tab, select <b>New &gt; Project</b>.</li> </ol> |
| Create a project from a Simulink® template.                                      | On the <b>Home</b> tab, select <b>New &gt; Project &gt; From Simulink Template</b> .        | Use the Simulink Start Page to create a project from a Simulink template. For more information, see <a href="#">Create New Project Using Templates (Simulink)</a> .                                                                                                                                                         |
| Add files to a project.                                                          | On the <b>Project</b> tab, in the <b>Tools</b> gallery, click <b>Add Files</b> .            | In the Project panel, right-click a file or a folder and select <b>Add to Project</b> or <b>Add Folder to Project (Including Contents)</b> .                                                                                                                                                                                |
| Check for project files with unsaved changes.                                    | On the <b>Project</b> tab, in the <b>Tools</b> gallery, click <b>Show Unsaved Changes</b> . | On the <b>Project</b> tab, in the <b>Environment</b> section, select <b>Project Issues &gt; Unsaved Changes</b> .                                                                                                                                                                                                           |

| Task                    | R2024b and Earlier                                                                      | Starting in R2025a                                                                                                                                                                                  |
|-------------------------|-----------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Run project checks.     | On the <b>Project</b> tab, in the <b>Tools</b> gallery, click <b>Check Project</b> .    | On the <b>Project</b> tab, in the <b>Environment</b> section, click <b>Project Issues</b> . In the <b>Checks</b> tab of the Project Issues panel, click <b>Run Checks</b> .                         |
| Share project by email. | On the <b>Project</b> tab, in the <b>File</b> section, select <b>Share &gt; Email</b> . | On the <b>Project</b> tab, in the <b>File</b> section, select <b>Share &gt; Archive</b> . Then, attach the project archive to an email.<br><br>For more ways to share projects, see Share Projects. |

## Project References: Componentize large projects with improved usability

Organize a project into smaller projects with improved usability. Improvements to referenced projects workflows include:

- Accessing shortcuts in all referenced projects from the top-level project
- Automatic updates when renaming, deleting, or removing files in the entire project hierarchy
- Identifying issues in your project hierarchy, such as circular or missing dependencies
- Support for moving files and folders between projects in the same project hierarchy

For more information, see Componentize Large Projects.

## Project Comparison: Compare MATLAB projects

Starting in R2025a, when you compare folders, MATLAB detects whether they are project root folders. MATLAB looks for and compares the project definition files stored in the **resources** folder. Project definition files contain information about the project path, project settings, shortcuts, labels, and referenced projects. For more information, see Compare MATLAB Projects.

## Dependency Analyzer: View project hierarchy and issues

You can use the **Project Hierarchy** view in the Dependency Analyzer to visualize the relationships between the top-level project and the referenced projects. The hierarchy graph also flags issues about missing projects and circular dependencies. For more information, see Explore the Dependency Graph, Views, and Filters.

## Dependency Analyzer: Analyze dependencies in referenced projects

You can now enable dependency analysis inside referenced projects. To specify the scope of the analysis to include referenced projects, in the Dependency Analyzer toolstrip, select **Analyze > Referenced Projects**. For more information, see Dependency Analyzer Scope and Limitations.

### Project API: List issues on project startup and shutdown

You can list all issues that occur when a project opens and closes using the `listStartupIssues` and `listShutdownIssues` functions.

### Project API: Add all folders and files to project

You can now programmatically add all folders, subfolders, and files to a project in one step using the `addFolderIncludingChildFiles` function.

### Project API: Analyze file dependencies in referenced projects

You can now include files in project references when you run a dependency analysis using the `listImpactedFiles`, `listRequiredFiles`, and `updateDependencies` functions.

### Project API: Process multiple project files simultaneously

Starting in R2025a, project functions support processing multiple files, folders, labels, and categories in one function call. This change reduces the need to make multiple function calls and create `for`-loops.

For example, this table shows how to update your code to add multiple files to a project at once.

| R2024b and Earlier                                                                                                              | Starting in R2025a                                                                           |
|---------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------|
| <pre>proj = currentProject; Files = ["File1", "File2", "File3"]; for i = 1:length(Files)     addFile(proj, Files{i}); end</pre> | <pre>proj = currentProject; Files = ["File1", "File2", "File3"]; addFile(proj, Files);</pre> |

The functions that support bulk processing are: `addFile`, `addFolderIncludingChildFiles`, `removeFile`, `addPath`, `removePath`, `addStartupFile`, `removeStartupFile`, `addShutdownFile`, `removeShutdownFile`, `addShortcut`, `removeShortcut`, `createLabel`, `addLabel`, `findLabel`, `removeLabel`, `createCategory`, `findCategory`, and `removeCategory`.

### Project API: Improved performance when interacting with project programmatically

When you interact with a MATLAB project programmatically, the project functions and operations perform significantly faster. For example, this code saves the path of 731 project files in a variable. The code is about 7.3x faster than in the previous release.

```
function timingTest
    proj = currentProject;
    allProjectFiles = proj.Files;
    filesPath = [allProjectFiles.Path];
end
```

The approximate execution times are:

**R2024b:** 0.921 s

**R2025a:** 0.126 s

The code was timed on a Windows 11, AMD EPYC 9474F 48-Core Processor @ 3.60 GHz test system using the `timeit` function.

```
timeit(@timingTest)
```

## Code Analyzer: Configure Custom Naming Conventions for MATLAB Identifiers

Configure the Code Analyzer to display a custom check when MATLAB identifiers do not adhere to specified naming conventions. For more information on configuring custom Code Analyzer checks, see [Configure Code Analyzer](#).

You can configure required or disallowed prefixes and suffixes for identifiers. A message will be displayed if an identifier does not follow the naming convention. Specified phrases can be disallowed and a message will be displayed if they are found anywhere in the identifier name. For example, you could require that property names start with the prefix `prop` or `property`.

```
"naming": {
    "property" : {
        "requiredPrefix" : ["prop","property"]
    }
}
```

You can configure casing so that identifiers conform to one or more casing standards by specifying a `"casing"` property containing a string array where each element is a casing standard. If more than one casing is specified, then identifiers must satisfy at least one specified casing. For example, you could require that function names use `lowerCamelCase` and classes use `UpperCamelCase`.

```
"naming": {
    "function" : {
        "casing" : "lowerCamelCase"
    },
    "class" : {
        "casing" : "UpperCamelCase"
    }
}
```

You can create a custom naming convention check by specifying a regular expression. A message will be displayed if the identifier name does not fully match the specified regular expression. Required prefixes and suffixes are not excluded from this check. Each identifier type can have only a single regular expression check. For example, you could require that variable names use only lowercase letters.

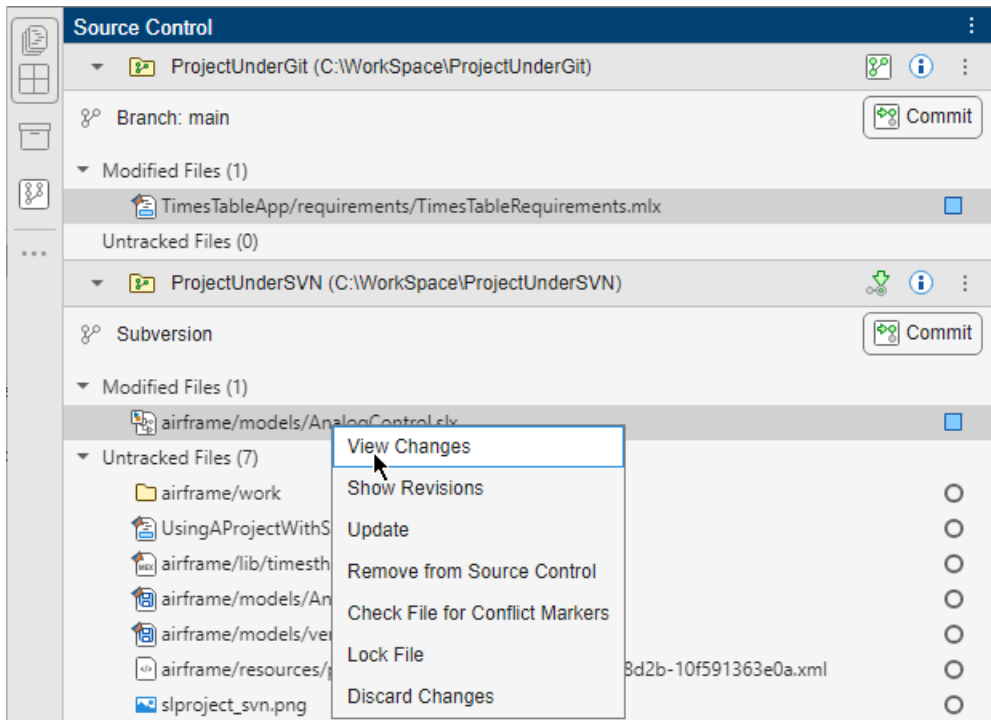
```
"naming": {
    "variable" : {
        "regularExpression" : "^[a-z]+$"
    }
}
```

## ★ Source Control Panel: Work with multiple folders and projects under Git and SVN source control

Use the Source Control panel to interact with folders and projects under Git™ and SVN source control, manage modified and untracked files, and perform source control operations. If the Source



Control icon is not in the sidebar, click the Open more panels button \*\*\* and select the Source Control panel.



For more information, see Source Control Integration in MATLAB.

## Source Control: Check for unsaved files before commit

Starting in R2025a, you can configure MATLAB to prompt you about unsaved changes before performing source control actions such as commit, merge, SVN update, and Git branch switch. On the **Home** tab, in the **Environment** section, click **Settings**. Select **MATLAB > Source Control**. Then, select **Check for unsaved files before source control operations**.

## Git Source Control: View line-by-line revision history for entire MATLAB file

You can view the line-by-line revision history for an entire MATLAB file under Git source control using Blame View directly within MATLAB. For more information, see Annotate Lines in MATLAB Editor Using Git History.

## Git Source Control: Create shallow clone of Git repository

You can perform a shallow clone by specifying the number of the latest commits you want to clone. For more information, see Clone Git Repository in MATLAB.

## **Git Source Control: Share local repository to GitHub**

You can share a local Git repository to GitHub® directly within MATLAB. For more information, see [Share Git Repository to Remote](#).

## **Git Source Control: Ignore files interactively**

You can add files and folders to the `.gitignore` file interactively from within MATLAB. For more information, see [Create Local Git Repository in MATLAB](#).

## **Git Source Control: Cherry-pick or squash Git commits, and rebase Git branches**

Starting in R2025a, the MATLAB Git integration supports these workflows:

- Cherry-pick Git commits from within MATLAB. For more information, see [Cherry-Pick Git Commits](#).
- Squash Git commits directly within MATLAB. For more information, see [Squash Git Commits in MATLAB](#).
- Rebasing a Git branch onto another branch directly within MATLAB. For more information, see [Rebase Git Branch in MATLAB](#).

## **Git Source Control: Create, merge, and manage Git branches and remotes with improved usability**

For folders and projects under Git source control, you can use the Branch Manager to manage multiple local and remote repositories, find and compare commits, and create, delete, and merge branches.

## **Git Source Control: Work with submodules in MATLAB with improved usability**

You can work with and manage multiple Git repositories at a time using the Source Control panel and the Branch Manager. Starting in R2025a, when you pull changes in the main repository, MATLAB updates the submodules recursively. For more information, see [Work with Git Submodules in MATLAB](#).

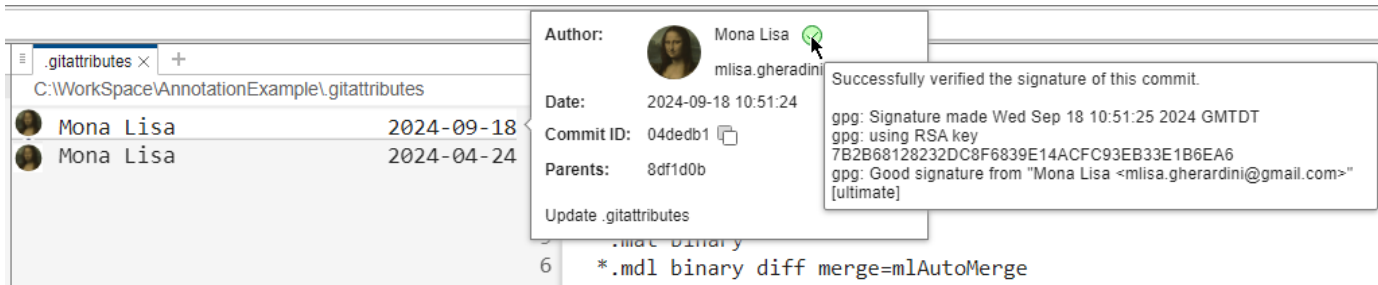
## **Git Source Control: Check out Git branch to separate folder using worktrees in MATLAB**

You can check out a branch to a separate folder using Git worktrees directly within MATLAB. For more information, see [Create and Manage Git Worktrees](#).

## **Git Source Control: Use avatars and verify commit signatures in MATLAB**

Starting in R2025a, you can enable MATLAB to generate local avatars or retrieve globally recognized avatars from the web.

When you enable commit signing, MATLAB now verifies commit signatures. A green verification icon  appears next to your avatar and username in both the Branch Manager and the line annotations in the MATLAB Editor.



For more information, see [Configure Git Settings](#).

## Git Source Control: MATLAB supports Git LFS process filter

Starting in R2025a, if you configure a process for an LFS filter, MATLAB launches and communicates with the process for multiple files at the same time, rather than running a new process for every file separately. This speeds up source control actions in MATLAB, such as adding files, committing files, and getting file statuses.

To configure a process for the LFS filter, install Git LFS and run `!git lfs install`. For more information, see [Configure Git to Use Git LFS](#).

Installing Git LFS defines this information in the `.gitconfig` file.

```
[filter "lfs"]
  clean = git-lfs clean -- %f
  smudge = git-lfs smudge -- %f
  process = git-lfs filter-process
  required = true
```

## Git API: List modified files between commits or branches and save file revisions to disk

Starting in R2025a, the MATLAB Git integration supports more continuous integration workflows:

- List the modified files between two Git commits or branches using the `diffCommits` function.
- Save a Git revision of a file to disk using the `saveFileRevision` function.

## Git API: Add and remove remotes

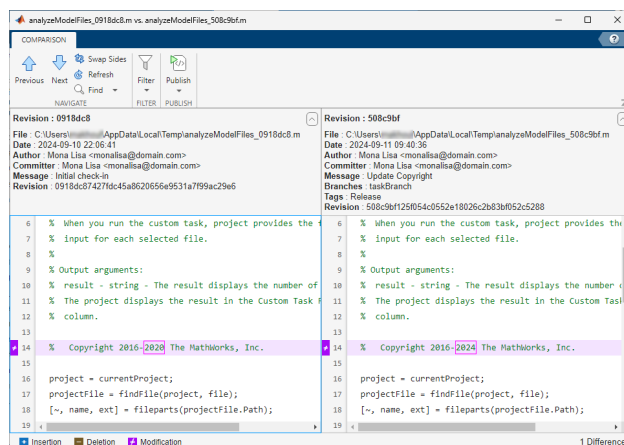
You can add a remote to a Git repository using the `addRemote` function. To delete a remote, use the `removeRemote` function.

## SVN Source Control: Remember credentials for SVN repositories

Starting in R2025a, MATLAB SVN integration remembers usernames and tokens when you interact with SVN repositories. To adjust or disable credential management, use the SVN source control settings. For more information, see [Configure SVN Settings](#).

## Comparison and Merge Tools: View tag and branch details for files under source control

Starting in R2025a, when you compare or merge files under source control, the comparison and merge tools include source control information such as the branch name and tags in the file details.



## Build Automation: Create custom reusable tasks

You can create custom task classes by subclassing the `matlab.buildtool.Task` class and specifying task actions, inputs, and outputs using the `TaskAction`, `TaskInput`, and `TaskOutput` attributes. For more information, see [Create Custom Reusable Tasks](#).

## Build Automation: Run tasks interactively from Editor or MATLAB project

If your build file named `buildfile.m` is open in the MATLAB Editor or if your MATLAB project contains a build file named `buildfile.m` in its root folder, then you can interactively run the default tasks or a specific task in the build file from the MATLAB Toolstrip. You can also customize your build run by selecting build options from the toolstrip. For more information, see [Run Build from Toolstrip](#).

## ▲ Build Automation: Expanded support for MATLAB projects

The build tool provides expanded support for MATLAB projects. If your build file is in a MATLAB project folder, including any subfolder:

- The `matlab.buildtool.Plan.load` method opens the project if it is not open.
- The `buildtool` command and the `run` method open the project if it is not open. In previous releases, they open the project only if the build file is in the project root folder.

- The `Project` property of the `matlab.buildtool.Plan` class provides programmatic access to the project.

## ⚠ Compatibility Considerations

Starting in R2025a, the `buildtool` command opens the project before running the main function of the build file and leaves the project open after running the build. In previous releases, the command opens the project after running the main function of the build file and closes the project after running the build.

## Build Automation: Display why tasks run

If you specify the `-verbosity` option of the `buildtool` command to display build output at the `matlab.automation.Verbosity.Detailed` level or above, then the build tool includes the reason for running each task. For instance, when rerunning a task in an incremental build, the build tool displays a summary of changes to the inputs and outputs of the task since its last successful run. For an example, see [Create and Run Tasks with Inputs and Outputs](#).

## Build Automation: Create group of `MexTask` instances using `forEachFile` static method

You can create a group of tasks that build MEX files by using the new `matlab.buildtool.tasks.MexTask.forEachFile` method.

## Build Automation: Filter test suite using selectors

You can select a subset of tests to run by using the `Selector` property of the `matlab.buildtool.tasks.TestTask` class. For example, create a task that runs the tests in the `feature1` subfolder of your current folder that have the `"Unit"` tag.

```
import matlab.buildtool.tasks.TestTask
import matlab.unittest.selectors.*

task = TestTask(Selector= ...
    HasBaseFolder(fullfile(pwd,"feature1")) & HasTag("Unit"));
```

## Unit Testing Framework: Rerun tests from most recent test run using Test Browser

You can rerun the tests from your most recent test run using the Test Browser app. Click the drop-down arrow ▼ to the right of the Run current suite button



select **Rerun Tests**.

on the toolbar and then

## Unit Testing Framework: Identify most recent test results in Test Browser

You can distinguish the test results produced in the most recent test run from those produced in earlier test runs. In the Test Browser panel, the statuses of tests that ran before the most recent test

run appear dimmed, allowing you to identify and manage outdated test data. For an example, see [Run Parameterized Tests](#).

## Unit Testing Framework: Control whether to use Test Browser from toolstrip

When you run a function-based or class-based test file using the **Run Tests** section on the toolstrip, you can control whether to run the tests and display results using the Test Browser app. To access this test option, with your test file open, on the **Editor** or **Live Editor** tab, select **Run Tests > Customize Test Run**, and then select or clear the **Use Test Browser** option.

If you choose not to use the test browser by clearing the **Use Test Browser** option, then your tests run in the Command Window instead. Your selection persists across MATLAB sessions.

## Unit Testing Framework: Create default test runner using withDefaultPlugins static method

You can use the new `matlab.unittest.TestRunner.withDefaultPlugins` method to create a test runner configured with default plugins. These plugins are used by the testing framework by default when running tests.

## ⚠️Toolbox Packaging: Create and package toolboxes using projects

Starting in R2025a, toolbox packaging is integrated with projects. You can use tasks within a project to configure and package your toolbox.

To package your toolbox files, create a project from the folder containing your files. Then, with the project open, go to the **Project** tab, and in the **Tools** section, click **Package Toolbox**. Configure your toolbox using the toolbox task and then click **Package Toolbox** at the top-right corner of the task to create your toolbox installation file.

If you have a toolbox created in a release before R2025a as well as its associated `.prj` file, MATLAB automatically upgrades your toolbox to the project workflow.

For more information, see [Create and Share Toolboxes](#).

## ⚠️Compatibility Considerations

In the new project workflow, toolbox image files must now be contained within the project folder. In addition, toolbox files must be contained within your selected toolbox folder.

## Extension Points: Extend the MATLAB desktop using extension points

Extend various functionalities of the MATLAB desktop using extension points. For example, you can use extension points to add items to the quick access toolbar or to the Files panel context menu. To use extension points, create a JSON file named `extensions.json`, place it in a folder named `resources`, and add a set of JSON declarations to the file. Then, to enable your customizations, add the folder containing the `resources` folder with the `extensions.json` file to the MATLAB path.

For more information, see [Extend MATLAB Using Extension Points](#).

## ▲ Functionality being removed or changed

### Referenced projects no longer support checkpoints

*Behavior change*

You can no longer use checkpoints in referenced projects. To track changes in referenced projects and compare changes in referenced project files to a specific version, use source control instead.

### gitinit automatically adds project files to source control

*Behavior change*

Starting in R2025a, if you use a MATLAB project to manage the files in the folder, the `gitinit` function automatically adds all files with the status **In project** to source control.

### SVN commands are no longer supported via bang operator in MATLAB on Linux

*Behavior change*

Starting in R2025a, on Linux systems, you cannot call `svn` commands from within MATLAB using the bang symbol `!`, for example, `!svn`.

### Test Browser retains tests when a specific test procedure is run from toolstrip

*Behavior change*

If you run a specific test procedure using the **Run Current Test** button on the MATLAB Toolstrip, the Test Browser app retains the tests already imported from the corresponding test file. In previous releases, the test browser removes the tests imported from the test file before running the specified test procedure.

### Single Graphics task replaces 2-D and 3-D tasks in bench function

*Behavior change*

In the `bench` function, the existing 2-D and 3-D tasks have been replaced by a single Graphics benchmarking task. The new Graphics task is a better benchmark for the updated graphics system, which uses more modern architecture, and represents a wider range of graphics workflows.

This table compares the R2024b and R2025a execution times using a Windows 11 Enterprise, Intel Core i9-12900 @ 2.4GHz test system. The measured values are expressed in seconds.

| Task     | Execution Times of Tasks in R2024b | Execution Times of Tasks in R2025a |
|----------|------------------------------------|------------------------------------|
| LU       | 0.2384                             | 0.2408                             |
| FFT      | 0.1538                             | 0.1535                             |
| ODE      | 0.0830                             | 0.0858                             |
| Sparse   | 0.2112                             | 0.2090                             |
| 2-D      | 0.1485                             | Not applicable                     |
| 3-D      | 0.2100                             | Not applicable                     |
| Graphics | Not applicable                     | 0.1342                             |

## External Language Interfaces

### Call C++ from MATLAB: Pass MATLAB structs to and from C++ functions

You can pass MATLAB structures directly to and from C++ functions. To configure a function to accept structure arguments, set `MLTYPE` to "struct" for input or output arguments in the library definition file. For information about configuring a C++ struct parameter, see [Struct Types](#).

When you set the type to `struct`, the help for the C++ member shows the type as a `struct`. For example, display the help text for the `MyStruct` class in library `libname`.

```
help clib.libname.MyStruct
```

```
MyStruct - clib.libname.MyStruct Representation of C++ class MyStruct.
```

```
struct with fields:
ID                5 element vector char
amount           single
x                double
```

For more information and examples, see [Pass struct Parameter](#).

### Call C++ from MATLAB: Pass environment variables in out-of-process mode

When your MATLAB interface to a C++ library uses out-of-process execution mode, you can pass system environment variables to the process hosting the interface by using the `clibConfiguration` function with the `OutOfProcessEnvironmentVariables` name-value argument. To use this argument, you must also specify the `ExecutionMode` name-value argument as "outofprocess".

### Call C++ from MATLAB: Stop out-of-process execution using Ctrl+C

Pressing **Ctrl+C** during C++ library interface out-of-process execution stops the execution of a MATLAB command. For more information, see [Load C++ Library In-Process or Out-of-Process](#).

### ▲ Call MATLAB from C++: Create and use single-precision sparse matrices

These MATLAB Data API for C++ templated classes, which already support double-precision sparse matrices, now support single-precision sparse matrices:

- `matlab::data::SparseArray<T>`
- `matlab::data::Reference<SparseArray<T>>`

### ▲ Compatibility Considerations

If you save a single-precision sparse matrix to a MAT file, you can load it into an earlier release of MATLAB. However, how the data is loaded depends on the release of MATLAB:

- MATLAB R2021b through R2024b — MATLAB warns that the data has an unsupported data type and loads it as a double-precision sparse matrix.



- MATLAB R2021a and earlier — MATLAB loads the data as a single-precision sparse matrix, but you cannot use the matrix because its data type is unsupported. Attempting to use the matrix can result in undefined behavior.

## **⚠ C MAT File API: Maximum length of variable names increased to 2048 characters**

The maximum allowed identifier length defined by the macro `mxMAXNAM` has increased to 2048 characters. This change allows variables, functions, classes, and many other entities to have names of up to 2048 characters. Using longer identifiers might negatively impact performance and memory. Operating systems might have maximum filename lengths that are shorter than this limit.

## **⚠ Compatibility Considerations**

Previously, the maximum length of identifiers was 63 characters. Recompile MEX files and engine or MAT file applications that use the constant `mxMAXNAM`.

## **Call MATLAB from Python: Automatically convert between Python and MATLAB datetime and duration types**

When working in Python®, you can now return MATLAB `datetime` and `duration` values directly to Python. If the MATLAB `datetime` or `duration` value is scalar, MATLAB Engine API for Python automatically converts the `datetime` or `duration` value to a Python `datetime.datetime` or `datetime.timedelta` object. If you pass a MATLAB `datetime` or `duration` array to Python and the NumPy module is available in the Python environment, the MATLAB engine automatically converts the `datetime` or `duration` array to a Python `numpy.datetime64` or `numpy.timedelta64` array.

If you pass a Python `datetime.datetime` or `datetime.timedelta` object to MATLAB, the MATLAB engine automatically converts it to a MATLAB `datetime` or `duration` value. The engine also automatically converts Python `numpy.datetime64` and `numpy.timedelta64` objects and arrays to MATLAB `datetime` and `duration` values and arrays.

If the NumPy module is not available when you return a MATLAB `datetime` or `duration` array to Python, the MATLAB engine converts the array to a Python object of type `matlab.object`.

For more information, see [How MATLAB Handles Datetime and Duration Types in Python](#).

## **Call MATLAB from Python: Automatically convert between MATLAB tables, timetables, and Python Pandas DataFrames**

When working in Python, you can now return MATLAB tables or timetables directly to Python. MATLAB Engine API for Python automatically converts the table or timetable to a Python Pandas DataFrame. The MATLAB engine also automatically converts a Python Pandas DataFrame to a MATLAB table or timetable when you pass it from Python to MATLAB.

If the Python Pandas module is not available when you pass a MATLAB table or timetable to Python, the MATLAB engine converts the table or timetable to a Python object of type `matlab.object`.

For more information, see [Use MATLAB Tables and Timetables in Python](#).

## Call Python from MATLAB: Switch execution modes without restarting MATLAB

When working with Python in MATLAB, you can now change the execution mode of the Python interpreter from `OutOfProcess` to `InProcess` after terminating the process associated with the Python interpreter. To change the execution mode, call `pyenv(ExecutionMode="InProcess")`. Previously, if you wanted to change the execution mode from `OutOfProcess` to `InProcess`, you had to restart MATLAB before calling `pyenv` with the new version information.

For more information, see `terminate`.

## Call .NET from MATLAB: Save .NET objects to MAT files

You can save (serialize) .NET objects to a MAT file. For more information, see `Save and Load .NET Objects in MAT Files`.

## Call .NET from MATLAB: Select specific numerical version of .NET

For a .NET Runtime environment specified as "core", you can select a specific numerical version using the `dotnetenv` function `Version` name-value argument. Use the `Frameworks` name-value argument to choose a target framework. For example:

```
ne = dotnetenv("core",Version="8",Frameworks="Microsoft.WindowsDesktop.App");
```

## Web Services: Keep sensitive information encrypted when using `weboptions` and `matlab.net.http.Credentials` objects

You can avoid saving credentials and other sensitive information as plain text in properties of `weboptions` and `matlab.net.http.Credentials` objects by using a `secretID` object. First, save sensitive information to your MATLAB vault as a secret using the `setSecret` function. Then when you set an object property, such as the `Username` and `Password` properties of a `matlab.net.http.Credentials` object, specify a `secretID` object to keep your sensitive information encrypted.

## MATLAB Packages: Java files supported in packages

Starting in R2025a, you can add Java code to MATLAB Packages. You can specify the language of package folders by setting the `languages` property of the `matlab.mpm.PackageFolder` object. When package is installed, all Java resources in a folder designated as `java` are added to both the dynamic and static class path.

If the `languages` property of a package folder is not designated as `matlab`, MATLAB files in that folder are ignored. If a package folder is not designated as `java`, JAR files in that folder are ignored.

## **Functionality being removed or changed**

### **Python interface converts MATLAB arrays to Python NumPy arrays when passed as input to Python functions**

*Behavior change*

When you pass a MATLAB array as input to a Python function and the NumPy module is available in the Python environment, Python Interface now automatically converts the array to a Python NumPy array.

If the NumPy module is not available when you pass a MATLAB array as input to a Python function, Python Interface now handles vector input as it does matrix input—Python Interface converts these inputs to Python memoryview objects.

Previously, when you passed a MATLAB vector to a Python function, Python Interface converted the vector to a Python `array.array` object.

For more information, see [Pass MATLAB Data to Python](#).

### **MATLAB truncates values of unsigned integer types when passing them to C shared library function**

#### *Behavior change*

Starting in R2025a, MATLAB truncates the value for unsigned integer types when you pass the value to a function in a C shared library loaded by the `loadlibrary` function. The behavior for unsigned integer types is now consistent with that of signed integer types. Previously, when you passed a value of an unsigned integer type and the value was out-of-range for the type, MATLAB threw an error.

| Example Code                                                                                                                                                                                                                               | Output in R2024b and Earlier                                                                                                                                                                                                                   | Output Starting in R2025a                                                                          |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------|
| This C code from a C library defines <code>value</code> as <code>short</code> and <code>unsigned short</code> types. <pre>struct S16 {     short value; % -32768 to 32767 }; struct SU16 {     unsigned short value; % 0 to 65535 };</pre> |                                                                                                                                                                                                                                                |                                                                                                    |
| This MATLAB code creates a <code>short</code> variable and assigns an out-of-range value. <pre>mls16.value = 123456; lps16 = libpointer('S16Ptr', mls16); lps16.Value</pre>                                                                | MATLAB truncates the value to 32767. <pre>ans =     struct with fields:         value: 32767</pre>                                                                                                                                             | Same behavior as R2024b.                                                                           |
| This MATLAB code creates an <code>unsigned short</code> variable and assigns an out-of-range value. <pre>mlsul16.value = 123456; lpsul16 = libpointer('SU16Ptr', mlsul16); lpsul16.Value</pre>                                             | MATLAB displays an error. <pre>Cannot convert data value for field value due to error: Value exceeds the legal range for the data type. Error in libpointer (line 21) struct with fields:     ptr: libpointer(varargin{:}); value: 65535</pre> | MATLAB truncates the value to 65535. <pre>ans =     struct with fields:         value: 65535</pre> |

### **actxcontrol, actxcontrollist, and actxcontrolselect functions have been removed**

#### *Errors*

The `actxcontrol`, `actxcontrollist`, and `actxcontrolselect` functions have been removed. If you call the functions, MATLAB returns an error. MATLAB supports COM server objects only.

## Hardware Support

### **Raspberry Pi Hardware: Support for 64-bit Debian 12 Bookworm and 64-bit Debian 11 Bullseye**

You can now use MATLAB Support Package for Raspberry Pi® Hardware with Raspberry Pi boards running on 64-bit Debian® 12 Bookworm and 64-bit Debian 11 Bullseye operating systems. This support is available only for the MATLAB desktop environment.

### **Arduino Hardware: Support for Arduino UNO R4 WiFi and Minima boards in MATLAB Online**

You can now use the `arduino` object in MATLAB Support Package for Arduino® Hardware to connect to Arduino UNO R4 WiFi and Minima boards over USB with MATLAB Online.

### **Arduino Hardware: Support for Servo Motors and Sensors with Arduino UNO R4 WiFi and Minima Boards**

You can now use MATLAB Support Package for Arduino Hardware to interface Arduino UNO R4 WiFi and Minima boards with servo motors and sensors supported by the support package. This feature is available for both MATLAB desktop and MATLAB Online environment.

### **Arduino Hardware: Enhancing voltage data acquisition capabilities with `readVoltage` function**

You can now use the `readVoltage` function in MATLAB Support Package for Arduino Hardware to acquire voltage data at a higher sampling rate. This function also allows you to acquire voltage data at the specified sample rate for a specific duration or number of samples, and return data in a particular output format. Starting this release, you can configure the `readVoltage` function using these name-value arguments: `NumSamples`, `Duration`, `SampleRate`, and `OutputFormat`.

### **⚠️ Functionality being removed or changed**

#### **Raspberry Pi Hardware: Support for Buster Raspberry Pi Linux operating system will be removed**

*Still runs*

Support for the 32-bit Buster Raspberry Pi Linux operating system will be removed in a future release for MATLAB Support Package for Raspberry Pi Hardware in the MATLAB desktop and MATLAB Online environment. You will no longer be able to use the support package to communicate with a Raspberry Pi board that runs a Buster operating system. You must upgrade the Raspberry Pi Linux operating system to 32-bit or 64-bit Bullseye or Bookworm in the MATLAB desktop environment and 32-bit Bullseye or Bookworm in the MATLAB Online environment.

# R2024b

---

**Version: 24.2**

**New Features**

**Bug Fixes**

**Compatibility Considerations**

## Environment

### ★ Help Center: View documentation in your system web browser

When you access the documentation from an installed version of MATLAB, the documentation now opens in your system web browser. Previously, the documentation opened in the Help browser. This change applies whether you view the web documentation or the documentation installed locally on your system. For more information about installing documentation for offline use, see [Install Documentation](#).

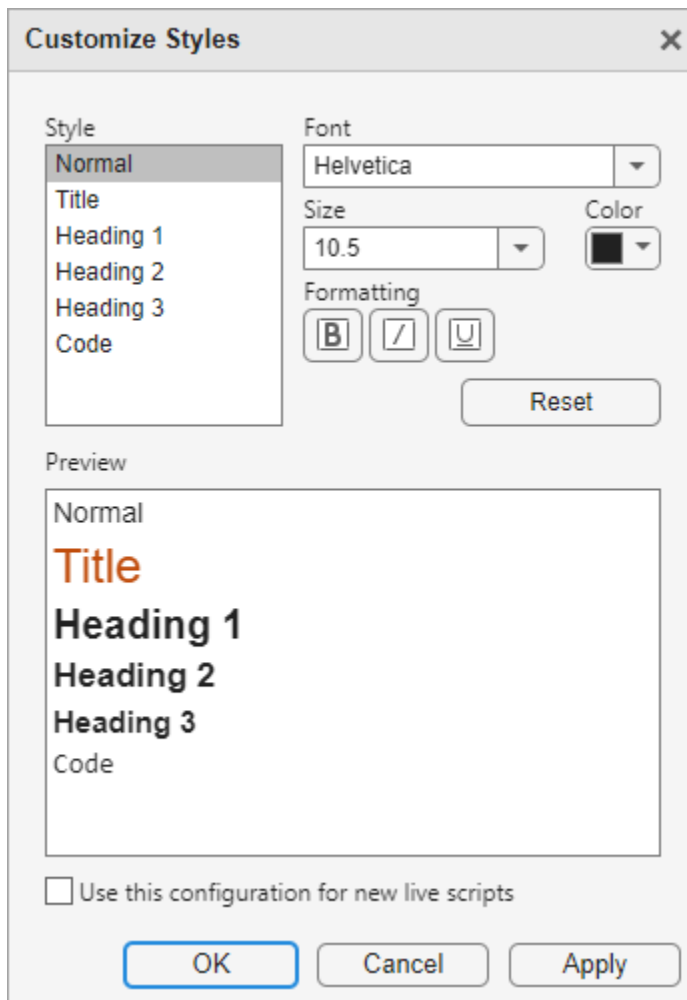
As part of this change, the Help browser has been removed. In addition, installed products are no longer indicated in the documentation. To copy an example and supporting files onto your system and open the example in MATLAB, click the **Copy Command** button next to the example in the documentation. Then, paste the copied command into the MATLAB Command Window. You also can open some examples directly in your web browser in MATLAB Online.

### ★ Live Editor Fonts: Customize font, size, color, and formatting of text and code styles

You can customize the font, size, color, and formatting of text and code styles in the Live Editor using the Customize Styles dialog box. To customize a text or code style, go to the **Live Editor** tab, and in the **Text** section, click the Customize styles button . Then, select the style that you want to customize and change the font, size, color, and formatting for the selected style. The Customize Styles dialog box shows a preview of each style.

The customized style applies only to the current document. To apply the customized style to all new live scripts as well, select the **Use this configuration for new live scripts** option.

For more information, see [Format Text](#) in the Live Editor.



## Live Editor Controls: Add range sliders to live scripts

Add a range slider to your live script to select a range interactively. To add a range slider, go to the **Live Editor** tab, and in the **Code** section, select **Control > Range Slider**.



For more information, see [Add Interactive Controls to a Live Script](#).

## Live Editor Controls: Populate drop-down list items using additional variable types

You can now populate the items in a drop-down list using values stored in a categorical array, character array, cell array, or double array. Previously, only string arrays were supported.

You also can now edit the item labels after populating the items in a drop-down list using the values stored in a variable. Previously, the item labels were read-only.

For more information, see [Link Variables to Controls](#).

## Live Editor Controls: Restore default values for all controls

You can restore the default values for all controls in a live script. To restore the default values for all controls, right-click any control in the live script and select **Restore Default Values for All Controls**.

For more information, see [Specify Default Values](#).

## help Function: Get updated help text for functions, methods, and classes

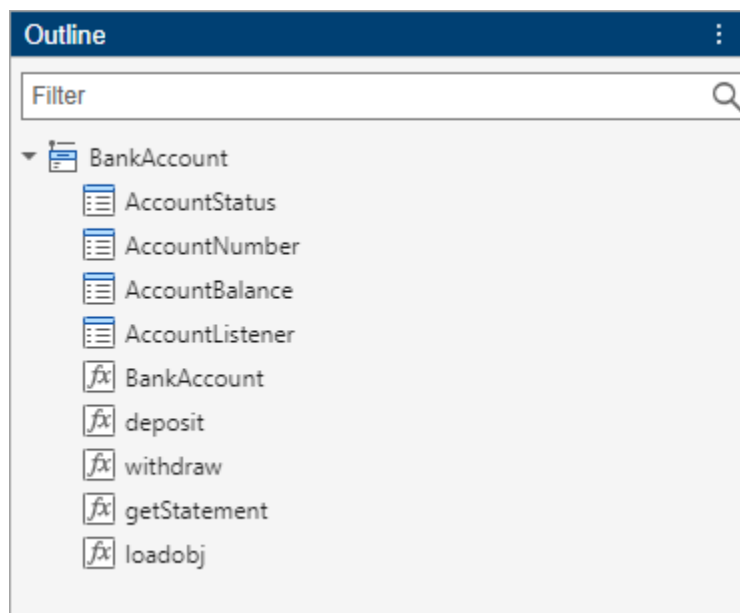
The updated help text for functions, methods, and classes provides more consistent and uniform support across all products.

Use the `help` function to display abbreviated help text in the Command Window. For more extensive help, use the `doc` function instead.

## Editor in MATLAB Online: View outline of your code using Outline panel

In the MATLAB Online Editor, you can use the Outline panel to view a high-level outline of scripts, functions, and classes. To view an outline of a file open in the Editor, go to the **View** tab, and in the **Tools** section, click **Outline**. To navigate to an area within your code file, double-click the related entry in the Outline panel.

For more information, see [Edit and Format Code](#).





## Editor TLC Support in MATLAB Online: View and edit TLC files with syntax highlighting and automatic indenting while typing

In MATLAB Online, the Editor displays Target Language Compiler (TLC) files with syntax highlighting for commands, comments, macros, and strings. In addition, the Editor automatically indents lines of code in TLC files while you type.

In MATLAB Online, you also can use preferences to configure how the Editor displays TLC files. On the **Home** tab, in the **Environment** section, click  **Preferences**. Select **MATLAB > Editor/Debugger > Other Languages > TLC**, and adjust the options as needed.

## Add-Ons in MATLAB Online: Update add-ons directly from Add-Ons panel

In MATLAB Online, you can update add-ons directly from the Add-Ons panel. To update an add-on, in the **Installed** section of the Add-Ons panel, click the **Update** button to the right of the add-on. The Add-Ons panel displays the **Update** button next to add-ons that have an update available.

To open the Add-Ons panel if it is not open, click the Add-Ons button  in the sidebar on the left side of the MATLAB Online desktop. Alternatively, go to the **Home** tab, and in the **Environment** section, select **Add-Ons > Add-Ons**.

For more information, see [Get and Manage Add-Ons](#).

## Find Files in MATLAB Online: Manage list of search results and export as plain text

When searching using the Find Files panel in MATLAB Online, you can collapse or clear the list of search results. To collapse or clear the list of search results, click the Find Files actions button (  ) at the top of the Find Files panel and select **Collapse Results** or **Clear Results**.

You also can export the list of search results as a plain text file ( `.txt` ) that can be viewed in an external text editor. To export the list, click the Find Files actions button (  ) and select **Export as Plain Text**.

For more information, see [Advanced Search in MATLAB Online](#).

## New Desktop for MATLAB (Beta): Try new features including dark theme, an updated layout, and expanded search capabilities

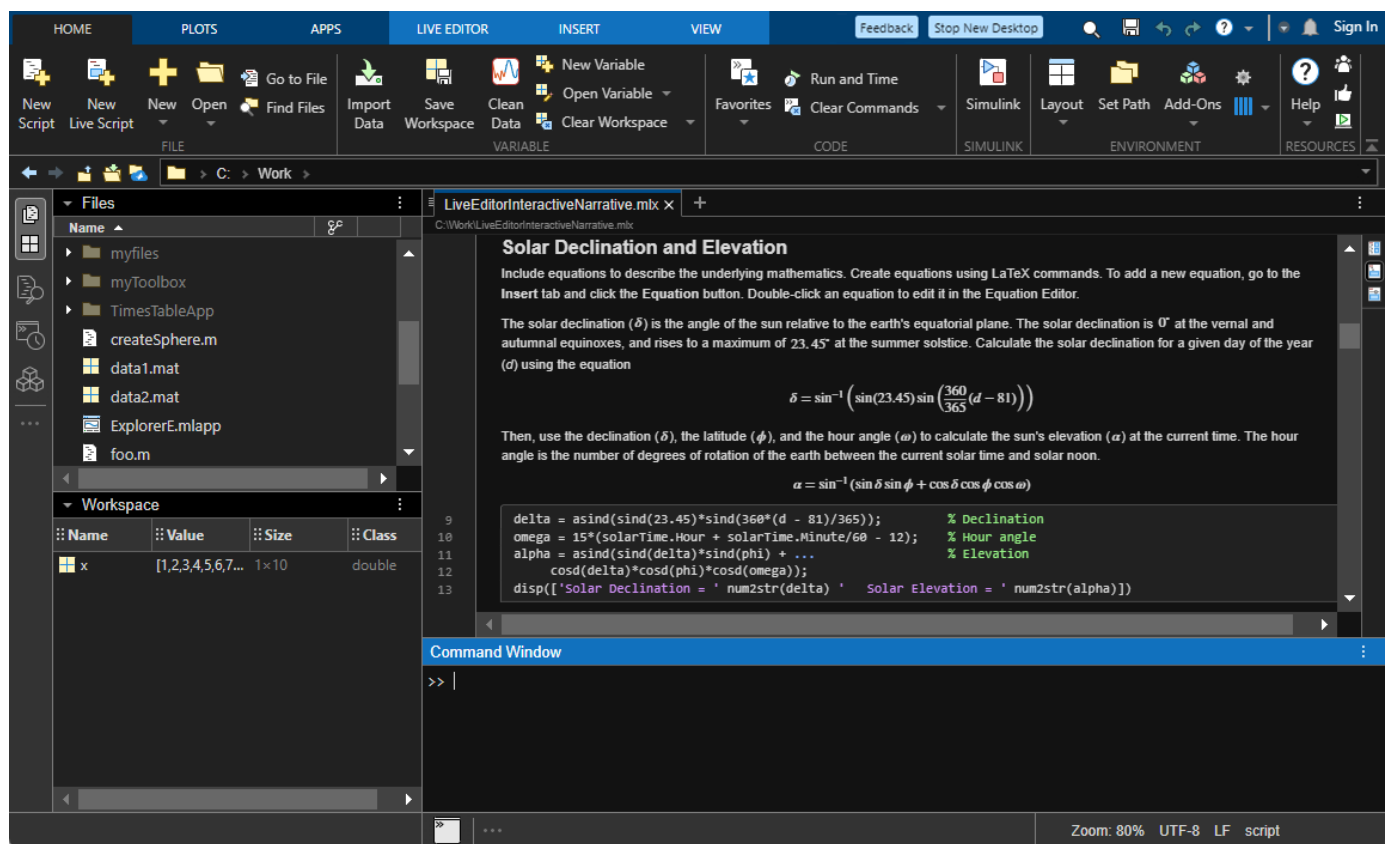
The new desktop for MATLAB includes new and updated features, such as:

- Dark theme support for the desktop, plots, and apps
- An updated layout
- Expanded search capabilities
- A new figure window experience with a toolstrip, integrated side panels, and a new tabbed container
- A new plain text file format for live scripts

- A new Debugger panel and Source Control panel
- Enhanced workflows for MATLAB projects
- Improved screen reader support
- Extension points

To try these new features, download the New Desktop for MATLAB (Beta) on File Exchange. For more information about these new features, see [Get Started with the New Desktop for MATLAB \(Beta\)](#).

**Note** The new desktop for MATLAB is in beta development and should not be used for development or production activities. Report all feedback and issues directly to the development teams by using the **Feedback** button directly above the MATLAB Toolstrip.



## ⚠ Functionality being removed or changed

### File version history in MATLAB Online is no longer supported

#### *Behavior change*

Restoring previous versions of a file in MATLAB Online is no longer supported. Previously, you could view the file version history and restore a previous version of a file by selecting it in the **File Versions** tab.

### demo function no longer supports full product names

#### *Behavior change*

When you use the `demo` function to open examples for products other than MATLAB and Simulink, full product names are no longer supported. You must use the product short name instead.

For example, to open the examples for Computer Vision Toolbox™, use the command `demo toolbox vision`.

**publish function will not longer support "entireFigureWindow" value for figureSnapMethod name-value argument**

*Still runs*

For the `publish` function, specifying the `figureSnapMethod` name-value argument as "entireFigureWindow" will not be supported in a future release. Use the "entireGUIWindow", "print", or "getframe" values instead.

## Language and Programming

### ★dbstop Function: Find missing semicolons by pausing execution on unsuppressed output

When debugging, you can use `dbstop` to pause execution when unsuppressed output is displayed because the line of code is not suppressed by a semicolon (;). Outputs that are displayed for other reasons, such as from `disp` or `fprintf`, do not stop execution.

### Error Messages: Error text now indicates specific location in code

When possible, MATLAB adds caret symbols (^) in error messages beneath the code that led to the error. For example, calling the function `getNumPermutations` with a value of `r` greater than `n` causes an error in the second call to `factorial`. That second call is now specifically highlighted with carets in the error message.

```
function numPerms = getNumPermutations(n,r)
    numPerms = factorial(n)/factorial(n - r);
end
```

```
x = getNumPermutations(2,3)
```

```
Error using factorial (line 20)
N must be an array of real non-negative integers.
```

```
Error in getNumPermutations (line 2)
    numPerms = factorial(n)/factorial(n - r);
                        ^^^^^^^^^^^^^^^^^^^
```

MATLAB also uses caret symbols to highlight inputs that fail argument validation in functions and class methods. For example, both the `Height` and `Width` input arguments for the function `rectangleArea` must be positive. The error message highlights which input does not pass validation.

```
function area = rectangleArea(Height,Width)
    arguments (Input)
        Height (1,1) {mustBePositive}
        Width  (1,1) {mustBePositive}
    end
    area = Height*Width;
end
```

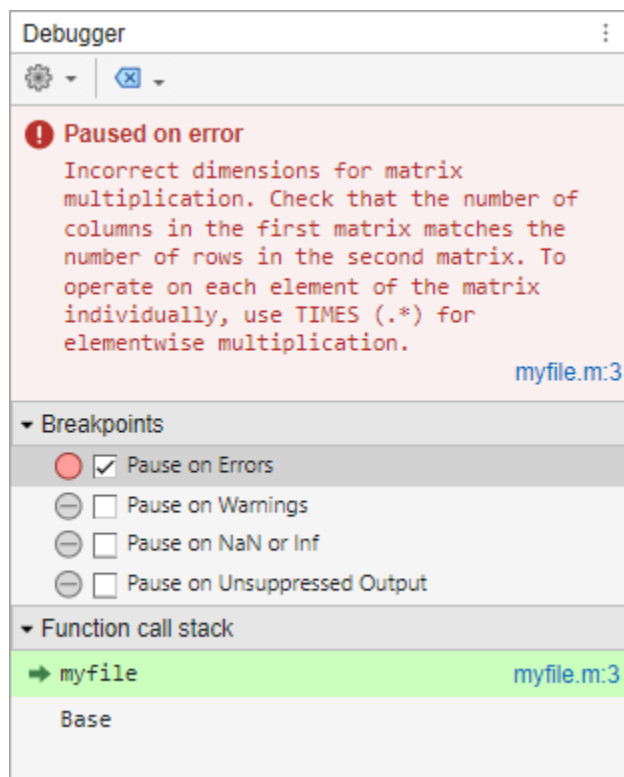
```
a = rectangleArea(5,4-7)
```

```
Error using rectangleArea (line 4)
a = rectangleArea(5,4-7)
    ^^^
```

```
Invalid argument at position 2. Value must be positive.
```

## Debugging in MATLAB Online: View error breakpoint messages in Debugger panel

When the **Pause on Errors** or **Pause on Warnings** breakpoints are enabled and an error or warning occurs, the Debugger panel displays the resulting message and the location at which MATLAB is paused.



For more information about these error breakpoints, see [Set Breakpoints](#).

## matlab.lang.obfuscateNames Function: Obfuscate names of local variables, local functions, and nested functions

Use the `matlab.lang.obfuscateNames` function to obfuscate the names of local variables, local functions, and nested functions in MATLAB source code. While the commands `pcode -R2022a` and `mcc -j` provide obfuscation using P-code files, this `matlab.lang.obfuscateNames` function provides advanced obfuscation options.

## MATLAB Vault: Access and identify existing MATLAB secrets in parallel environments

In parallel environments, you can access and identify existing secrets from your MATLAB vault using `getSecret`, `getSecretMetadata`, and `isSecret`. Creating, removing, and listing secrets in parallel environments is not supported.

## Validation Functions: Validate arguments with matrix and vector shapes

Use the `mustBeMatrix`, `mustBeRow`, and `mustBeColumn` validation functions to enforce the dimensionality of an argument to be 2-D, a 1-by-N row vector, or an M-by-1 column vector, respectively. These functions extend the existing checks `mustBeVector` and `mustBeScalarOrEmpty` for robust size checking in argument and property validation.

For more information, see [Function Argument Validation](#).

## Handle Objects: Define weak references for handle objects

A weak reference to a handle object works much like a regular handle, but a weak reference does not prevent the object it references from being deleted. In other words, if an object is only referred to by weak references, MATLAB automatically deletes the object. Using weak references in applications like caches or data structures with back pointers can help simplify object cleanup.

You can use the `WeakHandle` property attribute to define a weak reference to the value of a property. You can also use the class `matlab.lang.WeakReference` to construct weak references to existing handle objects. For an overview and examples of the functionality, see [Weak Reference Handles](#).

## Object Serialization: Customize how objects are serialized and deserialized

Three new classes provide you with more control over customizing the serialization (saving) and deserialization (loading) processes:

- `matlab.mixin.CustomElementSerialization` — Inherit from this class and implement its methods to control how an object is serialized, deserialized, or both.
- `matlab.serialization.ElementSerializationContent` — Use the methods of this class to add, rename, remove, or change the values of the properties of the object being serialized or deserialized.
- `matlab.serialization.SerializationContext` — Use this class to determine whether to customize serialization and deserialization based on whether the object will be stored in a human readable form, like objects serialized with the MATLAB Data API for C++.

Use these classes to preserve forward and backward compatibility for classes that you update over time. These classes add an extra level of control over serialization and deserialization beyond what the existing `saveobj` and `loadobj` methods provide. `saveobj` and `loadobj` will continue to work, but they cannot be combined with the new `CustomElementSerialization` mixin.

## `matlab.codeanalysis.validateConfiguration` Function: View severity of issues in configuration file

The issues table returned by the `matlab.codeanalysis.validateConfiguration` function includes a column named `Severity` that shows the severity of an issue, specified as `error`, `warning`, or `info`.

The `ErrorMessage` and `ErrorID` columns have been renamed to `Message` and `MessageID`, respectively.

## typecast Function: Convert data types using "like" syntax, and pass logical, character vector, and complex inputs

You can use the syntax `typecast(X, "like", p)` to convert the bit patterns of `X` to the same data type and complexity (real or complex) as the prototype `p`. You can also convert from and to logical or character vector data types, as well as complex numeric types. For more information, see `typecast`.

## matlab.indexing.isScalarClass Function: Identify scalar classes

Use the `matlab.indexing.isScalarClass` function to identify scalar classes. The function returns `true` when its input is a function handle or an instance of a class that inherits from `matlab.mixin.Scalar`.

## Class Diagram Viewer: Import classes from MATLAB packages

The Class Diagram Viewer app and the `matlab.diagram.ClassViewer` command-line API can import classes from MATLAB packages. For more information on packages, see [Organize and Distribute Code Using MATLAB Package Manager](#).

## ▲ Functionality being removed or changed

### mlintrpt function has been removed

#### Errors

The `mlintrpt` function has been removed. Use the Code Analyzer app instead to generate a report of code issues for a selected file or folder and its subfolders.

### Class definition updates after path changes

#### Behavior change

When a new class definition is added to the path and it takes precedence over an old definition of the same name, MATLAB uses the new definition. MATLAB updates existing instances of the class the next time those instances are accessed. For more details, see [Changing the Path to Update Class Definitions](#).

In previous releases, MATLAB immediately updated class definitions when the new definition was in a class folder (@-folder), but for classes defined in path folders, you had to clear the old definition before MATLAB recognized the new definition.

### Default display for sparse matrices includes additional information

#### Behavior change

The default display for sparse matrices with `double` values now explicitly identifies the matrices as sparse. The display also now includes the dimensions, class, and number of nonzero entries in the matrix. For example:

```
A = [0 0 0 5; 0 2 0 0; 1 3 0 0; 0 0 4 0];
sparse(A)
```

```
ans =
```

```
4x4 sparse double matrix (5 nonzeros)
```

```
(3,1)      1
(2,2)      2
(3,2)      3
(4,3)      4
(1,4)      5
```

The default display for sparse matrices with `logical` values already identified the matrices as sparse and included the dimensions and class, but the display now also includes the number of nonzero entries.

### **Implicit default value for properties whose class is a subclass of `matlab.mixin.Scalar`**

*Behavior change*

For properties whose class is a subclass of `matlab.mixin.Scalar`, MATLAB now creates an implicit default value by calling the no-argument constructor of the class. For example, if `MyScalarClass` is a subclass of `matlab.mixin.Scalar`, MATLAB calls the no-argument constructor of `MyScalarClass` to assign a default value to `Prop1` in `TestClass`.

```
classdef TestClass
    properties
        Prop1 MyScalarClass
    end
end
```

Define an explicit default value for `Prop1` if your class requires different behavior.

### **Implicit default value for properties whose class is an enumeration based on numeric or logical values**

*Warns*

When MATLAB assigns a default value to a property whose class is an enumeration based on numeric or logical values, it currently uses the zero-based enumeration member if such a member exists. MATLAB chooses the zero-based member even if that member is not listed first in the enumeration definition. For example, in the class `MyContainer`, the default value of `Prop1` is `MyZero` in R2024b.

```
classdef MyContainer
    properties
        Prop1 (1,1) MyEnum
    end
end

classdef MyEnum < uint8
    enumeration
        MyFirst (1)
        MyZero (0)
        MySecond (2)
        MyThird (3)
    end
end
```

In a future release, MATLAB will instead use the first listed enumeration member as the default, regardless of its underlying numeric value. In the case of `MyContainer`, the default value of `Prop1` will be `MyFirst`. To avoid a warning in R2024b and a future change in behavior, assign `MyEnum.MyZero` as the default value for `Prop1`.

```
classdef MyContainer
    properties
```



```
Prop1 (1,1) MyEnum = MyEnum.MyZero  
end  
end
```

**Defining classes: Using function syntax to define classes will not be supported in a future release***Still runs*

Support for defining classes using `function` syntax will be removed in a future release. With appropriate code changes, replace existing function-based classes with classes defined using the `classdef` keyword.

**Defining classes: Using `schema.m` will not be supported in a future release***Still runs*

Support for classes defined using `schema.m` files will be removed in a future release. Replace existing schema-based classes with classes defined using the `classdef` keyword.

**whos Function: Improved accuracy of array size information***Behavior change*

The `whos` function returns a more accurate array size measurement in the `Bytes` column of the displayed list and in the `bytes` field of the output structure.

## Data Analysis

### **summary Function: Summarize array data and configure summary contents**

The `summary` function can now display or return in a scalar structure a summary of array data, including `numeric`, `logical`, `datetime`, `duration`, and `calendarDuration` types. Previously, the function supported array data only when it was `categorical`.

You can configure the summary contents using one or more name-value arguments:

- `Statistics` — Specify which statistics to compute.
- `Detail` — For table or timetable data only, specify the level of table metadata detail to display in the summary.
- `DataVariables` — For table or timetable data only, specify the variables to summarize.

### **Compatibility Considerations**

When you display a summary of a categorical array, the summary now always includes the number of undefined elements. Previously, the summary omitted the number of undefined elements if the array contained no missing values.

If you want to omit the number of undefined elements from the summary, specify the `Statistics` name-value argument. For example, `summary(A,Statistics="counts")` displays only the number of elements in each category.

### **isbetween Function: Specify numeric, nonnumeric, or tabular input data**

Determine which elements are within a specified range using the `isbetween` function for any array, table, or timetable input data. Previously, supported only date and time input data.

For data in a table or timetable, you can return a table or timetable containing logical values instead of a logical array by using the `OutputFormat` name-value argument. You can also specify tabular variables to operate on by using the `DataVariables` name-value argument.

### **rmmissing and rmoutliers Functions: Define locations of missing entries or outliers using array or table**

Define the locations of missing entries or outliers using a vector, matrix, table, or timetable. Specify the `MissingLocations` name-value argument of the `rmmissing` function or the `OutlierLocations` name-value argument of the `rmoutliers` function.

Previously, `rmmissing` did not support `MissingLocations`, and `rmoutliers` supported specifying `OutlierLocations` only as a vector or matrix.

## Clean Outlier Data Live Editor Task and Data Cleaner App: Define outliers using range or workspace variable

In the Clean Outlier Data task and the **Clean Outlier Data** cleaning method in the Data Cleaner app, you can now detect outliers as elements outside of a range specified by an upper and lower threshold. Select the Range detection method.

In the Clean Outlier Data task, you can also define outliers using a logical workspace variable, where elements with a value of 1 (`true`) correspond to outliers. Select the `Workspace variable` detection method.

## Experiment Manager: Improved interface for experiment setup and result analysis

In the Experiment Manager app, you can set up your experiment using an improved interface.

- Create an initialization function that configures data or other experiment details before initiating the trial runs to reduce trial runtime. Previously, setup code was rerun for each trial.
- Incorporate a suggested parameter for your experiment for some preconfigured experiment templates, such as the template that supports solving ordinary differential equations.

You can also efficiently analyze experiment results.

- Your experiment function for a general purpose experiment can return an output containing multiple data types.
- When an experiment runs using the sequential execution mode, you can stop or cancel the experiment. Clicking the **Stop** button finishes the current trial and ends experiment execution. Clicking the **Cancel** button immediately ends experiment execution.
- When you select a trial in the results table, you can export the selected trial row to the MATLAB workspace as a table.

## table and timetable Data Types: Specify the data types of table and timetable variables

The `VariableTypes` property of tables and timetables specifies the data types of their variables. For more information, see `table` and `timetable`.

## timeseries2timetable Function: Convert events associated with timeseries objects to an event table associated with a timetable

You can convert `timeseries` objects that have events to a timetable that has an event table by using the `timeseries2timetable` function. When `timeseries2timetable` converts `timeseries` objects to a timetable, it also converts all associated `tsdata.event` event objects to an event table. Then the function attaches the event table to the timetable.

Before R2024b, `timeseries2timetable` ignored events when converting `timeseries` objects to a timetable.

## Retime Timetable and Synchronize Timetables Live Editor Tasks: Specify custom function as local function or function handle

In the Retime Timetable and Synchronize Timetables Live Editor tasks, when you select the Custom function option for the **General rule** method, you can either select a locally defined function or specify a function handle.

To select a local function:

- 1 Select Custom function.
- 2 Select Local function from the next menu.
- 3 If you want to add a new local function to the script, click **New**. The task generates template code for a custom aggregation function within the live script. To customize this code, edit it in the live script.
- 4 Select the name of the local function from the select menu.

To specify a function handle:

- 1 Select Custom function.
- 2 Select Function handle from the next menu.
- 3 Specify a function handle in the text box.

You can specify a handle to a MATLAB function or write an anonymous function in the text box. For example, to specify mean as the function, type @mean in the text box.

## Variables Editor in MATLAB Online: View variables with enhanced readability

Variables in the Variables editor in MATLAB Online have enhanced readability. The updated display has a streamlined look, including a simplified, borderless design and row striping for easier data review and analysis.

|     |           |          |             |        |                                                   |
|-----|-----------|----------|-------------|--------|---------------------------------------------------|
| T × | 3×5 table |          |             |        |                                                   |
|     | 1         | 2        | 3           | 4      | 5                                                 |
|     | FlightNum | Customer | Date        | Rating | Comment                                           |
| 1   | 1261      | "Jones"  | 20-Dec-2016 | Good   | "Flight left on time, not crowded"                |
| 2   | 547       | "Brown"  | 21-Dec-2016 | Poor   | "Late departure, ran out of dinner options"       |
| 3   | 3489      | "Smith"  | 22-Dec-2016 | Fair   | "Late, but only by half an hour. Otherwise fine." |

### ⚠ Functionality being removed or changed

#### min and max functions no longer support specifying second input array as character array *Errors*

The min and max functions now return an error if you specify the second input array as a character array. This change minimizes confusion with options that can be specified as character vectors, such

as the missing value condition. To maintain the previous functionality, you can convert the second input array to double, for example, `max(A,double(B),"includenan")`.

## Data Import and Export

### ★Reading Online Data: Read remote data over HTTP and HTTPS using file operation, low-level I/O, datastore, video, and HDF5 functions

In addition to the existing support for reading remote data, you can now use the following functions to read data from primary online sources through an internet URL. Specify a filename that contains the protocol `http://` or `https://`.

- File operation functions: `isfile`, `dir`, `copyfile`, and `exist`
- Low-level I/O functions: `fopen`, `fread`, `fgets`, `fgetl`, `feof`, `fclose`, `ferror`, `fseek`, `ftell`, `fscanf`, `frewind`, and `fileread`
- Datastore functions: `datastore`, `tabularTextDatastore`, `spreadsheetDatastore`, `imageDatastore`, `parquetDatastore`, and `fileDatastore`
- Video function: `VideoReader`
- HDF5 functions: `h5disp`, `h5info`, `h5read`, `h5readatt`, `H5F.open`, and `H5F.is_hdf5`

### JSON: Read and write dictionaries in JSON files

You can read and write dictionaries in JSON files using the `readdictionary` and `writedictionary` functions.

### Parquet: Import Parquet MAP data

You can read the Parquet MAP data type using `parquetread`. The function imports Parquet MAP data as cell arrays in which each cell is a table. Each table has two variables, `Key` and `Value`.

### ZIP Files: Compress and extract password-protected files

Compress password-protected, encrypted ZIP files by specifying a password and encryption method with the `zip` function. Extract the contents of password-protected ZIP files by specifying a password with the `unzip` function.

### Symbolic Links: Create and use symbolic links to manipulate files and folders

You can create and identify symbolic links using the `createSymbolicLink` and `isSymbolicLink` functions. Symbolic links allow you to refer to data in alternative locations.

You can delete, copy, and move files and folders using symbolic links. The deletion of a symbolic link has no effect on the target file, folder, or symbolic link. Use the `ResolveSymbolicLinks` name-value argument with the `delete` and `rmdir` functions, the `CopyLinkBehavior` name-value argument with the `copyfile` function, and the `MoveLinkBehavior` name-value argument with the `movefile` function.

## Spreadsheet Files: Specify how to read merged cells from spreadsheets

You can specify how to read merged cells from spreadsheets using the `MergedCellColumnRule` and `MergedCellRowRule` name-value arguments with these functions:

- `readtable`
- `readtimetable`
- `readmatrix`
- `readcell`
- `readvars`
- `detectImportOptions`

For example, you can specify to place the data from merged cells in the leftmost or rightmost cell, duplicate data in all merged cells, omit rows or columns with merged data, or return an error.

## Image Files: Automatically orient JPEG and TIFF images

If a JPEG or TIFF image file includes an `Exif Orientation` tag, you can automatically orient the image when reading it into MATLAB. Specify the `AutoOrient` name-value argument as `true` when you call the `imread` function.

The output structure of the `imfinfo` function includes `AutoOrientedWidth` and `AutoOrientedHeight` fields. These fields describe the width and height of the image data after the transformations specified by the `Exif Orientation` tag in the image file have been applied.

## sonify Function: Convert numeric data to sound

The `sonify` function converts numeric data to sound using a frequency-based method and provides immediate playback capabilities. You can specify sonification parameters, such as frequency range, immediate playback control, and playback duration.

## VideoReader Object: Read remotely stored video files

You can now read video files stored in remote locations, such as Amazon S3™, Windows Azure® Blob Storage, and HDFS™, using the `VideoReader` object. The `VideoReader` object, however, does not support reading Motion JPEG 2000 (.mj2) files from a remote location.

## FTP and SFTP: Specify time-out values for connections and transfers

You can specify the maximum amount of time allowed for a connection and for a transfer by using the `ConnectionTimeout` and `TransferTimeout` name-value arguments, respectively, with the `ftp` and `sftp` functions.

## FTP: Enable TLS when connecting to FTP servers

When you connect to an FTP server using the `ftp` function, you can enable Transport Layer Security (TLS) using the `TLSMode` name-value argument.

## **SFTP: Connect to servers that require passphrases or certificates**

You can connect to servers with passphrases and certificates by using the `PrivateKeyPassphrase` and `CertificateFilename` name-value arguments, respectively, with the `sftp` function.

## **HDF5 Interface: Use SZIP filter to read and write compressed datasets**

You can write HDF5 datasets using SZIP compression. You can also read HDF5 datasets that have been compressed using SZIP compression.

In the high-level interface, use the `SZIPEncodingMethod` name-value argument of the `h5create` function to specify one of two methods for SZIP encoding. Optionally, if you specify `SZIPEncodingMethod`, you also can use the `SZIPPixelsPerBlock` name-value argument to specify the number of pixels (HDF5 data elements) per block.

In the low-level interface, use the `H5P.set_szip` function to specify SZIP compression when creating a new dataset.

These low-level functions accept the value `"H5Z_FILTER_SZIP"` for the `filter` input argument:

- `H5P.get_filter_by_id`
- `H5P.modify_filter`
- `H5P.remove_filter`
- `H5P.set_filter`
- `H5Z.get_filter_info`

The `H5Z.filter_avail` function returns 1 (true) for the `"H5Z_FILTER_SZIP"` filter identifier.

## **HDF5 Interface: Work with file integrity checks (R2024b Update)**

In the low-level HDF5 interface, you can enable and suppress file integrity checks using the `H5P.set_relax_file_integrity_checks` function. You can query the status of file integrity checks using the `H5P.get_relax_file_integrity_checks` function.

Starting in R2024b, file integrity checks are enabled by default. For compatibility considerations, see “HDF5 file integrity checks are enabled by default (R2024b Update)” on page 3-21.

## **HDF Vgroup Interface: Define longer vgroup names and vgroup class names**

In the HDF Vgroup (V) interface, you can define names of arbitrary length for vgroups and vgroup classes. For more information about the Vgroup interface, see `hdfv`.

## **Scientific File Format Libraries: HDF5 library upgraded to version 1.14.4.3 (R2024b Update)**

The HDF5 library is upgraded to version 1.14.4.3.



▲ **Functionality being removed or changed**

**Version 7.3 MAT files that contain variables of certain data types have decreased performance (R2024b Update)**

Starting in R2024b, Version 7.3 MAT files use a newer version of the HDF5 library. This change causes decreased performance for some Version 7.3 MAT files that contain variables of any nonnumeric MATLAB data type besides `char` or `logical`. Functions that read from or write to MAT files, such as the `load`, `save`, and `matfile` functions, take up to twice as long to execute when called with Version 7.3 MAT files that contain variables of these data types.

**HDF5 file integrity checks are enabled by default (R2024b Update)**  
*Behavior change*

Starting in R2024b, file integrity checks are enabled by default in the MATLAB HDF5 interface. When these checks are enabled, they fail for HDF5 datasets and attributes that contain a datatype that satisfies these two conditions:

- The size of the datatype is at least 2 bytes.
- The precision of the datatype is less than half of the size of the datatype.

As a result, the behavior of these high-level HDF5 functions has changed:

- The `h5read`, `h5readatt`, and `h5writeatt` functions now issue errors for datasets and attributes that fail file integrity checks. Previously, these functions did not issue errors for such datasets or attributes. To read data or read or write attributes in such situations, use the `H5P.set_relax_file_integrity_checks` function and the low-level HDF5 interface.
- The `h5disp` and `h5info` functions now issue warnings for datasets and attributes that fail file integrity checks. These functions omit information about such datasets and attributes from their outputs. Previously, these functions did not issue warnings in this situation and included information about those datasets and attributes in their outputs. The warnings take this form:

Warning: Unable to read 'datasetName' from the file. A portion of the file may be corrupt.

You can suppress file integrity checks in the low-level HDF5 interface using the `H5P.set_relax_file_integrity_checks` function, but not in the high-level HDF5 interface. This table shows how to update your code to use the low-level HDF5 interface.

| R2024a and Earlier                                             | Starting in R2024b                                                                                                                                                                                                                                                                                                |
|----------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>myData = h5read("myFilename.h5","myDatasetName");</code> | <code>fapl = H5P.create("H5P_FILE_ACCESS");<br/>H5P.set_relax_file_integrity_checks(fapl,"H5F_RFIC_UN<br/>fid = H5F.open("myFilename.h5","H5F_ACC_RDONLY",fapl)<br/>dset_id = H5D.open(fid,"myDatasetName");<br/>myData = H5D.read(dset_id);<br/>H5D.close(dset_id)<br/>H5F.close(fid)<br/>H5P.close(fapl)</code> |

| R2024a and Earlier                                                              | Starting in R2024b                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|---------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <pre>myAttrValue = h5readatt("myFilename.h5","myDatasetName",myAttrName);</pre> | <pre>fapl = fopen('myFilename.h5','H5F_FILE_ACCESS'); H5P.set_relax_file_integrity_checks(fapl,"H5F_RFIC_UNSAFE"); fid = H5F.open("myFilename.h5","H5F_ACC_RDONLY",fapl); dset_id = H5D.open(fid,"myDatasetName"); attrID = H5A.open(dset_id,"myAttrName"); myAttrValue = H5A.read(attrID); H5A.close(attrID) H5D.close(dset_id) H5F.close(fid) H5P.close(fapl)</pre>                                                                                                             |
| <pre>h5writeatt("myFilename.h5","myDatasetName",myAttrValue,myAttrName);</pre>  | <pre>fapl = fopen('myFilename.h5','H5F_FILE_ACCESS'); H5P.set_relax_file_integrity_checks(fapl,"H5F_RFIC_UNSAFE"); fid = H5F.open("myFilename.h5","H5F_ACC_RDWR",fapl); dset_id = H5D.open(fid,"myDatasetName");  % Assumes myAttrName is an existing attribute. % If it does not already exist, use H5A.create. attrID = H5A.open(dset_id,"myAttrName");  H5A.write(attrID,"H5ML_DEFAULT",myAttrValue) H5A.close(attrID) H5D.close(dset_id) H5F.close(fid) H5P.close(fapl)</pre> |

## Mathematics

### ★ Solve ODE Live Editor Task: Interactively solve systems of ordinary differential equations

Use the Solve ODE task in the Live Editor to interactively solve a system of ordinary differential equations. You can specify the ODE structure and advanced problem details, select the solver, and visualize the solution in a plot. To add the task to a live script in the Live Editor, click **Task** on the **Live Editor** tab and select **Solve ODE**.

### ode Object: Specify ODE as implicit

You can now specify the Solver property for ode objects as "ode15i" for implicit ODEs. Alternatively, you can specify whether your ODE is implicit by setting the new EquationType property for ode objects to "standard" (default) or "fullyimplicit". If you specify the Jacobian when the equation type is "fullyimplicit", the Jacobian must be a function handle or cell array.

For example, `F = ode(ODEFcn=@myFcn,EquationType="fullyimplicit",Jacobian=@Fjac)` specifies an implicit ODE.

### ode Object: Detect stiffness to change solver after creating ode object

You can change the solver after creating an ode object by using the `selectSolver` method. Use the `DetectStiffness` name-value argument to incorporate a stiffness detection heuristic when selecting a solver.

For example, `d.Solver = selectSolver(d,DetectStiffness="on")` changes the solver for ode object `d` by using a stiffness detection heuristic.

### ode Options: Set minimum step size for several ODE solvers

You can set a lower bound on the size of any step taken by MATLAB ODE solvers by specifying the `MinStep` name-value argument of the `odeset` function.

See the `SolverOptions` property of the ode object for a list of the solvers that support setting a minimum step size.

### lsqminnorm Function: Apply Tikhonov regularization to least-squares solution

You can apply Tikhonov regularization to the least-squares solution of the `lsqminnorm` function by specifying the `RegularizationFactor` name-value argument as a real number. Specify the regularization factor `lambda` to return a solution `X` that minimizes  $\text{norm}(A*X-B)^2 + \text{lambda}^2 * \text{norm}(X)^2$  for each column of `X`.

For example, `X = lsqminnorm(A,B,RegularizationFactor=1e-6)` specifies a Tikhonov regularization factor of `1e-6`.

### **isapprox Function: Determine approximate equality of elements in two arrays**

Determine approximate equality of elements in two arrays and return a logical array using the `isapprox` function. You can define the maximum difference between approximately equal elements by specifying a tolerance level or the relative and absolute tolerances.

### **layoutcoords Function: Calculate node and edge coordinates for the layout of a graph or digraph**

Use the `layoutcoords` function to calculate the node and edge coordinates used when plotting a graph or digraph object. While the `layout` function changes the layout of an existing `GraphPlot` object, `layoutcoords` enables you to obtain the coordinates that would be used by a layout method without plotting the network.

### **polyshape Object: Control whether subtract and xor methods simplify output**

You can control whether the `subtract` and `xor` methods of `polyshape` simplify their output by specifying the `Simplify` name-value argument, which can be either `true` (default) or `false`. By default, the methods return a well-defined polygon by resolving boundary intersections and improper nesting and by removing duplicate points and degeneracies. However, you can specify `Simplify=false` to gain performance when performing a series of subtractions or exclusive-or operations. In this case, you can simplify the output once at the end either by specifying `Simplify=true` in the final method call or by using the `simplify` method on the final output polygon.

### **MATLAB Support Package for Quantum Computing: Solve QUBO problems using QAOA (November 2024, Version 24.2.2)**

You can solve a Quadratic Unconstrained Binary Optimization (QUBO) problem using the quantum approximate optimization algorithm (QAOA). Specify the algorithm for the `solve` function as a `qaoa` object. The resulting solution is a `quboResult` object that contains a `qaoaResult` object.

You can also solve a max-cut problem using the `solve` function. Convert the max-cut graph to an equivalent QUBO formulation by using the `maxcut2qubo` function, which returns a `qubo` object. For more information, see [Solve Max-Cut Problem Using QAOA](#).

### **MATLAB Support Package for Quantum Computing: Calculate expected value and convert QUBO to Ising (February 2025, Version 24.2.4)**

You can calculate the expected value of a quantum circuit or state using the `observe` function. Specify the basis to calculate the expected value in by using an `observable` object or Hermitian matrix.

You can also convert a QUBO problem to an equivalent Ising observable using the `qubo2ising` function, which returns an `observable` object.

## ▲ Functionality being removed or changed

### **ind2sub function no longer supports scalar input for array size**

#### *Errors*

When you use the `ind2sub` function, the input argument for the size of the array must be a vector of positive integers with two or more elements. In previous releases, `ind2sub(sz,ind)` treated a scalar input size `sz` as the number of rows in the array and assumed the size of the other dimensions to be 1. For example:

```
[row,col] = ind2sub(6,[2 3 4])
```

```
row =
```

```
     2     3     4
```

```
col =
```

```
     1     1     1
```

Starting in R2024b, a scalar input for `sz` results in an error. Instead, you must specify `sz` as a vector of positive integers with two or more elements. For example:

```
[row,col] = ind2sub([6 1],[2 3 4])
```

### **MATLAB Support Package for Quantum Computing: run function no longer supports OptimizationLevel argument (January 2025, Version 24.2.3)**

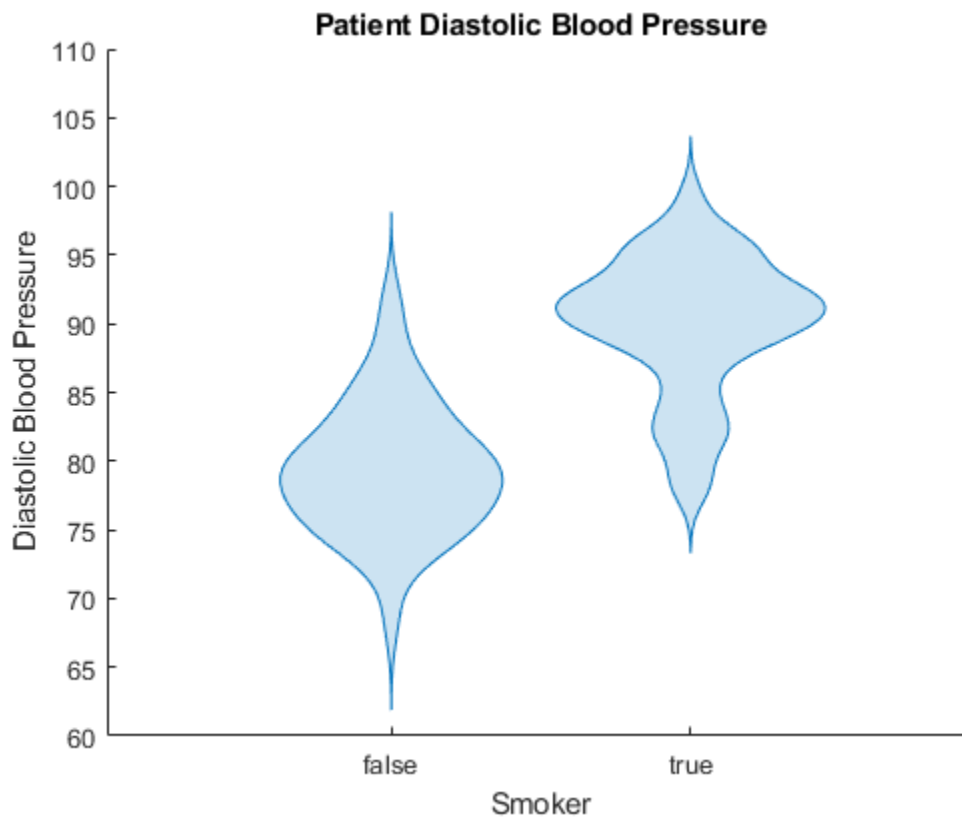
#### *Errors*

The `run` function no longer supports the `OptimizationLevel` name-value argument. Code that uses this argument generates an error. This change affects all versions of MATLAB Support Package for Quantum Computing.

## Graphics

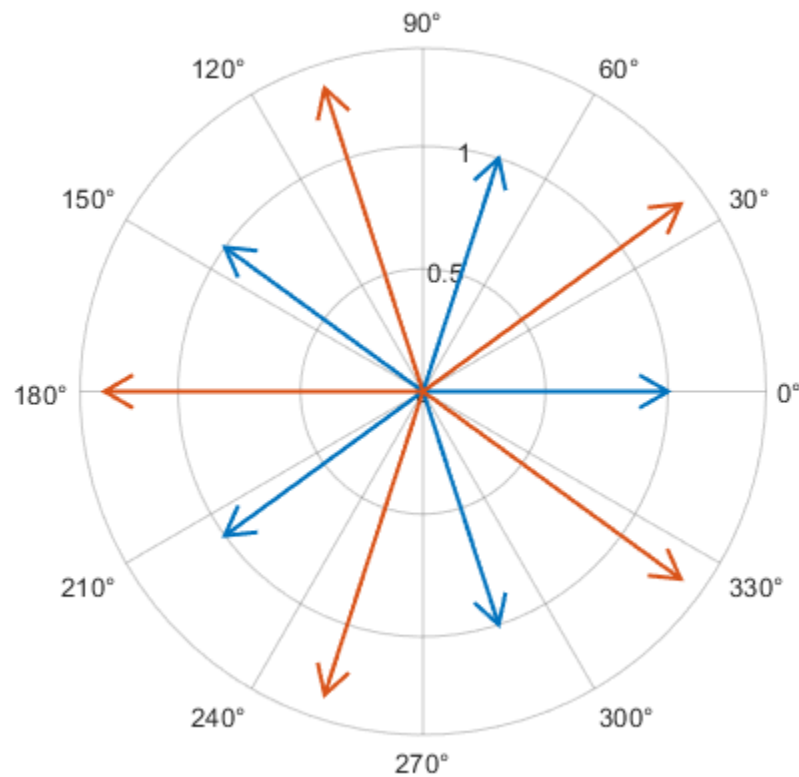
### ★**violinplot** Function: Visualize grouped numeric data by using violin plots

To create violin plots for grouped numeric data, use the `violinplot` function. Each violin plot displays the kernel density estimate for a group in the input data. If you specify a matrix of input data, `violinplot` creates a separate violin plot for each column in the matrix. You can also specify a positional grouping variable to split your data into groups.



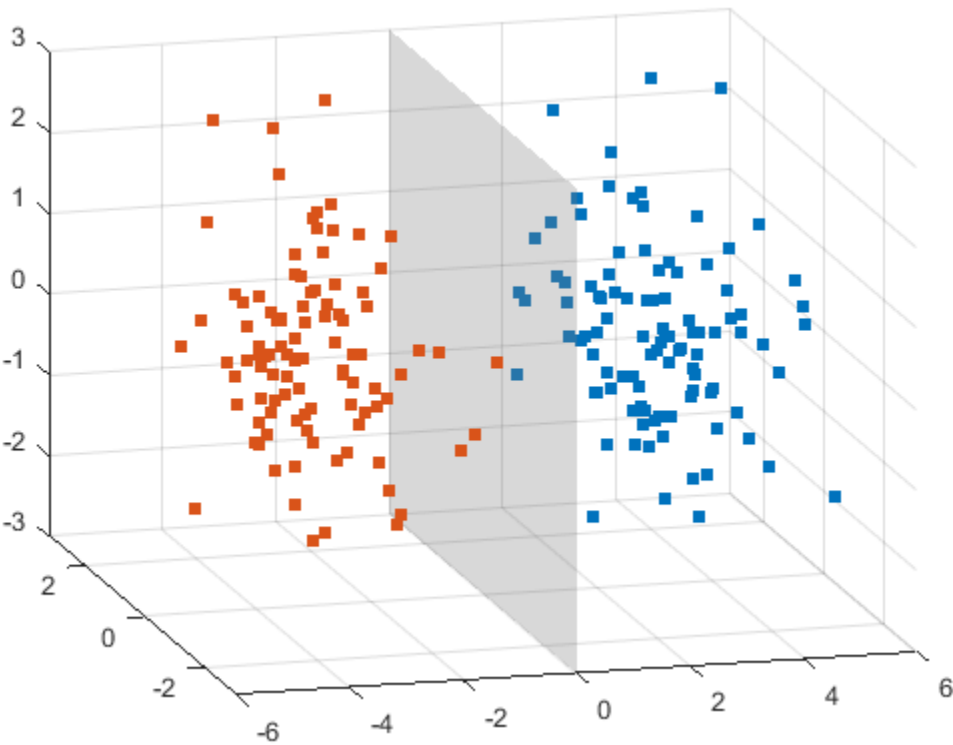
### ★**compassplot** Function: Create one or more compass plots in polar axes

Use the `compassplot` function to plot one or more sets of arrows originating from the origin in polar coordinates. This function is recommended over the `compass` function because this function uses a `PolarAxes` object, which provides more options for customization. For example, you can set `PolarAxes` properties to display either degree or radian *theta* units, adjust *theta*- and *r*-axis limits, or adjust aspects of the tick labels.



### **constantplane Function: Create infinite planes**

Use the `constantplane` function to create infinite planes for highlighting regions of interest in 3-D plots. This function creates semitransparent gray planes by default, but you can change the color and transparency by setting properties.

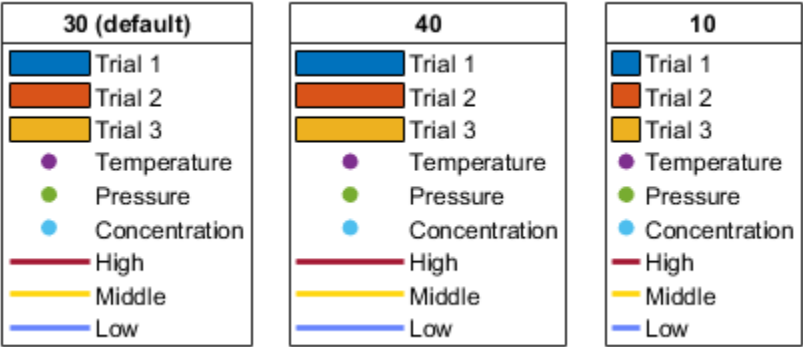


**Legends: Control width of icons**

Control the width of the legend icons and the width of the legend box by setting the `IconColumnWidth` property of a `Legend` object.

If the legend contains only marker icons, MATLAB automatically minimizes the white space around the icons. Previously, legends included extra white space around the marker icons.

For example, these three legends are the same except for their `IconColumnWidth` values (30, 40, and 10, respectively).





## Pie Charts and Donut Charts: Control slice order and number of slices

Simplify pie charts and donut charts with large numbers of slices by setting properties that control the order and number of slices. Use the `DisplayOrder` property to arrange slices by size, and use the `NumDisplayWedges` property to specify the number of slices to display separately (the remaining slice data is grouped together as a slice labeled "Others").

Other useful properties include:

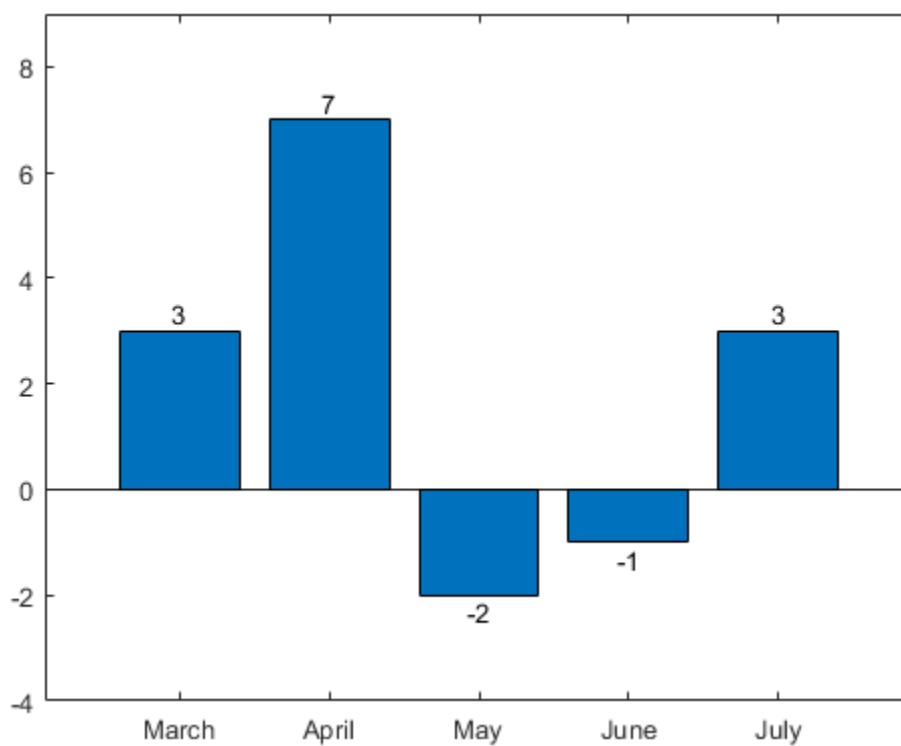
- `ShowOthers` — Show or hide a slice labeled "Others" to represent values not shown as individual slices. For example, `<undefined>` categorical values are represented in the "Others" slice.
- `WedgeDisplayNames` — Return the labels of the slices shown in the chart.
- `WedgeDisplayData` — Return the data values of the slices shown in the chart.

For example, these two donut charts represent the same data, but the chart on the right has a `NumDisplayWedges` property value of 3 and a `ShowOthers` property value of "on".



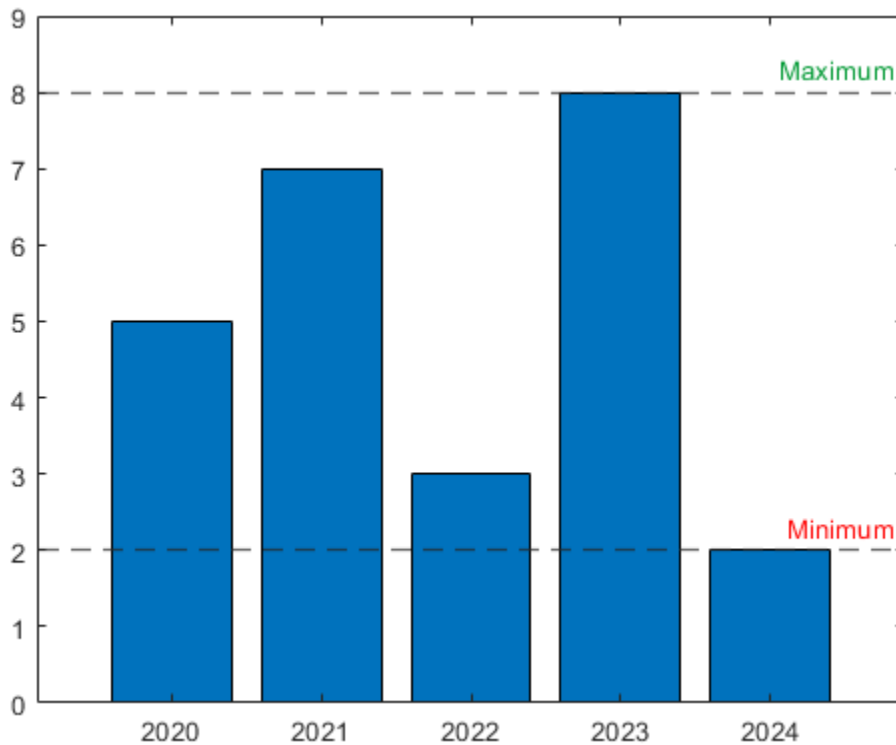
## Bar Charts: Add labels to ends of bars

Add labels to the ends of bars and position them using the `Labels` and `LabelLocation` properties of a `Bar` object. You can also adjust aspects of the font by setting the `LabelColor`, `FontName`, `FontSize`, `FontWeight`, and `FontAngle` properties. To create the labels with LaTeX markup, set the `Interpreter` property.



### **ConstantLine Object: Control text label color**

Use the `LabelColor` property to control the color of the text label of a `ConstantLine` object created with the `xline` or `yline` function.



### **tiledlayout Function: Create layouts without specifying dimensions or tile arrangement**

Call the `tiledlayout` function without specifying the dimensions (`m` and `n`) or the tile arrangement argument. When you omit these arguments, `tiledlayout` creates a layout in the "flow" configuration, which can accommodate any number of tiles.

### **savefig Function: Specify MAT file version**

Specify the MAT file version as `"-v7.3"`, `"-v7"`, or `"-v6"` when you create FIG files using the `savefig` function.

The `"-v7.3"` option supports FIG-files that are larger than 2 GB, and the `"-v7"` option supports up to 2 GB files. The `"-v6"` option stores the file using the MATLAB Version 5 format.

### **Name-Value Arguments: Set properties when you call `pcolor`, `sphere`, `cylinder`, and other plotting functions**

Specify name-value arguments to set properties when you call `pcolor`, `sphere`, `cylinder`, `streamline`, `waterfall`, `ellipsoid`, `feather`, `surfl`, or `ribbon`. Previously, these functions did not support name-value arguments.

For example, specify the `FaceColor` and `EdgeColor` name-value arguments when you call `pcolor`.

```
pcolor(peaks,FaceColor="interp",EdgeColor="none")
```

## Graphics in MATLAB Online: Copy images to clipboard

Now you can use the `copygraphics` function to copy images to the clipboard in MATLAB Online. Previously, you could use the `copygraphics` function only in an installed version of MATLAB.

### ⚠️ Functionality being removed or changed

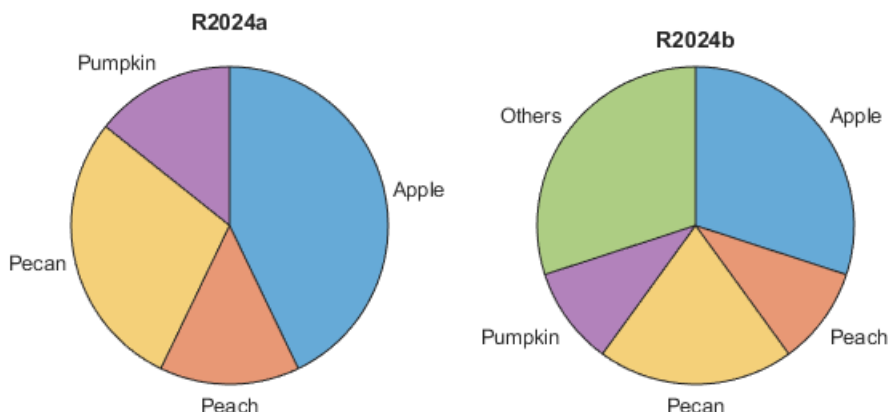
#### **piechart and donutchart show undefined categorical values by default**

*Behavior change*

The `piechart` and `donutchart` functions now include undefined categorical values and show them in a slice labeled "Others" by default. Previously, the functions excluded undefined values from the calculation of the `Proportions` property, and the resulting chart did not include undefined values. This change makes it easier to understand the nature of categorical data in the chart.

For example, create a pie chart using a categorical array containing three undefined values (""). In R2024a, the chart excludes the undefined values. In R2024b, the chart contains a slice labeled "Others" to represent the undefined values.

```
cats = categorical(["Peach" "Apple" "Apple" "Apple" ...  
                  "Pecan" "Pecan" "Pumpkin" "" "" ""]);  
piechart(cats,LabelStyle="name")
```



To exclude the "Others" slice from the chart and exclude the undefined values from the `Proportions` calculation, set the `ShowOthers` property to "off".

```
piechart(cats,LabelStyle="name",ShowOthers="off")
```

#### **savefig function saves FIG files that are no longer compatible with R2014a and earlier releases**

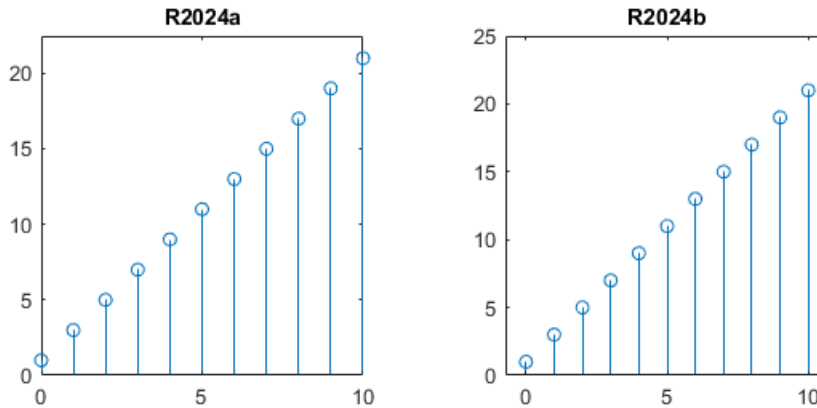
*Behavior change*

The `savefig` function now saves FIG files using the compact format by default. As a result, if you save FIG files in R2024b, you can open them only in R2014b and later releases. Previously, `savefig` stored an extra copy of the figure that you could open in R2014a and earlier releases. This change reduces the file size by approximately 50%.

If you specify the "compact" option when you call `savefig`, MATLAB stores the same compact version of the FIG file that it saves by default. Thus, the "compact" option is unnecessary, but it still runs in R2024b without error. In a future release, the "compact" option will be removed.

**x-axis limits of stem plots are padded by default***Behavior change*

Plots created with the `stem` function now have padding on the right and left sides of the x-axis to prevent overlap with the edges of the plot box.



To preserve the appearance from previous releases, use the `xlim` function to adjust the limits of the x-axis. For example, if the range of the x-coordinates is `[0,10]`, set the x-axis limits to `[0,10]`.

```
x = 0:10;
y = (2*x) + 1;
stem(x,y)
xlim(gca,[0 10])
```

**Passing multidimensional arrays to `isgraphics` and `ishghandle` returns output of same size***Behavior change*

Passing arrays that have three or more dimensions to the `isgraphics` and `ishghandle` functions returns arrays of the same size as the input. Previously, `isgraphics` and `ishghandle` returned 2-D arrays when the input had three or more dimensions. The higher dimensions were collapsed into the second dimension. For example, if the input was a 2-by-4-by-5 array, the output was a 2-by-20 array.

If you pass a scalar, 1-D, or 2-D array to either function, the output is the same size as in previous releases.

If you use nested calls to the `all` or `any` function, update your code to use the "all" argument instead of the nested function calls to preserve the shape of the output.

For example, this code returns a scalar logical value in R2024a. In R2024b, it returns a 1-by-1-by-4 array of logical values.

```
h = gobjects(2,3,4);
all(all(isgraphics(h)))
```

To preserve the scalar output, call the `all` function once with the "all" argument. The result is a scalar logical value.

```
h = gobjects(2,3,4);
all(isgraphics(h),"all")
```

**hgsave and support for saving legacy FIG files will be removed***Still runs*

The `hgsave` function will be removed in a future release. Use `savefig` to save FIG files instead. The `savefig` function supports the same arguments and syntaxes that the `hgsave` function supports, and it saves FIG files that are compatible with R2014b and later releases.

When `hgsave` is removed, you will no longer be able to save FIG-files compatible with R2014a and earlier releases.

**hold all is not recommended**

*Still runs*

The "all" option of the `hold` function is not recommended. Specify the "on" option instead, which has the same effect. However, there are no plans to remove the "all" option.

| Not Recommended             | Recommended                |
|-----------------------------|----------------------------|
| <code>hold all</code>       | <code>hold on</code>       |
| <code>hold("all")</code>    | <code>hold("on")</code>    |
| <code>hold(ax,"all")</code> | <code>hold(ax,"on")</code> |

**compass is not recommended**

*Still runs*

The `compass` function is not recommended. Use the `compassplot` function instead. The `compassplot` function uses a `PolarAxes` object, which provides more options for customization. For example, you can set `PolarAxes` properties to display either degree or radian *theta* units, adjust *theta*- and *r*-axis limits, or adjust aspects of the tick labels. However, there are no plans to remove the `compass` function.

Like other polar plotting functions, `compassplot` accepts polar coordinates instead of Cartesian coordinates. Use the `cart2pol` function to convert your data. This table shows some typical usages and how to update your code to use `compassplot`.

| Scenario                                                                                                                                                                                                                                                                                        | Not Recommended                 | Recommended                                                                                   |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------|-----------------------------------------------------------------------------------------------|
| <b>Plot vectors of coordinates</b> — Convert the Cartesian values of <i>u</i> and <i>v</i> to polar coordinates before passing them to <code>compassplot</code> .                                                                                                                               | <code>compass(u,v)</code>       | <code>[theta,rho] = cart2pol(u,v);<br/>compassplot(theta,rho)</code>                          |
| <b>Plot matrices of coordinates</b> — The <code>compass</code> function plots all of the arrows using the same color. <code>compassplot</code> uses a different color for each matrix column. To make all the arrows the same color, set the <code>SeriesIndex</code> name-value argument to 1. | <code>compass(u,v)</code>       | <code>[theta,rho] = cart2pol(u,v);<br/>compassplot(theta,rho,SeriesIndex=1)</code>            |
| <b>Plot complex values</b>                                                                                                                                                                                                                                                                      | <code>compass(Z)</code>         | <code>compassplot(Z)</code>                                                                   |
| <b>Specify line style and color</b>                                                                                                                                                                                                                                                             | <code>compass(u,v,"--r")</code> | <code>[theta,rho] = cart2pol(u,v);<br/>compassplot(theta,rho,LineStyle="--",Color="r")</code> |

**Calling legend with multiple outputs will not be supported***Warns*

Calling the `legend` function with multiple output arguments returns a warning that the syntax will be removed in a future release.

```
[lgd,icons,plots,txt] = legend(...)
```

This syntax creates a legend that does not support some functionality. For example, you cannot add a title to the legend or specify the number of legend columns. Also, the legend does not automatically update when you add or remove data series from the axes.

Instead, return the `Legend` object and set its properties. For a complete list, see [Legend Properties](#).

```
lgd = legend(...)
```

**opengl function will be removed***Warns*

The `opengl` function returns a warning that it will be removed in a future release. This function enables you to query or change the renderer that MATLAB uses.

- To query the renderer, use the `rendererinfo` function instead of the `opengl` function.
- Changing the renderer will no longer be necessary when the function is removed.

**axis function no longer returns limit selection mode, visibility, and y-axis direction***Errors*

Calling this three-output syntax of the `axis` function is no longer supported.

```
[m,v,d] = axis('state')
```

You can get the same information by querying these `Axes` properties:

- `XLimMode`, `YLimMode`, and `ZLimMode` properties — Axis limit selection mode
- `Visible` property — Visibility
- `XDir` and `YDir` properties — x- and y-axis direction

## App Building

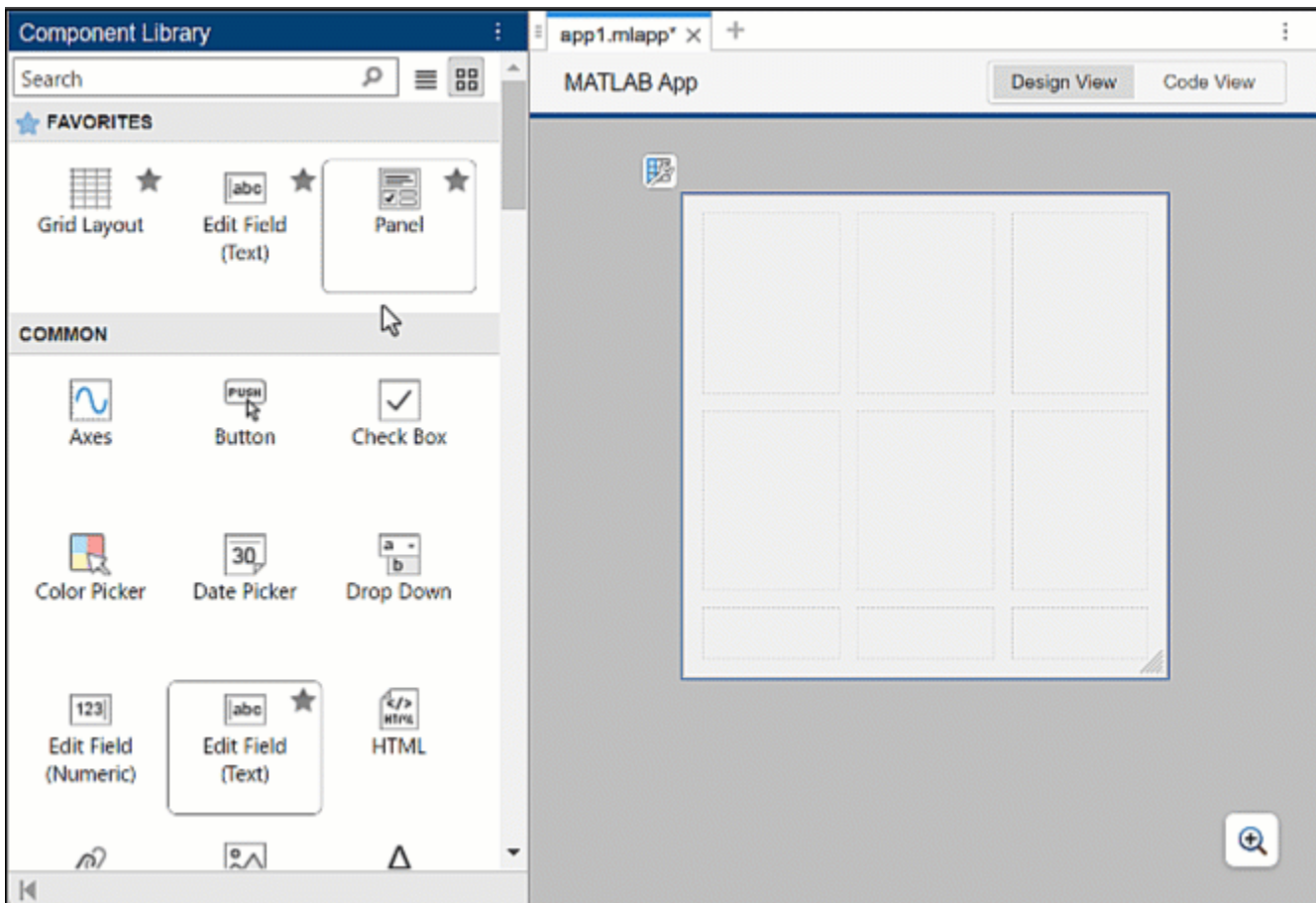
### uibutton and uitogglebutton Functions: Specify HTML markup and display equations in button text

Enable text markup for buttons created using the `uibutton` and `uitogglebutton` functions by setting the Interpreter property. Specify the interpreter as "html", "latex", "tex", or "none".

For more information, see `Button`, `StateButton`, or `ToggleButton`.

### App Designer: Add UI components that span multiple grid cells more easily

In **Design View**, you can interactively add UI components that span multiple cells in a grid layout manager by dragging the component over multiple cells before dropping it on the canvas.



For more information, see [Use Grid Layout Managers in App Designer](#).



## App Designer: View errors directly in Code View when debugging an app

When you debug an app in App Designer with the **Pause on Errors** option selected, **Code View** displays any error that occurs next to the line of code that caused it. Previously, the error was displayed only in the MATLAB Command Window.

## App Designer: Convert between text edit fields and numeric edit fields

To convert one type of edit field to another, right-click the edit field on the canvas and select **Replace With**. Then, select the edit field type to convert to. Replacing one edit field with another preserves relevant property values, such as font properties and callbacks that exist for both components.

## App Testing Framework: Programmatically interact with alert and confirmation dialog boxes

You can programmatically interact with a modal dialog box in front of a figure window by using two new methods in the app testing framework:

- To select an option in a confirmation dialog box, use the `chooseDialog` method.
- To dismiss an alert or confirmation dialog box, use the `dismissDialog` method.

## App Testing Framework: Specify location of gesture on figures that use nonpixel units

You can specify the location of a gesture on a UI figure that uses any unit of measurement. In previous releases, the `press`, `drag`, `hover`, and `chooseContextMenu` methods let you specify coordinates only for figures whose `Units` property is set to "pixels".

For example, create a figure with its `Units` property set to "normalized". Then, programmatically press a point near the lower-left corner of the figure.

```
fig = uifigure(Units="normalized");
testCase = matlab.uittest.TestCase.forInteractiveUse;
testCase.press(fig,[0.1 0.1])
```

## ⚠️ Functionality being removed or changed

### Figure menu bar and toolbar are unaffected by reset

*Behavior change*

When you reset the properties of a Figure object, for example by using `reset` or `clf("reset")`, the `MenuBar` and `ToolBar` properties are unaffected. For more information, see `reset` and `clf`.

### dismissAlertDialog method is not recommended

*Still runs*

The `dismissAlertDialog` method is not recommended. Use `dismissDialog` instead. There are no plans to remove `dismissAlertDialog`. However, the `dismissDialog` method provides more functionality and is not limited to alert dialog boxes. For example, you can use the `dismissDialog` method to close an alert dialog box or a confirmation dialog box.

This table shows how to update your code to use `dismissDialog` instead.

| Not Recommended                                                                                                                                                                  | Recommended                                                                                                                                                                           |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <pre>fig = uifigure;<br/>uialert(fig,"File not found","Invalid File")</pre> <pre>testCase = matlab.uitest.TestCase.forInteractiveUse;<br/>testCase.dismissAlertDialog(fig)</pre> | <pre>fig = uifigure;<br/>uialert(fig,"File not found","Invalid File")</pre> <pre>testCase = matlab.uitest.TestCase.forInteractiveUse;<br/>testCase.dismissDialog("uialert",fig)</pre> |

**Specifying location of gesture on axes or UI axes with active right y-axis is no longer supported**

*Errors*

Using the `press`, `drag`, `hover`, or `chooseContextMenu` method to specify the location of a gesture on axes or UI axes with an active right y-axis is no longer supported. In previous releases, when you specify the location on axes or UI axes with an active right side, the app testing framework interprets that location with respect to the left y-axis, which can cause the gesture to occur at an unexpected location or fail.

If your chart has two y-axes, activate the left side before performing a gesture at a specified location on the axes or UI axes. This table shows an example of how you can update your code.

| Before                                                                                                                                                                                                                          | After                                                                                                                                                                                                                                                 |
|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <pre>fig = uifigure;<br/>ax = axes(fig);<br/>yyaxis(ax,"left")<br/>plot(ax,1:10)<br/>yyaxis(ax,"right")<br/>plot(ax,10:-1:1)</pre> <pre>testCase = matlab.uitest.TestCase.forInteractiveUse;<br/>testCase.press(ax,[5 5])</pre> | <pre>fig = uifigure;<br/>ax = axes(fig);<br/>yyaxis(ax,"left")<br/>plot(ax,1:10)<br/>yyaxis(ax,"right")<br/>plot(ax,10:-1:1)</pre> <pre>testCase = matlab.uitest.TestCase.forInteractiveUse;<br/>yyaxis(ax,"left")<br/>testCase.press(ax,[5 5])</pre> |

## Performance

### datetime Format Parsing: Improved performance when parsing common formats that include localized names or time zone offsets

datetime format parsing performance is significantly faster when parsing common formats that include localized names or time zone offsets. This list shows a few examples of such formats, which are part of ongoing efforts to improve datetime format parsing performance:

- dd-MMM-uuuu HH:mm:ss
- MMM/dd/uuuu
- QQQ-uuuu
- uuuu-MM-dd HH:mm:ss Z
- uuuu-MM-dd HH:mm:ss.SSS xxx

For example, this code creates a string array of dates that use a localized abbreviation for the month of April. The dates in the string array use the dd-MMM-uuuu HH:mm:ss format, where the MMM specifier represents the abbreviated month name. Then the code uses the datetime function to parse and convert the string array. The code is about 17x faster than in the previous release.

```
function timingTest
s = "23-Apr-2024 11:30:" + randi([10,59],10000,1);
for i = 1:100
    d = datetime(s, ...
        InputFormat="dd-MMM-uuuu HH:mm:ss", ...
        Locale="en_US");
end
end
```

The approximate execution times are:

**R2024a:** 4.75 s

**R2024b:** 0.27 s

Similarly, this code creates a string array of dates that have a time zone offset of -0400. The dates in the string array use the uuuu-MM-dd HH:mm:ss.SSS Z format, where the Z specifier represents the time zone offset. Then the code uses the datetime function to parse and convert the string array. The code is about 5.8x faster than in the previous release.

```
function timingTest
s = "2024-04-23 11:32:48." + randi(999,10000,1) + " -0400";
for i = 1:100
    d = datetime(s, ...
        InputFormat="uuuu-MM-dd HH:mm:ss.SSS Z", ...
        TimeZone="local");
end
end
```

The approximate execution times are:

**R2024a:** 6.35 s

**R2024b:** 1.10 s

In both cases, the code was timed on a Windows 11, AMD EPYC 74F3 24-Core Processor @ 3.19 GHz test system using the `timeit` function.

```
timeit(@timingTest)
```

## **median Function: Improved performance when computing weighted median**

The `median` function shows improved performance when you specify a weighting scheme. For example, this code computes the weighted median of a 600-by-10 matrix. The code is about 1.8x faster than in the previous release.

```
function timingTest
A = rand(600,10);
W = rand(600,1);

for i = 1:3e3
    median(A,Weights=W);
end
end
```

The approximate execution times are:

**R2024a:** 1.00 s

**R2024b:** 0.55 s

The code was timed on a Windows 11, AMD EPYC 74F3 24-Core Processor @ 3.19 GHz test system using the `timeit` function.

```
timeit(@timingTest)
```

## **fillmissing Function: Improved performance when filling numeric entries using nearest neighbor rows**

The `fillmissing` function shows improved performance when you specify the fill method as "knn" for numeric data and use the Euclidean or scaled Euclidean distance function to find the nearest neighbor rows. The improvement is most significant when the length of the input data along the operating dimension is small.

For example, this code fills the NaN values in an 800-by-10 matrix using the nearest neighbor rows. The code is about 2.8x faster than in the previous release.

```
function timingTest
A = rand(800,10);
A(A>0.95) = NaN;

for i = 1:2e2
    F = fillmissing(A,"knn");
end
end
```

The approximate execution times are:

**R2024a:** 1.00 s

**R2024b:** 0.36 s

The code was timed on a Windows 11, AMD EPYC 74F3 24-Core Processor @ 3.19 GHz test system using the `timeit` function.

```
timeit(@timingTest)
```

## **isprime Function: Improved performance when determining which array elements are prime**

The `isprime` function shows improved performance when determining whether the integer elements of an array are prime. For example, this code checks if the elements of a 1-by-1,000,000 vector of integers are prime. The code is about 20x faster than in the previous release.

```
function timingTest
x = 1:1e6;
tf = isprime(x);
end
```

The approximate execution times are:

**R2024a:** 2.81 s

**R2024b:** 0.14 s

The code was timed on a Windows 11, AMD EPYC 74F3 24-Core Processor @ 3.19 GHz test system using the `timeit` function.

```
timeit(@timingTest)
```

## **simplify Function: Improved performance when simplifying polyshape objects with many boundaries**

The `simplify` function shows improved performance when operating on `polyshape` objects with many boundaries. For example, this code simplifies a `polyshape` object with 11,100 vertices. The code is about 23x faster than in the previous release.

```
function t = timingTest
n = 300;
x = (1:n)'*[sind(0:10:360),NaN];
y = (1:n)'*[cosd(0:10:360),NaN];
p = polyshape(x(:),y(:),Simplify=false);

f = @() simplify(p);

t = timeit(f);
end
```

The approximate execution times are:

**R2024a:** 0.28 s

**R2024b:** 0.012 s

The code was timed on a Windows 11, AMD EPYC 74F3 24-Core Processor @ 3.19 GHz test system by calling the `timingTest` function.

## waitbar Function: Improved performance when updating wait bar dialog box in loop

Updating a wait bar dialog box created using the `waitbar` function shows improved performance when the updates happen many times in a loop. For example, this code updates a wait bar dialog box progress bar and message while approximating the value of  $\pi$ . The code is about 3.9x faster than in the previous release.

```
function timingTest
f = waitbar(0,"1",Name="Approximating pi...");
pisqover8 = 1;
denom = 3;
valueofpi = sqrt(8*pisqover8);
steps = 20000;

tic
for step = 1:steps
    waitbar(step/steps,f,sprintf('%12.9f',valueofpi))
    pisqover8 = pisqover8 + 1/(denom*denom);
    denom = denom + 2;
    valueofpi = sqrt(8*pisqover8);
end
toc

delete(f)
end
```

The approximate execution times are:

**R2024a:** 58.7 s

**R2024b:** 14.9 s

The code was timed on a Windows 11, AMD EPYC 74F3 24-Core Processor @ 3.19 GHz test system by calling the `timingTest` function.

## Plot Animations: Improved performance when updating charts that have legends in loop

Updating graphics properties in a loop shows improved performance when the axes object contains multiple graphics objects and a legend. The improvement increases with the number of graphics objects in the axes.

For example, plot 10 lines and add a legend.

```
axes
hold on
numLines = 10;
for i = 1:numLines
    plot(i:i+10);
end
legend(Location="eastoutside")
```

Calculate values of a sine wave and store them as `y`. Create another line, `p1`, with NaN coordinate values. Animate the line in a loop by appending `y(i)` to the `YData` property of `p1` at each iteration. This code is about 5x faster than in the previous release.

```
% Calculate sine values and create p1
x = 1:0.05:11;
y = sin(x)*10+10;
p1 = plot(x, nan(1, numel(x)));

% Animate p1
tic
for i = 1:numel(x)
    p1.YData(i) = y(i);
    drawnow
end
toc
```

The approximate execution times are:

**R2024a:** 14.14 s

**R2024b:** 2.89 s

The code was timed on a Windows 11, Intel Xeon CPU W-2133 @ 3.60 GHz test system.

## Plots in Apps: Improved performance when creating surface plots of large data sets

In apps, creating surface plots of large data sets shows improved performance. For example, create an app window using the `uifigure` function, and create an axes object using the `axes` function. Call the `drawnow` function to complete the figure and axes setup. Then plot a 1000-by-1000 sampling of the peaks data set using the `surf` function. The code is about 1.3x faster than in the previous release.

```
% Create app window and axes
f = uifigure;
a = axes(f);
drawnow

% Plot surface
tic
surf(a, peaks(1000), ...
    FaceColor="interp", EdgeColor="none")
drawnow
toc
```

The approximate execution times are:

**R2024a:** 0.478 s

**R2024b:** 0.356 s

The code was timed on a Windows 11, AMD EPYC 74F3 24-Core Processor @ 3.19 GHz test system that has an NVIDIA® A16-2Q GPU with 2 GB RAM.

## Create Plot Live Editor Task: Improved loading performance

Loading the Create Plot task in the Live Editor is faster in R2024b than in R2024a. When you add the task to a live script, it takes less time for the task to appear in the Live Editor.

For example, if you add the Create Plot task to a new script in the Live Editor, the task appears about 1.2x faster than in the previous release.

The approximate load times are:

**R2024a:** 19.9 s

**R2024b:** 16.4 s

If you add another Create Plot task to the same live script, the additional task appears about 1.4x faster than in the previous release.

The approximate load times are:

**R2024a:** 5.7 s

**R2024b:** 4.0 s

The actions were timed on a Windows 11, Intel Xeon Gold 6246R CPU @ 3.40 GHz test system.



# Software Development Tools

## Project API: Specify dependency analysis scope

When you analyze a project using the `updateDependencies` function, you can specify the dependency analysis scope. For example, to analyze dependencies inside add-ons, use `updateDependencies(currentProject, AnalyzeAddOns=true)`.

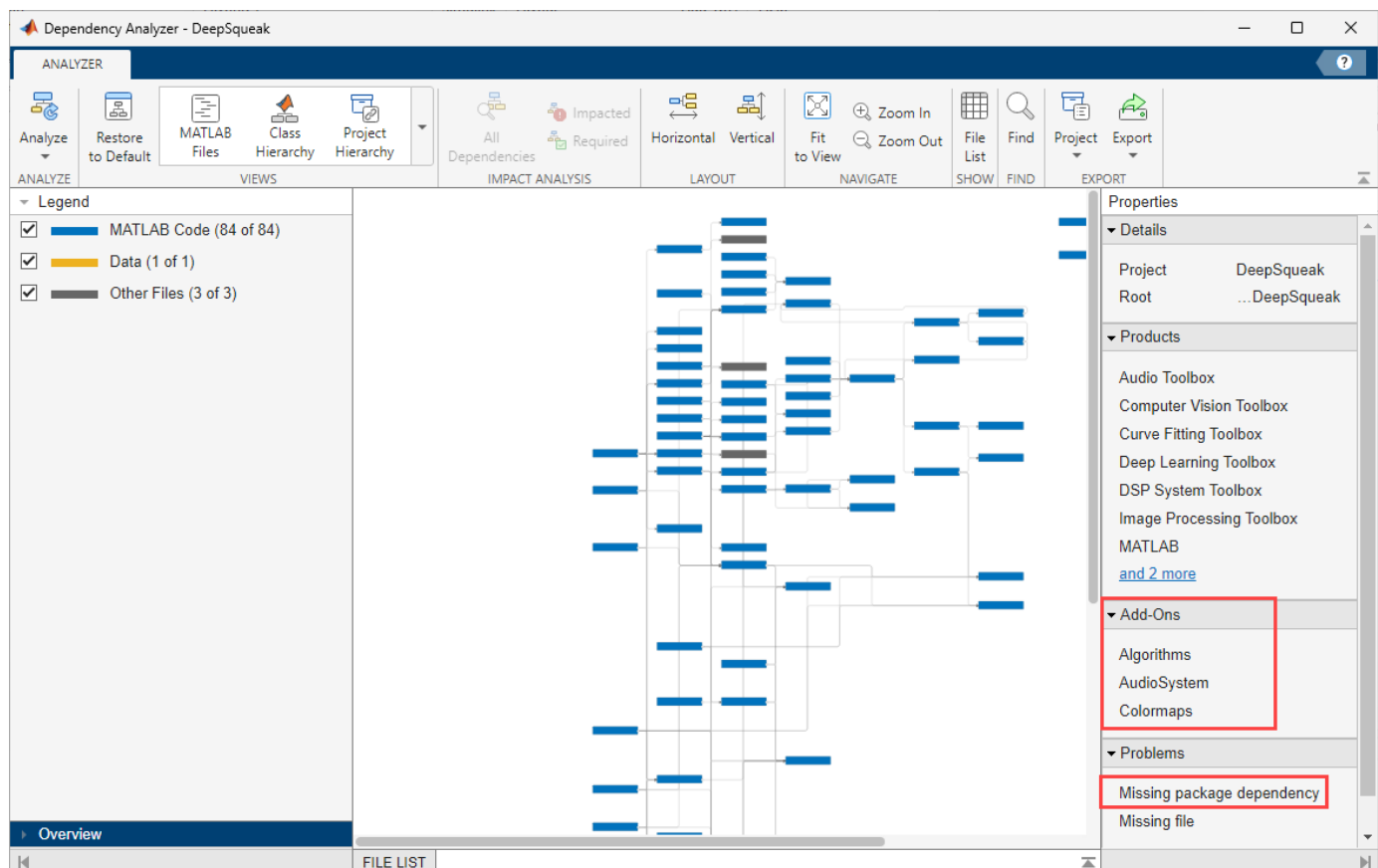
## Project API: Label multiple project files at once

You can programmatically attach a label to multiple project files at once using `addLabel`. You can also detach a label from multiple files using `removeLabel`.

## Dependency Analyzer: Find required packages and missing package dependencies

When you run a dependency analysis on your code, the Dependency Analyzer lists the required packages in the **Add-Ons** section of the **Properties** pane. For more information, see Find Required Products and Add-Ons.

The Dependency Analyzer also flags any missing package dependencies in the **Problems** section. For more information, see Investigate and Resolve Problems.



## Source Control in MATLAB Online: Expanded support for Git workflows

MATLAB Online now provides expanded support for Git workflows.

- Annotate lines in the MATLAB Editor using Git history.
- Check out a Git branch to a different folder using Git work trees.

## Project Issues in MATLAB Online: View and fix project startup and shutdown issues

In MATLAB Online, you can view all issues that occur during project startup and shutdown using the Project Issues panel. When possible, this panel provides automatic fixes or suggests next steps.

## Code Analyzer: Check code for specified variable names

Configure the Code Analyzer to display a custom check when specific variable names are used. To create a new check, you must assign a check ID that is a valid MATLAB identifier. For example, you can define "MyVariableCheck" to check for variables with the names `size`, `error`, or `length`.

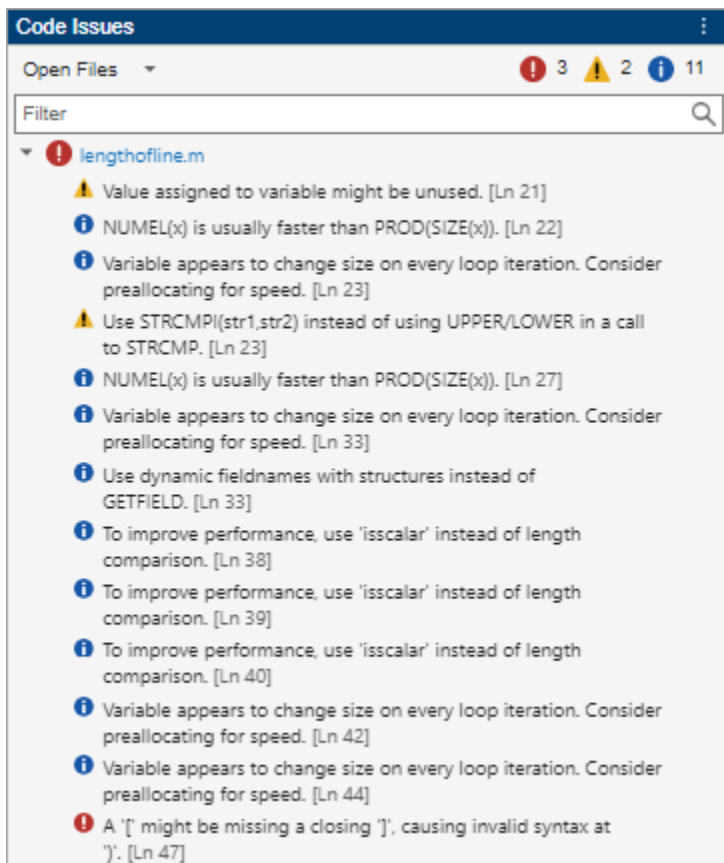
For more information, see [Add Custom Checks for Specified Variable Names](#).

## Code Analyzer in MATLAB Online: Check code for info messages

In MATLAB Online, you can view info messages about your code in the Code Issues panel, as well as in the Editor and Live Editor. Info messages are suggestions for how to improve your code, and they can be related to code readability, outdated functionality, and formatting improvements.

To open the Code Issues panel, go to the **Editor** or **Live Editor** tab, and in the **Analyze** section, click  **Code Issues**. To change which types of messages are underlined in the Editor and Live Editor, go to the **Home** tab, and in the **Environment** section, click  **Preferences**. Select **MATLAB > Code Analyzer**, and then select an **Underlining** option.

For more information, see [Check Code for Errors and Warnings Using the Code Analyzer](#).



## ★ Build Automation: Create and run group of tasks

You can treat similar tasks as a single unit of work in the build tool. For instance, you can create a group of all the tasks in your project that build binary MEX files, and then list, plot, or run the grouped tasks. For an example of how to create and run a task group, see [Build MEX Files Using Task Group](#).

## Build Automation: Specify task names more flexibly

You can specify the name of a `matlab.buildtool.Task` instance as a string scalar or character vector of any length that contains alphanumeric (A-Z, a-z, 0-9), underscores, dashes, and periods. In previous releases, the task name must be a valid MATLAB identifier.

## Build Automation: Produce code coverage results using coverage formats

To produce customized code coverage results, you can pass a vector of code coverage formats to the `addCodeCoverage` method of the `matlab.buildtool.tasks.TestTask` class. For more information and an example, see [results](#).

## Build Automation: Specify build file and verbosity level

You can specify additional options when running a build with the `buildtool` command:

- To run a build using a specified build file, use the `-buildFile` option.
- To control the amount of information displayed during a build run, use the `-verbosity` option.

For example, run the default tasks in a build file located in a subfolder of your current folder, and display build output at the maximum verbosity level.

```
buildtool -buildFile mySubfolder\buildfile.m -verbosity Verbose
```

## Build Automation: Load plan from folder into workspace

You can load a plan from a folder that contains a build file into the MATLAB workspace by using the `matlab.buildtool.Plan.load` method. The build file must be named `buildfile.m` and be located directly within the folder that you pass to the method.

## Build Automation: Delete additional task outputs

To delete outputs of a task, a `matlab.buildtool.tasks.CleanTask` instance searches through both the actual and expected outputs:

- Actual outputs — Outputs on disk that the task produced when it last ran successfully (stored in the task trace)
- Expected outputs — Outputs that the task would produce if it ran at cleanup time (specified in the build plan)

For example, if you change the inputs or outputs of a task in the build file, a `CleanTask` instance deletes the outputs on disk from the last task run even if the build file no longer represents them. In previous releases, a `CleanTask` instance misses such outputs because it searches only for outputs that are expected to exist on disk when cleanup takes place.

## Unit Testing Framework: Run function in environment provided by fixtures

You can use the `applyAndRun` method of the `matlab.unittest.fixtures.Fixture` class to run a function in the environment provided by one or more fixtures. The method sets up the fixtures before running the function and tears them down after running the function.

## Unit Testing Framework: Control test runner verbosity using testrunner function

You can specify the display level of test output detail and verbosity level of logged diagnostics when you create a test runner using the `testrunner` function. For example, run the tests in a test class by using a test runner that displays test output detail at the maximum verbosity level.

```
suite = testsuite("MyTestClass");  
runner = testrunner("textoutput",OutputDetail="verbose");  
results = run(runner,suite);
```

## Unit Testing Framework: Add tests from currently open test file or clear test results using Test Browser

You can use Test Browser menu options to:

- Add the tests from the currently open test file to the test browser. Click the drop-down arrow (▼) on the toolbar and then select **Current File**.
- Clear the test results and return all the tests to a Not Run status  

. Click the three-dot button  
(⋮) on the toolbar and  
then select **Clear test results**.

## Unit Testing Framework: Control whether to use Test Browser from toolstrip in MATLAB Online

In MATLAB Online, when you run a function-based or class-based test file using the **Run Tests** section on the toolstrip, you can control whether to run the tests and display results using the Test Browser app. To access this test option, with your test file open, on the **Editor** or **Live Editor** tab, select **Run Tests > Customize Test Run**, and then select or clear the **Use Test Browser** option.

If you choose not to use the test browser by clearing the **Use Test Browser** option, then your tests run in the Command Window instead. Your selection persists across MATLAB sessions.

## App Testing Framework: Programmatically interact with alert and confirmation dialog boxes

You can programmatically interact with a modal dialog box in front of a figure window by using two new methods in the app testing framework:

- To select an option in a confirmation dialog box, use the `chooseDialog` method.
- To dismiss an alert or confirmation dialog box, use the `dismissDialog` method.

## App Testing Framework: Specify location of gesture on figures that use nonpixel units

You can specify the location of a gesture on a UI figure that uses any unit of measurement. In previous releases, the `press`, `drag`, `hover`, and `chooseContextMenu` methods let you specify coordinates only for figures whose `Units` property is set to "pixels".

For example, create a figure with its `Units` property set to "normalized". Then, programmatically press a point near the lower-left corner of the figure.

```
fig = uifigure(Units="normalized");
testCase = matlab.uitest.TestCase.forInteractiveUse;
testCase.press(fig,[0.1 0.1])
```

## MATLAB Package Manager: Organize and distribute code using packages

Use the Package Management to create, find, install, and manage MATLAB packages. A package is a collection of MATLAB code, related files, and a package definition file that defines the package

identity and dependencies. The purpose of a package is to compartmentalize code so that it can be shared while maintaining its intended functionality. When installing a package, the MATLAB Package Manager installs all of the code, supporting files, and any other packages that it depends on. For more information, see [Organize and Distribute Code Using MATLAB Package Manager](#).

MATLAB keeps a list of known repositories. A package repository is a designated location where packages are available for distribution. Authors can place packages in a package repository, and other users can then install those packages. You can manage the set of known package repositories directly within MATLAB. For more information, see [Distribute Packages Using Folder-Based Repositories](#).

**⚠️ Functionality being removed or changed**

**dismissAlertDialog method is not recommended**

*Still runs*

The `dismissAlertDialog` method is not recommended. Use `dismissDialog` instead. There are no plans to remove `dismissAlertDialog`. However, the `dismissDialog` method provides more functionality and is not limited to alert dialog boxes. For example, you can use the `dismissDialog` method to close an alert dialog box or a confirmation dialog box.

This table shows how to update your code to use `dismissDialog` instead.

| Not Recommended                                                                                      | Recommended                                                                                               |
|------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------|
| <pre>fig = uifigure;<br/>uialert(fig,"File not found","Invalid File")</pre>                          | <pre>fig = uifigure;<br/>uialert(fig,"File not found","Invalid File")</pre>                               |
| <pre>testCase = matlab.uitest.TestCase.forInteractiveUse;<br/>testCase.dismissAlertDialog(fig)</pre> | <pre>testCase = matlab.uitest.TestCase.forInteractiveUse;<br/>testCase.dismissDialog("uialert",fig)</pre> |

**Specifying location of gesture on axes or UI axes with active right y-axis is no longer supported**

*Errors*

Using the `press`, `drag`, `hover`, or `chooseContextMenu` method to specify the location of a gesture on axes or UI axes with an active right y-axis is no longer supported. In previous releases, when you specify the location on axes or UI axes with an active right side, the app testing framework interprets that location with respect to the left y-axis, which can cause the gesture to occur at an unexpected location or fail.

If your chart has two y-axes, activate the left side before performing a gesture at a specified location on the axes or UI axes. This table shows an example of how you can update your code.

| Before                                                                                                                                                                                                                         | After                                                                                                                                                                                                                                                |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <pre>fig = uifigure;<br/>ax = axes(fig);<br/>yyaxis(ax,"left")<br/>plot(ax,1:10)<br/>yyaxis(ax,"right")<br/>plot(ax,10:-1:1)<br/><br/>testCase = matlab.uittest.TestCase.forInteractiveUse;<br/>testCase.press(ax,[5 5])</pre> | <pre>fig = uifigure;<br/>ax = axes(fig);<br/>yyaxis(ax,"left")<br/>plot(ax,1:10)<br/>yyaxis(ax,"right")<br/>plot(ax,10:-1:1)<br/><br/>testCase = matlab.uittest.TestCase.forInteractiveUse;<br/>yyaxis(ax,"left")<br/>testCase.press(ax,[5 5])</pre> |

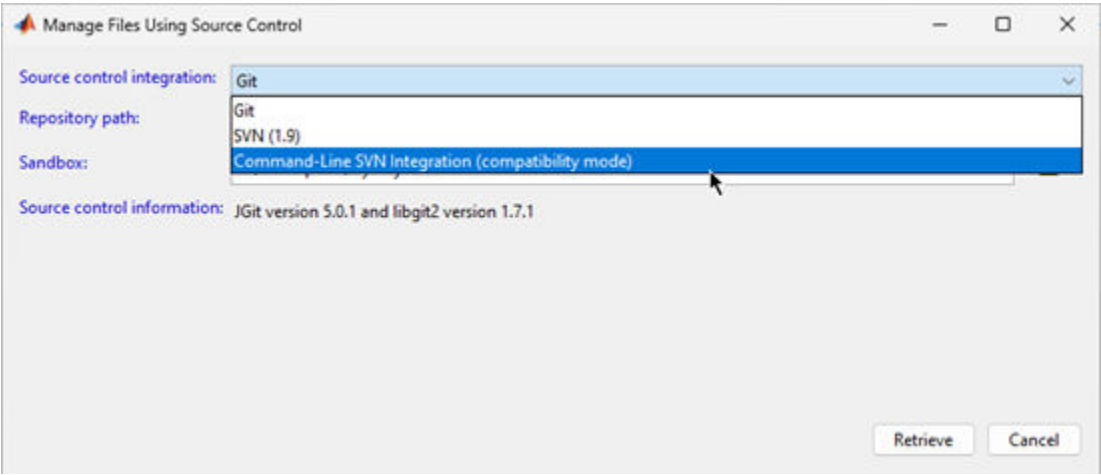
**Toolstrip in MATLAB Online no longer includes the Section section for test files**  
*Behavior change*

Because creating and running sections in function-based and class-based test files is not common, if you open a function-based or class-based test file in MATLAB Online, the **Section** section no longer appears on the **Editor** or **Live Editor** tab.

**Legacy command-line SVN integration is no longer supported**

Legacy command-line SVN integration had limited functionality and required installation of an additional command-line SVN client. Starting in R2024b, the command-line SVN integration is no longer supported. Use the built-in SVN integration instead.

This illustration shows the deprecated **Command-line SVN integration (compatibility mode)** option in the Manage Files Using Source Control dialog box. Use the **SVN (1.9)** option instead.



## External Language Interfaces

### **REST Function Service: Automatic restart after MathWorks Service Host update**

MATLAB REST function services rely on the MathWorks Service Host (MSH) for communication with MATLAB from client applications. MSH regularly releases updates and restarts automatically after updates are installed. If services are running before the update, the REST function connector stops during the update and then automatically restarts, along with the running services, when the MSH is available.

Any computations or requests made by your program during the update cannot be recovered. If you have long-running function calls, you can specify a preferred restart time to avoid interruption of your workflow and possible loss of data. For more information, see the `PreferredRestartHour` property of the `RESTFunctionConnector` object.

### **Call C++ from MATLAB: Convert MATLAB structures to C++ struct arrays when passing data from MATLAB**

You can pass a MATLAB structure as a parameter to a C++ function. MATLAB converts the structure to a C++ `struct` argument. For more information, see `Pass struct Parameter`.

You can convert MATLAB structures to C++ `struct` arrays using the `clibConvertArray` function.

For information about publishing C++ functions with `struct` parameters that accept MATLAB structures, see `Supported struct Types`.

### **Call MATLAB from C++: Create MATLAB data arrays from user-managed buffers**

You can create a MATLAB data array (MDA) using memory already allocated, without copying data or allocating extra memory, using the MATLAB Data API for C++ function `createArrayFromBuffer`. When you create an MDA from a buffer, you can specify a custom deleter function to manage the buffer. For more information, see `Create MATLAB Data Array and Manage Memory from User-Managed Buffer`.

### **Call MATLAB from C++: Create `matlab::data::CharArray` from UTF-8 input**

Use the `createCharArrayFromUTF8` function of the MATLAB Data API for C++ `matlab::data::ArrayFactory` class to create a `matlab::data::CharArray` object from an `std::string` of UTF-8 characters.

### **Java: Support for OpenJDK 21 Java**

MATLAB now supports OpenJDK® 21 Java from <https://adoptium.net/>. For supported version information, see `MATLAB Interfaces to Other Languages`.



## Python: Support for CPython version 3.12

MATLAB now supports CPython version 3.12, in addition to existing support for versions 3.9, 3.10, and 3.11. For supported version information, see [Versions of Python Compatible with MATLAB Products by Release](#).

## Call Python from MATLAB: Follow Python type hints when passing data from MATLAB

MATLAB can automatically convert a floating-point integer to a Python `int` when passing data to a Python function. This conversion occurs if a Python type hint specifies an integer data type. For more information, see [Pass MATLAB Data to Python](#).

## Call MATLAB from Python: Convert between Python and MATLAB dictionaries

When passing data from Python to MATLAB, you can more easily convert Python dictionaries to MATLAB dictionaries. To convert a Python `dict` to a MATLAB dictionary, first convert the Python `dict` to a `matlab.dictionary` object, and then pass the `matlab.dictionary` object to MATLAB.

Additionally, MATLAB can now pass a MATLAB dictionary to Python as a `matlab.dictionary` object. A `matlab.dictionary` object behaves like a Python `dict`.

For more information, see [Use MATLAB Dictionaries in Python](#).

## .NET: Cross-platform support for .NET 6 and higher

In addition to existing support on Microsoft Windows platforms, MATLAB now supports .NET Runtime 6.0 and higher on macOS and Linux platforms. For more information, see [System Requirements for Using MATLAB Interface to .NET](#).

## .NET: Support for .NET 8

MATLAB now supports .NET 8, in addition to existing support for the Microsoft .NET Framework. For supported version information, see [MATLAB Interfaces to Other Languages](#).

## Compilers: Support for Intel C++ icx and Fortran ifx

MATLAB supports two additional compilers for building C and C++ interfaces, MEX files, and standalone MATLAB engine and MAT-file applications on Windows platforms.

| Support | Compiler                              | Platform |
|---------|---------------------------------------|----------|
| Added   | Intel oneAPI DPC++/C++ Compiler (icx) | Windows  |
| Added   | Intel Fortran Compiler (ifx)          | Windows  |

For continued support for building your applications, consider upgrading to a supported compiler. For an up-to-date list of supported compilers, see [Supported and Compatible Compilers](#).

## Hardware Support

### **Arduino Hardware: Support for Arduino UNO R4 WiFi and Minima boards**

You can now use the MATLAB Support Package for Arduino Hardware to communicate with the Arduino UNO R4 WiFi board over USB and WiFi and the Arduino UNO R4 Minima board over USB. Use the `arduino` object to connect to the two boards.

However, you cannot use the support package to interface the two boards with a servo motor, Adafruit Motor Shield V2, motor carrier, serial devices, or sensors. Also, this feature is available only when you use the MATLAB Support Package for Arduino Hardware in the installed version of MATLAB.

### **Arduino Hardware: Use MATLAB Support Package for Arduino Hardware in MATLAB Online through web browser**

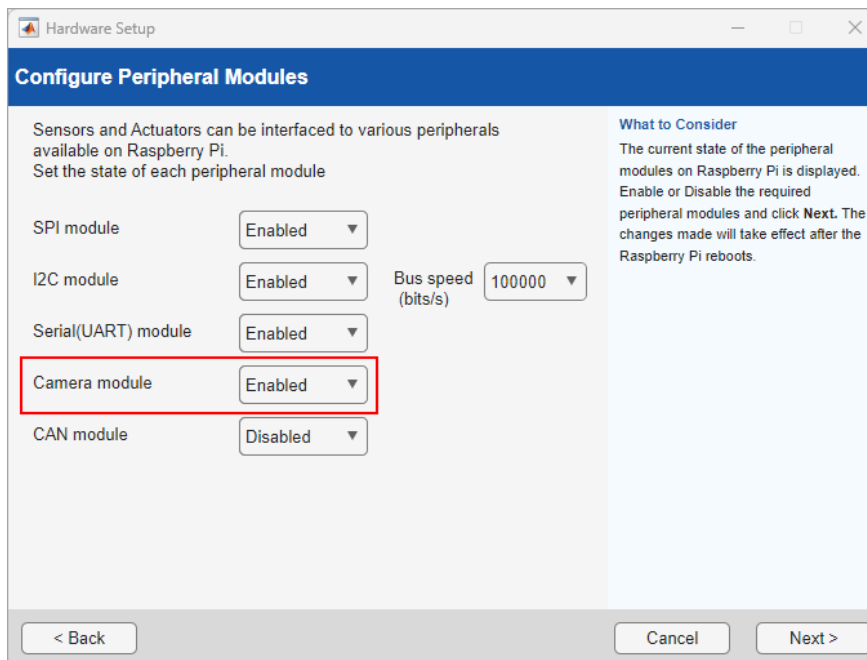
Starting in R2024b, you can use MATLAB Support Package for Arduino Hardware in MATLAB Online to communicate with Arduino hardware through your web browser. You must install the MATLAB Connector™ on your host computer to use the support package in MATLAB Online. For more information, see “Get Started with MATLAB Online for Arduino”.

### **Raspberry Pi Hardware: Support for Debian 12 Bookworm**

You can now use the MATLAB Support Package for Raspberry Pi Hardware with boards that have the Debian 12 Bookworm Raspberry Pi operating system installed.

### **Raspberry Pi Hardware: libcamera support on camera boards with Bullseye and Bookworm operating systems**

The `libcamera` packages are now installed as part of the MATLAB Support Package for Raspberry Pi Hardware for the Bullseye and Bookworm operating systems. The `libcamera` functionality is available when you enable the **camera module** peripheral in the Hardware Setup tool.



As part of the libcamera update, the cameraboard object no longer displays the ImageEffect and VideoStabilization properties for the Bullseye and Bookworm operating systems.

## Raspberry Pi Hardware: Support for Raspberry Pi 5

You can now use the MATLAB Support Package for Raspberry Pi Hardware with the Raspberry Pi 5 board. This support is available in the installed version of MATLAB and MATLAB Online.

However, you cannot use the Raspberry Pi 5 board with a camera board, servo motor, or pulse width modulation (PWM) signals.

## Raspberry Pi Hardware: Support for Camera with Raspberry Pi 5 (December 2024 Version 24.2.3)

Starting in version 24.2.3 of MATLAB Support Package for Raspberry Pi Hardware, you can interface a camera board and an ArduCam multi-camera adapter module with the Raspberry Pi 5 board. This support is available for both MATLAB Online and the desktop environment.

### ⚠️ Functionality being removed or changed

#### **enableI2C, disableI2C, enableSPI and disableSPI functions have been removed**

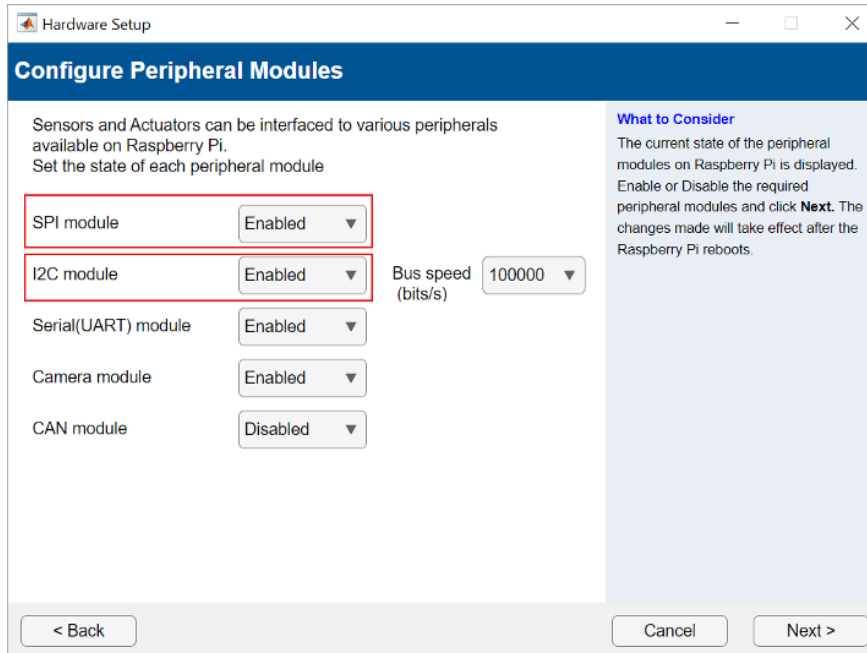
##### *Errors*

These functions have been removed:

- enableI2C
- disableI2C
- enableSPI

- `disableSPI`

Instead, in the Hardware Setup tool, use the Configure Peripheral Modules window to enable or disable I2C and SPI interfaces on Raspberry Pi.



# R2024a

---

**Version: 24.1**

**New Features**

**Bug Fixes**

**Compatibility Considerations**

## Environment

### ★ Editor Spell Checker: Check spelling in text and comments in MATLAB code files

In the Editor and Live Editor, you can check for spelling issues in text lines and comments. Spell checking is supported for US English in MATLAB code files (.m) and live code files (.mlx).

To enable spell checking, go to the **View** tab and click the **Spelling** button on. Words with a potential spelling issue are underlined in blue. To resolve the issue, click the word and select one of the suggested corrections. You also can choose to ignore the issue or add the flagged word to your local dictionary. To navigate between issues using the keyboard, use **Alt+F7** and **Alt+Shift+F7**.

To remove words from your local dictionary, go to your MATLAB preferences folder (the folder returned when you run `prefdir`) and edit the file `dict/en_US_userDictionary.tdi`.

### Live Editor Export: Interactively customize export options when exporting to Markdown files and Jupyter notebooks

You can interactively export live scripts and functions to Markdown files and Jupyter® notebooks and customize export options. For example, you can change the resolution and format of figures in the converted file (requires the files to be run before exporting).

To interactively export to a Markdown file or Jupyter notebook, on the **Live Editor** tab, click **Export** and select **Export to Markdown** or **Export to Jupyter Notebook**.

For more information, see [Ways to Share and Export Live Scripts and Functions](#).

### Live Editor Controls: Add date pickers to live scripts

Add a date picker to your live script to select a date interactively. To add a date picker, go to the **Live Editor** tab, and in the **Code** section, select **Control > Date Picker**.



For more information, see [Add Interactive Controls to a Live Script](#).

### Live Editor Controls: Replace with similar controls

In live scripts, you can replace some controls with other controls that have similar functionality. For example, you can replace a check box with a state button, and a numeric slider with a numeric spinner. Replacing one control with another preserves relevant configuration values, such as range and default values.

To replace a control with another, right-click the control and select **Replace with Control Name**. You also can select the control in the live script, go to the **Live Editor** tab, click **Control**, and select from the available options. The **Control** menu shows only the controls that can replace the selected control.

For more information, see [Add Interactive Controls to a Live Script](#).

## Live Editor Tasks: Create Live Editor task class from template

You can use a template to create a Live Editor task class. To create a Live Editor task class from a template, go to the **Editor** or **Live Editor** tab and select **New > Live Task Class**. Customize the template and then use the `matlab.task.configureMetadata` function to configure the task and make it available in the Live Editor.

For more information, see [Live Editor Task Development Overview](#).

## Live Editor Output: View table, timetable, structure array, and cell array output with enhanced readability

Tables, timetables, structure arrays, and cell arrays in the output of the Live Editor have enhanced readability. The updated displays have a streamlined look, including a simplified, borderless design and row striping for easier data review and analysis.

`outages = 1468x5 timetable`

|   | OutageTime           | Region    | Loss     | Customers  | RestorationTime      | Cause           |
|---|----------------------|-----------|----------|------------|----------------------|-----------------|
| 1 | 01-Feb-2002 12:18:00 | SouthWest | 458.9772 | 1.8202e+06 | 07-Feb-2002 16:50:00 | winter storm    |
| 2 | 23-Jan-2003 00:49:00 | SouthEast | 530.1399 | 2.1204e+05 | NaT                  | winter storm    |
| 3 | 07-Feb-2003 21:15:00 | SouthEast | 289.4035 | 1.4294e+05 | 17-Feb-2003 08:14:00 | winter storm    |
| 4 | 06-Apr-2004 05:44:00 | West      | 434.8054 | 3.4037e+05 | 06-Apr-2004 06:10:00 | equipment fault |
| 5 | 16-Mar-2002 06:18:00 | MidWest   | 186.4368 | 2.1275e+05 | 18-Mar-2002 23:23:00 | severe storm    |
| 6 | 18-Jun-2003 02:49:00 | West      | 0        | 0          | 18-Jun-2003 10:54:00 | attack          |
| 7 | 20-Jun-2004 14:39:00 | West      | 231.2947 | NaN        | 20-Jun-2004 19:16:00 | equipment fault |
| 8 | 06-Jun-2002 19:28:00 | West      | 311.8607 | NaN        | 07-Jun-2002 00:51:00 | equipment fault |
| 9 | 16-Jul-2003 16:23:00 | NorthEast | 239.9328 | 4.9434e+04 | 17-Jul-2003 01:12:00 | fire            |

## Find and Replace Dialog Box: Disable wrap-around search

You can disable the wrap-around search behavior in the Find and Replace dialog box using settings. To disable the wrap-around search behavior, use this code.

```
s = settings;
s.matlab.editor.find.WrapAround.PersonalValue = 0;
```

For more information including a list of additional search settings, see [matlab.editor Settings](#).

In MATLAB Online, you also can use preferences to disable the wrap-around search behavior. On the **Home** tab, in the **Environment** section, click  **Preferences**. Select **MATLAB > Editor/Debugger > Find and Replace** and in the **Search behavior** section, clear the **Wrap around** option. For more information, see [Editor/Debugger Preferences](#).

## Editor Indenting: Configure automatic indenting behavior

You can configure the behavior of automatic indenting in the Editor and Live Editor. For example, you can specify whether automatic indenting formats the entire document and whether it adds white space to empty lines.

To configure the behavior of automatic indenting, use settings. For example, enable formatting the entire document when automatic indenting is enabled.

```
s = settings;  
s.matlab.editor.indent.SmartIndentEntireDocument.PersonalValue = 1;
```

For more information, see `matlab.editor` Settings.

In MATLAB Online, you also can use preferences to configure automatic indenting behavior. On the **Home** tab, in the **Environment** section, click  **Preferences**. Select **MATLAB > Editor/Debugger > Indenting** and in the **Smart indenting** section, adjust the automatic indenting options as needed. For more information, see Editor/Debugger Preferences.

## Editor YAML and Markdown Support: View and edit YAML and Markdown files with syntax highlighting and automatic indenting while typing

The Editor displays YAML files with syntax highlighting for keywords, variables, strings, numbers, and comments. In addition, the Editor automatically indents lines of code in YAML files while you type.

The Editor also displays Markdown files with syntax highlighting for headings, horizontal rules, lists, code blocks, comments, and other entries. You can change the syntax highlighting colors for Markdown files using settings. For example, change the color of headings in Markdown files to purple.

```
s = settings;  
s.matlab.editor.language.markdown.HeadingColor.PersonalValue = [183 70 255];
```

For more information, see `matlab.editor` Settings.

In MATLAB Online, you also can use preferences to configure how the Editor displays YAML and Markdown files. On the **Home** tab, in the **Environment** section, click  **Preferences**. Select **MATLAB > Editor/Debugger > Other Languages** and then **Markdown** or **YAML**, and adjust the options as needed.

## Accessibility in MATLAB Online: Access the Command Window scroll buffer region using the keyboard

In MATLAB Online, you can access previous commands and output in the Command Window scroll buffer region using the keyboard. To enter the scroll buffer region, press **Shift+Tab**. The cursor moves to the last output or command. Use the up arrow and down arrow keys to move up and down between previous commands and output within the scroll buffer region. To move back to the command prompt, press **Tab**.



## Accessibility in MATLAB Online: Use a screen reader to create and edit live scripts and functions in the Live Editor

In MATLAB Online, you can now use a screen reader to interact with live scripts and functions. When using a screen reader in a live script or function, you hear additional information about text lines, code lines, and section breaks that help provide context. For example:

- When you are on a text line, you hear information about formatting, hyperlinks, tables, images, and equations within the line.
- When you are on a code line, you hear the line number, content, and whether the line has output associated with it.
- When you are on a code line containing a control or task, you hear the name of the control or task as well as its current value.

For more information, see [Use a Screen Reader in MATLAB Online](#).

## Add-Ons in MATLAB Online: Install and manage add-ons using Add-Ons panel

Use the Add-Ons panel to find, install, and manage add-ons in MATLAB Online. To open the Add-Ons panel, click the Add-Ons button  in the sidebar on the left side of the MATLAB Online desktop.

For more information, see [Get and Manage Add-Ons](#).

## Cloud Storage in MATLAB Online: View your connections using Connections panel

Use the Connections panel to view and navigate to your available connections in MATLAB Online, including MATLAB Drive. To open the Connections panel, in the sidebar on the left side of the MATLAB Online desktop, click the Open more panels button (\*\*\*) and select the Connections panel.

## Cloud Storage in MATLAB Online: Access your OneDrive files (June 2024)

In MATLAB Online, you can access your Microsoft OneDrive account and its contents by connecting to your account using the Connections panel. Once connected, you can view and edit your stored files and save new files to your account directly from the MATLAB Online desktop environment. For more information, see [Use MATLAB Online to Access Files in Microsoft OneDrive](#).

## Editor in MATLAB Online: Resolve conflicts with externally modified files

When saving changes to a file open in the MATLAB Online Editor, if the file has been modified outside of MATLAB, you can compare the two versions of the file before saving to avoid losing content.

## Editor in MATLAB Online: Reopen last closed file

In MATLAB Online, you can reopen the last closed file in the Editor and Live Editor. To reopen the file, go to the **Home** tab and select **Open > Reopen <filename>**.

## Find Files in MATLAB Online: Search for and replace text in filenames and file content

You can search for and replace text in filenames and file content using the Find Files panel in MATLAB Online. Replacing text is supported in text files (.txt) and MATLAB code files (.m).

To open the Find Files panel, click the Find Files button  in the sidebar on the left side of the MATLAB Online desktop. To limit the search to only filenames or file content, click the Filters button and clear the **Filename** or **Content within files** search type.

For more information, see Find Files.

## MATLAB Drive: Get recipient suggestions when sharing (April 2024)

When you share a folder using a personal invitation, MATLAB Drive suggests people to share the folder with based on who you have shared with most recently and frequently. Only people who have previously accepted an invitation from you are included in the list. You can filter the list of suggestions by entering the first name, last name, or email address of your intended recipient. For more information, see Share Folders Using MATLAB Drive.

## MATLAB Drive: Updated user interface with collapsible sidebar (May 2024)

MATLAB Drive Connector is now renamed to MATLAB Connector and includes several visual updates including a new sidebar that can be collapsed when needed. For more information about MATLAB Connector, see Use MATLAB Connector to Manage Your Files.

## Product Version Number: All products now use consistent version number based on release

Starting in R2023b, all products use a consistent version number based on the release. The version number includes the last two digits of the year followed by either .1 or .2, where .1 indicates an “a” release and .2 indicates a “b” release. For example, all products in R2023b have a version number of 23.2. Changes to the version number no longer indicate a major product update.

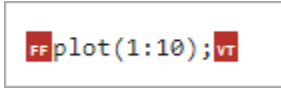
In previous releases, each individual product used a different version number. This change does not affect R2023a or earlier releases.

## ▲ Functionality being removed or changed

### Form feed characters and vertical tabs display in the Editor

*Behavior change*

Form feed characters and vertical tabs now display in the Editor as symbols. Previously, form feed characters and vertical tabs were not displayed.



### **AcceptHTML and EmbedImages name-value arguments of the export function are false by default**

#### *Behavior change*

For the `export` function, the default value of the `AcceptHTML` and `EmbedImages` name-value arguments is now `false`. Previously, the default value of the name-value arguments was `true`. Set `AcceptHTML` to `true` to include HTML markup when you convert live scripts or functions to Markdown files. Set `EmbedImages` to `true` to embed images and figures when you convert live scripts or functions to Markdown files or Jupyter notebooks.

# Language and Programming

## ★Local Functions: Define functions anywhere in scripts and live scripts

Local functions can be added anywhere in scripts and live scripts except within conditional contexts, such as if statements or for loops. Each local function must begin with its own function definition statement and end with the end keyword. For more information, see Add Functions to Scripts.

## createArray Function: Generate arrays of specified class and size

The `createArray` function enables you to create arrays of almost any MATLAB class and size. Specify a class name or prototype, a fill value, or a combination of these arguments to populate the array. For example, create a 1-by-3 array based on a complex-valued prototype of class `single` and a fill value of `NaN`.

```
X = createArray(1,3,Like=single(1+1i),FillValue=NaN)

X =

    1×3 single row vector

    NaN + 0.0000i    NaN + 0.0000i    NaN + 0.0000i
```

## MATLAB Language Terminology: Packages are now called namespaces

The MATLAB language feature known as a package is now called a namespace. Namespaces are used to organize code, and in MATLAB, they are defined by folders that begin with a `+` symbol. As of R2024a, the terminology is updated in both the documentation and software. The behavior remains the same.

## Metaclass Renaming: Metaclasses and class members have new names

In R2024a, metaclasses are contained in a new namespace, `matlab.metadata`. Some of the classes, properties, and methods also have new names. Their behavior remains the same.

| Class Name in R2023b and Earlier  | New Name as of R2024a                          |
|-----------------------------------|------------------------------------------------|
| <code>meta.package</code>         | <code>matlab.metadata.Namespace</code>         |
| <code>meta.class</code>           | <code>matlab.metadata.Class</code>             |
| <code>meta.property</code>        | <code>matlab.metadata.Property</code>          |
| <code>meta.MetaData</code>        | <code>matlab.metadata.MetaData</code>          |
| <code>meta.event</code>           | <code>matlab.metadata.Event</code>             |
| <code>meta.method</code>          | <code>matlab.metadata.Method</code>            |
| <code>meta.DynamicProperty</code> | <code>matlab.metadata.DynamicProperty</code>   |
| <code>meta.EnumeratedValue</code> | <code>matlab.metadata.EnumerationMember</code> |
| <code>meta.Validation</code>      | <code>matlab.metadata.Validation</code>        |

| Class Name in R2023b and Earlier | New Name as of R2024a                 |
|----------------------------------|---------------------------------------|
| meta.ArrayDimension              | matlab.metadata.ArrayDimension        |
| meta.FixedDimension              | matlab.metadata.FixedDimension        |
| meta.UnrestrictedDimension       | matlab.metadata.UnrestrictedDimension |

MATLAB will continue to recognize the old names. However, code that relies on string comparisons to identify metaclasses might need to be updated to continue to work as expected. See the Version History section on the individual class pages for more information.

## Class Diagram Viewer: Show associations for classes

The Class Diagram Viewer app can now identify and display associations between classes. When the type of a class property is explicitly defined as another class using class validation, the Class Diagram Viewer can show this association by connecting the property to the class card of its type with an arrow. Use the **Associations** check box in the toolbar of the app or the `ShowAssociations` property of `matlab.diagram.ClassViewer` to toggle associations on and off in the diagram.

## Class Constructor: `?.MyClass` syntax accepts immutable properties as name-value arguments

In previous releases, the `?.MyClass` syntax could be used in functions and methods to accept class properties as name-value arguments, but the syntax did not work with immutable properties. To enable immutable properties to be set as name-value arguments, you had to use a separate struct in the arguments block. For example, `opts.?.ExampleClass` could handle `Prop2` and `Prop3`, but a separate struct, `opts.Prop1`, was required for `Prop1`.

```
classdef ExampleClass
    properties (SetAccess=immutable)
        Prop1
    end
    properties (SetAccess=public)
        Prop2
        Prop3
    end

    methods
        function obj = ExampleClass(opts)
            arguments
                opts.Prop1
                opts.?.ExampleClass
            end
        end
    end
end
```

Starting in R2024a, `opts.?.ExampleClass` works with immutable properties as well, so you no longer need to handle `Prop1` separately. This syntax for using immutable properties as name-value arguments only works in the class constructor.

## MATLAB backgroundPool Function: save function supported in background threads

Use the `save` function in thread-based environments, including the MATLAB backgroundPool. Saving Version 7.3 MAT-files is not supported.

## hasActiveValue Function: Verify that setting has active value

The `hasActiveValue` function returns a logical value indicating whether a given setting currently has an active value. For example, this code checks if the maximum width setting for comments has an active value. If it does not, it sets the personal value for the setting, which also updates the active value.

```
s = settings;  
if ~hasActiveValue(s.matlab.editor.language.matlab.comments.MaxWidth)  
    s.matlab.editor.language.matlab.comments.MaxWidth.PersonalValue = 80  
end
```

## MATLAB Debugger: View private, protected, and hidden class property values using display and details

The MATLAB debugger now offers additional access to private, protected, and hidden class properties and their values using the `display` and `details` functions. For example, set a breakpoint at the end of the `incPrivateProp` method of the `DebugDemo` class.

```
classdef DebugDemo  
    properties  
        PublicProp = 0  
    end  
    properties (Access=private)  
        PrivateProp = 0  
    end  
    methods  
        function obj = incPrivateProp(obj)  
            obj.PrivateProp = obj.PrivateProp + 1;  
        end  
    end  
end
```

Construct an instance `test` of the `DebugDemo` class, then call the `incPrivateProp` method.

```
test = DebugDemo;  
test.incPrivateProp
```

Once you enter the debugger at the breakpoint, calling `display` on the object returns public properties and their values.

```
K>> obj % Or display(obj)
```

```
obj =  
  
    DebugDemo with properties:  
  
    PublicProp: 0
```

Show all accessible properties of DebugDemo

Click the Show all link to show all properties and values, including private, protected, and hidden properties.

PublicProp: 0  
PrivateProp: 1

Open Class Diagram Viewer for DebugDemo

Calling details on the object also shows all properties and values, including private, protected, and hidden properties.

K>> details(obj)

Debug view of DebugDemo with properties:

PublicProp: 0  
PrivateProp: 1

Methods

## Function Input Arguments: Automatically add input arguments to arguments block

When you complete an arguments block using automatic suggestions and completions, MATLAB automatically adds an entry for each input argument in the function declaration as well as an end keyword at the end of the block.

## Validation Functions: mustBeUnderlyingType accepts multiple data types

mustBeUnderlyingType now accepts multiple data types for the typename input argument.

## ⚠️ Functionality being removed or changed

### Some matlab.diagram.ClassViewer member names have changed

*Behavior change*

MATLAB now refers to packages as namespaces. Several members of matlab.diagram.ClassViewer have been renamed to reflect this terminology change. Their behavior remains the same.

| Member Name in R2023b and Earlier                   | New Name as of R2024a  |
|-----------------------------------------------------|------------------------|
| Constructor: Packages name-value argument           | Namespaces             |
| Constructor: IncludeSubpackages name-value argument | IncludeInnerNamespaces |
| Property: ShowPackageNames                          | ShowNamespace          |

| Member Name in R2023b and Earlier                                                            | New Name as of R2024a               |
|----------------------------------------------------------------------------------------------|-------------------------------------|
| Method: <code>importClassesFrom</code> , Packages name-value argument                        | Namespaces                          |
| Method: <code>importClassesFrom</code> , <code>IncludeSubpackages</code> name-value argument | <code>IncludeInnerNamespaces</code> |

**Negating a `matlab.lang.OnOffSwitchState` enumeration member preserves class**  
*Behavior change*

Before R2024a, negating a `matlab.lang.OnOffSwitchState` enumeration member returned a logical value. MATLAB now preserves the class of `matlab.lang.OnOffSwitchState` members when you negate them. See `matlab.lang.OnOffSwitchState` for more information.

**Change to precedence for constructors in class folders**  
*Warns*

When a class and a function on the path have the same name, MATLAB currently gives precedence to the class constructor if the class is defined inside a class folder (@-folder). The constructor takes precedence even if the function appears before the class on the path.

In a future release, this precedence rule will be removed. In other words, if a function appears before a class of the same name on the path, the function will have precedence regardless of whether the class is defined in a class folder. To give precedence to the class constructor, update your path configuration. To give precedence to the other function, remove the class from the path or rename one of the conflicting items. MATLAB currently displays a warning when a name is resolved to a class constructor instead of a function earlier on the path. For more information, see *Precedence of Classes in @-folders over Functions of the Same Name Will Be Removed in a Future Release*.

**Defining classes and namespaces: Using `schema.m` will not be supported in a future release**  
*Still runs*

Support for classes and namespaces defined using `schema.m` files will be removed in a future release. Replace existing schema-based classes with classes defined using the `classdef` keyword.

**MATLAB Identifiers: The maximum length of MATLAB identifiers will increase in a future release**  
*Behavior change in future release*

The maximum length of MATLAB identifiers will increase in a future release. This change allows variables, functions, classes, and many other entities to have names that exceed 63 characters. Code that depends on identifiers having a maximum length of 63 characters may no longer run as expected.

**MATLAB Scripts: Scripts without executable code will now error**  
*Behavior change*

Scripts without executable code will now error. For example, the following script contains no executable code:

```
% this is a script file
;
```



```
function foo  
end
```

Prior to R2024a this script would be considered valid. In R2024a, attempting to run this script would result in an error being thrown.

## Data Analysis

### **islocalmax2 and islocalmin2 Functions: Find local extrema in two dimensions**

Find local maxima or minima and the prominence of each extremum in two-dimensional data sets using the `islocalmax2` or `islocalmin2` function, respectively.

### **clip Function: Clip values to specified range**

Clip the values in an array, table, or timetable to a specified range using the `clip` function. `clip` replaces values less than the lower bound with the lower bound value and replaces values greater than the upper bound with the upper bound value.

When clipping table or timetable data, you can:

- Specify tabular variables to clip by using the `DataVariables` name-value argument.
- Append or replace tabular variables with variables containing clipped data by using the `ReplaceValues` name-value argument.

### **mean and median Functions: Compute weighted statistics**

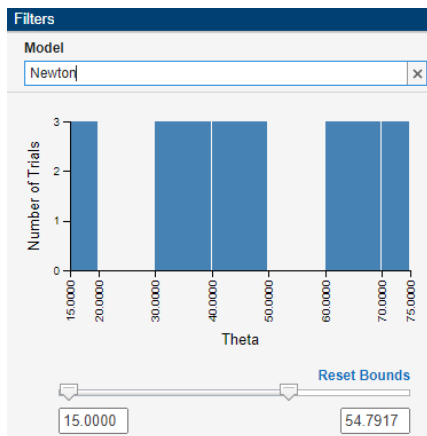
Compute the weighted mean or the weighted median of input data by specifying the `Weights` parameter of the `mean` or `median` function, respectively.

### **iqr Function: Return first and third quartiles**

The `iqr` function can now return the first and third quartiles of the input data by using a second output argument `q`. The first element of `q` along the smallest operating dimension is the first quartile of the input data, and the second element of `q` along the smallest operating dimension is the third quartile.

### **Experiment Manager: Apply and reset filters for numeric, character array, and string variables in results table**

In the Experiment Manager app, you can apply or reset a filter for a variable in the results table. Select **Filters** in the toolbar and configure the corresponding section of the **Filters** pane. Set the bounds for a numeric variable, or specify the contents of a character array or string variable. Previously, you could apply a filter only to numeric variables and you could not reset the numeric filter.



## Experiment Manager: Add supporting files

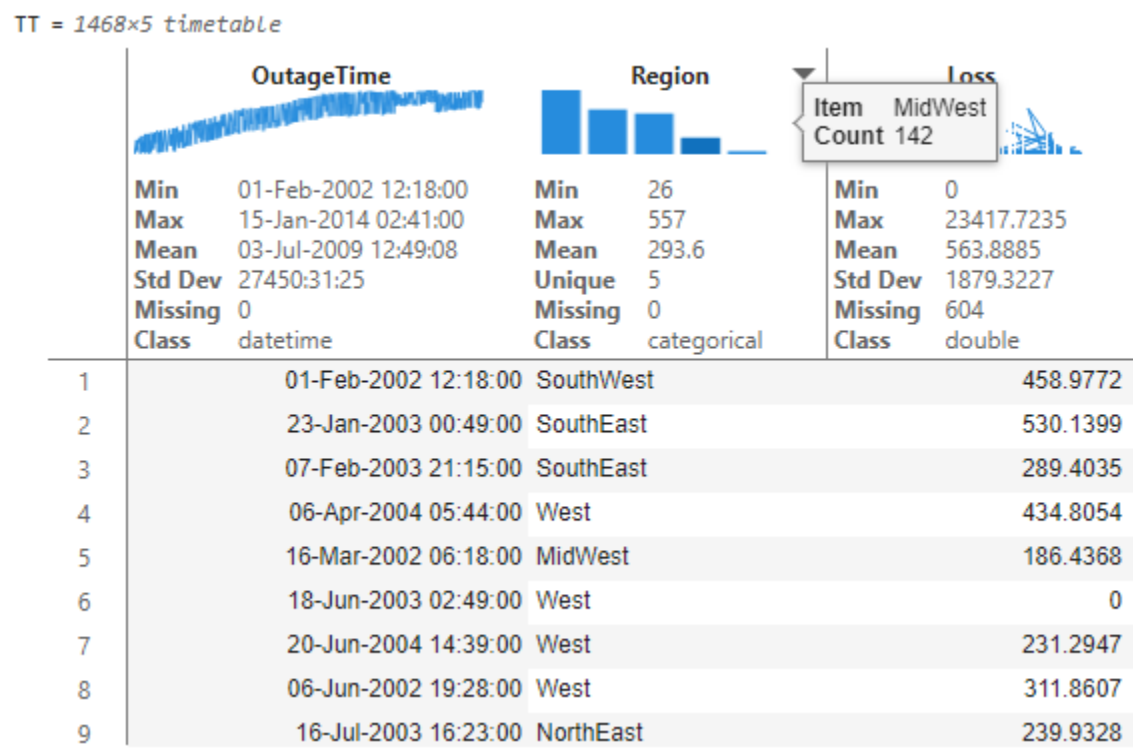
In the Experiment Manager app, add files required by your experiment using the **Supporting Files** section of the experiment definition tab.

After you run an experiment, the **Detected Files** section lists detected supporting files. If Experiment Manager does not detect some of your supporting files, your trials will produce an error. You can manually select files to include by clicking **Add** in the **Additional Files** section. You can also update the **Detected Files** section by clicking **Refresh**.

## Live Editor Output: View sparklines and summary statistics for tables

For tables or timetables in the output of the Live Editor, you can display sparklines and summary statistics to quickly visualize and interpret the data in each variable.

To display sparklines or summary statistics, right-click the table and show sparklines or summary statistics using the context menu. To show more information related to specific points, point to a sparkline.



Smooth Data Live Editor Task: Return moving window size

The Smooth Data task in the Live Editor can now return the moving window size. If you specify the moving window size, the returned window size matches the input value. If you do not specify the moving window size, then the returned window size is heuristically determined based on the input data. If the moving window is asymmetric, then the returned window is a two-element vector of nonnegative integers [b f], where the window contains the current element, b elements backward, and f elements forward.

fillmissing and filloutliers Functions: Define missing or outlier locations using table

Define the locations of missing entries or outliers using a table. Specify the MissingLocations name-value argument of the fillmissing function or the OutlierLocations name-value argument of the filloutliers function.

Previously, fillmissing and filloutliers supported only array indicators for missing entries or outliers.

Group-Wise Computations: Apply multiple binning methods to grouping variable

For these functions, apply multiple binning methods to one grouping variable or vector by specifying a cell array of binning methods.

- pivot

- groupsummary
- groupcounts
- groupfilter
- grouptransform

## datetime Data Type: Specify time zone offset using duration value

You can specify a time zone offset for a `datetime` array by using a duration value.

To specify an offset when you use the `datetime` function, specify the `TimeZone` name-value argument.

```
t = datetime("now",TimeZone=hours(5))
```

To specify an offset after you create a `datetime` array, assign the offset to its `TimeZone` property.

```
t = datetime("now")
t.TimeZone = hours(5)
```

## categories Function: Return list of categories as categorical array or string array

When using the `categories` function, you can specify the `OutputType` name-value argument to return the list of categories as a categorical array or string array. By default, the function returns the list of categories as a cell array of character vectors.

## Unstack Table Variables Live Editor Task: Generate template code for custom aggregation function

In the Unstack Table Variables Live Editor Task, when you select the `Custom` function option for the **Aggregator for new table variable values** method, you can generate template code that can help you write your own custom aggregation function.

- 1 Select `Custom` function.
- 2 Select `Local` function from the next menu.
- 3 Click **New** to add a new local function to the script.
- 4 Select the name of the new local function from the next menu.

The Live Editor task generates template code and places it within the live script. To customize this code, edit it in the live script.

## Data Import and Export

### ★ Secrets in MATLAB Vault: Remove sensitive information from code

You can remove sensitive information, such as passwords, from code by using secrets and a local MATLAB vault. Your MATLAB vault is accessible only from the exact combination of your operating system account and local machine. Removing sensitive information increases the security of your code and allows you to share your code more easily. Create and manage secrets in your vault using these new functions:

- `setSecret` - Move sensitive information into a secret in your vault.
- `getSecret` - Retrieve a secret from your vault.
- `isSecret` - Determine if a secret exists in your vault.
- `removeSecret` - Remove a secret from your vault.
- `setSecretMetadata` - Set the metadata of a secret in your vault.
- `getSecretMetadata` - Retrieve the metadata of a secret in your vault.
- `listSecrets` - Display the names and metadata of all the secrets in your vault.

### SFTP: Specify remote current working folder at login

When you connect to an SFTP server using the `sftp` function, you can specify a remote current working folder by using the `StartingFolder` name-value argument.

### openedFiles Function: Get file identifiers of all open files

Get the file identifiers of all open files using the `openedFiles` function.

### HDF5 Interface: Import HDF5 data using Import Tool in MATLAB Online and Import Data Live Editor task

You can import HDF5 data by using the Import Tool app in MATLAB Online or by using the Import Data Live Editor task in live scripts. Using these options, you can:

- Visualize the structure and contents of HDF5 data interactively.
- Search for specific groups, datasets, and attributes.
- Graphically select and deselect groups, datasets, and attributes to import.
- Generate reusable MATLAB code to import HDF5 data.

### HDF5 Interface in MATLAB Online: Import HDF5 data by double-clicking HDF5 files

You can open the Import Tool app to import HDF5 data by double-clicking an HDF5 file in MATLAB Online.

## Serial and TCP/IP Connections: Label and Find Connections

You can use the following functions to find existing `serialport` and `tcpclient` connections and return object handles for these connections.

- `serialportfind`
- `tcpclientfind`

These commands can return all existing connections of the respective type or only those connections having specified property values. In particular, each interface has a new `Tag` property that you can use to label and find connections. For instance, the following command finds existing `serialport` connections whose `Tag` property is "Scope".

```
S = serialportfind(Tag="Scope");
```

Such tags are useful when you create a connection in one function and use a different function to perform operations on the connection.

## Image File Format Libraries: LibTIFF library upgraded to version 4.6.0

The LibTIFF library is upgraded to version 4.6.0.

## Scientific File Format Libraries: HDF4 library upgraded to version 4.2.16.2

The HDF4 library is upgraded to version 4.2.16.2.

## Scientific File Format Libraries: HDF5 library upgraded to version 1.10.11

The HDF5 library is upgraded to version 1.10.11.

## ⚠️ Functionality being removed or changed

### **`fopen("all")` syntax will be removed**

*Warns*

The `fopen("all")` syntax will be removed in a future release. To get the file identifiers of all open files, use the `openedFiles` function instead.

## Mathematics

### ★ode Object: Solve ODEs and perform sensitivity analysis using SUNDIALS solvers

ode objects support three new values for the Solver property: "cvodesStiff", "cvodesNonStiff", and "idas". The new solvers provide access to a subset of the SUNDIALS suite of differential equation solvers.

- Adjust the values of InitialStep, MaxStep, and MaxOrder used by the new solvers with the SolverOptions property of the ode object.
- Examine how sensitive the equations are to variations in parameter values using sensitivity analysis, supported only by the new SUNDIALS solvers. To perform sensitivity analysis, set the Sensitivity property of the ode object using an odeSensitivity object.

### pagelsqminnorm Function: Calculate minimum-norm least-squares solutions to systems of linear equations in N-D arrays

Calculate the minimum-norm least-squares solutions to the systems of linear equations defined by N-D arrays using the pagelsqminnorm function. In this context, the N-D arrays are treated as containers for several 2-D matrices.

### pagepinv Function: Calculate Moore-Penrose pseudoinverses of pages of N-D array

Calculate the Moore-Penrose pseudoinverses of the pages of an N-D array using the pagepinv function. In this context, the N-D array is treated as a container for several 2-D matrices.

### scatteredInterpolant Object: Extend interpolation boundary into extrapolation domain

When using scatteredInterpolant, you can specify the extrapolation method as "boundary". The "boundary" extrapolation method extends the values on the scatteredInterpolant interpolation boundary into the extrapolation domain. This method mitigates extreme extrapolation values and maintains continuity between the extrapolation and interpolation domains.

### polydiv Function: Divide polynomials using long division

Use the polydiv function to divide two polynomials using long division.

The syntax `[q,r] = polydiv(b,a)` is equivalent to deconvolving vector `a` out of vector `b` using the polynomial long division method, such that `b = conv(a,q) + r`. However, using polynomial long division to find deconvolutions can result in numerically unstable results. For more numerically stable computations when finding the deconvolutions, use the least-squares method instead, as provided in the `deconv` function.



## polyfit Function: Return R-squared value

The error estimation structure `S`, returned as an optional second output of the `polyfit` function, contains a new field named `rsquared`. `rsquared` is the coefficient of determination, or (unadjusted) R-squared value. Use `S` with the `polyval` function to obtain error estimates.

## polyshape Object: Control when union and intersect methods simplify output

You can control when the `union` and `intersect` methods of `polyshape` simplify their output by specifying the new `Simplify` name-value argument, which can be either `true` (default) or `false`. By default, the methods return a well-defined polygon by resolving boundary intersections and improper nesting and also remove duplicate points and degeneracies. However, you can specify `Simplify=false` to gain performance when performing a series of unions or intersections. In this case, you can simplify the output once at the end either by specifying `Simplify=true` in the final method call or by using the `simplify` method on the final output polygon.

## spdiags Function: Specify diagonal elements as matrix, vector, or scalar

The `spdiags` function now supports implicit expansion for the diagonal elements argument `Bin`. You can specify the diagonal elements as a matrix, vector, or scalar, and the function expands the values if the argument is specified as a vector or scalar.

Before R2024a, you can replace the diagonal elements of `m`-by-`n` matrix `A` with the columns of a matrix `Bin` only if `Bin` has at least `min(m,n)` rows. Starting in R2024a, the `spdiags` function accepts the `Bin` argument as a matrix, vector, or scalar.

## sub2ind Function: Specify subscript inputs as scalars and vectors

The `sub2ind` function now supports scalar expansion and accepts a mix of scalars and vectors for subscript inputs. For example, `sub2ind(sz,[1 2 3],2)` is now the same as `sub2ind(sz,[1 2 3],[2 2 2])`. Previously, the subscript inputs were required to be the same size.

## ⚠️ Functionality being removed or changed

### colon operator will no longer support nonscalar operands

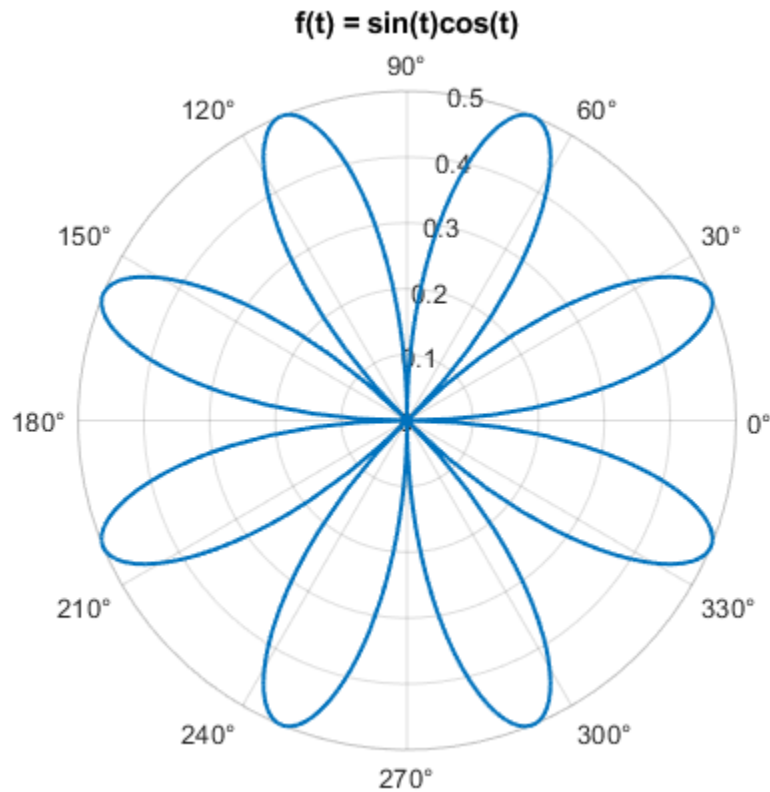
*Warns*

`colon` now issues a warning when creating vectors if one or more operands are not scalar. For example, expressions like `[1 2 3]:2:10` now warn. This warning will become an error in a future release, removing support for nonscalar operands. Previously, `colon` used the first element of any nonscalar operands to evaluate the expression.

## Graphics

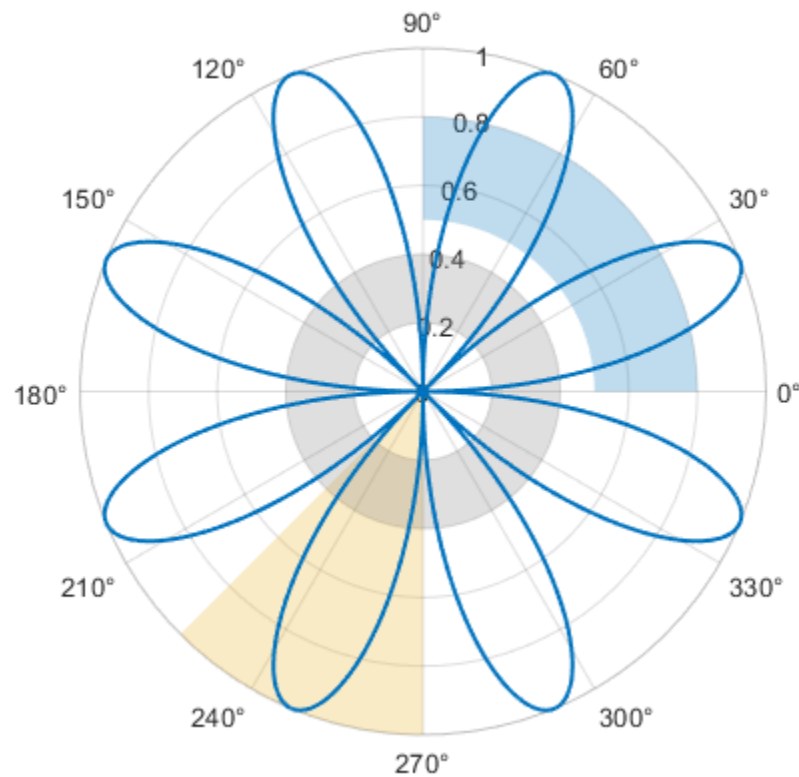
### **fpolarplot Function: Plot mathematical expressions in polar coordinates**

Plot mathematical expressions in polar coordinates by using the `fpolarplot` function. This function replaces the `ezpolar` function.



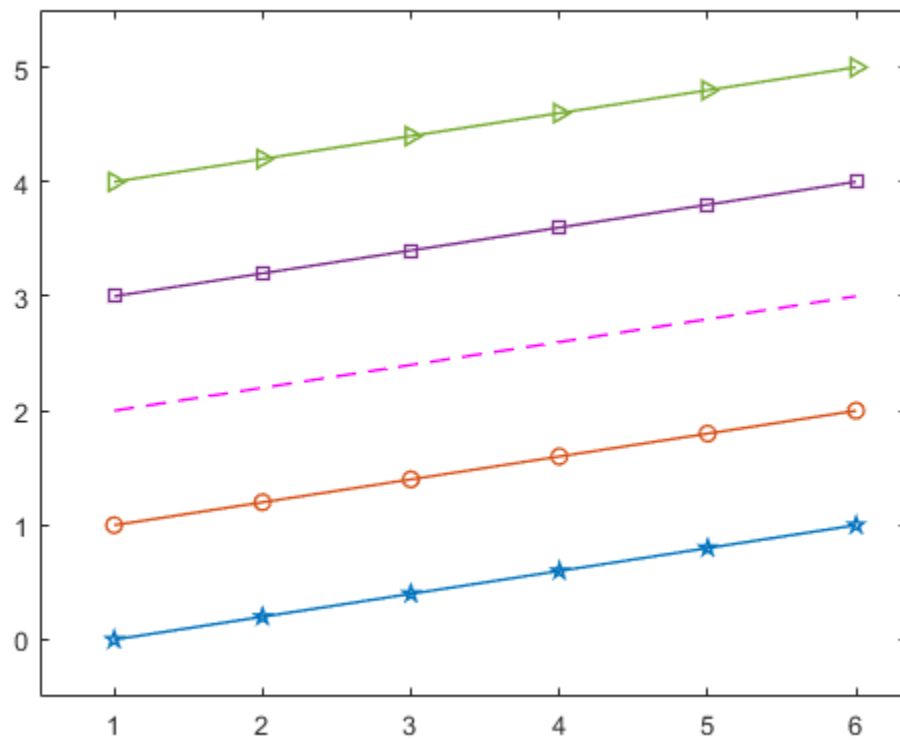
### **Polar Plots: Create filled regions in polar coordinates**

Create different types of filled regions in polar coordinates using the `thetaregion`, `radiusregion`, and `polarregion` functions. Filled regions are useful for highlighting one or more sections of a plot.



## **linestyleorder Function: Control line styles when plotting multiple data series**

Control the line styles of your plots by using the `linestyleorder` function. When you call this function, MATLAB uses the specified line styles for any new plots you create, and it updates the existing plots in the specified figure or axes.



## rgb2hex and hex2rgb Functions: Convert between RGB triplets and hexadecimal color codes

Use the `rgb2hex` function to convert RGB triplets to hexadecimal color codes. Use the `hex2rgb` function to convert hexadecimal color codes to RGB triplets. Both functions support RGB values as normalized single and double values as well as 8- and 16-bit unsigned integer values. Both functions also support three- and six-digit hexadecimal codes.

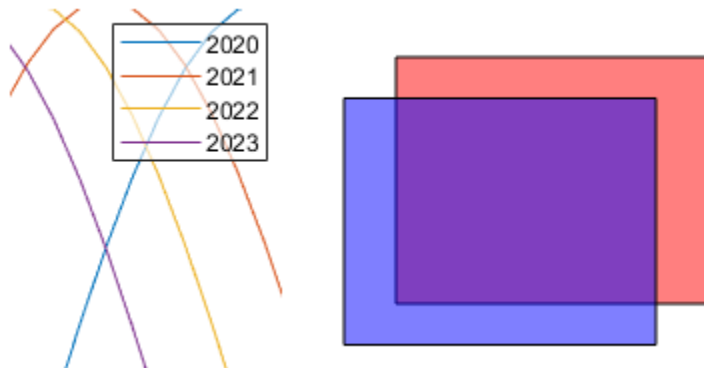
## Quiver Plots: Access the multiplier used to scale arrows

Query the `ScaleFactor` property of a Quiver object to access the value that MATLAB uses to scale the magnitudes of the arrows in your plot. The scale factor is useful when you want to maintain the scale between the arrows and other graphics objects in the axes.

## Legends and Rectangles: Control transparency of legend boxes and rectangle faces

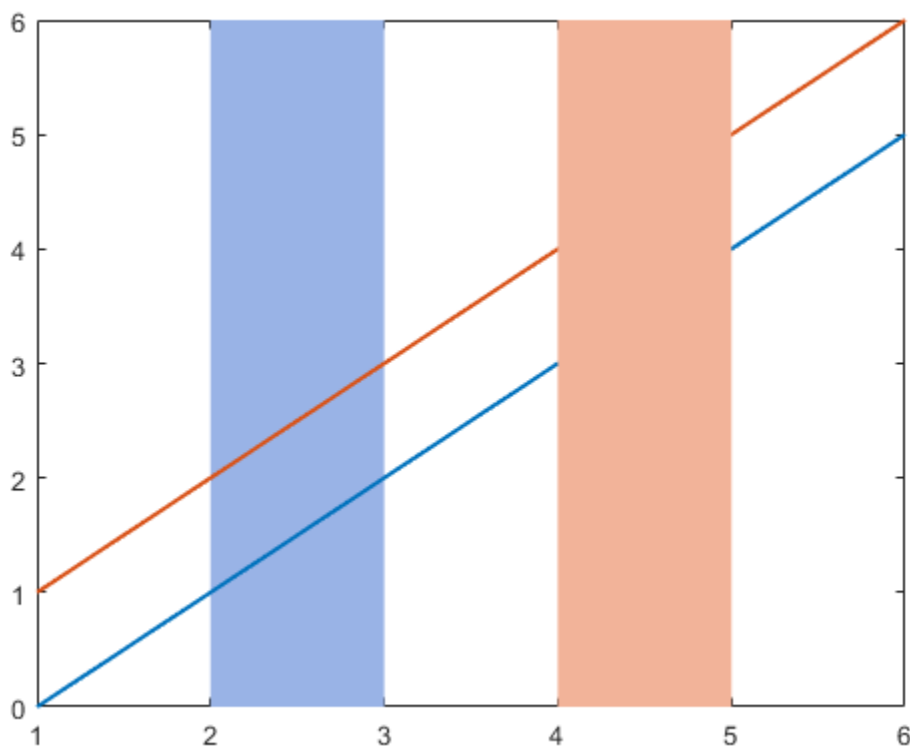
You can control the transparency of legend boxes and rectangles:

- Control the background transparency of a legend box by setting the `BackgroundAlpha` property of the `Legend` object to a value between 0 (completely transparent) and 1 (opaque).
- Control the transparency of a rectangle by setting the `FaceAlpha` property of the `Rectangle` object to a value between 0 and 1.



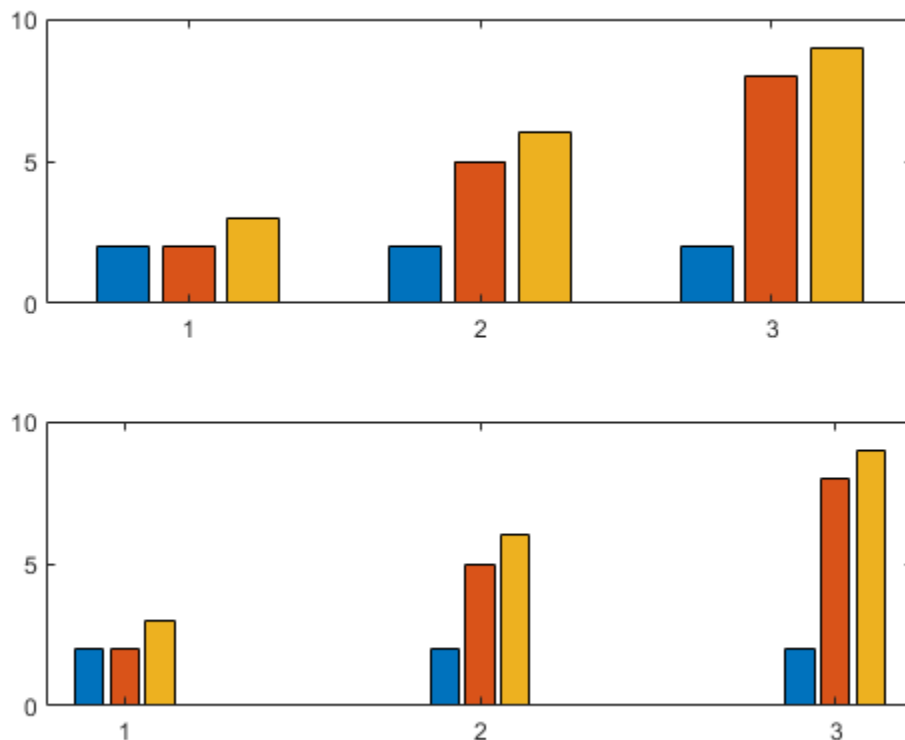
### ConstantLine and ConstantRegion Objects: Layer reference lines and filled regions in plots

You can move ConstantLine objects (created with `xline` and `yline`) and ConstantRegion objects (created with `xregion` and `yregion`) above or below other elements in a plot by setting the Layer property to "top" or "bottom".



## Bar Charts: Control bar group width

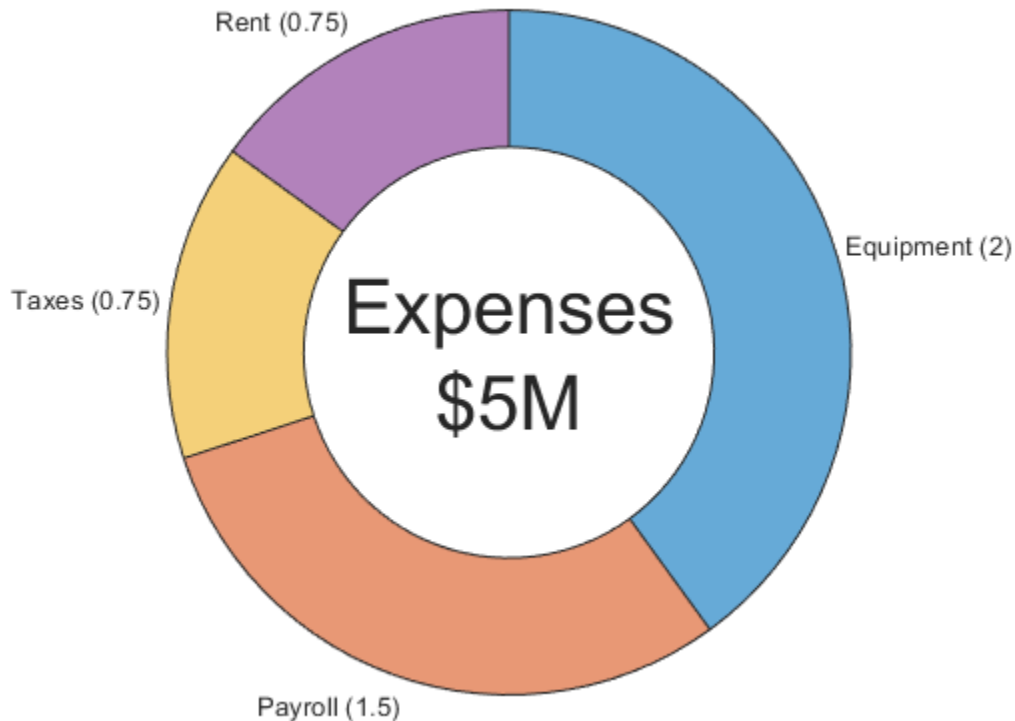
Control the bar group width by setting the `GroupWidth` property of at least one `Bar` object in the axes.



## Donut and Pie Charts: Label charts in different ways

You can use additional properties to label donut charts and pie charts:

- Create a large center label in a donut chart by setting the `CenterLabel` property of the chart. You can control the font size of the label by setting the `CenterLabelFontSize` property.
- Select different labeling schemes for a donut or pie chart by setting the `LabelStyle` property of the chart. You can choose from six different styles, including "name", which displays slice names, and "namedata", which displays the slice names and data values.



### **DatetimeRuler Object: Specify date format for secondary labels when plotting datetime data**

When you plot datetime data, you can specify the format of the secondary label of the datetime ruler by setting the `SecondaryLabelFormat` property of the `DatetimeRuler` object. For example, a value of "yyyy" displays a four-digit year.

### **clf, cla, and clim Functions: Pass array of objects as input**

Clear multiple figures or axes, or set the color limits of multiple axes at once by passing an array of objects to the `clf`, `cla`, and `clim` functions.

### **hold, axis, box, and grid Functions: Pass OnOffSwitchState values as input**

Before R2024a, you could set the state of certain axes properties by passing "on" and "off" values to the `hold`, `axis`, `box`, and `grid` functions. Now, those functions accept any `OnOffSwitchState` values, including 0, 1, false, and true. A value of true or 1 is equivalent to "on", and a value of false or 0 is equivalent to "off".

## scatterhistogram Function: Display smooth marginal histograms using kernel density estimates

Using the `scatterhistogram` function, you can create a scatter plot with marginal kernel density histograms by specifying the `HistogramDisplayStyle` name-value argument as `"smooth"`. Previously, kernel density estimation required a Statistics and Machine Learning Toolbox™ license.

## Graphics in MATLAB Online: Export graphics with specific dimensions and padding

Specify the width, height, padding, and whether to preserve the aspect ratio of the original graphic when you call the `exportgraphics` function.

Use these name-value arguments to control these aspects of the content:

- `Width` and `Height` — Specify the width and height of the output.
- `Padding` — Specify the amount of padding around the perimeter of the graphic.
- `Units` — Specify the units for the width, height, and padding values.
- `PreserveAspectRatio` — Specify whether to automatically add padding to preserve the original aspect ratio if the `Width` and `Height` values conflict with the original aspect ratio of the graphic.

## ⚠️Functionality being removed or changed

### ezpolar is not recommended

*Still runs*

The `ezpolar` function is not recommended. However, there are no plans to remove the function. To plot a mathematical expression in polar coordinates, use the `fpolarplot` function instead.

The benefits of using the `fpolarplot` function include:

- A more accurate visual representation of your function. Unlike `ezpolar`, `fpolarplot` adds more function evaluation points as needed.
- An argument for customizing the line style, marker symbol, and color.
- A `PolarAxes` object, which you can customize by setting `PolarAxes` properties. For example, you can change the range of angle values or choose whether to display angles in radians or degrees.

This table shows some typical usages of `ezpolar` and how to update your code to use `fpolarplot` instead. To display the plotted expression as `ezpolar` does, use the `title` function.

| Not Recommended                   | Recommended                          |
|-----------------------------------|--------------------------------------|
| <code>ezpolar(@(t) cos(t))</code> | <code>fpolarplot(@(t) cos(t))</code> |
| or                                |                                      |
| <code>ezpolar('cos(t)')</code>    |                                      |



| Not Recommended                                                   | Recommended                      |
|-------------------------------------------------------------------|----------------------------------|
| ezpolar(@(t) cos(t),[0 pi/4])<br>or<br>ezpolar('cos(t)',[0 pi/4]) | fpolarplot(@(t) cos(t),[0 pi/4]) |

**datetick is not recommended***Still runs*

The `datetick` function is not recommended. However, there are no plans to remove the function.

Instead of using `datetick`, plot datetime or duration values with any plotting function. Then you can configure the tick labels using the `xtickformat` and `ytickformat` functions or customize other aspects of your plot by setting properties of the `DatetimeRuler` object. For example, you can display datetime values in a variety of formats, include a time zone, or display a time zone offset.

**hgexport will be removed***Still runs*

The `hgexport` function will be removed in a future release. To save the contents of a plot as an image or vector graphics file, use the `exportgraphics` function. This function has a variety of options for controlling the output. For example, you can capture plots with tight cropping around the axes, control image resolution, embed fonts in PDFs, or create multipage PDFs and animated GIFs.

To save the contents of an app as an image file, use the `exportapp` function.

**colordef, graymon, and whitebg will be removed***Warns*

The `colordef`, `graymon`, and `whitebg` functions issue a warning that they will be removed in a future release.

**axis function will no longer support outputs of limit selection mode, visibility, and y-axis direction***Warns*

This syntax of the `axis` function, which returns the axis limit selection mode (`m`), visibility (`v`), and y-axis direction (`d`), issues a warning that it will be removed in a future release.

```
[m,v,d] = axis('state')
```

You can get the same information by querying these `Axes` properties:

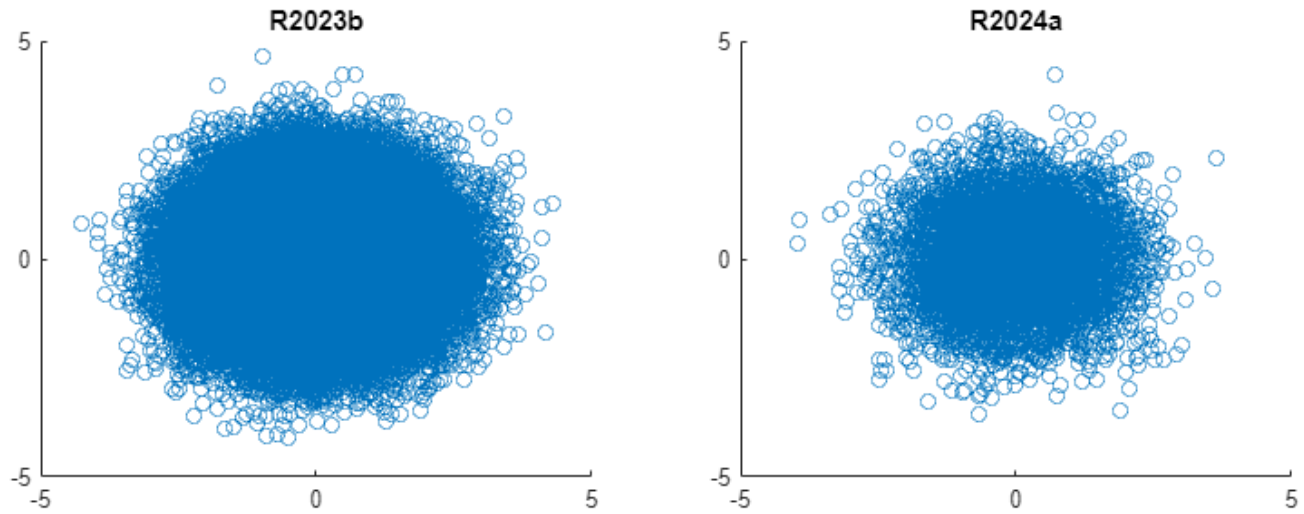
- `XLimMode`, `YLimMode`, and `ZLimMode` properties — Axis limit selection mode
- `Visible` property — Visibility
- `XDir` and `YDir` properties — x- and y-axis direction

**In apps, plots with high marker density look different without hardware-accelerated graphics***Behavior change*

When you create a plot that has a high density of markers in an app or in MATLAB Online using a system that does not have hardware-accelerated graphics, the plot displays fewer markers compared

to previous releases. This change helps to maintain responsiveness during plot interactions. This change affects only a subset of systems that do not meet the recommended requirements for running MATLAB.

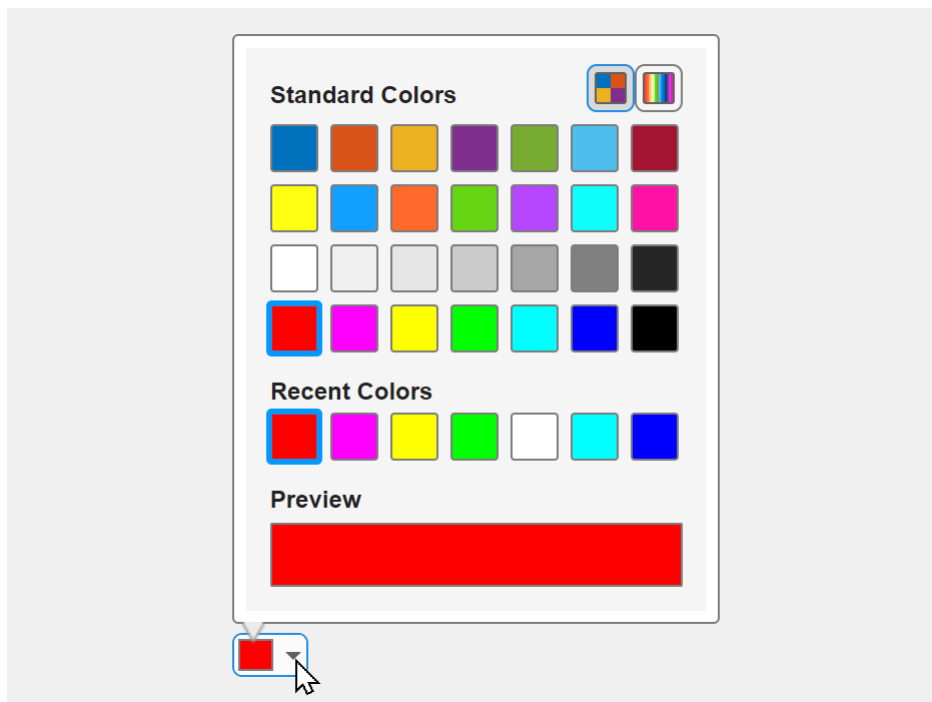
Here is a plot of 100,000 points on a system without hardware-accelerated graphics in R2023b and R2024a.



## App Building

### uicolorpicker Function: Add and configure color picker components in apps

To create a color picker in an app, use the `uicolorpicker` function or, in App Designer, drag a color picker UI component from the **Component Library** onto the canvas.



### uislider Function: Specify step size for range slider

When you create a range slider component using the `uislider` function, you can specify the amount that the value changes when a user interacts with the slider thumbs by setting the `Step` property.

For more information, see [RangeSlider Properties](#).

### uiradiobutton Function: Specify HTML markup and display equations in radio button text

Enable text markup for radio buttons created using the `uiradiobutton` function by setting the `Interpreter` property. Specify the interpreter as `"html"`, `"latex"`, `"tex"`, or `"none"`. Specifying an interpreter is supported only in App Designer apps and in figures created with the `uifigure` function.

For more information, see [RadioButton Properties](#).

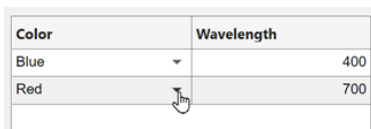
## uitable Function: Specify table colors using hexadecimal color codes or color names

When modifying colors in a table UI component using the `BackgroundColor` property, you can specify colors as hexadecimal color codes or color names, in addition to RGB triplets.

For more information, see [Table Properties](#).

## uitable Function: Visually identify table cells that have drop-down list editor

When you create a table UI component with editable cells that display categorical data, the cells include a visual indicator that the data can be edited using a drop-down list.



| Color | Wavelength |
|-------|------------|
| Blue  | 400        |
| Red   | 700        |

For more information about how data appears in a table UI component, see [Format Tabular Data in Apps](#).

## UI Figure Key-Press Callbacks: Access data about pressed key for larger list of keys

When you use a key-press callback, such as a `KeyPressFcn` or `WindowKeyReleaseFcn` callback, for a figure created using the `uifigure` function, the `Character` property of the callback event data can return an expanded set of keys using the Unicode® symbol for the pressed key. This change provides key data for a larger set of keyboard keys across many writing systems.

For more information, see [UI Figure Properties](#).

## App Designer: Build apps that interface with Simulink models

Interactively build a custom app that interfaces with a Simulink model by using App Designer. Use the App Designer design environment to:

- Perform common simulation tasks from the app, such as starting and stopping the simulation and viewing simulation progress, using Simulink UI components.
- Tune model variables from the app while the simulation runs by connecting UI components to variables.
- Visualize simulation signals in the app by connecting signals to a time scope UI component.

For more information, see [Create App for Simulink Model \(Simulink\)](#).

## App Designer: Convert between additional similar UI components

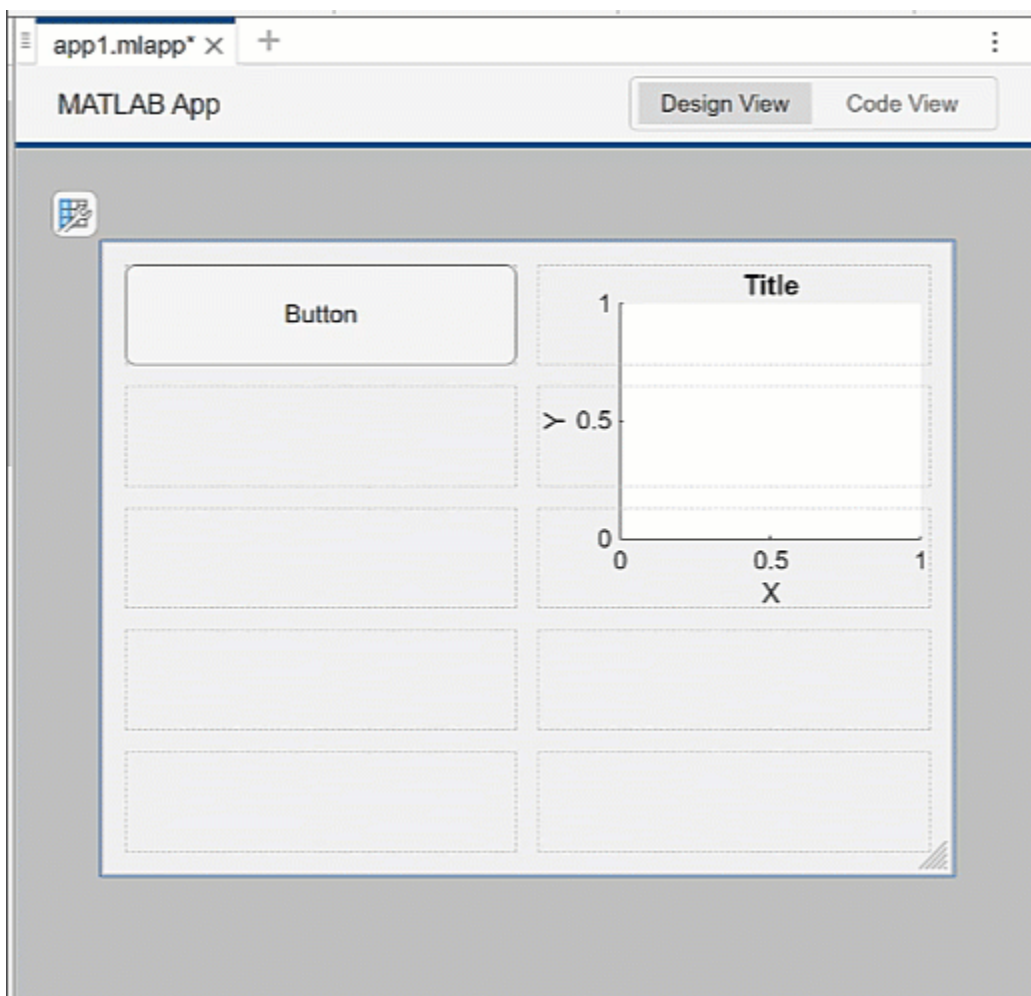
To convert one type of UI component to another with similar functionality, right-click the component on the canvas or in the **Component Browser** and select **Replace With**. Then, select the component to convert to. Replacing one component with another preserves relevant property values, such as font

properties and callbacks that exist for both components. For example, you can now convert component types within these additional groups:

- Push button and state button
- Tree and check box tree
- List box, drop-down component, and discrete knob

## App Designer: Configure grid layout using improved editor interactions

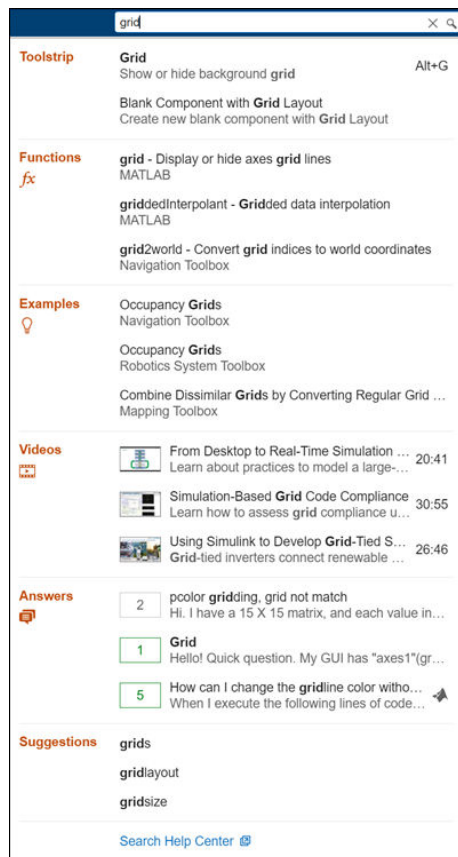
Use the grid layout manager in **Design View** with improved interactions to more efficiently add, remove, and manage the size of multiple grid rows and columns. For more information, see [Use Grid Layout Managers in App Designer](#).



## App Designer in MATLAB Online: Access toolbar actions and Help Center resources using search

In MATLAB Online, you can now use the Search box in the top-right corner of App Designer to easily access toolbar actions and Help Center resources such as functions, examples, and answers.

For example, to show the background grid on the App Designer canvas, enter the word `grid` in the Search box. In the list of results, select the **Grid** tooltip action.



## Axes Interactions: Customize behavior of interactions

For `polaraxes` and `geoaxes` objects created in App Designer or in figures created using the `uifigure` function, you can customize the behavior of axes interactions. Customize the behavior of data tips by using the `InteractionOptions` property of a `polaraxes` or `geoaxes` object. For `geoaxes` objects, you can also customize the panning and zooming behavior, as well as how to restore views, by using the same property. For more information, see `PolarAxesInteractionOptions` Properties and `GeographicAxesInteractionOptions` Properties.

## Axes Interactions: Interact with axes using the keyboard

You can pan, zoom, and rotate the view of axes using the keyboard. For example, when the pan interaction is enabled on axes with focus, you can pan the view of the axes using the arrow keys. For more information, see `panInteraction`, `zoomInteraction`, and `rotateInteraction`.

## App Testing Framework: Perform scroll gestures on axes and UI axes

You can use the `scroll` method to test scroll gestures on axes and UI axes. Specify the direction of the scroll gesture to mimic a user scrolling up, down, left, or right.

## App Testing Framework: Perform gestures on hyperlinks

You can use the `press` and `chooseContextMenu` methods to perform gestures on hyperlink components. For example, create a hyperlink and programmatically click it to open a URL.

```
fig = uifigure;
hlink = uihyperlink(fig);
hlink.Text = "MathWorks";
hlink.URL = "https://www.mathworks.com/";

testCase = matlab.uitest.TestCase.forInteractiveUse;
testCase.press(hlink)
```

## ⚠App Testing Framework: Node expanded callback executes when gestures programmatically expand tree nodes

To better mimic a user who must expand tree nodes to interact with a nested node, the node expanded callback executes when you perform a gesture on a collapsed tree node by using the `choose` or `chooseContextMenu` method. You can specify the callback by setting the `NodeExpandedFcn` property of the tree.

For example, perform a `choose` gesture on a nested node in a tree, and display the text of any programmatically expanded nodes.

```
fig = uifigure;
t = uitree(fig);
t.NodeExpandedFcn = @(src,event) disp(event.Node.Text);

parent = uitreenode(t,"Text","Runners");
child1 = uitreenode(parent,"Text","Joe");
child2 = uitreenode(parent,"Text","Linda");

testCase = matlab.uitest.TestCase.forInteractiveUse;
testCase.choose(child2)
```

## ⚠Compatibility Considerations

If you do not want the callback to execute, preserving the behavior in R2023b and earlier, expand the tree node before performing the gesture. In this code, the callback does not execute.

```
fig = uifigure;
t = uitree(fig);
t.NodeExpandedFcn = @(src,event) disp(event.Node.Text);

parent = uitreenode(t,"Text","Runners");
child1 = uitreenode(parent,"Text","Joe");
child2 = uitreenode(parent,"Text","Linda");

testCase = matlab.uitest.TestCase.forInteractiveUse;
expand(parent)
testCase.choose(child2)
```

## ▲ Functionality being removed or changed

### In apps, plots with high marker density look different without hardware-accelerated graphics

#### *Behavior change*

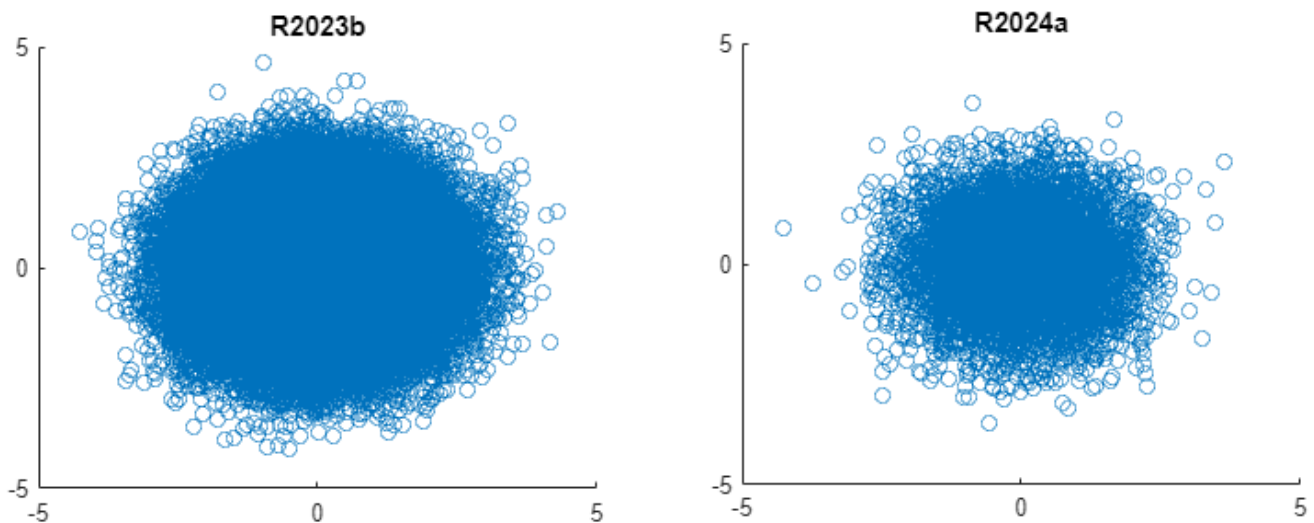
When you create a plot that has a high density of markers in an app or in MATLAB Online using a system that does not have hardware-accelerated graphics, the plot displays fewer markers compared to previous releases. This change helps to maintain responsiveness during plot interactions.

---

**Note** This change only affects a subset of systems that do not meet the recommended requirements for running MATLAB.

---

Here is a plot of 100,000 points on a system without hardware-accelerated graphics in R2023b and R2024a.





## Performance

### datetime Format Parsing: Improved performance when parsing common numeric formats

`datetime` format parsing performance is significantly faster when parsing some common formats. In general, performance is best for text timestamps in formats that use only numeric specifiers, such as the ISO 8601 format. This list shows a few examples of such formats, which are part of ongoing efforts to improve `datetime` format parsing performance:

- `uuuu-MM-dd` and `yyyy-MM-dd`
- `MM/dd/uuuu` and `MM/dd/yyyy`
- `uuuu-MM-dd 'T' HH:mm:ss.SSS 'Z'` and `yyyy-MM-dd 'T' HH:mm:ss.SSS 'Z'`

For example, this code creates and then parses a string array that contains a series of dates. The dates in the string array have the `uuuu-MM-dd` numeric format. The `datetime` function recognizes the format, parses the strings, and returns `datetime` values. The code is about 16x faster than in the previous release.

```
function timingTest
    d1 = datetime(2024,1,1:10000);
    s = string(d1,"uuuu-MM-dd");

    for i = 1:100
        d2 = datetime(s);
    end
end
```

The approximate execution times are:

**R2023b:** 3.13 s

**R2024a:** 0.19 s

The code was timed on a Windows 11, AMD EPYC 74F3 24-Core Processor @ 3.19 GHz test system using the `timeit` function.

```
timeit(@timingTest)
```

### imread and imwrite Functions: Improved performance with JPEG files

The `imread` and `imwrite` functions show improved performance when working with JPEG files. For example, this code reads a large JPEG image. The code is about 1.5x faster than in the previous release.

```
function t = imreadTimingTest
    filename = "world.topo.bathy.200407.3x21600x21600.C1.jpg";
    url = "https://eoimages.gsfc.nasa.gov/images/imagerecords/73000/73751/" + filename;
    websave(filename,url);

    f = @() imread(filename);

    t = timeit(f);
end
```

The approximate execution times are:

**R2023b:** 6.6 s

**R2024a:** 4.3 s

This code writes a large JPEG image. The code is about 1.6x faster than in the previous release.

```
function t = imwriteTimingTest
    filename = "world.topo.bathy.200407.3x21600x21600.C1.jpg";
    url = "https://eoimages.gsfc.nasa.gov/images/imagerecords/73000/73751/" + filename;
    A = imread(url);

    f = @() imwrite(A,filename);

    t = timeit(f);
end
```

The approximate execution times are:

**R2023b:** 38.3 s

**R2024a:** 23.8 s

The code was timed on a Windows 11, Intel Xeon Gold 6246R CPU @ 3.40 GHz test system by calling the `imreadTimingTest` and `imwriteTimingTest` functions.

## smoothdata2 and fillmissing2 Functions: Improved performance for moving average methods

The `smoothdata2` and `fillmissing2` functions show improved performance when you specify the "movmean", "movmedian", or (for `smoothdata2`) "gaussian" moving method. The performance improvement is more significant when the input data has at least several hundred elements.

For example, this code smooths a 100-by-100 matrix of noisy data using the median over a 3-by-3 moving window. The code is about 3.6x faster than in the previous release.

```
function timingSmoothdata2
A = rand(100,100);
for k = 1:700
    smoothdata2(A,"movmedian",3);
end
end
```

The approximate execution times are:

**R2023b:** 1.05 s

**R2024a:** 0.29 s

The code was timed on a Windows 11, AMD EPYC 74F3 24-Core Processor @ 3.19 GHz test system using the `timeit` function.

```
timeit(@timingSmoothdata2)
```

## mldivide Function: Improved performance with tridiagonal matrices

The `mldivide` function shows improved performance when solving linear systems  $A \cdot x = b$  with a full tridiagonal coefficient matrix  $A$ . For example, this code solves a linear system specified for tridiagonal matrix  $D$ . The code is about 6.5x faster than in the previous release.

```
function t = timingTest
n = 5e3;
A = randn(n);
[L,D,P] = ld1(A,"upper");
b = randn(n,1);

f = @() D\b;
t = timeit(f);
end
```

The approximate execution times are:

**R2023b:** 0.13 s

**R2024a:** 0.02 s

The code was timed on a Windows 11, AMD EPYC 74F3 24-Core Processor @ 3.19 GHz test system by calling the `timingTest` function.

## audioplayer Function : Improved performance with longer duration audio signals

The `audioplayer` function shows improved performance when operating on audio signals longer than 100 seconds. For example, this code creates an audio player for a 300-second audio signal using a sample rate of 48 KHz. The code is about 6.1x faster than in the previous release.

```
function timingTest
Fs = 48000;
audioSignal = rand(300*Fs,1);
audioplayer(audioSignal,Fs);
end
```

The approximate execution times are:

**R2023b:** 61.9 s

**R2024a:** 10.1 s

The code was timed by calling the `audioplayer` function. Because MATLAB Online is hosted, performance is independent of the computer hardware used to access it.

## Files Panel in MATLAB Online: Improved performance when previewing large MAT-files

The Files panel shows improved performance when previewing a large MAT-file in MATLAB Online.

The delay between clicking the  button and the preview being ready is reduced.

For example, create a large MAT-file in MATLAB Online. Then, pause on the MAT-file name in the Files panel and click the  button that appears. You can view the MAT-file preview 5.6x sooner than in the previous release.

```
x = randn(50000000,1);  
save sample.mat -v7.3
```

The approximate rendering times are:

**R2023b:** 14 s

**R2024a:** 2.5 s

Because MATLAB Online is hosted, performance is independent of the computer hardware used to access it.

## Workspace Panel in MATLAB Online: Improved performance when loading large number of variables

The Workspace panel shows improved performance when loading a large number of variables in MATLAB Online. The delay between loading a MAT-file that contains a large number of variables, such as one created by a Simulink model, and the Workspace panel showing the variables is reduced.

For example, create a MAT-file that contains 2000 scalar variables using a local function.

```
function createVariables  
for i = 1:2000  
    varName = sprintf('var%d',i);  
    eval([varName '=i;']);  
end  
save("largeFile.mat");  
end
```

After clearing any existing variables from the workspace, load variables from the file into the workspace. In R2023b, the Workspace panel takes about 3.5 seconds to load the variables. In R2024a, you can view the variables in the Workspace panel almost instantaneously.

```
createVariables  
load("largeFile.mat")
```

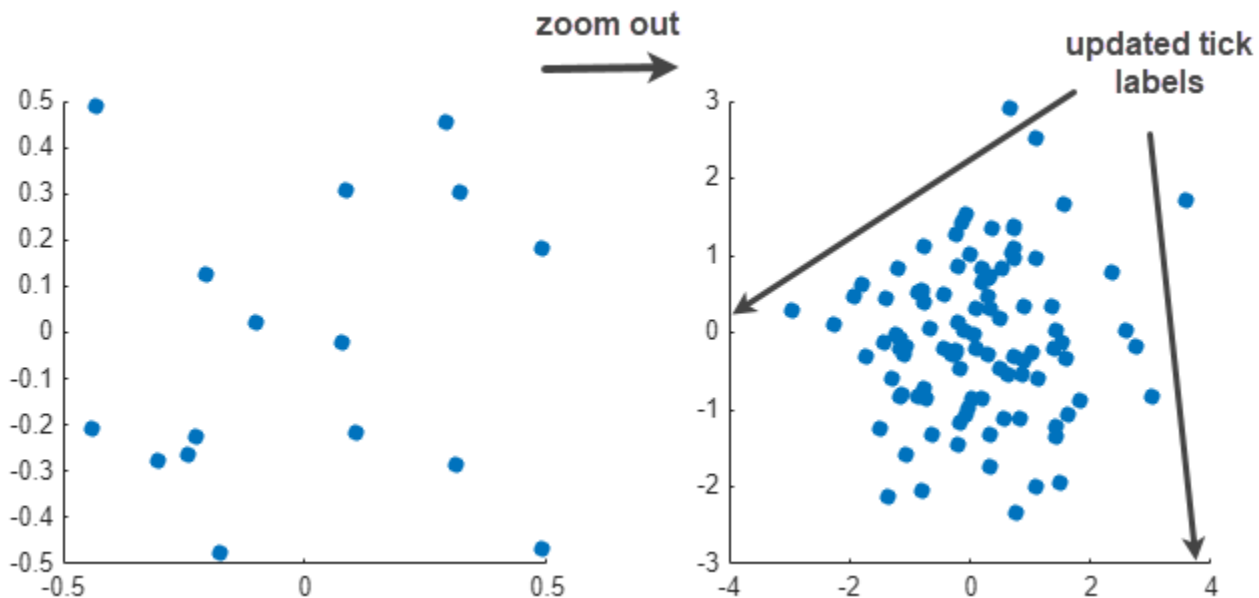
Because MATLAB Online is hosted, performance is independent of the computer hardware used to access it.

## Plots in Apps and MATLAB Online: Ticks and tick labels update faster when zooming

When you scroll to zoom within a plot, the ticks and tick labels update faster. To observe the improvement, the plot must be created in an app, in a figure created with the `uifigure` function, or in MATLAB Online.

For example, create a scatter plot, and adjust the x- and y-axes limits to exclude some of the plot. Then scroll to zoom in and out. The ticks and tick labels update more quickly on a Windows 11, Intel Xeon CPU W-2133 @ 3.60 GHz test system when you scroll in R2024a than in the previous release.

```
f = uifigure;
ax = axes(f);
rng(0)
x = randn(1,100);
y = randn(1,100);
scatter(ax,x,y,"filled")
xlim(ax,[-0.5 0.5])
ylim(ax,[-0.5 0.5])
```

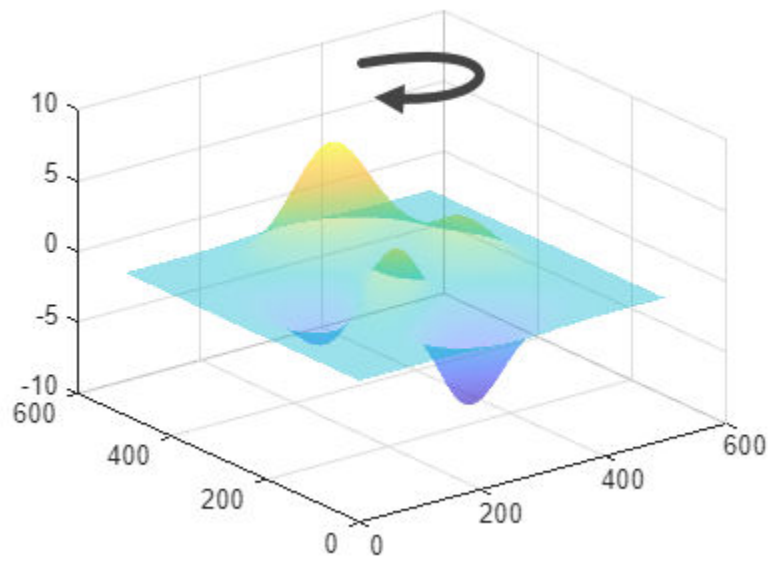


## Plots in Apps and MATLAB Online: Improved responsiveness when interacting with partially transparent 3-D plots

Interactions with partially transparent 3-D plots show improved responsiveness. To observe the improvement, the plot must be created in an app, in a figure created with the `uifigure` function, or in MATLAB Online.

For example, create a partially transparent surface plot of the `peaks` function evaluated over a 500-by-500 grid in an app window. Then drag to rotate the axes. On a Windows 11, Intel Xeon CPU W-2133 @ 3.60 GHz test system, the rotation is smoother and responds more quickly to the drag gesture in R2024a than in the previous release.

```
f = uifigure;
ax = axes(f);
Z = peaks(500);
surf(ax,Z,EdgeColor="none",FaceAlpha=0.4)
```



## Software Development Tools

### Project API: Find project files by label

You can now programmatically perform an advanced search in a project using the `findFiles` function. Supported workflows include:

- Listing all files in a project
- Filtering files that are not part of the project
- Finding files by label or category name
- Creating test suites from test files in a project and its references

### Project API: Reanalyze all project dependencies

You can now programmatically reanalyze all files in your project and perform a complete dependency analysis by using the `updateDependencies` function.

### Project API: Export list of project files to archive

You can now programmatically export a list of project files to an archive by using the `export` function.

For projects that have missing files, you can enable the `export` function to ignore the missing files.

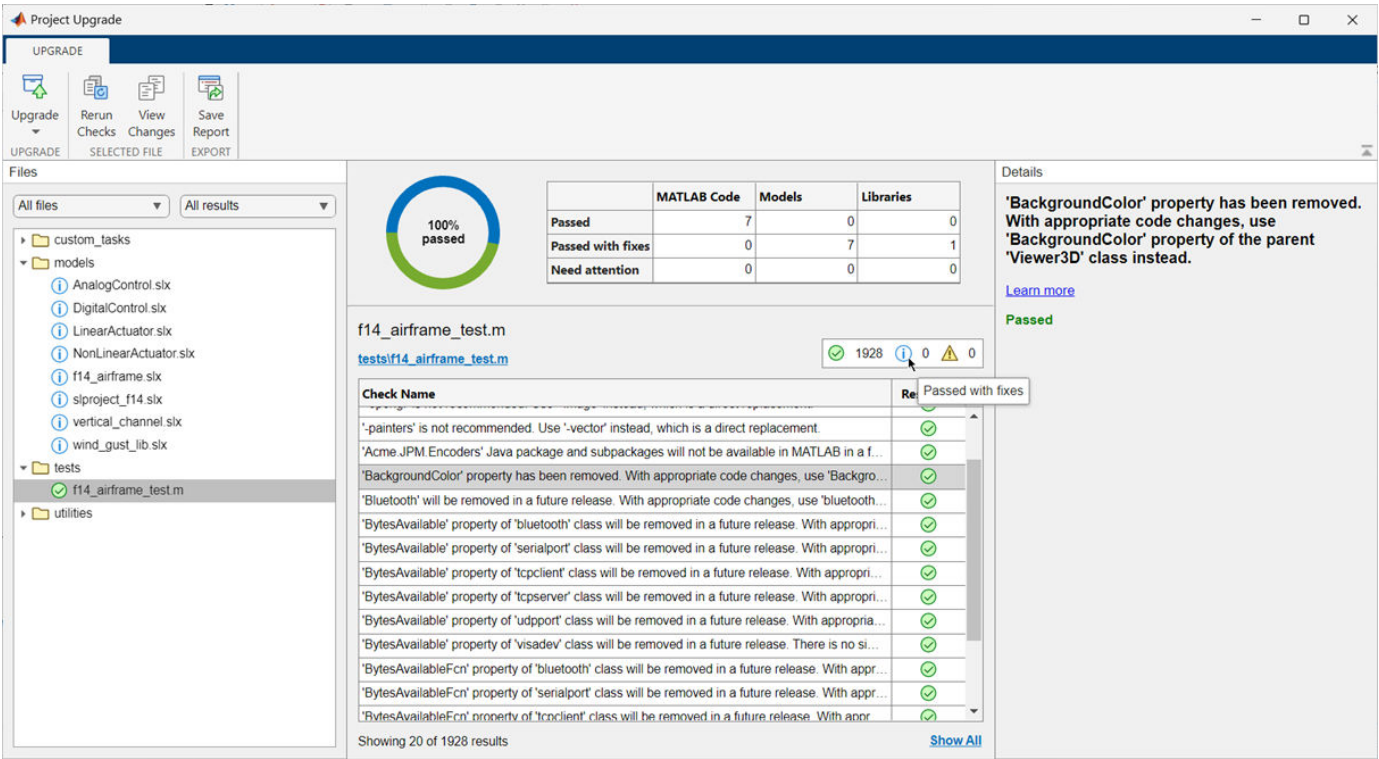
### Project Upgrade: Check for compatibility issues and upgrade project with improved usability and appearance

For projects that contain only MATLAB files, use Project Upgrade to check for compatibility issues with the current release. For projects that also contain Simulink models and libraries, you can apply fixes and automatically upgrade your project to the current release.

You can now easily interpret the upgrade results and examine the checks marked as need attention. You can also access frequent actions from the Project Upgrade toolstrip:

- Rerun checks.
- View changes that the upgrade applied.
- Save results in a report.

For more information, see [Check for Compatibility Issues Using Project Upgrade](#).



Source Control: Support for signing Git commits

You can now enable the MATLAB Git integration to sign Git commits automatically. For more information, see Enable Signing Commits.

Source Control: Support for Git hooks

MATLAB Git integration can now run Git hooks with no additional setup. Starting in R2024a, you do not need to install Cygwin on Windows.

Supported hooks are pre-commit, commit-msg, post-commit, prepare-commit-msg, pre-push, pre-merge-commit, post-checkout, and post-merge. For more information, see Set Up Git Source Control.

Source Control API: Discard changes in Git repositories programmatically

You can now programmatically restore modified files in a Git repository using the discardChanges function.

Source Control in MATLAB Online: Expanded support for Git workflows

MATLAB Online now provides expanded support for Git workflows:

- Squashing Git commits



- Rebasing Git branches

## Build Automation: Perform additional test and coverage analyses using TestTask class

You can produce additional test and coverage artifacts when using the `matlab.buildtool.tasks.TestTask` class:

- Use the `addCodeCoverage` method to enable a task to collect code coverage information for your MATLAB source code. If you have a MATLAB Test™ license, you can also use the method to produce decision coverage, condition coverage, and modified condition/decision coverage (MC/DC) metrics.
- If you have a Simulink Test license, you can export Simulink Test Manager results in MLDATX format by setting the `TestResults` property during task creation. If you also have a Simulink Coverage™ license, you can produce model coverage results by setting the `ModelCoverageResults` property during task creation or by using the `addModelCoverage` method.

## Build Automation: Obfuscate source code using PcodeTask class

You can use the `matlab.buildtool.tasks.PcodeTask` class to create P-code files by obfuscating MATLAB source code as part of your build.

## Build Automation: Build MEX file using MexTask class

You can use the `matlab.buildtool.tasks.MexTask` class to compile and link source files into a binary MEX file as part of your build.

## Build Automation: Programmatically access code analysis results using CodeIssuesTask class

You can use the `Results` property of the `matlab.buildtool.tasks.CodeIssuesTask` class to programmatically access the results of code analysis. For example, you can return the paths to the code analysis results by using `p = plan("check").Results.paths'`. You can also specify the code analysis results as inputs of other tasks, for instance, `plan("archive").Inputs = plan("check").Results`.

## Build Automation: Run tests in parallel

If you have a Parallel Computing Toolbox™ license, you can run tests in parallel by specifying the `-parallel` option of the `buildtool` command. This option affects only how `matlab.buildtool.tasks.TestTask` instances run. If you specify the `-parallel` option, the build runs the tests associated with `TestTask` instances in parallel.

## Build Automation: Create initial build file

You can use the `buildtool` command to create a build file. To create a build file named `buildfile.m` in your current folder, execute `buildtool -init`. The resulting build file contains

tasks created from built-in task classes in the `matlab.buildtool.tasks` namespace. You can use this build file as a starting point to define your build.

**Build Automation: Disable incremental builds for task**

You can disable incremental builds for a task. To force your task to rerun even if it is up to date, set the `DisableIncremental` property of the `matlab.buildtool.Task` instance to `true`.

**Unit Testing Framework: Use renamed methods and name-value arguments when working with namespaces**

MATLAB now refers to packages as namespaces. Some MATLAB unit testing methods and name-value arguments have been renamed to reflect this change. Their behavior remains the same.

This table shows the affected methods and their new names.

| R2023b and Earlier                                                 | Starting in R2024a                                                   |
|--------------------------------------------------------------------|----------------------------------------------------------------------|
| <code>matlab.unittest.TestSuite.fromPackage</code>                 | <code>matlab.unittest.TestSuite.fromNamespace</code>                 |
| <code>matlab.unittest.plugins.CodeCoveragePlugin.forPackage</code> | <code>matlab.unittest.plugins.CodeCoveragePlugin.forNamespace</code> |

This table shows the affected name-value arguments and their new names. Use the `IncludingInnerNamespaces` name-value argument with the `fromNamespace` and `forNamespace` static methods, and use the `IncludeInnerNamespaces` name-value argument with the `runtests`, `runperf`, and `testsuite` functions.

| R2023b and Earlier                | Starting in R2024a                    |
|-----------------------------------|---------------------------------------|
| <code>IncludingSubpackages</code> | <code>IncludingInnerNamespaces</code> |
| <code>IncludeSubpackages</code>   | <code>IncludeInnerNamespaces</code>   |

There are no plans to remove support for existing references to the methods and name-value arguments.

**Unit Testing Framework: Run tests using Test Browser in MATLAB Online**

You can run script-based, function-based, and class-based tests using the Test Browser app in MATLAB Online.

**Unit Testing Framework: Generate single-file HTML test report**

You can generate a single-file HTML test report by using the `matlab.unittest.plugins.TestReportPlugin.producingHTML` or `generateHTMLReport` method. In a single-file HTML test report, all the contents of the report, such as text and images, are included in a single HTML file. In previous releases, these methods support only multifile test reports.

To generate a single-file HTML test report, specify the name of the test report file as an input to the method. For example, run your tests and generate an HTML test report contained in a single file named "myTestReport.html".

```
results = runtests("myTestFile.m");
generateHTMLReport(results, "myTestReport.html")
```

## Unit Testing Framework: Control highlighting for covered or missed executables in interactive code coverage reports

You can control the highlighting of executables in an interactive code coverage report generated using either the `matlab.unittest.plugins.codecoverage.CoverageReport` class or the `generateHTMLReport` method. By default, the report highlights all the executables using different colors. To turn off highlighting for a group of executables, clear the corresponding check box in the **Overall Coverage Summary** section of the report. You can control the highlighting for covered, missed, or partially covered executables.

## Unit Testing Framework: Modify HTML document title of code coverage report

You can modify the HTML document title when you generate a code coverage report using the `matlab.unittest.plugins.codecoverage.CoverageReport` class or the `generateHTMLReport` method. To modify the document title, specify the `DocumentTitle` name-value argument. For example, run your tests and generate a code coverage report with a specific HTML document title from the coverage results.

```
[testResults,coverageResults] = runtests("myTestFile.m", ...
    ReportCoverageFor=pwd)
generateHTMLReport(coverageResults,DocumentTitle="My Coverage Report")
```

## Unit Testing Framework: Enable parallelizable plugins to work with default data if communication buffer is empty

The `retrieveFrom` method of the `matlab.unittest.plugins.Parallelizable` interface accepts a new input that lets you use default data if MATLAB workers cannot store the data that the plugin collects. For an example, see [Extend Reporting of Finalized Test Suite Portion](#).

## Unit Testing Framework: Use shared test fixtures in interactive tests

When you create an interactive test case from a test class using the `matlab.unittest.TestCase.forInteractiveUse` method, you can use the shared test fixtures associated with that test class for testing. For example, `testCase = matlab.unittest.TestCase.forInteractiveUse(MyTestClass,ApplySharedTestFixtures=true)` sets up the shared test fixtures associated with `MyTestClass` when creating an interactive test case. The testing framework automatically tears down the shared test fixtures when the test case goes out of scope.

## Unit Testing Framework: Respect element count when comparing sets

When you compare sets using the `matlab.unittest.constraints.IsSameSetAs` class, you can test whether elements occur the same number of times in the actual and expected sets. To respect the element count, set the `RespectCount` property of the constraint to `true`.

## App Testing Framework: Perform scroll gestures on axes and UI axes

You can use the `scroll` method to test scroll gestures on axes and UI axes. Specify the direction of the scroll gesture to mimic a user scrolling up, down, left, or right.

## App Testing Framework: Perform gestures on hyperlinks

You can use the `press` and `chooseContextMenu` methods to perform gestures on hyperlink components. For example, create a hyperlink and programmatically click it to open a URL.

```
fig = uifigure;
hlink = uihyperlink(fig);
hlink.Text = "MathWorks";
hlink.URL = "https://www.mathworks.com/";

testCase = matlab.unittest.TestCase.forInteractiveUse;
testCase.press(hlink)
```

## ⚠App Testing Framework: Node expanded callback executes when gestures programmatically expand tree nodes

To better mimic a user who must expand tree nodes to interact with a nested node, the node expanded callback executes when you perform a gesture on a collapsed tree node by using the `choose` or `chooseContextMenu` method. You can specify the callback by setting the `NodeExpandedFcn` property of the tree.

For example, perform a `choose` gesture on a nested node in a tree, and display the text of any programmatically expanded nodes.

```
fig = uifigure;
t = uitree(fig);
t.NodeExpandedFcn = @(src,event) disp(event.Node.Text);

parent = uitreenode(t,"Text","Runners");
child1 = uitreenode(parent,"Text","Joe");
child2 = uitreenode(parent,"Text","Linda");

testCase = matlab.unittest.TestCase.forInteractiveUse;
testCase.choose(child2)
```

## ⚠Compatibility Considerations

If you do not want the callback to execute, preserving the behavior in R2023b and earlier, expand the tree node before performing the gesture. In this code, the callback does not execute.

```
fig = uifigure;
t = uitree(fig);
```

```

t.NodeExpandedFcn = @(src,event) disp(event.Node.Text);

parent = uitreenode(t,"Text","Runners");
child1 = uitreenode(parent,"Text","Joe");
child2 = uitreenode(parent,"Text","Linda");

testCase = matlab.uitest.TestCase.forInteractiveUse;
expand(parent)
testCase.choose(child2)

```

## ▲ **Functionality being removed or changed**

### **Default build task name is empty string**

#### *Behavior change*

By default, the `Name` property of the `matlab.buildtool.Task` class contains an empty string. In previous releases, the default value is a missing value.

### **Test Browser handles valid test files differently in the presence of invalid test files**

#### *Behavior change*

The behavior of Test Browser has changed when a group of files includes both valid and invalid test files:

- If you add tests to the test browser by selecting multiple test files, then the test browser imports the tests in the valid test files. In previous releases, the test browser does not import any tests if the group of selected files includes invalid test files.
- If you run a group of both valid and invalid test files, then the test browser runs the tests in the valid test files. In previous releases, the test browser does not run any tests if the group includes invalid test files.

An invalid test file is a test file that the testing framework cannot run. Examples include a test file that contains syntax errors, a function-based test file that is missing local functions, and a file with a `Test` method that is passed an undefined parameterization property.

## External Language Interfaces

### ★REST Function Service: Call user-defined MATLAB functions from any local or remote client program using REST

You can write client programs to call MATLAB using the MATLAB REST Function Service. The service uses HTTPS protocol, which enables communication between your programs and MATLAB regardless of your platform and execution environment. You can call custom MATLAB functions using any programming language or application that can make a REST call, either locally or remotely. For more information, see [Call MATLAB Functions Using REST Function Service](#).

### ★Python Interface: Convert between MATLAB tables and Python Pandas DataFrames

You can now pass MATLAB tables or timetables directly to Python functions in MATLAB. MATLAB automatically converts the MATLAB table or timetable to a Python Pandas DataFrame.

Also, you can convert a Python Pandas DataFrame to a MATLAB table or timetable using the `table` or `timetable` function. For example, convert the Python Pandas DataFrame `df` to a MATLAB table and timetable.

```
T = table(df);
TT = timetable(df);
```

For more information, see [Use Python Pandas DataFrames in MATLAB](#).

### ★Python Interface: Interactively run Python code with Run Python Live Editor Task

Use the Run Python Code task in the Live Editor to interactively run Python code or files and generate the corresponding MATLAB code. You can also use the Live Editor task to return Python variables in MATLAB, as well as pass MATLAB data to the Python code.

To add the task to a live script in the Live Editor, click **Task** on the **Live Editor** tab and select the **Run Python Code** icon.

For more information, see [Run Python Code](#).

### Python Interface: Convert between MATLAB and Python dictionaries

You can now pass MATLAB dictionaries directly to Python functions in MATLAB. MATLAB automatically converts the MATLAB dictionary to a Python dict. For example, pass a MATLAB dictionary `d` to the Python function `print`.

```
d = dictionary([10 20 30],[1 2 3]);
py.print(d)
```

Also, you can convert a Python dict to a MATLAB dictionary using the `dictionary` function.

```
pyDict = py.dict({{10,1},{20,2},{30,3}});
d = dictionary(pyDict);
```

For more information, see [Use Python Dictionaries in MATLAB](#).

## Publish C++ Interface: Share library definition file with publisher

You can share a MATLAB interface to a C++ library, for example by uploading the library definition files to GitHub. If other publishers have the same C++ library version and file structure as you do, then they can download the definition files and build the interface.

The files used to generate the interface, such as those specified by the `InterfaceGenerationFiles` argument of the `clibgen.generateLibraryDefinition` function, contain file specifications based on the folder structure of your computer. Starting in R2024a, you can specify file paths relative to an absolute path by specifying the `RootPaths` name-value argument as the beginning of the absolute path to the library files. Then another publisher can build the interface by updating the absolute path specified using the `RootPaths` dictionary based on the folder structure on their computer.

For an example, see [Define Pathnames to Share Library Definition Files](#).

## Publish C++ Interface: Use `InterfaceName` name-value argument, renamed from `PackageName`, to identify MATLAB interface to C++ library

MATLAB now refers to packages as namespaces. Some name-value arguments and properties used to publish a MATLAB interface to a C++ library have been renamed to reflect this change. Their behavior remains the same.

- In the `clibgen.generateLibraryDefinition` function, the `PackageName` name-value argument is now named `InterfaceName`.
- In the `clibgen.buildInterface` function, the `PackageName` name-value argument is now named `InterfaceName`.
- In the `clibgen.LibraryDefinition` class, the `PackageName` property is now named `InterfaceName`.

Use `InterfaceName` to identify the MATLAB interface to a specific C++ library. For example, if you set `InterfaceName` to `"libname"`, then function calls to the library have this format:

```
retVal = clib.libname.funcname(arg1,arg2,...)
```

There are no plans to remove support for existing references to the arguments and property.

## Call C++ from MATLAB: Use string for C++ enum parameter

You can pass a MATLAB string to a C++ function or method in C++ library *libname* that accepts `clib.libname.enum` as an input argument. For more information, see the string scalar entry in the [MATLAB Type to C++ Type Mapping table](#).

## Call MATLAB from C++: Support for data types in `matlab::engine::MATLABEngine` functions `feval` and `fevalAsync`

The `matlab::engine::MATLABEngine` member functions `feval` and `fevalAsync` support native C++ `string`, `std::vector`, and `std::complex` data types. For more information, see the `RhsArgs&&... rhsArgs` entry in the `feval` and `fevalAsync` Parameters tables.

## Call MATLAB from C++: Support for handle classes in strongly typed C++ interface

You can include handle classes in a strongly typed C++ interface when you call the `matlab.engine.typedinterface.generateCPP` function. The function creates code with handle copy behavior. That is, functions and methods can modify the input data in the calling workspace without making copies. For more information, see [Use MATLAB Handle Classes in C++](#).

The function does not generate code to support dynamic properties, the `set` and `get` behavior for graphics handles, listener workflows, or the `findprop`, `findobj`, and `delete` handle methods.

## Call MATLAB from C++: Support for comparison operators

The `matlab.engine.typedinterface.generateCPP` function generates C++ comparison methods and the `isvalid` function as well as comparison operators for both handle and value classes, which implement the `eq`, `ne`, `lt`, `gt`, `le`, and `ge` methods in MATLAB.

Code generated for comparison operators like `==` have a Boolean output type. If you override the `eq` function in MATLAB with a non-Boolean output, then you must use the `eq` method in your application instead of `==`.

## Call MATLAB from C++: Use Namespaces name-value argument, renamed from Packages, with `matlab.engine.typedinterface.generateCPP` function

In the `matlab.engine.typedinterface.generateCPP` function, the `Packages` name-value argument is now named `Namespaces`. The behavior of this name-value argument remains the same. There are no plans to remove support for existing references to the argument.

## Java: MATLAB support for OpenJDK™ 17

As of R2024a, MATLAB supports OpenJDK 17 from <https://adoptium.net/>. For supported version information, see [MATLAB Interfaces to Other Languages](#).

## .NET Interface: Support for .NET 6

On Microsoft Windows platforms, MATLAB supports .NET 6 and higher, in addition to existing support for the Microsoft .NET Framework. To select a .NET environment, use the `dotnetenv` function. For more information, see [Selecting .NET Core in MATLAB](#).



## ▲ Compatibility Considerations

For more information about changes to .NET support, see “.NET Interface: .NET 5 is no longer supported” on page 4-55.

### Call MATLAB from .NET: Support for dictionary objects

You can convert a .NET object whose type implements the `System.Collections.Generic.IDictionary<TKey,TValue>` interface to a MATLAB dictionary.

You can convert a MATLAB dictionary to any of these .NET types:

- `System.Collections.Generic.Dictionary<TKey,TValue>`
- `System.Collections.Generic.IDictionary<TKey,TValue>`
- `System.Collections.Generic.IReadOnlyDictionary<TKey,TValue>`
- `System.Collections.IDictionary`

For more information, see:

- Pass .NET Dictionary to MATLAB.
- Handle MATLAB Dictionary Type in .NET.

For examples, see Use MATLAB Dictionary Objects in .NET.

### Call MATLAB from .NET: Support for handle classes in strongly typed C# interface

You can include handle classes in a strongly typed C# interface when you call the `matlab.engine.typedinterface.generateCSharp` function. The function creates code with handle copy behavior. That is, functions and methods can modify the input data in the calling workspace without making copies.

The function does not generate code to support dynamic properties, the `set` and `get` behavior for graphics handles, listener workflows, or the `findprop` and `findobj` handle methods. For more information, see Use MATLAB Handle Classes in C#.

### Call MATLAB from .NET: Support for comparison operators

The `matlab.engine.typedinterface.generateCSharp` function generates C# comparison methods and the `isvalid` function as well as comparison operators for both handle and value classes, which implement the `eq`, `ne`, `lt`, `gt`, `le`, and `ge` methods in MATLAB.

Code generated for comparison operators like `==` have a Boolean output type. If you override the `eq` function in MATLAB with a non-Boolean output, then you must use the `eq` method in your application instead of `==`.

### Call MATLAB from .NET: Support for IConvertible interface

You can use the .NET `IConvertible` interface to convert the value of a reference or value type to a common language runtime type with an equivalent value.

| C# Code (R2023b and earlier)                                                                                        | C# Code (Since R2024a)                                                                         |
|---------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------|
| <pre>MATLABArray retval = matlab.eval("int32(42)"); dynamic retvalDynamic = retval; double d = retvalDynamic;</pre> | <pre>MATLABArray retval = matlab.eval("int32(42)"); double d = Convert.ToDouble(retval);</pre> |

Widening conversion occurs when a value of one type is converted to another type that is of equal or greater size. The .NET type does not need to exactly match the MATLAB type, if the conversion is lossless.

```
double d = matlab.eval("int32(42)");
float f = matlab.eval("uint8(42)");
```

For conversion information, see MATLAB Numeric Types in .NET.

**Call MATLAB from .NET: Use Namespaces name-value argument, renamed from Packages, with matlab.engine.typedinterface.generateCSharp function**

In the `matlab.engine.typedinterface.generateCSharp` function, the `Packages` name-value argument is now named `Namespaces`. The behavior of this name-value argument remains the same. There are no plans to remove support for existing references to the argument.

**Compilers: Support for MinGW64 Compiler (Fortran)**

MATLAB supports the MinGW64 Compiler (Fortran) for building Fortran MEX files and standalone MATLAB engine and MAT-file applications on Windows platforms. For installation instructions, see MATLAB Support for MinGW-w64 C/C++/Fortran Compiler. For an up-to-date list of supported compilers, see Supported and Compatible Compilers.

**▲ Functionality being removed or changed**

**Ctrl+C stops execution of webread, webwrite, websave, and functions in the HTTP interface**

*Behavior change*

Pressing **Ctrl+C** while executing the MATLAB functions `webread`, `webwrite`, and `websave`, as well as functions in the MATLAB HTTP interface, stops execution of the MATLAB command. This behavior is consistent with how other MATLAB functionality handles **Ctrl+C**, as described in Use Keyboard Shortcuts to Navigate MATLAB. Previously, the functions threw a custom exception that you could catch inside a `try/catch` block. If your code contains `try/catch` blocks, you can update your code by removing them. For an example, see Ctrl+C stops execution of MATLAB command in the `matlab.net.http.Disposition` class.

Additionally, the `matlab.net.http.Disposition` enumeration class no longer supports the `Interrupt` enumeration member. Previously, `matlab.net.http.Disposition.Interrupt` created a `matlab.net.http.LogRecord` object if you interrupted a command by pressing **Ctrl+C**.

**load function no longer restores COM object from MAT-file**

*Behavior change*

If you save a COM object from the workspace to a file using the `save` function, you can no longer reload it into the workspace as a COM object using the `load` function. Starting in R2024a, the `load`

function displays a warning when trying to load COM objects and loads the data using another data type.

Previously, `load` attempted to load COM objects, but the restored data was limited. For example, if you saved an instance of a Microsoft Excel® COM server to a file, MATLAB saved the Excel session, but did not save any objects created in the session. If you tried to load the object into the workspace, `load` did not fully restore the original COM object.

### **.NET Interface: .NET 5 is no longer supported**

#### *Errors*

Support for .NET 5 is discontinued. For continued support for your applications, upgrade to a supported version of .NET. For information about .NET versions compatible with MATLAB, see [MATLAB Interfaces to Other Languages](#).

### **Perl 5.38.2: MATLAB support on Windows**

#### *Behavior change*

As of R2024a, MATLAB on Windows ships with an updated version of Perl, version 5.38.2.

If you use the `perl` command on Windows platforms, see <https://www.perl.org/> for information about using this version of the Perl programming language. For a standard distribution of `HTML::Parser`, source code, and information about using `HTML::Parser`, see <https://metacpan.org/pod/HTML:Parser>.

## Hardware Support

### Hardware Boards, Cameras, and Drones: Documentation for hardware support packages moved to MATLAB documentation

Starting in R2024a, the documentation for these support packages is included in the MATLAB documentation and all updates will be announced in the MATLAB release notes:

- MATLAB Support Package for Arduino Hardware
- MATLAB Support Package for Raspberry Pi Hardware
- MATLAB Support Package for BeagleBone® Black Hardware
- MATLAB Support Package for LEGO® MINDSTORMS® EV3 Hardware
- MATLAB Support Package for USB Webcams
- MATLAB Support Package for IP Cameras
- MATLAB Support Package for Parrot® Drones
- MATLAB Support Package for Ryze Tello Drones

In previous releases, the support package documentation was installed when you installed the support package software. To access archived release notes from the previous release, use these links:

- [R2023b Arduino Hardware Release Notes](#)
- [R2023b Raspberry Pi Hardware Release Notes](#)
- [R2023b BeagleBone Black Hardware Release Notes](#)
- [R2023b LEGO MINDSTORMS EV3 Hardware Release Notes](#)
- [R2023b USB Webcams Release Notes](#)
- [R2023b IP Camera Release Notes](#)
- [R2023b Parrot Drones Release Notes](#)
- [R2023b Ryze Tello Drones Release Notes](#)

### Arduino Hardware: Support on macOS with Apple silicon

You can now use MATLAB Support Package for Arduino Hardware on macOS with an Apple silicon processor in the installed version of MATLAB.

### Arduino Hardware: Read data from ADIS16505 sensor

MATLAB Support Package for Arduino Hardware enables you to read linear acceleration and angular velocity along the x-, y-, and z- axes as well as temperature from the ADIS16505 sensor connected to the SPI bus of an Arduino board. Use the new `adis16505` object and associated functions to create a connection to the ADIS16505 sensor and read values.

### Arduino Compatible Boards: ESP32 core version update

MATLAB Support Package for Arduino Hardware now supports ESP32 core version 2.0.11. The updated core is installed when you install the support package.

## **Raspberry Pi Hardware: Support for Debian Bullseye with kernel 6.1.21 and above**

MATLAB Support Package for Raspberry Pi Hardware now supports Debian Bullseye with kernel 6.1.21 and above. This support is available when you use the support package in the installed version of MATLAB or in MATLAB Online.

## **Raspberry Pi Hardware: Support on macOS with Apple silicon**

You can now use MATLAB Support Package for Raspberry Pi Hardware on macOS with an Apple silicon processor. This support is available when you use the support package in the installed version of MATLAB or in MATLAB Online.

## **USB Webcams: Retrieve property metadata for webcam object using `propertyInfo` function**

You can use the `propertyInfo` function of a webcam object to retrieve metadata, such as Name, Type, Default, ReadOnly, and AllowedValues, for all the properties supported by your webcam.

## **USB Webcams: Use webcam objects as dictionary keys and values**

Starting in R2024a, you can use webcam objects as keys or values in a dictionary. You can also perform basic dictionary operations on webcam objects using related functions. For more information, see [Dictionaries](#).

## **⚠️ Functionality being removed or changed**

### **Raspberry Pi Hardware: Support for Stretch operating system has been removed**

#### *Errors*

Starting in R2024a, you can no longer use MATLAB Support Package for Raspberry Pi Hardware to communicate with Raspberry Pi hardware that uses the Raspbian Stretch operating system.



# R2023b

---

**Version: 23.2**

**New Features**

**Bug Fixes**

**Compatibility Considerations**

# Environment

## ★Live Editor Tasks: Create Live Editor task from selected code with interactive controls

You can convert a selection in the Live Editor that contains code, text, and interactive controls into your own Live Editor task. Supported controls include numeric sliders, numeric spinners, drop-down lists, check boxes, edit fields, buttons, and file browsers.

To create your own Live Editor task from a selection, after making the selection, go to the **Live Editor** tab, and in the **Code** section, select **Task > Convert to Task**. When prompted, enter a name for the task, the name of the task class definition file to create, an optional description of the task, and some optional keywords. MATLAB creates your Live Editor task from the selection and adds the task to the Live Editor task gallery.

To add your task to a live script, on the **Live Editor** tab, click **Task** and select your task from the list.

For more information, see [Create Live Editor Task from Selection](#).

## ★Live Editor Tables: Add tables containing text and images to live scripts and functions

You can add tables to your live scripts and functions to format text and images. To insert a table, go to the **Insert** tab and select **Table ▾**. Move the cursor over the grid to highlight the numbers of rows and columns you want and then click to add the table. To create a larger table, select the Insert a table button  and specify the numbers of rows and columns in the dialog box.

After inserting the table, to modify its rows and columns, right-click the table, select **Table**, and select from the available options. For more information, see [Format Text in the Live Editor](#).

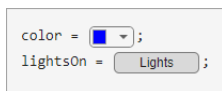
| Functions for Creating Line Plots                                           |                                                      |                                                                                     |
|-----------------------------------------------------------------------------|------------------------------------------------------|-------------------------------------------------------------------------------------|
| This table shows several common graphics functions for creating line plots. |                                                      |                                                                                     |
| Function                                                                    | Description                                          | Image                                                                               |
| <code>plot</code>                                                           | 2-D line plot                                        |  |
| <code>plot3</code>                                                          | 3-D point or line plot                               |  |
| <code>stairs</code>                                                         | Stairstep graph                                      |  |
| <code>errorbar</code>                                                       | Line plot with error bars                            |  |
| <code>area</code>                                                           | Filled area 2-D plot                                 |  |
| <code>stackedplot</code>                                                    | Stacked plot of several variables with common x-axis |  |

## ★Live Editor Controls: Add color pickers and state buttons to live scripts

You can add a color picker to your live script to select a color interactively. You also can add a state button to indicate and interactively set the value of a logical variable.



To add a color picker, go to the **Live Editor** tab, and in the **Code** section, select **Control > Color Picker**. To add a state button, select **State Button**.



For more information, see [Add Interactive Controls to a Live Script](#).

## Live Editor Controls: Select folders interactively in live scripts using file browser

You can use a file browser in a live script to select a folder interactively. To add a file browser, go to the **Live Editor** tab, and in the **Code** section, select **Control > File Browser**. To configure the control to select a folder rather than a file, right-click the control and select **Configure Control**. Then, in the **Type** section, select **Folder**.

For more information, see [Add Interactive Controls to a Live Script](#).

## Live Editor Accessibility: Interact with inline output using the keyboard

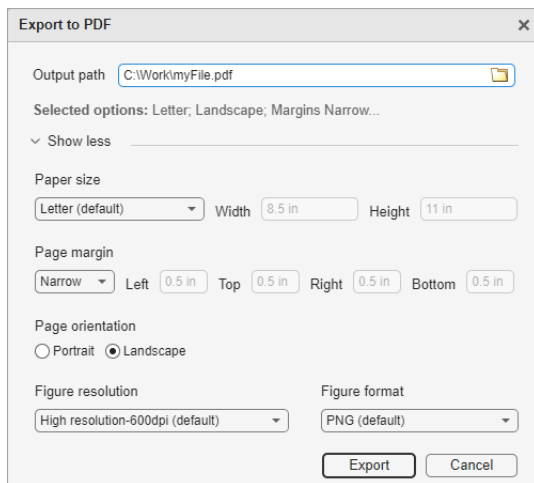
You can now use keyboard shortcuts to interact with inline output in live scripts. To move focus from the code to the output, use the down arrow and up arrow keys. To activate an output, press **Enter**. Once an output is activated, you can scroll text using the arrow keys, navigate through hyperlinks and buttons using the **Tab** key, and open the context menu by pressing **Shift+F10**. To deactivate an output, press **Esc**.

To disable using the keyboard to move focus to the output when output is inline, on the **Home** tab, in the **Environment** section, click  **Preferences**. Select **MATLAB > Editor/Debugger > Display**, and clear the **Focus outputs using keyboard when output is inline** option.

## Live Editor Export: Interactively customize export options, such as document paper size, orientation, margins, and figure format

When exporting to PDF, Microsoft Word documents, HTML, and LaTeX, you can use the Export dialog box to customize export options interactively. To open the Export dialog box, on the **Live Editor** tab, click **Export** and then select an export format. In MATLAB Online, click **Save** instead of **Export**.

You can change the paper size, orientation, and margins when exporting to PDF, Microsoft Word documents, and LaTeX. You also can change the resolution and format of figures when exporting to PDF, HTML, and LaTeX (requires the files to be run before exporting).



For more information, see [Ways to Share and Export Live Scripts and Functions](#).

## ★ **export Function: Convert live scripts and functions to Markdown files and Jupyter notebooks**

You can convert live scripts and functions to Markdown files and Jupyter notebooks by using the `export` function. To convert a live script or function to a Markdown file or Jupyter notebook, specify an output filename with a `.md` or `.ipynb` extension. Alternatively, you can set the `Format` name-value argument to "Markdown" or "ipynb".

For example, convert the live script "penny.mlx" to a Markdown file and a Jupyter notebook.

```
export("penny.mlx","penny.md")
export("penny.mlx","penny.ipynb")
```

## **Editor Execution: Run sections even if other sections contain syntax errors**

In the Editor and Live Editor, you can now run a code section even if another section in the file contains a syntax error. Previously, you could run a section only if the entire file contained no syntax errors.

For more information about creating sections, see [Create and Run Sections in Code](#).

## **Editor Variable Renaming: Automatically rename next references to variables or functions in file**

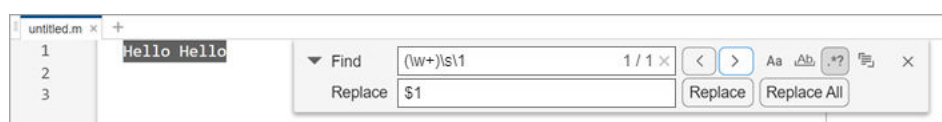
In the Editor and Live Editor, when you rename a variable or function in a file, you can choose whether to rename all instances or only the instances from the current cursor location to the end of the file. To rename all instances of a variable or function, press **Shift+Enter**. To rename only the instances from the current cursor location to the end of the file, press **Alt+Shift+Enter**. On macOS, use **Option+Shift+Enter** instead.

For more information, see [Find and Replace Text in Files and Go to Location](#).

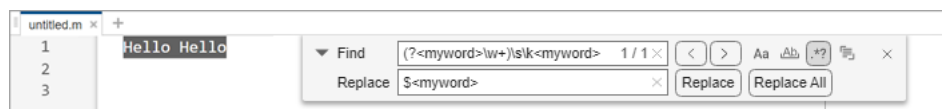
## Find and Replace Dialog Box: Use capture groups in regular expressions to search for and replace groups of characters

Search for and replace groups of characters in a file using capture groups in regular expressions. To begin searching using regular expressions, on the **Editor** or **Live Editor** tab, in the **Navigate** section, click  **Find**. Then, select the Regular expression option .

To create a capture group, surround the characters that you want to group with parentheses. Then, to access the capture group within the regular expression, use the format `\number`, where *number* refers to the capture group number. To access the capture group within a replacement pattern, use the format `$number`. For example, to find duplicate words in a file, use the expression `(\w+)\s\1`. To replace the two words with just one word, use the expression `$1`.



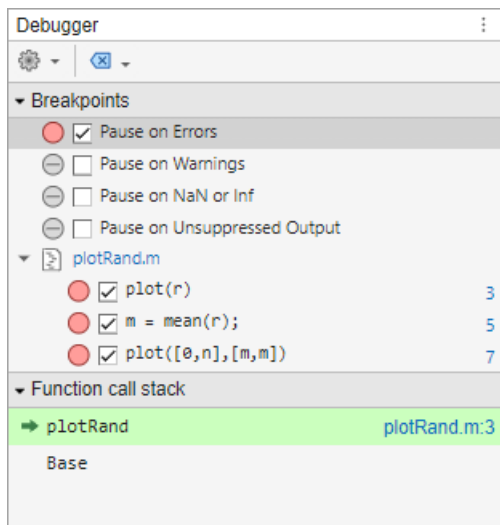
You also can create a named capture group using the format `?<name>`, where *name* is the name of the capture group. Then, to access the named capture group, use the format `\k<name>` within the regular expression, or `$<name>` within a replacement pattern. For example, to find duplicate words using a named capture group, use the expression `(?<myword>\w+)\s\k<myword>`. To replace the two words with just one word, use the expression `$<myword>`.



For more information about searching using the find and replace dialog box, see [Find and Replace Text in Files and Go to Location](#).

## Debugging in MATLAB Online: Manage breakpoints and navigate the function call stack using Debugger panel

Use the Debugger panel to manage breakpoints and navigate the function call stack while debugging in MATLAB Online. To open the Debugger panel, go to the **Editor** or **Live Editor** tab, and in the **Analyze** section, click  **Debugger**. You also can open the panel using the Open more panels button **(\*\*\*)** in the sidebar.

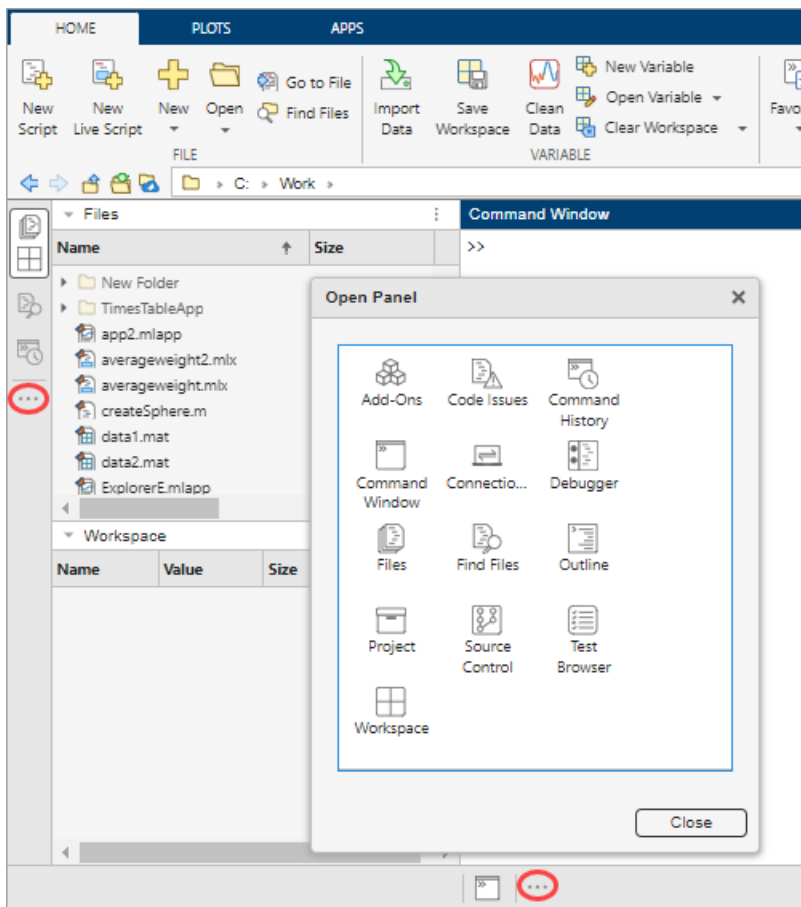


For more information, see [Debug MATLAB Code Files](#).

## Desktop Layout in MATLAB Online: Open additional panels directly from sidebars

In MATLAB Online, you can open additional panels (also referred to as tools) directly from the sidebars. To open additional panels, in a sidebar, click the Open more panels button (\*\*). Then, in the Open Panel dialog box, select from the available panels. You also can right-click in a sidebar and select **Open more panels**.

For example, to open the Code Issues panel, click the Open more panels button (\*\*), and select the Code Issues panel.



For more information, see [Change Desktop Layout](#).

## Themes in MATLAB Online: Save color customizations for individual themes

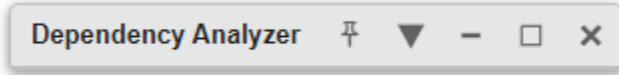
MATLAB Online now saves your color customizations for each available theme. If you customize the colors of the MATLAB Online desktop and then the MATLAB theme changes to a new theme or to match a change to the operating system color scheme, your customizations are saved.

With this change, in MATLAB Online, programmatically customizing syntax highlighting colors using the `matlab.colors` settings is no longer supported. To customize syntax highlighting colors, use the preferences in the **MATLAB Appearance Colors Preferences** page of the Preferences window instead. For more information, see [Change Desktop Colors](#).

In an installed version of MATLAB, programmatically customizing syntax highlighting colors using the `matlab.colors` settings is still supported.

## Window Management in MATLAB Online: Automatically collapse windows

In MATLAB Online, when you click the MATLAB background, open windows in the foreground collapse. Collapsing a window reduces the size of the window by displaying only its title bar.



To restore the collapsed windows, click the title bar of one of the previously open windows. To prevent a window from being collapsed or minimized, click  on its title bar.

For more information, see [Manage Windows in MATLAB Online](#).

## ★MATLAB on Macs: Better performance and improved battery life on MacBooks with MATLAB and Simulink running natively on Apple silicon

You can run MATLAB and Simulink natively on MacBook computers with Apple silicon.

## MATLAB on Macs: Java Runtime Environment required for MATLAB and MATLAB Runtime on Apple silicon

On Mac computers with Apple silicon, if MATLAB or MATLAB Runtime does not find a supported Java Runtime Environment (JRE™), the program displays a dialog box with a link to information about the Java requirement and the recommended solution. For information about the recommended solution, see <https://www.mathworks.com/support/requirements/apple-silicon.html>. For more information, see [Start MATLAB on macOS Platforms](#).

## MATLAB Drive: New URL for MATLAB Drive Online (January 2024)

You can now access MATLAB Drive Online at <https://drive.mathworks.com>. Previously, MATLAB Drive Online was located at <https://drive.matlab.com>.

## ⚠Functionality being removed or changed

### MATLAB Online keyboard shortcuts Ctrl+0, Ctrl+1, Ctrl+2, and Ctrl+3 have been removed

The **Ctrl+0**, **Ctrl+1**, **Ctrl+2**, and **Ctrl+3** keyboard shortcuts have been removed in MATLAB Online to avoid a conflict with the shortcuts to switch tabs in certain browsers. These shortcuts were previously used to make the Command Window, Command History, Files panel, or Workspace panel active. On macOS, the corresponding keyboard shortcuts (**Command+0**, **Command+1**, **Command+2**, and **Command+3**) have also been removed.

To make one of these tools active, use **Ctrl+F6** and **Ctrl+Shift+F6** to move forward or backward through the different areas of the MATLAB Online desktop until your intended tool is selected. Then, to navigate into the tool, press the **Tab** key. On macOS, use the **Command** key instead of the **Ctrl** key.

### XML comparison type for visdiff function will be removed

*Warns*

The XML comparison type for the `visdiff` function will be removed in a future release. Overriding the default comparison type by specifying "xml" will not be supported in a future release. In R2023b, scripts that use `visdiff(filename1,filename2,"xml")` issue a warning.

## Language and Programming

### **configureDictionary Function: Create dictionaries with configured keys and values**

Use the `configureDictionary` function to create an empty dictionary with specified types for keys and values.

For example, create a dictionary that is configured to accept strings as keys and doubles as values.

```
d = configureDictionary("string","double")  
  
d =  
  
    dictionary (string [] double) with no entries.
```

### **lookup, insert, and remove Functions: Find, add, and remove dictionary entries**

You can perform basic dictionary operations using these three new functions:

- **lookup** — Find the value that corresponds to a key. You can optionally specify a fallback value for the function to return if it cannot find the entry.
- **insert** — Add new dictionary entries. You can optionally specify whether to overwrite existing entries.
- **remove** — Remove dictionary entries.

### **codeIssues Object: Export issues identified in code**

Use the `export` object function of the `codeIssues` object to write its issues to a file. Exported files are in SARIF format by default, but you can specify to export issues in SonarQube or MATLAB encoded JSON formats.

### **Code Compatibility Analyzer App: Group and filter code issues with improved interface**

The Code Compatibility Analyzer app has a new interface that makes it easier to navigate issues found in code. You can now adjust how identified issues are grouped and filter results by text, severity, and issue type.



## Code Compatibility Analyzer

Analysis Date: 4/28/2023, 10:23:15 AM

Active Issues Only

This report lists instances of syntax errors, incompatibilities in your code and new functionalities that might improve your code. Syntax errors are not incompatibilities, but result in nonrunnable code and impact compatibility analysis. Some of the checks might flag code that is correct and does not need to be updated. The action indicates the likeliness of false positives. Code Compatibility Report is continuously being improved. There might be some incompatibilities in your code or some new functionalities that are not yet reported.  
 MATLAB Release: R2023b

80

Total Files Analyzed



10  
Error



20  
Warning



8  
Info

### Code Health Details [expand all](#)

Filter by Text

≡ Group by Issue Type

Filter by Severity

Filter by Issue Type

#### Incomplete Analysis (1)

[Help](#)

- Code analysis did not complete. Code Analyzer encountered an error. (1)

#### Syntax Errors (1)

[Help](#)

- A '(' might be missing a closing ')', causing invalid syntax at end of line. (1)

#### Compatibility Considerations (11)

[Help](#)

- 'bin2gray' has been removed. Use the appropriate modulation object or function to remap constellation points inst...
- 'gray2bin' has been removed. Use the appropriate modulation object or function to remap constellation points inst...
- 'zerom' has been removed. With appropriate code changes, use 'zeros' instead. (1)
- 'onem' has been removed. With appropriate code changes, use 'ones' instead. (1)
- 'nanm' has been removed. With appropriate code changes, use 'nan' instead. (1)
- 'spzerom' has been removed. With appropriate code changes, use 'sparse' instead. (1)
- 'wavinfo' has been removed. With appropriate code changes, use 'audioinfo' instead. (1)

## Class Diagram Viewer: Add known subclasses to diagrams

You can now automatically add one or more layers of known subclasses of a class in a diagram. Right-click the class in a diagram to see the options for adding subclasses. See Class Diagram Viewer for more information.

## mustBeSparse Function: Validate sparsity of arguments

Use the `mustBeSparse` validation function to check whether an argument is sparse. This function is designed for use in function argument validation and property validation.

## MATLAB backgroundPool Function: load function supported in background threads

Use the `load` function in thread-based environments, including the MATLAB backgroundPool. Saving Version 7.3 MAT-files is not supported.

### ⚠️ Functionality being removed or changed

#### Arrays contained in structure fields, cells, and class properties are displayed with partial contents as well as size and data type

*Behavior change*

Between R2021b and R2023b, MATLAB displayed the entire contents of an array in a structure field, cell, or class property if the contents fit in the display. If the array did not fit, MATLAB showed as much of the array as the display allowed.

Starting in R2023b, if the entire contents of an array in a structure field, cell, or class property do not fit in the display, MATLAB shows as much of the array as the display allows, as well as the size and data type of the array. For example, as of R2023b, MATLAB displays partial contents as well as size and data type for a cell that contains an array of 100 double values.

```
D = {3.14,[1:100]};  
D(2)
```

```
ans =
```

```
1×1 cell array
```

```
{[1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 ... ] (1×100 double)}
```

The `disp` function follows the same behavior for structures, cells, and class properties.

For more information, see the Version History notes in `cell` and `struct`.

#### Using the AbortSet attribute and a get method that sets the property value can cause infinite recursion

*Behavior change*

Generally, MATLAB calls a property set method, if one exists, when a property is set outside of the set method. When a property has a set method and uses the `AbortSet` attribute, MATLAB acquires the old value and compares it with the new value for any assignment to that property outside of its set method. If the property has a get method, the get method is called to acquire the old value.

For R2023b, the get method is no longer bypassed when it indirectly results in another call to the get method because of the `AbortSet` check. This removes overhead from many property access operations. As a result, MATLAB calls the get method again to acquire the old property value when a property fits all three of these conditions:

- The property uses the `AbortSet` attribute.
- The property has a set method.
- The property has a get method that assigns a value to the property.

This call to get could lead to infinite recursion if the property assignment in the get method is unconditional.

**Malformed and reserved import arguments issue warning***Behavior change*

Previously, the `import` function accepted malformed arguments, such as `#.b` or `a.*.*`, and MATLAB still executed the command even though nothing was imported. It also allowed importing of reserved names, such as `a.for` or `a.if` that do not satisfy MATLAB naming rules. Starting in R2023b, MATLAB issues a warning if you include invalid imports. Such imports will not be supported in a future release.

**ver and verlessthan are not recommended***Still runs*

Using `ver` to return the current MATLAB release information is not recommended. Use `matlabRelease` instead.

Using `verlessthan` to determine if the current MATLAB release is older than a specified release is not recommended. Use `isMATLABReleaseOlderThan` instead.

**Change to precedence for constructors in class folders***Behavior change in future release*

When a class and a function on the path have the same name, MATLAB currently gives precedence to the class constructor if the class is defined inside a class folder (@-folder). The constructor takes precedence even if the function appears before the class on the path.

In a future release, this precedence rule will be removed. In other words, if a function appears before a class of the same name on the path, the function will have precedence regardless of whether the class is defined in a class folder. Change the order of these items on your path to maintain your preferred precedence.

**Defining classes and packages: Using `schema.m` will not be supported in a future release***Still runs*

Support for classes and packages defined using `schema.m` files will be removed in a future release. Replace existing schema-based classes with classes defined using the `classdef` keyword.

## Data Analysis

### ★ Experiment Manager App: Design experiments to run MATLAB code, and visualize, filter, and compare results

Create an experiment to run your MATLAB code using a range of parameter values and compare the results by using the Experiment Manager app. For example, you can use an experiment to explore how the solution to a system of differential equations responds to different coefficient values or how it evolves from different initial conditions. When you run the experiment, Experiment Manager runs your code in multiple trials, each one using a different combination of parameter values.

Experiment Manager provides visualizations, filters, and annotations to help you manage your experiment results and record your observations. To improve reproducibility, Experiment Manager stores a copy of the experiment definition every time that you run an experiment. You can access past experiment definitions to keep track of the parameter values and MATLAB code that produce each of your results.

By default, Experiment Manager runs one trial at a time. If you have Parallel Computing Toolbox, you can run multiple trials at the same time or run a single trial at a time on multiple GPUs, on a cluster, or in the cloud. If you have MATLAB Parallel Server™, you can also offload experiments as batch jobs in a remote cluster so that you can continue working or close your MATLAB session while your experiment is running.

If you have Deep Learning Toolbox™ or Statistics and Machine Learning Toolbox, you can use Experiment Manager for your AI workflows. For more information, see Experiment Manager (Deep Learning Toolbox) or Experiment Manager (Statistics and Machine Learning Toolbox).

### ★ Pivot Table Live Editor Task: Interactively summarize tabular data in pivoted table

Use the Pivot Table task in the Live Editor to interactively perform a pivoting operation on data in a table or timetable. You can filter the input table, specify grouping variables to designate pivoted table rows and columns, customize the format and contents of the pivoted table, and visualize the pivoted table in a chart. To add the task to a live script in the Live Editor, click **Task** on the **Live Editor** tab and select the **Pivot Table** icon.

### resize, paddata, and trimdata Functions: Change the size of data by adding or removing elements

Change the size of array or tabular data by using the `resize`, `paddata`, and `trimdata` functions. You can specify the dimensions to operate along, the fill value or pattern for padding, and the side of the input data for resizing.

- `resize` adds or removes elements depending on if the length of the input data is less than or greater than the target length. The resized data matches the target length.
- `paddata` only adds elements. If the length of the input data is greater than or equal to the target length, the output data is the same as the input data.
- `trimdata` only removes elements. If the length of the input data is less than or equal to the target length, the output data is the same as the input data.

## smoothdata2 Function: Smooth noisy data in two dimensions

Smooth noisy entries in two-dimensional data sets with a moving window method by using the `smoothdata2` function. You can customize the smoothing method and moving window size, specify how missing values are handled, adjust the level of smoothing, and specify the sample points used by the smoothing method.

## kde Function: Estimate a probability function by using kernel density estimation for univariate data

Estimate a probability function with kernel density estimation for univariate data by using the `kde` function. Optionally define parameters such as the probability function to estimate, points at which to evaluate the estimated probability function, and type of kernel smoothing function.

## Join Tables Live Editor Task: Sort output timetable by row times when row times are not key values

When the left input of the Join Tables task in the Live Editor is a timetable, you can sort the output timetable by row times even when you do not specify row times as key values. To sort by row times in this case, select the **Sort result by row times** check box. In previous releases, Join Tables does not provide this sorting option.

This sorting option is available only when all three of these conditions are true:

- The left input is a timetable that is sorted by row times.
- You specify an outer join or an inner join.
- You do not specify row times as key values.

In all releases, the output timetable is automatically sorted by row times when you specify that row times are key values.

## Data Cleaning: Interactively fill missing data with values from nearest neighbor rows

Fill missing entries with the mean of nearby points by using the `K-nearest neighbors` fill method in the Clean Missing Data task in the Live Editor or the **Clean Missing Data** cleaning method in the Data Cleaner app. Specify the number of neighbors, and define the distance between rows using the Euclidean distance, the scaled Euclidean distance, or in the Clean Missing Data task, a custom function.

## Live Editor: Display the contents of nested tables as variables

Starting in R2023b, the output of the Live Editor displays the contents of tables or timetables that contain nested tables or timetables as variables inline. You can interactively manipulate a nested table, for example, by selecting or sorting a variable or copying a variable with its headers. Previously, the output of the Live Editor displayed the dimensions of nested tables, but not the nested table contents.

For example, use the Pivot Table task to create nested tables. The Live Editor output displays the complete pivoted table contents.

P = 12x5 table

|   | year_OutageTime | severe storm |            |           |            | thunderstorm |            |           |            |
|---|-----------------|--------------|------------|-----------|------------|--------------|------------|-----------|------------|
|   |                 | MidWest      |            | NorthEast |            | MidWest      |            | NorthEast |            |
|   |                 | [0, 700)     | [700, Inf] | [0, 700)  | [700, Inf] | [0, 700)     | [700, Inf] | [0, 700)  | [700, Inf] |
| 1 | 2002            | 2            | 0          | 4         | 0          | 0            | 0          | 2         | 3          |
| 2 | 2003            | 3            | 0          | 4         | 1          | 5            | 1          | 3         | 0          |
| 3 | 2004            | 8            | 1          | 5         | 1          | 8            | 0          | 8         | 1          |
| 4 | 2005            | 3            | 1          | 7         | 1          | 4            | 3          | 3         | 0          |
| 5 | 2006            | 7            | 0          | 5         | 1          | 11           | 1          | 10        | 2          |
| 6 | 2007            | 7            | 1          | 6         | 0          | 5            | 0          | 1         | 0          |

## Live Editor: Copy output data as tab-delimited text

You can copy data from the output of the Live Editor as tab-delimited text. Select some data in the output, and copy the data by pressing **Ctrl+C** or by right-clicking the data and selecting **Copy Selection**. Then, paste the tab-delimited text. The pasted text respects the rows and columns from the Live Editor output. For example, select and copy the elements in a row vector, and paste the selection into a Microsoft Excel worksheet. The first element of each worksheet column contains an element from the copied vector.

## pivot Function: Display empty groups in pivoted table

Include, rather than omit, empty groups in the pivoted table returned by the `pivot` function by setting the `IncludeEmptyGroups` name-value argument to `true`. An empty group occurs when a possible value of a variable specified by `Columns` or `Rows` is not represented in the input table, such as in a categorical, logical, or binned numeric variable.

## pivot Function: Specify row names for pivoted table as row group names

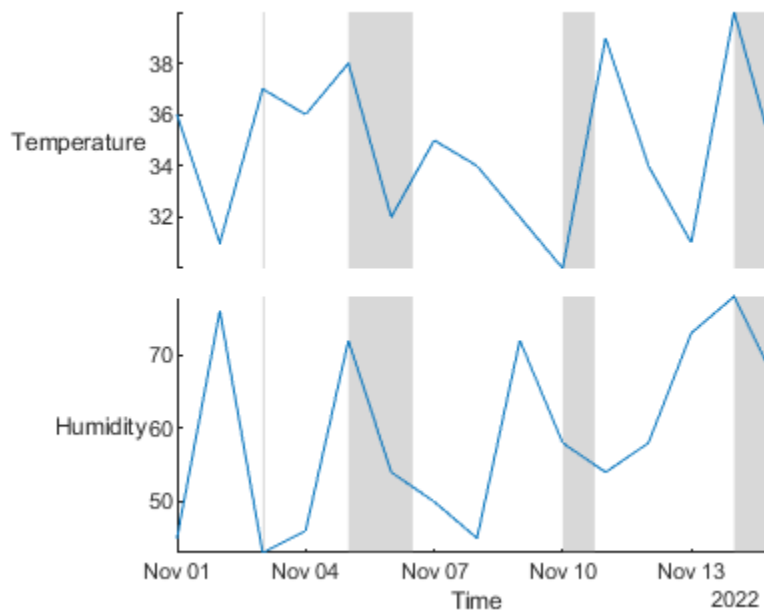
In a pivoted table returned by the `pivot` function, you can now place row labels to the left of the leftmost table variable by setting the `RowLabelPlacement` name-value argument to `"rownames"`. This option sets the `RowNames` property of the pivoted table to the row group names. If `Rows` specifies multiple variables, the row labels are the group names concatenated with an underscore. Previously, the pivoted table always placed row labels in the leftmost table variables.

## stackedplot Function: Plot events associated with timetables

The `stackedplot` function can plot events as lines or shaded regions on stacked plots created from timetables. To plot events associated with a timetable, you must attach an event table to it before you call `stackedplot`. For more information on event tables, see `eventtable`.

To support events, `stackedplot` has a new `EventsVisible` name-value argument:

- If `EventsVisible` is `"on"`, then `stackedplot` plots events as lines or shaded regions.
- If `EventsVisible` is `"off"`, then `stackedplot` hides events.



## Timetable Events: Perform calculations directly on event tables without extracting their data

You can now perform calculations directly on event tables without extracting their data. In R2023a, you cannot perform direct calculations on event tables, though you can perform direct calculations on tables and timetables.

For more information, see [Direct Calculations on Tables and Timetables](#) and [Rules for Table and Timetable Mathematics](#).

## containsrange, overlapsrange, and withinrange Functions: Specify time ranges by using event filters

You can specify a time range by using event filters when you use the `containsrange`, `overlapsrange`, or `withinrange` function. To use event filters, you must first attach an event table to the input timetable. For more information on using these functions with event filters, see [eventfilter](#).

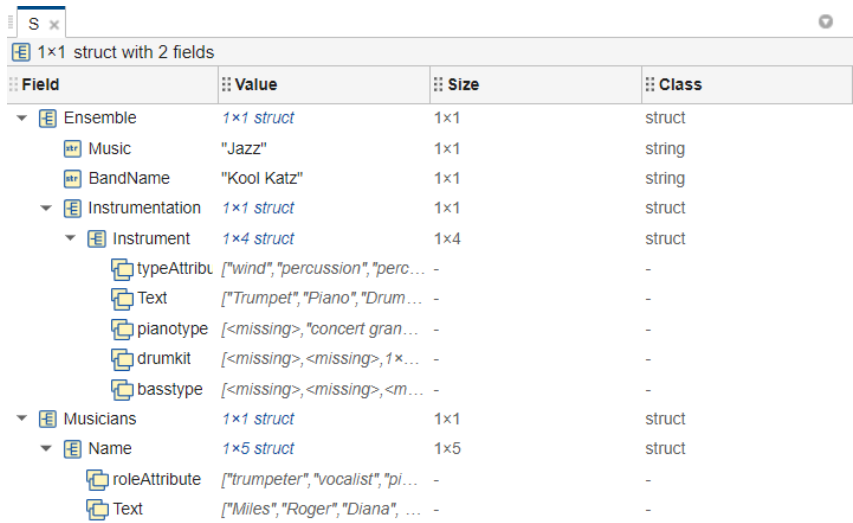
## Variables Editor in MATLAB Online: Interactively change display format for datetime or duration data

In the Variables editor in MATLAB Online, you can interactively modify the display format for `datetime` or `duration` variables. Right-click the variable or column header, select **Modify Date/Time Format**, and choose from the list of display formats or specify a custom display format. The Command Window displays the corresponding code. Previously, interactively changing the display format for `datetime` data was supported only in the installed version of MATLAB. Interactively changing the display format for `duration` data was not supported.

## Variables Editor in MATLAB Online: View and interactively edit scalar structures, object arrays, and tables with grouped variables

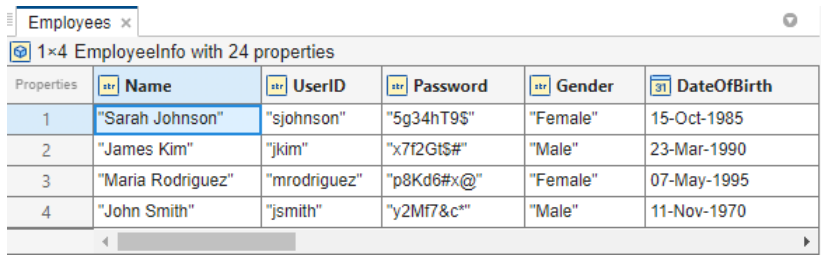
The Variables editor in MATLAB Online has improved functionality for these data types:

- Scalar structure — You can now view, expand, and interactively edit the fields and nested fields in a scalar structure. Previously, you had to open each field in another tab in the Variables editor to view and edit its contents.



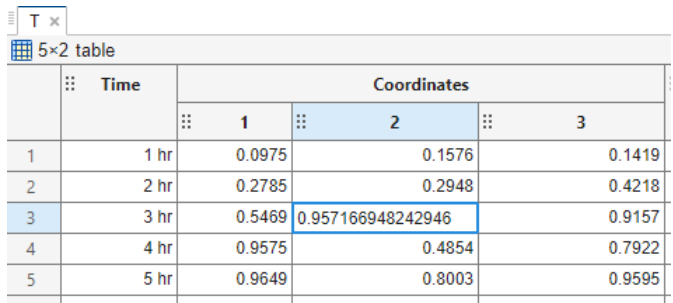
| Field           | Value                            | Size | Class  |
|-----------------|----------------------------------|------|--------|
| Ensemble        | 1×1 struct                       | 1×1  | struct |
| Music           | "Jazz"                           | 1×1  | string |
| BandName        | "Kool Katz"                      | 1×1  | string |
| Instrumentation | 1×1 struct                       | 1×1  | struct |
| Instrument      | 1×4 struct                       | 1×4  | struct |
| typeAttrib      | ["wind", "percussion", "perc...  | -    | -      |
| Text            | ["Trumpet", "Piano", "Drum...    | -    | -      |
| pianoType       | [<missing>, "concert gran...     | -    | -      |
| drumkit         | [<missing>, <missing>, 1×...     | -    | -      |
| basstype        | [<missing>, <missing>, <m...     | -    | -      |
| Musicians       | 1×1 struct                       | 1×1  | struct |
| Name            | 1×5 struct                       | 1×5  | struct |
| roleAttribute   | ["trumpeter", "vocalist", "pi... | -    | -      |
| Text            | ["Miles", "Roger", "Diana", ...  | -    | -      |

- Object array — You can now view and interactively edit the values of common properties of objects in an object array. Previously, you had to open each object in a separate tab in the Variables editor to view and edit its property values.



| Properties | Name              | UserID       | Password    | Gender   | DateOfBirth |
|------------|-------------------|--------------|-------------|----------|-------------|
| 1          | "Sarah Johnson"   | "sjohnson"   | "5g34hT9\$" | "Female" | 15-Oct-1985 |
| 2          | "James Kim"       | "jkim"       | "x7f2Gt\$#" | "Male"   | 23-Mar-1990 |
| 3          | "Maria Rodriguez" | "mrodriguez" | "p8Kd6#x@"  | "Female" | 07-May-1995 |
| 4          | "John Smith"      | "jsmith"     | "y2Mf7&c*"  | "Male"   | 11-Nov-1970 |

- Table with grouped variables — You can now select and interactively edit the contents of a subcolumn in a group of table variables. Previously, interactively editing a subcolumn in a group of table variables was not supported.



|   | Time | Coordinates |                   |        |
|---|------|-------------|-------------------|--------|
|   |      | 1           | 2                 | 3      |
| 1 | 1 hr | 0.0975      | 0.1576            | 0.1419 |
| 2 | 2 hr | 0.2785      | 0.2948            | 0.4218 |
| 3 | 3 hr | 0.5469      | 0.957166948242946 | 0.9157 |
| 4 | 4 hr | 0.9575      | 0.4854            | 0.7922 |
| 5 | 5 hr | 0.9649      | 0.8003            | 0.9595 |



## Accessibility in MATLAB Online: Use a screen reader to interact with the Workspace browser, Variables editor, Property Inspector, and Import Tool

In MATLAB Online, when using a screen reader, you hear additional information about these interactions:

- When you add variables to or remove variables from the MATLAB workspace, you hear the number of variables added or removed.
- When you open a variable in the Variables editor, you hear the summary of the variable.
- When you edit a graphics object property using a drop-down list in the Property Inspector, you hear the value. Previously, all Property Inspector components had screen reader support except for drop-down list items.
- When you successfully import a variable using the Import Tool, you hear the success message after the import is complete.

## Histogram Distribution Charts: Calculate values and create histograms using percentages

You can create histograms with percentages on the vertical axis by using the `histogram` and `histogram2` functions. You can normalize histogram values as percentages by using the `histcounts` and `histcounts2` functions. For each function, set the `Normalization` name-value argument to "percentage".

## tallrng Function: Specify random number algorithm without specifying seed

When using the `tallrng` function, you can specify just the algorithm for the random number generator to use. The `tallrng(generator)` syntax allows you to set the random number algorithm without specifying the seed, where `tallrng` uses a seed of 0. This syntax is equivalent to `tallrng(0,generator)`. For example, `tallrng("philox")` initializes the Philox 4x32 generator with a seed of 0.

## ⚠️ Functionality being removed or changed

### min and max functions will no longer support specifying second input array as character array

*Warns*

For the `min` and `max` functions, specifying a second input array as a character array issues a warning and will not be supported in a future release. This change minimizes confusion with options that can be specified as character vectors, such as the missing value condition. To maintain the previous functionality, you can convert the second input array to `double`, for example, `max(A,double(B),'includenan')`.

## Data Import and Export

### **readstruct Function: Read data from JSON files**

Read data from JSON files into MATLAB structure arrays using the `readstruct` function. For example, `S = readstruct("myFile.json")` creates a structure `S` from `myFile.json`.

### **writestruct Function: Write structure arrays to JSON files**

Write MATLAB structure arrays to JSON files using the `writestruct` function. For example, `writestruct(S, "myFile.json")` writes the data in structure `S` to `myFile.json`.

### **Parquet: Create ParquetDatastore objects more efficiently with partition control in parallel environments**

In parallel environments, you can create a `ParquetDatastore` object more efficiently by specifying the unit of partition and the size of partition blocks. Specify the new `PartitionMethod` and `Blocksize` name-value arguments during creation of the datastore.

### **cmdsep Function: Return command separator for current platform**

Return the platform-specific command separator character using the `cmdsep` function. You can create a chain of commands by using `cmdsep` with `system`. For example, `system("cd myfolder/" + cmdsep + "matlab")` opens a second instance of MATLAB in `myfolder`.

### **NetCDF Interface: Import netCDF data using Import Tool in MATLAB Online and Import Data Live Editor Task**

You can import netCDF data by using the Import Tool app in MATLAB Online or by using the Import Data Live Editor task in live scripts. Using these options, you can:

- Visualize the structure and contents of netCDF data interactively.
- Search for specific variables and attributes.
- Graphically select and deselect variables and attributes to import.
- Generate reusable MATLAB code to import netCDF data.

### **NetCDF Interface in MATLAB Online: Import netCDF data by double-clicking netCDF files**

You can launch the Import Tool app to import netCDF data by double-clicking a netCDF file in MATLAB Online.

### **Image File Format Libraries: LibTIFF library upgraded to version 4.5.0**

The LibTIFF library is upgraded to version 4.5.0.

## Scientific File Format Libraries: HDF4 library upgraded to version 4.2.16

The HDF4 library is upgraded to version 4.2.16.

## Scientific File Format Libraries: HDF5 library upgraded to version 1.10.10

The HDF5 library is upgraded to version 1.10.10.

## Scientific File Format Libraries: NetCDF library upgraded to version 4.9.1

The netCDF library is upgraded to version 4.9.1.

## serialbreak Function: Send break to device connected to serial port

Use the `serialbreak` function with a `serialport` object to send a serial break of a specified duration to your serial port device. For some devices, you can use the break signal as a way to clear the hardware buffer.

## ⚠️ Functionality being removed or changed

### **fileparts function output has same data type as input for all data types**

*Behavior change*

Starting in R2023b, when you use the `fileparts` function to get the parts of a filename, `fileparts` returns values of the same data type as the input, including scalar cell array inputs. Previously, a scalar cell array input produced character array outputs.

### **In the netCDF interface, use compression and shuffle filter only with fixed-length data types**

*Behavior change*

Compression and the shuffle filter can be used only with fixed-length data types and generate an error if applied to variables of type `NC_STRING` or `NC_VLEN`. Previously, if you specified compression or the shuffle filter for `NC_STRING` or `NC_VLEN` variables, the specification had no effect.

### **In the netCDF interface, compression level of zero disables deflate filter**

*Behavior change*

Setting the compression level to zero disables the deflate filter. Previously, the deflate filter could be enabled with a compression level of zero.

This table shows two examples of this behavior change.

| Example                                                                                                                                                                                                                                                                                                                                                                                                              | Result in R2023a and Earlier                                                                    | Result Starting in R2023b                                                                         |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------|
| Set the compression level to zero using <code>netcdf.defVarDeflate</code> .<br><br><pre>ncid = netcdf.create("myFile.nc", "NETCDF4"); latdimid = netcdf.defDim(ncid, "lat", 1800); varid = netcdf.defVar(ncid, "earthgrid", "double", latdimid); netcdf.defVarDeflate(ncid, varid, true, true, 0) [~, deflate, deflateLevel] = netcdf.inqVarDeflate(ncid, varid); netcdf.close(ncid)</pre><br>[deflate deflateLevel] | The deflate filter is enabled.<br><br><pre>ans =      0</pre>                                   | The deflate filter is disabled.<br><br><pre>ans =      0     0</pre>                              |
| Set the compression level to zero using <code>nccreate</code> with the <code>DeflateLevel</code> name-value argument.<br><br><pre>nccreate("myFile.nc", "earthgrid", Dimensions={ "lat" 1800 "col" 3600}, ...         Datatype="double", Format="netcdf4", ChunkSize=[180 360],         DeflateLevel=0) info = ncinfo("myFile.nc"); info.Variables.DeflateLevel</pre>                                                | The deflate filter is enabled and the compression level is zero.<br><br><pre>ans =      0</pre> | The deflate filter is disabled and the compression level is empty.<br><br><pre>ans =     []</pre> |

### **cdfepoch function is not recommended**

*Still runs*

There are no plans to remove `cdfepoch`. However, the `cdfepoch` function is not recommended. Use the `cdflib` low-level functions instead.

`cdfepoch` uses `datenum` and `datestr`, which are not recommended, and is also intended for use with `cdfwrite`, which is also not recommended.

### **todatenum function is not recommended**

*Still runs*

There are no plans to remove `todatenum`. However, the `todatenum` function is not recommended. Use the `cdfread` function instead.

The `todatenum` function converts `cdfepoch` objects to serial date numbers, which are not recommended. Update your code to convert this type of data to `datetime` or `double` values instead. For example, use `cdfread(filename, "DatetimeType", "datetime")`.

### **ConvertEpochToDatenum name-value argument of the cdfread function is not recommended**

*Still runs*

There are no plans to remove `ConvertEpochToDatenum`. However, the `ConvertEpochToDatenum` name-value argument of the `cdfread` function is not recommended. Use the `DatetimeType` name-value argument instead.

The `ConvertEpochToDatenum` argument converts CDF\_EPOCH data to serial date numbers, which are not recommended. Update your code to convert this type of data to `datetime` or `double` values instead. For example, use `cdfread(filename, "DatetimeType", "datetime")`.

## Mathematics

### Ordinary Differential Equations: Use object-oriented solution framework for ODEs

Solve ODE problems by using several new objects and methods:

- Use `ode` objects to define aspects of the ODE problem (such as initial time, initial value, equations, parameters, and so on) and set basic options (such as error tolerances). Use a specific solver by setting the value of `ode.Solver` explicitly, or allow the `ode` object to automatically select a solver based on properties of the problem by setting the value to `"auto"`, `"stiff"`, or `"nonstiff"`.
  - Common options are conveniently specified as properties of the `ode` object.
  - Less common options can be set in the `ode.SolverOptions` property. Separate objects of the form `matlab.ode.options.*` capture options specific to the selected solver, which simplifies the process of finding and setting relevant options.
  - Use the `solve` method to return the solution of the ODE problem at a discrete set of points, either points of your choosing or points based on solver steps.
  - Use the `solutionFcn` method to return the solution as a function that you can evaluate anywhere on a specified interval.
- For problems involving a mass matrix, use `odeMassMatrix` objects to store information about the mass matrix (such as state dependence or sparsity pattern).
- Use `odeJacobian` objects to store information about the Jacobian matrix for the problem (such as matrix elements or sparsity pattern).
- Use `odeEvent` objects to define and track events that occur during the solution of the ODE problem. The `odeEvent` object provides a new callback capability that simplifies solving systems with discontinuities occurring with events.

The new objects and methods simplify many aspects of solving ODEs, and all functionality that is available in existing ODE functions (`ode45`, `ode23`, `ode15s`, and so on) is also available in the objects. There are no plans to remove the existing ODE functions.

### rng Function: Change default algorithm and seed for random number generator, and specify random number algorithm without specifying seed

You can change the default algorithm and seed for the `rng` function from the MATLAB Preferences window. On the **Home** tab, in the **Environment** section, click  **Preferences**. Select **MATLAB > General**, and then select a different option for **Default algorithm** and select a different value for **Default seed** in the **Random Number Generation** preference.

To access and modify settings for the random number generator programmatically, you can access the `matlab.general.randomnumbers` settings using the root `SettingsGroup` object returned by the `settings` function. For example, show the default algorithm and seed that you have set for the random number generator.

```
s = settings;  
s.matlab.general.randomnumbers.DefaultAlgorithm  
s.matlab.general.randomnumbers.DefaultSeed
```

When you first start a MATLAB session or call `rng("default")`, MATLAB initializes the random number generator using the default algorithm and seed that you have set. If you do not change these settings, then `rng` uses the factory value of "twister" for the Mersenne Twister generator with seed 0, as in previous releases.

When you perform parallel processing (requires Parallel Computing Toolbox), by default, the MATLAB client uses the Mersenne Twister random number generator with seed 0 and the MATLAB workers use the Threefry 4x64 generator with 20 rounds with seed 0. Changing the default generator settings in the MATLAB Preferences window or using the `matlab.general.randomnumbers` settings affects only the default behavior of the client and does not affect the default behavior of the parallel workers.

When using the `rng` function, you can specify just the algorithm for the random number generator to use. The `rng(generator)` syntax allows you to set the random number algorithm without specifying the seed, where `rng` uses a seed of 0. This syntax is equivalent to `rng(0,generator)`. For example, `rng("philox")` initializes the Philox 4x32 generator with a seed of 0.

## **MATLAB Support Package for Quantum Computing: Solve Quadratic Unconstrained Binary Optimization (QUBO) problems**

The MATLAB Support Package for Quantum Computing enables you to create and solve QUBO problems. QUBO problems include a wide variety of combinatorial optimization problems, such as the Traveling Salesperson Problem with QUBO, the Capacitated Vehicle Routing Problem, and many others.

Create a QUBO problem using the `qubo` function. Solve the problem using the `solve` function. For more information, see [Workflow for QUBO Problems](#). For a complete guide to the QUBO functionality, see [Quadratic Unconstrained Binary Optimization \(QUBO\)](#).

To install the MATLAB Support Package for Quantum Computing, locate the support package in **Add-On Explorer** using the instructions in [Get and Manage Add-Ons](#).

## **MATLAB Support Package for Quantum Computing: Control circuit plot appearance and behavior with QuantumCircuitChart objects**

When you plot a circuit or composite gate with the `plot` method, you can now specify an output argument to return a `QuantumCircuitChart` object. Set properties of the object to control the appearance and behavior of the circuit plot. See [QuantumCircuitChart Properties](#) for more information.

## **MATLAB Support Package for Quantum Computing: Connect to quantum hardware through IBM (October 2023, Version 23.2.1)**

Run gate-based quantum algorithms by connecting to quantum hardware provided by the IBM® Qiskit® Runtime Services.

- Represent an IBM quantum device by using a `quantum.backend.QuantumDeviceIBM` object.
- Represent a task on an IBM quantum device by using a `quantum.backend.QuantumTaskIBM` object.

- Use the `run` function to run a quantum circuit on a `QuantumDeviceIBM` object.

To set up access using your IBM account, see [Run Quantum Circuit on Hardware Using IBM Qiskit Runtime Services](#).

## **deconv Function: Perform least-squares deconvolution with different convolved subsections**

You can now perform least-squares deconvolution by specifying the `Method` name-value argument as `"least-squares"` when using `deconv`. You can also specify different convolved subsections and the Tikhonov regularization factor with least-squares deconvolution.

In previous releases, `deconv` can perform deconvolution using only a polynomial long-division method. The new arguments allow you to perform least-squares deconvolution (`Method="least-squares"`), which returns more stable solutions compared to the default long-division deconvolution (`Method="long-division"`).

When you use the least-squares method to deconvolve a signal  $y$  with respect to an impulse response  $h$ , `deconv` returns the signal  $x$  that minimizes the norm of the residual signal (or remainder)  $r = y - \text{conv}(x, h)$ . That is,  $x$  is the solution that minimizes  $\text{norm}(r)$ . You can also specify the Tikhonov regularization factor  $\alpha$  to return a solution  $x$  that minimizes  $\text{norm}(r)^2 + \text{norm}(\alpha x)^2$  for ill-conditioned problems.

## **svdappend Function: Calculate revised singular value decompositions**

Starting with an existing singular value decomposition (SVD) of some data, you can use `svdappend` to revise the existing SVD after new rows or columns of data are added.

## **expmv Function: Calculate matrix exponential multiplied by vector**

Calculate the product of a matrix exponential and a vector without explicitly forming the matrix exponential by using the `expmv` function.

The `expmv` function is faster and more efficient than `expm` when computing the product of a matrix exponential of a sparse matrix and a vector. The `expmv` function also uses an efficient algorithm to compute the exponential integrators of ordinary differential equations, such as  $e^{tA}\vec{X}_0$ , where  $t$  is a time vector with a fixed time step,  $A$  is a square matrix, and  $\vec{X}_0$  is a column vector.

## **expm Function: Improved algorithm for single-precision matrices**

The `expm` function has an improved algorithm for input matrices that are single precision. The new algorithm can use fewer terms in the Padé approximation of some matrix exponentials.

## **scatteredInterpolant Object: Use multivalued interpolation to interpolate multiple data sets simultaneously**

The `scatteredInterpolant` object can now interpolate multiple data sets at the same query points. Specify the `Values` property as a matrix, where the number of rows is the same as the number of sample points and each column in `Values` represents the values of a different function at

the sample points. For example, if the sample points are column vectors with 10 elements, you can specify `Values` as a 10-by-4 matrix to interpolate using four different sets of values.



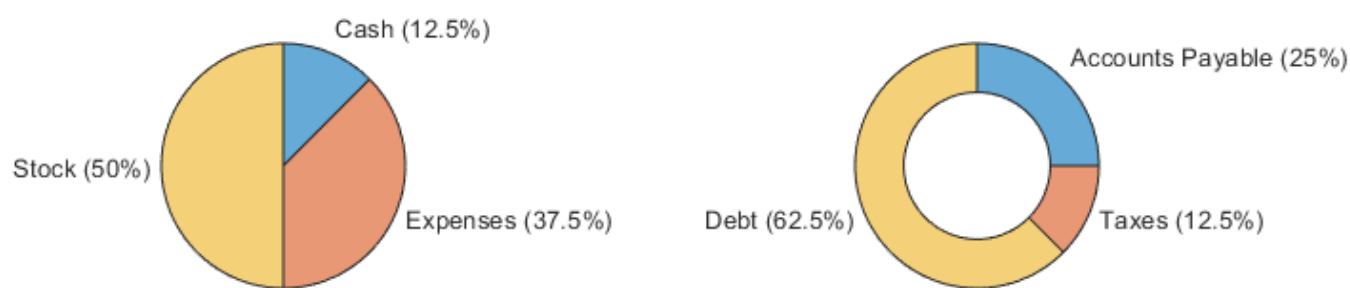
## Graphics

### piechart and donutchart Functions: Create configurable, interactive pie and donut charts

Use the `piechart` and `donutchart` functions to create charts with more configuration options and interactivity than pie charts had in previous releases.

Some of the improvements include:

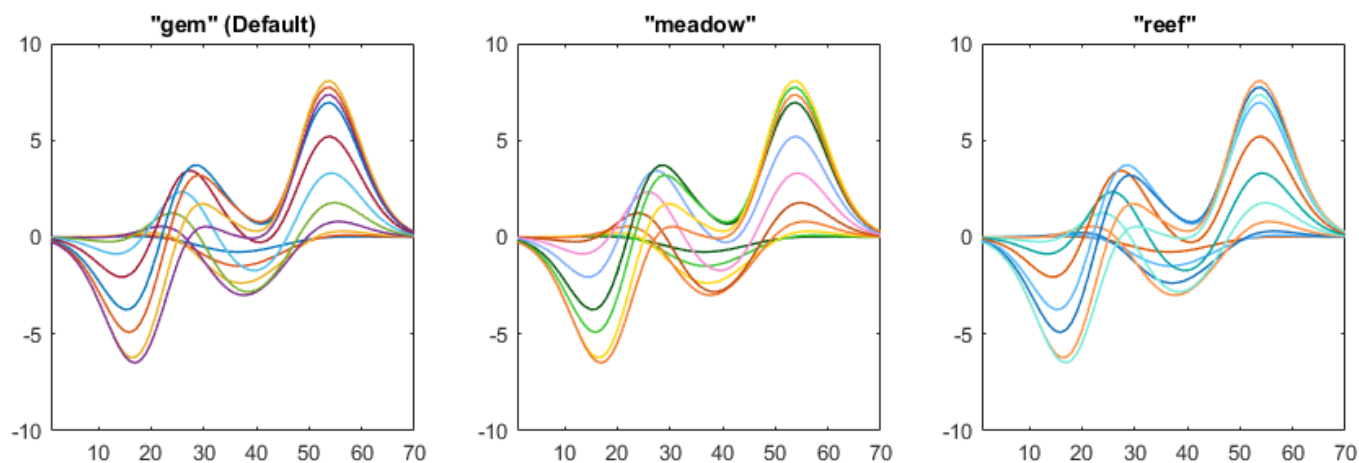
- More options for formatting slice labels, slice placement, and colors
- Data tips that appear when you move the cursor over the slices



### Plot Colors: Choose from nine color palettes

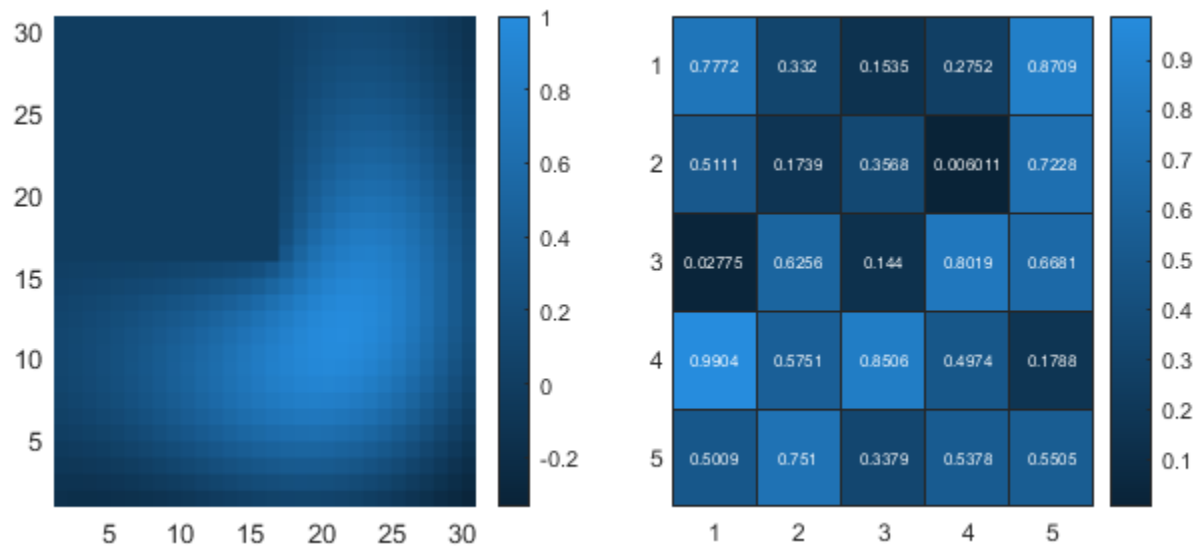
Plot multiple data series together using one of nine different color palettes to differentiate the individual data series. You can change the palette by passing the palette's name to the `colororder` function. The available names are "gem", "gem12", "glow", "glow12", "sail", "reef", "meadow", "dye", and "earth". The "gem" palette is the default for most plots.

You can also use the `orderedcolors` function to get the RGB triplets for any of the palettes.



## abyss Function: Apply blue-to-black colormap to charts and images

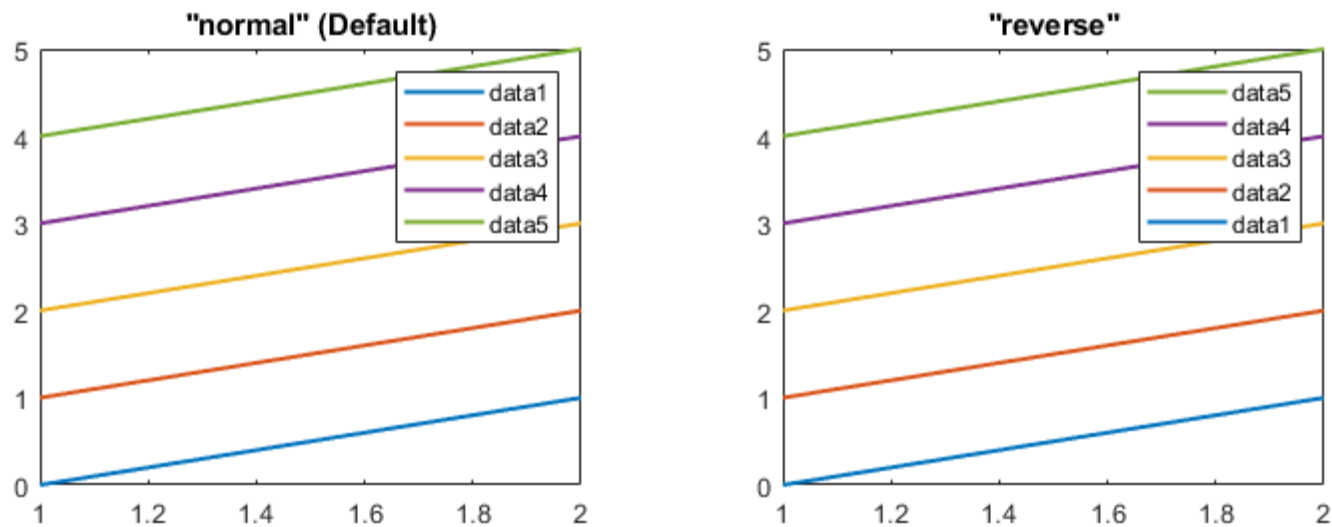
Use the `abyss` function to get a blue-to-black colormap for coloring your charts. Like for all predefined colormaps, you can optionally specify the number of colors for the `abyss` colormap.



## Legend: Control the order of legend entries

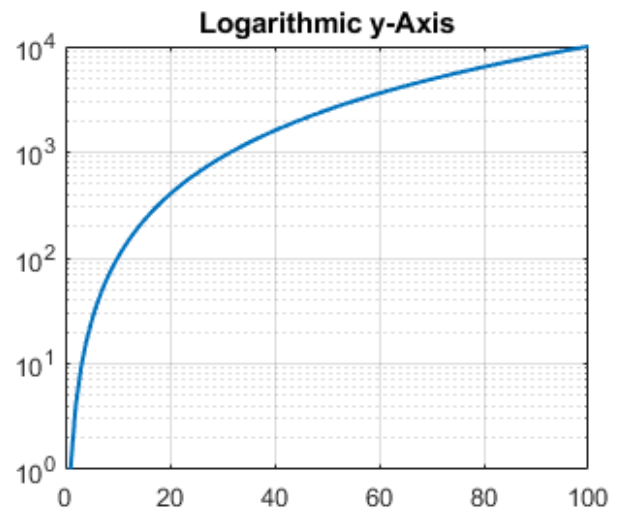
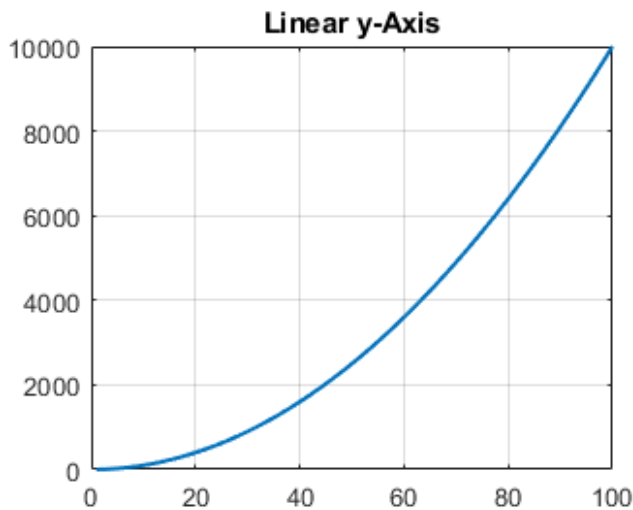
Control the order of legend entries by setting the `Direction` property of the legend to `"normal"` or `"reverse"`. In most cases, the default direction is `"normal"`.

Legends for stacked bar charts and area charts have a reverse order by default so the legend entries match the stacking order of the chart. For more information, see [Legend order is reversed for stacked bar charts and area charts](#).



## **xscale, yscale, and zscale Functions: Quickly change between logarithmic and linear axes scales after plotting**

View any dimension of a plot on a logarithmic or linear scale by calling the `xscale`, `yscale`, or `zscale` function after plotting. To switch between the different scales, call any of these functions with "linear" or "log" as an input argument. For example, `yscale("log")` changes the scale of the y-axis to be logarithmic.

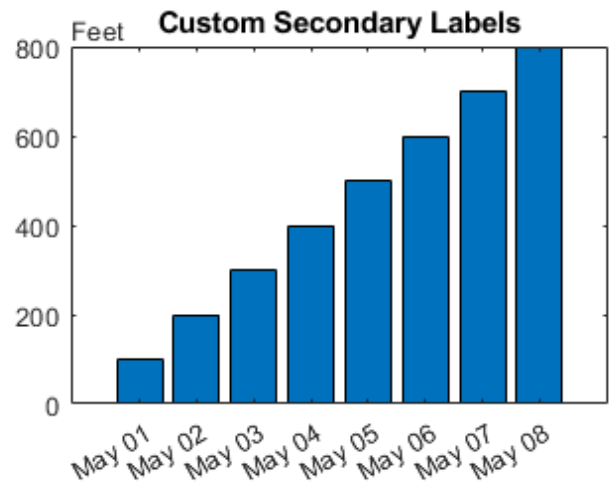
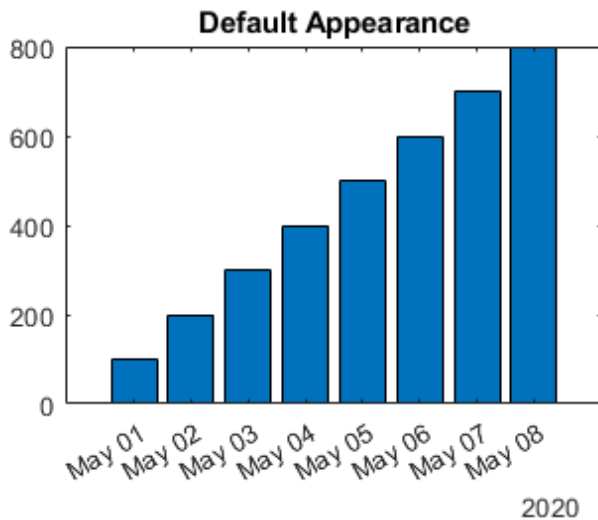


## **xsecondarylabel, ysecondarylabel, and zsecondarylabel Functions: Create, delete, or modify secondary axis labels**

Create, delete, or modify secondary axis labels by calling the `xsecondarylabel`, `ysecondarylabel`, and `zsecondarylabel` functions. Secondary labels are text labels that appear at the edge of the axes and provide additional information about the data. Often, they provide information about the units or scale of the data.

For example, create a bar chart with datetime values along the x-axis and feet along the y-axis. By default, the chart has a secondary x-axis label of "2020". Call the `xsecondarylabel` function with an empty string to delete the "2020" label. Then add a secondary label of "Feet" to the y-axis.

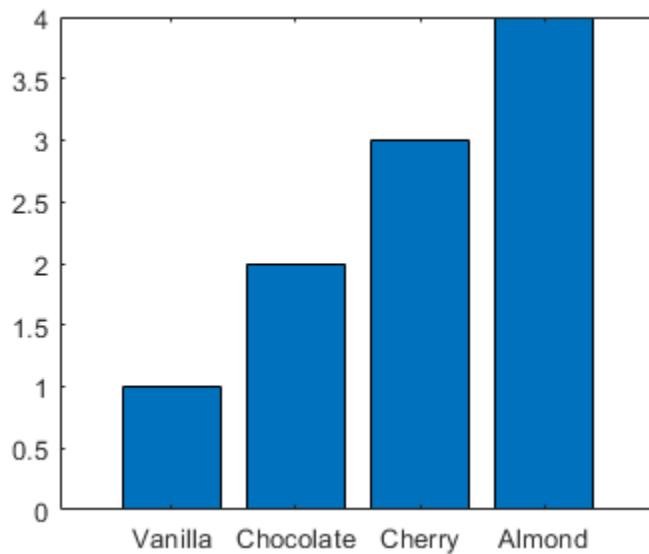
```
x = datetime(2020,5,1:8);
y = 100:100:800;
bar(x,y)
xsecondarylabel("");
ysecondarylabel("Feet")
```



## Bar Charts: Specify bar labels as string vectors

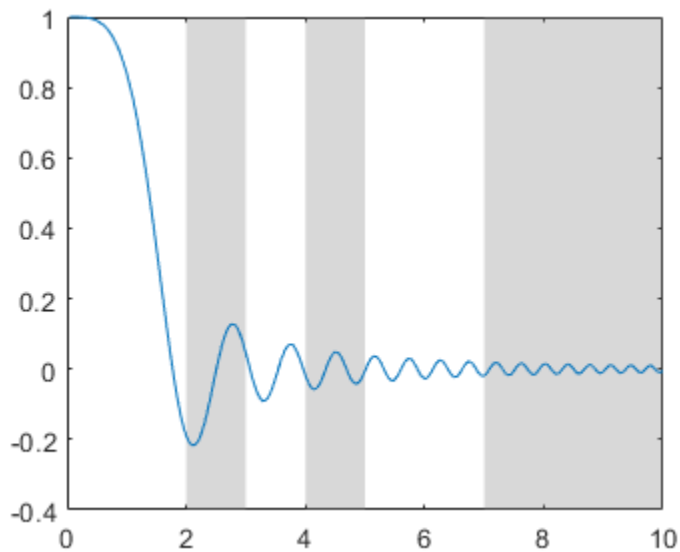
When you create bar charts using the `bar` and `barh` functions, you can specify the bar labels as string vectors. The bar tick labels appear in the order you specify them.

```
bar(["Vanilla","Chocolate","Cherry","Almond"],[1 2 3 4])
```



## xregion and yregion Functions: Create unbounded regions and specify multiple regions as matrix

Create unbounded filled regions by passing `Inf` or `-Inf` to the `xregion` and `yregion` functions. You can also create multiple regions by specifying one matrix input argument as an alternative to specifying two vectors of coordinates. For  $n$  regions, the matrix must be 2-by- $n$  or  $n$ -by-2 and contain the lower and upper bounds for all the regions.

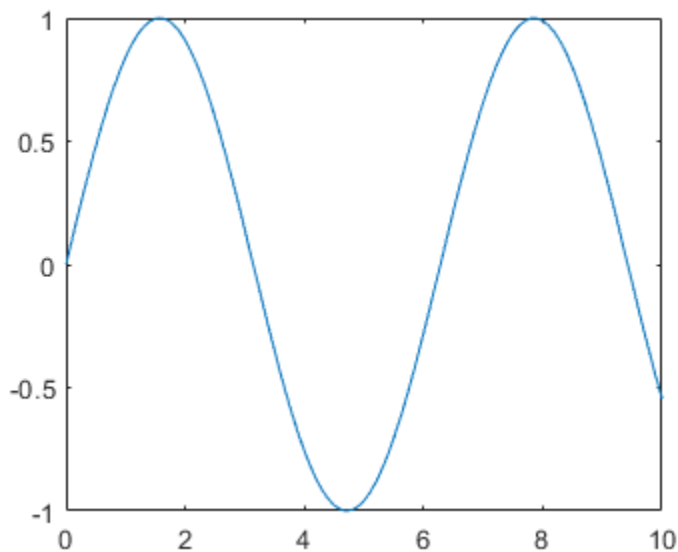


### Text Objects: Control whether text labels are included in axes limits

Create text objects with the anchor point positions included in the axes limits calculation. To include an anchor point in the calculation, set the `AffectAutoLimits` property of the text object to "on".

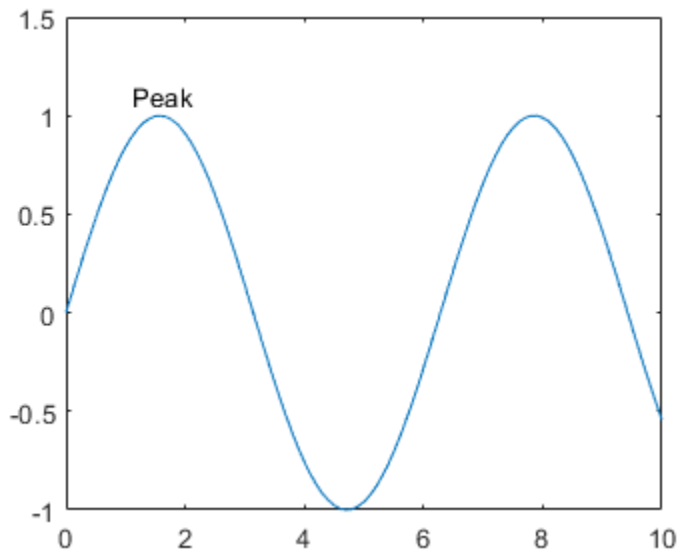
For example, create a line plot.

```
x = 0:0.1:10;  
y = sin(x);  
plot(x,y)
```



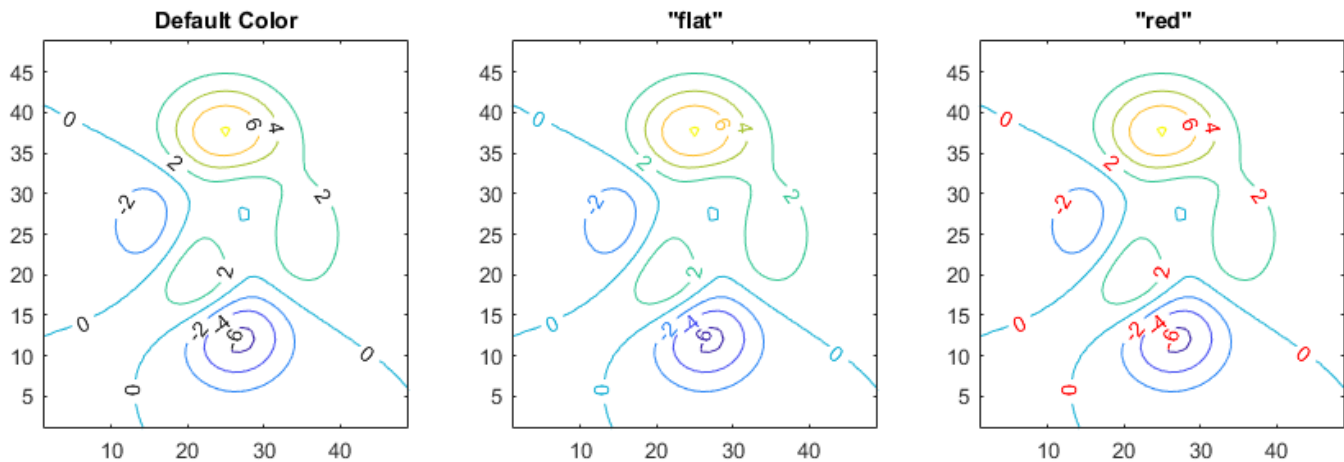
Create a text object outside of the current y-axis limits. Set the `AffectAutoLimits` property to "on" so that the axes limits adjust to include the anchor point of the text.

```
text(1.1,1.1,"Peak",AffectAutoLimits="on")
```



## Contour Plots: Customize colors of contour labels

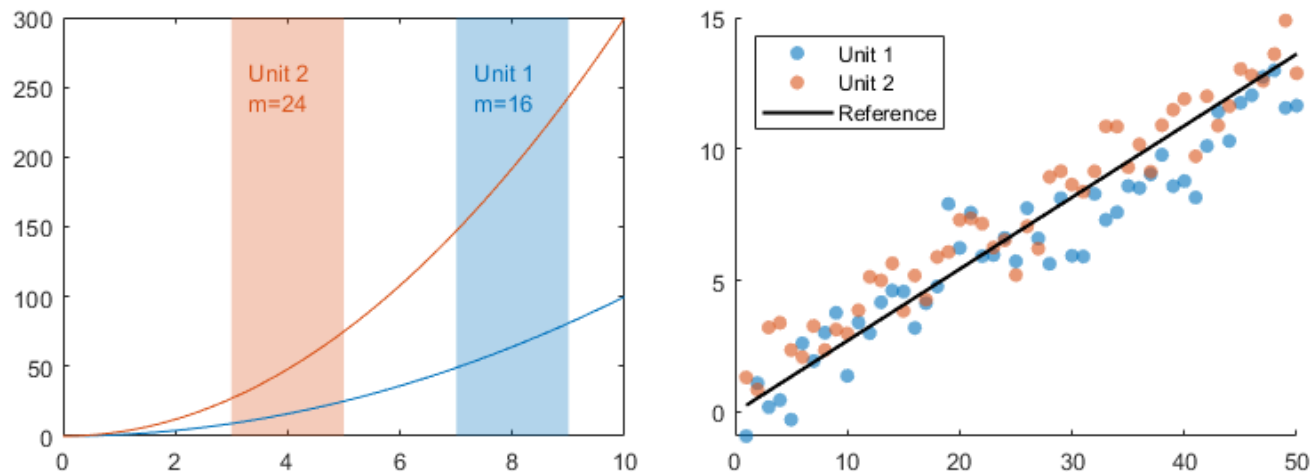
Set the `LabelColor` property of a `Contour` object to display labels that match the colors of the contour lines, or specify one color for all the labels.



## Colors and Line Styles: Control automatic color and line style selection for additional chart objects

Now you can match the colors and line styles of more objects in the axes by setting the `SeriesIndex` property of the objects to the same value. The `SeriesIndex` property is available for `Text`, `ConstantLine`, `ConstantRegion`, `Rectangle`, `Patch`, and `AnimatedLine` objects, and lines created by the `line`, `streamline`, and `streamslice` functions.

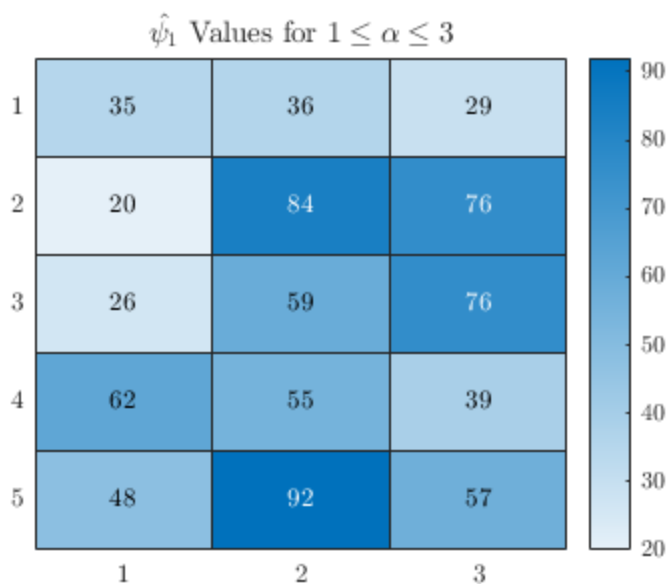
Also, the `SeriesIndex` property has a new option, "none", which enables you to opt out of automatic selection for certain objects, such as reference lines.



With the new `SeriesIndex` property support, lines created with the `line`, `streamline`, and `streamslice` functions have a different default color. For more information, see [Default color is different for lines created with the `line` function](#) and [Default color is different for plots created with the `streamslice` and `streamline` functions](#).

## Heatmap Charts: Control text interpreter for text labels in heatmap charts

Create text, such as titles and axis labels, for heatmap charts with TeX markup, LaTeX markup, or no markup by setting the `Interpreter` property of the chart.

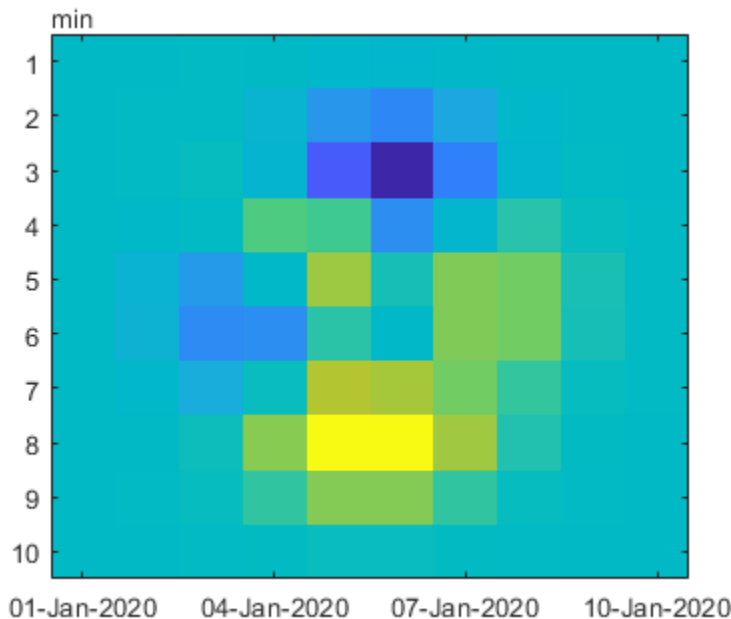


## Images: Display images with datetime, duration, or categorical pixel coordinates

You can now display images using datetime, duration, or categorical coordinate values with the `image` and `imagesc` functions. Previously, only numeric and logical coordinate values were supported.

For example, display an image with datetime values along the x-axis and duration values along the y-axis.

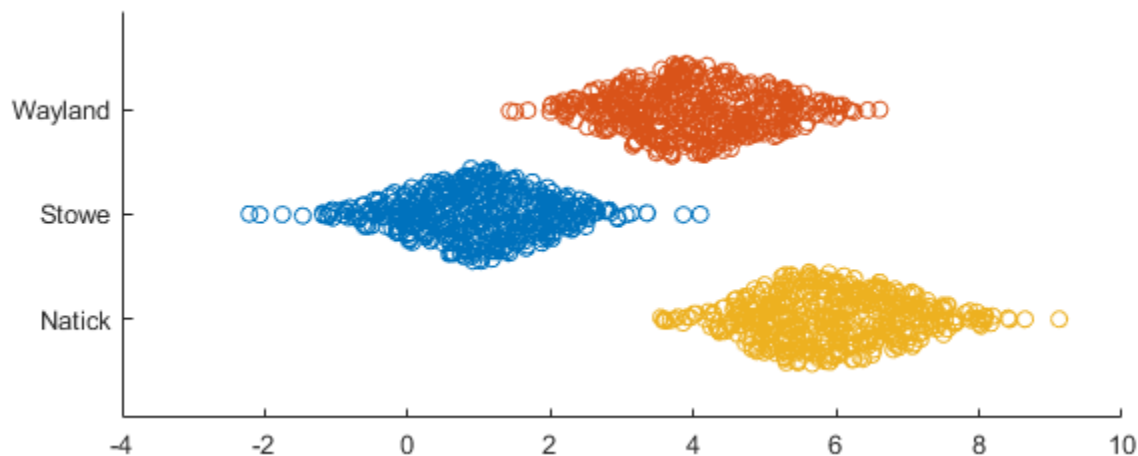
```
x = datetime(2020,1,[1 10]);  
y = minutes([1 10]);  
C = peaks(10);  
imagesc(x,y,C)
```



## swarmchart Function: Create horizontal swarm charts

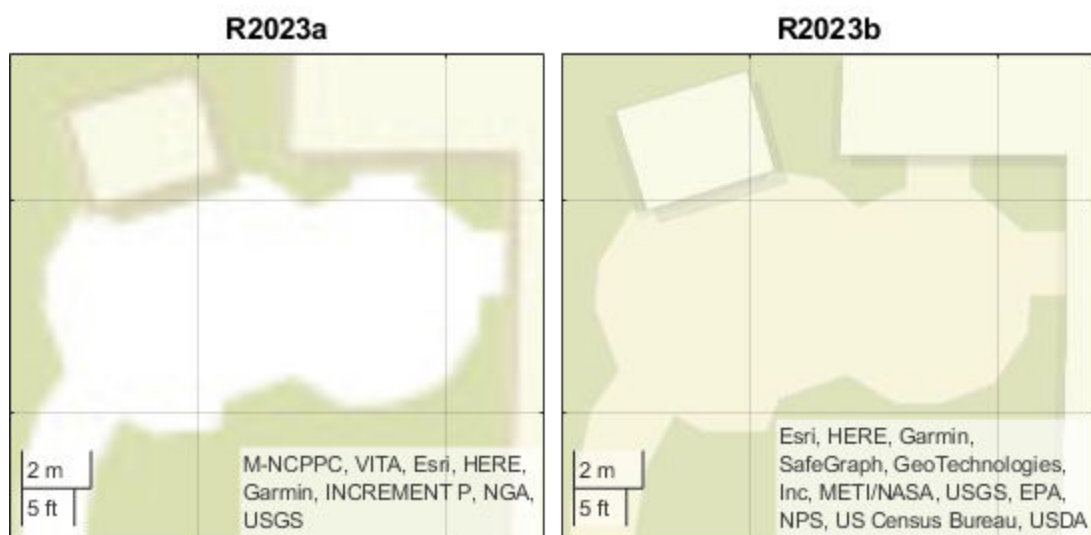
Create a horizontal swarm chart by setting the `YJitter` property when you call the `swarmchart` function. When you specify the `YJitter` property without specifying the `XJitter` property, MATLAB sets the `XJitter` property to "none", and the resulting distributions in the chart are horizontal.



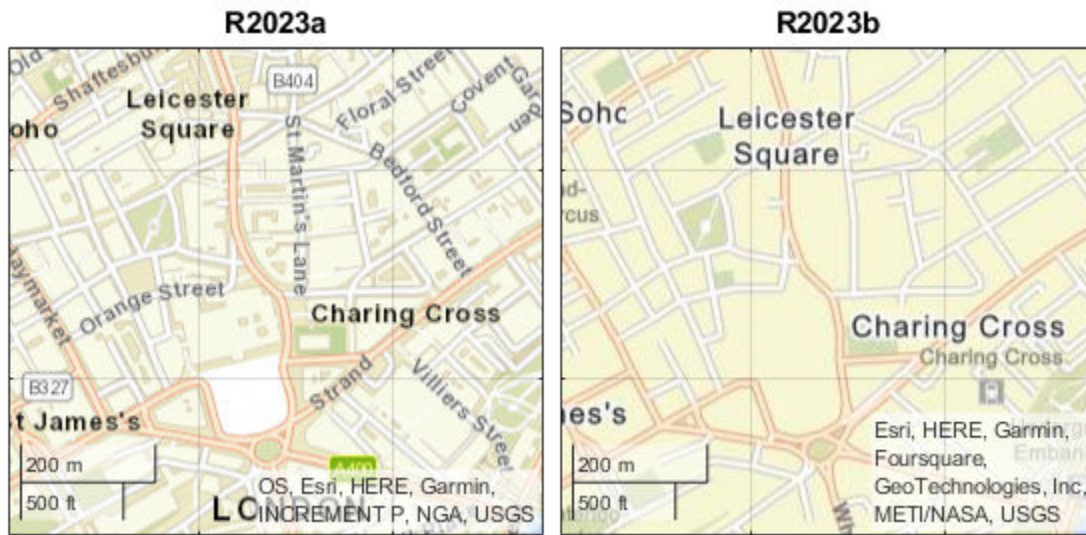


## Geographic Plots: Some basemaps have improved appearance at high zoom levels

The "streets-light", "streets-dark", "streets", and "topographic" basemaps hosted by Esri, which are used by geographic axes objects and other objects with a Basemap property, have an improved visual appearance at high zoom levels. For example, this image compares a basemap at zoom level 21 in R2023a with the same basemap and zoom level in R2023b.



The basemaps can also have different appearances at other zoom levels. For example, this image compares a basemap at zoom level 15 in R2023a with the same basemap and zoom level in R2023b.



For more information about changing the basemap of geographic axes, see [geobasemap](#).

The basemaps hosted by Esri update periodically. As a result, you might see differences in your visualizations over time.

## Figure Code in MATLAB Online: Generate code from Camera tab

For a 3-D figure in MATLAB Online, when you change the camera view using the **Camera** tab, MATLAB generates code for the camera motion. On the **Tools** tab of the figure window, click **Camera Tools** to open the **Camera** tab. Use the controls in the **Camera** tab to interactively change the camera view. Then, to view or copy the generated code, select **Show Code** in the **File** section of the **Figure** tab.

Previously, MATLAB did not generate code when changing the camera view using the **Camera** tab.

## Figure Code in MATLAB Online: Generate code from Property Inspector

For a figure in MATLAB Online, when you change the value of a graphics object property using the Property Inspector, MATLAB generates code to set the property value. On the **Format** tab of the figure window, click **Inspect** to open the Property Inspector. Use the controls in the inspector to set the value of a property. Then, to view or copy the generated code, select **Show Code** in the **File** section of the **Figure** tab.

MATLAB generates code only for properties that are specified as a 1-D array. Previously, MATLAB did not generate code when setting a property value using the Property Inspector.

## ⚠️ Functionality being removed or changed

### GraphicsSmoothing and FontSmoothing properties will be removed

*Still runs*

The `GraphicsSmoothing` property of figures and the `FontSmoothing` property of axes, rulers, geographic scales, and text objects will be removed in a future release. All graphics will be smooth and all text will use antialiasing when these properties are removed.

**spinmap function will be removed***Still runs*

The `spinmap` function will be removed in a future release. There is no replacement for this function.

**Some printing and exporting workflows will no longer support UI components***Warns*

These functions and menu items will no longer support printing or exporting UI components in a future release:

- The `print` function
- The `saveas` function
- The **File > Save As** menu item on the figure
- The **File > Export Setup** menu item on the figure
- The **Edit > Copy Options** menu item on the figure

To export a figure containing UI components, call the `exportapp` function. For example, create a simple app containing two buttons and a slider. Export the contents of the figure as a PDF file by calling the `exportapp` function.

```
% Create figure with three UI components
f = uifigure;
button1 = uibutton(f,"Position",[150 300 100 50]);
button2 = uibutton(f,"Position",[300 300 100 50]);
slider1 = uislider(f,"Position",[150 250 250 3]);
```

```
% Export the contents of the figure
exportapp(f,"myapp.pdf")
```

Alternatively, call the `getframe` function to capture the contents of the figure. Then call the `imwrite` function to save the content. This time, save the content as a JPEG file.

```
F = getframe(f);
imwrite(F.cdata,"myapp.jpg");
```

**Exporting workflows will no longer support BMP, HDF, PBM, PCX, PGM, and PPM files***Warns*

These functions and menu items will no longer support BMP, HDF, PBM, PCX, PGM, and PPM files in a future release:

- The `print` function
- The `saveas` function
- The **File > Save As** menu item on the figure
- The **File > Export Setup** menu item on the figure

To export graphics using one of these formats, use the `imwrite` function instead. For example, create a line plot, and capture the contents of the current figure using the `getframe` function. Then save the content as a BMP file.

```
plot([0 3 2 4 1]);
F = getframe(gcf);
imwrite(F.cdata,"myplot.bmp");
```

**Some exporting workflows will no longer support PostScript (.ps) files***Warns*

These functions and menu items will no longer support creating full-page PostScript (.ps) files in a future release:

- The `print` function
- The `saveas` function
- The **File > Save As** menu item on the figure
- The **File > Export Setup** menu item on the figure

To export graphics files, call the `exportgraphics` function. Specify an .eps, .pdf, or .emf file extension, and set the `ContentType` name-value argument to "vector". This function captures content that is tightly cropped around plots, and it does not create full-page output. For example, create a plot and save the contents of the current figure as a PDF file containing vector graphics.

```
plot([0 3 2 4 1]);  
exportgraphics(gcf,"myplot.pdf",ContentType="vector")
```

Alternatively, call the `print` function and specify an .eps, .emf, or .svg file extension. For example, create a plot and save the contents of the current figure as an EPS file.

```
plot([0 3 2 4 1]);  
print("myplot.eps","-depsc")
```

**The `getframe` function will no longer capture the figure toolbar, menu bar, or borders***Warns*

In a future release, the `getframe` function will no longer support capturing the figure toolbar, menu bar, or the border around the figure using the `rect` argument. You will still be able to define a subsection of the figure to capture, but the toolbar, menu bar, and borders will not be included.

As an alternative, you can use the `exportapp` function to capture the toolbar and menu bar, but not the borders.

**`printopt` will be removed***Warns*

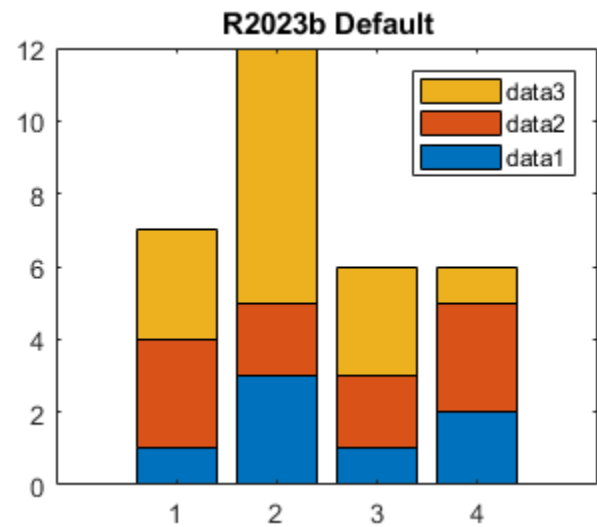
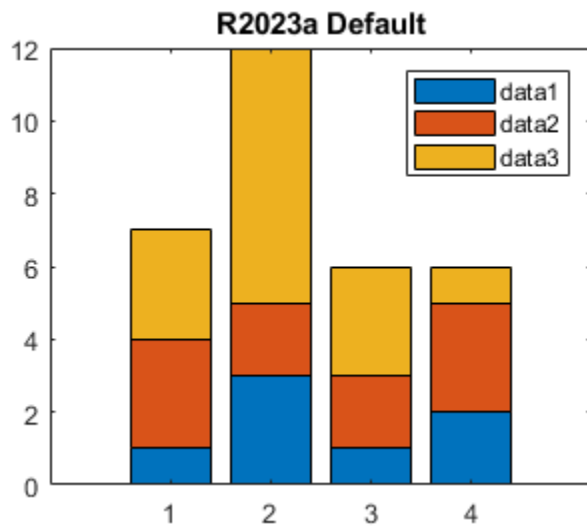
The `printopt` function will be removed in a future release. To configure printer defaults, use the options provided by your operating system.

**Legend order is reversed for stacked bar charts and area charts***Behavior change*

The default order of legend entries for stacked (vertical) bar charts and area charts is now reversed to match the stacking order of the chart. Previously, the legend entries were listed in the opposite order of stacked bars and area charts.

To preserve the order of previous releases, set the `Direction` property of the legend to "normal".

```
lgd = legend;  
lgd.Direction = "normal";
```



### Default color is different for lines created with the line function

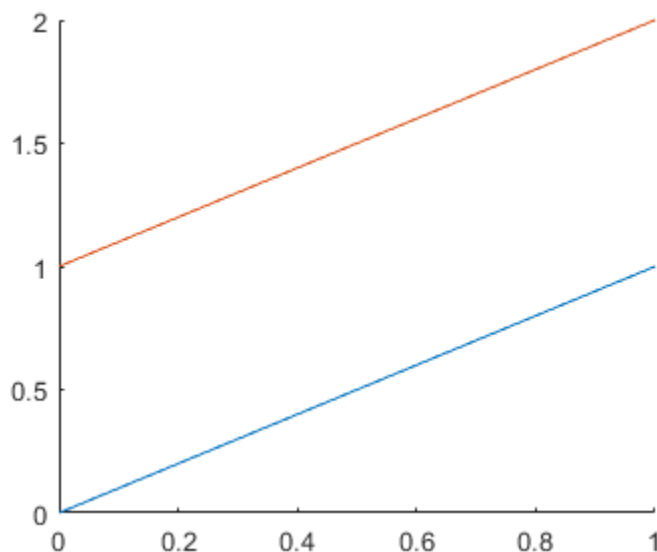
#### *Behavior change*

Now that the `SeriesIndex` property is available for lines created with the `line` function, the lines cycle through the same colors (and optional line styles) that most other plots do.

The default color change applies only to the lines you create when you specify the `x`, `y`, and optional `z` arguments. If you create lines with a syntax that uses name-value arguments only, the plots look the same as in previous releases.

For example, create two lines with `x` and `y` input arguments. In R2023b, the first line is blue and the second line is red-orange. Before R2023b, both lines were blue.

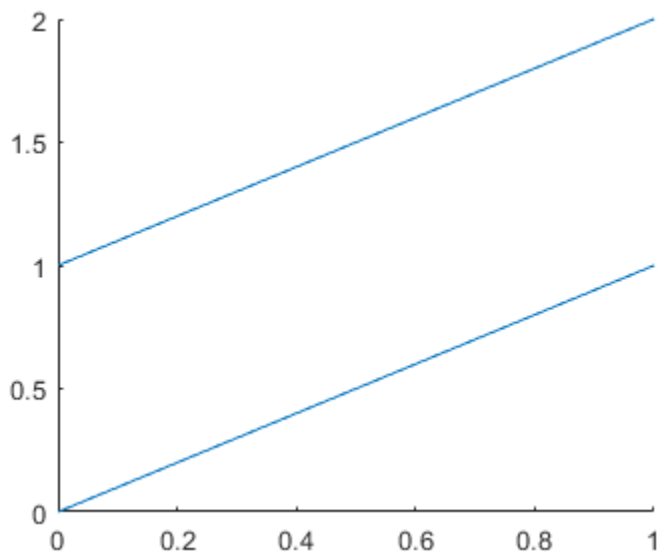
```
line1 = line([0 1],[0 1]);
line2 = line([0 1],[1 2]);
```



To preserve the behavior of previous releases, set the `SeriesIndex` property of the lines to 1. You can set the property using a name-value argument when you call the `line` function, or you can set the property of the `Line` object using dot notation later.

```
% Use a name-value argument
line1 = line([0 1],[0 1],SeriesIndex=1);

% Use dot notation
line2 = line([0 1],[1 2]);
line2.SeriesIndex = 1;
```

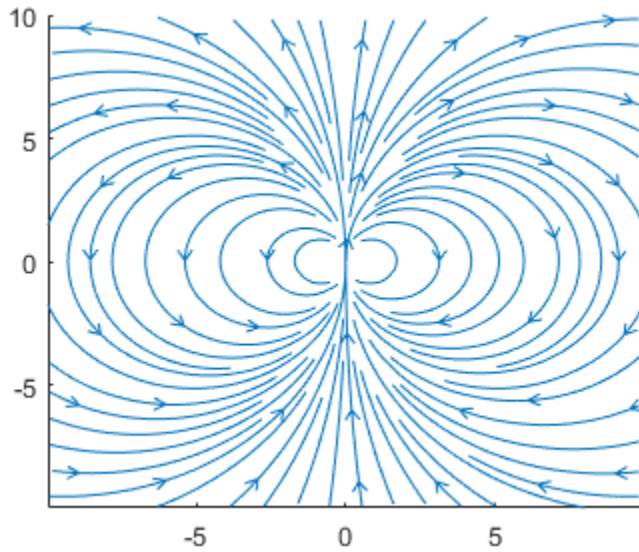


### Default color is different for plots created with the `streamslice` and `streamline` functions

#### *Behavior change*

When you create a `streamslice` or `streamline` plot, MATLAB automatically assigns colors and line styles the same way as for most other plots. For example, the first set of lines created with `streamslice` are now a soft blue color.

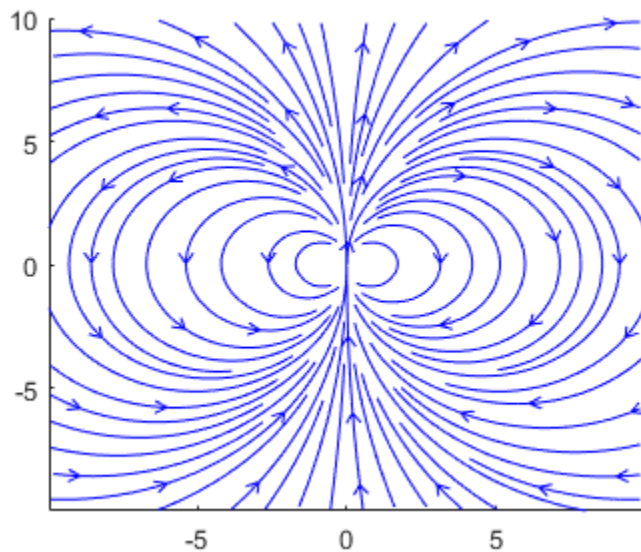
```
[x,y] = meshgrid(-10:10);
u = 2.*x.*y;
v = y.^2 - x.^2;
slicelines = streamslice(x,y,u,v);
axis tight
```



If you call either of the functions repeatedly, subsequent plots cycle through the same set of colors as other plots do. In previous releases, the lines were bright blue.

To preserve the appearance from previous releases, use the `set` function to set the `Color` property to `[0 0 1]`.

```
set(slicelines,Color=[0 0 1])
```



### **RotateStyle property of a rotate3d object will have no effect**

*Behavior change in future release*

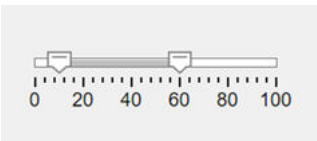
The `RotateStyle` property of a `rotate3d` object will have no effect in a future release. Plot-box outline rotation will no longer be supported, and rotation will be about the entire axes regardless of the value of the property.

# App Building

## uislider Function: Create slider to specify range of values in apps and on App Designer canvas

You can create a range slider using the `uislider` function by specifying the style as "range". Alternatively, in App Designer, you can drag a **Slider (Range)** component from the **Component Library** onto the canvas. The range slider has two thumbs that app users can use to adjust the range of values. For example, this code creates a range slider that specifies values from 10 to 60.

```
fig = uifigure;  
sld = uislider(fig, "range", Value=[10 60]);
```



## uicontextmenu and uimenu Functions: Access information about action used to open context menu

You can now more easily customize and share context menus in an app by using callback event data. The `ContextMenuOpeningFcn` and `MenuSelectedFcn` callbacks have additional event data when the callback is associated with a context menu on a UI component.

| Event Data Property    | Description                                                                                                               |
|------------------------|---------------------------------------------------------------------------------------------------------------------------|
| ContextObject          | Object that the app user right-clicked to open the context menu                                                           |
| InteractionInformation | Information about where in the component the app user right-clicked to open the context menu, such as the cell of a table |

For more information, see [ContextMenu Properties](#) and [Menu Properties](#).

## uispinner and uieditfield Functions: Create spinners and numeric edit fields with placeholder text

Create a spinner or numeric edit field that has an empty value by setting the `AllowEmpty` property to "on" and the `Value` property to []. The component appears with no default value. You can specify placeholder text to give app users a hint about the type of content the field accepts by using the `Placeholder` property.

For more information, see [Spinner Properties](#) and [NumericEditField Properties](#).



## uilibox and uidropdown Functions: Access index of component value in list of items

Access the index of the value in a list box or drop-down component by using the `ValueIndex` property. This property is useful when the component has information stored in both the `Items` and `ItemsData` properties and you need to:

- Query both the current item and its associated data.
- Specify a new value using the index of the value in the list of items.

For example, use the `ValueIndex` property to display both the current item and its associated data.

```
fig = uifigure;
dd = uidropdown(fig, ...
    Items=["Apple","Banana","Cherry"], ...
    ItemsData=[0.5 0.2 1.3]);
idx = dd.ValueIndex;
```

```
disp(dd.Items(idx) + ": " + dd.Value)
```

```
Apple: 0.5
```

For more information, see [ListBox Properties](#) and [DropDown Properties](#).

## uitable Function: Query selected table data in current table display

When you create a table, use the `DisplaySelection` property of the table to query the cells that are selected in the current table display. This property is useful when an app user has sorted or rearranged columns in the table and you want to know which cells are selected based on the app user's view of the data.

For more information, see [Table Properties](#).

## ▲uicontextmenu Function: Context menus on trees appear when you right-click anywhere in the tree

When you create a context menu and assign it to a `Tree` object, the context menu appears when you right-click anywhere in the tree, including the area with tree nodes. You can share a single context menu across all nodes in a tree by setting the context menu on the tree instead of each individual tree node.

If you set a context menu on an individual `TreeNode` object, the tree node context menu still appears when you right-click the node, even if the `Tree` object has an assigned context menu.

## ▲Compatibility Considerations

Previously, context menus on trees appeared only when you right-clicked on the portion of the tree below the tree nodes. To retain this behavior, create a `ContextMenuOpeningFcn` callback that shows the context menu only when the right-click occurs on a region without a tree node.

```
fig = uifigure;
cm = uicontextmenu(fig,ContextMenuOpeningFcn=@hideMenu);
ml = uimenu(cm,Text="Menu Item");
```

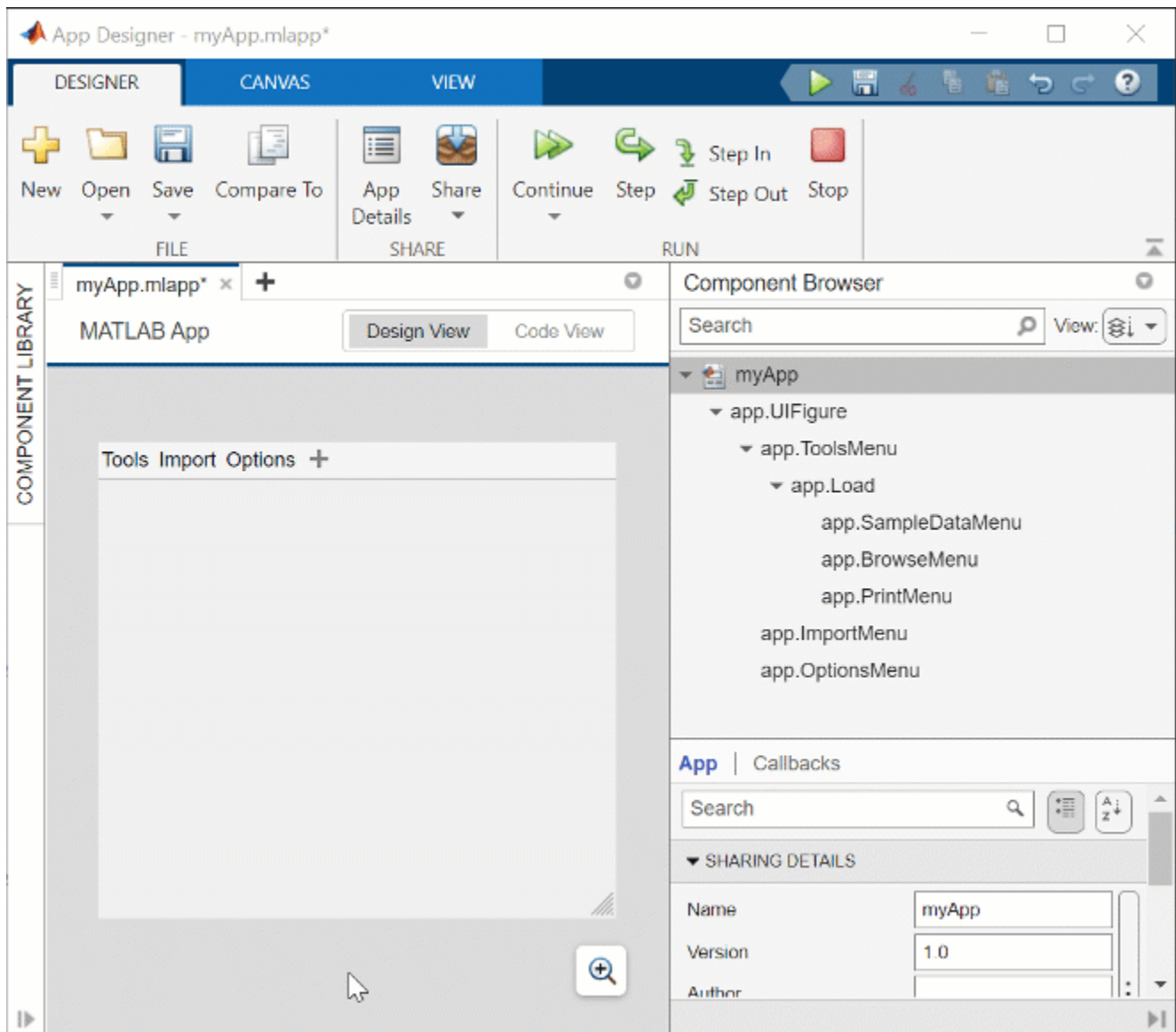
```
t = uitree(fig,ContextMenu=cm);  
n1 = uitreenode(t);  
  
function hideMenu(src,event)  
if isprop(event.InteractionInformation,"Node")  
    src.Visible = "off";  
end  
end
```

## **appmigration.migrateGUIDEApp Function: Migrating multiple apps from GUIDE to App Designer updates app references**

When you use the `appmigration.migrateGUIDEApp` function to migrate multiple apps at once from GUIDE to App Designer, the function updates app references. For example, to migrate a main app that opens a secondary app from GUIDE to App Designer, first download the GUIDE to App Designer Migration Tool for MATLAB. Then, use the `appmigration.migrateGUIDEApp` function to migrate the main and secondary apps using a single command. The function replaces all references to the apps that you migrate together. Therefore, your migrated main app opens the migrated secondary app without any manual changes to your app code.

## **App Designer: Interactively rearrange tabs, menus, tree nodes, and toolbar tools**

Interactively rearrange tabs, menus, tree nodes, and toolbar tools between parents by dragging the component in the **Component Browser**. For example, interactively move and change the parents of menus and menu items.



## App Designer: Write unit tests for custom UI components

You can write unit tests for custom UI components created in App Designer by using the app testing framework. To enable the unit tests to access the underlying UI components in your custom component, with the custom component open in App Designer, select the component node in the **Component Browser** and select **Give Test Cases Access** in the **Testing** section. Then, reference the underlying UI components in your unit tests to test the component behavior.

Running tests for custom UI components interactively using `matlab.unittest.TestCase.forInteractiveUse` is not supported.

For more information, see [Write Tests for Custom UI Component](#).

## App Designer: View changes to HTML UI component source while working in Design View

While designing an app that contains an HTML UI component, you can now more easily see updates made to the HTML source. To show the most up-to-date content in Design View when the `HTMLSource` property of the component is a path to an HTML source file, click **Refresh Source** while pointing to the HTML UI component in the canvas.

For more information about using HTML UI components, see [Create HTML Content in Apps](#).

## Comparison Tool: Save app comparison results as PDF or DOCX reports programmatically and interactively

You can now use the Comparison Tool to programmatically and interactively publish App Designer app comparison results as PDF or DOCX reports. For more information, see [visdiff](#).

## Axes Interactions: Interpolate data tip location

For axes in apps created in App Designer or using the `uifigure` function, control the method for placing data tips by using the `InteractionOptions` property of the axes. Create data tips at the closest interpolated location on the plot to the cursor location by setting the `DatatipsPlacementMethod` property of the `InteractionOptions` object to "interpolate". By default, data tips are created at the closest data point to the cursor location.

## Axes Interactions: Indicate azimuth and elevation angles

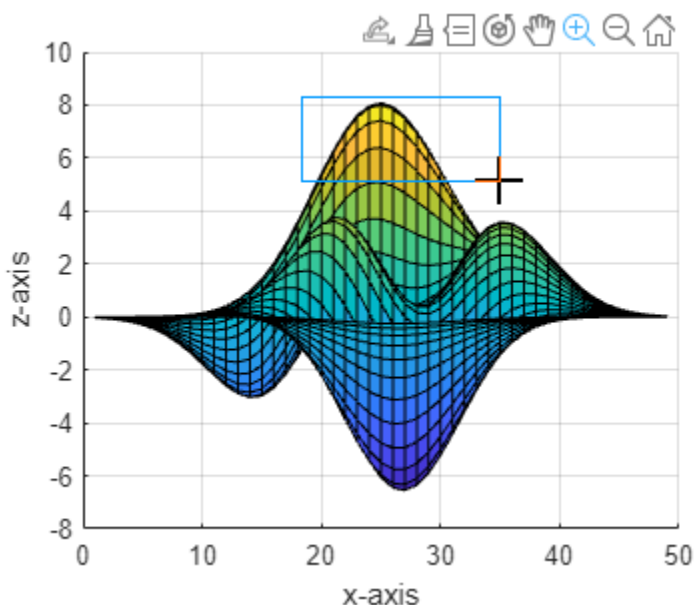
For axes in apps created in App Designer or using the `uifigure` function, indicate the azimuth and elevation angles during interactive rotation by using the `InteractionOptions` property of the axes. The azimuth and elevation angles define the camera line of sight and are displayed when the `RotateIndicator` property of the `InteractionOptions` object is set to "azimuthelevation". By default, the axes do not display the azimuth and elevation angles.

## Axes Interactions: Zoom into rectangular region of 2-D view of 3-D chart

For a 2-D view of a 3-D chart in apps created in App Designer or using the `uifigure` function, you can zoom into a rectangular region. To enable region-zooming, click the Zoom In or Zoom Out button in the axes toolbar, or set the `Interactions` property of the axes to the `regionZoomInteraction` object.

For example, create a 3-D surface plot and set the view to the x- and z-axes. Click the Zoom In button and then drag to select the region of interest.

```
f = uifigure;  
a = axes(f);  
surf(a,peaks);  
view(a,[0 0]);
```



Previously, for 3-D charts, the region-zoom interaction was supported only for the xy view.

## App Testing Framework: Automatically scroll to components outside viewable area

If you programmatically perform a gesture on a UI component that is not in the viewable area of an app, the app testing framework automatically scrolls to the specified component before performing the gesture. For the framework to bring the component into view, the component must be supported by the `scroll` function and all its ancestors must have their `Scrollable` property set to "on" or `true`. For example, create a scrollable figure with a state button that is outside the viewable area of the figure, and then press the button.

```
fig = uifigure(Scrollable="on");
b = uibutton(fig, "state", Position=[fig.Position(3)+100 100 100 22]);

testCase = matlab.uitest.TestCase.forInteractiveUse;
testCase.press(b)
```

## Web Apps and Standalone Applications: Data brushing supported in graphics

Graphics created in web apps and standalone applications support data brushing. Use data brushing in these applications just as you would in MATLAB figures.

## Performance

### Language and Programming: Improved performance for reading and writing class property values

Reading and writing class property values shows improved performance. The largest gains in performance are for accessing properties that do not use validation or get and set methods.

For example, the `movingAverage` method of `StorageClass` reads several elements from the `data` property and writes to the `average` property.

```
classdef StorageClass
    properties
        data
        average
        N
    end
    methods
        function obj = StorageClass(dataIn)
            obj.data = dataIn;
            obj.N = numel(dataIn);
            obj.average = zeros(1, obj.N-2);
        end
        function obj = movingAverage(obj)
            for ix = 1:(obj.N-2)
                obj.average(ix) = (obj.data(ix) +...
                    obj.data(ix+1) + obj.data(ix+2))/3;
            end
        end
    end
end
```

This code is about 17.5x faster than in the previous release.

```
s = StorageClass(1:1e6);
timeit(@()s.movingAverage)
```

The approximate execution times are:

**R2023a:** 0.497 s

**R2023b:** 0.0284 s

The code was timed on a Windows 10, Intel Xeon CPU E5-1650 v3 @ 3.50 GHz test system.

### dde23 Function: Improved performance integrating high-dimensional systems

The `dde23` function shows improved performance solving high-dimensional systems of DDEs over long integration intervals. The improvement gets better as the number of coupled DDEs in the system increases.

For example, this code solves a system of 500 coupled DDEs over the time interval [0 100]. The code is about 24x faster than in the previous release.

```
function timingdde23
rng default
N = 500;
C = rand(N);
lag = 1;
dderhs = @(t,y,ylag) -y-C*ylag;
sol = dde23(dderhs,lag,10+rand(1,N),[0 100]);
end
```

The approximate execution times are:

**R2023a:** 5.6 s

**R2023b:** 0.23 s

The code was timed on a Windows 10, Intel Xeon CPU W-2133 @ 3.60 GHz test system using the `timeit` function.

```
timeit(@timingdde23)
```

## nufftn Function: Improved performance with nonuniform sample points or query points

The `nufftn` function shows improved performance when operating on either nonuniformly spaced sample points or nonuniformly spaced query points.

For example, this code constructs a 262,144-by-3 matrix of nonuniform sample points `t` and calculates the nonuniform discrete Fourier transform along each dimension of a 64-by-64-by-64 array. The code is about 3.3x faster than in the previous release.

```
function timingSamplePoints
rng default
t = rand(64^3,3);
X = rand(64,64,64);
tic
    Y = nufftn(X,t);
toc
end
```

The approximate execution times are:

**R2023a:** 0.40 s

**R2023b:** 0.12 s

The code was timed on a Windows 10, Intel Xeon CPU E5-1650 v4 @ 3.60 GHz test system by calling the `timingSamplePoints` function.

As another example, this code constructs a 262,144-by-3 matrix of nonuniform query points `f` and calculates the nonuniform discrete Fourier transform along each dimension of a 64-by-64-by-64 array. The code is about 1.6x faster than in the previous release.

```
function timingQueryPoints
rng default
f = rand(64^3,3);
X = rand(64,64,64);
```

```
tic
    Y = nufftn(X,[],f);
toc
end
```

The approximate execution times are:

**R2023a:** 0.40 s

**R2023b:** 0.25 s

The code was timed on a Windows 10, Intel Xeon CPU E5-1650 v4 @ 3.60 GHz test system by calling the `timingQueryPoints` function.

## **fillmissing Function: Improved performance when filling entries with previous or next numeric value**

The `fillmissing` function shows improved performance when you specify the fill method as "previous" or "next" for numeric data.

For example, this code fills the NaN values in a 1000-element numeric vector with the previous nonmissing value. The code is about 6.9x faster than in the previous release.

```
function timingFillmissing
A = rand(1000,1);
idx = randperm(1000,50);
A(idx) = NaN;
for i = 1:2e4
    F = fillmissing(A,"previous");
end
end
```

The approximate execution times are:

**R2023a:** 1.59 s

**R2023b:** 0.23 s

The code was timed on a Windows 10, Intel Xeon CPU E5-1650 v4 @ 3.60 GHz test system using the `timeit` function.

```
timeit(@timingFillmissing)
```

## **ismember Function: Improved performance with small unsorted set array**

The `ismember` function and other set operations that call `ismember` show improved performance when the set array is unsorted and the product of the numbers of elements in the query array and the set array is less than 256.

For example, this code returns an array containing logical 1 (`true`) where the data in a 6-element query array is found in a 25-element set array. The code is about 2.9x faster than in the previous release.



```
function timingIsmember
a = [1 2 3 4 5 6];
b = magic(5);
for i = 1:1e5
    Lia = ismember(a,b);
end
end
```

The approximate execution times are:

**R2023a:** 0.70 s

**R2023b:** 0.24 s

The code was timed on a Windows 10, Intel Xeon CPU E5-1650 v4 @ 3.60 GHz test system using the `timeit` function.

```
timeit(@timingIsmember)
```

## islocalmax and islocalmin Functions: Improved performance when finding local minima and maxima

The `islocalmax` and `islocalmin` functions show improved performance. The improvement is most significant for data with large flat regions at the local maxima or minima, respectively.

For example, this code finds the local maxima of data that contains consecutive maxima values. The code is about 2.8x faster than in the previous release.

```
function timingIslocalmax
x = repelem([0.5 0 1 0 1 0 1 0 0.5],10000);
for i = 1:3e2
    TF = islocalmax(x);
end
end
```

The approximate execution times are:

**R2023a:** 0.91 s

**R2023b:** 0.32 s

The code was timed on a Windows 10, Intel Xeon CPU E5-1650 v4 @ 3.60 GHz test system using the `timeit` function.

```
timeit(@timingIslocalmax)
```

## Data Cleaner App: Improved rendering performance of Cleaning Parameters panel

In the Data Cleaner app, the **Cleaning Parameters** panel renders more quickly after selecting a cleaning method in R2023b than in R2023a. The delay before you can interact with the panel is reduced.

For example, if one table variable is selected in the Data Cleaner app and you select the **Clean Outlier Data** cleaning method, the **Cleaning Parameters** panel is ready about 1.14x faster in R2023b than in the previous release.

The approximate panel rendering times are:

**R2023a:** 2.4 s

**R2023b:** 2.1 s

The panel rendering was timed on a Windows 10, Intel Xeon CPU E5-1650 v4 @ 3.60 GHz test system by selecting the cleaning method in the Data Cleaner toolbar and measuring the time it takes for the **Cleaning Parameters** panel to be ready.

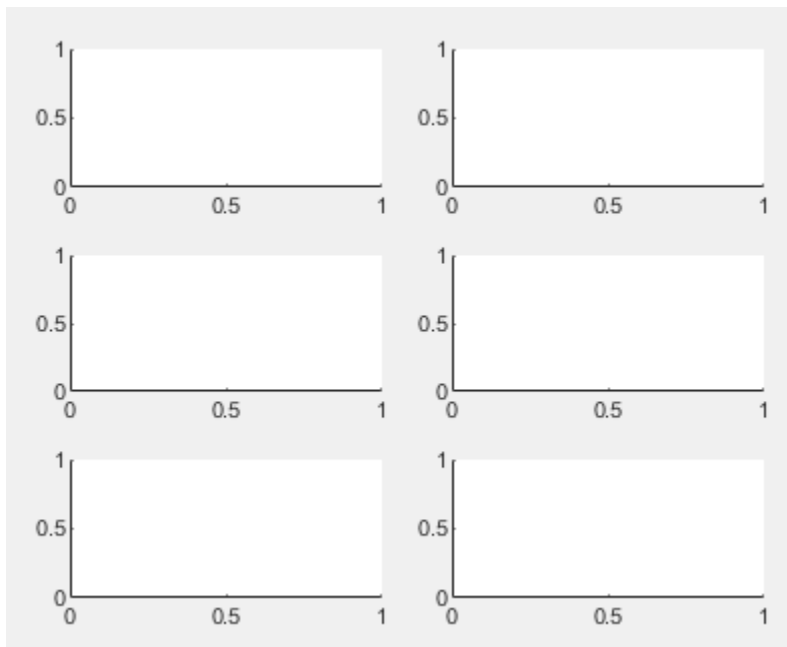
## Tiled Chart Layouts: Improved performance for flow layouts with spanned axes

Tiled chart layouts that have the "flow" tile arrangement and axes that span many tiles are more responsive when the tile arrangement changes. The tile arrangements change under these conditions:

- Adding new axes
- Resizing the figure
- Customizing the appearance of the axes by setting axes properties

For example, this code is about 1.4x faster than in the previous release.

```
function mylayout
    tiledlayout("flow")
    for i = 1:6
        nexttile([1000,1000]);
    end
end
```



The approximate execution times are:

**R2023a:** 0.0380 s

**R2023b:** 0.0276 s

The code was timed on a Windows 10, Intel Xeon CPU W-2133 @ 3.60 GHz test system using the `timeit` function.

```
timeit(@mylayout)
```

## Plots in Apps: Improved performance when scrolling to zoom

Zooming using a multi-touch trackpad or mouse shows improved performance in plots within apps and within figures created using the `uifigure` function. The two-finger scroll gesture and the zooming action are more synchronized in R2023b than in R2023a. The improvement results in a smoother experience when zooming.

For example, this code creates a figure with a plot of random data. On a Windows 10, Intel Xeon CPU E5-1650 v4 @ 3.60 GHz test system, when you zoom into the plot using the two-finger scroll gesture on a trackpad, the plot rendering is smoother and the zooming speed is more precise, especially at slower speeds, in R2023b than in the previous release.

```
f = uifigure;
ax = uiaxes(f);
plot(ax,magic(10))
```

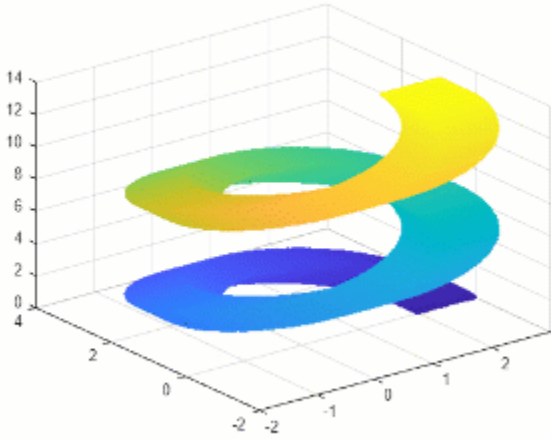
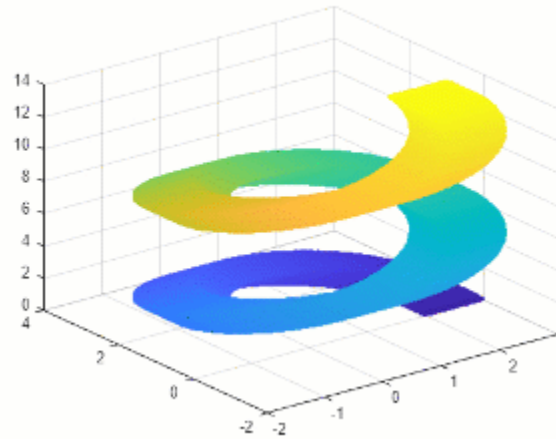
## Plots in Apps: Improved responsiveness for scatter plot interactions in apps

Interactions with scatter plots show improved responsiveness. To observe the improvement, the plot must be created in an app or in a figure created with the `uifigure` function, and the markers must either be points (".") or have a size less than 35. The improvement is most noticeable for plots with 1 million or more points.

For example, on a Windows 10, Intel Xeon CPU W-2133 @ 3.60 GHz test system, create a 3-D scatter plot with 7 million points in an app window, and then drag to rotate the axes. The rotation is smoother and responds more quickly to the drag gesture in R2023b than in the previous release.

```
function myscatter
f = uifigure;
ax = axes(f);
z = linspace(0,4*pi,7e6);
x = 2*cos(z) + rand(1,7e6);
y = 2*sin(z) + rand(1,7e6);
scatter3(ax,x,y,z,[],1:7e6, ".")
end
```

These animations start with the corner of the xy-plane at (-2, -2) in the center of the frame.

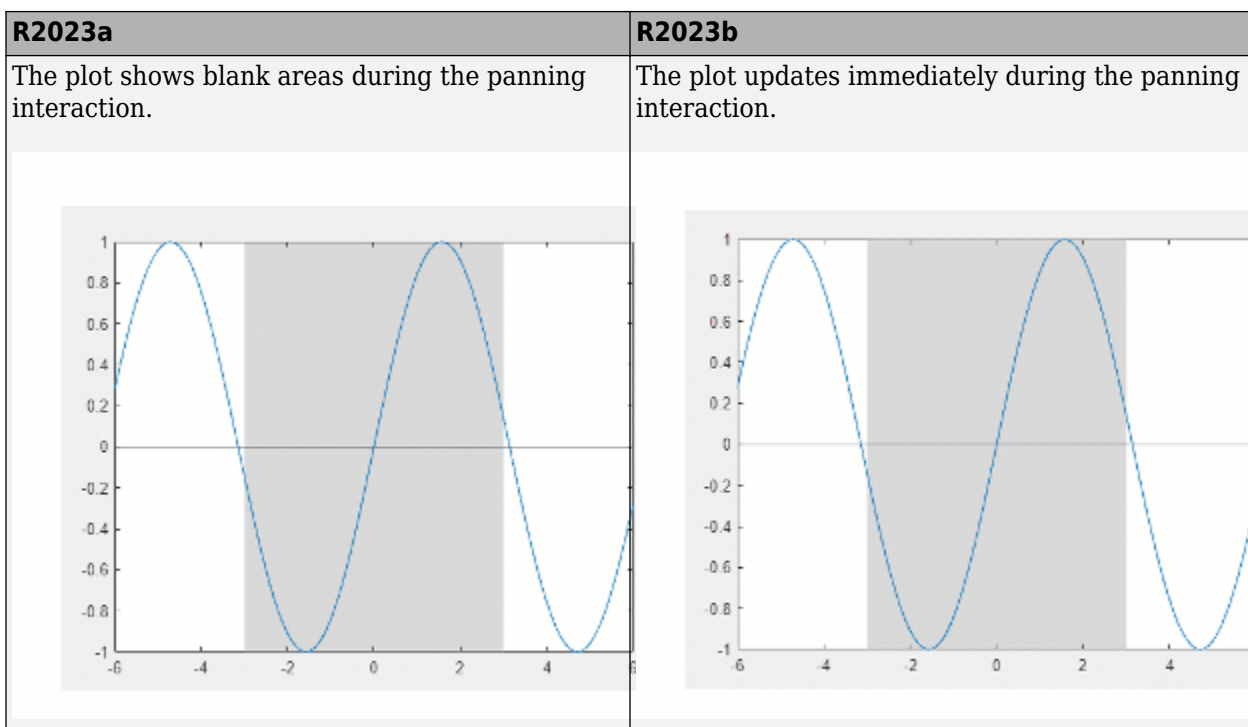
| R2023a                                                                                                                                                                                                                                                                                                                                            | R2023b                                                                                                                                                                                                                                                                                                                                                            |
|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| The rotation of the axes lags behind the movement of the cursor.                                                                                                                                                                                                                                                                                  | The rotation of the axes is smoother and follows the movement of the cursor more closely.                                                                                                                                                                                                                                                                         |
|  <p>A 3D plot of a helix curve, colored with a gradient from blue at the bottom to yellow at the top. The plot is shown in a 3D coordinate system with axes labeled from -2 to 2. The rotation of the axes appears to lag behind the movement of the cursor.</p> |  <p>A 3D plot of a helix curve, colored with a gradient from blue at the bottom to yellow at the top. The plot is shown in a 3D coordinate system with axes labeled from -2 to 2. The rotation of the axes is smoother and follows the movement of the cursor more closely.</p> |

## Plots in Apps: Constant lines and constant regions update continuously as you pan

In apps, panning a plot containing `ConstantLine` or `ConstantRegion` objects is smoother and the contents of the axes update immediately as you pan. Previously, panning showed blank areas when you panned outside of the original axes limits.

For example, on a Windows 10, Intel Xeon CPU W-2133 @ 3.60 GHz test system, if you run this code and then pan within the axes, the horizontal line and the gray vertical region update immediately as you pan.

```
function myapp
fig = uifigure;
ax = axes(fig);
x = -10:0.1:10;
y = sin(x);
plot(ax,x,y);
xlim(ax,[-6 6])
yline(ax,0)
xregion(ax,-3,3)
end
```



## App Building: Improved performance when resizing apps with nested containers

When a user resizes an app figure window for the first time, many apps reposition their content significantly faster in R2023b than in R2023a. The improvement occurs when resizing an app with one or more containers that have the `AutoResizeChildren` property set to "on", which is the default value. The improvement is more noticeable for apps with nested containers, such as apps with many tabs or tab groups.

For example, this code creates an app that contains five top-level tabs, each of which contains five nested tabs. Each of the nested tabs contains 100 edit fields. When you run this app and resize it, the resize operation is about 30x faster in R2023b than in the previous release.

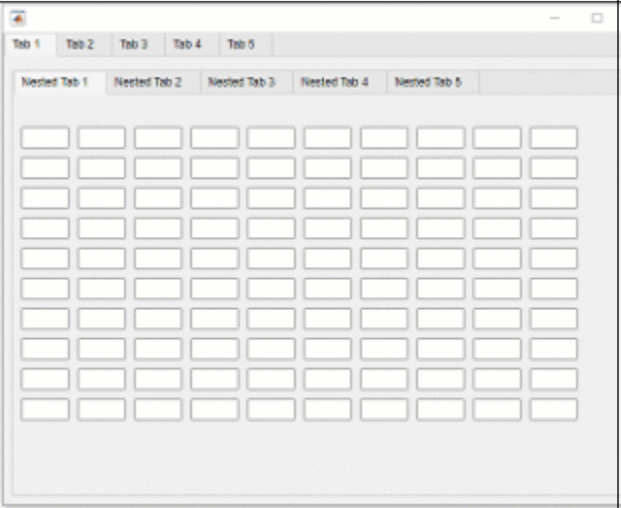
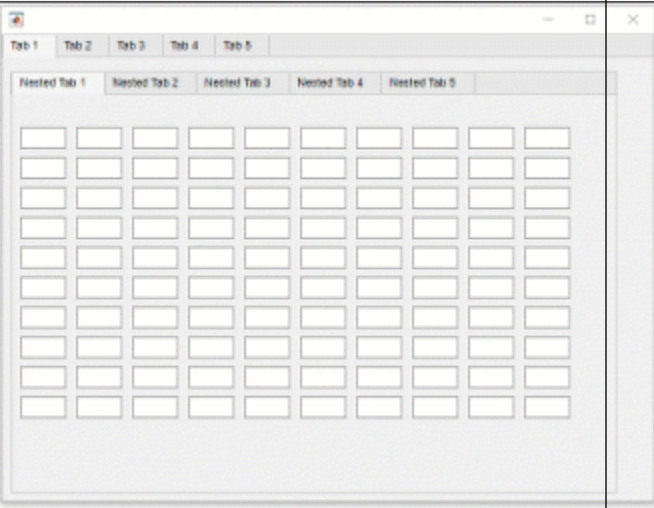
```
function myApp
f = uifigure;
f.Position = [100 100 700 500];

% Create five tabs in a tab group
tg = uitabgroup(f,"Position",[1 1 700 500]);
nTabs = 5;
for kTab = 1:nTabs
    tab = uitab(tg,"Title","Tab " + kTab);
    createTabContent(tab);
end

function createTabContent(tab)
    % Create a nested tab group containing five tabs
    nestedTg = uitabgroup(tab,"Position",[10 10 650 450]);
    nNestedTabs = 5;
    for kNTab = 1:nNestedTabs
```

```
        nestedTab = uitab(nestedTg,"Title","Nested Tab "+ kNTab);
        % Add 100 edit fields to the tab
        createNestedTabContent(nestedTab);
    end
end

function createNestedTabContent(tab)
    % Add 100 edit fields to a tab
    hParent = 400;
    n = 10;
    w = 50;
    h = 22;
    for kRow = 1:n
        for kCol = 1:n
            uieditfield(tab,"Position",[(kCol-1)*(w+10)+10,hParent-kRow*(h+10),w,h]);
        end
    end
end
end
end
```

| R2023a                                                                             | R2023b                                                                              |
|------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------|
| The app takes about 60 seconds to resize.                                          | The app takes about two seconds to resize.                                          |
|  |  |

The resize interactions were times on a Windows 10, Intel Xeon Gold 6246R CPU @ 3.40 GHz test system by running the myApp function and measuring the time it takes for the figure window to resize.

**App Building: Improved startup performance for apps with panels in multiple tabs**

In addition to the overall app startup performance improvement in R2023b, some apps that contain panels in multiple tabs show an even greater startup performance improvement. The reason is that MATLAB prioritizes creating the panel content in the visible tab over non-visible content when the app first runs.

For example, this code creates a tab group with five tabs, each containing a panel with 200 label components. The code is about 1.2x faster than in the previous release.

```
function timingTabApp
fig = uifigure;
tg = uitabgroup(fig);
for k1 = 1:5
    t = uitab(tg);
    p = uipanel(t,Position=[1 1 t.Position(3) t.Position(4)], ...
        Scrollable="on");
    for k2 = 1:200
        y = k2*20;
        uilabel(p,Position=[20 y 60 20]);
    end
end
end
```

The approximate execution times are:

**R2023a:** 7.2 s

**R2023b:** 6.1 s

The code was timed on a Windows 10, Intel Xeon CPU E5-1650 v4 @ 3.60 GHz test system by running the `timingTabApp` function and measuring the time it takes for the components to appear in the UI figure window.

If you have an app with many UI components, consider updating your app layout to take advantage of this improvement. For more information, see [Improve App Startup Time](#).

## App Designer: Improved app loading and running performance

Loading existing apps in App Designer is faster in R2023b than in R2023a. When you load an App Designer app, it takes less time for the app to appear in **Design View**.

For example, loading an app with about 400 UI components by selecting the app file on the App Designer Start Page is about 1.3x faster than in the previous release.

The approximate load times are:

**R2023a:** 17.1 s

**R2023b:** 13.6 s

In addition, running an app from App Designer is also faster in R2023b.

For example, for an app with about 400 UI components, running the app by clicking the **Run** button in App Designer results in the app appearing about 1.3x faster than in the previous release.

The approximate execution times are:

**R2023a:** 25.5 s

**R2023b:** 19.3 s

The actions were timed on a Windows 10, Intel Xeon CPU E5-1650 v4 @ 3.60 GHz test system by loading and running a saved app.

## **App Designer: Improved performance when adding custom UI components to an app**

When you drag a custom UI component from the **Component Library** onto the App Designer canvas, the UI component appears sooner in R2023b than in R2023a. This improvement is more noticeable the first time you add a custom UI component to an app.

For example, on a Windows 10, Intel Xeon CPU E5-1650 v4 @ 3.60 GHz test system, the first time you add a custom UI component to an app in **Design View**, the component appears on the canvas within one second. In R2023a, the component takes several seconds to appear.

## **Property Inspector in MATLAB Online: Improved performance when opening for the first time**

The Property Inspector shows improved performance when opening for the first time in MATLAB Online. The delay between clicking the **Property Inspector** icon or calling `inspect` and the inspector being ready is reduced.

For example, open the Property Inspector for the first time in MATLAB Online. You can use the Property Inspector 1.8x sooner than in the previous release.

```
ax = axes;  
inspect(ax)
```

The approximate rendering times are:

**R2023a:** 7.5 s

**R2023b:** 4.1 s

The rendering of the Property Inspector was timed by running the code and measuring the time it takes for the edit fields to appear in the Property Inspector.

## **Graphics in MATLAB Online: Images and 3-D plots in apps have significant performance gains**

Displaying images and 3-D plots in apps is significantly faster in MATLAB Online. You might also experience similar improvements outside the context of an app (by calling `figure` instead of `uifigure`).

For example, this code displays a scaled-up version of `peppers.png` using the `image` command. The code is about 22x faster than in the previous release.

```
function myapp  
f = uifigure;  
ax = axes(f);  
smallimg = imread("peppers.png");  
bigimg = uint8(zeros(3065,4089,3));  
bigimg(:,:,1) = uint8(interp2(double(smallimg(:,:,1)),3));
```



```
bigimg(:,:,2) = uint8(interp2(double(smallimg(:,:,2)),3));  
bigimg(:,:,3) = uint8(interp2(double(smallimg(:,:,3)),3));  
  
tic  
image(ax,bigimg)  
drawnow  
toc  
end
```



The approximate execution times are:

**R2023a:** 138.5 s

**R2023b:** 6.4 s

The code was timed by calling the `myapp` function. Because MATLAB Online is hosted, performance is independent of the computer hardware used to access it.

## Software Development Tools

### ★ **Source Control API: Interact with Git source control programmatically**

You can now programmatically interact with source control.

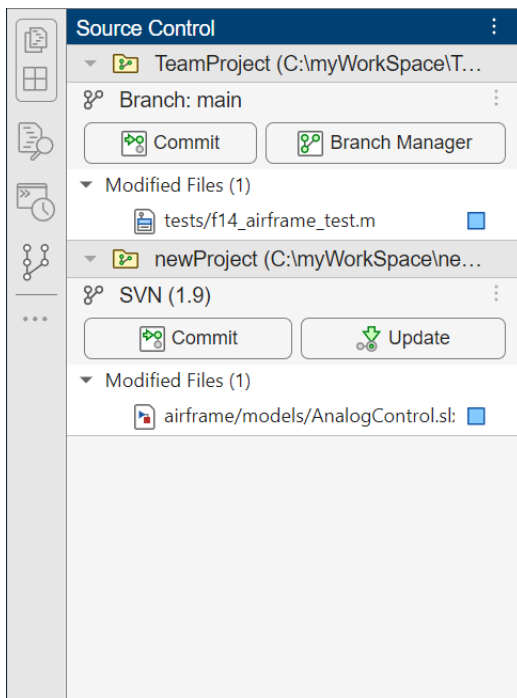
- Clone a Git repository using the `gitclone` function.
- Create a Git repository object using the `gitrepo` function.
- Initialize a Git repository using the `gitinit` function.
- Create, delete, and switch branches using the `createBranch`, `deleteBranch`, and `switchBranch` functions, respectively.
- Add files, remove files, and commit changes to a Git repository using the `add`, `rm`, and `commit` functions, respectively.
- Inspect the commit history in a Git repository using the `log` function.
- Display the status of files in a local Git repository using the `status` function.
- Fetch or pull new data from remote Git repositories using the `fetch` or `pull` functions, respectively.
- Merge Git branches and revisions into the current branch using the `merge` function.
- Publish your local changes to a remote Git repository using the `push` function.

### **Project API: Determine whether file belongs to a project**

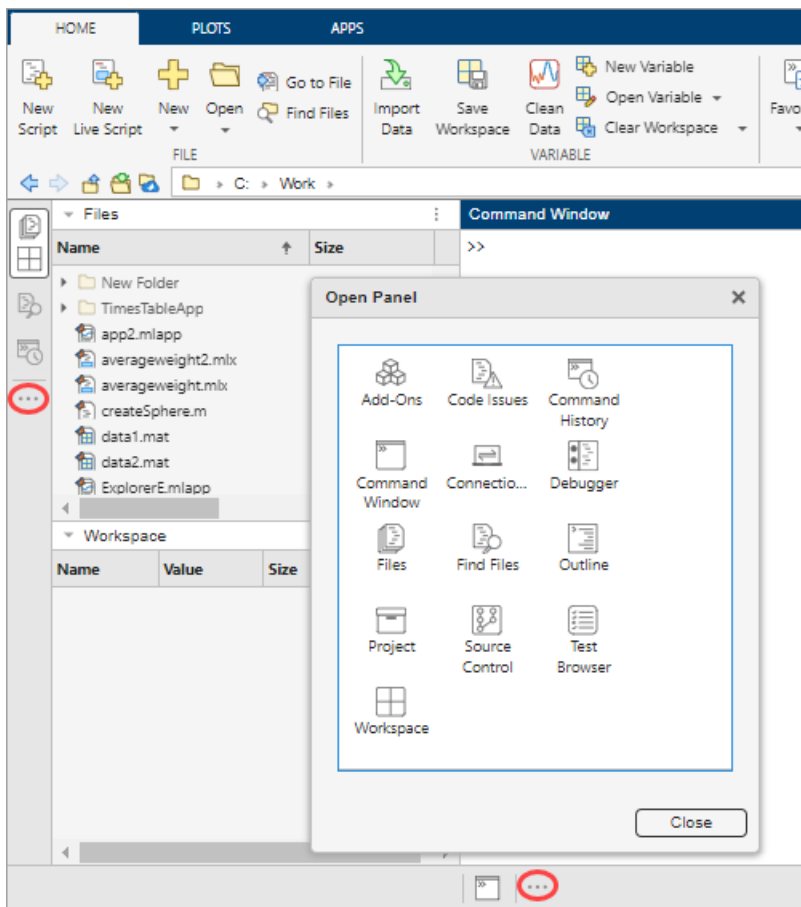
You can now programmatically determine whether a file or a folder belongs to a project by using the `matlab.project.isFileInProject` function.

### **Source Control in MATLAB Online: Perform source control operations using unified panel**

In MATLAB Online, you can use the Source Control panel to see all active source control repositories, manage modified files, and perform source control operations.



To open the Source Control panel, use the Open more panels button (\*\*\*) in the sidebar.



## Source Control in MATLAB Online: Expanded support for Git workflows

MATLAB Online now provides expanded support for Git workflows:

- Adding and managing Git submodules
- Sharing to GitHub
- Initializing Git repositories
- Shallow cloning Git repositories

## Projects in MATLAB Online: Added support for team collaboration workflows

Projects in MATLAB Online now provide support for the following team collaboration workflows:

- Creating referenced projects from project folders
- Adding source control to existing projects
- Managing project labels and custom tasks using **Project Settings**
- Displaying shadowed files on project startup
- Upgrading projects using **Project Upgrade**

## Project Examples: Identify and run tests in projects

This example shows how to use labels to identify tests in a project and how to create test suites from project test files interactively and programmatically. For large projects under source control, the example demonstrates how to run a subset of tests to reduce qualification runtime. For more information, see [Identify and Run Tests in MATLAB Projects](#).

## ★ Build Automation: Use built-in tasks to define common build actions

When using the build tool, you can create common tasks more quickly and conveniently by using the classes in the `matlab.buildtool.tasks` package. Instead of writing code to implement tasks, instantiate the classes in this package. Each class provides the flexibility to create a task tailored to your build requirements. For an example, see [Create and Run Tasks Using Build Tool](#).

## Build Automation: Automatically open MATLAB project when running builds

If your build file is in the root folder of a MATLAB project that is not already open, the build tool automatically opens the project before running the build and then closes the project after the build runs.

## Build Automation: Invoke build tool from project subfolders

You can call the `buildtool` command from the folder containing the build file or any of its subfolders. In previous releases, you can call `buildtool` only when the build file is in your current folder.

## Build Automation: Operate on file collections

The `matlab.buildtool.io.FileCollection` class has four new methods that enable you to create file collections and operate on them:

- To create file collections from the paths to files and folders, use the `matlab.buildtool.io.FileCollection.fromPaths` method.
- To replace substrings in the paths of a file collection, use the `replace` method.
- To transform the paths of a file collection, use the `transform` method.
- To filter a file collection, use the `select` method.

For an example, see [Replace Substring in File Collection Paths](#).

## Build Automation: Name the inputs and outputs of tasks for incremental builds

You can use the `matlab.buildtool.TaskInputs` and `matlab.buildtool.TaskOutputs` classes to name and group the inputs and outputs of your tasks. The `TaskInputs` and `TaskOutputs` objects behave like structures, where field names are task input or output names and field values are task input or output values. Input values can be of any data type, and output values must be `matlab.buildtool.io.FileCollection` arrays. For an example, see [Create Task with Named Inputs and Outputs](#).

## Build Automation: Run tasks using build options

You can use build options to run tasks with the `buildtool` command or the `run` method:

- To run the subsequent tasks upon a build environment setup or task failure, use the `-continueOnFailure` option of `buildtool` or the `ContinueOnFailure` name-value argument of `run`.
- To specify a task to skip, use the `-skip` option of `buildtool` or the `Skip` name-value argument of `run`.

For example, run the default tasks in your build plan as well as the tasks on which they depend, and continue running the build if a failure occurs.

```
buildtool -continueOnFailure
```

## Unit Testing Framework: Programmatically access code coverage results using the `runtests` function

You can use the `runtests` function to programmatically access the results of code coverage analysis for your source code. To run tests and return coverage results, specify the additional output argument of `runtests` as well as the `ReportCoverageFor` name-value argument. When you invoke `runtests` by specifying both these arguments, the function returns the coverage results as a `matlab.coverage.Result` vector. For example, run your tests and return the results of code coverage analysis for the source code in your current folder.

```
[testResults,coverageResults] = runtests("myTestFile.m", ...  
    ReportCoverageFor=pwd)
```

## Unit Testing Framework: Test for missing elements

The `matlab.unittest.constraints.HasMissing` class provides a constraint to test if an array has any missing elements. The constraint is satisfied by an array that has at least one missing element.

## Unit Testing Framework: Modify test report title

You can modify the test report title when you generate a test report using a method of the `matlab.unittest.plugins.TestReportPlugin` or `matlab.unittest.TestResult` class. To modify the title, specify the `Title` name-value argument. For example, run your tests and generate a PDF test report with a specified title.

```
results = runtests("myTestFile.m");  
generatePDFReport(results,Title="My Test Report")
```

## App Testing Framework: Automatically scroll to components outside viewable area

If you programmatically perform a gesture on a UI component that is not in the viewable area of an app, the app testing framework automatically scrolls to the specified component before performing the gesture. For the framework to bring the component into view, the component must be supported by the `scroll` function and all its ancestors must have their `Scrollable` property set to `"on"` or

true. For example, create a scrollable figure with a state button that is outside the viewable area of the figure, and then press the button.

```
fig = uifigure(Scrollable="on");
b = uibutton(fig,"state",Position=[fig.Position(3)+100 100 100 22]);

testCase = matlab.uitest.TestCase.forInteractiveUse;
testCase.press(b)
```

## **matlab.test.behavior.Missing Class: Test against constructor arguments that result in incompatible values**

You can verify whether objects of your class created with certain constructor arguments satisfy the missing value contract in MATLAB. To test your class with a constructor argument, specify the `ExtraConstructorArguments` property of the `matlab.test.behavior.Missing` class as that argument. This property is useful when different constructor arguments result in incompatible values, such as values that cannot be concatenated or compared.

## **⚠️ Functionality being removed or changed**

### **Build tool restores the current folder after running each task**

*Behavior change*

To prevent a task from affecting subsequent tasks by changing the current folder, the build tool restores the current folder after running each task. In previous releases, the build tool does not restore the current folder after running each task.

### **files converts relative paths to absolute paths**

*Behavior change*

If you create file collections with the `files` method, the paths of the file collections are absolute paths. Previously, the method did not convert any relative paths that you specified to absolute paths. To create file collections with relative paths, use the `matlab.buildtool.io.FileCollection.fromPaths` method, or use a string vector to specify an argument or property that requires file collections.

## External Language Interfaces

### Call MATLAB from C++: Generate output types for strongly typed methods and functions

The `matlab.engine.typedinterface.generateCPP` function generates output types for functions and methods in the strongly typed C++ interface. For more information, see [Output Argument Validation](#). For an example, see [Write MATLAB Code for Strongly Typed C++ Interface](#).

### Publish C++ Interface: Support for pointer return of string and fundamental types

When publishing a MATLAB interface to a C++ library function, you can define pointer return arguments for non-const string and fundamental types. You also can specify a deleter function for both const and non-const string and fundamental pointer return types. MATLAB uses the deleter function to manage the life cycle of the memory represented by the argument.

For examples, see [Define Output Pointer Argument](#).

### Call MATLAB from .NET: Generate C# code interface for MATLAB packages, classes, and functions

The `matlab.engine.typedinterface.generateCSharp` function creates C# .cs files from MATLAB packages, classes, and functions. For more information, see [Strongly Typed Interface for C#](#).

### Call .NET from MATLAB: Create explicit interface view of .NET object

You can create an explicit view of a .NET object as one of its implemented interfaces using the `NET.interfaceView` function. Interfaces can define methods or properties that are not accessible from the .NET object directly.

### Python: Support for version 3.11

MATLAB now supports CPython version 3.11, in addition to existing support for versions 3.9 and 3.10. For supported version information, see [Versions of Python Compatible with MATLAB Products by Release](#).

### Python Interface: Use of environment variable PYTHONHOME to load interpreter

MATLAB accesses these settings when loading the Python interpreter:

- User settings for the Python environment, configured using the `pyenv` function. If the version you specify conflicts with the version specified by `PYTHONHOME`, then delete the environment variable before calling `pyenv`. The value set by `pyenv` is persistent across MATLAB sessions. For more information, see [Set Python Version on Windows Platform](#) or [Set Python Version on Mac and Linux Platforms](#).



- PYTHONHOME environment variable, specifies where to find the standard Python libraries. To view the value of the variable in MATLAB, type `getenv("PYTHONHOME")`. For more information, see Use PYTHONHOME Environment Variable.
- System PATH environment variable. To view the system path in MATLAB, type `getenv("PATH")`.
- Windows registry.

## Python Interface: Convert list and tuple of Python datetime or timedelta objects to MATLAB arrays

If a Python function returns multiple `datetime` objects as a list or a tuple of Python `datetime` objects, use the MATLAB `datetime` function to convert the list or tuple to a MATLAB `datetime` array. For more information, see Handle Multiple Python `datetime` Objects Returned from Python Function.

If a Python function returns multiple `timedelta` objects as a list or a tuple of Python `timedelta` objects, use the MATLAB `duration` function to convert the list or tuple to a MATLAB `duration` array. For more information, see Handle Multiple Python `timedelta` Objects Returned from Python Function.

## Python Interface: Resolve Python configuration issues

Error messages provide additional help for you to resolve these Python configuration issues:

- Unable to find a supported Python version. See Versions of Python Compatible with MATLAB Products by Release.
- Python installed from the Microsoft Store is not supported. See Configure Your System to Use Python.
- Python shared library not found on UNIX® platforms. See Set Python Version on Mac and Linux Platforms.

## ▲ Compilers: Download MinGW-w64 8.1 from Add-On Explorer

As of R2023b, you can download MinGW®-w64 8.1 from the Add-On Explorer.

To download or configure other supported versions of MinGW, see MATLAB Support for MinGW-w64 C/C++ Compiler. For an up-to-date list of supported compilers, see Supported and Compatible Compilers.

## Compilers: Support for NAG Fortran on Apple silicon

MATLAB supports the NAG® Fortran compiler on Apple silicon platforms. For an up-to-date list of supported compilers, see Supported and Compatible Compilers.

## Compilers: Support for Microsoft Visual Studio Build Tools

MATLAB supports Build Tools for Visual Studio® 2022 and 2019 for building C and C++ interfaces, MEX files, and standalone MATLAB engine and MAT-file applications on Windows platforms. For an up-to-date list of supported compilers, see Supported and Compatible Compilers.

## ▲ **Functionality being removed or changed**

### **Python: Version 3.8 is no longer supported**

#### *Errors*

Support for Python version 3.8 is discontinued. For continued support for your applications, upgrade to a supported version of Python. For supported version information, see [Versions of Python Compatible with MATLAB Products by Release](#).

### **Perl: MATLAB support for version 5.38.0 on Windows**

#### *Behavior change*

As of R2023b, MATLAB on Windows ships with an updated version of Perl, version 5.38.0, and supports an updated version of `HTML::Parser`, version 3.81.

If you use the `perl` command on Windows platforms, see <https://www.perl.org/> for information about using this version of the Perl programming language. For a standard distribution of `HTML::Parser`, source code, and information about using `HTML::Parser`, see <https://metacpan.org/pod/HTML::Parser>.

## Hardware Support

### **Support for ArduCam Multi Camera Adapter Module on Raspberry Pi Hardware**

Starting in R2023b, you can use the new `arducam` object to set up a connection between MATLAB and ArduCam Multi Camera Adapter Module V2.2. You can sequentially acquire images from up to four cameras attached to the ArduCam Multi Camera Adapter Module V2.2. MATLAB Support Package for Raspberry Pi Hardware supports 5MP OV5647 cameras, which are compatible with ArduCam Multi Camera Adapter Module V2.2.



# R2023a

---

**Version: 9.14**

**New Features**

**Bug Fixes**

**Compatibility Considerations**

## Environment

### ▲ **MATLAB Install: Reduce footprint using default installation without local documentation**

Starting in R2023a, the documentation is not installed as part of a MATLAB or other product installation. This change significantly reduces the installation footprint of products. In most cases, not installing the documentation has no effect on the availability of documentation, as the Help Center displays the web documentation by default.

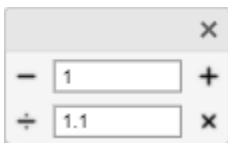
### ▲ **Compatibility Considerations**

The documentation is not installed as part of a product installation. If you run MATLAB on a system with no internet connection (permanently offline), or if you plan to work offline occasionally on an otherwise internet-connected machine, you can install the documentation on your computer after installing products. For more information, see [Install Documentation](#).

### **Editor: Interactively increment numeric values within section and run section after every change**

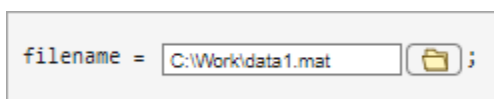
You can increment, decrement, multiply, or divide numeric values in the Editor and then run the current section after every change. This workflow can help you fine-tune and experiment with your code.

To adjust a numeric value, select the value or place your cursor next to the value. Next, right-click and select **Increment Value and Run Section**. In the dialog box that appears, specify a step value for addition and subtraction or a scale value for multiplication and division. Then, click one of the operator buttons to add to, subtract from, multiply, or divide the selected value in your section. MATLAB runs the section after every click.



### **Live Editor Controls: Add file browser to select file interactively in live script**

You can add a file browser to your live script to interactively select a file by opening a file selection dialog box. To add a file browser, go to the **Live Editor** tab, and in the **Code** section, click  **Control**. Then, select **File Browser**.



For more information, see [Add Interactive Controls to a Live Script](#).

## Live Editor Controls: Align controls within a section when code is hidden

The Live Editor automatically left-aligns drop-down lists, edit fields, numeric sliders, and numeric spinners within a block of code when the code is hidden.

For example, this live script contains three controls in two different blocks of code. When the code is visible, the controls appear inline with the code.

Specify the panel location.

`pLocation =`   `;`

Specify the panel size and efficiency.

`pSize = 17`  `;`

`pEfficiency =`   `;`

When the code is hidden, the Live Editor automatically aligns the second and third controls, as they are in the same block of code. The first control is not in the same block of code and is therefore not aligned to the other two controls.

Specify the panel location.

Location:

Specify the panel size and efficiency.

Panel Size (m<sup>2</sup>):

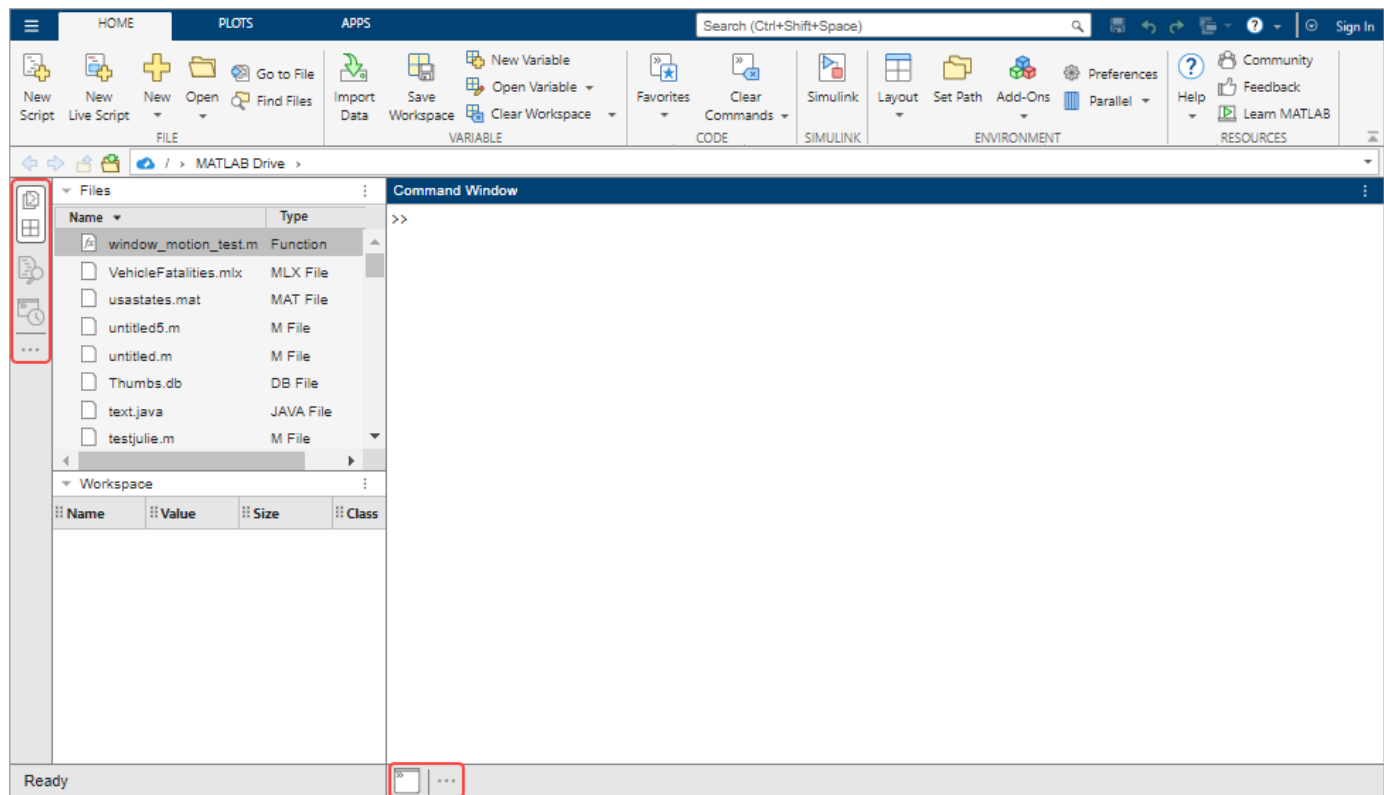
Panel Efficiency:

The Live Editor does not align check boxes, buttons, and file browsers.

## Desktop Layout in MATLAB Online: Access desktop tools and change the desktop layout using sidebars

The MATLAB Online desktop includes sidebars on either side of the desktop to access desktop tools and change the desktop layout. The sidebars show the tools, such as the Workspace panel and the Files panel, that are docked on either side of the desktop. If there are no tools docked on one side, the sidebar for that side is hidden. You can use the sidebars to show and hide tools, group them together, and move them from one location to another.

When a tool is docked on the left or right side of the desktop, the sidebar on that side displays an icon for the tool. To show or hide the tool, click its icon in the sidebar. To show and hide multiple tools together, group them by dragging one of the tool icons next to another tool icon. To move a tool to a different location on the desktop, drag the tool or the icon for the tool to the new location. If there are no tools docked on one side, then the sidebar on that side is hidden.

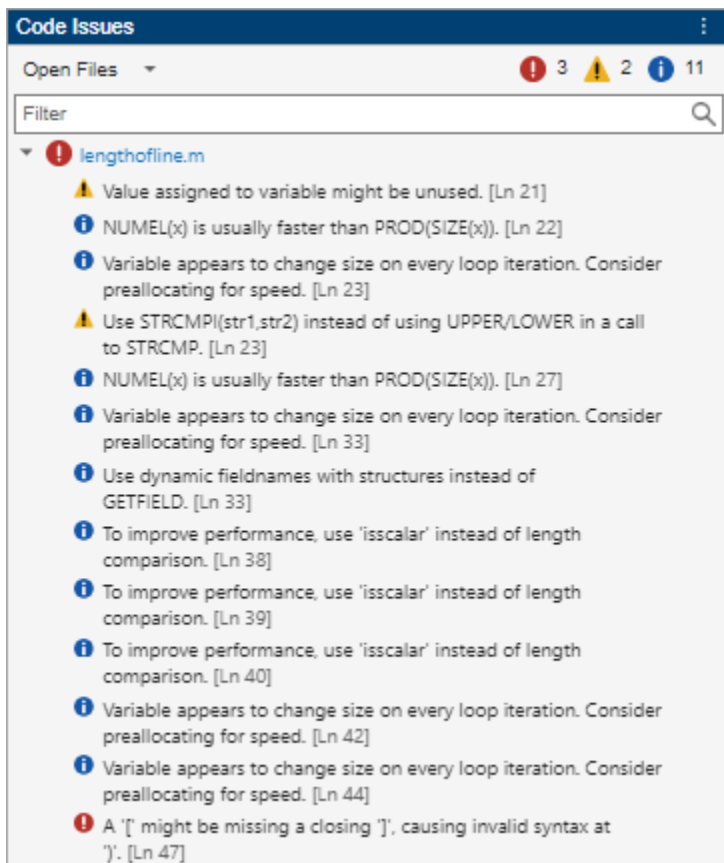


## Code Issues Tool in MATLAB Online: Check code for errors and warnings using Code Issues tool

You can use the Code Issues tool to view error and warning messages about your code. The Code Issues tool displays the coding problems found by the MATLAB Code Analyzer as it automatically checks your code. Using the Code Issues tool, you can choose to view the errors and warnings for the current file or for all open files. You also can filter the list of messages by type (error or warning) as well as by message text.

To open the Code Issues tool, go to the **Editor** or **Live Editor** tab, and in the **Analyze** section, click  **Code Issues**. By default, the Code Issues tool opens on the right side of the desktop. To hide the Code Issues tool, click the Code Issues  icon in the sidebar.

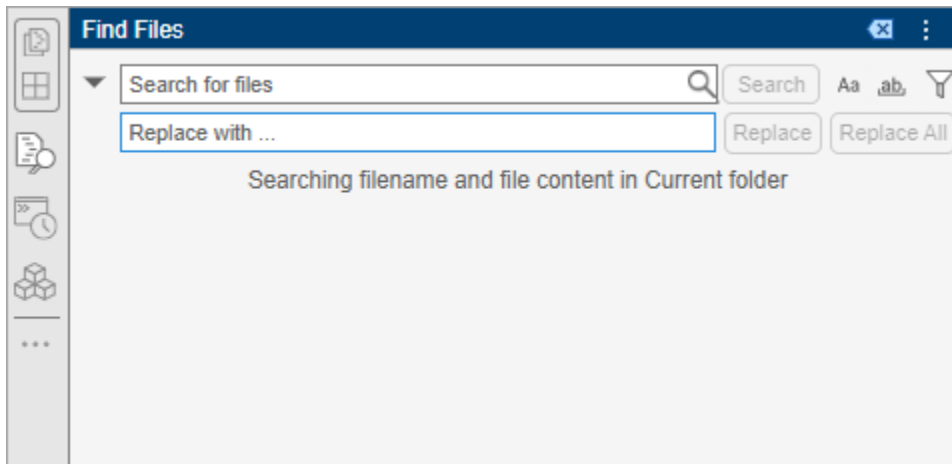




## Find Files Tool in MATLAB Online: Search for files with improved Find Files tool

You can use the improved Find Files tool to search for files based on name or content. When searching, you can choose whether to match the case of your search text as well as whether to match the whole word. You also can select what folder to search in and filter results by file extension. To

search for files, click the Find Files icon  in the sidebar on the left side of the MATLAB Online desktop.



## Editor in MATLAB Online: Edit read-only files

You can now edit read-only files in MATLAB Online. When you save the file, MATLAB prompts you to overwrite the file or save the file using a different name.

## Search in MATLAB Online: Access videos using Search box

In MATLAB Online, you can now use the Search box in the upper-right corner of the desktop to easily access videos that show you how to use MathWorks products. In addition to videos, the Search box results include toolstrip actions, preferences, and Help Center resources such as functions, blocks, examples, and answers. To navigate to the Search box using a keyboard, press **Ctrl+Shift+Space** (or **Command+Shift+Space** on macOS systems).

## System Command Functions: Set and get multiple variables using operating system commands

Set and get multiple environment variables using the `setenv`, `unsetenv`, `getenv`, and `isenv` functions with string arrays and cell arrays of character vectors as input.

## loadenv Function: Load environment variables from .env and plain text files

You can load environment variables from a `.env` or other plain text file by using the `loadenv` function. By using a `.env` file you can separate sensitive configuration data from code and provide different sets of configurations for different workflows.

## Comparison Tool: Save text comparison results as PDF or DOCX reports programmatically and interactively

You can now use the Comparison Tool to programmatically and interactively publish text comparison results as PDF or DOCX reports. For more details, see `Compare Text Files` and `visdiff`.

## Toolbox Packaging: Specify packaging options when creating custom toolbox programmatically

When creating a custom toolbox programmatically using the `matlab.addons.toolbox.packageToolbox` function, you can specify packaging options using a `ToolboxOptions` object. Using the `ToolboxOptions` object, you can specify information about the toolbox including what platforms the toolbox supports and what MATLAB releases the toolbox is compatible with. You also can specify what files to package and what additional software to install with the toolbox. For an example, see [Specify packaging options using ToolboxOptions object](#).

### **Functionality being removed or changed**

#### **XML comparison type for visdiff function will be removed**

*Still runs*

The XML comparison type for the `visdiff` function will be removed in a future release. Overriding the default comparison type by specifying "xml" will not be supported in a future release. In R2023a, scripts that use `visdiff(filename1,filename2,"xml")` continue to work.

#### **lookfor function searches help text in third-party and user-authored MATLAB program files**

*Behavior change*

The `lookfor` function searches help text in third-party and user-authored MATLAB program files. In R2022b, `lookfor` does not search help text in third-party and user-authored MATLAB program files.

# Language and Programming

## Code Analyzer: Enable custom checks and configure existing checks

You can configure existing checks displayed in the MATLAB editor by the Code Analyzer and add custom checks by placing a file named `codeAnalyzerConfiguration.json` in a resources folder. The configuration file is effective in the folder containing the resources folder and any subfolders.

You can modify existing Code Analyzer checks, including whether the check is enabled and its severity, message text, and parameters if the check has any, such as to limit the number of input and output arguments for a function. You can also create custom checks that trigger when specific functions are used. For more information on configuring Code Analyzer checks, see [Configure Code Analyzer](#).

Validate your `codeAnalyzerConfiguration.json` configuration file for proper formatting by using `matlab.codeanalysis.validateConfiguration`.

## fix Function: Fix code issues from the command line

Fix certain code issues directly from the command line using the `fix` function on `codeIssues` objects.

For example, create a script file with the following code:

```
x = [1 2 3]
for n = 1:3
    y(n) = x
end
```

Run `codeIssues` on the file to identify code issues.

```
issues = codeIssues("exampleScript")
```

```
issues =
```

```
codeIssues with properties:
    Date: 03-Nov-2022 10:46:40
    Release: "R2023a"
    Files: "C:\MyCode\exampleScript.m"
    CodeAnalyzerConfiguration: "active"
    Issues: [3x10 table]
    SuppressedIssues: [0x11 table]
```

Issues table preview

| Location          | Severity | Fixability |                                                |
|-------------------|----------|------------|------------------------------------------------|
| "exampleScript.m" | info     | auto       | "Add a semicolon after the statement to hide t |
| "exampleScript.m" | info     | manual     | "Variable appears to change size on every loop |
| "exampleScript.m" | info     | auto       | "Add a semicolon after the statement to hide t |

Note that if the value for `Fixability` is `auto` then the issue can be fixed using `fix`. Issues with `manual` will not be fixed. Use `fix` to apply the recommended fix to the issue identified.

```
fix(issues,"NOPTS")

ans =

codeIssues with properties:

    Date: 03-Nov-2022 10:47:04
   Release: "R2023a"
    Files: "C:\MyCode\exampleScript.m"
CodeAnalyzerConfiguration: "active"
    Issues: [1x10 table]
SuppressedIssues: [0x11 table]
```

Issues table preview

| Location          | Severity | Fixability |                                                 |
|-------------------|----------|------------|-------------------------------------------------|
| "exampleScript.m" | info     | manual     | "Variable appears to change size on every loop" |

Code Analyzer App: Apply fixes to code issues interactively

Fix certain code issues directly from the Code Analyzer app using the **Fix** button.

Info (42)

▼ ⓘ Add a semicolon after the statement to hide the output (in a script). (14)

Fix All

|        |                 |                                                            |     |
|--------|-----------------|------------------------------------------------------------|-----|
| Line 1 | CCR_test.m      | foo 4                                                      | Fix |
| Line 3 | CCR_test.m      | bar 5                                                      | Fix |
| Line 1 | blah.m          | SP whitespacePattern ""                                    | Fix |
| Line 2 | blah.m          | AFT [";",",",";"]                                          | Fix |
| Line 3 | blah.m          | pat ":" + SP + AFT                                         | Fix |
| Line 1 | evalText.m      | local 1                                                    | Fix |
| Line 1 | exampleScript.m | x [1:2                                                     |     |
| Line 1 | maskpatTest.m   | hexDigit characterListPattern('0','9')   characterListPatt | Fix |
| Line 2 | maskpatTest.m   | hexNum "0x" + asManyOfPattern(hexDigit,1)                  | Fix |
| Line 3 | maskpatTest.m   | hexDigit maskPattern(hexDigit)                             | Fix |
| Line 4 | maskpatTest.m   | hexNum "0x" + asManyOfPattern(hexDigit,1)                  | Fix |
| Line 1 | t.m             | foo 1                                                      | Fix |
| Line 6 | test.m          | x 0                                                        | Fix |
| Line 2 | test_fcns.m     | F=@() sin(0.1)+cos(0.2)                                    | Fix |

▶ ⓘ For readability, use '~contains(str1, str2)' instead of 'isempty(strfind(str1, str2))'. (7)

dictionary Object: Access and assign dictionary cell values with curly braces

Data in cells can be looked up and assigned directly using curly braces, {}. When dictionary values are cells, a lookup using parentheses, (), returns a cell. Accessing the contents of that cell requires indexing into the cell. This feature allows contents of cell values to be accessed directly. For example, the dictionary d has three cell values.

```
d =  
  
dictionary (double [] cell) with 3 entries:  
  
1 [] {"Hello"}  
2 [] {[4 5 6]}  
3 [] {@sin}
```

Perform a lookup using parentheses. The lookup returns the value as a cell containing an array.

```
d(2)  
  
ans =  
  
1×1 cell array  
  
{[4 5 6]}
```

Perform a lookup using curly braces. The lookup returns the value as an array.

```
d{2}  
  
ans =  
[4 5 6]
```

## Output Argument Validation: Debug within output argument blocks

You can now use the MATLAB debugger within output arguments blocks of functions. While debugging an arguments block, the workspace is read-only. For more information on using the debugger, see [Debug MATLAB Code Files](#).

### Functionality being removed or changed

#### Warning about indexing with no subscripts (MATLAB:subscripting:noSubscriptsSpecified) has been removed

The warning about indexing into a built-in type with no subscripts (MATLAB:subscripting:noSubscriptsSpecified) has been removed. In previous releases, the warning was off by default, but when users opted in, the warning would appear when indexing with no subscripts.

```
warning("on", "MATLAB:subscripting:noSubscriptsSpecified");  
x = [1 2 3 4 5];  
y = x()
```

Warning: A value of class "double" was indexed with no subscripts specified. Currently the result of this operation is the indexed value itself, but in a future release, it will be an error.

```
y =  
  
1      2      3      4      5
```

Starting in R2023a, the behavior of indexing with no subscripts remains the same and does not error, but the warning no longer appears. Using the warning command to turn the warning on or off has no effect.

**Defining classes and packages: Using `schema.m` will not be supported in a future release***Still runs*

Support for classes and packages defined using `schema.m` files will be removed in a future release. Replace existing schema-based classes with classes defined using the `classdef` keyword.

## Data Analysis

### **pivot Function: Summarize tabular data using pivot table**

Perform a pivoting operation on data in a table or timetable by using the `pivot` function. Specify grouping variables that define variables or rows in the pivoted table using `colvars` or `rowvars`.

Optionally define parameters such as the data variable, function to apply to the data variable, and grouping variable binning schemes by specifying name-value arguments.

### **table and timetable Data Types: Perform calculations directly on tables and timetables without extracting their data**

You can now perform calculations directly on tables and timetables without extracting their data. All the variables in your tables and timetables must have data types that support calculations. You can also perform operations where one operand is a table or timetable and the other is a numeric or logical array. Previously, all calculations required you to extract data from your tables and timetables by indexing into them.

For more information, see [Direct Calculations on Tables and Timetables](#) and [Rules for Table and Timetable Mathematics](#).

### **Timetable Events: Find and label events in timetables using attached event tables**

To find and label events in a timetable, attach an *event table* to it. An event table is a timetable of *events*. An event consists of an event time (when something happened), often an event length or event end time (how long it happened), often an event label (what happened), and sometimes additional information about the event. Event tables are designed to be attached to timetables. When you attach an event table to a timetable, you can find or label rows in the timetable that occur during events.

MATLAB provides these functions to create event tables from input data, filter timetable rows on event times, and synchronize events to timetables:

- `eventtable`
- `extractevents`
- `eventfilter`
- `syncevents`

### **combinations Function: Generate all element combinations of arrays**

Generate all element combinations of arrays of varying sizes and data types by using the `combinations` function. Each row of the output table is a combination. Applying the `combinations` function is equivalent to finding the Cartesian product of sets of elements.



## fillmissing2 Function: Fill missing entries in two-dimensional data

Fill missing entries in two-dimensional data sets using the `fillmissing2` function. You can fill missing entries using interpolation or moving window methods.

## fillmissing Function: Use values from nearest neighbors to fill missing data

You can use the `'knn'` method of the `fillmissing` function to fill missing entries in your data with the corresponding values in the nearest row, based on the pairwise Euclidean distance between rows. You can optionally specify a value `k` for the `'knn'` method to fill missing entries with the mean of the corresponding values in the `k` nearest rows.

You can also specify a distance function, using the `Distance` name-value argument, to measure the distance between rows with a specified metric. `Distance` can have any of these values:

- `'euclidean'` — Euclidean distance (default)
- `'seuclidean'` — Scaled Euclidean distance
- A function handle — User-specified distance function

## Descriptive Statistics and Arithmetic: Omit or include missing data of multiple data types

When analyzing and preprocessing data, you can optionally specify to omit or include numeric, datetime, duration, and categorical missing data. Use the `"omitmissing"` or `"includemissing"` flags in addition to any previously supported missing condition flags for these functions:

- Basic statistics — `min`, `max`, `bounds`, `median`, `mean`, `std`, `var`, `rms`
- Basic arithmetic — `sum`, `prod`
- Forecasting metrics — `rmse`, `mape`
- Covariance — `cov`
- Cumulative statistics — `cummax`, `cummin`, `cumsum`, `cumprod`
- Moving statistics — `movmin`, `movmax`, `movmean`, `movmedian`, `movmad`, `movstd`, `movvar`, `movsum`, `movprod`
- Data preprocessing — `smoothdata`, `detrend`

## Find and Remove Trends Live Editor Task: Interactively find and remove periodic and polynomial trends

The Find and Remove Trends task in the Live Editor can now identify periodic trends for regularly spaced input data. Select the **Periodic** trend type, and choose the SSA or STL algorithm.

The task can also return polynomial and periodic trends, in addition to the detrended data. Specify **Output** as Trends.

Previously, this task identified only polynomial trends and returned only the detrended data.

## ⚠ Compatibility Considerations

To reflect the enhanced functionality of finding and removing periodic and polynomial trends, this task in the Live Editor is now named Find and Remove Trends. Previously, this task was named Remove Trends.

## Data Cleaner App: Save session as MAT-file and reload session

When working in the Data Cleaner app, save the session as a binary MAT-file containing the data and cleaning steps. To save the session file, in the **File** section of the **Home** tab, click **Save**. To reload the session, in the **File** section of the **Home** tab, click **Open**.

## head and tail Functions: Get top or bottom rows of array

Display or return the top rows of a vector, matrix, multidimensional array, or cell array using the `head` function, or the bottom rows using the `tail` function. Previously, `head` and `tail` supported only table, timetable, and tall array input data.

## ⚠ groupsummary Function: Compute number of unique elements

Compute the number of distinct nonmissing elements in each group of data. Specify the "numunique" or "all" method of the `groupsummary` function, or select the **Number of unique values** or **Select all** computation method in the Compute by Group task in the Live Editor.

## ⚠ Compatibility Considerations

The "all" computation method now returns the number of unique values in addition to the computation methods in the previous release.

## ⚠ movevars Function: Move table variables after last variable without After name-value argument

When using the `movevars` function, you can now move variables after the last variable without specifying the `After` name-value argument.

For example, create a table. Then move the first variable after the last variable.

```
T = table([1;3;5],[2;4;6],[3;6;9])
```

```
T =  
3×3 table  
    Var1    Var2    Var3  
    ____    ____    ____  
     1         2         3  
     3         4         6  
     5         6         9
```

```
T = movevars(T,"Var1")
```

```
T =  
3×3 table  
    Var2    Var3    Var1  
    ____    ____    ____
```

|   |   |   |
|---|---|---|
| 2 | 3 | 1 |
| 4 | 6 | 3 |
| 6 | 9 | 5 |

## ⚠ Compatibility Considerations

In previous releases, calling `movevars` without specifying either the `After` or `Before` name-value argument resulted in an error. Moving a variable after the last variable required one of these two syntaxes.

```
T = movevars(T, "Var1", "After", width(T))
% or
T = movevars(T, "Var1", "After", "Var3")
```

## rowfun and varfun Functions: OutputFormat name-value argument can take "auto" as value

The `OutputFormat` name-value argument can take `"auto"` as a value. This value is the default value. It causes the `rowfun` and `varfun` functions to return outputs whose data types match the data types of their inputs.

## day Function: Return ISO day of week

To return the ISO day of week number, use the `"iso-dayofweek"` option with the `day` function. In the ISO 8601 standard, Monday is day 1 of the week.

For example, return the ISO day of week number for today.

```
D = datetime("today");
dayNumber = day(D, "iso-dayofweek")
```

## week Function: Return ISO week of year or week of month

To return the ISO week of year number, use the `"iso-weekofyear"` option with the `week` function. In the ISO 8601 standard, a week begins on Monday. Week 1 of a year is defined as the first week in the year with at least four days.

To return the week of month number, use the `"iso-weekofmonth"` option with the `week` function. A week begins on Monday. Week 1 of a month is defined as the first week in the month with at least four days. ISO 8601 does not specifically define the week of month number. However, this option returns a week of month number that is consistent with the ISO week of year number.

For example, return the ISO week of year and week of month numbers for today.

```
D = datetime("today");
weekOfYear = week(D, "iso-weekofyear")
weekOfMonth = week(D, "iso-weekofmonth")
```

## **Variables Editor: Edit categories of categorical table variable in MATLAB Online**

In the MATLAB Online Variables editor, you can create, remove, or merge categories in a categorical table or timetable variable. To edit the categories, pause on the header of a categorical variable and click the triangle icon, or right-click the variable, and select **Edit Categories**.

## Data Import and Export

### Import Data Live Editor Task: Import data in live scripts

The Import Data Live Editor task allows you to import various types of data in a live script within a single framework. You can import these types of data:

- MAT-file (for example, .mat)
- Text (for example, .csv)
- Spreadsheet (for example, .xlsx)
- Image (for example, .png)
- Audio (for example, .wav)
- Video (for example, .avi)

To add the task to a live script in the Live Editor, click **Task** on the **Live Editor** tab and select the **Import Data** icon.

### audiowrite Function: Write MP3 audio files

You can write MP3 audio files using the `audiowrite` function.

### imfinfo Function: Get information about XMP metadata embedded in JPEG files

The `imfinfo` function returns Extensible Metadata Platform (XMP) metadata embedded in JPEG files in the 'XMPData' field of the output structure. The function also returns International Press Telecommunications Council (IPTC) metadata embedded in the XMP namespace. The IPTC data is stored in the 'Iptc4xmpCore' (core metadata) and 'Iptc4xmpExt' (extension metadata) subfields of 'XMPData'.

### Parallel Processing: Use readtable in thread-based environments

You can use the `readtable` function in thread-based environments. Parallel processing results in improved performance when reading data, especially with remote data.

### Scientific File Format Libraries: CDF library upgraded to version 3.8.1

The CDF library is upgraded to version 3.8.1.

### Scientific File Format Libraries: CFITSIO library upgraded to version 4.1.0

The CFITSIO library is upgraded to version 4.1.0.

## ⚠️ **Functionality being removed or changed**

### **web function will return handle to most recent MATLAB web browser as MATLAB class**

*Behavior change in future release*

In a future release, the `web` function will return a handle to the most recent MATLAB web browser as a MATLAB class. Currently, the `web` function returns the handle as a Java class. With this change, some methods that were previously supported in the returned handle will no longer be supported.

In most cases, you will not need to make any changes to your code. However, if you are using methods that are not supported in the returned MATLAB class, you will need to update your code.

## Mathematics

### MATLAB Support Package for Quantum Computing: Build, simulate, and run quantum algorithms

The MATLAB Support Package for Quantum Computing enables you to:

- Build circuits to implement quantum algorithms using a variety of built-in gate functions.
  - `quantumCircuit`
  - `quantum.gate.SimpleGate`
  - For a complete list of built-in gate functions, see [Types of Quantum Gates](#).
- Leverage composite gates to create custom gates from available built-in gates, capture complex operations, and organize circuits.
  - `quantum.gate.CompositeGate`
- Verify implementation of quantum algorithms with simulations on your local computer. Analyze simulation results to determine the outcome of a measurement.
  - `simulate`
  - `quantum.gate.QuantumState`
- Run gate-based quantum algorithms by connecting to quantum hardware on Amazon® Web Services (AWS®).
  - `run`
  - `quantum.backend.QuantumDeviceAWS`
  - `quantum.backend.QuantumTaskAWS`
  - `quantum.gate.QuantumMeasurement`

See [Quantum Computing](#) and [Introduction to Quantum Computing](#) for more information.

To install the MATLAB Support Package for Quantum Computing, locate the support package in **Add-On Explorer** using the instructions in [Get and Manage Add-Ons](#).

### pageeig Function: Perform eigenvalue decomposition on pages of N-D arrays

Use the `pageeig` function to calculate eigenvalues and eigenvectors of the pages of N-D arrays. In this context, the N-D array is treated as a container for several 2-D matrices.

### randi Function: Create random logical array

Use `randi` to create a random logical array by specifying the `typename` argument as `"logical"` or the prototype `p` as a logical array.

For example, you can create a 5-by-5 random logical array using `randi([0 1],5,"logical")`.

## ⚠️ **Functionality being removed or changed**

### **spy plots have adjusted default behavior for aspect ratio and marker size**

#### *Behavior change*

The aspect ratio of spy plots has a 1-to-10 limit, after which the plot stops adjusting to the matrix shape. You can use `pbaspect("auto")` for no special aspect ratio or `pbaspect([size(A,2) size(A,1) 1])` for the previous behavior of a matching aspect ratio, even for very "squeezed" cases.

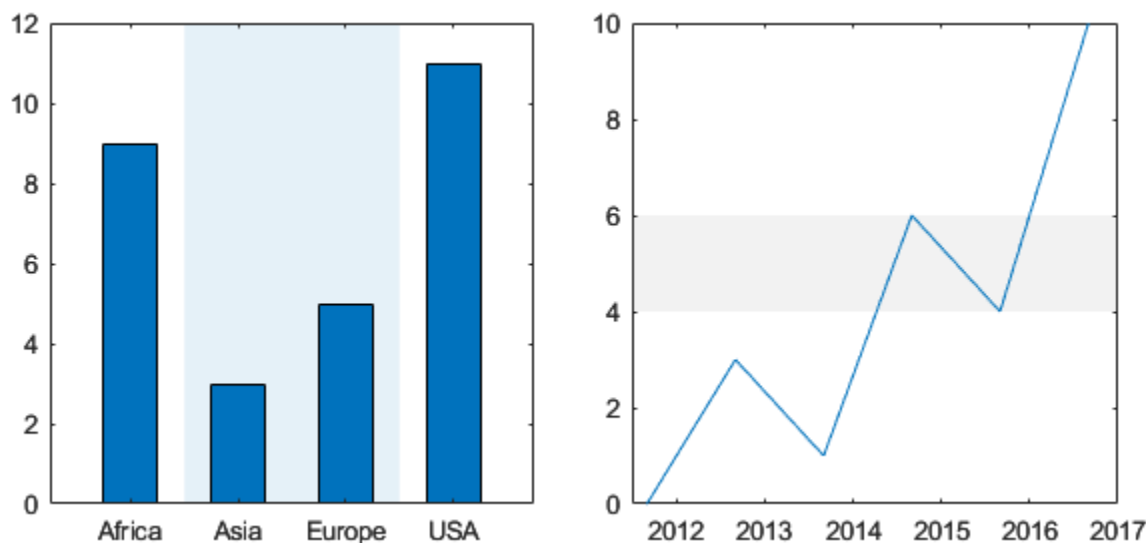
Additionally, the default marker size is based only on the matrix size. Previously, the point size of the axes on creation could also affect the marker size.



## Graphics

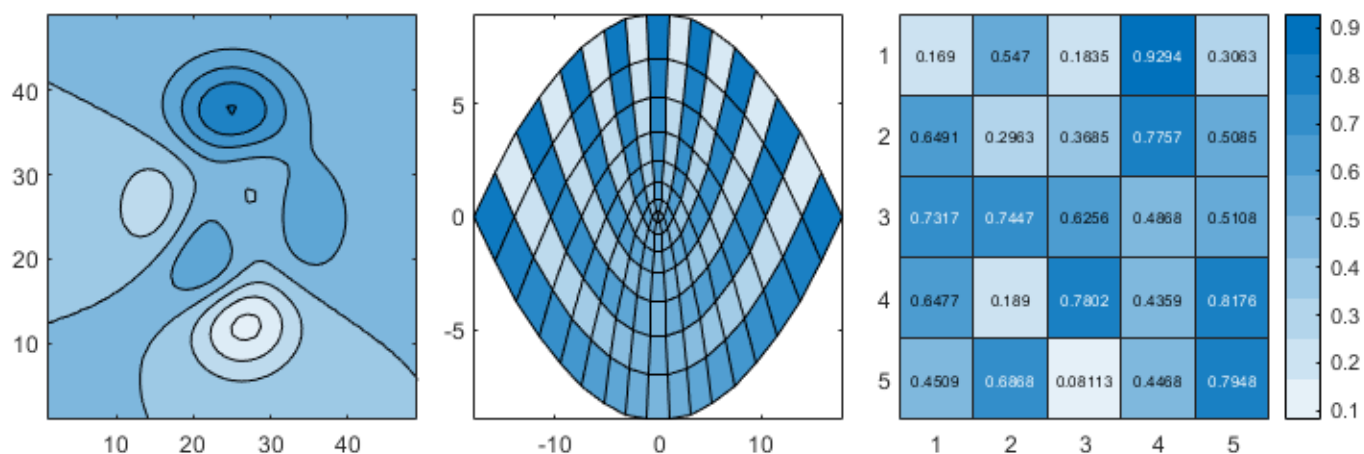
### xregion and yregion Functions: Highlight horizontal or vertical regions of plots

Highlight one or more horizontal or vertical regions of a plot with the `xregion` and `yregion` functions. You can set properties to customize the color and boundary lines of these shaded regions.



### sky Function: Apply monochromatic colormap to charts

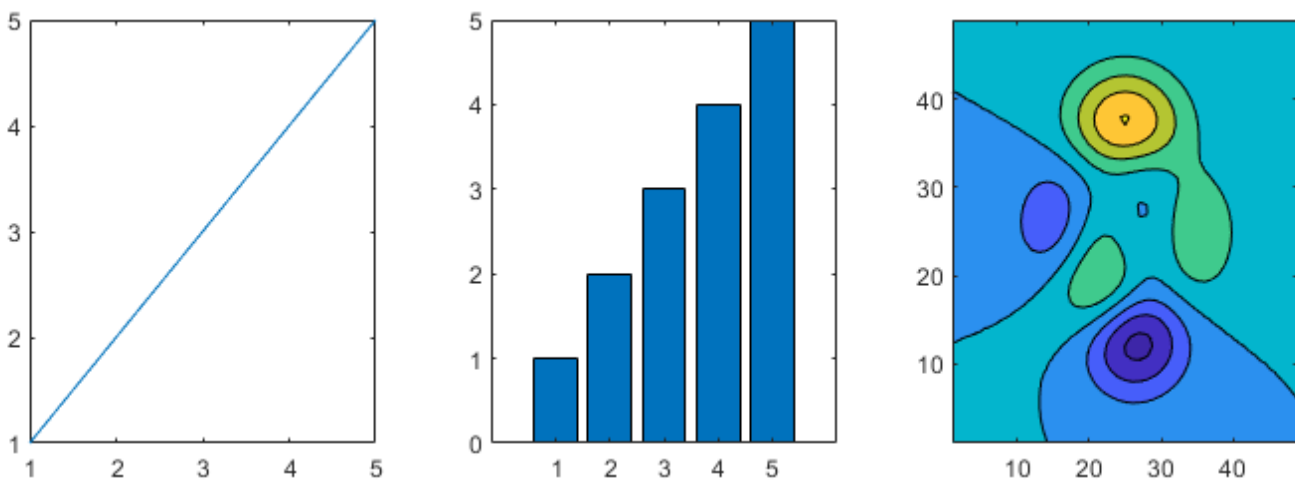
Use the `sky` function to color charts with the same monochromatic colormap that heatmap charts use. Like for all predefined colormaps, you can optionally specify the number of colors for the sky colormap.



## tiledlayout Function: Create horizontal or vertical layouts

Create horizontal or vertical layouts by specifying "horizontal" or "vertical" as the first input argument to the tiledlayout function. For example, create a horizontal layout and add three plots.

```
tiledlayout("horizontal")
x = 1:5;
nexttile
plot(x)
nexttile
bar(x);
nexttile
contourf(peaks)
```

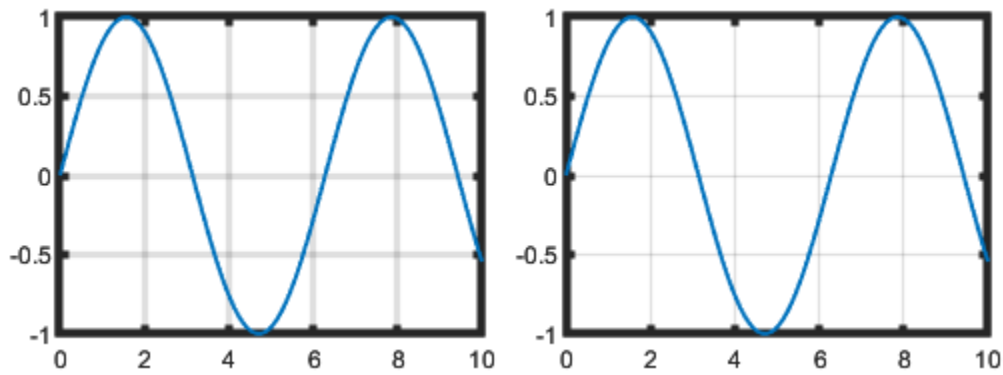


## animatedline Function: Create animated lines using numeric, datetime, or duration data

Use the animatedline function to create animated lines using single, double, integer, datetime, or duration data for the x-, y-, and z-coordinates.

## Grid Lines: Customize grid line thickness

Change the thickness of grid lines independently of the box outline and tick marks by setting the GridLineWidth and MinorGridLineWidth properties of the axes. Before R2023a, the LineWidth property of the axes was the only property for controlling the grid line width. However, that property controlled the grid lines, box outline, and tick marks together. Now you can control the thickness of the grid lines separately.

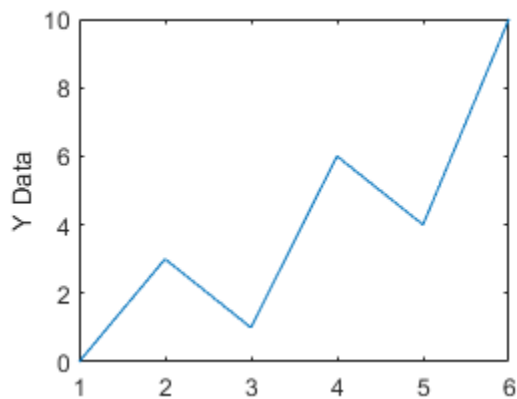


### Axes Labels: Rotate x- and y-axis labels without overlapping the axes

When you change the `Rotation` property of an axis label in a 2-D plot, the `HorizontalAlignment` and `VerticalAlignment` properties of the label automatically change to prevent overlap between the label and the axes.

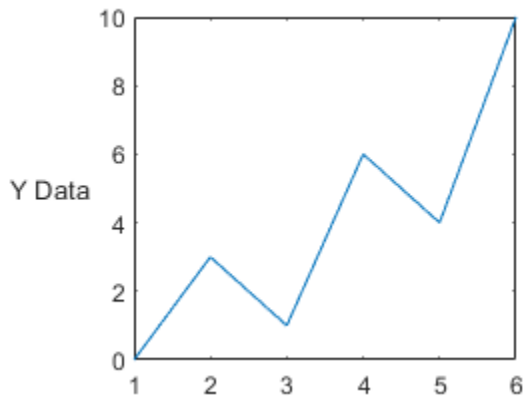
For example, create a plot with a y-axis label.

```
plot([0 3 1 6 4 10])
ylab = ylabel("Y Data");
```



Rotate the label so that the text is horizontal. MATLAB automatically adjusts the `HorizontalAlignment` and `VerticalAlignment` properties to prevent the overlap.

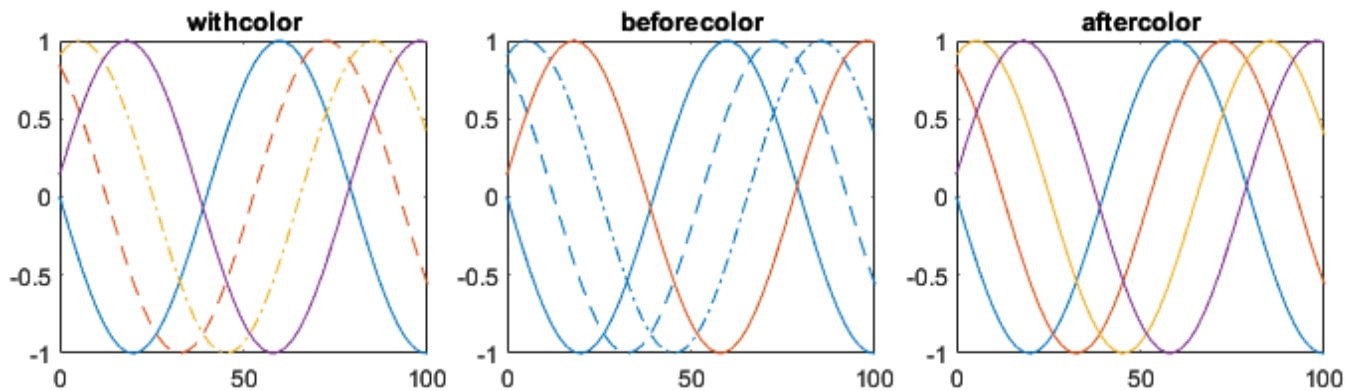
```
ylab.Rotation = 0;
```



## Plotting Series of Lines: Control cycling order of line styles

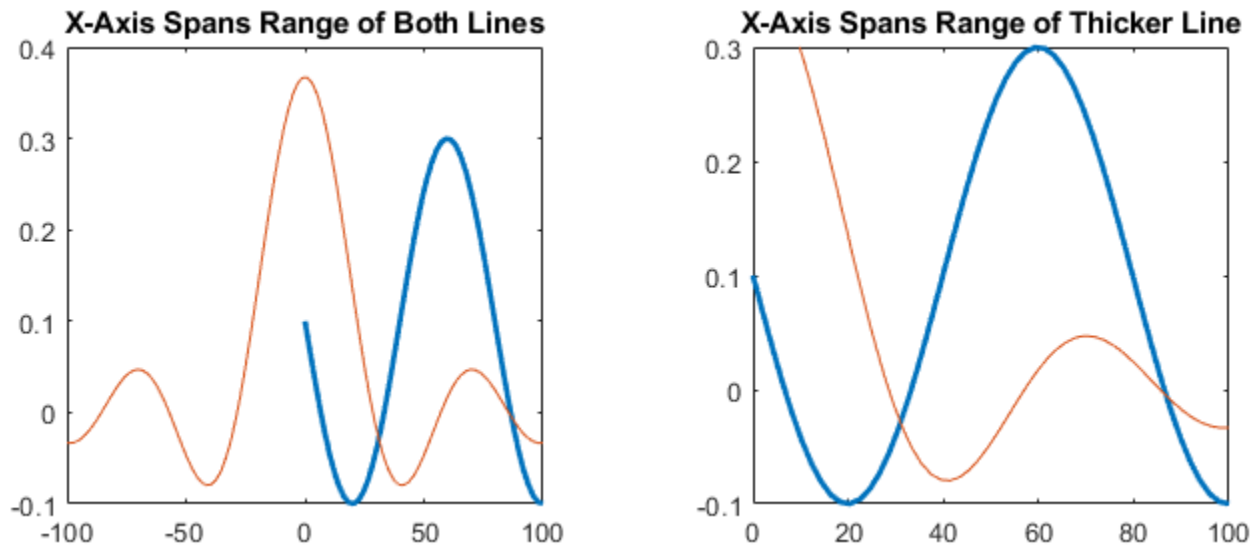
When plotting a series of multiple lines, you can use the `LineStyleCyclingMethod` property of the axes to control how different lines are distinguished from one another. Specify this property as one of these values:

- "withcolor" — Cycle through the line styles with the colors
- "beforecolor" — Cycle through the line styles before cycling through the colors
- "aftercolor" — Cycle through the line styles after cycling through the colors (default)



## Plotting Series of Lines: Control whether the data range of a line affects automatic axes limits

Specify whether a specific line affects the automatically selected axes limits by setting the `AffectAutoLimits` property. By default, the axes limits change to encompass the data range for each successive line you create. Setting this property enables you to focus on the range of a subset of lines in the axes.



### fontsize and fontname Functions: Optionally specify the object containing the text

When calling the `fontsize` or `fontname` functions, you can omit the object argument when you want the functions to affect the current figure.

### DatetimeRuler Object: Set or get the reference date for plotting datetime values

Set the `ReferenceDate` property of a `DatetimeRuler` object when you plot datetime values. This property is useful for synchronizing tick placement across different axes and for plotting data from different time zones together.

### Image Display Preferences: Access and update imshow preferences in MATLAB Online

In MATLAB Online, you can set the default values for these aspects of images displayed using `imshow`:

- **Axes visible** — Control whether `imshow` displays images with the axes box outline and tick labels.
- **Border Style** — Control whether `imshow` draws a tight or loose border around images in the figure window.
- **Initial Magnification** — Control whether `imshow` initially fits images to the figure window or magnifies them by a specified percentage.

To open these image display preferences, on the **Home** tab, in the **Environment** section, click  **Preferences**. Select **MATLAB > Image Display**.

## ▲ Functionality being removed or changed

### **BaseValue property of bar, stem, and area plots no longer changes with axes limits**

#### *Behavior change*

The BaseValue property of bar, stem, and area plots no longer depends on the axes limits. The property value stays the same when you change axes limits or pan within the axes.

This change does not affect the appearance of the plots, but it provides a more predictable experience when you change the axes limits or pan within the axes.

### **MATLAB Online limits imshow image display resolution**

#### *Behavior change*

MATLAB Online limits the maximum imshow image display resolution to improve rendering speeds for large images. This behavior affects the on-screen display, but it does not affect the image data. Before displaying an image, imshow resizes the largest dimension to a maximum size of 512 pixels. The smaller dimension adjusts to preserve the aspect ratio of the image. To view images at their full resolution, use MATLAB desktop or set the MaxRenderedResolution property of the output Image object to "none". For details about the MaxRenderedResolution property, see Image Properties.

### **Plot Catalog tool will be removed**

#### *Warns*

The Plot Catalog tool will be removed in a future release. Instead, to interactively create and explore visualizations for your data, use the **Plots** tab in the MATLAB Toolstrip or the Create Plot task in the Live Editor.

For more information about visualizations, see Types of MATLAB Plots or toolbox-specific documentation.

### **Figure Tools menu will no longer include interaction modes and options**

#### *Still runs*

In a figure, the **Tools** menu will no longer contain these items in a future release:

- **Zoom In**
- **Zoom Out**
- **Pan**
- **Rotate 3D**
- **Data Tips**
- **Brush**
- **Restore View**
- **Options**
- **Align Distribute Tool**

Instead, to enable interaction modes, use the axes toolbar. Customize the interaction by right-clicking in the chart when an interaction mode is enabled, or for apps, by using the InteractionOptions property of the axes.

## App Building

### **addStyle Function: Add styles to items in list box or drop-down UI component**

Create styles for specific items in a list box or drop-down UI component using the `uistyle` and `addStyle` functions. For example, you can add icons to items in a list box. To get information on applied styles, query the `StyleConfigurations` property of the `ListBox` or `DropDown` object. To remove a style from a component, use the `removeStyle` function.

### **uistack Function: Change stacking order of UI components in UI figure**

You can now use the `uistack` function to change the stacking order of UI components and containers in a figure created using the `uifigure` function. Previously, `uistack` supported UI components only in figures created using the `figure` function.

### **uipanel and uibuttongroup Functions: Specify container border color**

You can specify the border color of panels and button groups by using the `BorderColor` property. For more information, see [Panel Properties](#) or [ButtonGroup Properties](#).

### **uihtml Function: Send events between MATLAB and HTML**

When you create an HTML UI component using the `uihtml` function, you can send events between MATLAB and HTML. Send events when a specific action occurs to one object and another object needs to know about or react to that action. For example, you can send an event from HTML to MATLAB whenever a user clicks a button HTML element, and then write a callback in MATLAB that updates your app in response. You also can send an event from MATLAB to HTML whenever a user clicks a `Button` UI component in your app, and then write a callback in your HTML source file that updates the HTML code in response.

To send an event from HTML to MATLAB, call the `sendEventToMATLAB` function on the `htmlComponent` JavaScript® object in your setup method.

```
htmlComponent.sendEventToMATLAB(eventName,eventData)
```

React to this event by writing MATLAB code that creates an `HTMLEventReceivedFcn` callback for the HTML MATLAB object.

```
comp.HTMLEventReceivedFcn = @myCallbackFunction
```

To send an event from MATLAB to HTML, call the `sendEventToHTMLSource` function on the HTML MATLAB object.

```
sendEventToHTMLSource(comp,eventName,eventData)
```

React to this event by writing JavaScript code in your `setup` method that listens for the event and executes a callback function in response.

```
htmlComponent.addEventListener(eventName,eventData,callbackFunction)
```

For more information, see [Send Event from JavaScript to MATLAB](#) and [Send Event from MATLAB to JavaScript](#).

## uiimage Function: Specify image alt text for use with screen readers

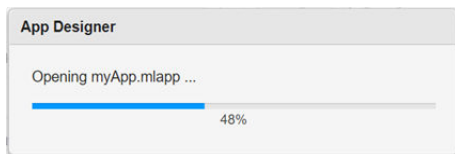
Provide a description of an image created using the `uiimage` function by specifying the `AltText` property. This property is used by screen readers to describe the image when an app user navigates through the app.

## appmigration.migrateGUIDEApp Function: Programmatically migrate existing GUIDE apps to App Designer

Programmatically migrate existing GUIDE apps to App Designer apps by using the `appmigration.migrateGUIDEApp` function. You can specify a single app, multiple apps, or a folder of apps to migrate as a batch. The function uses the GUIDE to App Designer Migration Tool for MATLAB to perform the migration.

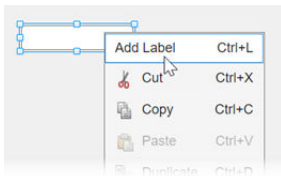
## App Designer: View progress when loading an app

When you load an app in App Designer, a progress bar now displays an estimate of its loading progress.



## App Designer: Add label to unlabeled UI component

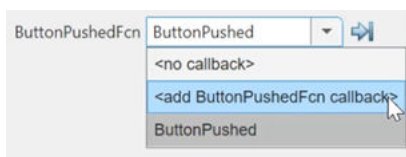
To add a label to a UI component without one, in **Design View**, right-click the component and select **Add Label** or use the keyboard shortcut **Ctrl+L**.



## App Designer: Replace assigned callback with new callback

You can now more efficiently create a new callback for a UI component that already has a callback assigned. To replace an existing callback with a new one, select the component in the **Component Browser** and, in the **Callbacks** tab, select the option to add a callback. For example, for a component with a `ButtonPushedFcn` callback already assigned, select `<add ButtonPushedFcn callback>` from the associated drop-down list. App Designer creates a new callback function, assigns it to the component, and unassigns the previous callback function.





Previously, you had to first unassign the assigned callback from the component before creating a new callback.

## App Testing Framework: Test context menus within labels

You can use the `chooseContextMenu` method to test a right-click that opens a context menu within label components. For example, assign a context menu with two items to a label, and then choose the first menu item.

```
fig = uifigure;
lbl = uilabel(fig);

cm = uicontextmenu(fig);
m1 = uimenu(cm,Text="Menu1");
m2 = uimenu(cm,Text="Menu2");
lbl.ContextMenu = cm;

testCase = matlab.uittest.TestCase.forInteractiveUse;
testCase.chooseContextMenu(lbl,m1)
```

## Axes Interactions: Customize behavior of interactions with axes view

For apps created in App Designer and using the `uifigure` function, customize axes interaction behavior using the `InteractionOptions` property of the axes. Customize the behavior of panning, zooming, rotating, adding data tips, data brushing, and restoring the original view by setting the value of `InteractionOptions` properties. For a complete list of properties, see [InteractionOptions Properties](#).

The options set by the `InteractionOptions` object apply to these interactions on the associated axes:

- The built-in interactions specified by the `Interactions` property of the axes
- Interactions enabled by using mode functions, such as `pan` and `zoom`
- Interactions enabled using the axes toolbar

For example, limit all pan and zoom interactions to the x-dimension only.

```
fig = uifigure;
ax = uiaxes(fig);
ax.InteractionOptions.LimitsDimensions = "x";
```

## Plots in Apps: Enable data cursor mode

For apps created in App Designer and using the `uifigure` function, use data cursor mode to interactively create and edit data tips. For supported charts, select the Data Tips  icon in the axes toolbar or use the `datacursormode` function.

## Plots in Apps: Specify axes for interaction mode

For apps created in App Designer and using the `uifigure` function, set the interaction mode for axes. Specify the Axes object as the first argument for these functions:

- `pan`
- `zoom`
- `rotate3d`
- `datacursormode`
- `brush`

For example, for a figure with two axes, enable pan mode for only axes `ax1`.

```
fig = uifigure;  
t = tiledlayout(fig,1,2);  
ax1 = nexttile(t);  
ax2 = nexttile(t);  
pan(ax1,"on")
```

When setting the interaction mode for axes, these functions do not return `pan`, `zoom`, `rotate3d`, `DataCursorManager`, or `brush` objects. Previously, these functions set the interaction mode for all Axes children of the current or target figure.

## ▲ Functionality being removed or changed

### **uistack function has different stacking behavior for menus, toolbars, push tools, and toggle tools**

#### *Behavior change*

When you modify the stacking order of menus, toolbars, push tools, and toggle tools using the `uistack` function, the behavior is different than in previous releases. For example, starting in R2023a, calling `uistack(comp,"up")` has this behavior:

- Menu items parented to a figure — The menu item moves one place to the left in the menu bar. Previously, the menu item moved one place to the right.
- Menu items parented to a context menu — The menu item moves one place up in the context menu. Previously, the menu item moved one place down.
- Toolbars — The toolbar moves one place up within the collection of toolbars in the figure. Previously, the toolbar moved one place down.
- Push tools and toggle tools — The push tool or toggle tool moves one place to the left in the toolbar. Previously, the push tool or toggle tool moved one place to the right.

If your code uses `uistack` to modify the stacking order of menus, toolbars, push tools, or toggle tools, make these updates to the code to retain the previous behavior.

| Original Code in R2022b or Earlier   | Updated Code in R2023a                 |
|--------------------------------------|----------------------------------------|
| <code>uistack(comp,"top")</code>     | <code>uistack(comp,"bottom")</code>    |
| <code>uistack(comp,"up",step)</code> | <code>uistack(comp,"down",step)</code> |
| <code>uistack(comp,"bottom")</code>  | <code>uistack(comp,"top")</code>       |

| Original Code in R2022b or Earlier     | Updated Code in R2023a               |
|----------------------------------------|--------------------------------------|
| <code>uistack(comp,"down",step)</code> | <code>uistack(comp,"up",step)</code> |

**HighlightColor property of panel and button group containers is not recommended**

*Still runs*

Using the `HighlightColor` property to specify the border color of a panel or button group is not recommended. Use the `BorderColor` property instead. The `BorderColor` property has the same effect and accepts the same values as the `HighlightColor` property. For more information, see [Panel Properties](#) or [ButtonGroup Properties](#).

There are no plans to remove support for the `HighlightColor` property at this time. However, the `HighlightColor` property no longer appears in the list returned by calling the `get` function on a UI container.

## Performance

### Language and Programming: Improved performance when calling functions and methods

Calling most functions and methods shows improved performance. For example, in a file named `myFun.m` in your current folder, create the `myFun` function.

```
function y = myFun(x)
y = x;
end
```

In a file named `timingTest.m` in your current folder, create a function that calls `myFun`. The `timingTest` function is about 1.6x faster than in the previous release.

```
function out = timingTest
n = 1e7;
for i = 1:n
    out = myFun(3);
end
end
```

The approximate execution times are:

**R2022b:** 0.18 s

**R2023a:** 0.11 s

The code was timed on a Windows 10, Intel Xeon CPU E5-1650 v4 @ 3.60 GHz test system using the `timeit` function.

```
timeit(@timingTest)
```

### Function Handles: Improved performance when invoking handles to named functions

Invoking handles to named functions that are not nested shows improved performance. Invoking such function handles no longer results in an overhead compared to calling functions directly. For example, in a file named `myFun.m` in your current folder, create the `myFun` function.

```
function y = myFun(x)
y = x;
end
```

In a file named `timingTest.m` in your current folder, create a function that invokes a handle to `myFun`. The `timingTest` function is about 40x faster than in the previous release.

```
function t = timingTest
f = @myFun;
n = 1e7;
tic
for i = 1:n
    out = f(3);
end
```

```
t = toc;
end
```

The approximate execution times are:

**R2022b:** 4.4 s

**R2023a:** 0.11 s

The code was timed on a Windows 10, Intel Xeon CPU E5-1650 v4 @ 3.60 GHz test system by calling the `timingTest` function.

## varargin Argument: Improved performance when specifying zero or more inputs

Specifying a variable number of input arguments using `varargin` shows improved performance. For example, in a file named `timingTest.m` in your current folder, create a function that expects one input and accepts an additional number of inputs.

```
function timingTest(x,varargin)
n = 1e6;
tic
for i = 1:n
    y = myFun(x,varargin{:});
end
toc
end

function y = myFun(x,varargin)
if nargin == 1
    y = x;
elseif nargin == 3
    y = x + varargin{1} + varargin{2};
else
    y = NaN;
end
end
```

The amount of improvement depends on whether `varargin` is empty. The performance improvement is most significant when `varargin` is empty.

- Empty `varargin` — Time this code by running `timingTest(1)`. The code is about 22x faster than in the previous release. The approximate execution times are:

**R2022b:** 0.404 s

**R2023a:** 0.018 s

- Nonempty `varargin` — Time this code by running `timingTest(1,2,3)`. The code is about 2x faster than in the previous release. The approximate execution times are:

**R2022b:** 1.428 s

**R2023a:** 0.734 s

The code was timed on a Windows 10, Intel Xeon CPU E5-1650 v4 @ 3.60 GHz test system.

## timetable Data Type Indexing: Improved performance when subscripting with times or with withtol subscript

timetable subscripting when subscripting with times or with a withtol subscript is significantly faster in R2023a than in R2022b.

- For example, when you use a vector of 100 datetime values to subscript into a timetable that has  $10^7$  rows, performance in R2023a is about 114x faster than in R2022b.

```
function timingTest()
    rng default

    % 10^7 rows
    N = 10000000;
    rowtimes = datetime(2023,1,1,0,0,0:N-1);
    rowtimes.Format = rowtimes.Format + ".SSS";
    tt = timetable(rand(N,1),RowTimes=rowtimes);

    % 100 values chosen in steps of 10
    n = 1000;
    t = datetime(2023,1,1,0,0,0:10:n-1);

    tic
    tt2 = tt(t,:);
    toc
end
```

The approximate execution times are:

**R2022b:** 9.10 s

**R2023a:** 0.08 s

- Similarly, when you use a vector of 100 duration values to subscript into a timetable that has  $10^7$  rows, performance in R2023a is about 14x faster than in R2022b.

```
function timingTest()
    rng default

    % 10^7 rows
    N = 10000000;
    rowtimes = seconds(0:N-1);
    tt = timetable(rand(N,1),RowTimes=rowtimes);

    % 100 values chosen in steps of 10
    n = 1000;
    t = seconds(0:10:n-1);

    tic
    tt2 = tt(t,:);
    toc
end
```

The approximate execution times are:

**R2022b:** 1.29 s

**R2023a:** 0.09 s

- When you use a `withtol` subscript with a timetable that has  $10^7$  rows, performance in R2023a is about 44x faster than in R2022b.

```
function timingTest()
    rng default

    % 10^7 rows
    N = 10000000;
    rowtimes = seconds(0:N-1);
    tt = timetable(rand(N,1),RowTimes=rowtimes);

    % 100 values chosen in steps of 10
    n = 1000;
    t = seconds(0:10:n-1);

    tt.Time = tt.Time + .1*seconds(rand(N,1));
    wt = withtol(t,seconds(.1));

    tic
    tt2 = tt(wt,:);
    toc
end
```

The approximate execution times are:

**R2022b:** 3.92 s

**R2023a:** 0.09 s

The code was timed on a Windows 10, AMD® EPYC 74F3 24-Core Processor @ 3.19 GHz test system by calling each version of the `timingTest` function.

## Complex Matrices: Improved performance when using colon indexing to copy complex matrices

Copying a complex matrix using colon indexing shows improved performance. This improvement is greater for larger matrices.

For example, this code is about 105x faster than in the previous release.

```
a = rand(100)*1j;

tic;
for i = 1:1e6
    b = a(:,:);
end
toc;
```

The approximate execution times are:

**R2022b:** 22.3 s

**R2023a:** 0.212 s

This improvement is most noticeable when the copy is not modified. However, examples like these still execute noticeably faster.

- Modify no elements of the copy based on a conditional (95x improvement).

```
a = rand(100)*1j;

tic;
for k = 1:1e6
    b = a(:,:,:);
    if (isreal(b(1,1)))
        b(1,1) = b(1,1)*1j;
end
toc;
```

**R2022b:** 21.3 s

**R2023a:** 0.223 s

- Modify an element of the copy based on a conditional (2x improvement).

```
a = rand(100)*1j;

tic;
for k = 1:1e6
    b = a(:,:,:);
    if (~isreal(b(1,1)))
        b(1,1) = b(1,1)*1j;
end
toc;
```

**R2022b:** 23.5 s

**R2023a:** 10.1 s

- Use an implicit copy in an operation (4x improvement).

```
a = rand(100)*1j;
b = rand(100)*1j;

tic;
for k = 1:1e6
    c = a(:,:,:) + b;
end
toc;
```

**R2022b:** 41.1 s

**R2023a:** 9.87 s

- Resize from 4-D to 2-D during copy (63x improvement).

```
a = rand(10,10,10,10)*1j;

tic;
for k = 1:1e6
    b = a(:,:,:);
end
toc;
```

**R2022b:** 28.2 s

**R2023a:** 0.451 s



The code was timed on a Windows 10, Intel Xeon CPU E5-2650 v2 @ 2.60 GHz test system.

## mean, std, var, and rmse Functions: Improved performance when computing along default vector dimension

The mean, std, var, and rmse functions show improved performance when computing over a real vector when the operating dimension is not specified. The functions determine the default operating dimension more quickly in R2023a than in R2022b.

For example, this code computes the mean along the default vector dimension. The code is about 2.2x faster than in the previous release.

```
function timingMean
A = rand(10,1);
for i = 1:8e5
    mean(A);
end
end
```

The approximate execution times are:

**R2022b:** 0.91 s

**R2023a:** 0.41 s

The code was timed on a Windows 10, Intel Xeon CPU E5-1650 v4 @ 3.60 GHz test system using the `timeit` function.

```
timeit(@timingMean)
```

## Moving Statistics Functions: Improved performance when computing over matrix with sample points

Moving statistics functions show improved performance when computing over a matrix when there are sample points. These functions show improved performance:

- `movmin`
- `movmax`
- `movmean`
- `movmedian`
- `movmad`
- `movstd`
- `movvar`
- `movsum`
- `movprod`

For example, this code computes the moving sums of a 300-by-300 matrix with corresponding sample points. The code is about 3x faster than in the previous release.

```
function timingMovsum
A = randn(300);
```

```
t = sort(rand(300,1));
tic
for k = 1:2000
    movsum(A,0.1,"SamplePoints",t);
end
toc
end
```

The approximate execution times are:

**R2022b:** 1.04 s

**R2023a:** 0.34 s

The code was timed on a Windows 10, Intel Xeon CPU E5-1650 v4 @ 3.60 GHz test system by calling the `timingMovsum` function.

## histcounts Function: Improved performance with small numeric and logical input data

The `histcounts` function shows improved performance for numeric and logical data due to faster input parsing. The performance improvement is more significant when input parsing is a greater portion of the computation time. This situation occurs when the size of the data to distribute among bins is smaller than 2000 elements.

For example, this code calculates histogram bin counts for a 1000-element vector. The code is about 3x faster than in the previous release.

```
function timingHistcounts
X = rand(1,1000);
for k = 1:3e3
    histcounts(X,"BinMethod","auto");
end
end
```

The approximate execution times are:

**R2022b:** 0.62 s

**R2023a:** 0.21 s

The code was timed on a Windows 10, Intel Xeon CPU E5-1650 v4 @ 3.60 GHz test system using the `timeit` function.

```
timeit(@timingHistcounts)
```

## fzero function: Improved performance

The `fzero` function shows improved performance. The performance improvement is most significant when the objective function is fast to compute and `fzero` does not use an options argument.

For example, this code runs about 4x faster than in the previous release.

```
rng default
N = 1e5;
```

```

levels = 1.5 * rand(N,1);
out = zeros(N,1);
myfun = @(x,lv)x*sin(x)-lv;
tic
for i=1:N
    out(i) = fzero(@(x)myfun(x,levels(i)),0);
end
toc

```

The approximate execution times are:

**R2022b:** 2.67 s

**R2023a:** 0.63 s

The code was timed on a Windows 10, Intel Xeon CPU E5-1650 v3 @ 3.5 GHz test system.

## Plots in Apps: Improved performance when rerendering axes

Axes rerender more quickly in R2023a than in R2022b. The reduced rerendering time is most noticeable when many sequential updates to the axes occur.

For example, first create axes to specify as the input to the `timingAnimation` function. Then, call the `timingAnimation` function to add points to an animated line. The time for the axes to rerender is reduced, resulting in an animation that is about 1.35x faster in R2023a than in the previous release.

```

function timingAnimation(ax)
h = animatedline(ax);
x = linspace(0,4*pi,1000);
y = sin(x);
tic
for k = 1:length(x)
    addpoints(h,x(k),y(k));
    drawnow
end
toc
end

```

The approximate durations of the animation are:

**R2022b:** 5.66 s

**R2023a:** 4.20 s

The code was timed on a Windows 10, Intel Xeon CPU E5-1650 v4 @ 3.60 GHz test system by calling the `timingAnimation` function.

## Plots in Apps: Improved performance when creating axes

Axes render more quickly within apps and within figures created with the `uifigure` function in R2023a than in R2022b. The delay before the axes appear in an existing figure is reduced.

For example, if you run the code `uiaxes(f)` for an existing figure `f`, the axes appear about 3.7x faster in R2023a than in R2022b when creating axes for the first time in a MATLAB session and about 2.4x faster for subsequent axes.

The approximate axes rendering times are:

|        | First Axes | Subsequent Axes |
|--------|------------|-----------------|
| R2022b | 14.94 s    | 1.06 s          |
| R2023a | 4.02 s     | 0.44 s          |

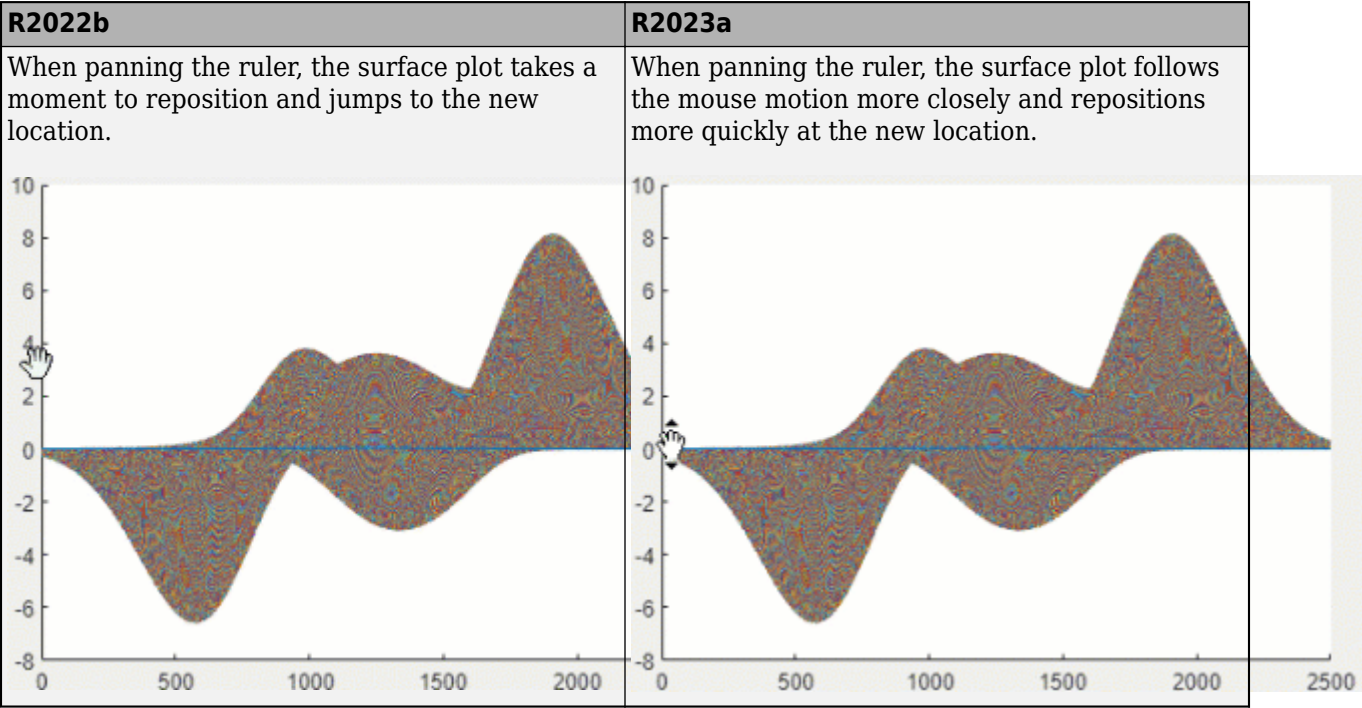
These operations were timed on a Windows 10, Intel Xeon CPU E5-1650 v4 @ 3.60 GHz test system.

Plots in Apps: Improved responsiveness of ruler-pan interaction

The ruler-pan interaction is faster and smoother within apps and within figures created with the `uifigure` function in R2023a than in R2022b. The improvement is most noticeable for plots that display a large number of data points.

For example, this code creates a figure with a plot of a 2500-by-2500 matrix. When you pan the ruler, the ruler-pan interaction is smoother and the axes track your mouse motion more closely in R2023a than in the previous release.

```
f = uifigure;  
ax = uiaxes(f);  
p = peaks(2500);  
plot(ax,p)
```

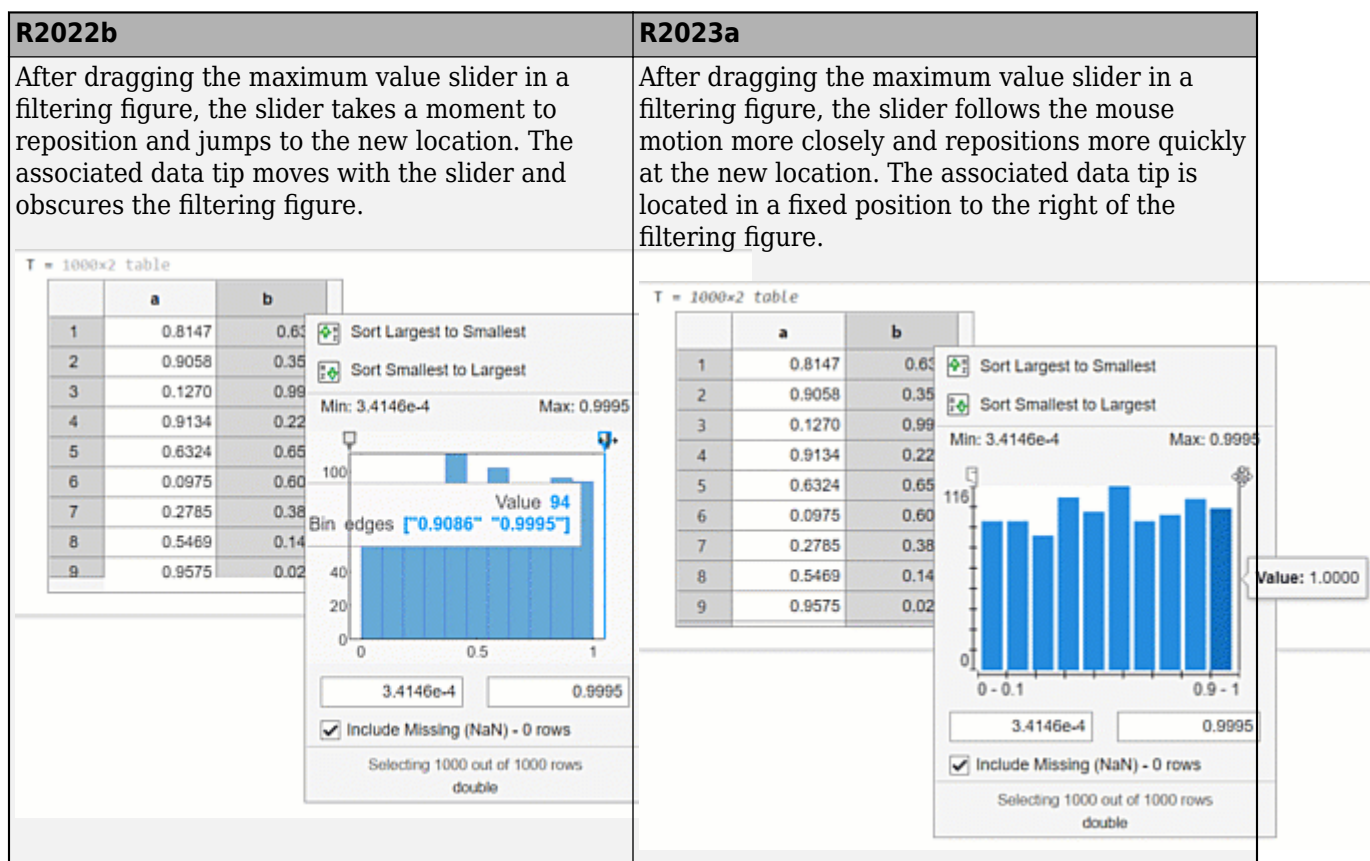


The ruler-pan interaction was performed on a Windows 10, Intel Xeon CPU E5-1650 v4 @ 3.60 GHz test system by running the above script and panning the y-axis ruler.

## Live Editor: Improved performance when filtering numeric table variables

In the output of the Live Editor, the performance of filtering a numeric table or timetable variable is improved. When dragging the maximum or minimum value slider, the drag interaction is smoother and faster in R2023a than in R2022b, and the data tip displaying the current slider value appears to the side of the filtering figure.

For example, for a table output in the Live Editor, pause on the header of a numeric variable and click the triangle icon. Then, to filter the data, adjust the maximum value by dragging the slider. The drag interaction is smoother and tracks your mouse motion more closely in R2023a than in the previous release.



The filtering interaction was performed on a Windows 10, Intel Xeon CPU E5-1650 v4 @ 3.60 GHz test system by dragging the maximum value slider in the filtering figure for a 1000-row numeric table variable.

## Property Inspector: Improved performance when opening for the first time

The Property Inspector shows improved performance when opening for the first time in a MATLAB session. The delay between clicking the Property Inspector icon or calling `inspect` and the inspector being ready is reduced.

For example, open the Property Inspector for the first time in a MATLAB session. You can use the Property Inspector 1.17x sooner than in the previous release.

```
ax = axes;  
inspect(ax)
```

The approximate rendering times are:

**R2022b:** 13.5 s

**R2023a:** 11.5 s

The rendering of the Property Inspector was timed on a Windows 10, Intel Xeon CPU E5-1650 v4 @ 3.60 GHz test system by running the code and measuring the time it takes for the edit fields to appear in the Property Inspector.

## Property Inspector: Improved performance when switching between objects

The Property Inspector shows improved performance when switching between objects. The delay between selecting a different object and an existing instance of the Property Inspector rendering the properties of the newly selected object is reduced.

For example, open the Property Inspector. Then, create and inspect an Axes object. The axes properties render 1.7x faster than in the previous release.

```
inspect  
ax = axes;  
inspect(ax)
```

The approximate times for the Property Inspector to render the properties of the axes are:

**R2022b:** 3.5 s

**R2023a:** 2.1 s

The rendering of the Property Inspector was timed on a Windows 10, Intel Xeon CPU E5-1650 v4 @ 3.60 GHz test system by running the code and measuring the time it takes for the Property Inspector to be ready.

## Variables Editor: Improved performance of cell editing in MATLAB Online

In the MATLAB Online Variables editor, when you interactively edit the value of a cell, the cell updates to display the new value more quickly in R2023a than in R2022b.

For example, create a 1000-element cell array and open the cell array in the Variables editor.

```
C = cell(1000);  
openvar C
```

Then, double-click on a cell and enter a new value. On a Windows 10, Intel Xeon CPU E5-1650 v4 @ 3.60 GHz test system, when you move the focus from the edited cell, the cell value updates immediately. In R2022b, the cell value updates after a 3.5-second delay.

## Variables Editor: Improved speed of data display when scrolling in MATLAB Online

In the MATLAB Online Variables editor, the performance of vertical and horizontal scrolling is improved. When scrolling within 1000 rows below the current element or 100 variables to the right of the current element, the data appears more quickly in R2023a than in R2022b.

For example, create a 1000-by-1000 matrix and open the matrix in the Variables editor.

```
X = rand(1000);
openvar X
```

When you scroll down 100 rows, the values of all visible matrix elements in the Variables editor are rendered about 6.7x faster than in the previous release.

The approximate times for the Variables editor to render the values of all visible matrix elements are:

**R2022b:** 1.80 s

**R2023a:** 0.27 s

This interaction was timed on a Windows 10, Intel Xeon CPU E5-1650 v4 @ 3.60 GHz test system by running the above script and scrolling in the Variables editor.

## App Building: Improved app startup performance

Apps created in App Designer and using the `uifigure` function start up faster in R2023a than in R2022b and previous releases. The improvement is more noticeable for apps with many UI components.

For example, this code creates an app with 1000 edit field components. The code is about 1.5x faster than in the previous release and about 3x faster than in R2021b.

```
function timingApp
fig = uifigure;
gl = uigridlayout(fig,Scrollable="on");
gl.RowHeight = repmat({'fit'},1,100);
gl.ColumnWidth = repmat({'fit'},1,10);

for k = 1:1000
    uieditfield(gl);
end
end
```

The approximate execution times are:

**R2021b:** 21.5 s

**R2022b:** 10.6 s

**R2023a:** 7.2 s

The code was timed on a Windows 10, Intel Xeon CPU E5-1650 v4 @ 3.60 GHz test system by running the `timingApp` function and measuring the time it takes for the edit fields to appear in the UI figure window.

## ▲ App Building: Improved startup performance for apps with multiple tabs

In addition to the overall app startup performance improvement in R2023a, some apps that contain multiple tabs show an even greater startup performance improvement. The reason is that MATLAB prioritizes creating the content in the visible tab over non-visible content when the app first runs.

The particular performance improvement that you see depends on the app layout and UI component types. The improvement is more noticeable for apps with these types of UI components in unselected tabs:

- Labels and spinners with a grid layout manager
- Table UI components without a grid layout manager

For example, this code creates a tab group with five tabs, each containing 200 label components. The code is about 1.9x faster than in the previous release.

```
function timingTabApp
fig = uifigure;
tg = uitabgroup(fig);

for k1 = 1:5
    t = uitab(tg);
    gl = uigridlayout(t,Scrollable="on");
    gl.RowHeight = repmat({'fit'},1,20);
    gl.ColumnWidth = repmat({'fit'},1,10);
    for k2 = 1:200
        uilabel(gl);
    end
end
end
```

The approximate execution times are:

**R2022b:** 9.8 s

**R2023a:** 5.2 s

The code was timed on a Windows 10, Intel Xeon CPU E5-1650 v4 @ 3.60 GHz test system by running the `timingTabApp` function and measuring the time it takes for the components to appear in the UI figure window.

If you have an app with many UI components, consider updating your app layout to take advantage of this improvement. For more information, see [Improve App Startup Time](#).

## ▲ Compatibility Considerations

When an app user switches to a new tab for the first time after running an app, the interaction might take more time than in previous releases. The reason is that MATLAB might create some content in the tab only after the user selects the tab. If the user later switches to the same tab again, the interaction does not take the additional time.



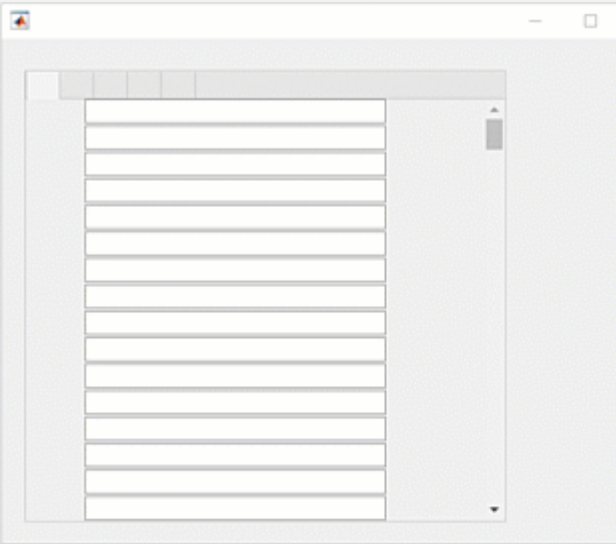
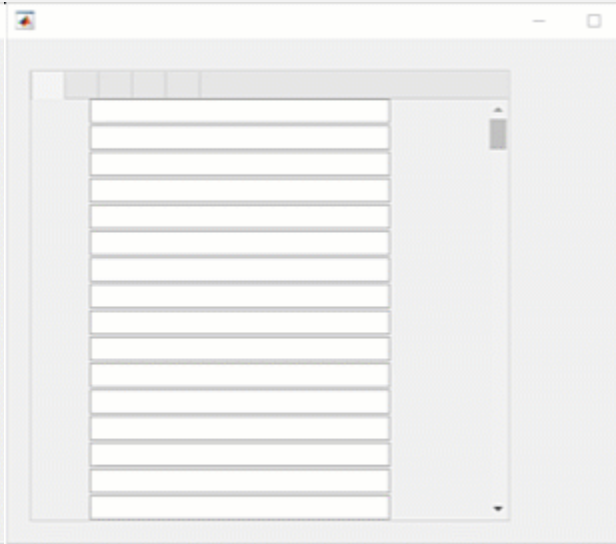
## App Building: Improved performance when resizing some apps

When a user resizes an app figure window, some apps reposition their content faster in R2023a than in R2022b. The types of apps that show this improvement are:

- Large apps with tabs that have an `AutoResizeChildren` value of `'on'`
- Large apps with panels and button groups that have a `SizeChangedFcn` callback

For example, this code creates an app with a tab group where each tab contains many edit fields that are resized automatically. The resize operation is smoother and faster in R2023a than in the previous release.

```
function tabResize
fig = uifigure;
tg = uitabgroup(fig,"Position",[20 20 400 375]);
for k1 = 1:5
    t = uitab(tg,"Scrollable","on");
    for k2 = 1:100
        ef = uieditfield(t,"Position",[50 22*k2 250 20]);
    end
end
end
```

| R2022b                                                                              | R2023a                                                                               |
|-------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------|
| When the app resizes, the app content takes multiple seconds to reposition.         | When the app resizes, the app content takes less than one second to reposition.      |
|  |  |

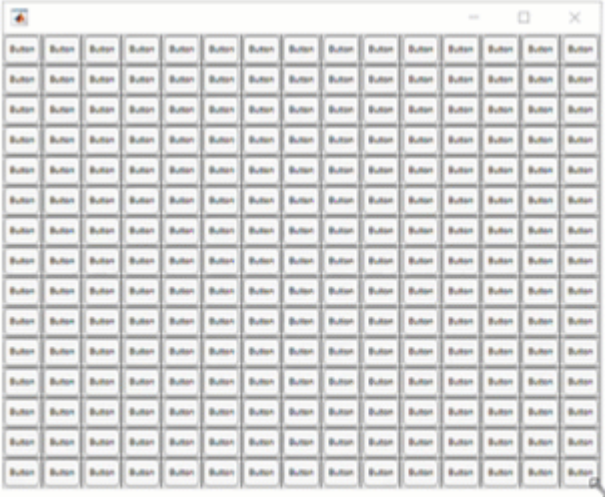
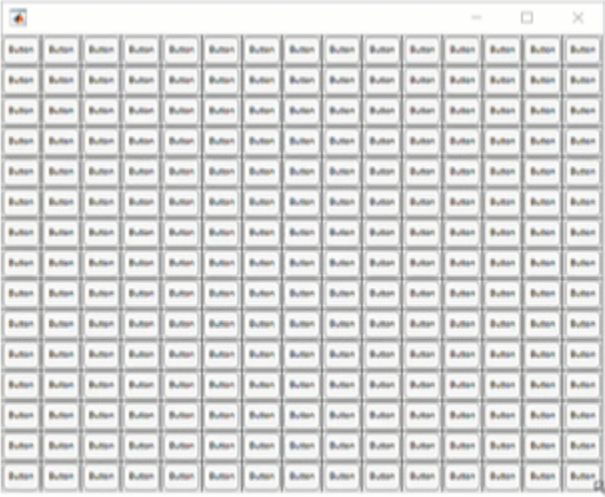
As another example, this code creates an app with many panels, each of which resizes a button using a `SizeChangedFcn` callback whenever the app size changes. The resize operation is smoother and faster in R2023a than in the previous release.

```
function panelResize
fig = uifigure("AutoResizeChildren","off", ...
```

```
"SizeChangedFcn",@resizePanels);
for k = 1:225
    p = uipanel(fig, ...
        "AutoResizeChildren","off", ...
        "SizeChangedFcn",@resizeButtons);
    btn = uibutton(p,"Position",[2 2 20 20]);
end
end

function resizeButtons(src,~)
for k = 1:length(src.Children)
    src.Children(k).Position(3:4) = 0.9*src.Position(3:4);
end
end

function resizePanels(src,~)
xscale = src.Position(3)/15;
yscale = src.Position(4)/15;
for k = 1:length(src.Children)
    p = src.Children(k);
    p.Position = [xscale*(mod((k-1),15)), ...
        yscale*(floor((k-1)/15)), ...
        xscale, ...
        yscale];
end
end
```

| R2022b                                                                              | R2023a                                                                               |
|-------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------|
| When the app resizes, the app content takes multiple seconds to reposition.         | When the app resizes, the app content takes about one second to reposition.          |
|  |  |

The resize interactions were performed on a Windows 10, Intel Xeon CPU E5-1650 v4 @ 3.60 GHz test system by running the `tabResize` and `panelResize` functions and resizing the figure window.

## Software Development Tools

### **Build Automation: Improve build speed and efficiency with incremental builds**

The build tool supports incremental builds. Incremental builds avoid redundant work by skipping tasks that are up to date. If you want the build tool to skip a task when it is up to date, specify the inputs or outputs of the task. The build tool keeps track of the inputs and outputs every time the task runs and skips the task if they have not changed. For more information, see [Improve Performance with Incremental Builds](#).

### **Build Automation: Create and run tasks that accept arguments**

The build tool lets you create and run tasks that accept arguments. You can use task arguments to customize the actions that tasks perform when they run. For more information, see [Create and Run Tasks That Accept Arguments](#).

### **Dependency Analyzer: Analyze files and folders with or without a project**

Starting in R2023a, you can access Dependency Analyzer from the MATLAB apps gallery. You can now perform a dependency analysis on files and folders that do not belong to a project. For more information, see [Dependency Analyzer](#).

### **Project Preferences: Recreate empty project folders in Git repositories**

Git does not track empty folders and ignores them when you commit. MATLAB now enables you to recreate an empty folder structure in a project under Git source control. Doing so is useful for small projects intended for training or as procedure templates.

For large projects, to avoid performance issues on startup, clear **Recreate empty project folders in a project under Git**. For more information, see [Set MATLAB Projects Preferences](#).

### **Project API: Determine if file is under project root folder**

You can now programmatically determine whether a file or a folder is under a project root folder by using the `matlab.project.isUnderProjectRoot` function.

### **Project API: Export subset of project files to archive**

You can now programmatically export a subset of project files to an archive by specifying a user-defined export profile in the `export` function.

## **Project Sharing: Include only specific files in project archive using export profile**

You can now use an export profile to include only files with particular labels in a project archive. This option is useful if the files you need to share are only a small subset of a large project. For more information, see [Create an Export Profile](#).

## **Comparison Tool: Automate comparison report generation for continuous integration (CI) workflows**

Starting in R2023a, you can programmatically publish comparison reports for plain text files, MATLAB scripts, and text-based source code files. Automate report generation for continuous integration (CI) workflows using the `visdiff` function.

```
comparison = visdiff(textfile1,textfile2);  
file = publish(comparison);  
web(file)
```

## **Source Control in MATLAB Online: Save uncommitted changes by creating a Git stash**

In MATLAB Online, you can now save uncommitted changes by creating a Git stash.

## **Source Control in MATLAB Online: Manage Git remote repositories locally using Branch Manager**

In MATLAB Online, you can now manage multiple remote repositories from a local Git repository. Use Branch Manager to perform these tasks:

- Add, edit, and delete remote repositories.
- Fetch from all remotes or individual remotes.
- Prune remote branches from all or individual remotes.
- Open selected remotes in a web browser.
- Create new local branches that track remote branches.
- Delete remote branches.

## **Source Control in MATLAB Online: Detect and extract conflict markers from text and binary files**

In MATLAB Online, you can now detect conflict markers added by Git in text and binary files. Extract conflict markers to repair corrupted files.

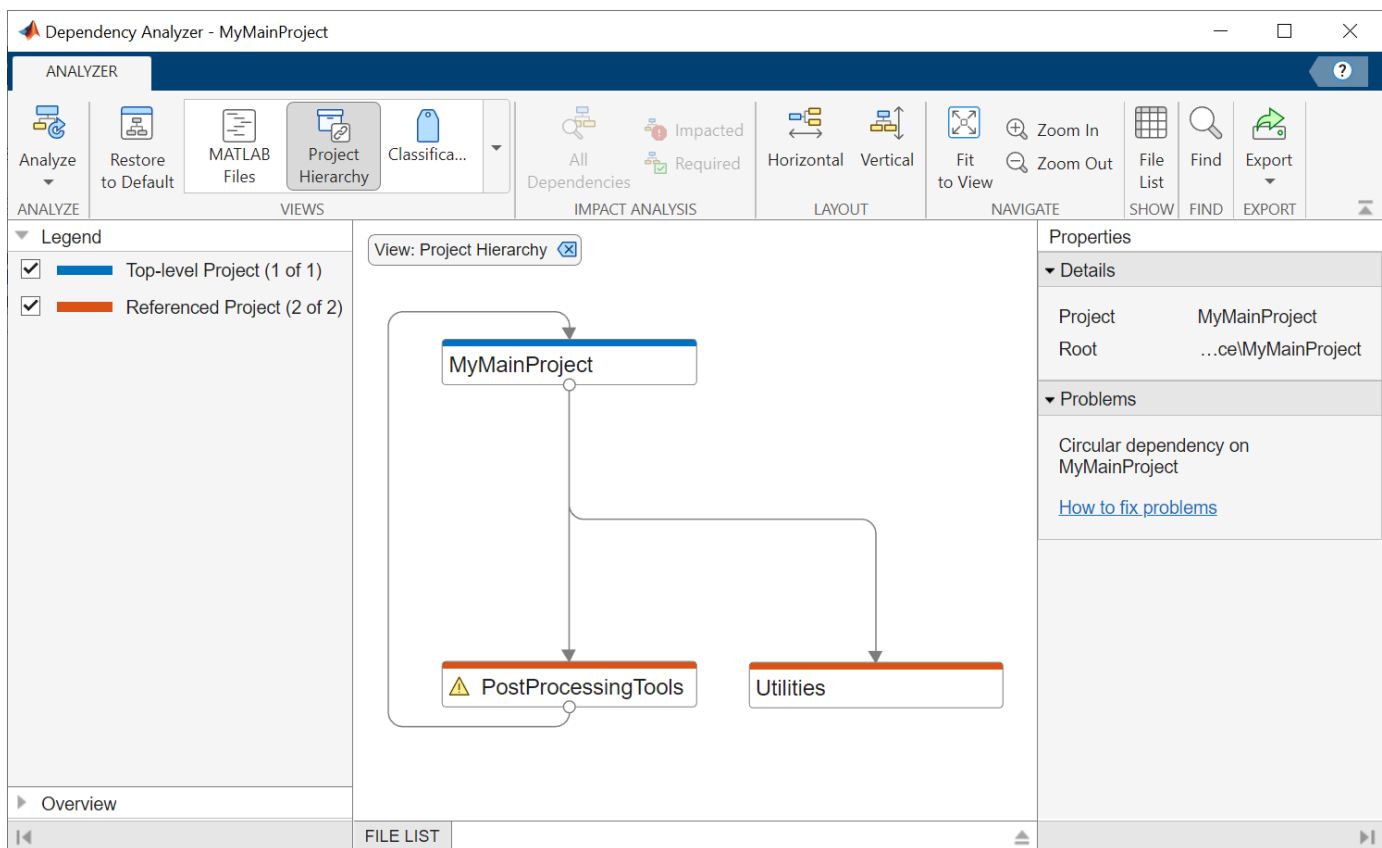
## **Comparison Tool in MATLAB Online: Compare project definition files**

Starting in R2023a, when you compare folders in MATLAB Online, MATLAB detects whether they are project root folders. MATLAB looks for and compares the project definition files stored in the `resources` or `.SimulinkProject` folder. Project definition files contain information about the

project path, project settings, shortcuts, labels, and referenced projects. For more information, see [Compare MATLAB Projects in MATLAB Online](#).

## Dependency Analyzer in MATLAB Online: Investigate circular dependencies using the Project Hierarchy view

You can now investigate how projects in your hierarchy relate to each other and identify projects that introduce circular dependencies using the **Project Hierarchy** view in MATLAB Online. For more information, see [Analyze Project Dependencies](#).



## Unit Testing Framework: Run tests interactively by using Test Browser

The Test Browser app enables you to run script-based, function-based, and class-based tests interactively. You can use the test browser to:

- Create a test suite from files and folders.
- Run all or part of the specified tests.
- Access diagnostics and debug test failures.
- Customize a test run with options, such as running tests in parallel (requires Parallel Computing Toolbox) or specifying a level of test output detail.
- Generate an HTML code coverage report for MATLAB source code.

For more information, see [Run Tests Using Test Browser](#).

## Unit Testing Framework: Programmatically access code coverage results

You can use the `matlab.unittest.plugins.codecoverage.CoverageResult` class to programmatically access the results of code coverage analysis for your source code. To generate and access the coverage results, create a `CodeCoveragePlugin` instance using a `CoverageResult` object, and add the plugin to the test runner. After the test run, the `Result` property of the `CoverageResult` object holds the coverage results as an array of `matlab.coverage.Result` objects. Each element of the array provides information about one of the files in your source code that was covered by the tests.

For more information, see [Collect Statement and Function Coverage Metrics for MATLAB Source Code](#).

## Unit Testing Framework: Temporarily set environment variables

The `matlab.unittest.fixtures.EnvironmentVariableFixture` class provides a fixture for setting an operating system environment variable. Once the testing framework tears down the fixture, the fixture restores the environment variable to its original state.

## Unit Testing Framework: Test for handle validity

The `matlab.unittest.constraints.IsValid` class provides a constraint to test if a handle array is valid. The constraint is satisfied if all array elements are valid handles.

## Unit Testing Framework: Write text to files in thread-based environment

You can use the `ToFile` and `ToUniqueFile` classes in a thread-based environment to write text to UTF-8 encoded files.

## Unit Testing Framework: Use renamed classes in testing and other automated workflows

To reflect support for additional automated workflows, a group of `matlab.unittest` classes have been renamed. For example, `matlab.unittest.Verbosity` is now named `matlab.automation.Verbosity`.

This table shows the affected classes and their new names. The behavior of these classes remains the same, and existing instances of these classes in your code continue to work as expected. There are no plans to remove support for existing instances of these classes.

| R2022b and Earlier                                        | Starting in R2023a                                          |
|-----------------------------------------------------------|-------------------------------------------------------------|
| <code>matlab.unittest.Verbosity</code>                    | <code>matlab.automation.Verbosity</code>                    |
| <code>matlab.unittest.diagnostics.Diagnostic</code>       | <code>matlab.automation.diagnostics.Diagnostic</code>       |
| <code>matlab.unittest.diagnostics.DiagnosticResult</code> | <code>matlab.automation.diagnostics.DiagnosticResult</code> |

| R2022b and Earlier                                                | Starting in R2023a                                                  |
|-------------------------------------------------------------------|---------------------------------------------------------------------|
| <code>matlab.unittest.diagnostics.DisplayDiagnostic</code>        | <code>matlab.automation.diagnostics.DisplayDiagnostic</code>        |
| <code>matlab.unittest.diagnostics.FileArtifact</code>             | <code>matlab.automation.diagnostics.FileArtifact</code>             |
| <code>matlab.unittest.diagnostics.FunctionHandleDiagnostic</code> | <code>matlab.automation.diagnostics.FunctionHandleDiagnostic</code> |
| <code>matlab.unittest.diagnostics.StringDiagnostic</code>         | <code>matlab.automation.diagnostics.StringDiagnostic</code>         |
| <code>matlab.unittest.plugins.OutputStream</code>                 | <code>matlab.automation.streams.OutputStream</code>                 |
| <code>matlab.unittest.plugins.ToFile</code>                       | <code>matlab.automation.streams.ToFile</code>                       |
| <code>matlab.unittest.plugins.ToStandardOutput</code>             | <code>matlab.automation.streams.ToStandardOutput</code>             |
| <code>matlab.unittest.plugins.ToUniqueFile</code>                 | <code>matlab.automation.streams.ToUniqueFile</code>                 |

## App Testing Framework: Test context menus within labels

You can use the `chooseContextMenu` method to test a right-click that opens a context menu within label components. For example, assign a context menu with two items to a label, and then choose the first menu item.

```
fig = uifigure;
lbl = uilabel(fig);

cm = uicontextmenu(fig);
m1 = uimenu(cm,Text="Menu1");
m2 = uimenu(cm,Text="Menu2");
lbl.ContextMenu = cm;

testCase = matlab.uiunit.TestCase.forInteractiveUse;
testCase.chooseContextMenu(lbl,m1)
```

## ⚠ Performance Testing Framework: Use fewer samples to meet the objective margin of error

The default number of times that the framework exercises the test code to warm it up in a frequentist time experiment (created using either the `runperf` function or the `limitingSamplingError` static method) has increased from four to five. This change results in typically fewer samples required to meet the objective relative margin of error.

## ⚠ Compatibility Considerations

If your code relies on the previous default value, you might need to update your code. For example, if you use `warmupTable = results(1).TestActivity(1:4,:)` to create a table of warm-up measurements, replace 4 with 5. Also, if you want to keep using the previous default value, explicitly specify the value in your code. This table shows an example of how to update code that runs tests using four warm-up measurements.

| Before                                                                                                                                          | After                                                                                                                                                  |
|-------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------|
| <pre>import matlab.perftest.TimeExperiment experiment = TimeExperiment.limitingSamplingError(1000000); results = run(experiment,mySuite);</pre> | <pre>import matlab.perftest.TimeExperiment experiment = TimeExperiment.limitingSamplingError("NumOfSamples"); results = run(experiment,mySuite);</pre> |

**⚠️Functionality being removed or changed**

**TaskAction constructor method has been removed**

*Errors*

The constructor method of the `matlab.buildtool.TaskAction` class has been removed. To specify a task action, use a function handle instead of the constructor method. This table shows an example of how to update code that calls the `TaskAction` constructor method.

| Before                                                                                                                                                                                                                           | After                                                                                                                                                                |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <pre>import matlab.buildtool.Task import matlab.buildtool.TaskAction  plan = buildplan; plan("test") = Task( ...     Actions=TaskAction( ...         @(~)assertSuccess(runtests(IncludeSubfolders=true)),Name="Testing"));</pre> | <pre>import matlab.buildtool.Task  plan = buildplan; plan("test") = Task( ...     Actions=@(~)assertSuccess(runtests(IncludeSubfolders=true)),Name="Testing");</pre> |



## External Language Interfaces

### Publish C++ Interface: Publish interface for C++ library in Live Editor

The `clibPublishInterfaceWorkflow` function creates a live script that guides you through the steps to publish a MATLAB interface to a C++ library. For more information, see [Steps to Publish a MATLAB Interface to a C++ Library and Generate C++ Interface](#).

### Interface to C++ Library: Execute C++ library functions out-of-process

Run C++ library functions in processes that are separate from the MATLAB process. For more information, see [Load C++ Library In-Process or Out-of-Process](#). Use out-of-process mode to call functions in third-party libraries that are not compatible with MATLAB. Publishers can use this mode while developing an interface, eliminating the need to restart MATLAB while testing.

### Interface to C++ Library: Support for default arguments

If a C++ function is defined with default arguments, then you can call the function without providing one or more trailing arguments. MATLAB supports default arguments for scalar integer and floating-point types.

The MATLAB interface to C++ libraries displays default arguments in function signatures in the library definition file and in the help text. For example, the argument `arg` for the function `funcname` has a default value of 5.

```
% C++ Signature: void funcname(double arg = 5.000000)
```

These calls to `funcname` produce the same result:

```
clib.libname.funcname
clib.libname.funcname(5)
```

For more information, see [Call Function with Default Arguments](#).

### Interface to C++ Library: Support for comments in function templates

The generated help text in a MATLAB interface to a C++ library includes Doxygen comments from template functions and template methods of a class. For information about viewing these comments, see [Display Help for MATLAB Interface to C++ Library](#). For information about modifying comments when publishing an interface, see [Publish Help Text for MATLAB Interface to C++ Library](#).

### Publish C++ Interface: Put libraries on run-time path

Use the `copyRuntimeDependencies` function to copy dependent libraries to the run-time path so that they are available when you test. This function collects the necessary files for distributing to end users so that they do not have to set environment variables to call functions in the library. For information, see [Set Up and Copy Run-Time Libraries](#).

## Publish C++ Interface: Resolve multiple redefinition and unresolved external symbol errors

Error messages for multiple redefinition and unresolved external symbol compiler errors provide additional help for you to resolve the error. For more information, see [Resolve Build Error: Multiple Redefinition Linker Errors](#) and [Resolve Build Error: Unresolved External Symbols](#).

## Publish C++ Interface: Information to debug C++ library functions

You can debug C++ library functions by using a debug-version of the MATLAB interface to the library. For steps to build a debug-version, see [Debug C++ Library from MATLAB Interface](#).

## Publish C++ Interface: Support for MATLAB operators for C++ methods

In a MATLAB interface to a C++ library, MATLAB operators are supported when corresponding methods are defined in a C++ class.

| Operation in MATLAB    | Method to Define in C++ Class |
|------------------------|-------------------------------|
| <code>a &lt; b</code>  | <code>lt(a,b)</code>          |
| <code>a &gt; b</code>  | <code>gt(a,b)</code>          |
| <code>a &lt;= b</code> | <code>le(a,b)</code>          |
| <code>a &gt;= b</code> | <code>ge(a,b)</code>          |
| <code>a ~= b</code>    | <code>ne(a,b)</code>          |
| <code>a == b</code>    | <code>eq(a,b)</code>          |

## Java Interface: Support for Java 11 JDK and JRE

MATLAB supports Java 11 JDK™ and JRE. To use this version, located in the folder `jre_path`, call the `jenv` function. At the MATLAB prompt, type:

```
e = jenv("jre_path")
```

You might have to restart your MATLAB session to change to this version.

## Java Interface: `jenv` and `matlab_jenv` provide environment information

The `jenv` function returns a `JavaEnvironment` object, which contains information about the Java program on your system. The `matlab_jenv` command displays Java environment information at the operating system prompt, but it does not return the information.

## Python Interface: Convert between MATLAB datetime and Python datetime, NumPy datetime64 types

You can convert between MATLAB datetime values and Python datetime or NumPy datetime64 values. For examples, see [Use MATLAB datetime Types with Python](#).

## **Python Interface: Convert between MATLAB duration and Python timedelta, NumPy timedelta64 types**

You can convert between MATLAB duration values and Python `timedelta` or NumPy `timedelta64` values. For examples, see [Use MATLAB duration Types with Python](#).

## **Python Objects: Use Python objects as keys in dictionary**

You can use Python objects as keys in dictionaries. For more information about dictionary keys, see [dictionary](#).

## **.NET Interface: Convert between MATLAB dictionary and .NET System.Collections.Generic.Dictionary objects**

You can convert a MATLAB dictionary to a .NET dictionary. For more information, see [Pass Data to .NET Objects](#).

To explicitly create a .NET dictionary from a MATLAB dictionary, call the `NET.createDictionary` function.

To convert a .NET dictionary to a MATLAB dictionary, see [How MATLAB Handles .NET Dictionary Objects](#).

## **.NET Objects: Use .NET objects as keys in dictionary**

You can use .NET objects as keys or values in dictionaries. For more information about dictionary keys, see [dictionary](#).

## **.NET Engine: Support for MATLAB structs**

Use the `.NET MathWorks.MATLAB.Types.MATLABStruct` class to represent MATLAB struct objects. For examples, see [Use MATLAB Structures in .NET](#).

## **▲Perl 5.36.0: MATLAB support on Windows**

As of R2023a, MATLAB on Windows ships with an updated version of Perl, version 5.36.0, and supports an updated version of `HTML::Parser`, version 3.78.

## **▲Compatibility Considerations**

If you use the `perl` command on Windows platforms, see <https://www.perl.org/> for information about using this version of the Perl programming language. For a standard distribution of `HTML::Parser`, source code, and information about using `HTML::Parser`, see <https://metacpan.org/pod/HTML::Parser>.

## ⚠ Compiler support changed for building C and C++ interfaces, MEX files, and standalone MATLAB engine and MAT-file applications

| Support          | Compiler                                                                                                                                  | Platform         |
|------------------|-------------------------------------------------------------------------------------------------------------------------------------------|------------------|
| Added            | Intel oneAPI 2023 with Microsoft Visual Studio 2019 and 2022 for C, C++, and Fortran                                                      | Windows<br>macOS |
| Added            | Intel oneAPI 2022 with Microsoft Visual Studio 2017, 2019, and 2022 for C, C++, and Fortran                                               | Windows          |
| Added            | MinGW-w64 version 8.1 compiler. For installation instructions, see this MATLAB Answers™ article FAQ: How do I install the MinGW compiler? | Windows          |
| To be phased out | Intel Parallel Studio XE for C/C++                                                                                                        | Windows          |

For continued support for building your applications, consider upgrading to a supported compiler. For an up-to-date list of supported compilers, see [Supported and Compatible Compilers](#).

## ⚠ Functionality being removed or changed

### Python version 2.7 is no longer supported

#### Errors

Support for Python version 2.7 is discontinued. For continued support for your applications, upgrade to a supported version of Python. For supported version information, see [Versions of Python Compatible with MATLAB Products by Release](#).

### -std=c++11 flag is no longer included in MEX options files

#### Behavior change

MEX options files for building C++ code with MinGW and Linux compilers no longer include the CXXFLAGS option `-std=c++11`.

MEX options files for macOS compilers include the CXXFLAGS option `-std=c++14` instead.

However, you can still build MEX files with the `-std=c++11` option. For example, build the MEX file `myFunc.cpp` with that option.

```
mex myFunc.cpp 'CXXFLAGS=$CXXFLAGS -std=c++11'
```

### Continuous console and writer output for MATLAB API for Java functions

#### Behavior change

When you call a MATLAB function that displays output using one of the evaluate functions in `com.mathworks.engine.MatlabEngine`, then the output continuously displays on the console and writer output stream. This behavior applies to `feval`, `fevalAsync`, `eval`, and `evalAsync`.

Before R2023a, the output displays after the MATLAB function completes.

### .NET Interface: Dictionary with cell types map to .NET System.Object instead of System.Object[]

#### Behavior change

A MATLAB dictionary with entries of type `cell` are converted to a .NET dictionary with entries of type `System.Object`. Previously, the conversion was to `System.Object[]`. For more information, see [Pass Data to .NET Objects and How MATLAB Handles .NET Dictionary Objects](#).

## Hardware Support

### **Support for MJPEG format in USB Webcams on Windows**

Starting R2023a, the MATLAB Support Package for USB Webcams installed on a Windows machine supports the MJPEG image format. The MJPEG format provides improved frame rates for acquiring high-resolution images from a USB webcam.

### **Support for 32-bit Debian Bullseye on Raspberry Pi**

MATLAB Support Package for Raspberry Pi Hardware now supports 32-bit Debian Bullseye for MATLAB desktop and MATLAB Online. You can now customize Debian Bullseye running on your Raspberry Pi hardware to make it compatible with MATLAB while doing the Hardware Setup.

### **Support for Raspberry Pi Zero 2 W and Raspberry Pi Compute Module 4**

You can now use the MATLAB Support Package for Raspberry Pi Hardware with Raspberry Pi Zero 2 W and Raspberry Pi Compute Module 4 for MATLAB desktop and MATLAB Online.

# R2022b

---

**Version: 9.13**

**New Features**

**Bug Fixes**

**Compatibility Considerations**

## Environment

### Live Editor Controls: Add numeric spinners to increment and decrement variable values in live scripts

You can add numeric spinners to your live scripts to interactively increment and decrement the value of variables. To add a numeric spinner, go to the **Live Editor** tab, and in the **Code** section, click  **Control**. Then, select **Numeric Spinner**.



For more information, see [Add Interactive Controls to a Live Script](#).

### Live Editor Tasks: Specify what code to run when control value changes

By default, when you change a parameter in a Live Editor task, the Live Editor runs the section that contains the task. You can now configure a Live Editor task to run all sections, run the current section and all modified sections above that section, run the current section and all remaining sections, or to do nothing.

To configure the task, click the Options button (  ) in the top-right corner of the task. Select **Autorun** and then select from the available options. For more information, see [Add Interactive Tasks to a Live Script](#).

### Suggestions and Completions: Change when to show suggestions and how to accept them

Starting in R2022b, you can use **Ctrl+Space** to show code suggestions in the Editor, Live Editor, App Designer, and MATLAB Online Command Window. In addition, you can change when to show suggestions and how to accept them. On the **Home** tab, in the **Environment** section, click  **Preferences**. Select **Editor/Debugger > Automatic Completions** and in the **Suggestions and completions** section, adjust one or more of the options:

- To disable showing suggestions automatically, clear the **Show suggestions automatically** option.
- To disable showing suggestions after you press the **Tab** key, clear the **Use tab to show suggestions (in addition to ctrl+space)** option. When this option is cleared, you can still show suggestions by typing **Ctrl+Space**.
- To disable accepting a suggestion using the **Right Arrow** key, clear the **Use right arrow to accept suggestions (in addition to tab and enter)** option. When this option is cleared, you can still accept a suggestion using the **Tab** and **Enter** keys.

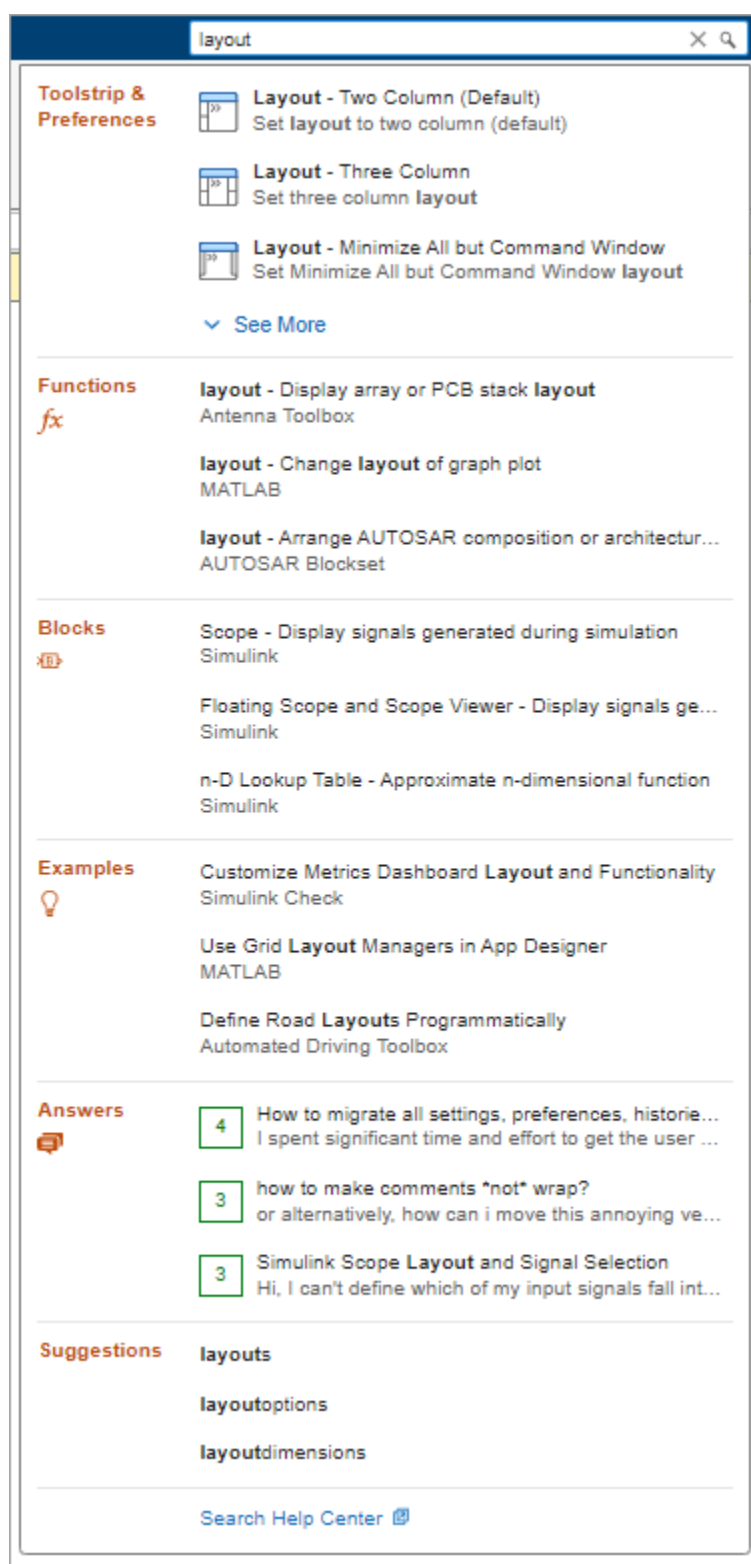
For more information, see [Code Suggestions and Completions](#).



## Search in MATLAB Online: Access toolbar actions, preferences, and Help Center resources using search

In MATLAB Online, you can now use the Search box in the top-right corner of the desktop to easily access toolbar actions, preferences, and Help Center resources such as functions, blocks, examples, and answers. To navigate to the Search box using a keyboard, press **Ctrl+Shift+Space** (or **Command+Shift+Space** on macOS systems).

For example, to restore the MATLAB desktop to its default layout, enter the word `layout` in the Search box. In the list of results, select the **Layout - Two Column (Default)** toolbar action.



## System Theme in MATLAB Online: Change colors of MATLAB desktop to match operating system color scheme

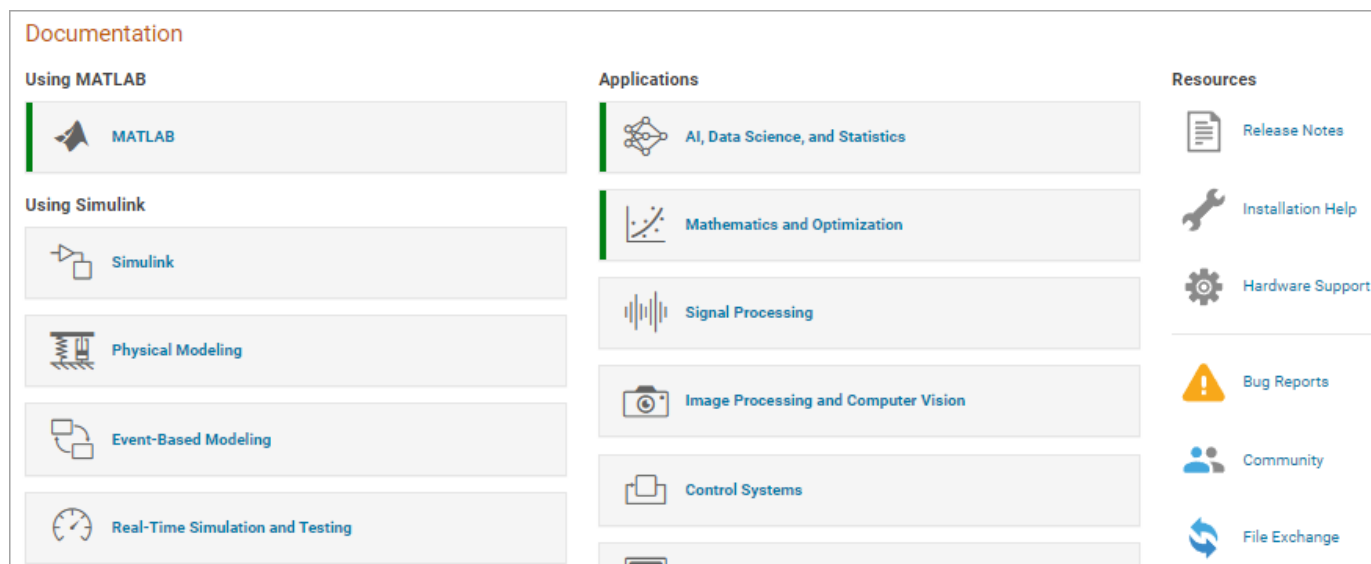
Windows and macOS systems support light and dark color schemes. Starting in R2022b, MATLAB Online is configured to match the current operating system color scheme by default using the system theme. If the selected color scheme for the operating system changes, the colors of the MATLAB desktop change to match the new scheme.

To change the selected theme, on the **Home** tab, in the **Environment** section, click  **Preferences**. Select **MATLAB > Appearance** and select a theme from the **Theme** field. For more information, see [Change Desktop Colors](#).

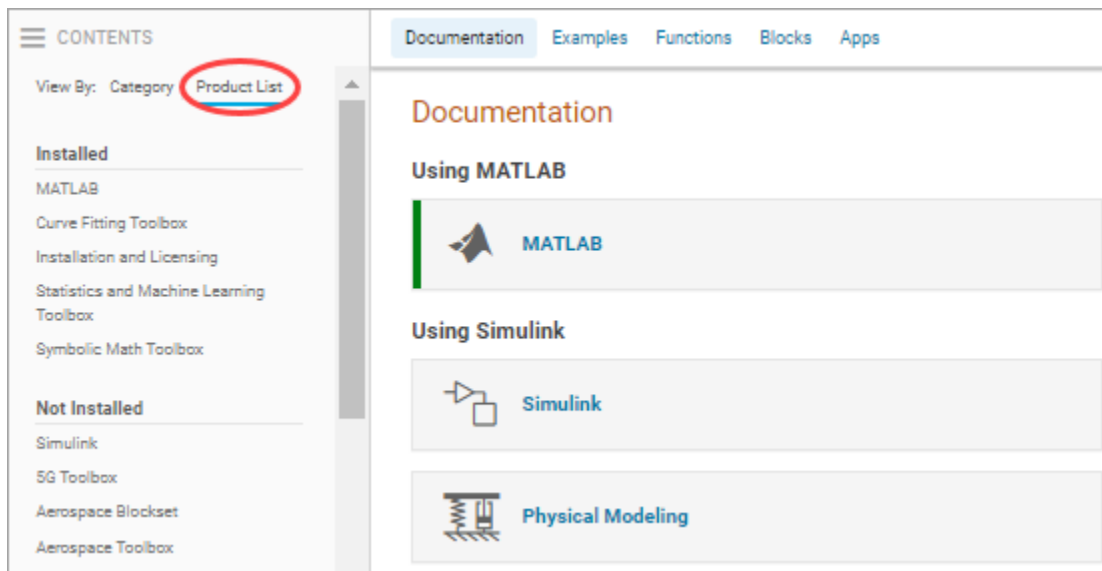
If you customize the colors of the MATLAB desktop and then the MATLAB desktop theme changes to a new theme or to match a change to the operating system color scheme, your customizations will be lost. To avoid unexpectedly losing color customizations when the operating system color scheme changes, select the **Light** or **Dark** option from the **Theme** field before making any color customizations.

## Help Center: View documentation for all products and determine which products are installed

When viewing the documentation from an installed version of MATLAB, you can now view the documentation for all products. Previously, you could view only the documentation for installed products. The Help browser indicates installed products using a green bar to the left of the product name or its category.



To view the list of installed products, select **Product List** at the top left of the Help browser. The Help browser displays the list of products that are installed for the current version of MATLAB. If MATLAB is configured to display the web documentation (default), the Help browser also displays the list of products that are not installed.



## unsetenv Function: Remove environment variable

You can remove an environment variable from the operating system by using the `unsetenv` function.

## isenv Function: Determine if environment variable exists

To determine if an environment variable exists in the operating system, you can use the `isenv` function.

## Comparison Tool: Compare MAT files, FIG files, live scripts, and live functions in MATLAB Online

Starting in R2022b, you can compare MAT and FIG files in MATLAB Online. You also can compare live scripts and functions.

You can access the comparison tool from:

- The MATLAB Current Folder browser context menu
- The Current Project browser context menu
- The MATLAB Command Window by using the `visdiff` function

## ⚠ Functionality being removed or changed

**`matlab.keyboard.ShowAutomatically` and `matlab.commandwindow.ShowCompletionsAutomatically` settings have been removed**  
*Errors*

The `matlab.keyboard.ShowAutomatically` and `matlab.commandwindow.ShowCompletionsAutomatically` settings have been removed. Use the `matlab.editor.suggestions.ShowAutomatically` and `matlab.commandwindow.suggestions.ShowAutomatically` settings instead.

To update your code, change instances of the setting `matlab.keyboard.ShowAutomatically` to `matlab.editor.suggestions.ShowAutomatically` and instances of the setting `matlab.commandwindow.ShowCompletionsAutomatically` to `matlab.commandwindow.suggestions.ShowAutomatically`. For more information, see `matlab.editor` and `matlab.commandwindow`.

### **lookfor function no longer searches third-party and user-authored help text**

#### *Behavior change*

The `lookfor` function no longer searches help text in third-party and user-authored MATLAB program files. In previous releases, `lookfor` searches all MATLAB program files on the path, including third-party and user-authored program files.

To search the documentation including third-party and custom documentation, use the `docsearch` function instead. Alternatively, you can use the find features in the Current Folder browser. For example, you can look for all occurrences of a specified word in all the MATLAB program files in the current folder and its subfolders. For more information, see [Find Files and Folders](#).

## Language and Programming

### dictionary Object: Map unique keys to values for fast lookup

Dictionaries are useful for fast lookup of values in a large data set. A dictionary stores data as values, which can be accessed using corresponding unique keys. For example, create a dictionary that contains three key-value pairs.

```
d =  
  
    dictionary (string → double) with 3 entries:  
  
    "Unicycle" → 1  
    "Bicycle"  → 2  
    "Tricycle" → 3
```

Use a key to look up the corresponding value in a dictionary.

```
d("Bicycle")  
  
ans =  
  
    2
```

In almost all use cases, `dictionary` performs faster than `containers.Map`. For more information, see `dictionary`.

### Output Argument Validation: Validate output arguments in functions and class methods

You can now define output argument blocks for functions and class methods. Use many of the available argument validation techniques on output arguments, including size and data type restrictions as well as validation functions. Input validation checks that function users provide valid inputs. Output validation helps function authors ensure that their functions return the expected types of outputs. It also helps identify errors in the function definition that cause unexpected outputs.

For example, `realPolyRoots` is based on the `roots` function. The input argument block restricts the degree of the input polynomial to a quadratic. The output arguments block only allows the function to return results when the roots are real valued.

```
function r = realPolyRoots(p)  
    arguments (Input)  
        p (1,3)  
    end  
  
    arguments (Output)  
        r {mustBeReal}  
    end  
  
    r = roots(p);  
end
```

Calling `realPolyRoots` on  $x^2 - 1$  (`p = [1 0 -1]`) returns roots 1 and -1, but calling the function on  $x^2 + 1$  errors.

```
realPolyRoots([1 0 1])
```

```
Invalid output 'r'. Value must be real.
```

```
Error in realPolyRoots (line 11)
end
```

For more information, see [Function Argument Validation](#).

## Code Analyzer App: Identify code issues

The MATLAB Code Analyzer is now available as an app. Use the Code Analyzer app to identify issues in code and implement suggested fixes when possible.

## codeIssues Object: Identify and store code issues from one or more files

Create a `codeIssues` object to identify and store code issues found within specified files. When displayed in the Command Window, the `codeIssues` object shows the location of identified issues in the code.

## Custom Compact Display: Live Editor, Variables editor, and Workspace browser follow CustomCompactDisplayProvider implementations

Starting in R2022b, the Live Editor, the Variables editor in MATLAB Online, and the single-line Value field in the Workspace browser in MATLAB Online follow rules for customized object display implemented using the `matlab.mixin.CustomCompactDisplayProvider` class. For more information on custom compact displays, see [Custom Compact Display Interface](#).

## Customized Dot Indexing: New methods handle parentheses-dot combinations

The `matlab.mixin.indexing.RedefinesDot` class has two new concrete methods—`parenDotAssign` and `parenDotListLength`—that handle built-in parentheses indexing immediately followed by customized dot indexing, such as `obj(idx).prop = val`. The two methods have default implementations that provide the expected behavior, but you can override them if needed.

## ⚠️ Functionality being removed or changed

### **isequal and isequaln return false when comparing empty arrays of the same size but of different classes**

*Behavior change*

Previously, `isequal` and `isequaln` returned true when comparing two or more empty object arrays of the same size, regardless of the class of the objects. Starting in R2022b, both functions return false when comparing empty object arrays of the same size but different classes.

### **Defining classes and packages: Using `schema.m` will not be supported in a future release**

*Still runs*

Support for classes and packages defined using `schema.m` files will be removed in a future release. Replace existing schema-based classes with classes defined using the `classdef` keyword.

### **Multiple repeating arguments blocks throw an error on empty blocks**

*Behavior change*

If a function contains multiple arguments blocks with the `Repeating` property, MATLAB now throws an error regardless of if the blocks are empty or have declared repeating arguments. Previously, MATLAB threw an error only if there was more than one repeating block with repeating arguments specified.

### **Multiple repeating arguments validate individual elements of all repeating arrays together**

*Behavior change*

If a function contains multiple arguments with the `Repeating` property, argument validation functions that accept multiple inputs will now make element-wise comparisons. For example, this function uses a validation function with multiple repeating arguments as inputs.

```
function MyFoo(x,y)
    arguments(Repeating)
        x
        y {mustBeGreaterThan(x,y)}
    end
end
```

The validation function `mustBeGreaterThan(x,y)`, validates each single element of `y` against a single corresponding element of `x`. Previously, `mustBeGreaterThan(x,y)` validated a single element of `y` against the entire cell array of repeating values of `x`.

### **Dynamic Regular Expressions: Commands in replacement expressions now check only the local workspace for variables**

*Behavior change*

Commands in replacement expressions using the syntax `${cmd}` will only check the local workspace for variables. Caller and global workspaces will not be available to commands in replacement expressions. This means that commands in replacement expressions now work in `parfor` loops.



## Data Analysis

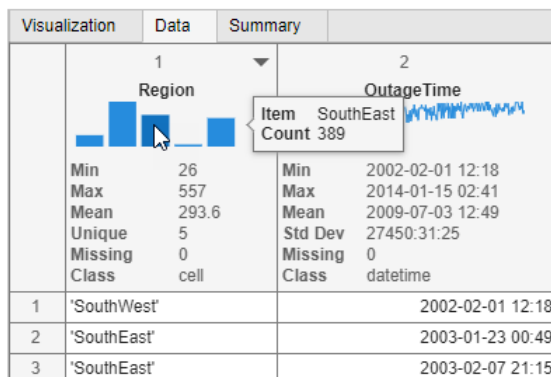
### Data Cleaner App: Clean data in table

The Data Cleaner app supports importing and cleaning data in a table. You can access a table in the MATLAB workspace or import a table from a file. Previously, the app cleaned only timetable data.

You can open the Data Cleaner app from the **MATLAB** section of the apps gallery in the **Apps** tab. Alternatively, enter `dataCleaner` in the MATLAB Command Window.

### Data Cleaner App: View sparklines and summary statistics

The **Data** view in the Data Cleaner app displays sparklines and summary statistics to quickly visualize and interpret the data in each table or timetable variable. Show more information related to specific points by pointing to a sparkline.



### stackedplot Function: Plot data from multiple tables or timetables

The `stackedplot` function can now plot variables from multiple input tables or timetables. In previous releases, `stackedplot` can plot variables only from a single table or timetable.

For example, to plot data from the two sample timetables `indoors` and `outdoors`, load the two MAT-files and then call `stackedplot`.

```
load indoors.mat
load outdoors.mat
stackedplot(indoors,outdoors)
```

If you specify multiple inputs, then they must be either all tables or all timetables.

To work with multiple inputs, `stackedplot` has new name-value arguments:

- **CombineMatchingNames** — If `true`, then plot variables from different inputs but with the same names in the same y-axis. If `false`, then plot them in different y-axes.
- **LegendLabels** — If specified, then create a legend with the specified labels instead of the names of the input tables or timetables.
- **LegendVisible** — If `"on"`, then display the legend. If `"off"`, then hide the legend.

- **LegendOrientation** — If "horizontal", then display the legend horizontally. If "vertical", then display the legend vertically.

## **detrend Function: Detrend tabular data**

The `detrend` function can detrend data in a table or timetable.

When detrending table or timetable data, you can:

- Specify tabular variables to detrend by using the `DataVariables` name-value argument.
- Append or replace tabular variables with variables containing detrended data by using the `ReplaceValues` name-value argument.
- Specify the sample points as a table variable by using the `SamplePoints` name-value argument. `SamplePoints` is not supported when the input data is a timetable.

## **rmoutliers Function: Define outlier locations, and optionally return outlier indicator, thresholds, and center value**

When using the `rmoutliers` function, you can define the location of outliers in the input data with a known outlier indicator. Define outlier locations, rather than using an outlier detection method, by setting the `OutlierLocations` name-value argument to a logical array the same size as the input data.

Regardless of whether the outliers are defined or detected, the `rmoutliers` function can return a logical outlier indicator array indicating the locations of outliers in the input data.

`rmoutliers` can also return the lower threshold value, upper threshold value, and center value used by the outlier detection method. The code generated by the Clean Outlier Data task in the Live Editor includes the new output arguments.

## **mape and rmse Functions: Calculate error between forecast and actual data**

Calculate the mean absolute percentage error and the root-mean-square error between arrays by using the `mape` and `rmse` functions, respectively. You can specify the dimensions to operate along, whether to include or omit zero values (`mape` only) or NaN values from the calculation, or a weighting scheme.

## **isuniform Function: Determine if vector is uniformly spaced**

Determine if a numeric vector is uniformly spaced up to round-off tolerance by using the `isuniform` function. You can optionally return the step size between consecutive elements.

## **⚠Serial date numbers and date strings are not recommended**

To represent dates and times in MATLAB, use the `datetime`, `duration`, and `calendarDuration` data types. The `datetime` data type represents points in time, and the `duration` and `calendarDuration` data types represent lengths of time. In particular, the `datetime` data type provides flexible date and time formats, storage out to nanosecond precision, and properties to account for time zones and daylight saving time.

## ⚠ Compatibility Considerations

Serial date numbers and date strings are not recommended. As a consequence, MATLAB functions that create or modify serial date numbers or date strings, such the `datenum` and `datestr` functions, are not recommended.

- For more information on MATLAB date and time functions that are no longer recommended, see [Functionality being removed or changed on page 7-16](#).
- For more information on replacing these functions in your code, see [Replace Discouraged Instances of Serial Date Numbers and Date Strings](#).

MATLAB functions that accept serial date numbers or date strings as inputs also accept `datetime` arrays as inputs.

There are no plans to remove support for serial date numbers or date strings.

## **datetime and convertTo Functions: Convert between CDF TT2000 times and datetime arrays**

The NASA Common Data Format (CDF) defines the `CDF_TIME_TT2000` (or `TT2000` for short) data type as a high-resolution time type that handles leap seconds for data in CDF files. `TT2000` times represent the number of nanoseconds that have elapsed since J2000.

You can convert `int64` arrays that represent `TT2000` times to `datetime` arrays. To convert `TT2000` times, call the `datetime` function with `ConvertFrom` set to `"tt2000"` and `TimeZone` set to `"UTCLeapSeconds"`. To account for leap seconds, you must specify the time zone of the output `datetime` array.

```
X = int64([702077514184000000; ...
           702163914184000000; ...
           702250314184000000])
D = datetime(X,"ConvertFrom","tt2000","TimeZone","UTCLeapSeconds")
```

Also, you can convert `datetime` arrays to `int64` arrays that represent `TT2000` times. To convert `datetime` arrays, call the `convertTo` function using the `"tt2000"` option. To account for leap seconds, the input `datetime` array must have its time zone set to `"UTCLeapSeconds"`.

```
D = datetime(2022,4,1:3,9,30,45, ...
            "TimeZone","UTCLeapSeconds")
X = convertTo(D,"tt2000")
```

## **Data Preprocessing Live Editor Tasks: Plot multiple table variables**

These Live Editor tasks can plot multiple table variables:

- Clean Missing Data
- Clean Outlier Data
- Find Change Points
- Find Local Extrema
- Normalize Data
- Smooth Data

- Remove Trends

For table or timetable data, to visualize all selected table variables at once in a tiled chart layout, set the **Variable to display** field.

## Data Preprocessing Live Editor Tasks: Append cleaned table variables and specify logical tabular output

These Live Editor tasks can append cleaned table variables to the input table variables:

- Clean Missing Data
- Clean Outlier Data
- Normalize Data
- Smooth Data
- Remove Trends

Additionally, in these Live Editor tasks, you can specify to output a table with logical variables:

- Find Change Points
- Find Local Extrema

To append cleaned table variables or specify a tabular output with logical variables, set the **Output format** field.

## Clean Missing Data Live Editor Task: Plot nonnumeric table data, specify minimum number of missing entries, and use custom fill method

The Clean Missing Data task in the Live Editor can now plot nonnumeric data. To display a categorical histogram, select a nonnumeric input array or set the **Variable to display** field to a nonnumeric table variable containing categorical, string, cellstr, calendarDuration, or char data types.

You can specify a minimum number of missing entries required to remove a row of data. First, select the **Remove missing** cleaning method, and then specify the minimum number of missing entries by using the **Min missing for removal** field.

You can also specify a custom fill method for filling missing data. Select the **Fill missing** cleaning method, and then specify a custom fill method by selecting the **Custom function** cleaning method parameter and the local function or function handle option.

## Clean Outlier Data Live Editor Task: Convert outliers to missing

Use the Clean Outlier Data task in the Live Editor to convert outlier data to missing data indicated by the value NaN. Convert outlier data to missing data by setting the **Cleaning method** field to **Fill outliers** and selecting the **Convert to missing** option.

## Live Editor and Variables Editor: Control placement of missing values when sorting numeric data in MATLAB Online

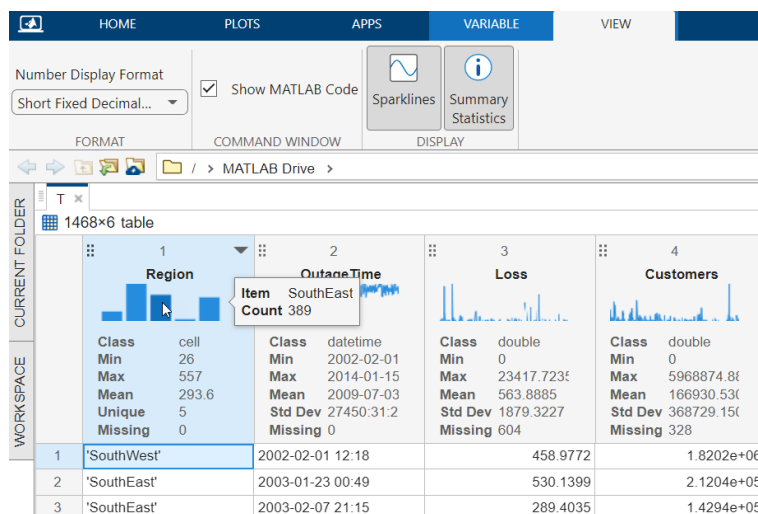
In the Live Editor output and Variables editor in MATLAB Online, you can control the placement of missing values (NaN, NaT, <undefined>, and missing) when sorting rows of numeric data.

For example, to place missing elements first, on the **Home** tab, in the **Environment** section, click  **Preferences**. Select **MATLAB > Variables** and set **Placement of missing values** to **first**.

In the Variables editor, you can also control missing value placement using the **Variable** tab. In the **Transform** section, from the **Sort** list, select an option for **Missing Placement**.

## Variables Editor: View sparklines and summary statistics for tabular data in MATLAB Online

The Variables editor in MATLAB Online now displays sparklines and summary statistics to quickly visualize and interpret the data in each table or timetable variable. Show more information related to specific points by pointing to a sparkline. Show or hide the sparklines or summary statistics using the **Display** section in the **View** tab.



## Variables Editor: Navigate variable with Find and Go To dialog boxes in MATLAB Online

In the Variables editor in MATLAB Online, search for any text in the current variable or selection. On the **Variable** tab, in the **Navigate** section, click **Find**. You can also use the **Ctrl+F** keyboard shortcut. In the Find dialog box, enter the text that you want to search for and use the arrow buttons to search backward or forward through the current variable. You can also change how the Variables editor searches for text by selecting the match case, whole word, or regular expression search options.

Additionally, in the Variables editor in MATLAB Online, navigate to a particular location in the current variable. On the **Variable** tab, in the **Navigate** section, click **Go To**. You can also use the **Ctrl+G** keyboard shortcut. In the Go To dialog box, specify with numeric or text indices the row and column that you want to navigate to.

## Variables Editor: Interactively save logical indices of data selection in MATLAB Online

In the Variables editor in MATLAB Online, create a new logical variable indicating the indices of the selected data by right-clicking the selection or by clicking **New from Selection** in the **Variable** tab.

For tabular data:

- Append a logical table variable indicating the selection
- Create a logical column vector in the workspace indicating the selected rows
- Create a logical row vector in the workspace indicating the selected columns

For all other data types except for structure arrays and objects:

- Create a logical matrix in the workspace indicating the selection
- Create a logical column vector in the workspace indicating the selected rows
- Create a logical row vector in the workspace indicating the selected columns

## Variables Editor: Replace tabular data with empty arrays in MATLAB Online

In the Variables editor in MATLAB Online, replace the current selection of table or timetable elements with empty 0-by-0 arrays by pressing the **Delete** key or by right-clicking the selection and selecting **Replace with Empty**. The class of an empty array is determined by the class of the table variable.

Previously, you could interactively replace the current data selection with empty arrays for nontabular data only.

### Functionality being removed or changed

#### Character arrays have no standard missing value

*Behavior change*

For these functions, character arrays have no default definition of a standard missing value:

- `ismissing`
- `rmmissing`
- `fillmissing`
- `standardizeMissing`
- `mustBeNonmissing`
- `anymissing`
- `nummissing` method of `groupsummary`

Blank char elements ( ' ') are treated as nonmissing. For example, `ismissing(['a'; ' '])` returns logical array `[0; 0]`. In previous releases, it returns `[0; 1]`.

To treat blank character array elements as missing, use an indicator. For example, `ismissing(['a'; ' '], ' ')` specifies a blank char as a missing value and returns `[0; 1]`.

## Join Tables Live Editor task automatically selects merging variables based on scoring algorithm

### *Behavior change*

If the Join Tables Live Editor task fails to automatically select the first pair of merging variables based on row labels or variable names, then it tries to select them based on a scoring algorithm. The Join Tables task selects and tests candidate pairs of merging variables using these steps:

- 1 Select row names (in a table) or row times (in a timetable) as the first pair of merging variables.
- 2 If step 1 fails, then select variables with names that exactly match as the first pair.
- 3 If steps 1 and 2 fail, then score pairs of variables using the scoring algorithm. Select the pair of variables with the highest score as the first pair of merging variables.
- 4 If all previous steps fail, then select the first items in the **Merging variable** drop-down lists as the first pair of merging variables.

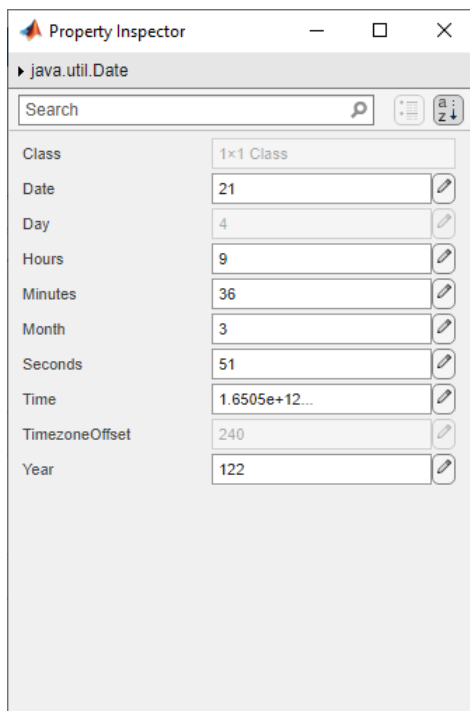
In previous releases, step 3 was to select the pair of variables whose names gave the best partial match as the first pair of merging variables.

## Java objects open in an improved Property Inspector

### *Behavior change*

Java objects now open in an improved Property Inspector that is consistent with the one used for graphics and other handle objects. For example, view and edit the properties of a Java object using the improved Property Inspector.

```
myDate = java.util.Date;
inspect(myDate)
```



**Calling head and tail functions without specified output arguments does not store output in ans***Behavior change*

When you call the `head` and `tail` functions without specified output arguments, they display the selected rows of input tables, but they do not store output in the `ans` variable. In previous releases, calling `head` and `tail` without specified output arguments causes output to be stored in `ans`.

Calling `head` and `tail` in a live script is usually not recommended. Instead, display the table or timetable by typing the variable name with no semicolon. The Live Editor provides a widget that enables you to examine the entire table or timetable. However, if you do call `head` or `tail` in a live script, you should assign the output to a variable so that the live script creates a widget for the output.

**addtodate function is not recommended***Still runs*

There are no plans to remove `addtodate`. However, the `datetime`, `duration`, and `calendarDuration` data types are recommended instead. The `datetime` data type provides flexible date and time formats, storage out to nanosecond precision, and properties to account for time zones and daylight saving time. You can add lengths of time to `datetime` values by using `duration` values (for hours, minutes, and seconds) or `calendarDuration` values (for calendar years, quarters, months, weeks, or days).

For example, convert a serial date number to a `datetime` value. Then add an array of hours, in 4-hour increments, by using the `hours` function to create an array of `duration` values. The result is an array of `datetime` values.

```
d = 738522;
d = datetime(d, "ConvertFrom", "datenum")

d = datetime
    01-Jan-2022

d = d + hours(0:4:12)

d = 1x4 datetime array
    01-Jan-2022 00:00:00    01-Jan-2022 04:00:00    01-Jan-2022 08:00:00    01-Jan-2022 12:00:00
```

Add 3 calendar months and 2 calendar weeks to a `datetime` value that represents January 1, 2022, 12:00 p.m. by using the `calmonths` and `calweeks` functions to create `calendarDuration` values. The result is a `datetime` value that represents April 15, 2022, 12:00 p.m.

```
d = datetime("01-Jan-2022 12:00:00");
d = d + calmonths(3) + calweeks(2)

d = datetime
    15-Apr-2022 12:00:00
```

**clock function is not recommended***Still runs*

There are no plans to remove `clock`. However, the `datetime` function is recommended instead. The `datetime` data type provides flexible date and time formats, storage out to nanosecond precision, and properties to account for time zones and daylight saving time.

To return the current time as a `datetime` value, call `datetime` with `"now"` as the input argument.



```
d = datetime("now")
```

```
d = datetime
    15-Apr-2022 15:53:28
```

### **date function is not recommended**

*Still runs*

There are no plans to remove `date`. However, the `datetime` function is recommended instead. The `datetime` data type provides flexible date and time formats, storage out to nanosecond precision, and properties to account for time zones and daylight saving time.

To return the current date as a `datetime` value, call `datetime` with `"today"` as the input argument.

```
d = datetime("today")
```

```
d = datetime
    15-Apr-2022
```

### **datenum function is not recommended**

*Still runs*

There are no plans to remove `datenum`. However, the `datetime`, `duration`, and `calendarDuration` data types are recommended instead. The `datetime` data type provides flexible date and time formats, storage out to nanosecond precision, and properties to account for time zones and daylight saving time. MATLAB functions that accept serial date numbers as inputs also accept `datetime` arrays as inputs.

To convert a serial date number to a `datetime` value, call `datetime` with the `ConvertFrom` name-value argument set to `"datenum"`.

```
d = 738522;
d = datetime(d, "ConvertFrom", "datenum")
```

```
d = datetime
    01-Jan-2022
```

### **datestr function is not recommended**

*Still runs*

There are no plans to remove `datestr`. However, the `string` and `char` functions are recommended instead for converting `datetime` arrays to text. The `datetime` data type provides flexible date and time formats, storage out to nanosecond precision, and properties to account for time zones and daylight saving time.

For example, represent the current date and time as a `datetime` value. Then convert it to a string.

```
d = datetime("now")
```

```
d = datetime
    15-Apr-2022 15:53:28
```

```
str = string(d)
```

```
str =
    "15-Apr-2022 15:53:28"
```

**Using datevec function on serial date numbers is not recommended***Still runs*

Using `datevec` with serial date numbers as inputs is not recommended. Use the `datetime` data type to represent points in time instead. The `datetime` data type provides flexible date and time formats, storage out to nanosecond precision, and properties to account for time zones and daylight saving time.

To split `datetime` values into date and time components, use the `hms`, `ymd`, or `datevec` functions, or use the `Year`, `Month`, `Day`, `Hour`, `Minute`, and `Second` properties of `datetime` values.

For example, call `datevec` on a `datetime` value that represents the current date and time.

```
d = datetime("now")

d = datetime
    15-Apr-2022 15:53:28

format shortg
dateVector = datevec(d)

dateVector =

    2022         4        15        15        53        28
```

To access one component, use the corresponding `datetime` property. For example, get the value of the `Month` property of `d`.

```
monthValue = d.Month

monthValue =

    4
```

**etime function is not recommended***Still runs*

There are no plans to remove `etime`. However, the `datetime`, `duration`, and `calendarDuration` data types are recommended instead. The `datetime` data type provides flexible date and time formats, storage out to nanosecond precision, and properties to account for time zones and daylight saving time.

To calculate the elapsed time between two `datetime` values, either subtract one from the other or use the `between` function. For example, subtract the start of today from the current date and time. Return the elapsed time as a `duration` value.

```
startOfToday = datetime("today")
currentTime = datetime("now")

startOfToday = datetime
    15-Apr-2022

currentTime = datetime
    15-Apr-2022 15:53:28

elapsedTime = currentTime - startOfToday
```

```
elapsedTime = duration
15:53:28
```

To return elapsed time as a `calendarDuration` value, use `between`.

```
d1 = datetime("2022-01-01")
d2 = datetime("now")
elapsedTime = between(d1,d2)

elapsedTime = calendarDuration
6mo 18d 13h 19m 18.55619499999974s
```

### **now function is not recommended**

*Still runs*

There are no plans to remove `now`. However, the `datetime` function is recommended instead. The `datetime` data type provides flexible date and time formats, storage out to nanosecond precision, and properties to account for time zones and daylight saving time.

To return the current date and time as a `datetime` value, call `datetime`. You can call it without any input arguments, or with `"now"` as the input argument.

```
d = datetime
% or
d = datetime("now")

d = datetime
15-Apr-2022 15:53:28
```

## Data Import and Export

### Parquet: Use Parquet files containing nested structured data

Write nested table and timetable variables to Parquet files using `parquetwrite`. Read nested data from Parquet files as nested tables using `parquetread` and `parquetDatastore`.

### SequentialDatastore Object: Sequentially read data from multiple datastores

Sequentially read data from multiple underlying datastores without concatenation using a `SequentialDatastore` object. Create a `SequentialDatastore` object using the `combine` function with the `ReadOrder` name-value argument.

### Datastores: Create subsets of arbitrarily nested transformations and combinations of datastores

Create subsets of data from `TransformedDatastore`, `CombinedDatastore`, and `SequentialDatastore` objects using the `subset` function. You can subset these datastore objects only if their underlying datastores are subsetting (able to be subset). The underlying datastores can also be transformations or combinations of datastores that are subsetting.

Test whether you can create subsets of these datastore objects using the `isSubsetting` function. `isSubsetting` returns `true` when the underlying datastore can be subset.

### Parallel Processing: Use low-level file I/O operations in thread-based environments

Use low-level file operations, such as `fread`, in thread-based environments. Parallel processing results in improved performance when reading data, especially with remote data. For a list of low-level functions that support thread-based environments, see [Low-level File I/O](#).

### CDF Interface: Read CDF\_INT8 and CDF\_TIME\_TT2000 data

You can now use `cdfinfo`, `cdfread`, and the low-level CDF package functions to display and read `CDF_INT8` and `CDF_TIME_TT2000` variables and attributes from CDF datasets.

- High-level interface — `cdfread` reads `CDF_TIME_TT2000` data as `datetime` values by default. To read the data as `int64` values, use the new name-value argument `DatetimeType`.
- Low-level interface — `CDF_TIME_TT2000` data is read as `int64` values.

Use these two new functions to convert to and from `CDF_TIME_TT2000` `int64` timestamps:

- `cdflib.computeTT2000` — Calculate a `CDF_TIME_TT2000` `int64` timestamp from an input vector of UTC-based date and time components.
- `cdflib.breakdownTT2000` — Decompose a `CDF_TIME_TT2000` `int64` timestamp into an output vector of UTC-based date and time components.

## **imwrite Function: Use datetime values to write metadata for PNG images**

You can write PNG image metadata using `datetime` values by specifying the `CreationTime` and `ImageModTime` name-value arguments with the `imwrite` function.

## **Image File Format Libraries: LibTIFF library upgraded to version 4.4.0**

The LibTIFF library is upgraded to version 4.4.0.

## **Scientific File Format Libraries: HDF4 and HDF-EOS2 libraries are upgraded**

The HDF4 library is upgraded to version 4.2.15, and the HDF-EOS2 library is upgraded to version 3.0.

## **▲ Functionality being removed or changed**

### **NetCDF Interface: Use Fletcher32 checksum filter only with fixed-length data types**

*Behavior change*

The Fletcher32 checksum filter can be used only with fixed-length data types and generates an error if you apply it to variables of type `NC_STRING` or `NC_VLEN`. Previously, if you applied the Fletcher32 filter to datasets that included `NC_STRING` or `NC_VLEN` data, the filter was silently ignored.

### **Use system web browser when opening links to external sites (recommended) preference has been removed**

*Behavior change*

The **Use system web browser when opening links to external sites (recommended)** preference has been removed. Open all external sites using your system web browser.

Previously, you could use this preference to specify whether to open external sites using the system web browser or the MATLAB web browser.

## Mathematics

### **pagenorm Function: Calculate norms using pages of N-D arrays**

Use the `pagenorm` function to calculate vector and matrix norms of the pages of N-D arrays. In this context, the N-D array is treated as a container for several 2-D matrices. This function is particularly useful for calculating error metrics for the outputs of other page-wise functions, such as `pagesvd` and `pageldivide`.

### **⚠️ Functionality being removed or changed**

#### **"0" syntaxes of `qr`, `gsvd`, and `svd` are not recommended for economy-size decompositions** *Still runs*

These `qr`, `gsvd`, and `svd` syntaxes for performing economy-size decompositions are no longer recommended:

- `[___] = qr(A,0)` and `[___] = qr(S,B,0)`
- `[U,V,X,C,S] = gsvd(A,B,0)`
- `[___] = svd(A,0)`

There are no plans to remove support for the "0" syntax. However, using the "econ" option to perform economy-size decompositions with these functions is recommended instead.

#### **gsvd output sizes are based on numerical rank of `[A; B]`** *Behavior change*

With the function call `[U,V,X,C,S] = gsvd(A,B)`, where `A` is `m`-by-`p` and `B` is `n`-by-`p`, the `gsvd` function returns:

- `X` as `p`-by-`q`
- `C` as `m`-by-`q`
- `S` as `n`-by-`q`

Additionally, with one output argument, the function call `sigma = gsvd(A,B)` returns a vector of the generalized singular values with length `q`.

The behavior change is that in all of these cases `q` is now equal to the numerical rank of `[A; B]`. The numerical rank is calculated from the QR factorization of `[A; B]`. This change ensures that nonzero elements of `C` and `S` are uniquely determined.

Previously, the value of `q` was `min(m+n,p)`, and `gsvd` returned extra columns (or elements) in the outputs when `[A; B]` was not full rank.

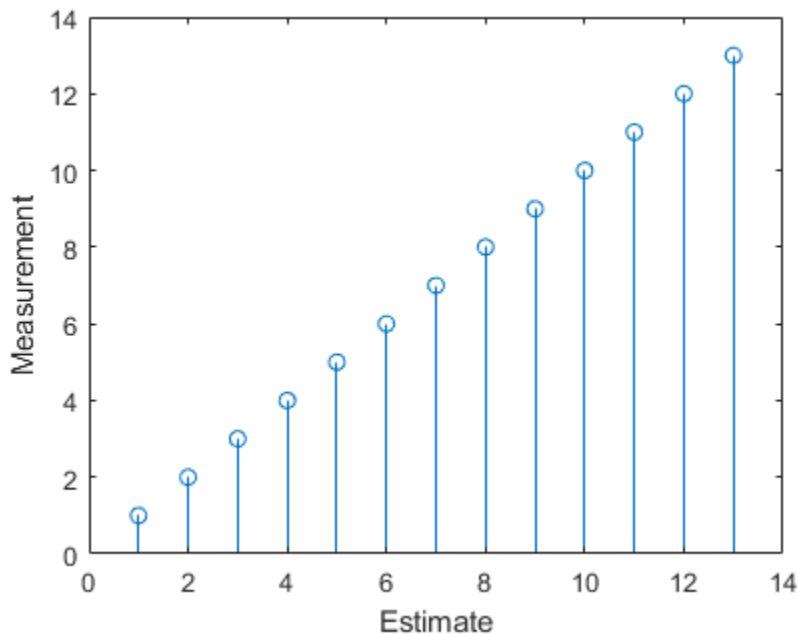
## Graphics

### Plotting Table Data: Create stem, staircase, and geographic plots by passing tables directly to plotting functions

Create plots by passing a table directly to any of these functions: `stem`, `stem3`, `stairs`, `geoplot`, and `geoscatter`. If you create any of these plots with a table and then add a legend, the legend items are automatically labeled with the table variable names. If you create a stem or staircase plot, the axis labels are automatically set to the corresponding variable names (geographic plots always have "Longitude" and "Latitude" axis labels regardless of the variable names).

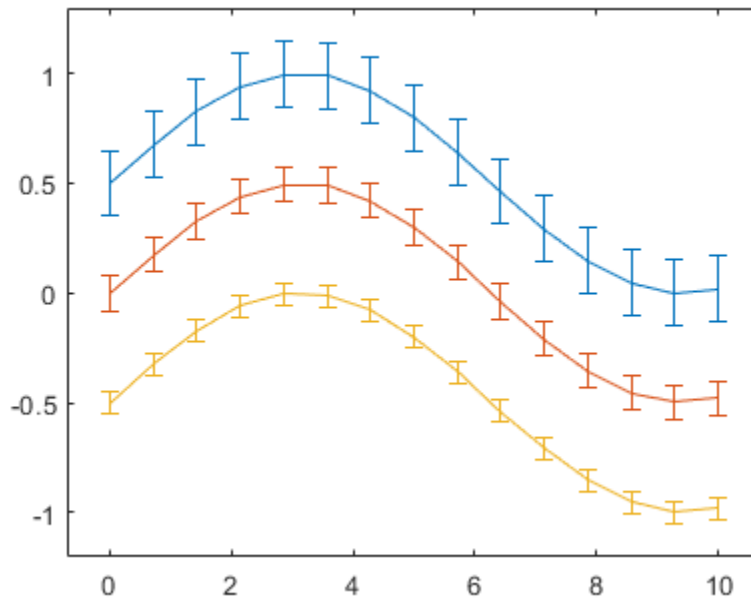
For example, create a table with the variables `Estimate` and `Measurement`. Pass the table to the `stem` function as the first argument, and specify the variables you want to plot.

```
Estimate = (1:13)';
Measurement = (1:13)';
T = table(Estimate,Measurement);
stem(T,"Estimate","Measurement")
```



### errorbar Function: Plot multiple lines with error bars at once

The `errorbar` function now accepts the same combinations of matrices and vectors as the `plot` function does. As a result, you can plot multiple lines at once rather than calling the `hold` function between plotting commands.



### **tightPosition Function: Get the location and size of axes plotting area**

Call the `tightPosition` function to get the location and size of the plotting area of the axes. This function is useful for aligning objects in the figure with the plotting area, particularly when you are working with polar axes, axes that display images, or other visualizations that have nondefault aspect ratios.

### **tilenum and tilerowcol Functions: Convert between rows and columns and tile numbers in tiled chart layouts**

When working with tiled chart layouts, you can find the tile number for a specific row and column by calling the `tilenum` function. To find the row and column numbers that correspond to a specific tile number, use the `tilerowcol` function. You can also use these functions to locate the objects within the layout.

### **Tiled Chart Layout: Define GridSizeChangedFcn callback that executes when the grid size changes**

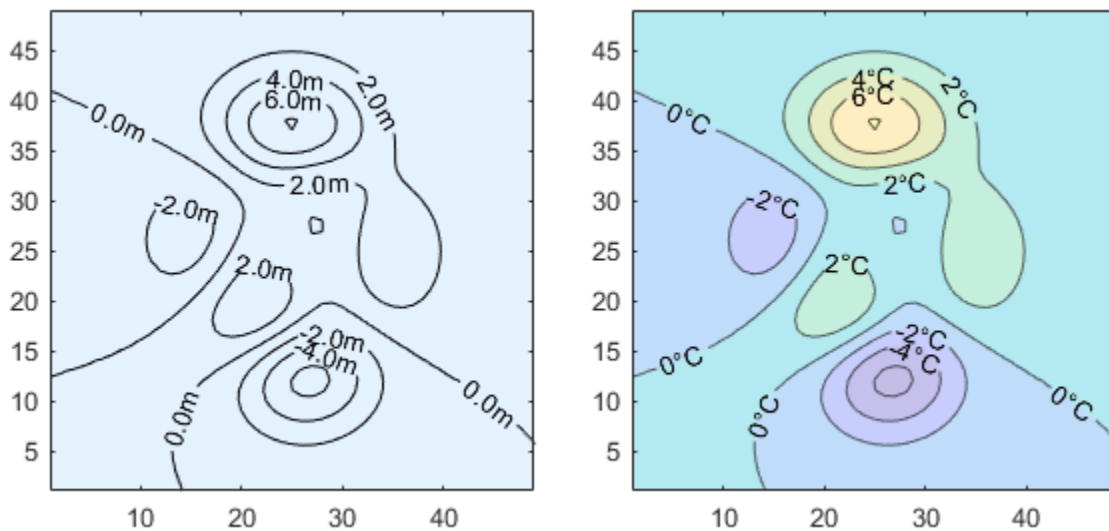
Define the `GridSizeChangedFcn` callback function on a tiled chart layout. The callback function executes when the `GridSize` property of the layout changes. This callback is primarily useful for layouts that use the "flow" tile arrangement. For example, you can define a callback that displays the x-axis tick labels only in the bottom row of axes.

### **Contour Plots: Customize contour labels, colors, and transparency**

Customize the appearance of a contour plot by setting these properties of the `Contour` object:

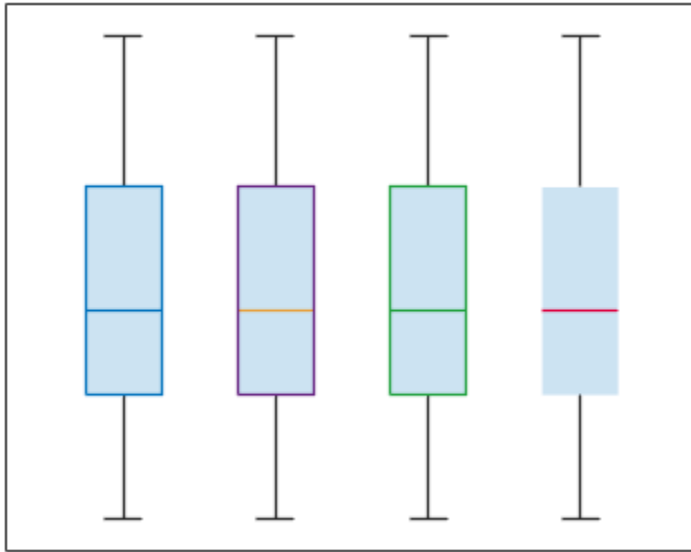


- **LabelFormat** — Specify the formatting of the contour labels as a format operator that the `compose` function accepts. You can also specify a handle to a function that performs calculations when formatting the labels.
- **FaceColor** and **EdgeColor** — Specify the fill color between contour lines with the **FaceColor** property, and specify the colors of the contour lines with the **EdgeColor** property. These new properties replace the **Fill** and **LineColor** properties from previous releases.
- **FaceAlpha** and **EdgeAlpha** — Specify the transparency of the colors between contour lines with the **FaceAlpha** property, and specify the transparency of the contour lines with the **EdgeAlpha** property.



### **boxchart Function: Specify the box edge color and box median line color**

The `boxchart` function now includes options for the box edge color and box median line color. You can specify these colors using the `BoxEdgeColor` and `BoxMedianLineColor` name-value arguments, respectively.



### 3-D Stem and Bar Plots: Expanded data type support

The `stem3`, `bar3`, and `bar3h` functions now support more data types. The available data types depend on the specific arguments.

These arguments can be any numeric values as well as datetime, duration, and categorical values:

- `stem3`: x, y, and z (stem x and y positions and corresponding z-values)
- `bar3`: y (bar positions along the y-axis)
- `bar3h`: z (bar positions along the z-axis)

These arguments can be any numeric values and duration values:

- `bar3`: z (bar heights)
- `bar3h`: y (bar lengths)

### Image Objects: Control the maximum resolution for displaying images

Control the maximum displayed resolution along the larger dimension of an image by setting the `MaxRenderedResolution` property when calling `image` or `imagesc`. The value affects the on-screen display, but it does not affect the image data, which is stored in the `CData` property of the image.

### Figure Code: Generate code for figure formatting in MATLAB Online

When you edit graphics object properties for a figure in MATLAB Online using the **Format** tab, MATLAB generates code for the figure formatting. On the **Figure** tab, click the **Format** button to open the **Format** tab. Use the options in the **Format** tab to interactively customize properties. Then, to view, copy, or export the generated code, select **Show Code** in the **File** section of the **Figure** tab.

Previously, code was not generated when editing properties using the **Format** tab. MATLAB does not generate code when editing properties using the Property Inspector.

## Plot Options: Customize figure creation, data linking, and labeling in MATLAB Online

In the Variables editor in MATLAB Online, use **Plot Options** in the **Variable** tab to customize plots and generate corresponding code in the command window. You can create a figure, link numeric data with supported charts, and generate titles, legends, and labels.

In addition, you can brush numeric data in a linked plot and create a new logical variable from selected indices. Create the logical variable by right-clicking the selection in the Variables editor or by clicking **New from Selection** in the **Variable** tab.

### ⚠️ Functionality being removed or changed

#### Legends update when you delete lines or other plot objects

*Behavior change*

If you delete an object from the axes, such as a Line or Scatter object, the legend updates to reflect the change regardless of whether the `AutoUpdate` property is set to "on" or "off".

Previously, the legend did not update when you removed objects from the axes while the `AutoUpdate` property was set to "off".

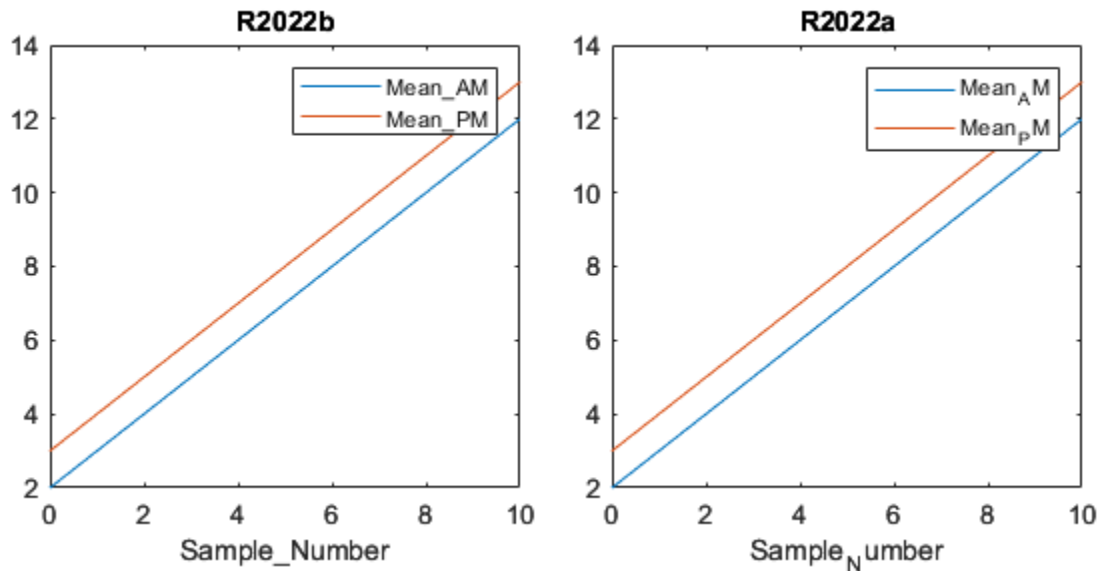
#### Plots created with tables preserve special characters in axis and legend labels

*Behavior change*

When you pass a table and one or more variable names to a plotting function, the axis and legend labels now display any special characters that are included in the table variable names, such as underscores. Previously, special characters were interpreted as TeX or LaTeX characters.

For example, this code creates a line plot from a table that has underscores in the variable names. In R2022b, the underscores appear in the x-axis label and the legend labels. In R2022a and earlier releases, the underscores are interpreted as subscripts.

```
Sample_Number = (0:10)';
Mean_AM = (2:12)';
Mean_PM = (3:13)';
t = table(Sample_Number,Mean_AM,Mean_PM);
plot(t,"Sample_Number",["Mean_AM" "Mean_PM"])
legend
```



To display axis and legend labels with TeX or LaTeX formatting, specify the labels manually. In this case, call the `xlabel` and `legend` functions with the desired label strings.

```
xlabel("Sample_Number")
legend(["Mean_AM" "Mean_PM"])
```

### Adding new plots to geographic axes does not reset basemap

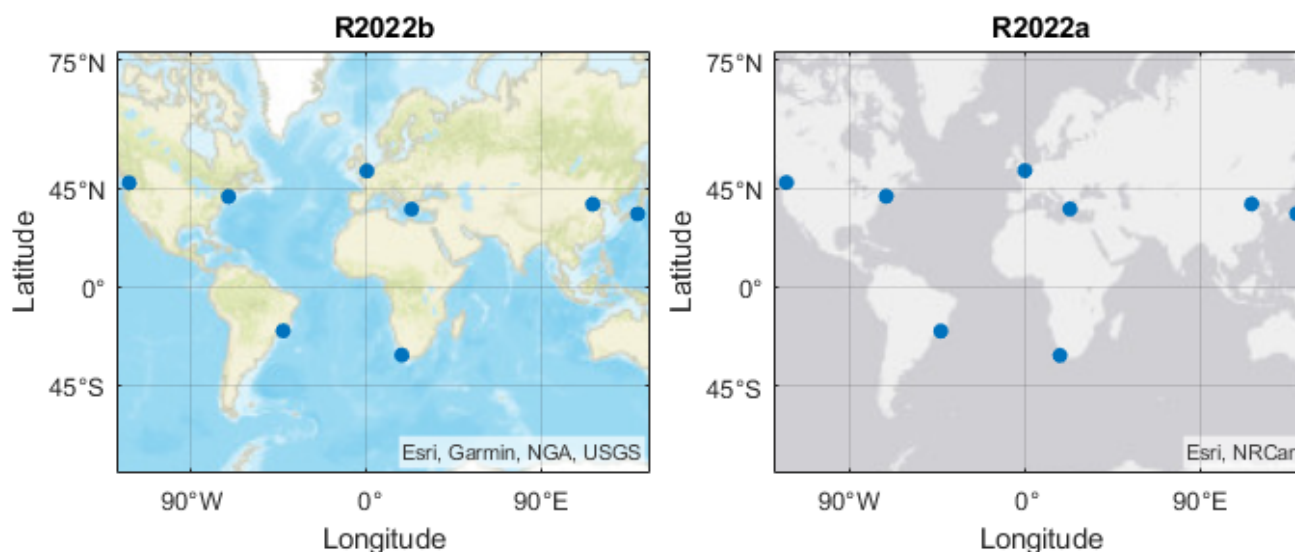
#### *Behavior change*

When you plot into geographic axes by using functions such as `geoplot` and `geoscatter`, MATLAB does not reset the basemap. In R2022a and earlier releases, the basemap resets when you add new plots.

As a result, you can specify a basemap and then visualize data without using the `hold` function between commands. For example, this code creates a map using the `streets` basemap. Then it displays a scatter plot over the basemap. In R2022b, the basemap does not reset. In R2022a and earlier releases, the basemap resets to the default `streets-light`.

```
lat = [35 -22 51 39 37 42 47 -33];
lon = [139 -43 0 116 23 -71 -122 18];

figure
geobasemap('streets')
geoscatter(lat,lon,"filled")
```



This change does not affect existing code that sets the hold state to "on" between commands.

To reset the basemap when you add a new plot, use the `cla reset` syntax of the `cla` function before you create the plot. For example, to update the preceding code, use `cla reset` between the calls to `geobasemap` and `geoscatter`.

```
lat = [35 -22 51 39 37 42 47 -33];
lon = [139 -43 0 116 23 -71 -122 18];
```

```
figure
geobasemap streets
cla reset
geoscatter(lat,lon,"filled")
```

Alternatively, you can change the basemap to the default `streets-light` by using the `geobasemap` function. For more information about changing the basemap of geographic axes, see [Access Basemaps for Geographic Axes and Charts](#).

### **errorbar returns error for 3-D and higher dimensional data arrays**

*Behavior change*

The `errorbar` function now returns an error when you specify 3-D or higher dimensional arrays for the x-coordinates, y-coordinates, or the error bar lengths. Previously, `errorbar` plotted the data from the first two dimensions and silently ignored the data in the higher dimensions.

### **Fill and LineColor properties of Contour objects are not recommended**

*Still runs*

Setting or getting the value of the `Fill` property or the `LineColor` property of a `Contour` object is not recommended. Use the `FaceColor` and `EdgeColor` properties instead. The `FaceColor` property provides more options for specifying colors than the `Fill` property does, and the `EdgeColor` property provides the same options as the `LineColor` property does.

This table shows typical usages of the `Fill` and `LineColor` properties and how to update your code to use the `FaceColor` and `EdgeColor` properties instead.

| Not Recommended                                 | Recommended                                     |
|-------------------------------------------------|-------------------------------------------------|
| <code>contour(peaks,"Fill",true)</code>         | <code>contour(peaks,"FaceColor","flat")</code>  |
| <code>contour(peaks,"LineColor",[0 0 0])</code> | <code>contour(peaks,"EdgeColor",[0 0 0])</code> |

There are no plans to remove the `Fill` or `LineColor` properties, but they are no longer listed when you call the `set`, `get`, or `properties` functions on the `Contour` object.

### Some printing and exporting workflows will no longer support UI components

*Still runs*

These functions and menu items will no longer support printing or exporting UI components in a future release:

- The `print` function
- The `saveas` function
- The `hgexport` function
- The **File > Save As** menu item on the figure
- The **File > Export Setup** menu item on the figure
- The **Edit > Copy Options** menu item on the figure

To export a figure containing UI components, call the `exportapp` function. For example, create a simple app containing two buttons and a slider. Export the contents of the figure as a PDF file by calling the `exportapp` function.

```
% Create figure with three UI components
f = uifigure;
button1 = uibutton(f,"Position",[150 300 100 50]);
button2 = uibutton(f,"Position",[300 300 100 50]);
slider1 = uislider(f,"Position",[150 250 250 3]);

% Export the contents of the figure
exportapp(f,"myapp.pdf")
```

Alternatively, call the `getframe` function to capture the contents of the figure. Then call the `imwrite` function to save the content. This time, save the content as a JPEG file.

```
F = getframe(f);
imwrite(F.cdata,"myapp.jpg");
```

### Exporting workflows will no longer support BMP, HDF, PBM, PCX, PGM, and PPM files

*Still runs*

These functions and menu items will no longer support BMP, HDF, PBM, PCX, PGM, and PPM files in a future release:

- The `print` function
- The `saveas` function
- The `hgexport` function
- The **File > Save As** menu item on the figure
- The **File > Export Setup** menu item on the figure

To export graphics using one of these formats, use the `imwrite` function instead. For example, create a line plot, and capture the contents of the current figure using the `getframe` function. Then save the content as a BMP file.

```
plot([0 3 2 4 1]);
F = getframe(gcf);
imwrite(F.cdata, "myplot.bmp");
```

### **Some exporting workflows will no longer support PostScript (.ps) files**

*Still runs*

These functions and menu items will no longer support creating full-page PostScript (.ps) files in a future release:

- The `print` function
- The `saveas` function
- The `hgexport` function
- The **File > Save As** menu item on the figure
- The **File > Export Setup** menu item on the figure

To export graphics files, call the `exportgraphics` function. Specify an .eps, .pdf, or .emf file extension and set the `ContentType` option to "vector". This function captures content that is tightly cropped around plots, and it does not create full-page output. For example, create a plot and save the contents of the current figure as a PDF file containing vector graphics.

```
plot([0 3 2 4 1]);
exportgraphics(gcf, "myplot.pdf", "ContentType", "vector")
```

Alternatively, call the `print` function and specify an .eps, .emf, or .svg file extension. For example, create a plot and save the contents of the current figure as an EPS file.

```
plot([0 3 2 4 1]);
print("myplot.eps", "-depsc")
```

### **The `getframe` function will no longer capture the figure toolbar, menu bar, or borders**

*Still runs*

In a future release, the `getframe` function will no longer support capturing the figure toolbar, menu bar, or the border around the figure using the `rect` argument. You will still be able to define a subsection of the figure to capture, but the toolbar, menu bar, and borders will not be included.

As an alternative, you can use the `exportapp` function to capture the toolbar and menu bar, but not the borders.

### **The `printopt` function will be removed**

*Still runs*

The `printopt` function will be removed in a future release. To configure printer defaults, use the options provided by your operating system.

## App Building

### **ClickedFcn and DoubleClickedFcn Callbacks: Program a response to a user clicking or double-clicking a tree, check box tree, list box, table, or drop-down component**

Program a response to a user clicking and double-clicking a UI component by using the `ClickedFcn` and `DoubleClickedFcn` callback properties. You can specify both of these callbacks for tree, check box tree, list box, and table UI components in App Designer and apps created using the `uifigure` function. You can also specify the `ClickedFcn` callback for drop-down components.

For more information, see the UI component properties pages:

- [Tree Properties](#)
- [CheckBoxTree Properties](#)
- [ListBox Properties](#)
- [Table Properties](#)
- [DropDown Properties](#)

### **uieditfield Function: Specify valid length and input type for edit field text**

You can now specify the length and character type of valid input text for a text edit field created using the `uieditfield` function.

- Use the `CharacterLimits` property to specify a maximum and minimum number of allowed characters.
- Use the `InputType` property to restrict the allowed character types. Specify `InputType` as "letters", "digits", "alphanumerics", or "text".

For more information, see [EditField Properties](#).

### **uiimage Function: Create an image hyperlink**

To open a web address when a user clicks an image in your app, specify the `URL` property of the image component created using the `uiimage` function.

For more information, see [Image Properties](#).

### **uipanel and uibuttongroup Functions: Specify container border width**

In apps created using the `uifigure` function, use the `BorderWidth` property to change the border width of panels and button groups created using the `uipanel` and `uibuttongroup` functions.

For more information, see [Panel Properties](#) and [ButtonGroup Properties](#).



## **uigridlayout Function: Query grid layout manager size and location**

Use the `Position`, `InnerPosition`, and `OuterPosition` properties of a grid layout manager created using the `uigridlayout` function to access its size and location. These properties are read-only.

For more information, see [GridLayout Properties](#).

## **uibutton and uitogglebutton Functions: Specify additional icon and icon alignment options**

You can now use these additional options when adding an icon to a push button, state button, or toggle button:

- Specify the `Icon` property as a predefined icon, such as `"success"`.
- Specify the `IconAlignment` property as `"leftmargin"` or `"rightmargin"` to display the icon on the far left or far right of the button, regardless of the button text location.

For more information, see [Button Properties](#).

## **uitoolbar Function: Specify background color of toolbar**

Specify the background color of a toolbar created using the `uitoolbar` function by using the `BackgroundColor` property. You can modify this property only in App Designer and apps created using the `uifigure` function.

For more information, see [Toolbar Properties](#).

## **scroll Function: Programmatically scroll to specified component in container**

To programmatically scroll to a component in a container or button group with the `scroll` function, use the syntax `scroll(container, component)`.

## **isInScrollView Function: Determine if any component in a container is visible, not only direct children**

When you use the `isInScrollView` function to determine if a UI component is visible in a scrollable container, that component can be any descendent of the scrollable container, not only a direct child. For example, you can check whether a button in a panel is visible in a scrollable UI figure.

## **App Designer: Add and delete callbacks more efficiently**

When you select multiple components in App Designer that have a callback type in common, you can now create a single shared callback for all the selected components. For example, in an app with an edit field and a slider, you can select the two components, right-click one of them, and select **Callbacks > Add ValueChangingFcn callback**. App Designer creates a single new callback and assigns it to both the edit field and the slider.

You can also now more easily find callbacks in your app that are not assigned to any components. In **Code View**, navigate to the **Callbacks** tab in the **Code Browser**. An unassigned callback appears

with a  icon next to its name. If you do not need the callback, you can delete it by right-clicking the callback name and selecting  **Delete**.

## App Designer: Move components between tabs

To move a component from one tab to another, in **Design View**, drag the component and pause on the new tab. App Designer switches the selected tab to the new tab. You can then position the component in the new tab.

## Graphics Support: Restore axes view and display context menu for interaction modes

In a figure created with the `uifigure` function or a figure in MATLAB Online, when an interaction mode is enabled, you can double-click in the axes to restore the original axes limits and view. You can also right-click to display a mode-based context menu.

For example, create a plot in a UI figure and enable the pan interaction mode.

```
fig = uifigure;  
ax = axes(fig);  
plot(ax,1:10)  
pan(ax,"on");
```

Interactively pan the axes. Then, restore the original axes limits and view by double-clicking in the axes. You can display a mode-based context menu by right-clicking in the axes.

Previously, these workflows were supported only for figures created with the `figure` function.

## Graphics Support: Use visual camera toolbar interface

Syntaxes of `cameratoolbar` that make the visual toolbar interface visible are now supported by figures created with the `uifigure` function. Previously, figures created with the `uifigure` function supported only syntaxes of the `cameratoolbar` function that did not directly make the toolbar visible. Now, the visual camera toolbar interface and all syntaxes are supported by figures and UI figures.

## Comparison Tool: Toggle the ability to navigate only mergeable changes

When using the Comparison Tool to merge changes between two app files, you can now choose to

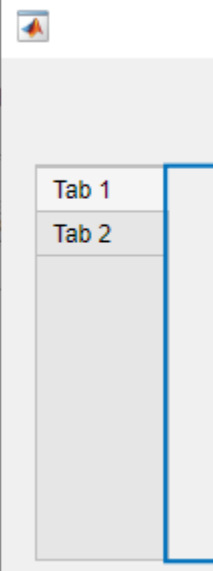
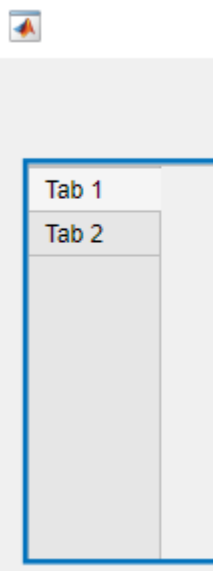
navigate only mergeable changes. To toggle this ability, while in merge mode, click  **Mergeable Only** in the toolbar. If **Mergeable Only** is selected, when you navigate changes using the **Previous** and **Next** buttons, the tool skips changes that are not mergeable, such as changes to the noneditable code generated by App Designer. The **Mergeable Only** button is selected by default.

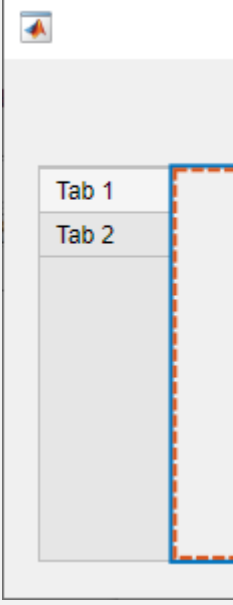
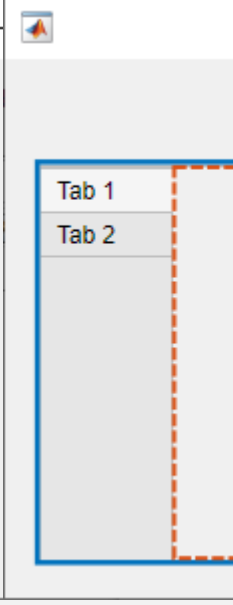
## Functionality being removed or changed

**InnerPosition** property of tab group returns the position excluding borders and titles  
*Behavior change*

For apps created in App Designer and using the `uifigure` function, when you query the `InnerPosition` property of a tab group created using the `uitabgroup`, the property returns a different value than in the previous release. In addition, if the `TabLocation` is 'left' or 'bottom', when you query the `Position`, `InnerPosition`, and `OuterPosition` properties of a tab created using the `uitab` function, the properties might return different values than in the previous release.

This table describes the differences in the property values between R2022b and previous releases.

| Property                     | Starting in R2022b                                                                                                                                                                                                           |                                                                                    | R2022a and Earlier                                                                                                                                                                                                           |                                           |
|------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------|
|                              | Description                                                                                                                                                                                                                  | Image                                                                              | Description                                                                                                                                                                                                                  | Image                                     |
| InnerPosition of a tab group | A four-element vector of the form [left bottom width height] that represents the size and location of the tab group relative to the drawable area of its parent, <i>excluding</i> borders and titles of the tabs it contains | Tab group InnerPosition (blue solid line)                                          | A four-element vector of the form [left bottom width height] that represents the size and location of the tab group relative to the drawable area of its parent, <i>including</i> borders and titles of the tabs it contains | Tab group InnerPosition (blue solid line) |
|                              |                                                                                                                                                                                                                              |  |                                                                                                                                          |                                           |

| Property                                            | Starting in R2022b                                                                                                                                                                                                          |                                                                                                                                                                               | R2022a and Earlier                                                                                                                                                                                                          |                                                                                                                                                                                 |
|-----------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                                                     | Description                                                                                                                                                                                                                 | Image                                                                                                                                                                         | Description                                                                                                                                                                                                                 | Image                                                                                                                                                                           |
| Position, InnerPosition, and OuterPosition of a tab | A four-element vector of the form [left bottom width height] that represents the size and location of the tab relative to the tab group drawable area, where the tab group drawable area <i>excludes</i> borders and titles | Tab Position (orange dashed line) relative to tab group drawable area (blue solid line)<br> | A four-element vector of the form [left bottom width height] that represents the size and location of the tab relative to the tab group drawable area, where the tab group drawable area <i>includes</i> borders and titles | Tab Position (orange dashed line) relative to tab group drawable area (blue solid line)<br> |

If you notice changes in your app layout, consider updating your code in this way:

- Tab group InnerPosition — Replace references to the tab group InnerPosition property with the tab group Position property to use the size and location of the tab group including the borders and tab titles.
- Tab Position, InnerPosition, and OuterPosition — Replace references to the first two elements of the position vector to instead use the space taken up on the left or bottom by the tab titles. For example, if your app contains a tab group with a TabLocation value of 'left' and you reference the Position property of a tab in that tab group, replace that reference with the horizontal distance between the tab group border and the tab drawable area.

**Table Extent property will return the default table UI component width and height**  
*Behavior change in future release*

In a future release, when you query the Extent property of a table UI component created using the `uitable` function in a figure-based app, the property will always return [0 0 300 300]. The third and fourth elements of the vector represent the default table width and height. Currently, querying Extent returns a vector where the third and fourth elements represent the width and height of the rectangle outlining the table area, estimated using the table data, row heights, and column widths. If you use the Extent property to specify the position of UI components in your app, the table UI component might have a different size or the layout of your app might appear shifted.

Consider updating your code to remove references to the table `Extent` property by using one of these alternatives:

- Lay out the table and other UI components by manually specifying the `Position` property.
- Transition your `figure`-based app to use `uifigure`, and use a grid layout manager with a row height or column width of `'fit'` to automatically scale your app layout based on the size of the table with its data. For more information, see [Update figure -Based Apps to Use uifigure](#).

## Performance

### **prctile, quantile, and iqr Functions: Improved performance with small input data**

The `prctile`, `quantile`, and `iqr` functions show improved performance due to faster input parsing. The performance improvement is most significant when input parsing is a greater portion of the computation time. This situation occurs when:

- The size of the input data is small.
- The number of percentages or cumulative probabilities is small.
- Computation is along the default operating dimension.

For example, this code calculates four percentiles for a 3000-element matrix. The code is about 5x faster than in the previous release.

```
function timingPrctile
A = rand(300,10);
for k = 1:3e3
    P = prctile(A,[20 40 60 80]);
end
end
```

The approximate execution times are:

**R2022a:** 1.0 s

**R2022b:** 0.2 s

The code was timed on a Windows 10, Intel Xeon CPU E5-1650 v4 @ 3.60 GHz test system using the `timeit` function.

```
timeit(@timingPrctile)
```

### **mldivide and pagemldivide Functions: Improved performance with small matrices**

The `mldivide` and `pagemldivide` functions show improved performance when solving linear systems  $A*x = b$  with a small coefficient matrix  $A$ . The performance improvement applies to real matrices that are 16-by-16 or smaller, and complex matrices that are 8-by-8 or smaller.

- Using `mldivide`, this code solves a linear system specified by a real 12-by-12 matrix. The code is about 1.7x faster than in the previous release.

```
function mldividePerf
A = rand(12);
for k = 1:1e5
    x = A\A;
end
end
```

The approximate execution times are:

**R2022a:** 0.72 s

**R2022b:** 0.42 s

- Using `pagemldivide`, this code solves three triangular linear systems specified by an 11-by-11-by-3 array. The code is about 2.2x faster than in the previous release.

```
function pagemldividePerf
a = triu(rand(11));
A = cat(3,a,a,a);
for k = 1:1e5
    x = pagemldivide(A,A);
end
end
```

The approximate execution times are:

**R2022a:** 0.65 s

**R2022b:** 0.30 s

In both cases, the code was timed on a Windows 10, Intel Xeon CPU W-2133 @ 3.60 GHz test system using the `timeit` function:

```
timeit(@mldividePerf)
timeit(@pagemldividePerf)
```

## **conv, conv2, and convn Functions: Improved performance when convolving two vectors, matrices, and arrays**

The `conv`, `conv2`, and `convn` functions show improved performance when convolving two vectors, matrices, and arrays for almost all input sizes.

- Using `conv`, this code convolves a 1,000,000-by-1 vector and a 10,000-by-1 vector. The code is about 3x faster than in the previous release.

```
function timingTest
rng default;
u = randn(1000000,1);
v = randn(10000,1);
tic
w = conv(u,v);
toc
end
```

The approximate execution times are:

**R2022a:** 0.40 s

**R2022b:** 0.13 s

- Using `conv2`, this code convolves a 512-by-512 matrix and a 256-by-256 matrix. The code is about 4x faster than in the previous release.

```
function timingTest
rng default;
A = randn(512);
B = randn(256);
tic
```

```
C = conv2(A,B);  
toc  
end
```

The approximate execution times are:

**R2022a:** 0.97 s

**R2022b:** 0.24 s

- Using `convn`, this code convolves a 128-by-64-by-32 array and a 64-by-32-by-16 array. The code is about 4.9x faster than in the previous release.

```
function timingTest  
rng default;  
A = randn(128,64,32);  
B = randn(64,32,16);  
tic  
C = convn(A,B);  
toc  
end
```

The approximate execution times are:

**R2022a:** 1.08 s

**R2022b:** 0.22 s

All of the code was timed on a Windows 10, Intel Xeon CPU E5-1650 v4 @ 3.60 GHz test system by calling the `timingTest` function in each case.

## Tall Arrays: Improved performance when indexing tall arrays backed by `parquetDatastore`

Tall arrays backed by `parquetDatastore` show improved performance with row-indexing operations. The performance improvement arises from the use of predicate pushdown to reduce the amount of data that is imported into MATLAB. So, the improved performance is most noticeable with data that has a large number of rows compared to the number of rows returned by the indexing operation. The improvement gets better as the number of rows being filtered increases.

For example, this code creates a tall array backed by a Parquet datastore for the `airlinesmall.parquet` file, and then performs row indexing to extract rows that meet several specified conditions. The operation returns 10 rows out of 1.2 million and executes about 2.3x faster than in the previous release.

```
function parquetPerf  
mapreducer(0)  
pds = parquetDatastore("airlinesmall.parquet");  
pds.Files = repelem(pds.Files,10);  
T = tall(pds);  
conditions = T.DepDelay >= seconds(10) & T.DepDelay <= seconds(80) ...  
    & T.UniqueCarrier == "AA" ...  
    & T.DepTime >= datetime(2000,1,1) & T.DepTime < datetime(2000,1,7);  
  
idx_rows = T(conditions,:);  
g = gather(idx_rows);  
end
```



The approximate execution times are:

**R2022a:** 8.0 s

**R2022b:** 3.5 s

The code was timed on a Windows 10, Intel Xeon CPU W-2133 @ 3.60 GHz test system using the `timeit` function:

```
timeit(@parquetPerf)
```

## App Building: Improved app startup performance

Apps created in App Designer and using the `uifigure` function start up faster in R2022b than in R2022a. The improvement is more noticeable for apps with many UI components and apps that use a grid layout manager. The startup performance is faster both the first time you run your app during a MATLAB session and in subsequent times.

For example, this code creates an app with 1000 edit field components. The code is about 1.6x faster than in the previous release.

```
function timingApp
fig = uifigure;
gl = uigridlayout(fig,Scrollable="on");
gl.RowHeight = repmat({'fit'},1,100);
gl.ColumnWidth = repmat({'fit'},1,10);

for k = 1:1000
    uieditfield(gl);
end
drawnow
end
```

The approximate execution times are:

**R2022a:** 17.3 s

**R2022b:** 10.6 s

The code was timed on a Windows 10, Intel Xeon CPU E5-1650 v4 @ 3.60 GHz test system by running the `timingApp` function and measuring the time it takes for the edit fields to appear in the UI figure window.

## ▲ App Building: Improved startup performance for apps with multiple tabs

In addition to the overall app startup performance improvement in R2022b, apps that contain multiple tabs show an even greater startup performance improvement. The reason is that MATLAB prioritizes creating the content in the visible tab over non-visible content when the app first runs. This improvement is more noticeable for apps with a large number of UI components in unselected tabs. The particular performance improvement that you see depends on the app layout and UI component types.

For example, this code creates a tab group with five tabs, each containing 200 edit field components. The code is about 1.9x faster than in the previous release.

```
function timingTabApp
fig = uifigure;
tg = uitabgroup(fig);

for k1 = 1:5
    t = uitab(tg);
    gl = uigridlayout(t,Scrollable="on");
    gl.RowHeight = repmat({'fit'},1,20);
    gl.ColumnWidth = repmat({'fit'},1,10);
    for k2 = 1:200
        uieditfield(gl);
    end
end
drawnow
end
```

The approximate execution times are:

**R2022a:** 11.8 s

**R2022b:** 6.3 s

The code was timed on a Windows 10, Intel Xeon CPU E5-1650 v4 @ 3.60 GHz test system by running the `timingTabApp` function and measuring the time it takes for the components to appear in the UI figure window.

If you have an app with many UI components, consider updating your app layout to take advantage of this improvement. For more information, see [Improve App Startup Time](#).

## ▲ Compatibility Considerations

When an app user switches to a new tab for the first time after running an app, the interaction might take more time than in previous releases. The reason is that MATLAB might create some content in the tab only after the user selects the tab. If the user later switches to the same tab again, the interaction does not take the additional time.

## Plots in Apps: Improved responsiveness when interacting with large images

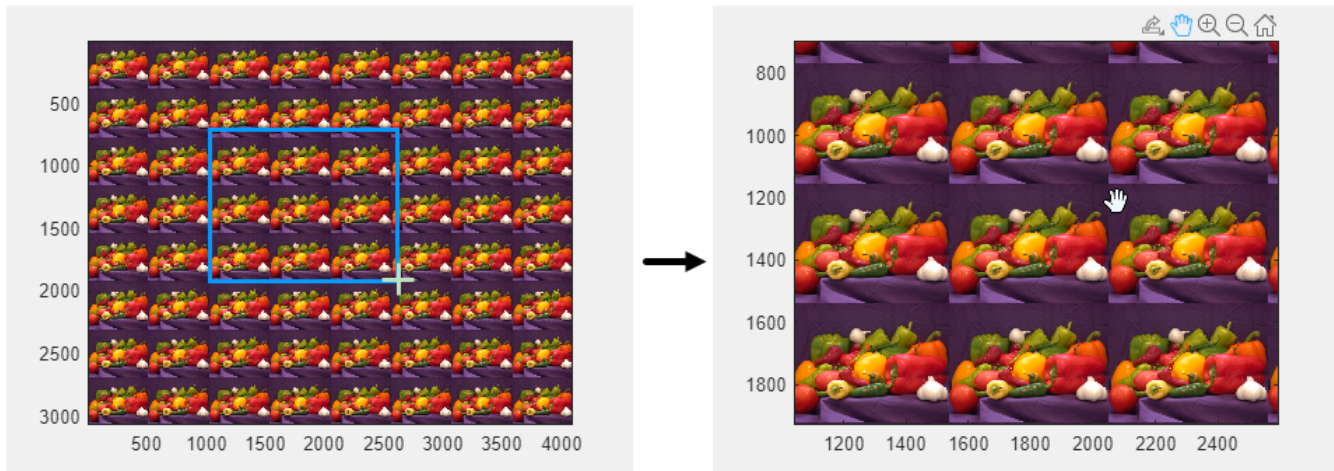
Interactions with large images are more performant within apps and within figures created with the `uifigure` function. The improvement is noticeable for images with at least 3000 pixels along one dimension. In particular, these interactions are more responsive in R2022b than in R2022a:

- **Zooming out** — The image updates immediately after you zoom out of an image. Previously, there was a delay before the image updated.
- **Panning** — The image updates as you are panning within an image. Previously, the image updated after you finished panning (for example, after releasing the mouse).

For example, if you run this code on a Windows 10, Intel Xeon CPU W-2133 @ 3.60 GHz test system with a modern graphics card, and then zoom into the resulting image and pan around, the image updates immediately as you interact.

```
imdata = imread("peppers.png");
imdata2 = repmat(imdata,8,8,1);
```

```
f = uifigure;
ax = axes(f);
image(ax,imdata2)
```



## Plots in Apps: Ticks and grid lines update as you pan

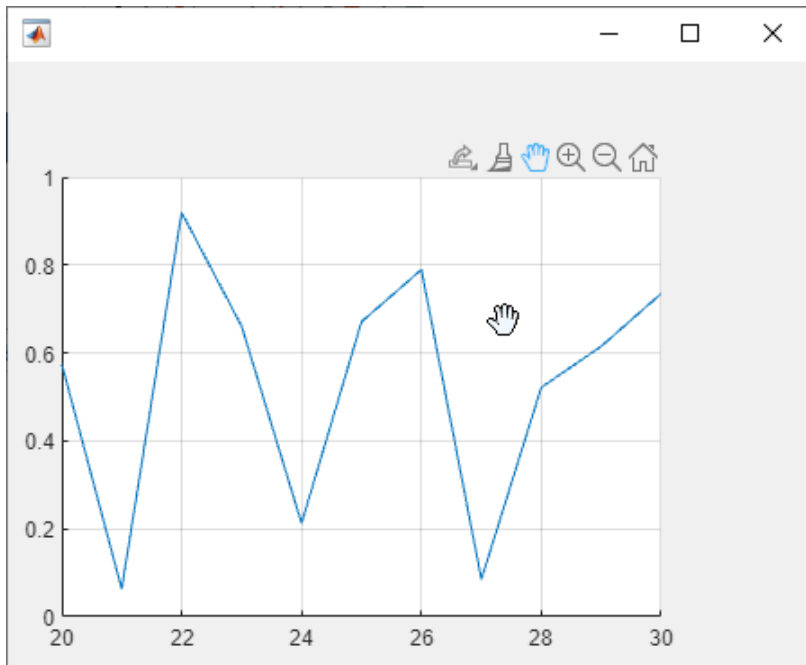
Axes ticks and grid lines update immediately as you pan within many types of Cartesian plots that are displayed in apps. Previously, the ticks and grid lines updated after you released the mouse. Now, the ticks update immediately for most plots that meet all of these conditions:

- They are in 2-D Cartesian space
- They display numeric data on a linear scale
- They have the default tick values and labels
- They have the default axis locations

Grid lines update immediately under the same conditions, but you can see the improvement in 3-D views in addition to 2-D views.

For example, if you run this code on a Windows 10, Intel Xeon CPU W-2133 @ 3.60 GHz test system and then pan within the axes, the ticks and the grid lines update immediately as you pan.

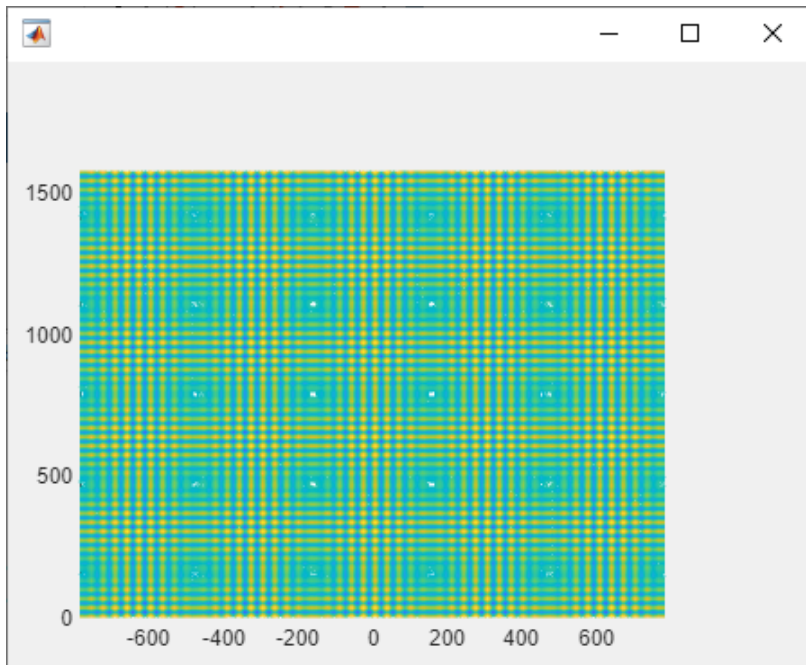
```
f = uifigure;
ax = uiaxes(f);
x = 1:50;
y = rand(1,50);
plot(ax,x,y);
ax.XLim = [20,30];
grid(ax,"on")
```



## Plots in Apps: Improved performance and smaller PDF files when exporting complex plots

The `exportgraphics` function has improved performance and creates a smaller file when you export a complex graphic as a PDF containing vector graphics. To observe the improvement, the plot must be displayed in an app or in a figure created with the `uifigure` function. For example, this code creates a figure containing a complex contour plot. Then it calls the `exportgraphics` function to export the plot as a PDF containing vector graphics content. The `exportgraphics` function executes about 2.1x faster, and the file size is about 23% smaller than in the previous release.

```
function myapp
f = uifigure;
ax = uiaxes(f);
x = linspace(-250*pi,250*pi);
y = linspace(0,500*pi);
[X,Y] = meshgrid(x,y);
Z = sin(X)+cos(Y);
contour(ax,X,Y,Z,50)
tic
exportgraphics(ax,"mycontours.pdf","contentType","vector")
toc
end
```



The approximate execution times (and file sizes) are:

**R2022a:** 16.3 s (3.45 MB)

**R2022b:** 7.7 s (2.66 MB)

The code was timed on a Windows 10, Intel Xeon CPU E5-2660 @ 2.20 GHz test system by calling the `myapp` function:

`myapp`

## Tiled Chart Layouts: Improved performance for flow layouts with spanned axes

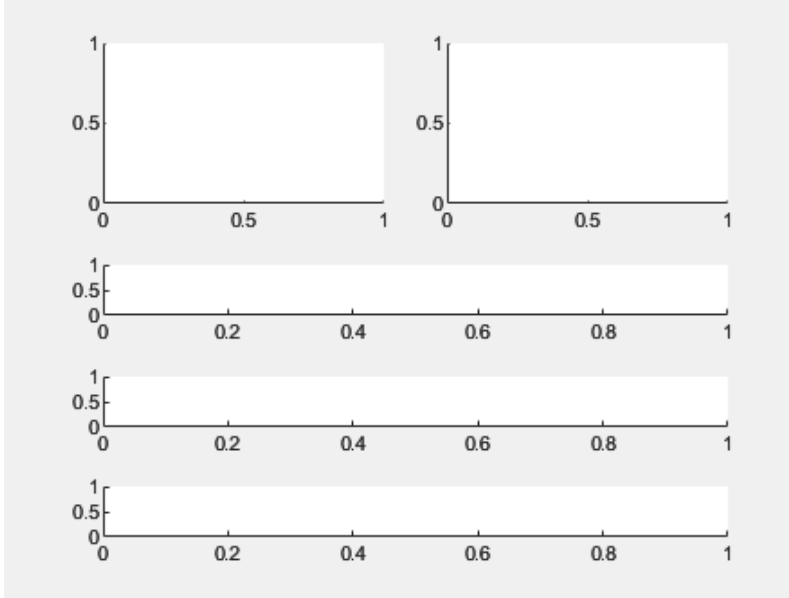
Tiled chart layouts that have the "flow" tile arrangement and axes that span several tiles are more performant when they update. Layouts update when you make changes that affect the arrangement of the tiles, including:

- Adding new axes
- Resizing the figure
- Customizing the appearance of the axes by setting axes properties

The performance improvement is more significant when there are axes that span many tiles. For example, this code is about 1.4x faster than in the previous release.

```
function mylayout
t = tiledlayout("flow");
nexttile(t,[20 15])
nexttile(t,[20 15])
nexttile(t,[10 30])
nexttile(t,[10 30])
```

```
nexttile(t,[10 30])  
drawnow  
end
```



The approximate execution times are:

**R2022a:** 0.1082 s

**R2022b:** 0.0788 s

The code was timed on a Windows 10, Intel Xeon CPU W-2133 @ 3.60 GHz test system using the `timeit` function:

```
timeit(@mylayout)
```

## ArrayDatastore Object: Improved performance when reading data using readall

The `readall` function shows improved performance when reading data from an `ArrayDatastore` object that has an `OutputType` property set to "same". The `readall` function increases performance by removing unnecessary data copies. For example, this code is significantly faster than in the previous release:

```
function t = timingTest  
    data = parquetread("airlinesmall.parquet");  
    arrds = arrayDatastore(data,OutputType="same");  
    tic;  
    t = readall(arrds);  
    toc;  
end
```

The approximate execution times are:

**R2022a:** 1584.36 seconds

**R2022b:** 0.12 seconds

The code was timed on a Windows 10, Intel Xeon (R) W-2133 @ 3.60 GHz test system by calling the `timingTest` function.

## ArrayDatastore Object: Improved performance when specifying number of rows to read

The `read` function shows improved performance when reading data from an `ArrayDatastore` object that has an `OutputType` property set to "same" and a value specified for the `ReadSize` property. The performance improvement increases as the `ReadSize` value gets larger. For example, this code is about 4.5x faster than in the previous release:

```
function c = timingTest
    data2 = parquetread("outages.parquet");
    arrds = arrayDatastore(data2, OutputType="same",ReadSize=100);

    index = 1;
    c = cell([ceil(numpartitions(arrds)/100) 1]);

    tic;
    while hasdata(arrds)
        c{index} = read(arrds);
        index = index + 1;
    end
    toc;
end
```

The approximate execution times are:

**R2022a:** 0.90 seconds

**R2022b:** 0.20 seconds

The code was timed on a Windows 10, Intel Xeon (R) W-2133 @ 3.60 GHz test system by calling the `timingTest` function.

## parquetread Function: Improved performance reading string data from Parquet files

The `parquetread` function shows improved performance when reading string data from a Parquet file. For example, this code is about 1.23x faster than in the previous release:

```
function t = timingTest
    info = parquetinfo("airlinesmall.parquet");
    varNames = info.VariableNames(info.VariableTypes == "string");

    tic;
    for i = 1:20
        t = parquetread("airlinesmall.parquet",SelectedVariableNames=varNames);
    end
    toc;
end
```

The approximate execution times are:

**R2022a:** 4.93 seconds

**R2022b:** 4.00 seconds

The code was timed on a Windows 10, Intel Xeon (R) W-2133 @ 3.60 GHz test system by calling the `timingTest` function.

## **parquetDatastore Function: Improved performance when creating datastore with large number of Parquet files**

The `parquetDatastore` function shows improved performance when creating a datastore with a large number of Parquet files. For example, this code runs about 2.5x faster than in the previous release:

```
function pds = timingTest
    filenames = repmat("airlinesmall.parquet",1000,1);

    tic;
    for i = 1:20
        pds = parquetDatastore(filenames);
    end
    toc;
end
```

The approximate execution times are:

**R2022a:** 105.31 seconds

**R2022b:** 42.03 seconds

The code was timed on a Windows 10, Intel Xeon (R) W-2133 @ 3.60 GHz test system by calling the `timingTest` function.

## **Workspace Browser: Improved performance of variable-deletion dialog boxes in MATLAB Online**

In MATLAB Online, confirmation dialog boxes for interactively deleting variables from the MATLAB Workspace show improved performance. The delay between selecting **Delete** or **Clear Workspace** for variables in the Workspace browser and the confirmation dialog box appearing is reduced.

For example, on a Windows 10, Intel Xeon CPU E5-1650 v4 @ 3.60 GHz test system, when you right-click a single variable in the workspace and select **Delete**, the confirmation dialog box appears more quickly in R2022b than in R2022a.

## **Variables Editor and Live Editor: Improved speed of data display when scrolling in MATLAB Online**

In MATLAB Online, for variables in the Variables editor or in the generated output of the Live Editor, the performance of vertical and horizontal scrolling is improved. Improved infinite scrolling queries data more efficiently, so data appears more quickly after scrolling in R2022b than in the previous release.



For example, on a Windows 10, Intel Xeon CPU E5-1650 v4 @ 3.60 GHz test system, when you scroll through a table with 3000 rows, the displayed data renders more quickly in R2022b than in R2022a.

## lookfor Function: Improved performance when searching

The `lookfor` function shows improved performance when searching for keywords.

For example, if you use the `lookfor` function to search for the keyword `inverse` in the summary line of all the references pages in the MathWorks documentation, search results appear about 36x faster than in the previous release.

```
function timingTest
lookfor inverse
end
```

The approximate execution times are:

**R2022a:** 6.8885 s

**R2022b:** 0.1909 s

Similarly, if you use the `lookfor` function to search for the keyword `inverse` in the summary line and the Syntax, Description, Input Arguments, Output Arguments, and See Also sections of all the reference pages in the MathWorks documentation, search results appear about 5x faster than in the previous release.

```
function timingTest
lookfor inverse -all
end
```

The approximate execution times are:

**R2022a:** 9.5596 s

**R2022b:** 1.7926 s

The code was timed on a Windows 10, Intel Xeon CPU W-2133 @ 3.60 GHz test system using the `timeit` function.

```
timeit(@timingTest)
```

## Software Development Tools

### Build Tool: Create and run software-build tasks

You can use a standard programming interface to create and run software-build tasks in a uniform and efficient way. For example, you can create tasks that identify code issues, run tests, and package a toolbox in a single build file in your project root folder, and then invoke the build tool to run these tasks. For more information, see [Overview of MATLAB Build Tool](#).

### Dependency Analyzer: New warnings to identify problems

When you run a dependency analysis, the Dependency Analyzer now warns about files that contain a syntax error. See [Investigate and Resolve Problems](#).

### Project API: Extract project from archive

You can now extract a project from an archive by using `matlab.project.extractProject`.

### Source Control in MATLAB Online: Manage Git branches and repositories

You can now manage Git branches and repositories in MATLAB Online:

- Create, switch, merge, and delete branches.
- Find and compare commits.
- Create branches from a tag or a revision.
- View the history of a Git repository.
- Manage multiple Git repositories at once.

### Unit Testing Framework: Add methods and properties to test classes interactively

You can add code to your test classes interactively. With a file defining your class-based test in the Editor, go to the **Editor** tab and in the **Test** section, choose whether to add a method or parameterization property at the test level, method-setup level, or class-setup level. For more information, see [Insert Test Code Using Editor](#).

### Unit Testing Framework: Specify action to take against invalid test files

You can specify whether the framework issues a warning or throws an error when it encounters an invalid test file in a folder or package. To specify the action, use the `InvalidFileFoundAction` name-value argument. For example, `s = testsuite(pwd,InvalidFileFoundAction="error")` creates a test suite only if the current folder does not contain invalid test files and throws an error otherwise. Examples of invalid test files include a test file that contains syntax errors, a function-based test file that is missing local functions, and a file with a `Test` method that is passed an undefined parameterization property.

You can specify the `InvalidFileFoundAction` name-value argument for these methods and functions:

- `matlab.unittest.TestSuite.fromFolder`
- `matlab.unittest.TestSuite.fromPackage`
- `testsuite`
- `runtests`
- `runperf`

## ▲ Unit Testing Framework: Generate descriptive test parameter names using cell arrays

Starting in R2022b, when you assign a nonempty cell array to a parameterization property, the testing framework generates parameter names from the elements of the cell array by taking into account their values, types, and dimensions. In previous releases, if the property value is a cell array of character vectors, the framework generates parameter names from the values in the cell array. Otherwise, the framework specifies parameter names as `value1`, `value2`, ..., `valueN`.

For example, create a test class with parameterization properties that are set using cell arrays.

```
classdef SampleTest < matlab.unittest.TestCase
    properties (TestParameter)
        numericArray = {int16(1),single(zeros(1,4)),magic(3)};
        functionHandle = {@false,@() size([])};
    end
    methods (Test)
        function test1(testCase,numericArray)
            testCase.verifyNotEmpty(numericArray)
        end
        function test2(testCase,functionHandle)
            testCase.verifyWarningFree(functionHandle)
        end
    end
end
```

Create a test suite from the class and return the names of the suite elements.

| R2022a and Earlier                                                                                                                                                                                                                                                                                      | Starting in R2022b                                                                                                                                                                                                                                                                                                        |
|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <pre>suite = testsuite("SampleTest"); {suite.Name}'  ans =  5×1 cell array  {'SampleTest/test1(numericArray=value1)'} {'SampleTest/test1(numericArray=value2)'} {'SampleTest/test1(numericArray=value3)'} {'SampleTest/test2(functionHandle=value1)'} {'SampleTest/test2(functionHandle=value2)'}</pre> | <pre>suite = testsuite("SampleTest"); {suite.Name}'  ans =  5×1 cell array  {'SampleTest/test1(numericArray=int16_1)'} {'SampleTest/test1(numericArray=1x4_single)'} {'SampleTest/test1(numericArray=3x3_double)'} {'SampleTest/test2(functionHandle=@false)'} {'SampleTest/test2(functionHandle=function_handle)'}</pre> |

⚠Compatibility Considerations

If you use parameter names in your code to create or filter test suites, use the descriptive parameter names. This table shows an example of how to update code that filters the test suite created from the `SampleTest` class.

| Before                                                                                                                            | After                                                                                                                                        |
|-----------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------|
| <pre>import matlab.unittest.selectors.HasParameter s = ~HasParameter("Property","numericArray") suite2 = selectIf(suite,s);</pre> | <pre>import matlab.unittest.selectors.HasParameter s = ~HasParameter("Property","numericArray","Name","i") suite2 = selectIf(suite,s);</pre> |

Unit Testing Framework: Compare dictionaries in tests

The `IsEqualTo` constraint supports MATLAB dictionaries. The constraint treats two dictionaries as equal if they have the same keys and the values assigned to corresponding keys also satisfy the constraint. For example, this test passes.

```
import matlab.unittest.TestCase
import matlab.unittest.constraints.IsEqualTo
import matlab.unittest.constraints.DictionaryComparator
import matlab.unittest.constraints.NumericComparator

testCase = TestCase.forInteractiveUse;
actual = dictionary(["key1" "key2"],[1 2]);
expected = dictionary("key1",1,"key2",2);
testCase.verifyThat(actual,IsEqualTo(expected, ...
    Using=DictionaryComparator(NumericComparator)))
```

To override how dictionaries are compared by `IsEqualTo`, use the `matlab.unittest.constraints.DictionaryComparator` class.

⚠Functionality being removed or changed

pack function has been removed  
*Errors*

The `pack` function has been removed. There is no replacement for this function because you do not need to use it on a 64-bit system. For more information about strategies for reducing memory usage, see [Strategies for Efficient Use of Memory and Resolve “Out of Memory” Errors](#).

Interactively running a test class requires the class definition file to explicitly use framework-specific data  
*Behavior change*

When you open a class-based test file in the MATLAB Editor, you can interactively run the tests in the file only if the test class explicitly subclasses a `TestCase` class, uses framework-specific attributes to specify test content, or both. In previous releases, the testing framework tries to examine the class hierarchy whenever it cannot determine if a class definition file contains a test class. This change of behavior prevents checks that might take a long time to complete.

For example, consider an abstract test class defined to test if a function supports implicit expansion of a scalar input.

```

classdef ScalarExpansionTest < matlab.unittest.TestCase
    properties (Abstract)
        FcnHandle
    end
    methods (Test)
        function supportsScalarExpansion(testCase)
            fcn = testCase.FcnHandle;
            result = fcn(1,1:5);
            testCase.verifySize(result,[1 5], ...
                "Result must be a 1-by-5 vector.");
        end
    end
end

```

To test scalar expansion with a specific function interactively, make sure to specify framework-specific data when you create a concrete test class from `ScalarExpansionTest`. If your test class does not require any framework-specific attributes (such as `Test`, `TestParameter`, and `TestTags`), consider deriving it explicitly from `matlab.unittest.TestCase`. In this example, you can run `PlusTest` interactively because the framework can determine the class type using the contents of the class definition file.

```

classdef PlusTest < ScalarExpansionTest & matlab.unittest.TestCase
    properties
        FcnHandle = @plus;
    end
end

```

## External Language Interfaces

### **.NET Interface: Support for .NET 5 and .NET Core**

On Microsoft Windows platforms, MATLAB supports loading libraries compiled for .NET 5 and higher and for .NET Core, in addition to existing support for the Microsoft .NET Framework. To select a .NET environment, use the `dotnetenv` function. For more information, see [Selecting .NET Core in MATLAB](#).

### **.NET Engine API: Call MATLAB from .NET applications**

The MATLAB Engine API for .NET provides an interface between .NET programming languages and MATLAB. This API enables programs to launch MATLAB, evaluate MATLAB functions with arguments, and exchange data between MATLAB and .NET programs. For more information, see [Call MATLAB from .NET](#) and [MathWorks.MATLAB.Engine.MATLABEngine](#).

If you write code that calls MATLAB as a COM Automation server using the MATLAB COM Automation Server Interface, consider using the MATLAB Engine API for .NET instead.

### **Publish C++ Interface: Support for `char*` output as `nullTerminated string`**

You can specify a `nullTerminated string` for `char*` and `const char*` return types. For details, see the **C++ Return Type** table with the C++ to MATLAB data type mapping for C++ `char*` and `char[]` types.

### **Call MATLAB from C++: Use MATLAB enumeration classes in strongly-typed interface**

To generate C++ code from a MATLAB enumeration, use the `Classes` name-value argument with the `matlab.engine.typedinterface.generateCPP` function. For example, to create a header file `InterfaceCode.hpp` from a MATLAB enumeration class `MyColorEnum` with a function `printMyText`, type:

```
matlab.engine.typedinterface.generateCPP("InterfaceCode.hpp",Classes="MyColorEnum",Functions="printMyText")
```

### **Python Interface: Display keyword (key-value pair) arguments created by `pyargs`**

The `pyargs` function displays keyword (key-value pair) arguments.

```
pyargs(A=1,B=2)
```

```
ans =
```

```
    'pyargs' with pairs:
```

```
    A: 1
    B: 2
```

## Python Interface: Convert scalar logical and numeric Python types to MATLAB types

You can convert Python data types `py.int`, `py.long`, `py.float`, and `py.bool` to MATLAB types using these MATLAB functions:

- `double`
- `single`
- `int8`
- `uint8`
- `int16`
- `uint16`
- `int32`
- `uint32`
- `int64`
- `uint64`
- `logical`

For information about conversion functions, see [MATLAB to Python Data Type Mapping](#).

## Python Engine: Pass NumPy arrays directly to MATLAB functions

Calls to MATLAB functions from Python engine applications accept any object that implements the Python buffer protocol, like NumPy arrays, as an input argument.

For example, this code calls the built-in MATLAB function `sum` with a NumPy ndarray `buf` as input.

```
import matlab.engine
import numpy
eng = matlab.engine.start_matlab()
buf = numpy.array([[1, 2, 3], [4, 5, 6]], dtype='uint16')

# Supported in R2022a and earlier: must initialize a matlab.uint16 from
# the numpy array and pass it to the function
array_as_matlab_uint16 = matlab.uint16(buf)
res = eng.sum(array_as_matlab_uint16, 1, 'native')
print(res)

# Supported as of R2022b: can pass the numpy array
# directly to the function
res = eng.sum(buf, 1, 'native')
print(res)
```

For information about using the MATLAB Engine API for Python, see [Call MATLAB from Python](#).

## Python Engine: Install MATLAB Engine API with pip Command

You can use the `pip` command to install the MATLAB Engine API. For more information, see [Install MATLAB Engine API for Python](#).

## Python: Support for Version 3.10

MATLAB now supports CPython version 3.10, in addition to existing support for versions 2.7, 3.8, and 3.9. For more information, see [Versions of Python Compatible with MATLAB Products by Release](#).

### **Functionality being removed or changed**

#### **C++ library definition file has .m file extension**

*Behavior change*

The `clibgen.generateLibraryDefinition` function creates library definition files with `.m` file extensions. The function no longer creates a file with the `.mlx` file extension. You can continue to modify and build existing library definition files with the `.mlx` file extension.

For information about editing the library definition file, see [How to Complete Definitions in Library Definition File](#). While following the instructions, note that when you uncomment the code defining a function, do not uncomment the first two lines in the code section. These comments contain the section title and the C++ signature help.

#### **Python: Version 2.7 support will be discontinued in a future release**

*Behavior change in future release*

Support for Python version 2.7 will be discontinued in a future release.

To ensure continued support for interacting between Python and MATLAB, consider upgrading to a supported version of Python. For supported version information, see [Versions of Python Compatible with MATLAB Products by Release](#).



## Hardware Support

### **Upload Libraries and Configure Pins for I2C, SPI, and Serial Communication Using the Arduino Explorer App**

The Arduino Explorer app, which was introduced in R2021b release of MATLAB Support Package for Arduino Hardware, now supports selecting and uploading the required libraries for the initial setup of the Arduino board. The modified app also allows you to configure pins on the Arduino board for I2C, SPI, and Serial communication, and plot the values. You can also add decode logic on the read values and view the values in the Log panel of the app.

### **▲Support for servo and rotary encoder peripherals connected to Arduino-compatible ESP32 boards**

In addition to the existing support for communicating with I2C, SPI and serial peripherals connected to Arduino-compatible ESP32 board, the MATLAB Support Package for Arduino Hardware now supports communicating with servo and rotary encoder peripherals connected to the board.



# R2022a

---

**Version: 9.12**

**New Features**

**Bug Fixes**

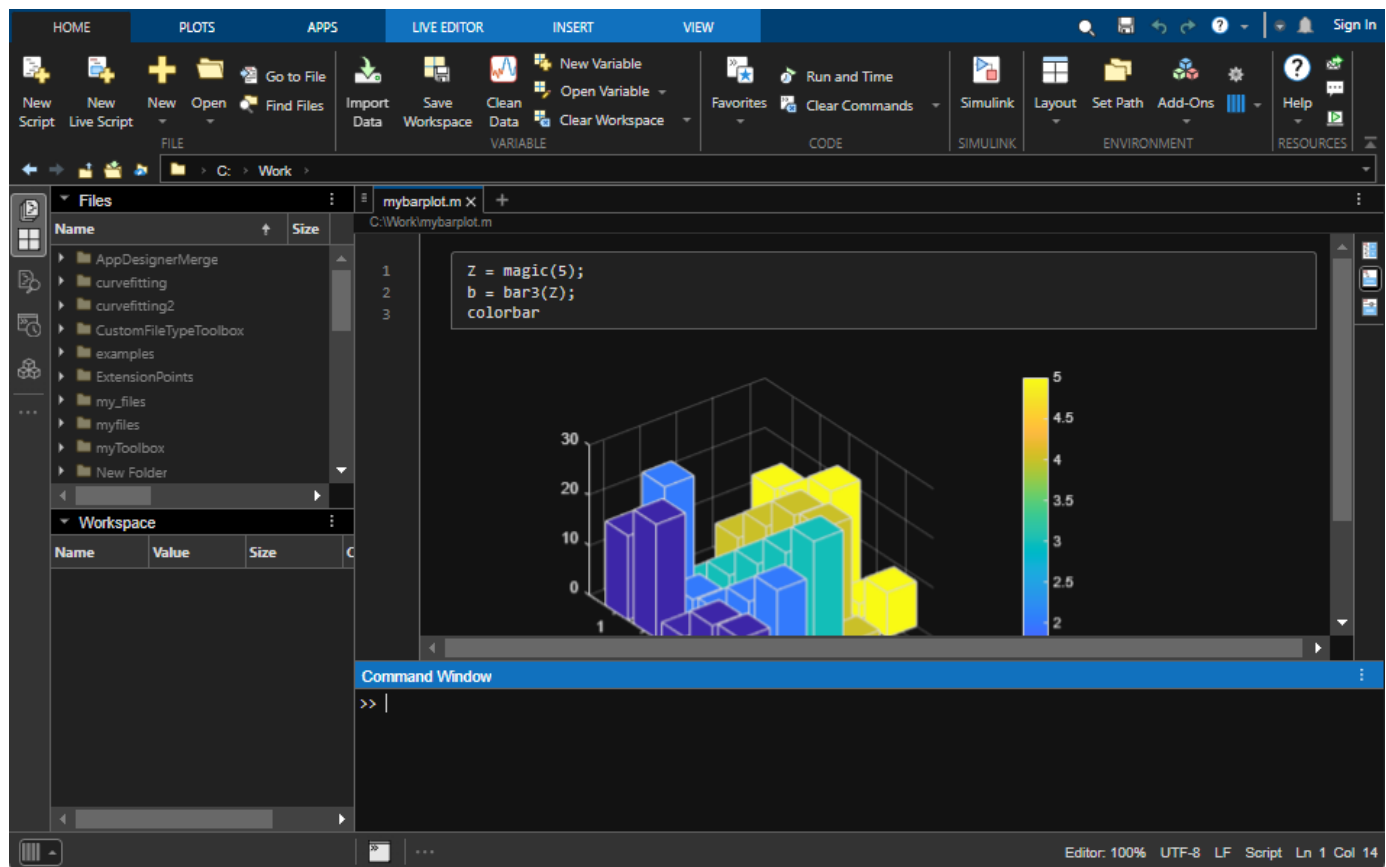
**Compatibility Considerations**

## Environment

### Themes in MATLAB Online: Change the colors of the MATLAB desktop by selecting a dark or light theme

In MATLAB Online, you can change the colors of the MATLAB desktop using themes.

For example, to select a dark theme, on the **Home** tab, in the **Environment** section, click  **Preferences**. Select **MATLAB > Appearance** and set the **Theme** to **Dark**.



To further customize the colors of the MATLAB desktop, select **MATLAB > Appearance > Colors**. Then, change the colors in the **Desktop tool colors**, **MATLAB syntax highlighting colors**, and **MATLAB output colors** sections.

As part of this change, icons in the MATLAB Online desktop have an improved visual appearance.

For more information about changing the colors of the MATLAB desktop, see [Change Desktop Colors](#).

### Live Editor Colors: Change the text and background colors of live scripts and functions

You can change the text and background colors in the Live Editor by changing the MATLAB desktop tool colors.

To change the text and background colors:

- 1 On the **Home** tab, in the **Environment** section, click  **Preferences**.
- 2 Select **MATLAB > Colors**  
In MATLAB Online, select **MATLAB > Appearance > Colors**.
- 3 In the **Desktop tool colors** section, clear the **Use system colors** check box.  
In MATLAB Online, the **Use system colors** check box is not available and this step can be skipped.
- 4 Use the **Text** and **Background** fields to change the colors. For example, select white for the text color and black for the background color.

The Live Editor automatically selects colors for titles and headings based on the selected background color. To further customize the colors of titles and headings, use settings. For more information, see `matlab.fonts` Settings.

## Live Editor Hyperlinks: Insert hyperlinks to specific locations in separate live scripts or live functions

Use hyperlinks to navigate to a location in a separate, existing live script or function. To insert a hyperlink, select the text to link in the current file, go to the **Insert** tab, and click  **Hyperlink**. Edit your display text (optional), select **Location in existing document**, and enter or browse for the file path. Then, select a location in the document preview that displays on the right.

For more information about inserting hyperlinks, see *Format Text* in the Live Editor.

## Live Editor Export: Export live scripts and functions programmatically using the export function

Use the `export` function to programmatically export live scripts and functions to a standard format. Available formats include PDF, Microsoft Word, HTML, and LaTeX.

For example, to export the live script `homework1.mlx` as HTML, type:

```
export("homework1.mlx","homework1.html")
```

## Live Editor Accessibility: Interact with output in live scripts using the keyboard

You can now use keyboard shortcuts to interact with output in live scripts when output is on the right. To move focus from the code to the output display panel, press **Ctrl+Shift+O**. On macOS, press **Option+Command+O**. To activate an output, press **Enter**. Once an output is activated, you can scroll text using the arrow keys, navigate through hyperlinks and buttons using the **Tab** key, and open the context menu by pressing **Shift+F10**.

## Live Editor Tasks: View and interact with tasks when code is hidden

When you hide code in a live script, the Live Editor now displays Live Editor tasks along with formatted text, labeled controls, and output.

To hide code, select the Hide Code button  to the right of the live script or in the **View** tab. Alternatively, if you are using the `export` function, you can hide the code using the `HideCode` name-value argument.

If a Live Editor task is configured to show only code and no controls, then the task does not display when you hide code.

## Component Browser: Reorder children in App Designer or the Property Inspector

You can now drag one or more Axes object children, Group object children, or Transform object children sharing the same parent to reorder them in the Component Browser in App Designer or the Property Inspector for figures.

When reordering children, use visual feedback for allowed moves. You can undo and redo the reordering of the children with **Undo** or **Redo** or the corresponding keyboard shortcuts.

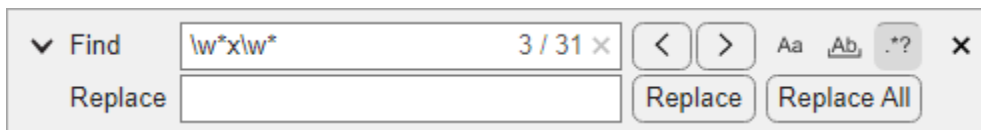
## Editor Python Support: View and edit Python files with syntax highlighting, auto-indenting, and delimiter matching

The Editor now displays Python files with syntax highlighting for keywords, strings, comments, and errors. In addition, the Editor auto-indents Python files and indicates matched and mismatched delimiters such as parentheses, brackets, and braces.

## Find and Replace Dialog Box: Search text in the Editor and Live Editor using regular expressions

You can use a subset of regular expressions to search for text that matches a pattern in an open file in the Editor or Live Editor. To search using a regular expression, on the **Editor** or **Live Editor** tab, in the **Navigate** section, click  Find. Then, in the find and replace dialog box, enter a regular expression, and select the Regular Expression button .

For example, to find all the words in a file that contain the letter x, enter the expression `\w*x\w*` and select the Regular Expression button .



For more information, see Find and Replace Text in Files and Go to Location.

## Profiler: Access the Profiler from the Apps tab

The MATLAB Profiler is now available as an app and can be found in the MATLAB section of the apps gallery in the **Apps** tab.

You can still access the Profiler from the **Home** tab, in the **Code** section, by clicking the **Run and Time** button, or programmatically using the `profile` function.

## Internationalization: UTF-8 system encoding on Windows platforms

MATLAB now uses UTF-8 as its system encoding on Windows, completing the adoption of Unicode across all supported platforms. MATLAB has used UTF-8 as the default encoding for MATLAB files and file I/O since R2020a.

If you see garbled characters on a Windows Server® 2019 platform, then enable the **Beta: Use Unicode UTF-8 for worldwide language support** option in **Region Settings**.

## Installation Settings: Configure persistent settings for MATLAB installations

Installation-level settings provide a new layer of MATLAB configuration that lies between factory settings and personal settings. Installation settings override the factory settings for all users of a given MATLAB installation, and they are persistent across sessions.

In previous versions of MATLAB, if administrators wanted to limit RAM usage for a set of MATLAB users by lowering the `ArraySizeLimit`, they had to create and distribute a script to change the personal setting for each individual user. Starting in R2022a, the administrator can apply the change to all users of their MATLAB installation using the installation setting for `ArraySizeLimit`.

Access installation-level settings using the new `InstallationValue` property of the `Setting` object. Verify and clear installation settings with two new object functions, `hasInstallationValue` and `clearInstallationValue`, respectively.

## Comparison Tool: Save results as HTML report

You can now use the Comparison Tool to publish text comparison results in an HTML report. For more details, see [Compare Text Files](#).

## Comparison Tool: Compare folders in MATLAB Online

Starting in R2022a, you can compare folders and zip files in MATLAB Online.

You can access the comparison tool from:

- The MATLAB Current Folder browser context menu
- The Current Project browser context menu
- The MATLAB Command Window using the `visdiff` function

## MATLAB Drive: macOS 10.15 Catalina will no longer receive updates to MATLAB Drive Connector (April 2022)

After this release, macOS 10.15 Catalina will no longer receive updates to MATLAB Drive Connector.

Installing later versions of MATLAB Drive Connector on macOS 10.15 Catalina will not be supported. In addition, the Connector will not automatically update to a later version when one becomes available.

## ⚠️ **Functionality being removed or changed**

### **Live Editor figure size is bounded upon saving**

#### *Behavior change*

When opening a saved live script, existing images in the output have a maximum size equivalent to the figure size upon saving. To adjust the size past this maximum limit, you can run the live script and increase the figure size.



## Language and Programming

### **Class Introspection: Description and DetailedDescription properties of metaclasses contain text from code comments**

The `Description` and `DetailedDescription` properties of these metaclasses pull content from code comments:

- `meta.class`
- `meta.method`
- `meta.property`
- `meta.event`
- `meta.EnumeratedValue`

For user-defined classes with appropriately placed code comments, the `Description` and `DetailedDescription` properties of the metaclasses are populated with text pulled from those comments. For more information on how to use code comments to store custom help text for user-defined classes, see [Custom Help Text](#).

### **Class Introspection: Access class aliases from meta.class instance**

The aliases of a class are stored in the new `Aliases` property of `meta.class`. For more information on class aliasing, see [Creating and Managing Class Aliases](#).

### **Background Pool: See futures in the background**

Starting in R2022a, you can query all queued and running futures in the background by using the `FevalQueue` property of the pool. To create futures, use `parfeval` and `parfevalOnAll`. For more information on futures, see [Future](#).

### **cancelAll Method: Cancel currently queued and running futures in the background pool**

`cancelAll` cancels all futures currently queued or running in the background pool. Queued or running futures are listed in the `FevalQueue` property.

### **Background Pool: Check the status of the background pool**

Starting in R2022a, you can query to determine if the background pool is currently running by using the `Busy` property of the pool. This property indicates whether the background pool is busy, specified as `true` or `false`. The pool is busy if there is outstanding work for the pool to complete.

### **pcode Function: Create P-code files with enhanced obfuscation**

The `pcode` function now has the option `"-R2022a"`, which creates P-code files using a more complex obfuscation algorithm. Files created with this option run only in MATLAB releases R2022a and later.

**str2num Function: Restrict evaluation to basic math expressions**

str2num is implemented using the eval function, which evaluates the input argument. Starting in R2022a, you can set the name-value argument Evaluation to "restricted" to restrict accepted inputs to basic math expressions, such as 200 and 1+2i.

**assert Function: Output displays which assertion threw an error and the location in the code**

When an assertion fails, the error thrown includes the specific assertion that failed and the location in the code.

| Previous Output                                                                                                    | New Output                                                                                                                                                                              |
|--------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Error using repro>checkScalarInteger<br>Assertion failed.<br><br>Error in repro (line 4)<br>checkScalarInteger(pi) | Error using assert<br>Assertion failed.<br><br>Error in assert_test>checkScalarInteger (line 6)<br>assert(x == floor(x))<br><br>Error in assert_test (line 3)<br>checkScalarInteger(pi) |

**⚠️Functionality being removed or changed**

**Defining classes and packages: Using schema.m will not be supported in a future release**  
*Still runs*

Support for classes and packages defined using schema.m files will be removed in a future release. Replace existing schema-based classes with classes defined using the classdef keyword.

**dec2hex and dec2bin input types are now restricted**  
*Behavior change*

Input types allowed by dec2hex and dec2bin have been restricted. Supported input types are primitive numeric types and classes that inherit from a primitive numeric type.

In addition, dec2bin(0,0) will now return '0' rather than a 1x0 character vector.

**cd no longer removes leading spaces for Windows drive letter paths**  
*Behavior change*

Before R2022a, on Windows platforms, the cd function removed leading spaces in input paths specifying the drive letter. Input paths containing leading spaces now cause an error to be thrown instead. If an input path is invalid with leading spaces, then use strip to remove the spaces before using the cd function.

**TruncateScalarObject name-value argument for widthConstrainedDataRepresentation method renamed to AllowTruncatedDisplayForScalar**  
*Behavior change in future release*

The name of the TruncateScalarObject name-value argument for the widthConstrainedDataRepresentation method is now AllowTruncatedDisplayForScalar.

The functionality of the option will not change. Support for the name `TruncateScalarObject` will be removed in a future release.

### **cast returns consistent output for subclass of MATLAB numeric types**

#### *Behavior change*

The syntax `cast(A, "like", p)` now returns output consistent with the prototype `p` when the data type of `p` is a subclass of MATLAB numeric types.

For example, this code returns an output that has the same data type as `p`:

```
p = matlab.lang.OnOffSwitchState.on;  
x = cast(1, "like", p)  
  
x =
```

```
OnOffSwitchState enumeration
```

```
on
```

In previous releases, the code returns `x = 1` with data type `logical`.

### **Error reports will no longer include line number**

#### *Behavior change*

Thrown error reports will no longer include the line number of where the error occurred.

## Data Analysis

### Data Cleaner App: Interactively preprocess and organize column-oriented data

The new Data Cleaner app enables you to:

- Access column-oriented data in the MATLAB workspace or import column-oriented data from a file.
- Explore data by using the visualization, data, and summary views.
- Sort by a variable, rename a variable, or remove a variable.
- Retime data in a timetable, stack or unstack table variables, clean missing data, clean outlier data, smooth data, or normalize data.
- Edit previously performed cleaning steps by using the **Cleaning Steps** panel.
- Export cleaned data to the MATLAB workspace, or export code for cleaning data as a script or function.

You can open the Data Cleaner app from the MATLAB section of the apps gallery in the **Apps** tab. Alternatively, enter `dataCleaner` in the MATLAB command window.

The Data Cleaner app currently supports cleaning only timetable data and importing only one timetable at a time.

### **allfinite, anynan, and anymissing Functions: Determine if all array elements are finite, any element is NaN, and any element is missing**

Use the `allfinite`, `anynan`, and `anymissing` functions to examine the elements of an input array.

- `allfinite`: Determine if all array elements are finite.
- `anynan`: Determine if any array element is NaN.
- `anymissing`: Determine if any array element is missing.

### **quantile, prctile, and iqr Functions: Calculate quantiles, percentiles, and interquartile range**

Calculate quantiles, percentiles, and the interquartile range of a data set by using the `quantile`, `prctile`, and `iqr` functions.

Previously, the `quantile`, `prctile`, and `iqr` functions required Statistics and Machine Learning Toolbox.

### **rms Function: Calculate root-mean-square value**

Calculate the root-mean-square (RMS) value of input data with `rms`.

You can specify the dimensions to operate along and whether to include or omit NaN values in the calculation:

- Use "all" to calculate the RMS value of all elements of the input array.
- Use the `dim` input argument to calculate the RMS value along one dimension.
- Use the `vecdim` input argument to calculate the RMS value along multiple dimensions.
- Use "includenan" or "omitnan" to include or omit NaN values in the RMS calculation.

Previously, the `rms` function required Signal Processing Toolbox™.

## std and var Functions: Optionally return mean as a second output

The `std` and `var` functions can now return the mean of the elements used to calculate the standard deviation or variance by using a second output argument `M`. If a weighting scheme is specified, then the weighted mean is returned.

## ⚠ Date and Time Functions: Some Financial Toolbox functions combined with MATLAB functions

The following date and time functions from Financial Toolbox™ are combined with functions having the same names in MATLAB. Before R2022a, these Financial Toolbox functions supported serial date numbers and text timestamps as inputs, while the MATLAB functions supported `datetime` arrays. Starting in R2022a, the MATLAB functions support `datetime` arrays, serial date numbers, and text timestamps as inputs. The functions are removed from Financial Toolbox.

- `day`
- `hour`
- `minute`
- `month`
- `quarter`
- `second`
- `year`

## ⚠ Compatibility Considerations

While these functions support serial date number and text inputs, these types of inputs are not recommended. Use `datetime` values as inputs instead. The `datetime` data type provides flexible date and time formats, storage out to nanosecond precision, and properties to account for time zones and daylight saving time. To convert serial date numbers or text timestamps to `datetime` values, use the `datetime` function.

- To convert serial date numbers to `datetime` values, call `datetime` with the `ConvertFrom` name-value argument:

```
dt = datetime(738457, "ConvertFrom", "datenum")
```

```
dt =
```

```
datetime
```

```
28-Oct-2021
```

- To convert text timestamps, call `datetime`.

```
dt = datetime("2021-10-28")
```

```
dt =
```

```
datetime
```

```
28-Oct-2021
```

There are no plans to remove support for serial date numbers or text timestamps from these MATLAB functions.

## ⚠ Date and Time Functions: Some Financial Toolbox functions moved to MATLAB

These date and time functions are removed from Financial Toolbox and moved to MATLAB:

- `eomdate`
- `lweekdate`
- `m2xdate`
- `months`
- `nweekdate`
- `today`
- `weeknum`
- `x2mdate`

## ⚠ Compatibility Considerations

While MATLAB supports these functions, they are not recommended because they use serial date numbers in their calculations. The table shows recommended replacements that either accept or return `datetime` values. The `datetime` data type is recommended because it provides flexible date and time formats, storage out to nanosecond precision, and properties to account for time zones and daylight saving time.

There are no plans to remove these functions from MATLAB.

| Transferred Function   | Recommended Replacement                                                                                              |
|------------------------|----------------------------------------------------------------------------------------------------------------------|
| <code>eomdate</code>   | <code>dateshift</code> , with <code>datetime</code> values as inputs                                                 |
| <code>lweekdate</code> | <code>lweekdate</code> , with <code>outputType</code> specified as "datetime" to return <code>datetime</code> output |
| <code>m2xdate</code>   | <code>exceltime</code> , with <code>datetime</code> values as inputs                                                 |
| <code>months</code>    | <code>between</code> , with <code>datetime</code> values as inputs                                                   |
| <code>nweekdate</code> | <code>nweekdate</code> , with <code>outputType</code> specified as "datetime" to return <code>datetime</code> output |
| <code>today</code>     | <code>datetime</code> , with "today" as the input argument                                                           |
| <code>weeknum</code>   | <code>week</code> , with <code>datetime</code> values as inputs                                                      |

| Transferred Function | Recommended Replacement                      |
|----------------------|----------------------------------------------|
| x2mdate              | datetime, with dateType specified as "excel" |

## matlab.datetime.compatibility.convertDatenum Function: Convert text timestamps and serial date numbers to datetime values in a backward-compatible way

To convert text timestamps and serial date numbers to `datetime` values, use the `matlab.datetime.compatibility.convertDatenum` function. For backward compatibility, this function supports the subset of `datestr` formats that the `datenum` function recognizes when it converts text timestamps without a format specifier.

Use this function in code where you intend to return `datetime` values, but to preserve compatibility you need to interpret text inputs in the same way that `datenum` interprets them. This function is designed to be a compatibility layer for function authors.

To explicitly convert serial date numbers to `datetime` values, use the `datetime` function instead, with the `ConvertFrom` name-value argument:

```
dt = datetime(738457, "ConvertFrom", "datenum")

dt =

    datetime
    28-Oct-2021
```

## categorical Data Type: Use a pattern object to specify category names that match a pattern

When you specify category names of a `categorical` array, you can use a `pattern` object to specify names that match a pattern.

For example, suppose you have a categorical array that has many different categories that can represent "yes" and "no". This categorical array has six values and six categories because the values in the input array are different.

```
C = categorical(["Y" "Yes" "Yeah" "N" "No" "Nope"])

C =

    1x6 categorical array

      Y      Yes     Yeah      N      No      Nope
```

To combine all the different "yes" categories into one category and all the different "no" categories into another category, use the `mergecats` function and wildcard patterns to match the category names. The categorical array still has six values. But it has only two categories, "yes" and "no".

```
C = mergecats(C, "Y" + wildcardPattern, "yes");
C = mergecats(C, "N" + wildcardPattern, "no")

C =
```

```
1x6 categorical array  
  
    yes    yes    yes    no    no    no
```

These functions provide support for using patterns when you specify category names:

- `histcounts` (when you specify the `Categories` argument)
- `iscategory` (when you specify the `catnames` argument)
- `mergocats` (when you specify the `oldcats` argument)
- `removecats` (when you specify the `oldcats` argument)
- `reordercats` (when you specify the `neworder` argument)

**table and timetable Data Types: Use a pattern object to specify row, variable, and property names that match a pattern**

When you specify rows, variables, or properties of a table or timetable, you can use a pattern object to specify names that match a pattern.

You can use patterns when you subscript into a table by row names and variable names, or when you subscript into a timetable by variable names.

For example, read a table into MATLAB.

```
T = readtable("outages.csv","TextType","string")
```

T =

| 1468x6 table |                  |        |            |                  |                |  |
|--------------|------------------|--------|------------|------------------|----------------|--|
| Region       | OutageTime       | Loss   | Customers  | RestorationTime  | Cause          |  |
| "SouthWest"  | 2002-02-01 12:18 | 458.98 | 1.8202e+06 | 2002-02-07 16:50 | "winter storm" |  |
| "SouthEast"  | 2003-01-23 00:49 | 530.14 | 2.1204e+05 | NaT              | "winter storm" |  |
| "SouthEast"  | 2003-02-07 21:15 | 289.4  | 1.4294e+05 | 2003-02-17 08:14 | "winter storm" |  |
| :            | :                | :      | :          | :                | :              |  |

To subscript into the table and select all variables whose names end with "Time", use a wildcard pattern.

```
T2 = T(:,wildcardPattern + "Time")
```

T2 =

| 1468x2 table     |                  |
|------------------|------------------|
| OutageTime       | RestorationTime  |
| 2002-02-01 12:18 | 2002-02-07 16:50 |
| 2003-01-23 00:49 | NaT              |
| 2003-02-07 21:15 | 2003-02-17 08:14 |
| :                | :                |

These functions provide support for using patterns when you specify variables by name:

- `convertvars`



- `innerjoin`
- `issortedrows`
- `join`
- `movevars`
- `mergevars`
- `outerjoin`
- `removevars`
- `rowfun`
- `rows2vars`
- `sortrows`
- `splitvars`
- `stack`
- `topkrows`
- `unstack`
- `varfun`

This function provides support for using patterns when you specify properties by name:

- `rmprop`

### **Data Preprocessing Functions: Append transformed variables to input data using the `ReplaceValues` name-value argument**

When you preprocess tables and timetables, you can now append variables containing the transformed values to the input table. Set the `ReplaceValues` name-value argument to `false` for these functions:

- `smoothdata`
- `normalize`
- `filloutliers`
- `fillmissing`
- `standardizeMissing`

### **Data Preprocessing Functions: Return table with logical values using the `OutputFormat` name-value argument**

When you preprocess tables and timetables, you can now output a table or timetable containing logical values instead of a logical array. Set the `OutputFormat` name-value argument to `"tabular"` for these functions:

- `ischange`
- `islocalmax`
- `islocalmin`
- `ismissing`

- `isoutlier`

## ▲ **ismissing, rmmissing, and groupsummary Functions: Accept data types with no standard missing value**

The `ismissing` syntax `ismissing(A)` now returns logical 0 (false) when the input data type has no default definition of a standard missing value.

`rmmissing` and the `nummissing` and `nnz` methods of `groupsummary` no longer error for input data types with no default definition of a standard missing value.

An example of code that used to error but now executes is:

```
A = [struct struct struct];
TF = ismissing(A)
```

```
TF =
    1x3 logical array
     0     0     0
```

## ▲ **Compatibility Considerations**

Some input types that used to throw an error now execute. If your code relies on the errors that MATLAB threw for those inputs, such as within a `try/catch` block, then your code may no longer catch those errors.

## ▲ **Functionality being removed or changed**

### **Plot a variable multiple times in a stacked plot**

*Behavior change*

You can now display the same table or timetable variable multiple times when you call the `stackedplot` function. In previous releases, specifying a variable more than once results in an error.

For example, create a timetable from the `outages.csv` file. Then plot the `RestorationTime` variable under each of the other variables that you specify.

```
tbl = readtimetable("outages.csv");
tbl = sortrows(tbl);
stackedplot(tbl,["Loss","RestorationTime","Customers","RestorationTime"])
```

### **Live Editor tasks for arrays, tables, and timetables do not run automatically if inputs have more than 1 million elements**

*Behavior change*

Many Live Editor tasks for arrays, tables, and timetables do not run automatically if inputs have more than 1 million elements. In previous releases, these tasks run automatically for input arrays, tables, and timetables of any size. If the inputs have a large number of elements, then the code generated by these tasks can take a noticeable amount of time to run (more than a few seconds).

This change in behavior affects these tasks:

- Join Tables

- Retime Timetable
- Stack Table Variables
- Synchronize Timetables
- Unstack Table Variables

**Live Editor tasks for preprocessing data do not run automatically if inputs have more than 1 million elements**

*Behavior change*

Many Live Editor tasks for preprocessing data do not run automatically if inputs have more than 1 million elements. In previous releases, these tasks run automatically for input data of any size. If the inputs have a large number of elements, then the code generated by these tasks can take a noticeable amount of time to run (more than a few seconds).

This change in behavior affects these tasks:

- Clean Missing Data
- Clean Outlier Data
- Compute by Group
- Find Change Points
- Find Local Extrema
- Normalize Data
- Smooth Data
- Remove Trends

## Data Import and Export

### Parquet: Read Parquet file data more efficiently using rowfilter to conditionally filter rows

Conditionally filter and read data faster (Predicate Pushdown) from Parquet files when using `parquetread` and `parquetDatastore`. You can create conditions for filtering by using the `rowfilter` function, `matlab.io.RowFilter` object, and `RowFilter` name-value argument. Due to its metadata-accelerated processing, the `rowfilter` workflow is the recommended approach for filtering Parquet data to import.

### Parquet: Determine and define row groups in Parquet file data

A Parquet file can store a range of rows as a distinct row group for increased granularity and targeted analysis. `parquetread` uses the `RowGroups` name-value argument to determine row groups while reading Parquet file data. `parquetwrite` uses the `RowGroupHeights` name-value argument to define row groups while writing Parquet file data.

### Parquet: Convert, import, and export nested data structures

Use `parquetread` to import nested Parquet file data with:

- `LogicalType` as `LIST`.
- `LogicalType` as `NONE` and `PhysicalType` as either `BYTE_ARRAY` or `FIXED_LEN_BYTE_ARRAY`.

The `parquetread` function converts and imports these data structures as cell arrays.

Use `parquetwrite` to export nested cell arrays as `LIST` arrays. Nested data is beneficial to working with irregularly structured data such as jagged arrays.

### writelines Function: Write plain text to a file

Use the `writelines` function to write a string array or a cell array of character vectors as plain text to a file. The `writelines` function is the writing equivalent of the `readlines` function.

### Reading Online Data: Use web options when reading files over HTTP and HTTPS

Read files over HTTP and HTTPS using the `weboptions` function and specifying the `WebOptions` name-value argument with these functions:

- `readtable`
- `readtimetable`
- `readvars`
- `readstruct`
- `readmatrix`
- `readcell`

- `readlines`

## **Opus Files: Work with Opus (.opus) audio files.**

Use `audioread`, `audiowrite`, and `audioinfo` to read, write, and analyze Ogg Opus audio files.

## **HDF5 Interface: Write datasets using dynamically loaded filters**

You can read and write HDF5 datasets using dynamically loaded filters with both the high-level and low-level interfaces. For details, see [Import HDF5 Files](#) and [Export to HDF5 Files](#).

The `h5create` function introduces two name-value arguments, `CustomFilterID` and `CustomFilterParameters`, to enable compression using dynamically loaded filters.

## **NetCDF Interface: Enable byte-range reading of remote datasets**

You can now use the existing high-level and low-level interfaces for read-only access to remote datasets using the HTTP byte-range capability. The latter assumes that the remote server supports byte-range access.

## **NetCDF Interface: Read and write variable length array data types (NC\_VLEN)**

You can now use the existing high-level functions to read variable length array data types (NC\_VLEN) from NetCDF-4 files. You can read and write NC\_VLEN types using low-level functions.

Use these additional low-level functions to create NC\_VLEN types and retrieve information about them:

- `netcdf.defVlen`
- `netcdf.inqUserType`
- `netcdf.inqVlen`

## **Scientific File Format Libraries: NetCDF library is upgraded**

The NetCDF library is upgraded to version 4.8.1.

## **Hardware Manager App: Discover and connect to your hardware from MATLAB**

The new Hardware Manager app allows you to discover and connect to your hardware from MATLAB by providing access to the necessary add-ons and apps. For more information, see [Get Started with Hardware Manager](#).

## **TCP/IP Client Interface: Specify transfer delay options**

You can now enable or disable a transfer delay to allow delayed acknowledgement from the connected server for `tcpclient` objects and in the TCP/IP Explorer app. The transfer delay is

enabled by default. Enabling the delay turns on Nagle's algorithm, which causes the client to collect small segments of outstanding data and send them in a single packet when acknowledgement (ACK) arrives from the server. Disabling it turns off Nagle's algorithm, which immediately sends data to the network.

For the `tcpclient` interface, you can set the `EnableTransferDelay` property as a name-value argument during object creation. For TCP/IP Explorer, you can select **Transfer Delay** options during connection configuration.

For more information about this functionality, see `EnableTransferDelay` and `Configure Connection` in TCP/IP Explorer.

## ⚠ **Functionality being removed or changed**

### **seriallist function will be removed**

*Warns*

`seriallist` will be removed. Use `serialportlist` instead. For more information about updating your code to use the recommended functionality, see [Transition Your Code to serialport Interface](#).

### **serial function will be removed**

*Warns*

`serial` and its object properties will be removed. Use `serialport` and its properties instead.

This example shows how to connect to a serial port device using the recommended functionality.

| Functionality                                                        | Use Instead                               |
|----------------------------------------------------------------------|-------------------------------------------|
| <pre>s = serial("COM1");<br/>s.BaudRate = 115200;<br/>fopen(s)</pre> | <pre>s = serialport("COM1",115200);</pre> |

For more information about updating your code to use the recommended functionality, see [Transition Your Code to serialport Interface](#).

### **MATLAB Variable Editor: timeseries will no longer be supported**

*Warns*

Viewing `timeseries` objects using the MATLAB Variable Editor will no longer be supported. To view time-indexed data in the Variable Editor, use `timetable` instead.

## Mathematics

### **pagemldivide, pagemrdivide, and pageinv Functions: Solve linear equations and calculate matrix inverses using pages of N-D arrays**

Use the `pagemldivide`, `pagemrdivide`, and `pageinv` functions to perform linear algebra operations on the pages of N-D arrays. In this context, the N-D array is treated as a container for several 2-D matrices.

- `pagemldivide` and `pagemrdivide`: Solve linear equations using the pages of N-D arrays.
- `pageinv`: Calculate the matrix inverse of the pages of an N-D array.

### **tensorprod Function: Calculate tensor products between two arrays**

Use the `tensorprod` function to calculate tensor products between two N-D arrays. You can perform an inner product, outer product, or a combination of the two by specifying a subset of dimensions to contract (multiply and sum) with each other.

### **▲round Function: Control tiebreak behavior**

The `round` function has a new `TieBreaker` name-value argument to specify how to break ties. You can now specify to round ties away from zero, towards zero, to the nearest even or odd integer, or towards positive or negative infinity.

### **▲Compatibility Considerations**

Starting in R2022a, the `round` function always rounds ties (that are within roundoff errors) away from zero by default. In previous releases, the `round` function sometimes returns inconsistent results, where ties are rounded towards zero by default.

### **null and orth Functions: Specify tolerance to treat singular values below a threshold as zero**

The `null` and `orth` functions now have a second input argument that specifies a tolerance. The tolerance determines which singular values of the input matrix are treated as zero, which can change the number of columns returned by `null` and `orth`.

### **norm Function: Frobenius norm calculations support N-D arrays**

Frobenius norm calculations of the form `norm(X, "fro")` now support N-D arrays. See `norm` for more information.

### **equilibrate Function: Specify output format of factorization**

`equilibrate` now has an option with values of `"vector"` or `"matrix"` to specify whether the output arguments are returned as vectors or matrices. For large factorizations, returning the outputs as vectors can save memory and improve efficiency.

## **rand, randi, and randn Functions: Support for complex input and RandStream object with the "like" syntax**

The `rand`, `randi`, and `randn` functions now support complex input and a `RandStream` object for the "like" syntax.

For example, you can use `X = rand(m,n,"like",p)` to create an `m`-by-`n` array of random numbers of the same data type and complexity (real or complex) as `p`. You can also use `X = rand(s,m,n,"like",p)` to generate random numbers like `p` from the random number stream `s` (`RandStream` object) instead of the default global stream.

## **eps, flintmax, intmax, intmin, realmax, and realmin Functions: Use "like" syntax to return scalars based on prototype object**

The `eps`, `flintmax`, `intmax`, `intmin`, `realmax`, and `realmin` functions now accept the "like" syntax to return scalars based on a prototype object.

For example, you can use `f = realmax("like",p)` to return the largest finite floating-point number with the same data type, sparsity, and complexity (real or complex) as the floating-point variable `p`.

## **qr and gsvd Functions: Option for economy-size decompositions**

`qr` and `gsvd` have a new "econ" option for economy-size decompositions.

- For `qr`, the functionality is the same as `qr(A,0)` unless a third output is specified.
- For `gsvd`, the functionality is the same as `gsvd(A,B,0)`.

## **⚠️ Functionality being removed or changed**

### **One-output qr syntax always returns upper-triangular factor**

*Behavior change*

The syntax `R = qr(A)` always returns `R` as an upper-triangular matrix, regardless of whether `A` is full or sparse. Previously, for full `A`, the one-output syntax returned an `R` matrix with intermediate data used in the calculation located in the lower triangular portion of the matrix. See `qr` for more information.

### **mldivide no longer uses LDL factorization for full matrices**

*Behavior change*

`mldivide` no longer uses an LDL factorization for full matrices that are Hermitian indefinite. Instead, the LU factorization is used for these matrices.



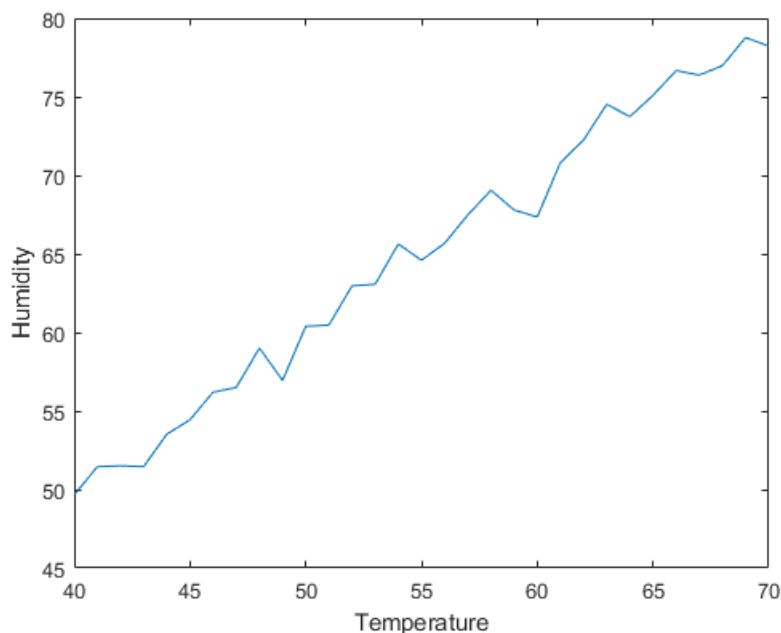
## Graphics

### Plotting Table Data: Create line plots by passing tables directly to plotting functions

Create plots by passing a table directly to any of these functions: `plot`, `plot3`, `loglog`, `semilogx`, `semilogy`, and `polarplot`. When you specify your data as a table, Cartesian axis labels and the legend (if present) are automatically labeled using the table variable names.

For example, create a table with the variables `Temperature` and `Humidity`. Pass the table to the `plot` function as the first argument, and specify the variables you want to plot.

```
Temperature = (40:70)';
Humidity = (50:80)' + randn(31,1);
T = table(Temperature,Humidity);
plot(T,"Humidity","Temperature")
```

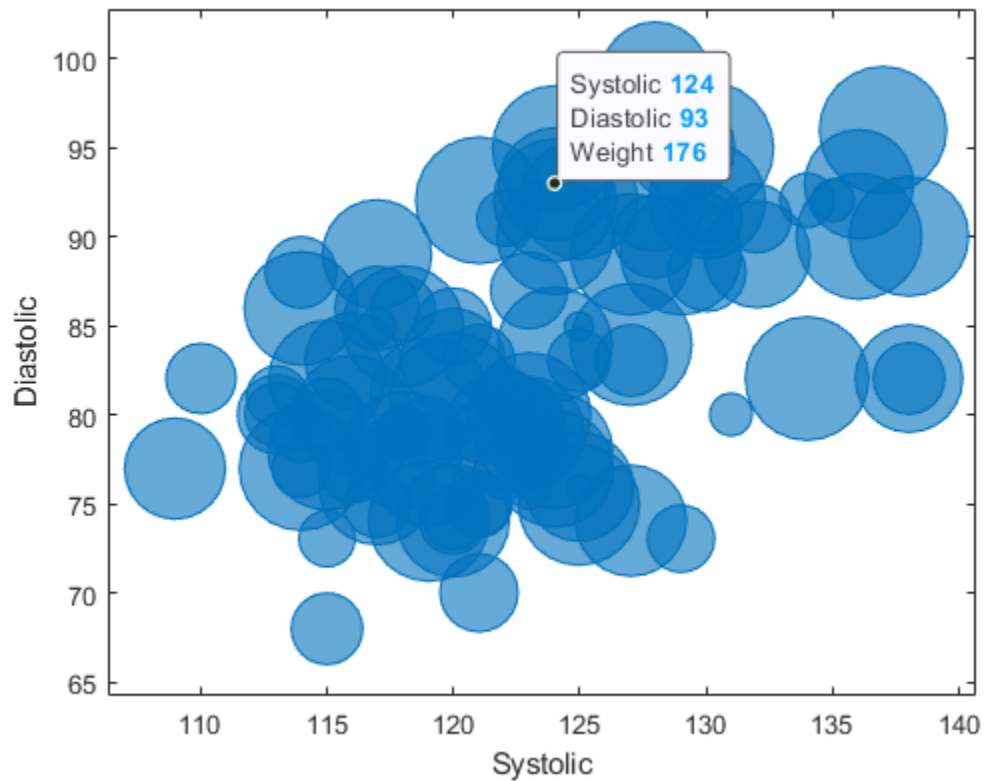


### Data Tips: View table variable names as row labels

When plotting tabular data, the default row labels of data tips created interactively or with the `datatip` function are the names of the table variables associated with the data point.

For example, create a table using the sample file `patients.xls`. Then, plot the `Systolic`, `Diastolic`, and `Weight` variables in a bubble chart. A data tip created with `datatip(b)` displays three rows. The row labels are `"Systolic"`, `"Diastolic"`, and `"Weight"`.

```
tbl = readtable("patients.xls");
b = bubblechart(tbl,"Systolic","Diastolic","Weight");
datatip(b);
```

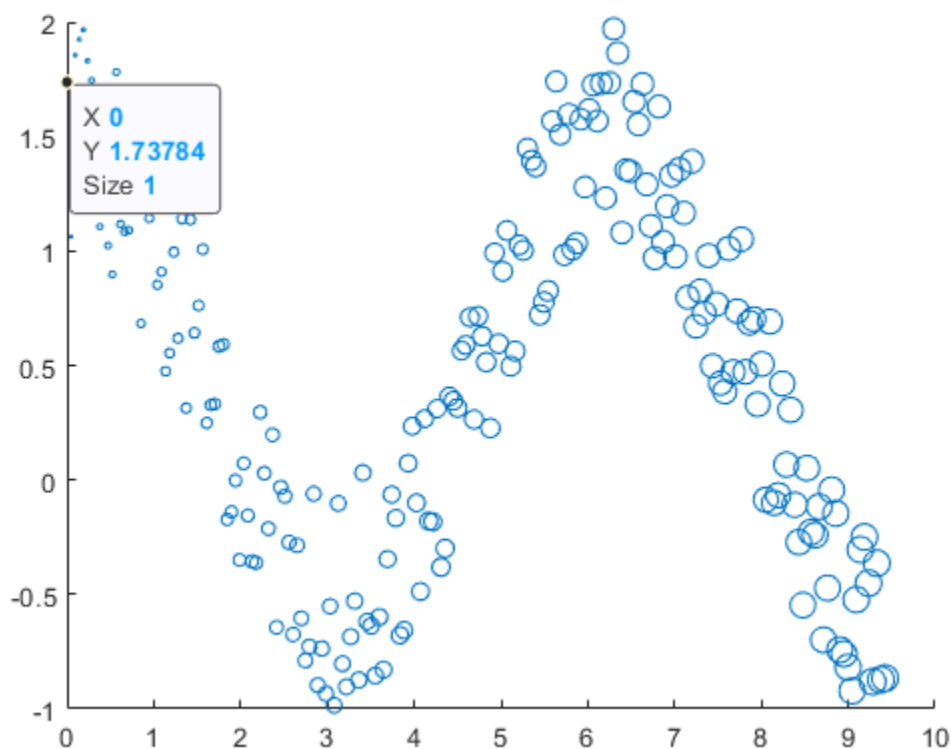


## Data Tips: View visual property values for scatter plots and bubble charts

For scatter plots and bubble charts, data tips created interactively or with the `datatip` function include by default rows for visual properties such as size, color, or transparency that are specified with vector data.

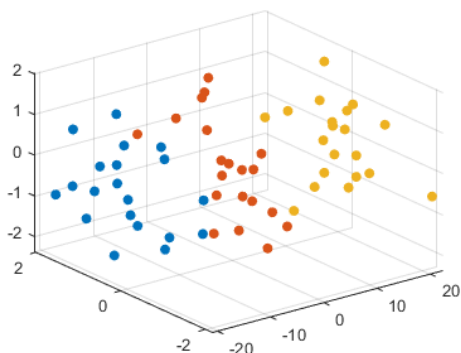
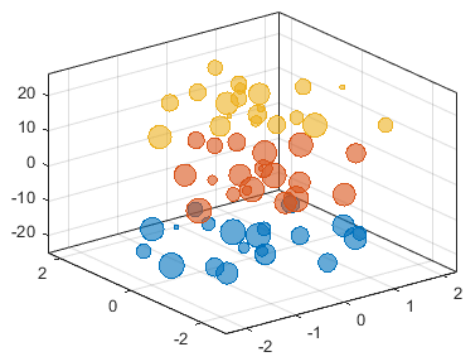
For example, create a scatter plot of random data and define the marker sizes as vector `sz`. A data tip created with `datatip(s)` displays three rows: X, Y, and Size. The Size row in the data tip displays the marker size specified by `sz` for the associated data point.

```
x = linspace(0,3*pi,200);
y = cos(x) + rand(1,200);
sz = linspace(1,100,200);
s = scatter(x,y,sz);
datatip(s);
```



## Bubble Charts and 3-D Scatter Plots: Plot multiple data sets at once

The `bubblechart`, `bubblechart3`, `polarbubblechart`, and `scatter3` functions now accept the same combinations of matrices and vectors as the `plot` function does. As a result, you can visualize multiple data sets at once rather than using the `hold` function between plotting commands.



## fontname and fontsize Functions: Specify the font and font size for graphics objects

Use the `fontname` and `fontsize` functions to modify the fonts displayed with graphics objects such as figures, axes, legends, tiled chart layouts, standalone visualizations, and UI components. MATLAB applies your changes to the specified object and all the objects it contains. For example, if you change the font on a figure, all the axes, annotations, and UI components within the figure use the new font.

## exportgraphics Function: Create animated GIF files

Create animated GIF files by calling the `exportgraphics` function multiple times with the Append name-value argument.

## Annotation Graphics Objects: Change the annotation rotation angle with the Rotation property

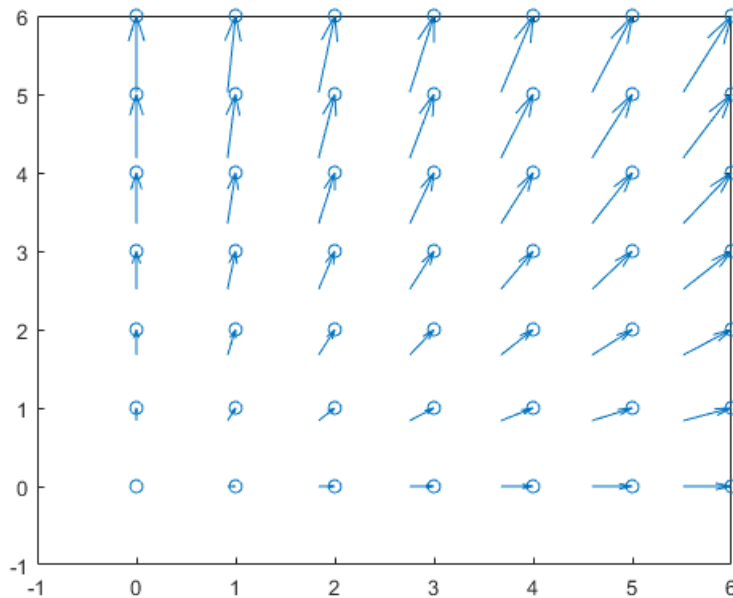
For text box, rectangle, and ellipse annotation objects, rotate the annotation a specified number of degrees by using the `Rotation` property. The anchor point for rotation is the location specified by the first two elements of the `Position` property, so the `Position` property is unaffected by rotation.

For more information, see [TextBox Properties](#), [Rectangle Properties](#), and [Ellipse Properties](#).

## Quiver Plots: Align the heads, centers, or tails of arrows with data points

Set the `Alignment` property of a Quiver object to control how the arrows align with the data points. Valid values are "head", "center", and "tail". For example, plot a grid of vectors with the arrow heads positioned at the data points. Specify a marker symbol to show the locations of the data points.

```
[X,Y] = meshgrid(0:6,0:6);  
U = 0.25*X;  
V = 0.5*Y;  
quiver(X,Y,U,V,"Alignment","head","Marker","o")
```



## **xlim, ylim, and zlim Functions: Query the axis limit method**

Query the method MATLAB uses to set the axis limits by calling the `xlim`, `ylim`, and `zlim` functions and specifying "method" as an input argument.

## **view Function: Change the view on multiple axes simultaneously**

Change the view of multiple axes objects at the same time by passing an array of axes objects to the `view` function.

## **rendererinfo Function: Get renderer information without specifying the axes**

Call the `rendererinfo` function without any arguments to query the default graphics renderer information. This new syntax allows you to call the `rendererinfo` function in a way that is consistent with the `opengl` syntax. Since R2019a, the `rendererinfo` function has been recommended instead of the `opengl` function for querying the renderer.

## **linkaxes Function: Synchronize axes in all dimensions by default**

The `linkaxes` function now supports 3-D Cartesian axes and synchronizes the x-axis, y-axis, and z-axis limits by default. The supported values of the `dimension` input argument are now 'xyz' (default), 'x', 'y', 'z', 'xy', 'xz', 'yz', and 'off'.

Before R2022a, `linkaxes` supported only 2-D Cartesian axes and synchronized the x-axis and y-axis limits by default.

## cameratoolbar Function: Syntax support for figures created with the uifigure function

Syntaxes of the `cameratoolbar` function that do not directly make the toolbar visible are now supported by figures created with the `uifigure` function.

## Callbacks in Live Editor: Create callbacks for figures in the Live Editor

You can now create callbacks for figures created in the Live Editor. The callback workflow supports optional source and event-data parameters.

Keyboard-based callback properties and anonymous function callbacks using `Figure` objects from the MATLAB workspace are not currently supported in the Live Editor.

To define and execute a figure callback in the Live Editor, use one of these techniques:

- Create a figure callback and pass source and event data as parameters in the callback.
- Create a figure callback and do not pass source or event data as a parameter in the callback.
- Create a callback that includes a function for identifying a graphics object, such as `gca` or `findobj`.

For example, define a callback function called `colorchangeCallback`. With the `colorchangeCallback` function on the MATLAB path, use the `@` operator to assign the function handle to the `WindowButtonDownFcn` property of the figure `fig`.

```
fig = figure;  
axis([-4 4 -4 4]);  
plot(1:10)  
fig.WindowButtonDownFcn = @(src,eventdata)colorchangeCallback(src);
```

Define the callback and set the `Color` property for the `Axes` object in the figure:

```
function colorchangeCallback(f,~)  
% Change the axes color on button down  
ax = f.Children;  
ax.Color = rand(1,3);  
end
```

For more information, see [Callbacks in Live Editor](#).

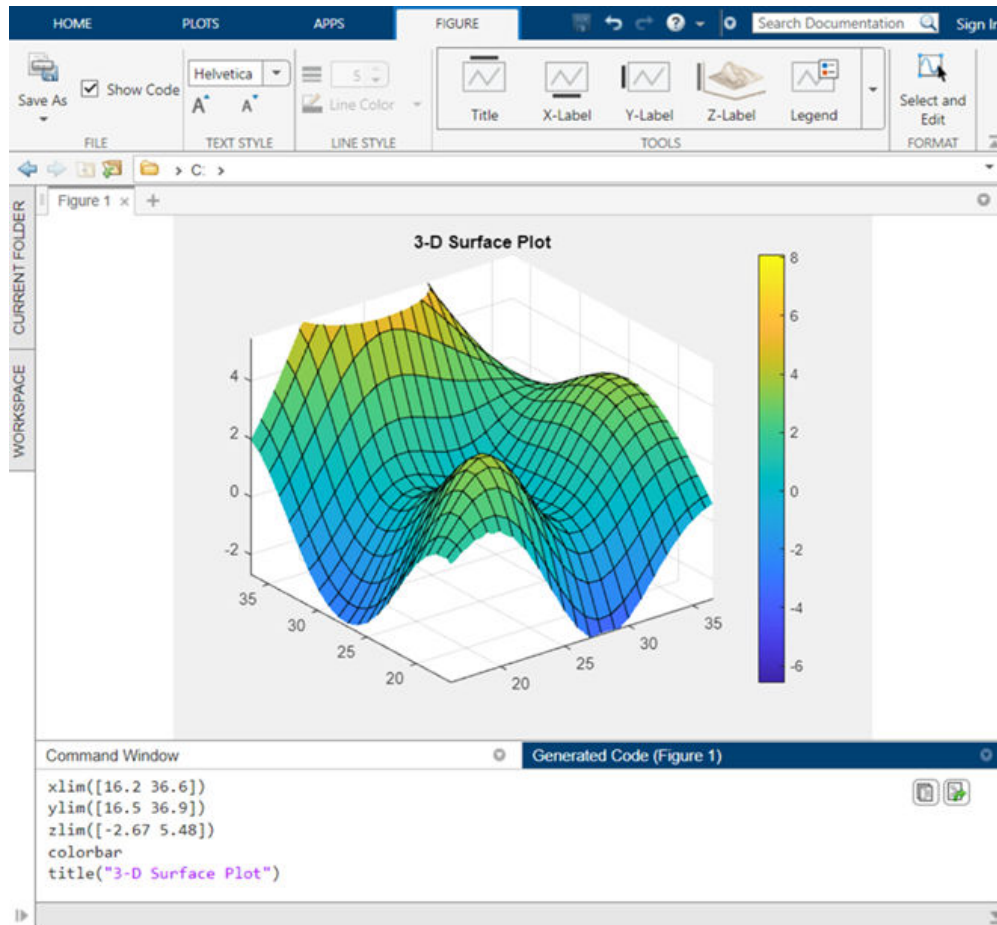
## Figure Code: Generate code for figure interactions in MATLAB Online

When you modify a figure in MATLAB Online using the **Figure** tab, MATLAB generates code that you can view, copy, and export. To view the generated code, select **Show Code** in the **File** section of the **Figure** tab. MATLAB generates code for these actions:

- Adding a title, axis label, legend, color bar, grid, or annotation
- Changing the text or line style
- Using the pan, zoom, rotate, or data tip interactions

MATLAB does not currently generate code for the **Select and Edit** option in the **Figure** tab.

For example, you can create a surface plot with `surf(peaks)`. Then, interactively add a title and colorbar, zoom into the axes, and view generated code.



## ⚠️ Functionality being removed or changed

### Polar axes display angle values with degree symbols

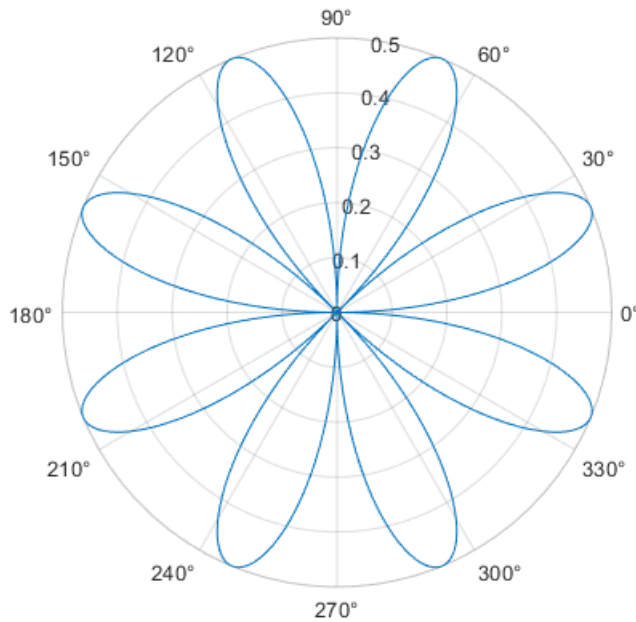
#### Behavior change

Polar axes now display tick values in degrees with degree symbols when the `ThetaAxisUnits` property is set to "degrees". For example, create a polar plot. By default, the *theta*-axis displays the tick values with degree symbols.

```

theta = 0:0.01:2*pi;
rho = sin(2*theta).*cos(2*theta);
polarplot(theta,rho)

```



This change clarifies which units are being used for the *theta* tick values. You can use the `ThetaAxisUnits` property to display the tick values in degrees or radians. To remove the degree symbols, change the tick label format for the *theta*-axis:

```
pax = gca;  
pax.ThetaAxis.TickLabelFormat = "%g";
```

### The `caxis` function is not recommended

*Still runs*

The `caxis` function is no longer recommended. However, the function continues to work, and there are no plans to remove it at this time.

To update your code, call the `clim` function instead. It accepts the same input arguments and returns the same output as the `caxis` function.

### The `im2java` function will be removed

*Still runs*

`im2java` will be removed in a future release. There is no replacement for this function.

### The Plot Catalog tool will be removed

*Still runs*

The Plot Catalog tool will be removed in a future release. Instead, to interactively create and explore visualizations for your data, use the **Plots** tab in the MATLAB Toolstrip or the Create Plot task in the Live Editor.

For more information about visualizations, see [Types of MATLAB Plots](#) or toolbox-specific documentation.



**The opengl function will be removed***Still runs*

The `opengl` function will be removed in a future release.

- To query the renderer, use the `rendererinfo` function instead of the `opengl` function.
- Changing the renderer with the `opengl` function will no longer be necessary when the function is removed.

**The renderer startup options will be removed***Still runs*

In a future release, the MATLAB startup options for selecting the graphics renderer will be removed. Specifically, these startup scenarios will no longer be available:

- `matlab -softwareopengl`
- `matlab -nosoftwareopengl`
- `matlab -softwareopenglmesa`
- `matlab -noopengl`

It will no longer be necessary to specify the renderer when these options are removed.

**The Renderer property of figures will have no effect***Behavior change in future release*

The `Renderer` and `RendererMode` properties of figures will have no effect in a future release. It will no longer be necessary to change the renderer when these properties are disabled.

**The FontSmoothing property will have no effect***Behavior change in future release*

The `FontSmoothing` property for all types of axes, rulers, geographic scales, and text objects will have no effect in a future release. Font smoothing will be enabled regardless of the value of the property.

**Some plot tools functions will redirect to the Figure Toolstrip and Property Inspector***Behavior change in future release*

Calling these plot tools functions will open a configuration of the Figure Toolstrip and Property Inspector. For more information, see the Version History section in the documentation for each function.

- `figurepalette`
- `plotbrowser`
- `propertyeditor`
- `propedit`
- `plottools`
- `showplottool`
- `plottedit`

Currently, calling plot tools functions opens the Figure Palette, Plot Browser, and Property Editor.

## App Building

### uistyle Function: Add icons and format text in table cells and tree nodes

You can now create styles for table and tree UI components that specify an icon and a text interpreter using the `uistyle` function.

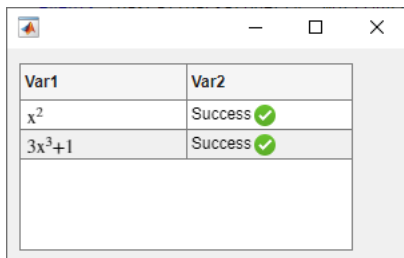
- Specify the `Icon` property of the style object to add icons to table cells and tree nodes.
- Specify the `IconAlignment` property of the style object to modify where the icon appears in relation to the text in table cells.
- Specify the `Interpreter` property of the style object to format text or add links using HTML markup, or to add equations using TeX or LaTeX markup to table cells and tree nodes.
- Specify the `HorizontalClipping` property of the style object to control whether long text is clipped on the left or the right in table cells and tree nodes.

Add a style to a UI component using the `addStyle` function.

For example, this code creates two styles, one that specifies an icon and one that specifies the text interpreter as TeX, and applies the styles to columns of a table.

```
T = table(["x^2";"3x^3+1"],["Success";"Success"]);
fig = uifigure(Position=[500 500 300 160]);
t = uitable(fig,Position=[10 10 250 140],Data=T);

s1 = uistyle(Interpreter="tex");
s2 = uistyle(Icon="success",IconAlignment="right");
addStyle(t,s1,column=1)
addStyle(t,s2,column=2)
```



| Var1     | Var2                                                                                        |
|----------|---------------------------------------------------------------------------------------------|
| $x^2$    | Success  |
| $3x^3+1$ | Success  |

### uitable Function: Rearrange columns of table UI components interactively

You can specify the ability to interactively rearrange table columns in an app by using the `ColumnRearrangeable` property. In a table UI component with the `ColumnRearrangeable` value set to 'on', rearrange table columns in the app by clicking and dragging the column header.

In App Designer and apps created using the `uifigure` function, you can program an app to respond when a user rearranges table columns by creating a `DisplayDataChangedFcn` callback function.

For more information, see Table Properties.

## focus Function: Give keyboard focus to UI components programmatically

Use the `focus` function to programmatically give focus to keyboard-focusable UI components. When a UI component is focused, it is displayed with a blue focus ring, and app users can interact with the component using the keyboard.

## isInScrollView Function: Determine if a component is visible in a scrollable container

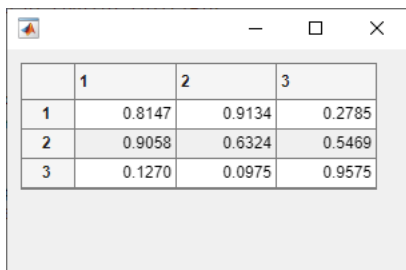
Use the `isInScrollView` function to programmatically identify which components are visible given the size and scroll location of a scrollable container. For example, you can determine which axes are visible inside a scrollable figure window and then update the data only for those axes.

## ⚠️ uigridlayout Function: Resize table, list box, and image UI components to fit content

Grid layout managers with row heights or column widths of `'fit'` now resize to fit the contents of table, list box, and image UI components.

For example, when you create a table UI component inside a grid layout manager with a row height or column width of `'fit'`, the height of the row or the width of the column resizes to fit the data in the table.

```
fig = uifigure(Position=[680 558 300 170]);
gl = uigridlayout(fig);
gl.RowHeight = {'fit'};
gl.ColumnWidth = {'fit'};
tbl = uitable(gl,Data=rand(3));
```



|   | 1      | 2      | 3      |
|---|--------|--------|--------|
| 1 | 0.8147 | 0.9134 | 0.2785 |
| 2 | 0.9058 | 0.6324 | 0.5469 |
| 3 | 0.1270 | 0.0975 | 0.9575 |

## ⚠️ Compatibility Considerations

In R2021b, grid layout managers with row heights or column widths of `'fit'` scaled to a fixed size when the row or column contained a table, list box, or image UI component.

- Table UI component — Row height and column width previously resized to 300 pixels.
- List box UI component — Row height previously resized to display at most four items. The exact pixel value to display four items might vary depending on your settings.
- Image UI component — Row height and column width previously resized to 100 pixels.

To display a table, list box, or image at its size in a release before R2022a, set the corresponding elements of the `RowHeight` and `ColumnWidth` properties of the `GridLayout` object to their respective fixed sizes.

## Live Editor Tasks: Develop your own Live Editor tasks for use in live scripts and functions

Live Editor tasks are simple point-and-click interfaces that can be embedded into a live script. Tasks represent a series of MATLAB commands that are automatically generated as users explore parameters.

You can develop your own custom Live Editor tasks by creating a subclass of the `LiveTask` base class. Develop tasks to perform your own specific set of operations within a live script.

For more information, see [Live Editor Task Development Overview](#).

## Custom UI Components: Interactively create custom UI components in App Designer

Use App Designer to interactively build your own UI components. Open a new blank custom UI component in App Designer, lay out the component by combining existing MATLAB UI components or graphics objects, and configure the component interface by creating public properties and public callbacks that can be set when the component is used in an app.

Creating a custom UI component has these benefits:

- **Modularization** — Separate the display of large apps into independent, maintainable pieces.
- **Reusability** — Provide a convenient interface for adding and customizing similar components in apps.
- **Flexibility** — Extend the appearance and behavior of existing UI components.

For more information, see [Create a Simple Custom UI Component in App Designer](#).

## App Designer: Modify tab focus order of components

You can view and modify the order in which components in your app receive keyboard focus when the app user presses **Tab**. First, sort and filter the **Component Browser** by tab order by selecting **Sort & Filter by Tab Order** from the drop-down list labeled **View**. The **Component Browser** lists only the components in the app that can have focus, in the order of focus. You can then change the tab order of the components by clicking and dragging the component names in the **Component Browser**.

Alternatively, App Designer can automatically apply a left-to-right and then top-to-bottom tab focus order for components in a container. Right-click the name of the container in the **Component Browser** and select **Apply Auto Tab Order**.

## App Designer: Specify error handling options and navigate from error messages when debugging an app

To specify error handling options when debugging code in App Designer, configure the  **Run** button by clicking **Run**⌵. You can choose to pause code execution when an error occurs, when a warning occurs, or when a NaN or Inf value is returned.

Additionally, error messages in App Designer now contain links to relevant files and functions. Use these links to navigate more easily to the documentation or to line numbers in your code when debugging your apps.

## App Designer: Manage image files in your app with an improved workflow

When you specify image data for your app, such as the image source of an image component or the icon of a button, select an image that is in the same folder as the MLAPP file or one of its subfolders. The image will then load whenever the app is opened or run without it needing to be on the MATLAB path. Alternatively, you can continue to use images in any location by adding the image files to the MATLAB path.

## App Designer: Convert components in a grid layout manager to use pixel-based positioning

You can delete a grid layout manager and convert the components in the grid to use pixel-based positioning. To use pixel-based positioning when you were previously using a grid layout manager, right-click the container with the grid layout manager in the canvas, and select **Remove Grid Layout**.

For more information, see [Use Grid Layout Managers in App Designer](#).

## App Designer: Use App Designer in most modern web browsers in MATLAB Online

You can now use App Designer in MATLAB Online with the current versions of Mozilla® Firefox®, Apple Safari, and Microsoft Edge®, in addition to Google Chrome®.

## App Designer: Customize design environment layout

You can now customize the locations of the side panels and tabs in the App Designer design environment.

To change the location of side panels such as the **Component Library** and the **Component Browser**, click the panel header and drag it to a new location in the App Designer environment. To change the location of your open tabs to display on the left, right, or bottom of the working area, right-click the tab bar and select **Tab Position**.

Your changes to the design environment layout now persist even after you close and reopen App Designer.

## Comparison Tool: Compare and merge app files in MATLAB Online

Compare and merge two versions of an app file in MATLAB Online using the Comparison Tool. To open the Comparison Tool, click  **Compare** in the **Designer** tab of the App Designer Toolstrip.

For more information, see [Compare and Merge Apps](#).

## ⚠️ Functionality being removed or changed

### RearrangeableColumns property of table UI components is not recommended

*Still runs*

Starting in R2022a, using the `RearrangeableColumns` property to specify the ability to rearrange columns in a table UI component is not recommended. Use the `ColumnRearrangeable` property instead. The new property can have the same values as the old one.

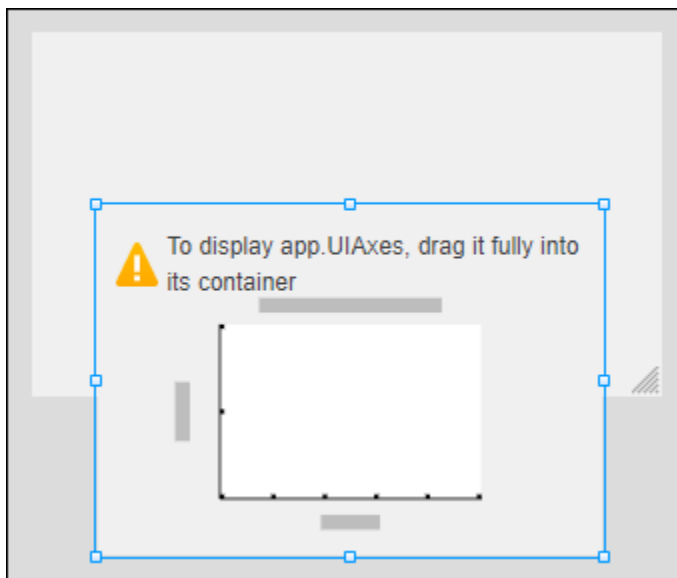
There are no plans to remove support for the `RearrangeableColumns` property at this time. However, the `RearrangeableColumns` property no longer appears in the list returned by calling the `get` function on a table UI component.

### UIAxes content is not displayed in App Designer Design View when the component is off the canvas

*Behavior change*

When creating an app with a `UIAxes` component in App Designer, the `UIAxes` component content is not displayed in **Design View** when the component is partially or fully off the canvas. Instead, the `UIAxes` component is shown as a placeholder image. To see the content of the component, drag it fully onto the canvas.

You can still modify properties of the `UIAxes` component when it is off the canvas, but you will not be able to see a visual reflection of those changes in **Design View** until the component is dragged onto the canvas.



## ComponentContainer class assigns a parent before executing the setup method

### *Behavior change*

When you create an instance of a custom UI component created using the `matlab.ui.componentcontainer.ComponentContainer` class, the class now assigns the component parent before executing the `setup` method. As a result, you might see unexpected behavior if your `setup` method creates underlying UI components that can be parented to either a figure created using the `figure` function or a figure created the `uifigure` function, such as panels, tab groups, or button groups.

To update your class code, when you create such a component, specify the property value explicitly for any property where the default value differs depending on the parent. For example, to create a panel in your custom component that is sized using normalized units, specify `Units` as "normalized" before setting the `Position` property.

## ButtonDownFcn callback cannot be interactively assigned for custom UI components in App Designer

### *Behavior change*

Starting in R2022a, when you use a custom UI component in an app in App Designer, you cannot interactively assign a `ButtonDownFcn` callback to the component. If you have an existing App Designer app that contains a custom UI component with a `ButtonDownFcn` callback that you assigned interactively, opening the app in R2022a disconnects the callback from the component. To reassign the callback to the component, follow these steps:

- 1 If your app does not contain a `StartupFcn` callback, right-click the app node from the top of the **Component Browser** hierarchy and select **Callbacks > Add StartupFcn Callback**.
- 2 In **Code View**, in the `startupFcn` function, add this code to assign the appropriate `ButtonDownFcn` callback programmatically:

```
app.CustomUIComponentName.ButtonDownFcn = ...
    @(src,event)CallbackFunctionName(app,event);
```

For example, if your app contains a custom UI component named `app.IPAddress` with a `ButtonDownFcn` callback named `IPAddressButtonDown`, add this code to the `startupFcn` function of your app:

```
app.IPAddress.ButtonDownFcn = @(src,event)IPAddressButtonDown(app,event);
```

## Performance

### **table Data Type Indexing: Improved performance when subscripting with dot notation or multiple levels of indexing**

table subscripting when using dot notation is significantly faster in R2022a than in R2021b. Also, subscripting with multiple levels of indexing is faster.

- For example, when you use dot notation to refer to a table variable with  $10^6$  elements, performance in R2022a is more than 4x faster than in R2021b.

```
function timingTest()
    t = table(zeros(1e6,1), ones(1e6,1), nan(1e6,1));
    indices = 1:1e5;

    tic;
    % Refer to variable using dot notation
    for i = indices
        x = t.Var1;
    end
    toc
end
```

The approximate execution times are:

**R2021b:** 1.55 s

**R2022a:** 0.36 s

- Similarly, when you use dot notation to assign an array to a table variable with  $10^6$  elements, performance in R2022a is about 3x faster than in R2021b.

```
function timingTest()
    t = table(zeros(1e6,1), ones(1e6,1), nan(1e6,1));
    indices = 1:1e5;
    x = randi(1e6,1e6,1);

    tic;
    % Assign to variable using dot notation
    for i = indices
        t.Var1 = x;
    end
    toc
end
```

The approximate execution times are:

**R2021b:** 2.15 s

**R2022a:** 0.72 s

- Also, when you use dot notation and parentheses to assign individual values to elements of a table variable, performance in R2022a is more than 4x faster than in R2021b.

```
function timingTest()
    t = table(zeros(1e6,1), ones(1e6,1), nan(1e6,1));
    indices = randi(1e6,1,1e5);
```



```

tic;
% Assign to elements using dot notation and parentheses
for i = indices
    t.Var1(i) = rand;
end
toc
end

```

The approximate execution times are:

**R2021b:** 5.08 s

**R2022a:** 1.20 s

The code was timed on a Windows 10, Intel Xeon CPU W-2133 @ 3.60 GHz test system by calling each version of the `timingTest` function.

## Classes: Improved performance for static methods, constant property access, and package functions in scripts

The overhead times for these three actions when performed in scripts are reduced:

- Executing a package function
- Executing a static method
- Accessing a constant property

The overhead times for these actions in a script are now comparable with the times of execution in a function, and the overheads are small enough that they can generally be ignored for performance considerations.

This code performs each of these actions 1,000,000 times. (The code called in each loop is shown at the end of this note.)

```

tic;
for j = 1:1000000
    out = pkg1.pkg2.packageFunction(2);
end
toc;

tic;
for j = 1:1000000
    out = MyClass.staticMethod(1);
end
toc;

tic;
for j = 1:1000000
    out = pkg1.PackageClass.constantProperty;
end
toc;

```

The approximate times to complete each loop are:

**R2021b:**

- Package function: 8.4 s
- Static method: 7.8 s
- Constant property access: 32 s

**R2022a:**

- Package function: 0.04 s (210x faster)
- Static method: 0.031 s (252x faster)
- Constant property access: 0.039 s (821x faster)

The code was timed on a Windows 10, Intel Xeon CPU W-2133 @ 3.60 GHz test system.

For use with the test code, `PackageClass` must be in folder `+pkg1`, and `packageFunction` must be in folder `+pkg1/+pkg2`.

```
function out = packageFunction(in)
    out = in;
end

classdef MyClass
    methods (Static)
        function out = staticMethod(in)
            out = in;
        end
    end
end

classdef PackageClass
    properties (Constant)
        constantProperty = 3;
    end
end
```

## try Block: Improved performance when statements run error-free

The `try` block shows improved performance when the statements within the block run error-free. For example, this code is approximately 6x faster than in the previous release:

```
function testTryPerformance
x = 1;
for i = 1:1e8
    try
        x = x * i;
    catch
        warning("Assignment was not successful.")
        x = 1;
    end
end
end
```

The approximate execution times are:

**R2021b:** 2.3 s

**R2022a:** 0.4 s

The code was timed on a Windows 10, Intel Xeon CPU E5-1650 v4 @ 3.60 GHz test system using the `timeit` function:

```
timeit(@testTryPerformance)
```

## Python Data Type Conversion: Improved performance when converting between Python and MATLAB data types in out-of-process mode

When you run Python code out of process, conversions between Python data types and MATLAB data types show improved performance.

- The `timingPythonList` function converts a Python list object to a MATLAB cell array. This code is about 118x faster than in the previous release:

```
function timingPythonList
    l = py.list(['MATLAB', 'R', 2022, 'a', 'and', 1.2, 10^2, ...
                2021, 12, 25, '2021', '12', '25', 'and', ...
                2021, 12, 25, '2021', '12', '25']);
    tic
    ml = cell(l);
    toc
end
```

The approximate execution times are:

**R2021b:** 1.18 s

**R2022a:** 0.01 s

- The `timingPythonDict` function converts a Python dictionary object to a MATLAB structure. This code is about 57.9x faster than in the previous release:

```
function timingPythonDict
    d = py.dict(pyargs('a', 1, 'b', 2, 'c', 3, 'd', 4, 'e', 5, ...
                      'f', 6, 'g', 7, 'h', 8, 'i', 9, 'j', 10, 'k', 11, 'l', 12, ...
                      'm', 13, 'n', 14, 'o', 15, 'p', 16, 'q', 17, 'r', 18, ...
                      's', 19, 't', 20));
    tic
    ms = struct(d);
    toc
end
```

The approximate execution times are:

**R2021b:** 0.521 s

**R2022a:** 0.009 s

- The `timingDataTransfer` function converts an array with  $10^8$  elements from a MATLAB double array to a Python memoryview object and then back to a MATLAB double array. This code is about 10x faster when converting from MATLAB to Python and approximately 11x faster when converting from Python to MATLAB than in the previous release:

```
function timingDataTransfer
    data = rand(100, 10^6);
    tic
```

```
    pydata = py.memoryview(data);  
    toc  
    tic  
    mdata = double(pydata);  
    toc  
end
```

The approximate execution times for converting the MATLAB double array to the Python memoryview object are:

**R2021b:** 4.0 s

**R2022a:** 0.4 s

The approximate execution times for converting the Python memoryview object to the MATLAB double array are:

**R2021b:** 7.7 s

**R2022a:** 0.7 s

All of the code was timed on a Windows 10, Intel Core i7-10510U CPU @ 1.80 GHz 2.30 GHz test system by using Python 3.9 in out-of-process mode and calling the `timingPythonList`, `timingPythonDict`, and `timingDataTransfer` functions.

## MATLAB Engine API for Python: Improved performance with large multidimensional arrays in Python

The Python multidimensional array component used by the MATLAB Engine API for Python shows improved performance when:

- 1 Converting data from Python sequences to the data types defined by the `matlab` module
- 2 Transferring data back and forth between Python and MATLAB

In both cases, the improvement is noticeable when operating on arrays with at least 10 elements. When transferring data back and forth, the improvement increases as the size of the array increases.

For example, this Python code measures the execution times of two operations:

- 1 Converting a Python array of size  $10^8$  to a MATLAB double array
- 2 Summing the elements of the MATLAB array using the MATLAB engine

The first operation is about 12x faster than in the previous release, and the second operation is about 110x faster than in the previous release:

```
import random  
import time  
import matlab.engine  
eng = matlab.engine.start_matlab()  
  
rand_array = [random.random() for i in range(10**8)]  
s0 = time.perf_counter()  
array_md = matlab.double(rand_array,size=(1, 10**8))  
s1 = time.perf_counter() - s0  
print('conversion to matlab.double(): {} seconds'.format(s1))
```

```
s0 = time.perf_counter()
sum_of_elems = eng.sum(array_md,1)
s1 = time.perf_counter() - s0
print('sum(): {} seconds'.format(s1))
```

The approximate execution times for the first operation are:

**R2021b:** 42 s

**R2022a:** 3.6 s

The approximate execution times for the second operation are:

**R2021b:** 210 s

**R2022a:** 1.9 s

The code was timed on a Windows 10, Intel Xeon CPU E5-1650 v4 @ 3.60 GHz test system by using Python `perf_counter()` statements.

## Matrix multiplication: Improved performance when multiplying sparse and full matrices

Matrix multiplication shows improved performance when:

- One of the operands is a sparse matrix, and the other is a full matrix.
- The sparse operand has at least 50,000 nonzero elements.
- The full operand has at least 32 columns (or at least 32 rows when transposed).

The performance improvement arises from added support for multithreading in the operation, and therefore the speedup improves as the matrix size and number of nonzero elements increase.

For example, multiplying a 102,400-by-102,400 sparse matrix with a 102,400-by-128 full matrix on a machine with 6 physical cores is about 2.7x faster than in the previous release.

```
function timingSparseDenseMult
A = delsq(numgrid('S',322));
B = rand(size(A,2),128);
tic
for k = 1:10
    C = A*B;
end
toc
end
```

The approximate execution times are:

**R2021b:** 0.8 s

**R2022a:** 0.3 s

The code was timed on a Windows 10, Intel Xeon CPU W-2133 @ 3.60 GHz test system by calling the `timingSparseDenseMult` function.

## inv Function: Improved performance when inverting large triangular matrices

The `inv` function shows improved performance when operating on large triangular matrices.

For example, inverting a 5,000-by-5,000 upper triangular matrix is about 3.7x faster than in the previous release.

```
function timingInv
    rng default
    A = randn(5e3);
    [~,R] = lu(A);

    tic
    Y = inv(R);
    toc
end
```

The approximate execution times are:

**R2021b:** 1.1 s

**R2022a:** 0.3 s

The code was timed on a Windows 10, Intel Xeon CPU W-2133 @ 3.60 GHz test system by calling the `timingInv` function.

## sprand and sprandn Functions: Improved performance when generating random sparse matrices

The `sprand` and `sprandn` functions show improved performance when generating random sparse matrices if the number of nonzero elements in the output is larger than the number of rows.

For example, generating a 10,000-by-10,000 matrix with 10% density of nonzero elements is about 2.5x faster than in the previous release.

```
function timingSprand
    n = 1e4;
    d = 0.1;
    rng default

    tic
    sprand(n,n,d);
    toc
end
```

The approximate execution times are:

**R2021b:** 2.7 s

**R2022a:** 1.1 s

The code was timed on a Windows 10, Intel Xeon CPU W-2133 @ 3.60 GHz test system by calling the `timingSprand` function.

## fzero Function: Improved performance

The `fzero` function shows improved performance for objective functions specified as function handles. This improvement is most noticeable for functions that take little time to evaluate. The following example, which solves for  $1e5$  roots of a simple function, takes less than one third of the time of the previous release:

```
N = 1e5;
rng default
levels = 1.5*rand(N,1);
out = zeros(N,1);
tic
for i = 1:N
    out(i) = fzero(@(x)myfun(x,levels(i)),[0 2]);
end
toc

function u = myfun(x,lv)
u = x*sin(x) - lv;
end
```

The approximate execution times are:

**R2021b:** 5.83 s

**R2022a:** 1.66 s

The code was timed on a Windows 10, Intel Xeon CPU E5-1650 v4 @ 3.60 GHz test system by running the above script.

## diff Function: Improved performance with large number of elements

The `diff` function shows improved performance when operating on vectors with at least  $10^5$  elements or when operating along the first or second dimension of matrices and multidimensional arrays with at least  $5 \times 10^5$  elements.

For example, this code creates a `double` array with  $2.5 \times 10^7$  elements and calculates differences between adjacent elements. It is approximately 2.4x faster than in the previous release.

```
function timingDiff
rng default
N = 5000;
A = rand(N);

tic
for k = 1:40
    D = diff(A);
end
toc
end
```

The approximate execution times are:

**R2021b:** 2.43 s

**R2022a:** 1.00 s

The code was timed on a Windows 10, Intel Xeon CPU E5-1650 v4 @ 3.60 GHz test system by calling the `timingDiff` function.

## groupsummary, groupfilter, and grouptransform Functions: Improved performance with small group size

Grouping functions `groupsummary`, `groupfilter`, and `grouptransform` show improved performance, especially when the data count in each group is small.

For example, this code performs group summary computations on a matrix with 500 groups with a count of 10 each. It is about 2.18x faster than in the previous release.

```
function timingGroupsummary
data = (1:5000)';
groups = repelem(1:length(data)/10,10)';
p = randperm(length(data));
data = data(p);
groups = groups(p);

tic
for k = 1:300
    G = groupsummary(data,groups,"mean");
end
toc
end
```

The approximate execution times are:

**R2021b:** 2.14 s

**R2022a:** 0.98 s

The code was timed on a Windows 10, Intel Xeon CPU E5-1650 v4 @ 3.60 GHz test system by calling the `timingGroupsummary` function.

## nufftn Function: Improved performance with nonuniform sample points or query points

The `nufftn` function shows improved performance when operating on either nonuniformly spaced sample points or nonuniformly spaced query points.

For example, this code constructs a 32,768-by-3 matrix of nonuniform sample points `t` and calculates the nonuniform discrete Fourier transform along each dimension of a 32-by-32-by-32 array. The code is about 14.5x faster than in the previous release.

```
function timingSamplePoints
rng default
t = rand(32^3,3);
X = rand(32,32,32);
tic
    Y = nufftn(X,t);
toc
end
```

The approximate execution times are:



**R2021b:** 2.76 s

**R2022a:** 0.19 s

The code was timed on a Windows 10, Intel Xeon CPU E5-1650 v4 @ 3.60 GHz test system by calling the `timingSamplePoints` function.

As another example, this code constructs a 65,536-by-3 matrix of nonuniform query points `f` and calculates the nonuniform discrete Fourier transform along each dimension of a 64-by-32-by-32 array. The code is about 42.6x faster than in the previous release.

```
function timingQueryPoints
rng default
f = rand(64*32*32,3);
X = rand(64,32,32);
tic
    Y = nufftn(X,[],f);
toc
end
```

The approximate execution times are:

**R2021b:** 4.26 s

**R2022a:** 0.10 s

The code was timed on a Windows 10, Intel Xeon CPU E5-1650 v4 @ 3.60 GHz test system by calling the `timingQueryPoints` function.

## Variables Editor and Live Editor: Improved speed of data display when scrolling

For text and datetime data types in the Variables editor or in the generated output of the Live Editor, the performance of vertical and horizontal scrolling is improved. Displayed data is optimized and the rendering mechanism is faster, so data appears more quickly after scrolling in R2022a than in previous releases.

For example, on a Windows 10, Intel Xeon CPU E5-1650 v4 @ 3.60 GHz test system, when you scroll through a timetable with 3000 rows, the displayed data renders more quickly in R2022a than in R2021b.

## App Building: Improved performance when creating UI components

Creating UI components is faster in R2022a than in R2021b. As a result, apps start up faster when you run them. This improvement is more noticeable for apps with many UI components.

For example, this code measures the time it takes to create 500 edit field components. The code is about 1.25x faster than in the previous release.

```
function timingApp
fig = uifigure;
gl = uigridlayout(fig,Scrollable="on");
gl.RowHeight = repmat({'fit'},1,50);
gl.ColumnWidth = repmat({'fit'},1,10);
```

```
drawnow

tic
for k = 1:500
    uieditfield(gl);
end
drawnow
toc
end
```

The approximate execution times are:

**R2021b:** 7.5 s

**R2022a:** 6.0 s

The code was timed on a Windows 10, Intel Xeon CPU E5-1650 v4 @ 3.60 GHz test system by calling the `timingApp` function.

## uitable Function and UI Containers: Improved performance when updating properties successively

Updating property values for certain UI components and containers is faster in R2022a than in R2021b. This performance improvement applies to table UI components created using the `uitable` function, UI containers created using the `uipanel`, `uibuttongroup`, `uitab`, and `uitabgroup` functions, and custom UI components created using the `ComponentContainer` base class, when these objects are parented to a figure created using the `uifigure` function.

For example, this code creates a table UI component and then updates the table cell values for 1000 cells. The code is about 5.2x faster than in the previous release.

```
function timingTableUpdates
fig = uifigure;
tbl = uitable(fig, "Data", rand(1000,15));
drawnow

tic
for k = 1:1000
    tbl.Data(k,1) = 0;
end
drawnow
toc
end
```

The approximate execution times are:

**R2021b:** 2.6 s

**R2022a:** 0.5 s

The code was timed on a Windows 10, Intel Xeon CPU E5-1650 v4 @ 3.60 GHz test system by calling the `timingTableUpdates` function.

## UI Components: Improved performance when setting a property with an unchanged value

Setting a property value of a UI component when the value is unchanged is faster in R2022a than in R2021b. This speed increase leads to improved performance in apps that update many UI component properties at once, even if not all property values have changed.

For example, this code sets the `Value` property of an edit field to the same value 1000 times. The code is about 23x faster than in the previous release.

```
function timingValueSet
fig = uifigure;
ef = uieditfield(fig);
drawnow

tic
for k = 1:1000
    ef.Value = "Text";
    drawnow
end
toc
end
```

The approximate execution times are:

**R2021b:** 2.3 s

**R2022a:** 0.1 s

The code was timed on a Windows 10, Intel Xeon CPU E5-1650 v4 @ 3.60 GHz test system by calling the `timingValueSet` function.

## App Designer: Improved performance when loading apps with UIAxes components off the canvas

In App Designer, apps with `UIAxes` components that lie partially or fully off the canvas in **Design View** have these performance improvements:

- The app loads faster when you open it.
- The `UIAxes` components update faster in response to property edits.

The speed increase improves as the number of `UIAxes` components that lie off the canvas increases.

For example, create an app in App Designer using these steps:

- 1 Drag five `UIAxes` components onto the canvas.
- 2 Drag each of the axes components so that they lie partially off the canvas.
- 3 Save and close the app.

Reopen the app. The time it takes for App Designer to fully load the app is about 5.2x faster than in the previous release.

The approximate load times are:

**R2021b:** 10.5 s

**R2022a:** 2 s

Once the app is fully loaded, select a `UIAxes` component and change its title in the Property Inspector. The property updates almost instantaneously. In the previous release, the property update takes approximately 2 seconds.

These interactions were timed on a Windows 10, Intel Xeon CPU E5-1650 v4 @ 3.60 GHz test system.

## Plots in Apps: Improved responsiveness for event-driven updates in apps

Apps that involve continuous event-driven updates, such as animations implemented with a timer, are more responsive to those events. To observe the improvement, the animation must be created in an app or in a figure created with the `uifigure` function. For example, an app containing an animation controlled by a timer object is more responsive when you call the timer's `start` and `stop` methods.

This code creates an app using a timer object to plot a random number every 0.01 second. If you run this app on a Windows 10, Intel Xeon CPU W-2133 @ 3.60 GHz test system, the animation responds more quickly when you click the **Start/Stop** button than in the previous release.

```
function mytimerapp
uif = uifigure("CloseRequestFcn",@CloseRequest);
uibutton(uif,"State","Position",[235 59 100 22], ...
    "ValueChangedFcn",@StateButtonChanged,"Text","Start/Stop");
ax = uiaxes(uif,"Position",[25 100 508 300],"XDir","reverse");

% Create initial plot line
p = plot(ax,0:60,zeros(1,61));

% Create timer object
RandTimer = timer(...
    "ExecutionMode","fixedRate", ...
    "Period",0.01, ...
    "BusyMode","queue", ...
    "TimerFcn",@RandTimerFcn);

% Timer function
function RandTimerFcn(~,~,~)
    % Generate a random number and update plot line
    ydata = p.YData;
    ydata = circshift(ydata,1);
    ydata(1) = rand;
    p.YData = ydata;
end

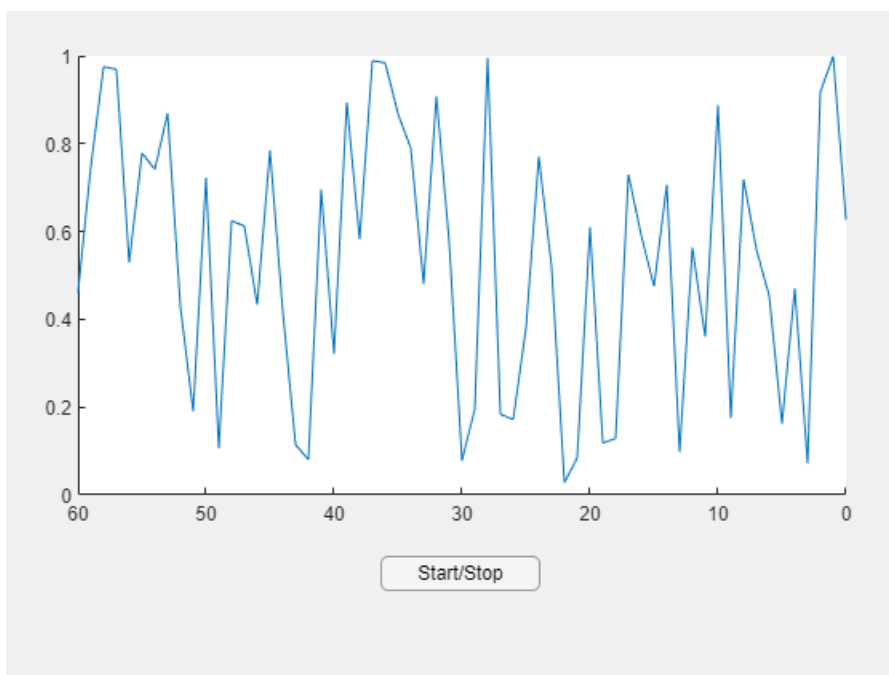
% State button callback function
function StateButtonChanged(obj,~)
    switch obj.Value
        case 0
            stop(RandTimer);
        case 1
            % If timer is not running, start it
            if strcmp(RandTimer.Running,"off")
                start(RandTimer);
```

```

        end
    end
end

% Figure close request function
function CloseRequest(~,~)
    % Stop timer, then delete timer and figure
    stop(RandTimer);
    delete(RandTimer);
    delete(uif);
end
end

```



## Plots in Apps: Improved responsiveness of axes interactions within apps

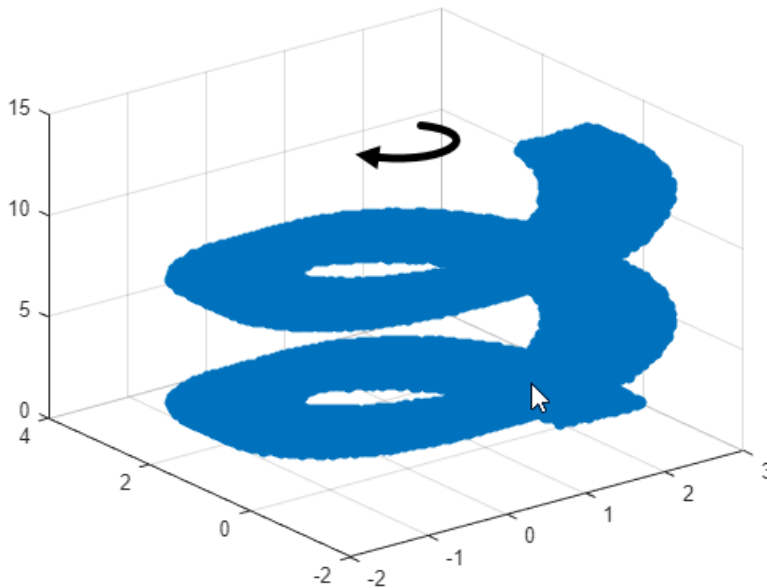
Responsiveness is improved for panning, rotating, and zooming into a region of interest within a plot. To observe the improvement, the plot must be created in an app or in a figure created with the `uifigure` function. The improvement is most noticeable for plots with large numbers of points, or those that involve complex effects such as lighting, transparency, or texture maps. Systems equipped with modern GPUs are more likely to show the improvement.

For example, create a 3-D scatter plot with 200,000 points. If you run this code on a Windows 10, Intel Xeon CPU W-2133 @ 3.60 GHz test system with an NVIDIA Quadro P620 GPU, and then drag to rotate the plot, the rotation is smoother and responds more quickly to the drag gesture than in the previous release.

```

z = linspace(0,4*pi,200000);
x = 2*cos(z) + rand(1,200000);
y = 2*sin(z) + rand(1,200000);
f = uifigure; a = axes(f);
scatter3(a,x,y,z,"filled")

```

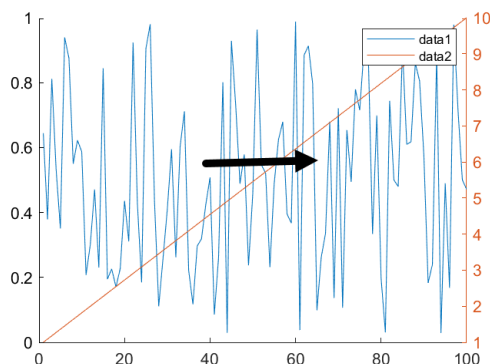


## Plots in Apps: Improved responsiveness of axes interactions in plots with two y-axes

Responsiveness is improved for panning, rotating, and zooming into a region of interest within a UIAxes or Axes object in MATLAB Online that contains a chart created with `yyaxis` and a legend. For such charts, updates are faster, and interactions are smoother in R2022a than in the previous release.

For example, create a UIAxes object with two y-axes and a legend. If you run this code on a Windows 10, Intel Xeon CPU E5-1650 v4 @ 3.60 GHz test system, when you drag the cursor to pan the view of a UIAxes object, the axes pan faster and track the cursor more closely in R2022a than in R2021b.

```
f = uifigure;
a = uiaxes(f);
plot(a,1:100,rand(1,100));
yyaxis(a,"right");
plot(a,1:100,linspace(1,10,100));
legend(a);
```



## Plots in Apps: Faster animations in apps when multiple figures are open

Animations show improved performance in apps when multiple figures are open. To observe the improvement, all figures must be created with the `uifigure` function.

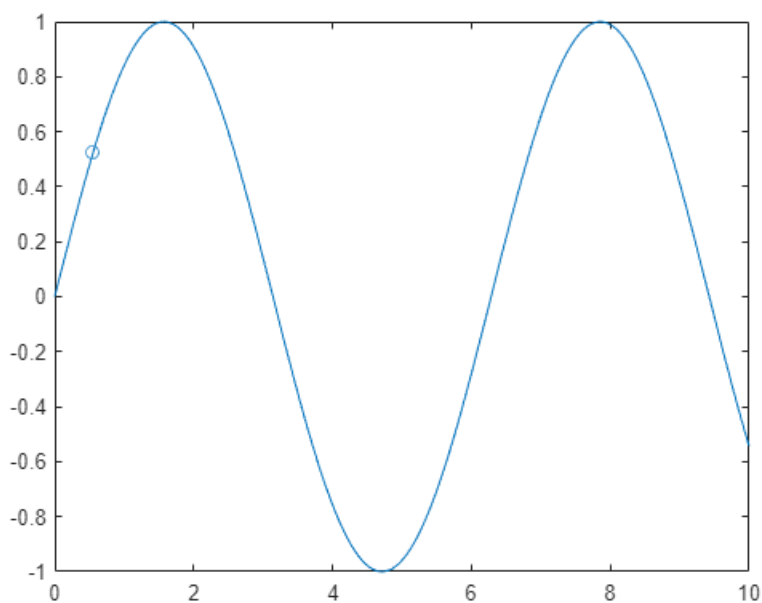
For example, this code opens 10 empty figures and an additional figure containing a plot. The plot displays one marker that traces the path of a line at every iteration of a loop. The loop is about 6x faster than in the previous release.

```
function movingmarker

% Open 10 empty figures
for n = 1:10
    uifigure("Position",[10+n*3 10+n*3 200 200]);
end

% Create a figure and a plot with one marker
uif = uifigure;
x = linspace(0,10,200);
y = sin(x);
ax = axes(uif);
p = plot(ax,x,y,"-o","MarkerIndices",1);

% Move the marker along the sine wave
tic
for idx = 2:200
    p.MarkerIndices = idx;
    drawnow
end
toc
end
```



The approximate execution times for the loop are:

**R2021b:** 32.6 s

**R2022a:** 5.4 s

The code was timed on a Windows 10, Intel Xeon CPU W-2133 @ 3.60 GHz test system by calling the `movingmarker` function.

## **Property Inspector: Improved performance when opening for the first time**

The Property Inspector for a figure window shows improved performance when opening for the first time in a MATLAB session. The delay between clicking the Property Inspector icon or calling `inspect` and the inspector being ready is reduced. The improvement is most noticeable as the plot in the figure window becomes more complex.

For example, open the Property Inspector for a plot of a 5-by-5 matrix by calling `inspect`. If you run this code on a Windows 10, Intel Xeon CPU E5-1650 v4 @ 3.60 GHz test system, you can use the Property Inspector sooner in R2022a than in R2021b.

```
r = rand(5,5);  
p = plot(r);  
inspect(p)
```



## Software Development Tools

### Projects: Reduce test runtime in continuous integration workflows using the dependency cache

You can now specify where your project stores the dependency analysis results. In agile development workflows that use Git and a continuous integration (CI) server, share the dependency cache file (.graphml) to run an incremental dependency analysis and reduce the test suite runtime. See Continuous Integration Using MATLAB Projects and Jenkins.

To set the project dependency cache file, on the **Project** tab, in the **Environment** section, click **Details**. In **Dependency cache file**, browse to and specify a GraphML file. If the cache file does not exist, the project creates it for you.

Alternatively, you can create and set the project dependency cache programmatically:

```
matlab.project.example.timesTable
proj = currentProject;
proj.DependencyCacheFile = "myProjectCacheFile"
```

### Dependency Analyzer: Save dependency graph as image

You can now save the dependency analysis results as an image. See Export Dependency Analysis Results.

### Code Compatibility Analyzer App: Identify and address compatibility issues against current version of MATLAB

The MATLAB Code Compatibility Report is now available as an app. You can access the Code Compatibility Analyzer from the apps gallery in MATLAB or from the command line using codeCompatibilityAnalyzer. For more information, see MATLAB Code Compatibility Analyzer.

### Unit Testing Framework: Create test classes interactively using the Current Folder browser

You can now create a test class using the Current Folder browser in MATLAB and MATLAB Online. To create a new test class, right-click in the Current Folder browser and then select **New > Test Class**.

### Unit Testing Framework: Create temporary folders that are automatically removed

The matlab.unittest.TestCase class has a new method createTemporaryFolder that creates a temporary folder for your tests. The lifecycle of the folder is tied to the test case. Once the test case goes out of scope, the testing framework removes the folder.

## Unit Testing Framework: Generate DOCX, HTML, and PDF reports after test execution

The `matlab.unittest.TestResult` class has three new methods that enable you to generate various test reports from test results. You can run your tests and collect the test results, and then generate test reports from part or all of your results:

- To generate a DOCX report from the test results, use the `generateDOCXReport` method.
- To generate an HTML report from the test results, use the `generateHTMLReport` method.
- To generate a PDF report from the test results, use the `generatePDFReport` method.

With this feature, you are no longer required to run tests using a `TestReportPlugin` instance. For example, run your tests and then generate an HTML report from the test results. Save the report as `report.html` in a folder named `myResults`.

```
suite = testsuite("MyTestClass");  
runner = testrunner;  
results = run(runner,suite);  
generateHTMLReport(results,"myResults",MainFile="report.html")
```

## Unit Testing Framework: Debug uncaught errors in tests

Starting in R2022a, when a test runner with a `StopOnFailuresPlugin` instance encounters an uncaught error, MATLAB enters debug mode at the source of the error and lets you use debugging commands to investigate the cause of the error. In previous releases, while the plugin stops the test run to report the error, debugging capabilities are limited because the error disrupts the stack.

## Unit Testing Framework: Collect statement and function coverage metrics for your source code

Starting in R2022a, when you generate an HTML code coverage report using the `CoverageReport` format, the report displays statement and function coverage metrics:

- Use statement coverage to see whether every MATLAB statement in your source code is executed at least once.
- Use function coverage to see whether every function in your source code is called at least once.

In previous releases, you can generate only line coverage metrics for your source code. Compared to line coverage, statement and function coverage provide a more detailed analysis of the source code covered by the tests.

## ⚠️ Functionality being removed or changed

### pack function will be removed in a future release

*Warns*

The `pack` function will be removed in a future release. There is no replacement for this function because you do not need to use it on a 64-bit system. For more information about strategies for reducing memory usage, see [Strategies for Efficient Use of Memory and Resolve “Out of Memory” Errors](#).

## **matlab.unittest.TestSuite.fromFolder includes tests from package folders when creating a test suite**

### *Behavior change*

Starting in R2022a, the `matlab.unittest.TestSuite.fromFolder` method treats folders and packages the same way, and includes tests defined within package folders when creating a test suite. For example, `suite = matlab.unittest.TestSuite.fromFolder(pwd, IncludingSubfolders=true)` creates a suite from all the test files in the current folder and any of its subfolders, including package folders. In previous releases, the method ignores any tests defined in a package folder and its subfolders.

This behavior change also applies to the `testsuite`, `runtests`, and `runperf` functions when they operate on a folder containing tests. With the consistent treatment of folders and packages, creating a suite from all test files within a folder and its subfolders becomes more convenient and independent of the folder structure.

To exclude tests defined within packages, filter the suite being constructed or returned by `fromFolder`. For example, create a filtered test suite comprising tests whose names do not include any dots (that is, do not refer to any packages).

```
import matlab.unittest.TestSuite
import matlab.unittest.selectors.HasName
import matlab.unittest.constraints.ContainsSubstring
suite = TestSuite.fromFolder(pwd, HasName(~ContainsSubstring(".")), ...
    IncludingSubfolders=true);
```

## **builddocsearchdb creates searchable database with new name**

### *Behavior change*

When building a searchable documentation database for custom toolboxes, the `builddocsearchdb` function now creates the subfolder `helpsearch-v4` to contain the search database files. Previously, `builddocsearchdb` created a subfolder named `helpsearch-v3`.

To ensure the documentation for the custom toolbox is searchable in R2022a, run `builddocsearchdb` against your help files using MATLAB R2022a. Maintain the `helpsearch-v4` subfolder containing the search database files created in R2022a and the `helpsearch-v3` subfolder containing the search database files created in previous releases side by side. Then, when you run any MATLAB release, the Help browser automatically uses the appropriate database for searching your documentation.

## External Language Interfaces

### C++ Interface: Array size help text for functions and methods

If a function or method takes a clib or MATLAB array, the generated help text displays size information for the argument. For more information, see [Array Size Help for Functions and Methods](#).

### C Interface: Build third-party C library interface using `clibgen.generateLibraryDefinition`

Create interfaces for libraries with C files that are built with C compilers using `clibgen.generateLibraryDefinition`. Use this function to get the benefits of publishing an interface as described in [Build MATLAB Interface to C++ Library](#) instead of calling the `loadlibrary` function described in [Call C from MATLAB](#).

For information about using `clibgen.generateLibraryDefinition` with C files, see **Files in Your Library** under the Tips section. To build an interface to C libraries, use the `CLinkage` name-value argument.

### C++ Interface: Support for C++ language features

The C++ interface supports these additional C++ language features.

- `std::complex` support for complex scalars and arrays for fundamental types of `double`, `float`, `int8`, `uint8`, `int16`, `uint16`, `int32`, `uint32`, `int64`, and `uint64`. For information about mapping these types to MATLAB data, see [Numeric Types](#).
- If the size of a 1-D array parameter is not specified, you can represent the size in MATLAB using multiple dimensions. For example, the input to this function is a 2-D array `mat` of size `len-by-typeSz`.

```
void readMatrix2DArr(int const [] mat, size_t len, size_t typeSz)
```

You can specify the SHAPE of the argument as `["len", "typeSz"]`. For more information, see [Define Missing SHAPE Parameter](#).

- `std::vector` as data member.

### C++ Interface: Publisher options

The C++ interface supports these build configuration features.

- Specify compiler and linker flags used to build an interface. Use the `AdditionalCompilerFlags` or `AdditionalLinkerFlags` name-value arguments in `clibgen.generateLibraryDefinition` or `clibgen.buildInterface` to pass flags to the compiler and linker. These functions do not validate the flags. The publisher needs to know how the flags affect the build process. For more information, see [Build C++ Library Interface and Review Contents](#).
- Build an interface using a specific compiler standard. For example, to build a library defined by `A.hpp` with C++17, type:

```
clibgen.generateLibraryDefinition("A.hpp", AdditionalCompilerFlags="-std=c++17")
```

For more information, see `Specify C++ Compiler Standard`.

- Include static libraries with `.a` file extension on Linux and macOS platforms, and on Windows if the library is compiled with a supported MinGW compiler. To include a static library, use the `Libraries` name-value argument in `clibgen.generateLibraryDefinition` or `clibgen.buildInterface`.

## Call MATLAB from C++: Generate C++ code Interface for MATLAB Packages, Classes, and Functions

The `matlab.engine.typedinterface.generateCPP` creates a C++ header file from MATLAB packages, classes, and functions. For more information, see `What is Strongly Typed Interface for C++?`.

## MATLAB Data Array API: `matlab::data::Array` support for row-major order

The MATLAB data array API supports iterating through the data of a `matlab::data::Array` in either row-major or column-major order. For information about specifying the layout when creating an array, see the `inputLayout` parameter for `createArray`. For information about iterating through an array, see `matlab::data::ColumnMajor`, `matlab::data::ColumnMajorIterator<T>`, `matlab::data::RowMajor`, and `matlab::data::RowMajorIterator<T>`.

## MEX Functions: UTF-8 system encoding on Windows platforms

MATLAB now uses UTF-8 as its system encoding on Windows, completing the adoption of Unicode across all supported platforms. System calls made from within a MEX file take and return UTF-8 encoded strings. If your MEX file contains code or links to third-party libraries that assume a different system encoding, then you might see garbled text and thus need to update the code to be Unicode compliant.

## Python: Use `Name=Value` syntax to pass keyword arguments to Python functions

You can use MATLAB `Name=Value` syntax as an alternative to the `pyargs` function to pass keyword arguments to Python functions. However, do not mix `Name=Value` arguments with the use of the `pyargs` function.

MATLAB does not support `Name, Value` syntax for passing keyword arguments.

## Python: Convert Python list and tuple types to MATLAB types

You can convert Python `list` and `tuple` types using MATLAB string and numeric converters. For details, see the `py.list` and `py.tuple` entry in the `Explicitly Convert Python Types to MATLAB Types` table. For examples, see `Use Python list Variables in MATLAB` and `Use Python tuple Variables in MATLAB`.

Conversion functions might not work on lists or tuples that contain elements which cannot be converted to the requested type:

```
double(py.list({3.0, 'MATLAB'}))
```

```
Error using py.list/double
Conversion of Python element at position 2 to type 'double' failed. All
Python elements must be convertible as scalar to the requested type.
```

Related documentation

For the related documentation, see [Error Converting Elements of list or tuple](#).

## ▲Perl 5.34.0: MATLAB support on Windows

As of R2022a, MATLAB on Windows ships with an updated version of Perl, version 5.34.0.

## ▲Compatibility Considerations

If you use the `perl` command on Windows platforms, see <https://www.perl.org/> for information about using this version of the Perl programming language.

## Compilers: Support for Microsoft Visual Studio 2022

As of R2021b Update 3, MATLAB supports Microsoft Visual Studio 2022 for building C and C++ interfaces, MEX files, and standalone MATLAB engine and MAT-file applications.

## ▲Functionality being removed or changed

### Python: Version 3.7 is no longer supported

*Errors*

Support for Python version 3.7 is discontinued. For continued support for your applications, upgrade to a supported version of Python. For supported version information, see [Versions of Python Compatible with MATLAB Products by Release](#).

### MEX file macro `FORTRAN_COMPLEX_FUNCTIONS_RETURN_VOID` has been removed

*Behavior change*

For MEX files built in R2021b and earlier, MATLAB provided a macro, `FORTRAN_COMPLEX_FUNCTIONS_RETURN_VOID`, to handle platform-dependent calling syntax differences for passing complex numbers to Fortran BLAS and LAPACK functions. As of R2022a, you no longer need a different calling syntax on different platforms, and the macro for handling this difference has been removed.

To update your code, replace statements such as these using `FORTRAN_COMPLEX_FUNCTIONS_RETURN_VOID`:

```
/* Call BLAS function */
/* Use a different call syntax on different platforms */
#ifdef FORTRAN_COMPLEX_FUNCTIONS_RETURN_VOID
    zdotu(&result, &nElements, zinA, &incx, zinB, &incy);
#else
    result = zdotu(&nElements, zinA, &incx, zinB, &incy);
#endif
```

with:

```
/* Call BLAS function */  
zdotu(&result, &nElements, zinA, &incx, zinB, &incy);
```

See `dotProductComplex.c`.

### **NET.addAssembly no longer removes leading spaces for Windows drive letter paths**

#### *Behavior change*

Before R2022a, on Windows platforms, the `NET.addAssembly` function removed leading spaces in paths specifying the drive letter. If the full path to your assembly is invalid with leading spaces, then remove the spaces before calling `NET.addAssembly`.





# R2021b

---

**Version: 9.11**

**New Features**

**Bug Fixes**

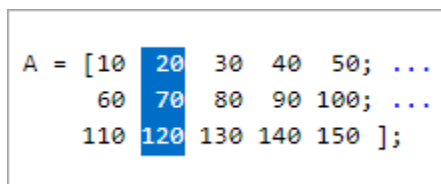
**Compatibility Considerations**

## Environment

### Editor Selection: Select and edit a rectangular area of code

In the Editor, you now can select a rectangular area in your code (also known as *column selection* or *block edit*) by pressing the **Alt** key while making a selection with the mouse. On macOS systems, use the **Option** key instead. Selecting and editing a rectangular area of code is useful if you want to copy or delete several columns of data, or if you want to edit multiple lines at one time.

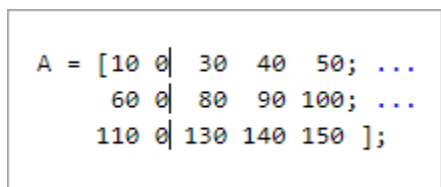
For example, select the second column of data in A.



```
A = [10 20 30 40 50; ...
     60 70 80 90 100; ...
     110 120 130 140 150];
```

The image shows a MATLAB code editor with a matrix A. The second column, containing the values 20, 70, and 120, is highlighted with a blue background, indicating it has been selected using column selection.

Type 0 to set all the selected values to 0.



```
A = [10 0 30 40 50; ...
     60 0 80 90 100; ...
     110 0 130 140 150];
```

The image shows the same MATLAB code editor after the selection has been replaced with zeros. The second column now contains the values 0, 0, and 0.

### Editor Display: Zoom in and out in the Editor

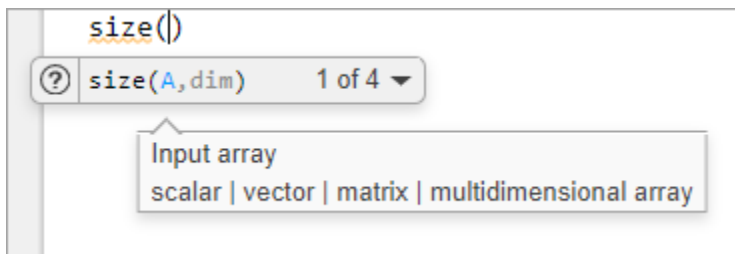
To zoom in or out in the Editor, go to the **View** tab, and in the **Zoom** section, select the  **Zoom In** or  **Zoom Out** button. As you zoom, MATLAB displays the current scale in the bottom-right corner of the Editor. You also can hold the **Ctrl** key and move the scroll wheel, or press **Ctrl+Plus** and **Ctrl+Minus**. On macOS systems, use the **Command** key and move the scroll wheel, or press **Command+Shift+Plus** and **Command+Shift+Minus**.

To return to the default scale, in the **View** tab **Zoom** section, select  **Reset Zoom**. You also can press **Ctrl+Alt+0** (**Command+Alt+0** on macOS).

### Editor Code: Show code suggestions and completions automatically

Starting in R2021b, when you write commands in the Editor, MATLAB automatically displays contextual hints for arguments, property values, and alternative syntaxes. In previous releases, MATLAB only completes names in the Editor after a **Tab** key press.

For example, if you want to use the `size` function, MATLAB automatically displays the syntax information to help you write the command as you type.



MATLAB also automatically suggests and completes the names of functions, models, MATLAB objects, files, folders, variables, structures, graphics properties, parameters, and options.

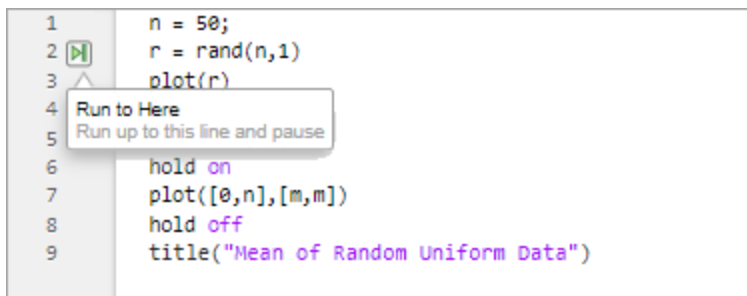
You can disable automatic completions in the Editor and Live Editor by having MATLAB suggest and complete names only after you press the **Tab** key. To do so, on the **View** tab, in the **Display** section, click the **Automatic Completions** button off. You also can go to the **Home** tab, and in the

**Environment** section, click **Preferences**. Then, select **Editor/Debugger > Automatic Completions** and in the **Suggestions and completions** section, select **Show on tab**.

For more information, see [Check Syntax as You Type](#).

## Editor Debugging: Diagnose problems in scripts and functions using inline debugging controls and a breadcrumb-style function call stack

When debugging code in the Editor, you now can diagnose problems using inline debugging controls. For example, to run to a specific line of code and then pause, click the button to the left of the line.



To step into a file, click the button directly to the left of the function you want to step into. After stepping in, click the button at the top of the file to run the rest of the called function, leave the called function, and then pause.

By default, the button only appears for user-defined functions and scripts. To show the button for MathWorks functions as well, on the **Home** tab, in the **Environment** section, click **Preferences**. Then, select **MATLAB > Editor/Debugger**, and in the **Debugging in the Live Editor** section, clear the **Only show Step in button for user-defined functions** option.

When you step into a called function or file, the Editor displays an improved breadcrumb-style list of the functions MATLAB executed before pausing at the current line (also called the function call stack). The function call stack is shown at the top of the file and displays the functions in order, starting on the left with the first called script or function, and ending on the right with the current script or function in which MATLAB is paused.



For more information, see [Debug MATLAB Code Files](#).

## Editor Refactoring: Automatically convert selected code to a function

Break large scripts or functions into smaller pieces by converting selected code into functions in files or local functions. With one or more lines of code selected, on the **Editor** tab, in the **Code** section, click the  **Refactor** button, and then select from the available options. MATLAB creates a function with the selected code and replaces the original code with a call to the newly created function.

## Editor Code: Automatically complete block endings, match delimiters, and wrap comments while editing code

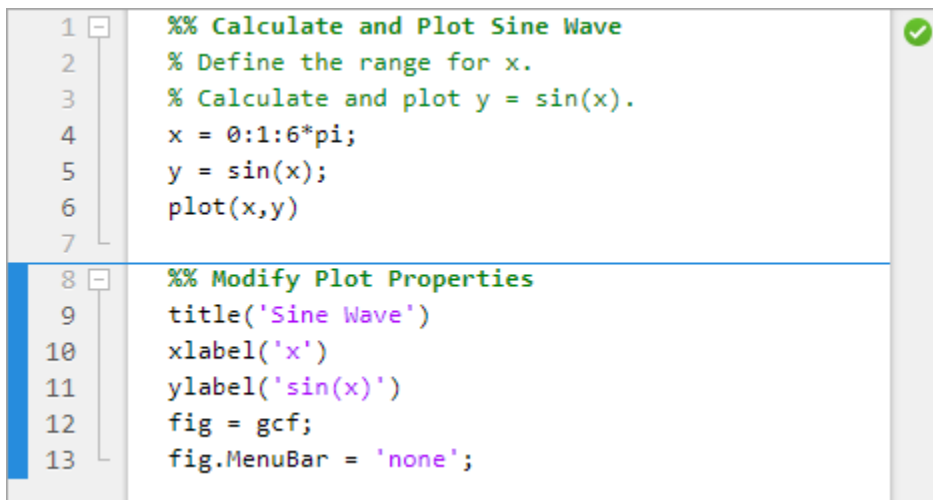
MATLAB now automatically completes parentheses and quotes when you enter code in the Editor. For example, if you type an open parenthesis in the Editor, MATLAB automatically adds the closing parenthesis. MATLAB also automatically completes comments, character vectors, strings, and parentheses split across two lines.

You also can have MATLAB automatically complete block endings. To do so, on the **Home** tab, in the **Environment** section, click  **Preferences**. Select **Editor/Debugger > Automatic Completions** and in the **Autocoding options** section, select one or more of the **Autocomplete block endings** options.

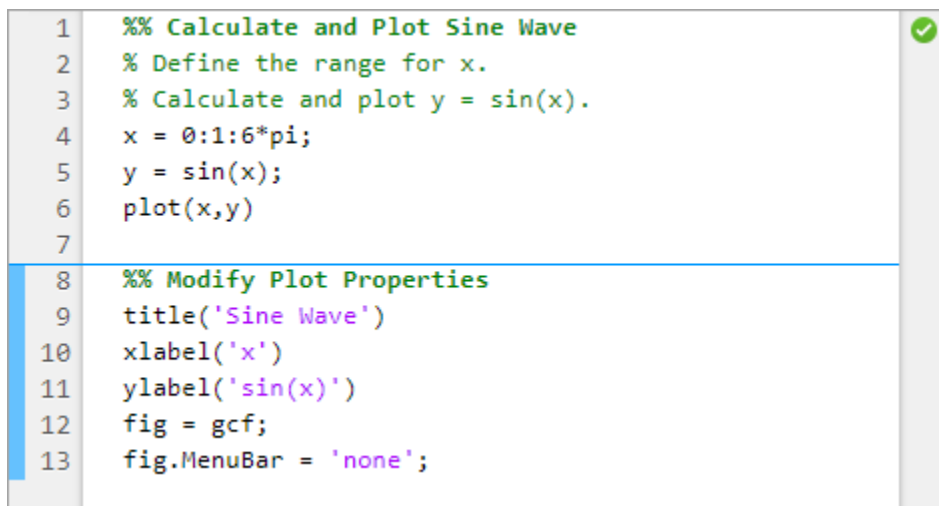
To undo an automatic code completion, press **Ctrl+Z** or the **Undo** button. To disable automatic code completions, in the **Editor/Debugger > Automatic Completions** preferences, clear one or more of the options in the **Autocoding options** section.

## Editor Sections: Create sections with an improved appearance

Starting in R2021b, sections in the Editor have an improved appearance. To create a new section, go to the **Editor** tab and in the **Section** section, click the **Section Break** button. The new section is highlighted with a blue border, indicating that it is selected.



To maximize the space available for editing code in the Editor, you can hide the Run to Here and Code Folding margins. This minimizes the gray area to the left of your code. To hide the two margins, right-click the gray area to the left of your code and clear the **Show Run to Here Margin** and **Show Code Folding Margin** options.



```

1  %% Calculate and Plot Sine Wave
2  % Define the range for x.
3  % Calculate and plot y = sin(x).
4  x = 0:1:6*pi;
5  y = sin(x);
6  plot(x,y)
7
8  %% Modify Plot Properties
9  title('Sine Wave')
10 xlabel('x')
11 ylabel('sin(x)')
12 fig = gcf;
13 fig.MenuBar = 'none';

```

As part of this change, the options for changing the appearance of code sections in the Editor have been removed. These options were previously available in the **MATLAB > Colors > Programming Tools** preferences, in the **Section display options** section.

For more information about sections in the Editor, see [Create and Run Sections in Code](#).

## Editor Code: Change the case of text and code

You can change the case of selected text or code in the Editor from all uppercase to lowercase, or vice versa. To change the case, select the text, right-click, and select **Change Case**. You also can press **Ctrl+Shift+A** to change the case. If the text contains both uppercase and lowercase text, MATLAB changes the case to all uppercase.

## Editor Bookmarks: Maintain bookmarks after closing a file

Starting in R2021b, MATLAB maintains all bookmarks after you close a file in the Editor. In previous releases, MATLAB does not maintain bookmarks after closing a file.

For more information, see [Go To Location in File](#).

## Live Editor Controls: Set default values for sliders, drop-down lists, check boxes, and edit fields

You can set the default values for sliders, drop-down lists, check boxes, and edit fields in your live scripts. To set the default value for a control, right-click the control and select **Configure Control**. Then, in the **Defaults** section, specify a default value by entering the value or by selecting a workspace variable from the list. The list shows only valid variables for the control. For drop-down lists, select the default value from the list of items.

To restore the default value for a control, right-click the control and select **Restore Default Value**.

For more information, see [Add Interactive Controls to a Live Script](#).

## Live Editor Animations: Export animations to movies or animated GIFs

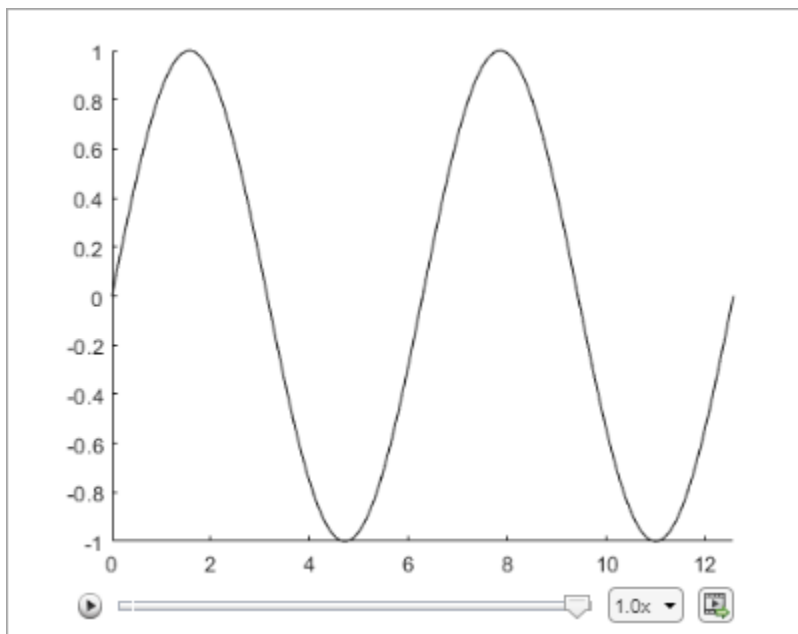
Export animations to movies or animated GIFs using the new  Export Animation button in the Live Editor animation playback controls. The Export Animation button is not supported for animations generated by the `movie` function.

For example, this code animates a line growing as it accumulates 2000 data points in the Live Editor.

When the animation is done playing, playback controls, including the new  Export Animation button, display within the figure window.

```
h = animatedline;
axis([0 4*pi -1 1])
x = linspace(0,4*pi,2000);

for k = 1:length(x)
    y = sin(x(k));
    addpoints(h,x(k),y);
    drawnow
end
```



For more information about creating animations, see [Animation Techniques](#).

## Live Editor Figures: Interact with real MATLAB figures and resize them with improved layouts

Live Editor output figures are now real MATLAB figures with most of the interaction capabilities of standalone MATLAB figures. In addition, when you resize a figure in the Live Editor, the font sizes and spacing between elements in the figure now automatically adjust to provide the best possible presentation for the new size.

## Live Editor: Improved performance when saving live scripts or functions

Saving live scripts and live functions in the Live Editor is faster in R2021b than in R2021a. The improvement is most noticeable when you save live functions with more than 1000 lines of code and live scripts with fewer than 100 lines of code.

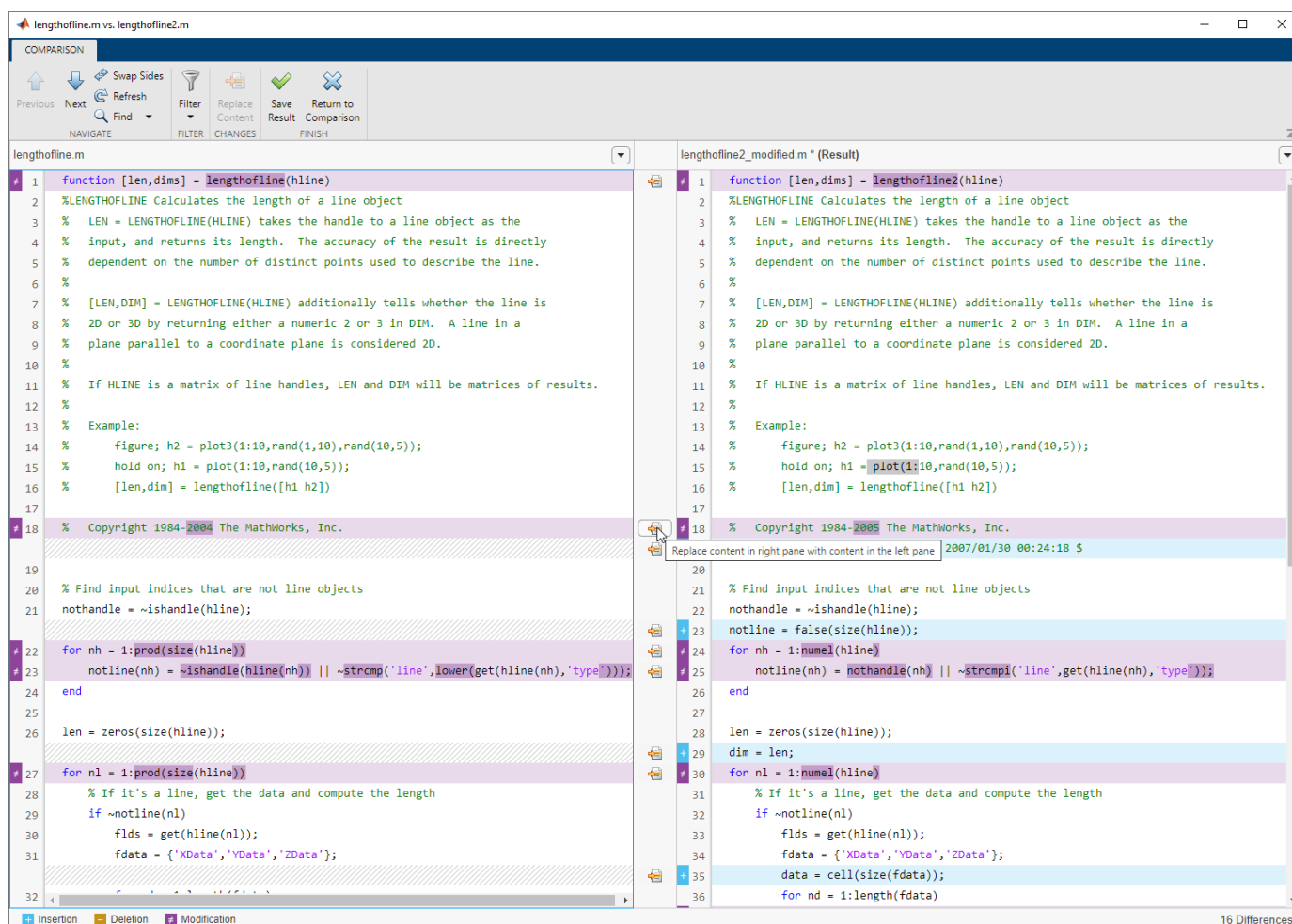
For example, on a Windows 10, Intel Xeon E5-1650 CPU @ 3.60 GHz test system, saving an example live function containing 4000 lines of code takes approximately 2.05 seconds in R2021b. In R2021a, saving the same live function takes approximately 2.57 seconds.

## Comparison Tool: Compare and merge text files with improved usability, appearance, and syntax highlighting

In R2021b, the comparison tool uses MATLAB Editor syntax highlighting. Text changes and merge choices are now easier to understand. Changes are highlighted with strong colors. Modified lines are

highlighted and flagged with the comparison icons , , or . Merging line by line is

straightforward, and merge choices are flagged with the merge content icon .



## ⚠ Compatibility Considerations

Starting in R2021b, you no longer can save the comparison report as HTML or specify whether to show only the differences or the entire files.

## Importing Preferences from Previous Releases: MATLAB checks for preferences from R2019b or newer

During start up, MATLAB checks for a preferences folder that matches the current release. If that folder is not found, MATLAB checks for preferences folders going back as far as R2019b. Releases before R2021b continue to check for up to three previous releases.

| MATLAB Release | Previous Release Preferences Folders |
|----------------|--------------------------------------|
| R2021b         | R2021a, R2020b, R2020a, R2019b       |
| R2021a         | R2020b, R2020a, R2019b               |
| R2020b         | R2020a, R2019b, R2019a               |

## Display language: MATLAB uses Windows display language settings for selecting desktop language

MATLAB uses the **Windows display language** settings on Windows 10 to control the MATLAB desktop language. The display language you select on Windows changes the default language used by Windows features like settings and file explorer.

Prior to R2021b, MATLAB controlled the desktop language using the Windows locale setting which is managed in the **Region** settings.

For information about Windows locale settings in MATLAB, see [Set Locale on Microsoft Windows Platforms](#).

For information about managing display language settings, refer to your Windows 10 documentation.

## ⚠ Functionality being removed or changed

### Increment Value and Run Section tool has been removed

The Increment Value and Run Section tool previously available in the Editor has been removed.

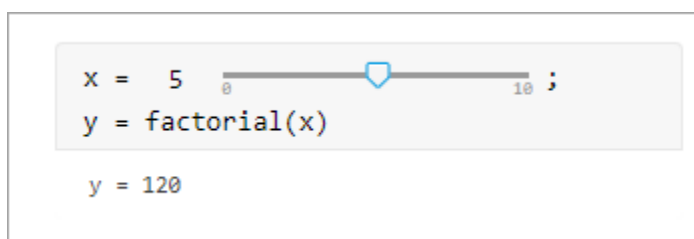
To increment a numeric value within a section, use controls in the Live Editor. For example, this code calculates the factorial of the variable `x`.

```
x = 5;
y = factorial(x)

y =
    120
```

To interactively change the value of `x`, in a live script, replace the value 5 with a numeric slider. By default, MATLAB reruns the current section when the value of the slider changes.





For more information, see [Add Interactive Controls to a Live Script](#).

## Language and Programming

### **cast Function: Consistent output for all syntaxes with the same data type conversion**

The `cast` function now returns consistent output for all syntaxes with the same data type conversion.

For example, starting in R2021b, both `b = cast(fi(1), 'like', sym(1))` and `b = cast(fi(1), 'sym')` return `b = 1` of the `sym` data type. In previous releases, `b = cast(fi(1), 'like', sym(1))` returns `b = 1` of the `sym` data type, but `b = cast(fi(1), 'sym')` throws an error.

### **Run Code in the Background: Use parallel language to run code asynchronously**

You can now run code in the background using `backgroundPool`. When you run code in the background, you can:

- Run other MATLAB code at the same time as a long running calculation
- Create more responsive user interfaces

Use the background pool with the following parallel language:

- `parfeval` and related functionality such as `afterEach` and `afterAll`
- `parallel.pool.DataQueue` and related functionality such as `afterEach`
- `parallel.pool.PollableDataQueue` and related functionality such as `poll`

For more information, see [Background Processing](#).

### **Portable Parallel Code: Share parallel code and seamlessly run in parallel**

You can now run parallel code even if you do not have Parallel Computing Toolbox. When you run portable parallel code without Parallel Computing Toolbox, you run the code in serial on your machine. When you run this code with Parallel Computing Toolbox, you can automatically scale up and run the code in parallel on your local machine, on a remote cluster, or in the cloud.

The following parallel language features are available for prototyping:

- `parfeval` — Seamlessly run multiple functions at once (since R2021b)
- `parfor` — Seamlessly run for-loops in parallel (since R2008a)

For more information, see [Run Parallel Language in MATLAB](#).

## Compact Display for Classes: Customize display of information about classes when space is limited

Use the `matlab.mixin.CustomCompactDisplayProvider` class to customize how information about your classes is displayed in a container variable—such as a struct, cell array, or table—where space is limited. Options for customization include:

- Displaying partial sets of data
- Adding annotations
- Controlling how and when class names are displayed

For example, an enumeration class of the days of the week, `WeekDays`, can be customized so that when arrays of `WeekDays` members cannot be fully displayed, MATLAB displays the size, the class, and an annotation. When the width of the Command Window is large enough, the full array is shown:

```
myStruc =  
  
    struct with fields:  
  
    prop1: [Monday    Wednesday    Friday    Saturday    Sunday]
```

When the Command Window is not wide enough to display the full array, the size, the class, and an annotation are shown:

```
myStruc =  
  
    struct with fields:  
  
    prop1: 1x5 Weekdays (Enum of days of week)
```

## Class Aliasing: Create aliases for renamed classes to maintain backward compatibility

When you need to change a class name, you can create an alias to preserve compatibility with code written before the name change. The `matlab.alias.AliasFileManager` class provides an API for defining and implementing aliases. Once you define an alias, you can use the alias anywhere you use the class name. Aliases maintain backward compatibility when reloading objects with the old name into a MATLAB version that uses the new name, as well as forward compatibility when loading objects saved in a newer version into an older version that predates the definition of the alias. MATLAB substitutes the new class name whenever it encounters the old name.

## Modular Indexing: Customize class indexing operations individually using new superclasses

To customize how indexing operations behave with your class in previous versions of MATLAB, you need to overload the `subsref` and `subsasgn` methods. Doing so requires implementing code for all indexing reference and assignment operations, including parentheses, dot, and brace operations, even if you want to change only one type of indexing operation.

Starting in R2021b, you can inherit from three new superclasses to customize parentheses, dot, and brace indexing operations individually:

- `matlab.mixin.indexing.RedefinesParen` – Customize parentheses reference, assignment, and deletion operations.
- `matlab.mixin.indexing.RedefinesDot` – Customize dot reference and assignment operations.
- `matlab.mixin.indexing.RedefinesBrace` – Customize brace reference and assignment operations.

You can inherit from one or more of these classes without affecting how the other indexing operations work. These classes also enable you to forward levels of indexing in compound statements to other MATLAB values. For example, your class can implement custom parentheses indexing for the first level of a compound reference and then allow MATLAB to apply the other levels to an object contained by your class.

## Scalar Classes: Inherit from the `matlab.mixin.Scalar` superclass to ensure instances behave as scalars

Instances of classes that inherit from `matlab.mixin.Scalar` must behave as scalars. You cannot form arrays of instances of such a class, including empty arrays, and you cannot concatenate instances. This class is useful for cases in which concatenation does not make sense, like a dictionary or other container class in which parentheses indexing is customized to access data in the class, not to form or access arrays.

## startat Function: Time zone information in datetime objects now supported

The `startat` function now recognizes time zone and daylight savings time information of `datetime` inputs.

## ⚠️ Functionality being removed or changed

### **newclass input argument of the syntax `cast(A,newclass)` is now case-sensitive**

*Behavior change*

Starting in R2021b, the `newclass` input argument of the syntax `cast(A,newclass)` is case-sensitive. You must specify `newclass` as a character vector or a string of lowercase letters that represents the new data type.

For example, to convert a double value to the `int8` data type, you must use `b = cast(1.234,'int8')`. The function syntax `b = cast(1.234,'Int8')` now throws an error.

### **Defining classes and packages: Using `schema.m` will not be supported in a future release**

*Still runs*

Support for classes and packages defined using `schema.m` files will be removed in a future release. Replace existing schema-based classes with classes defined using the `classdef` keyword.

### **InexactCaseMatch and InexactCaseMatchForExtension Errors: These errors are replaced by UndefinedFunction error**

*Behavior change*

The `InexactCaseMatch` and `InexactCaseMatchForExtension` errors have been removed and MATLAB throws an `UndefinedFunction` error instead. This change does not generate new errors in code that did not previously throw an error.

## Data Analysis

### **Compute by Group Live Editor Task: Interactively summarize, transform, or filter groups of data**

Use the Compute by Group Live Editor task to interactively compute statistics, transform data, or filter data by group. To open the task in the Live Editor, use the **Task** menu on the **Live Editor** tab.

### **Normalize Data Live Editor Task: Interactively center and scale data**

Use the Normalize Data Live Editor task to visualize the effects of centering and scaling data using various methods, such as the z-score. To open the task in the Live Editor, use the **Task** menu on the **Live Editor** tab.

### **Clean Missing Data Live Editor Task: Define missing values**

When using the Clean Missing Data Live Editor task, you can now define missing value indicators that are different from the standard MATLAB missing values.

### **trenddecomp Function: Find trends in data**

Use the `trenddecomp` function to additively decompose data into a long-term trend and seasonal trends.

### **min and max Functions: Specify the comparison method for determining minimum and maximum values**

The `min` and `max` functions now accept the `'ComparisonMethod'` parameter, which specifies a method for determining the minimum and maximum values of the input while preserving the sign in the output.

### **unique\_tol Function: Options to control element selection and preserve range of data**

`unique_tol` has two new options to control behavior:

- `occurrence` argument: Specify whether the algorithm begins with the highest or lowest elements in the input data. This can change which element, among several that are within tolerance of each other, is selected as being unique. The default is to begin with the lowest elements.
- `'PreserveRange'` name-value argument: Specify whether the range of the output data should be the same as the input data.

### **Data Preprocessing Functions: Specify table variable as sample points vector**

When you operate on table input data, the following functions now allow you to specify which variable in the table to use with the `'SamplePoints'` parameter:

- `fillmissing`
- `filloutliers`
- `ischange`
- `islocalmax`
- `islocalmin`
- `isoutlier`
- `rmoutliers`
- `smoothdata`

### **dateshift Function: Shift to next occurrence of weekday or weekend day**

You can now use the `'weekday'` and `'weekend'` arguments to shift the elements of a `datetime` array when using the `dateshift` function.

- To shift to the next occurrence of a weekday on or after each element of the input `datetime` array, use `'weekday'`.
- To shift to the next occurrence of a weekend day on or after each element of the input `datetime` array, use `'weekend'`.

### **isbetween Function: Support for open, closed, and half open intervals**

The `isbetween` function now supports open, closed, and half open intervals. In previous releases, `isbetween` supports only closed intervals.

### **isregular Function: Support for datetime and duration data types**

You can now use the `isregular` function to determine if a `timetable`, `datetime` vector, or `duration` vector is regular. In previous releases, you can use `isregular` only on a `timetable`.

### **istabular Function: Determine if input is a table or timetable**

To determine if an input variable is either a `table` or a `timetable`, use the `istabular` function.

Using this function is equivalent to using the statement `tf = istable(A) || istimetable(A)`, but is more convenient.

### **retime and synchronize Functions: Median and mode methods supported**

When you synchronize data in `timetables`, you can now specify `'median'` and `'mode'` as aggregation functions. For more information, see `retime` and `synchronize`.

### **timeofday Function: Return the date as the second output argument**

You can now return the dates from the elements of a `datetime` array as the second output argument from the `timeofday` function.

## **timeseries2timetable Function: Convert timeseries objects to timetables**

To convert `timeseries` objects to `timetables`, use the `timeseries2timetable` function.

### **⚠️ Functionality being removed or changed**

#### **`isordinal` accepts input argument that has any data type**

*Behavior change*

The `isordinal` function now accepts an input argument that has any data type. In previous releases, `isordinal` threw an error if the input argument was not a categorical array.

#### **Synchronize Timetables Live Editor task synchronizes an unlimited number of timetables**

*Behavior change*

The Synchronize Timetables Live Editor task can now synchronize an unlimited number of timetables. In previous releases, the task can synchronize no more than five timetables.

#### **`timeseries2timetable` replaces `ts2timetable`**

*Behavior change*

The `timeseries2timetable` function replaces the `ts2timetable` function, although `ts2timetable` is still provided. The two functions are synonyms. In R2021a, MATLAB provides `ts2timetable` only.



## Data Import and Export

### sftp Function: Connect to SFTP servers

MATLAB can connect to SFTP servers for encrypted data transfers. Create an SFTP connection object using the `sftp` function to read data from an SFTP server.

### Datastores: Specify FileSet objects as data locations for some datastores

Some datastore functions and objects accept `FileSet` objects as the locations of files to include in the datastore. `FileSet` objects provide increased performance compared to file paths or `DsFileSet` objects. This functionality is supported by these functions:

- `tabularTextDatastore`
- `spreadsheetDatastore`
- `fileDatastore`
- `keyValueDatastore`
- `tallDatastore`
- `parquetDatastore`
- `imageDatastore`
- `signalDatastore` (Signal Processing Toolbox)
- `audioDatastore` (Audio Toolbox)
- `mdfDatastore` (Vehicle Network Toolbox)

### Table Import: Read tables from HTML and Microsoft Word documents

The `readtable` function now supports reading tables from HTML and Microsoft Word files.

To customize import options for HTML and Microsoft Word files, use `htmlImportOptions` and `wordDocumentImportOptions`, respectively. To automatically detect import options from files, use the `detectImportOptions` function.

### HDF5 Interface: Use new functionality in support of HDF5 1.10.7

Use these new capabilities of the MATLAB HDF5 function interfaces:

- Single-Writer/Multiple-Reader (SWMR) — Write data to an HDF5 file in one process while you concurrently read from the file in one or more reader processes. For more information, see [Read and Write Data Concurrently Using Single-Writer/Multiple-Reader \(SWMR\)](#).
- Virtual Dataset (VDS) — Use the MATLAB low-level interface to access data stored across multiple HDF5 files, including files in remote locations, as a single, unified HDF5 dataset. You can also read data stored in Virtual Datasets using the HDF5 high-level interface. For more information, see [Work with HDF5 Virtual Datasets \(VDS\)](#).
- Metadata Cache Fine-Tuning — Improve performance by controlling the parameters of the metadata cache, such as limiting the number of file reading attempts.

- Partial Edge Chunk — Control whether to filter partial edge chunks.

## **NetCDF Interface: Read and write NC\_STRING data**

You can now use the existing high-level and low-level functions to read NC\_STRING data from NetCDF-4 files and write text data as type NC\_STRING.

For more information on data type mapping between the NetCDF API and MATLAB, see [Map NetCDF API Syntax to MATLAB Syntax](#).

## **Scientific File Format Libraries: HDF5 and NetCDF libraries are upgraded**

The HDF5 library is upgraded to version 1.10.7, and the NetCDF library is upgraded to version 4.7.4.

## **Audio, Video, and Image I/O Functions: Run functions in a thread-based environment**

You can now run the following functions in the background using MATLAB backgroundPool:

- `audioread`
- `audiowrite`
- `imwrite`
- `VideoReader`
- `VideoWriter`

For more information, see [Run MATLAB Functions in Thread-Based Environment](#).

## **Image File Format Libraries: LibTIFF library upgraded to version 4.2.0**

The LibTIFF library is upgraded to version 4.2.0.

## **New Serial Explorer and TCP/IP Explorer apps**

Two new apps offer functionality for communicating with your device, instrument, or server:

- The Serial Explorer app provides a user interface to connect to and communicate with a serial port device on your machine.
- The TCP/IP Explorer app provides a user interface to create a TCP/IP client that communicates with a TCP/IP server.

Launch these apps from the **Apps** tab, under the **Test and Measurement** section. You can also call the `serialExplorer` and `tcpipExplorer` commands in the Command Window.

You can use the apps to perform the following operations on your serial port device or TCP/IP client.

- Configure connection and communication properties.
- Write binary or string data.

- Read binary or string data.
- Plot data in a separate figure window.
- Analyze data by viewing it in the Signal Analyzer app.
- Export data to the MATLAB workspace.
- Generate a MATLAB script for app interactions that uses the `serialport` or `tcpclient` interface.

For more information about these apps, see [Serial Explorer](#) and [TCP/IP Explorer](#).

## ▲ **Functionality being removed or changed**

### **Video and Image I/O Functions: Pixel value differences might exist between JPEG 2000 images in R2021b and previous versions of MATLAB**

*Behavior change*

In R2021b, when you use the `imread`, `imwrite`, `VideoReader`, or `VideoWriter` functions to read or write JPEG 2000 image files, the image you import or export in R2021b might have pixel value differences with the same image in previous versions of MATLAB.

### **HDF5 Interface: Linux users need to rebuild filter plugins using MATLAB HDF5 1.10.7 shared library**

*Behavior change*

Starting in R2021b, in certain cases, Linux users using a filter plugin with callbacks to core HDF5 library functions need to rebuild the plugin using the shipping MATLAB HDF5 1.10.7 shared library, `/matlab/bin/glnxa64/libhdf5.so.103.3.0`. If you do not rebuild the plugin using this version of the shared library, you might experience issues ranging from undefined behavior to crashes. For more information, see [Build HDF5 Filter Plugins on Linux Using MATLAB HDF5 Shared Library or GNU Export Map](#).

### **ftp Function: FTPClientConfig class, properties, and methods are no longer supported**

The `ftp` function no longer supports the Apache® `FTPClientConfig` class or any associated objects, properties, or methods. To customize how to parse the `LIST` command output of the FTP server use the `ftp` function's `DirParserFcn` name-value argument.

### **MATLAB Variable Editor: timeseries will no longer be supported in a future release**

*Still runs*

Viewing `timeseries` objects using the MATLAB Variable Editor will no longer be supported in a future release. To view time-indexed data in the Variable Editor, use `timetable` instead.

## Mathematics

### **ode78 and ode89 Functions: High-order Runge-Kutta solvers for ordinary differential equations**

The MATLAB ODE suite has been expanded with two new solvers:

- `ode78` uses 7th- and 8th-order Runge-Kutta formulas
- `ode89` uses 8th- and 9th-order Runge-Kutta formulas

The new solvers expand on the existing Runge-Kutta solvers `ode23` and `ode45`. In particular, `ode78` and `ode89` can be more efficient than `ode45` on nonstiff problems that are smooth, and `ode89` can be more efficient than `ode78` on very smooth problems, when you integrate over long time intervals or when tolerances are tight.

### **pagesvd Function: Perform singular value decomposition on pages of N-D arrays**

Use the `pagesvd` function to perform batched singular value decompositions on the pages of N-D arrays. In this context, the N-D array is treated as a container for several 2-D matrices.

### **svd Function: Option to control output format of singular values**

`svd` has a new option `outputFormat` to control whether the singular values are returned as a vector or diagonal matrix.

### **mpower Function: Improved algorithm for defective matrices**

The `mpower` function has an improved algorithm to handle defective matrices raised to a real power. In previous releases, `mpower` uses an algorithm based on eigenvalue decomposition for these inputs that can return incorrect results for defective matrices. The new algorithm for defective matrices is instead based on the Schur decomposition.

### **⚠️Functionality being removed or changed**

#### **`svd`, `eig`, `cond`, and `pinv` functions return NaN for nonfinite inputs**

*Behavior change*

In R2021b, the `svd`, `eig`, `cond`, and `pinv` functions return NaN values when the input contains nonfinite values (`Inf` or `NaN`).

In previous releases, these functions throw an error when the input contains nonfinite values.

## Graphics

### Plotting Table Data: Create scatter plots, bubble charts, and swarm charts by passing tables directly to plotting functions

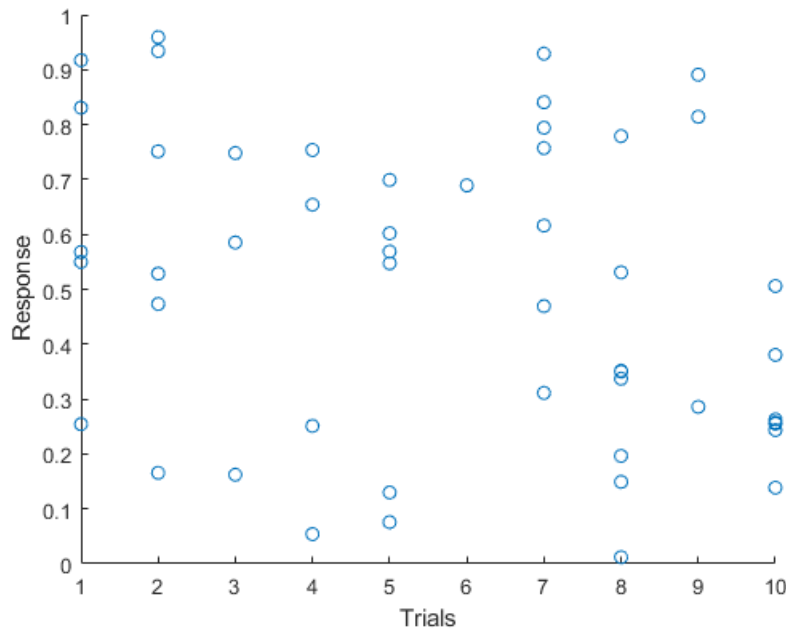
Create plots by passing a table directly to any of these functions: `scatter`, `scatter3`, `bubblechart`, `bubblechart3`, `swarmchart`, `swarmchart3`, `polarscatter`, and `polarbubblechart`. When you specify your data as a table, the axis labels and the legend (if present) are automatically labeled using the table variable names.

The objects returned by these functions have new properties to support tables.

| Property                            | Description                                                            |
|-------------------------------------|------------------------------------------------------------------------|
| SourceTable                         | Table containing the data to plot                                      |
| XVariable, YVariable, and ZVariable | Table variables containing the x, y, and z values for Cartesian plots  |
| ThetaVariable and RVariable         | Table variables containing the angle and radius values for polar plots |
| SizeVariable                        | Table variable containing the marker size data                         |
| ColorVariable                       | Table variable containing the marker color data                        |
| AlphaVariable                       | Table variable containing the marker transparency data                 |

For example, create a table with the variables "Trials" and "Response". Pass the table to the `scatter` function as the first argument, and indicate the variables you want to plot by name.

```
Trials = randi(10,50,1);  
Response = rand(50,1);  
t = table(Trials,Response);  
scatter(t,"Trials","Response")
```



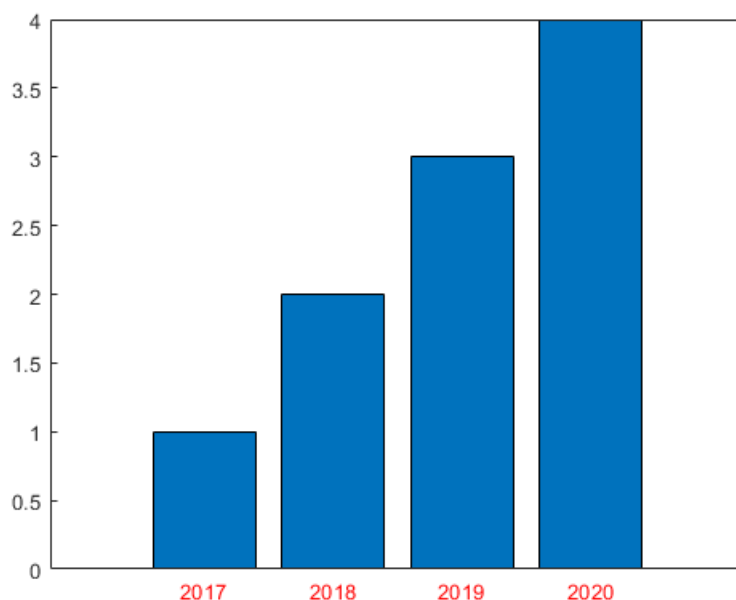
## Axes Ticks and Colors: Control the appearance of axis tick marks and tick label colors

Now, you can remove tick marks and customize tick label colors independently of other elements in the axes.

- **Removing Tick Marks** — Remove all the tick marks from an axes, polar axes, or geographic axes object by setting the `TickDir` property to `'none'`. To remove the tick marks from a specific axis, for example the x-axis, set the `TickDirection` property of the ruler to `'none'`.
- **Customizing Tick Label Colors** — Customize the color of the tick labels on an axis by setting the `TickLabelColor` property of the corresponding ruler object. You can customize tick label colors for any axes, polar axes, or geographic axes.

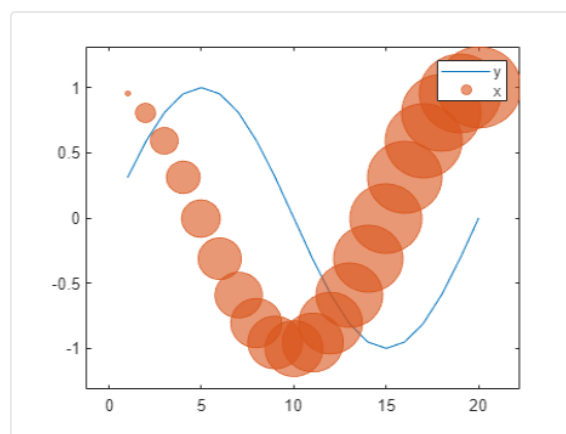
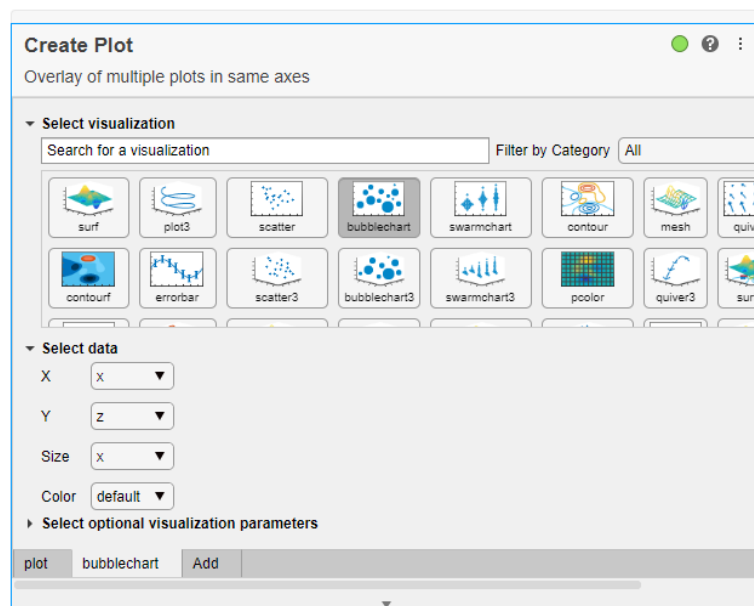
For example, create a bar chart, and then get the current axes. Remove the x-axis tick marks and change the color of the x-axis tick labels to red.

```
bar([2017 2018 2019 2020],1:4)
ax = gca;
ax.XAxis.TickDirection = 'none';
ax.XAxis.TickLabelColor = 'r';
```



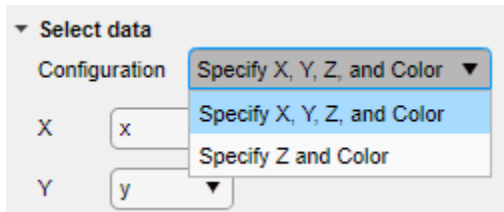
## Create Plot Live Task: Add additional visualizations to generated plots

Easily add additional visualizations to plots generated using the Create Plot Live Task. To add a new plot, click the **Add** tab at the bottom of the Live Task Panel and select the visualization and data. This Live Task combines the plot using the hold function.



## Create Plot Live Task: Control chart input syntax using configuration drop-down

The Create Plot Live Task now supports multiple configurations of charting functions with multiple input syntaxes, including `surf` and `mesh`. Use the **Configurations** drop-down menu to select the desired configuration.



## exportgraphics Function: Capture and append graphics to existing PDFs

Capture and append graphics to an existing PDF file by calling the `exportgraphics` function and setting the 'Append' name-value argument to `true`. For example, create a plot and export it as a PDF called 'mycharts.pdf'. Then, create a bar chart and append it to the end of 'mycharts.pdf'. The resulting PDF file has two pages. The plots appear in the PDF in the order that you export them.

```
plot([0 3 1 6 4 10])
exportgraphics(gca,'mycharts.pdf')
bar([10 20 30 40])
exportgraphics(gca,'mycharts.pdf','Append',true)
```

## stackedplot Function: Support for semilog y-axes

You can create plots using the `stackedplot` function where individual y-axes can be plotted on a log scale. To set a log scale for the y-axis of a plot, set the `YScale` property of the `StackedAxesProperties` object associated with the plot. For more information, see `StackedAxesProperties` Properties.

## Text Objects: Use editInteractions in the Interactions property to click or tap on text to edit

Click or tap to edit text when the `Interactions` property has the value `editInteraction`. The edit interaction is default behavior for title, subtitle, xlabel, ylabel, and zlabel text objects for axes, geographic axes, and polar axes.

## dataTipTextRow Function: Customize data tip content using data properties, such as UserData

You can now assign information to the `DataTipTemplate` property by passing it to the `dataTipTextRow` function as a property name, such as `UserData`.



```
p = patch;
p.UserData = p.XData;
p.DataTipTemplate.DataTipRows(3) = dataTipTextRow('XDataAsUserData', 'UserData');
```

## MATLAB Online™ Accessibility: Use a screen reader to interact with figures

In MATLAB Online™, you can use a screen reader and keyboard commands to pan, zoom, and rotate when you work with plotted data. Using a screen reader is not supported in the Live Editor.

For more information, see [Use a Screen Reader in MATLAB Online](#).

For more details on interacting with MATLAB figures, see [Control Chart Interactivity](#).

## ⚠️ Functionality being removed or changed

### The print options `-opengl` and `-painters` are not recommended

*Still runs*

The following print options are no longer recommended. There are no plans to remove the values, and they will continue to behave the same way as in previous releases. The following table lists the recommended replacement options.

| Not Recommended                                                                                                      | Replacement Option                                                                                          |
|----------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------|
| The <code>-opengl</code> renderer option. For example:<br><code>print('-opengl', '-dpdf', 'myfigure.pdf')</code>     | Use the <code>-image</code> option. For example:<br><code>print('-image', '-dpdf', 'myfigure.pdf')</code>   |
| The <code>-painters</code> renderer option. For example:<br><code>print('-painters', '-dpdf', 'myfigure.pdf')</code> | Use the <code>-vector</code> option. For example:<br><code>print('-vector', '-dpdf', 'myfigure.pdf')</code> |

### plottools functions will be removed in a future release

*Still runs*

The `plottools` functions listed below will be removed in a future release. Use `inspect` to launch the Property Inspector instead.

#### plottools functions

- `plottools`
- `showplottool`
- `figurepalette`
- `plotbrowser`
- `propertyeditor`
- `propedit`
- `plottedit` options `'showtoolsmenu'` and `'hidetoolsmenu'`

## App Building

### **UIAlert, UIConfirm, and UIProgressDlg Functions: Mark up text and display equations in dialog boxes**

When you create dialog boxes using the `UIAlert`, `UIConfirm`, and `UIProgressDlg` functions, enable markup in the dialog box text using the `Interpreter` name-value argument. Specify the interpreter as `'html'`, `'latex'`, `'tex'`, or `'none'`.

### **addStyle Function: Add styles to nodes and levels in a tree UI component**

Create styles for specific tree nodes or tree node levels in a tree UI component using the `uistyle` and `addStyle` functions. For example, you can make the tree nodes at the top level of the tree red with italic font. To get information on applied styles, query the `StyleConfigurations` property of the `Tree` object. To remove a style from a tree, use the `removeStyle` function.

### **uitable Function: Set and query table selections programmatically and control table selection options**

You can now configure selection options of table UI components.

- Set and query the table selection using the `Selection` property.
- Specify whether a user can select table cells, rows, or columns using the `SelectionType` property.
- Specify whether a user can select single or multiple table elements using the `Multiselect` property.
- Update your app whenever a user selects table data by specifying a `SelectionChangedFcn` callback.

Selection options in table UI components are supported only in App Designer apps and in figures created with the `uifigure` function.

For more information, see [Table Properties](#).

### **uitextarea Function: Program apps to respond while a user is typing in a text area component**

You can now specify a `ValueChangingFcn` callback for a `TextArea` component. The component executes the callback function repeatedly while a user types in the text area.

For more information, see [TextArea Properties](#).

### **Run Code in the Background: Use parallel language to create more responsive apps**

You can now create apps that remain responsive while performing calculations in the background by using `backgroundPool`.

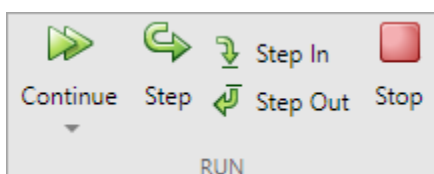
Use the background pool with the following parallel language:

- `parfeval` and related functionality such as `afterEach` and `afterAll`
- `parallel.pool.DataQueue` and related functionality such as `afterEach`

For more information, see [Use the Background to Make Your Apps More Responsive](#).

## App Designer: Debug code in Code View

When debugging code in App Designer, you now can diagnose problems using debugging controls in **Code View**. You can use the controls in the **Run** section of the **Editor** tab to run to the next breakpoint, run the next line of code, or step into or out of a function.



You can also debug your app code using inline debugging controls. For example, to run to a specific line of code and then pause, click the  button to the left of the line. To step into a function, click the  button directly to the left of the function you want to step into. After stepping in, click the  button at the top of the file to run the rest of the called function, leave the called function, and then pause.

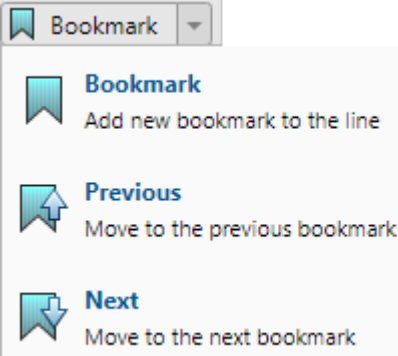
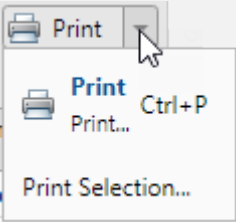
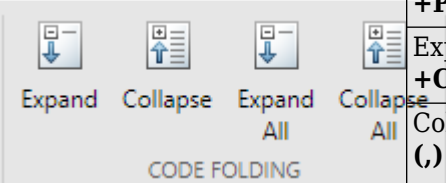
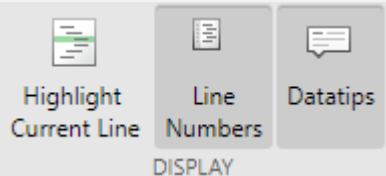
When you step into a called function or file, App Designer displays a breadcrumb-style list of the functions MATLAB executed before pausing at the current line (also called the function call stack). The function call stack is shown at the top of the file and displays the functions in order, starting on the left with the first called script or function, and ending on the right with the current script or function in which MATLAB is paused.



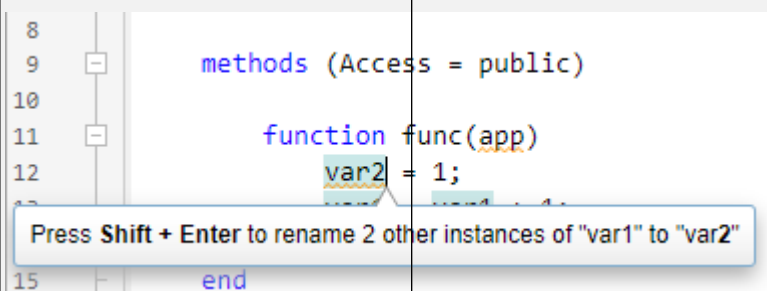
## ▲ App Designer: Efficiently manage your app code with tools and shortcuts from Live Editor

Many of the tools and shortcuts for navigating and organizing code that are available in the Live Editor can now be used in App Designer **Code View**. This table lists the functionality that is new to App Designer.

| Functionality                 | Menu Item                                                                                                                                                                                                                | Keyboard Shortcut                     |
|-------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------|
| Wrap comment                  | Right-click the comment and select <b>Wrap Comments</b> , or in the <b>Editor</b> tab, in the <b>Code</b> section, click the  button. | <b>Ctrl+J</b>                         |
| Navigate code using bookmarks | In the <b>Editor</b> tab, in the <b>Navigate</b> section, click <b>Bookmark</b> . Then, select                                                                                                                           | Set or clear bookmark: <b>Ctrl+F2</b> |

| Functionality              | Menu Item                                                                                                                                                                                                                     | Keyboard Shortcut                                    |
|----------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------|
|                            | <b>Bookmark</b> to set or clear a bookmark on the current line, or select <b>Previous</b> or <b>Next</b> to navigate between the existing bookmarks.                                                                          | Move to previous bookmark: <b>Shift+F2</b>           |
|                            |                                                                                                                                                                                                                               | Move to next bookmark <b>F2</b>                      |
|                            |                                                                                                                                             | <div>Ctrl+F2</div> <div>Shift+F2</div> <div>F2</div> |
| Print app code             | In the <b>Editor</b> tab, in the <b>File</b> section, click <b>Print</b> . You can print the entire document or the current selection.<br> | <b>Ctrl+P</b>                                        |
| Fold and expand code       | In the <b>View</b> tab, click the buttons in the <b>Code Folding</b> section.<br>                                                         | Expand current fold: <b>Ctrl+Shift+Period (.)</b>    |
|                            |                                                                                                                                                                                                                               | Collapse current fold: <b>Ctrl+Period (.)</b>        |
|                            |                                                                                                                                                                                                                               | Expand all folds: <b>Ctrl+Shift+Comma (,)</b>        |
|                            |                                                                                                                                                                                                                               | Collapse all folds: <b>Ctrl+Comma (,)</b>            |
| Toggle display preferences | In the <b>View</b> tab, in the <b>Display</b> section, toggle line highlighting and line numbers.<br>                                     | N/A                                                  |

| Functionality                          | Menu Item                                                                                                                               | Keyboard Shortcut                  |
|----------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------|------------------------------------|
| Duplicate line                         | Right-click a line and select <b>Duplicate Line(s)</b> .                                                                                | <b>Ctrl+Shift+C</b>                |
| Insert section break                   | Right-click a line and select <b>Section Break</b> .                                                                                    | <b>Ctrl+Alt+Enter</b> , or type %% |
| Convert text to uppercase or lowercase | Highlight the text, right-click it, and select <b>Change Case</b> .                                                                     | <b>Ctrl+Shift+A</b>                |
| Variable rename                        | When you rename a variable, App Designer gives you the option to automatically update all other instances of the variable in your code. | <b>Shift+Enter</b>                 |



## ⚠ Compatibility Considerations

Code folding in App Designer persists even after you close and then reopen the file. In R2021a and earlier releases, when opening a file, App Designer expands all the code.

## App Designer: Interactively modify canvas zoom level and fit canvas to view

In **Design View**, use the zoom controls in the lower right corner of the App Designer canvas, indicated by the  button, to modify the canvas zoom level.

To automatically zoom to fit the entire app in the view, press **Space**. Alternatively, click  **Fit to View** in the **Zoom** section of the **View** tab, or right-click on the canvas and select **Zoom > Fit to View**.

## App Designer: Convert between similar UI components

To convert one type of UI component to another with similar functionality, right-click the component on the canvas or in the **Component Browser** and select **Replace With**. Then, select the component to convert to. Replacing one component with another preserves relevant property values, such as font properties and callbacks that exist for both components. You can convert component types within each of these families:

- Numeric edit field, spinner, slider, and knob

- Edit field and text area
- Label and hyperlink

## App Designer: Add help text for your app

You can now provide help for apps that you create. Help text appears in the Command Window when an app user calls the `help` function and specifies the name of the app.

To add help text, in the **Editor** tab in **Code View**, click  **App Help Text**. Use the App Help Text dialog box to specify the app summary and detailed explanation.

In addition, when an app user views the documentation for your app (for example, by calling the `doc` function or by clicking the documentation link in the help text for the app), the documentation page now displays additional information:

- The top of the page displays the app summary and detailed explanation.
- The Methods Summary section displays the public functions. For each function, it also displays any comment that is inserted after the function definition statement.

## App Designer: Remove auto-reflow behavior from an app with auto-reflow

To convert an app with auto-reflow to an app without auto-reflow, in the **Canvas** tab in **Design View**, click **Convert**. Select the **App without Auto-Reflow** option. Doing so creates a duplicate of your app with the auto-reflow behavior removed.

For more information, see Apps with Auto-Reflow.

## Deployed Web Apps: Deploy web apps directly to the MATLAB Web App Server from within App Designer

Once you have MATLAB Compiler™ installed on the system running MATLAB, package your MATLAB app into a web app from within App Designer by clicking **Share** in the **Designer** tab and selecting **Web App**. In the packaging dialog, specify the server URL to directly deploy your web app to the server once packaging is complete. Authentication must be enabled on the server for this to work. For details, see Authentication (MATLAB Web App Server).

## App Testing Framework: Perform press gestures on axes and UI axes with different selection types

The app testing framework now supports mouse selection types in **press** gestures that are performed on axes and UI axes. For example, create an axes with a plot and then test a double-click gesture at the point (3, 2).

```
f = uifigure;  
ax = axes(f);  
plot(ax,1:10)  
tc = matlab.uitest.TestCase.forInteractiveUse;  
tc.press(ax,[3 2], 'SelectionType', 'open')
```

## App Testing Framework: Perform drag gestures on axes and figures with different selection types

Starting in R2021b, the app testing framework supports drag gestures on UI figures. Additionally, when you test a drag gesture on an axes, UI axes, or UI figure, you can specify the mouse selection type. For example, create a figure and drag on it from the point (100, 200) to the point (200, 300) using a right-click gesture.

```
f = uifigure;
tc = matlab.uitest.TestCase.forInteractiveUse;
tc.drag(f,[100 200],[200 300], 'SelectionType', 'alt')
```

## App Testing Framework: Use any units of measurement in gestures at the center of components

Starting in R2021b, when you perform a gesture at the center of a component, the component or its parent containers can use any units of measurement. In previous releases, the framework does not support containers that use nonpixel units.

For example, create a figure and set its `Units` property to `'normalized'`. Then, create a panel in the figure and press at the center of the panel.

```
f = uifigure;
f.Units = 'normalized';
p = uipanel(f);
tc = matlab.uitest.TestCase.forInteractiveUse;
tc.press(p)
```

If you perform a gesture at the center of a component using a syntax that accepts location as an input (for instance, `press(testcase, comp, location)`), then the figure or parent containers can use only `'pixels'` as their units of measurement.

## ⚠️ Functionality being removed or changed

### Ctrl+Click selects and deselects cells in a table UI component

*Behavior change*

In tables created using the `uitable` function, you can select and deselect noncontiguous table cells by holding **Ctrl** and clicking a cell. In R2021a and earlier releases, **Ctrl+Click** gives focus to a cell and **Shift+Click** selects the cell that has focus.

### App Designer toolstrip organization has changed

*Behavior change*

The organization of the tools in the App Designer toolstrip in **Design View** and **Code View** has changed.

In **Design View**, use the tools in the **Canvas** tab to lay out your app, and use the tools in the **View** tab to manage your **Design View** preferences.

In **Code View**, use the tools in the **Editor** tab to program your app behavior and to run and debug your app, and use the tools in the **View** tab to manage your **Code View** preferences.

### matlab.fonts.editor.codefont.Size setting has been removed

*Errors*

The `matlab.fonts.editor.codefont.Size` setting has been removed. Use the `matlab.fonts.codefont.Size` setting instead. The `matlab.fonts.codefont.Size` setting controls both the App Designer **Code View** font size and the desktop code font size.

To update your code, change instances of the setting `matlab.fonts.editor.codefont.Size` to `matlab.fonts.codefont.Size`. For more information, see `matlab.fonts` Settings.

### **App Designer Smart Indent applies to individual lines**

#### *Behavior change*

When you apply Smart Indent to code in App Designer, the indentation change applies only to the current line. In R2021a and earlier releases, the Smart Indent option applied to the entire document.

To apply Smart Indent to the entire document, in **Code View**, first select all the code (for example, by pressing **Ctrl+A**). Then, apply Smart Indent by clicking the  button in the **Editor** tab, or pressing **Ctrl+I**.

### **CellSelectionCallback property of table UI components is not recommended in uifigure-based apps**

#### *Still runs*

Starting in R2021b, using the `CellSelectionCallback` property to program a response to table selection is not recommended for table UI components in App Designer apps and in figures created with the `uifigure` function. Use the `SelectionChangedFcn` property instead.

To update your code, assign all callback functions assigned to the `CellSelectionCallback` property to the `SelectionChangedFcn` property instead. If a callback function accesses the callback event data, you might need to update the event property names. For example, to access the indices of the elements the user selected, use the `Selection` property of the `TableSelectionChangedData` object. For more information, see `Table` Properties.



## Performance

### table Data Type Indexing: Improved performance when assigning elements by subscripting with curly braces

table subscripted assignment using curly braces is significantly faster in R2021b than in R2021a.

For example, when you assign into three table variables with  $10^6$  elements, performance in R2021b is approximately 4.4x faster, as shown below.

```
function timingTest()
    t = table(zeros(1e6,1), ones(1e6,1), nan(1e6,1));
    indices = randi(1e6,1,10000);

    tic;
    % Assign row vector of random values to randomly chosen row
    for i = indices
        t{i,:} = rand(1,3);
    end
    toc
end
```

The approximate execution times are:

**R2021a:** 7.4 s

**R2021b:** 1.7 s

The code was timed on a Windows 10 system with a 3.6 GHz Intel Xeon W-2133 CPU by calling the timingTest function in R2021a and R2021b.

### qrinsert and qrdelete Functions: Improved performance modifying QR factorizations

The qrinsert and qrdelete functions show improved performance inserting and deleting rows and columns in a QR factorization. The speedup is most noticeable for square matrices of order 1000 or less and is similar in magnitude for both rows and columns.

For example, this code uses a loop to insert and delete columns from the QR factorization of a random 200-by-200 matrix. qrinsert and qrdelete are about 12x faster than in the previous release.

```
function timingQRMod
X = rand(200);
[Q,R] = qr(X);
y = rand(200,1);
tic
for k = 1:1000
    [Qn,Rn] = qrinsert(Q,R,100,y);
end
toc
tic
for k = 1:1000
    [Qn,Rn] = qrdelete(Q,R,100);
end
```

```
toc
end
```

The approximate execution times are:

**R2021a:** 1.7 s (insertion) and 1.2 s (deletion)

**R2021b:** 0.15 s (insertion) and 0.10 s (deletion)

The code was timed on a Windows 10, Intel Xeon W-2133 CPU @ 3.60 GHz test system by calling the `timingQRMod` function.

### Titles and Labels in Plots: Improved performance when creating and querying titles or labels in a loop

Creating and querying the following types of titles and labels in a loop has improved performance.

- Plot titles, such as those created with the `title` or `subtitle` functions
- Axis labels, such as those created with the `xlabel`, `ylabel`, or `zlabel` functions

For example, this code creates 100 axes with titles in a tiled chart layout. It runs 11.9x faster than in the previous release:

```
function timingTitle
tiledlayout(10,10);
for n = 1:100
    nexttile
    title(n)
end
end
```

The approximate execution times are:

**R2021a:** 9.5 s

**R2021b:** 0.8 s

The code was timed on a Windows 10, Intel Xeon CPU E5-1650 v4 @ 3.60 GHz test system by calling the `timeit` function:

```
timeit(@timingTitle)
```

The performance gains increase with the number of axes, titles, and axis labels you are working with. For example, this table shows the improvements for looping over 10, 20, 50, and 100 axes with titles.

| Number of Axes with Titles | Performance Gain |
|----------------------------|------------------|
| 10                         | 2.5x             |
| 20                         | 4.2x             |
| 50                         | 7.0x             |
| 100                        | 11.9x            |

## Plot Interactions: Improved performance for rendering data tips and rotating scatter plots of large data sets

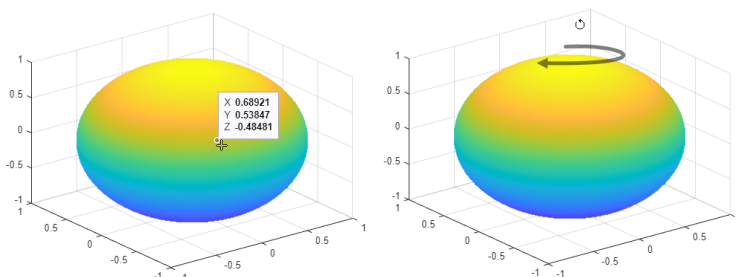
In figures created with the `uifigure` function and in MATLAB Online, interactions with scatter plots of large data sets have the following performance improvements:

- Data tip markers track the mouse motion more closely.
- 3-D scatter plots are more responsive to rotation gestures.

This improvement can be seen when the axes are created with either the `axes` or `uiaxes` function.

For example, on a Windows 10, Intel Xeon CPU E5-1650 v4 @ 3.60 GHz system, when you hover the cursor over the following sphere, the cursor changes to a crosshair more quickly, and the data tip markers track the cursor more closely. When you click and drag the cursor within the axes, the sphere rotates more quickly and tracks the cursor more closely.

```
f = uifigure;
ax = axes(f);
[X,Y,Z] = sphere(900);
scatter3(ax,X(:),Y(:),Z(:),[],Z(:),'r')
```



## Plots in Apps: Improved performance for creating plots

The performance is improved for creating plots in apps or in figures created with the `uifigure` function. For example, create a figure and an axes object. Then plot 10,000 points. This code runs 14x faster in R2021b.

```
function timingPlot
% Create figure and axes
f = uifigure;
ax = axes(f);
drawnow;

% Create data vector
y = rand(1,10000);

% Plot the data
tic;
plot(ax,y);
toc;
end
```

The approximate execution times are:

**R2021a:** 0.14 s

**R2021b:** 0.01 s

The code was timed on a Windows 10, Intel Xeon W-2133 CPU @ 3.60 GHz test system by calling the `timingPlot` function.

## App Designer: Improved performance when opening Start Page and loading apps

When you use App Designer, these operations have improved performance:

- Opening the App Designer Start Page
- Opening an existing app

For example, opening App Designer by entering `appdesigner` in the Command Window loads the Start Page approximately 1.8x faster in R2021b than in R2021a the first time it is opened, and 2.1x faster in subsequent times. The approximate startup times are:

| Release       | First Startup | Subsequent Startups |
|---------------|---------------|---------------------|
| <b>R2021a</b> | 6.6 s         | 3.6 s               |
| <b>R2021b</b> | 3.6 s         | 1.7 s               |

Also, loading an app in App Designer shows improved performance. For example, after creating and saving a new blank app, opening the app in App Designer is about 1.3x faster in R2021b than in R2021a. The approximate loading times are:

**R2021a:** 1.98 s

**R2021b:** 1.56 s

The performance improvement is larger if you have additional toolboxes installed.

These operations were timed on a Windows 10, Intel Core i7-5600 CPU @ 2.60 GHz test system.

## App Designer: Improved performance when saving apps

Saving apps in App Designer after you edit an app function or property is faster in R2021b than in R2021a. The more lines of code in the app file, the greater the performance improvement becomes.

For example, on a Windows 10, Intel Xeon W-2133 CPU @ 3.60 GHz test system, in an app containing 10,000 lines of code, if you click **Property**  to create a new property and then click **Save**  to save the app, you can run the updated app sooner in R2021b than in R2021a.

The approximate save times are:

**R2021a:** 20 s

**R2021b:** 1.5 s

## Comparison Tool: Improved performance when loading and saving MLAPP files

When you use the Comparison Tool to compare and merge changes between app code in MLAPP files, these operations have improved performance:

- Loading the files into the Comparison Tool
- Saving the files after merging changes

For example, if you load two apps with 5000 lines of code into the Comparison Tool by clicking

**Compare**  in the App Designer toolstrip, you can compare and merge the files sooner in R2021b than in R2021a.

The approximate loading times are:

**R2021a:** 13 s

**R2021b:** 8 s

Also, if you use the Comparison Tool to merge changes between two apps with 5000 lines of code (for example, by clicking **Merge Mode** , merging the changes, and then clicking **Save Result** ) , you can compare the saved files sooner in R2021b than in R2021a.

The approximate save times are:

**R2021a:** 24 s

**R2021b:** 8 s

Both of these operations were timed on a Windows 10, Intel Xeon W-2133 CPU @ 3.60 GHz test system.

## uigridlayout Function: Improved performance when adding components spanning multiple columns with 'fit' width

The performance of parenting components to a grid layout manager created using the `uigridlayout` function has improved when the components span multiple columns with a `ColumnWidth` value of `'fit'`. The performance improvement gets better as the number of components spanning multiple columns and the number of columns spanned increases.

For example, this code creates a grid layout manager with 10 columns with a `ColumnWidth` value of `'fit'`, and then creates 50 labels that span all 10 columns. Performance in R2021b is about 4.7x faster than in R2021a.

```
function timingGridLayout
f = uifigure;
numrows = 50;
numcols = 10;
g = uigridlayout(f);
g.Scrollable = 'on';
g.RowHeight = repmat({'fit'},1,numrows);
g.ColumnWidth = repmat({'fit'},1,numcols);
drawnow
```

```
tic
for row = 1:numrows
    txt = ['This is a label in row ' num2str(row) ' that spans ' ...
          num2str(numcols) ' columns in the grid.'];
    lbl = uilabel(g,'Text',txt);
    lbl.Layout.Column = [1 numcols];
end
drawnow
toc
end
```

The approximate execution times are:

**R2021a:** 5.2 s

**R2021b:** 1.1 s

The code was timed on a Windows 10, Intel Xeon CPU E5-1650 v4 @ 3.60 GHz test system by calling the function `timingGridLayout`.

## **uigridlayout Function: Improved resizing performance when wrapping text in resizable columns**

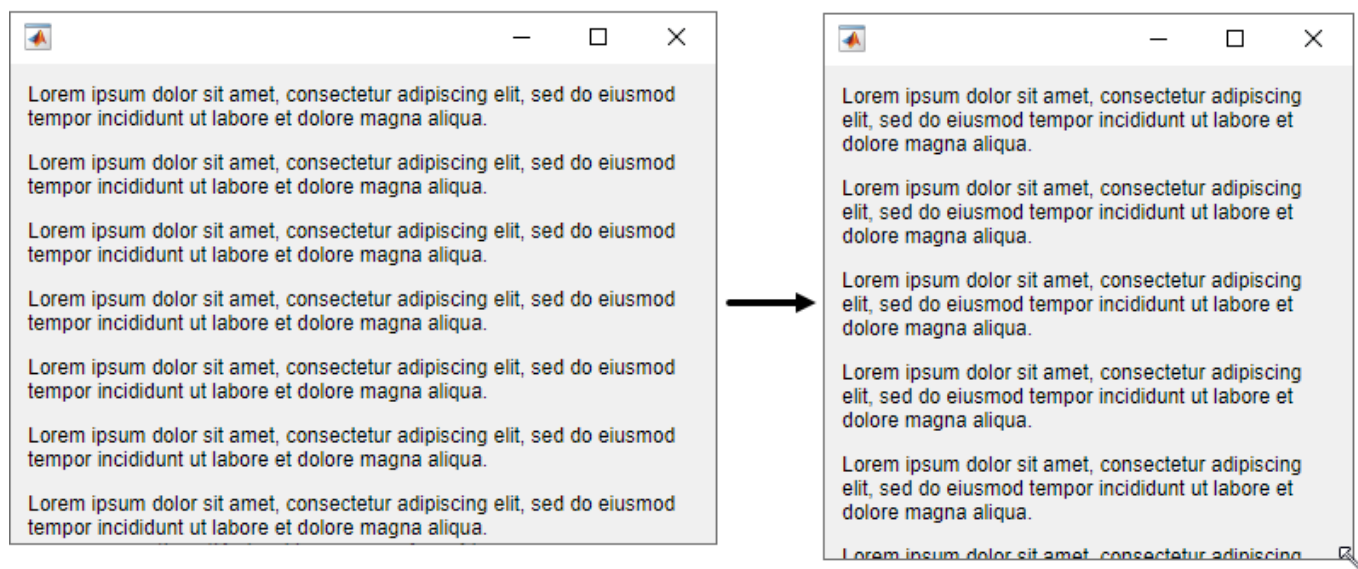
The performance when you resize apps containing a grid layout manager created using the `uigridlayout` function has improved when both of these conditions hold:

- The grid layout manager contains a component with a `WordWrap` value of `'on'`.
- The row and column containing the component with word wrap have a `RowHeight` of `'fit'` and a `ColumnWidth` that is resizable, such as `'1x'`.

For example, on a Windows 10, Intel Xeon CPU E5-1650 v4 @ 3.60 GHz test system, if you create 100 labels with wrapping text in a grid layout manager with fit height rows and resizable columns, and then resize the figure window by dragging the corner of the figure, the label text adjusts to fit the size of the figure almost immediately. In R2021a, there is a delay of about 2 seconds before the text adjusts.

```
f = uifigure;
g = uigridlayout(f);
g.ColumnWidth = {'1x'};
numrows = 100;
g.RowHeight = repmat({'fit'},numrows,1);

for row = 1:numrows
    c = uilabel(g);
    c.Text = ['Lorem ipsum dolor sit amet, consectetur adipiscing elit,' ...
             ' sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.'];
    c.WordWrap = 'on';
end
```



## Live Editor: Improved performance when saving live scripts or functions

Saving live scripts and live functions in the Live Editor is faster in R2021b than in R2021a. The improvement is most noticeable when you save live functions with more than 1000 lines of code and live scripts with fewer than 100 lines of code.

For example, on a Windows 10, Intel Xeon E5-1650 CPU @ 3.60 GHz test system, saving an example live function containing 4000 lines of code takes approximately 2.05 seconds in R2021b. In R2021a, saving the same live function takes approximately 2.57 seconds.

## Data Processing Dialog Boxes: Improved resizing performance

The Basic Fitting UI, **Data Statistics UI**, Colormap Editor, and Linked Plot Data Sources dialog now use `uigridlayout` to manage positions of UI components. This change results in a smoother experience when adjusting the size of these dialog boxes. For more information about the **Data Statistics UI**, see [Computing with Descriptive Statistics](#).

For example, on a Windows 10, Intel Xeon E5-1650 CPU @ 3.60 GHz test system, when you increase the size of the **Colormap Editor**, the size changes smoothly.

## Figure Interactions: Improved performance when using built-in axes interactions

Performance of figure interactions has been improved by coalescing built-in axes interactions so that there are significantly fewer interactions to process. These changes make interacting with a plot smoother and reduce the delay between an input and a response.

For example, on a Windows 10, Intel Xeon E5-1650 CPU @ 3.60 GHz test system, while `panInteraction` mode is active, when you click and drag the cursor within the axes the figure pans more quickly and tracks the cursor more closely.

## **UI Figures: Improved performance when displaying axes toolbar**

The performance of the axes toolbar in UI figures has been improved to reduce the delay before the toolbar appears.

For example, on a Windows 10, Intel Xeon E5-1650 CPU @ 3.60 GHz test system, when you pause the cursor on the axes, the axes toolbar appears more quickly.

## **UI Figures: Improved performance when interacting with linked axes**

Interacting with linked axes has improved performance when using figures created with the `uifigure` function or figures created in MATLAB Online™.



## Software Development Tools

### Projects: Collaborate using projects in MATLAB Online

Starting in R2021b, MATLAB Online provides support for basic projects workflows:

- Create an empty project and add files and folders.
- Clone a project from Git.
- Explore your project and run a dependency analysis.
- Create a project and manage your project files programmatically.

### Source Control: Work with files under Git in MATLAB Online

Starting in R2021b, MATLAB Online provides support for basic Git workflows:

- Cloning a remote Git repository
- Committing files to Git
- Pulling, pushing, and fetching files with Git

### Unit Testing Framework: Use the `TestCase` class template to create tests more quickly and accurately

You can now create a `TestCase` class, including basic test functionality, in MATLAB and MATLAB Online. To create a new test class, select **New > Test Class** on the **Home**, **Editor**, or **Live Editor** tabs.

Use the `TestCase` class template to create tests more conveniently. The template includes a `TestClassSetup` methods block, a `TestMethodSetup` methods block, and a `Test` methods block that defines a simple `Test` method. To customize your test class, add code to the file or remove unused code that is included by default. For more information about class-based tests, see [Author Class-Based Unit Tests in MATLAB](#).

### Unit Testing Framework: Run live-function-based tests interactively in MATLAB Online

Starting in R2021b, you can run live-function-based tests interactively in MATLAB Online. When you open an MLX file defining a function-based test in MATLAB Online, the toolstrip lets you run all tests in the file or just the current test.

To run tests and customize your test run interactively, use the **Run Tests** section in the **Live Editor** tab of the toolstrip. For more information, see [Run Tests in Editor](#).

### App Testing Framework: Perform press gestures on axes and UI axes with different selection types

The app testing framework now supports mouse selection types in `press` gestures that are performed on axes and UI axes. For example, create an axes with a plot and then test a double-click gesture at the point (3, 2).

```
f = uifigure;  
ax = axes(f);  
plot(ax,1:10)  
tc = matlab.uitest.TestCase.forInteractiveUse;  
tc.press(ax,[3 2], 'SelectionType', 'open')
```

## App Testing Framework: Perform drag gestures on axes and figures with different selection types

Starting in R2021b, the app testing framework supports drag gestures on UI figures. Additionally, when you test a drag gesture on an axes, UI axes, or UI figure, you can specify the mouse selection type. For example, create a figure and drag on it from the point (100, 200) to the point (200, 300) using a right-click gesture.

```
f = uifigure;  
tc = matlab.uitest.TestCase.forInteractiveUse;  
tc.drag(f,[100 200],[200 300], 'SelectionType', 'alt')
```

## App Testing Framework: Use any units of measurement in gestures at the center of components

Starting in R2021b, when you perform a gesture at the center of a component, the component or its parent containers can use any units of measurement. In previous releases, the framework does not support containers that use nonpixel units.

For example, create a figure and set its `Units` property to `'normalized'`. Then, create a panel in the figure and press at the center of the panel.

```
f = uifigure;  
f.Units = 'normalized';  
p = uipanel(f);  
tc = matlab.uitest.TestCase.forInteractiveUse;  
tc.press(p)
```

If you perform a gesture at the center of a component using a syntax that accepts location as an input (for instance, `press(testcase, comp, location)`), then the figure or parent containers can use only `'pixels'` as their units of measurement.

## ⚠️ Functionality being removed or changed

### Specifying diagnostic after name-value arguments in qualifications is not recommended *Still runs*

Starting in R2021b, when you test for equality using the `verifyEqual`, `assumeEqual`, `assertEqual`, or `fatalAssertEqual` methods, specifying name-value arguments before the diagnostic input argument is not recommended. Place name-value arguments after all of the positional arguments instead. Although not recommended, you still can specify diagnostic after the name-value arguments when these arguments use the `name, value` syntax.

The reason for this change is that, starting in R2021a, MATLAB supports a new syntax for passing name-value arguments. In the new syntax, the name and value arguments are connected by an equals sign (`name=value`), and the name is not enclosed in quotes. To use the new syntax with qualification methods, specify positional arguments, including diagnostic, before the `name=value` arguments. If you specify diagnostic after `name=value` arguments, MATLAB produces an error.

This table shows an example of how you can update your code.

| R2021a and Earlier                                                                                                                                                                                                                                                    | Starting in R2021b                                                                                                                                                                                                                          |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <pre>testCase = matlab.unittest.TestCase.forInteractiveUse; verifyEqual(testCase,1.5,2, ...     "RelTol",0.1,"Difference must be within relative tolerance.",R assumeEqual(testCase,1,2, ...     "AbsTol",0.5,"Difference must be within absolute tolerance.",A</pre> | <pre>testCase = matlab.unittest.TestCase.forInteractiveUse; verifyEqual(testCase,1.5,2, ...     "Difference must be within relative tolerance.",R assumeEqual(testCase,1,2, ...     "Difference must be within absolute tolerance.",A</pre> |

For more information, see `verifyEqual`.

### **matlab.unittest.TestSuite.fromProject ignores the files that do not define test procedures when creating a test suite**

*Behavior change*

Starting in R2021b, if your project includes files with the `Test` classification, `matlab.unittest.TestSuite.fromProject` ignores the files that do not define test procedures when you create a test suite. For example, if an abstract `TestCase` class definition file is labeled with the `Test` classification, `fromProject` ignores it. In previous releases, MATLAB produces an error if `fromProject` is called on a project that uses the `Test` classification for any files other than concrete test files. With this change, `fromProject` becomes consistent with the `matlab.unittest.TestSuite.fromFolder` method: both methods create a test suite from all the concrete test files and ignore any other files in the folder.

This behavior change also applies to the `testsuite`, `runtests`, and `runperf` functions when they operate on code organized into files and folders within a project.

### **Test suites created from projects cannot run without the Java Virtual Machine (JVM) software**

*Behavior change*

Starting in R2021b, if you start MATLAB without the Java Virtual Machine (JVM®) software and create a suite from the test files in a project using `testsuite`, the function uses the `matlab.unittest.TestSuite.fromProject` method to create the suite. If you then try to run the test suite without the JVM software, MATLAB produces an error because the project cannot be opened without the JVM software. In previous releases, when MATLAB runs without the JVM software, `testsuite` uses `matlab.unittest.TestSuite.fromFolder` to create a suite from the test files in the project, and the testing framework runs the resulting test suite.

This behavior change also applies to the `runtests` and `runperf` functions when they operate on code organized into files and folders within a project.

## External Language Interfaces

### C++ interface: Support for C++ language features

The C++ interface supports these additional C++ language features.

#### Support for `void**` parameters

MATLAB returns a `void*` argument for `void**` parameters. For more information, see `void**` Input Argument Types. For information about memory management of `void**` parameters, see `Pass Ownership of Memory to MATLAB`.

#### `char []` parameters behave like `char *` parameters

MATLAB supports `char []` parameters as either integer or character (string), the same as `char*` parameters. Likewise, the Unicode types `wchar_t[]`, `char16_t[]`, and `char32_t[]` behave like `wchar_t*`, `char16_t*`, and `char32_t*`. For more information, see C++ `char*` and `char []` Types.

#### Support for static data members

Public static and public const static data members are treated as read-only properties in MATLAB. You cannot modify the value of a C++ static data member in MATLAB. For more information, see `Static Data Members`.

You can use a public static data member (property) as the data type of an input argument or return type in a class constructor, method, or function. You also can use a static property or method to define the shape of an argument. For information about using static properties to define the shape, see `Use Property or Method as SHAPE`.

### C++ interface: Publisher options

The C++ interface supports these build configuration features.

#### Overwrite existing library definition files

Publishers can automatically overwrite existing library definition MLX files when calling `clibgen.generateLibraryDefinition`. Set the `OverwriteExistingDefinitionFiles` name-value argument to `true`. This option is useful when you create and modify the definition file for a library *libname*.

When you use this option, MATLAB deletes `definelibname.mlx` and `definelibname.m`, including any edits you made to the files.

#### Options for defining arguments

- By default, when a MATLAB input has fewer dimensions than the corresponding C++ argument, then MATLAB inserts singleton dimensions at the beginning of the `Shape` argument. For more information, see `Dimension Matching`. To insert singleton dimensions at the end, set the `'AddTrailingSingletons'` name-value argument to `true` in the `defineArgument` functions - `defineArgument (ConstructorDefinition)`, `defineArgument (FunctionDefinition)`, and `defineArgument (MethodDefinition)`.
- If a C++ function has a character array parameter used to return a C++ string, you can define the argument so that the function returns a null-terminated string. Use the

'NumElementsInBuffer' name-value argument in the `defineArgument` (`FunctionDefinition`) or `defineArgument` (`MethodDefinition`) functions. For an example, see the `getMessage` function in the Define String Argument table.

- You can transfer ownership of the memory of function or method parameters of double pointer type with scalar output to MATLAB using the `DeleteFcn` name-value argument. For more information, see `defineArgument` (`FunctionDefinition`) and `defineArgument` (`MethodDefinition`).
- MATLAB lets you control the lifetime management of objects created with a constructor by specifying the `ReleaseOnCall` name-value argument in the library definition file. For more information, see `defineArgument` (`ConstructorDefinition`) and Lifetime Management of C++ Objects in MATLAB.
- You can use a method or a property to define the shape of an argument. For more information, see Use Property or Method as `SHAPE`.
- You can transfer ownership to MATLAB of the memory of double pointer input arguments `void**` and `object**`. Use the 'DeleteFcn' name-value argument for `defineArgument`, as described in Pass Ownership of Memory to MATLAB.

## Java interface: Specify JRE path for MATLAB

You can run MATLAB with your system version of the Java Runtime Environment (JRE). For information about Java versions compatible with MATLAB, see MATLAB Interfaces to Other Languages.

To set the JRE path in MATLAB, call `jenv`. You must restart MATLAB to use the updated path. This command sets the path for all future MATLAB sessions but does not change the path for other applications on your computer.

Alternatively, you can set the path from the operating system prompt. Call `matlab_jenv`, then start MATLAB.

## Java: Call into MATLAB from a Java program called by MATLAB

Java developers can use the `com.mathworks.engine.MatlabEngine` API `getCurrentMatlab` method to call back into MATLAB from Java. Incorporating this method in your application allows MATLAB users to call functionality from your Java program.

For information about developing these Java programs, see Call Back into MATLAB from Java.

## Python interface: Run Python commands and scripts from MATLAB

The `pyrun` and `pyrunfile` functions let you call Python commands and scripts from MATLAB. For more information, see Directly Call Python Functionality from MATLAB

## Python: Support for complex multidimensional arrays

MATLAB supports passing complex multidimensional array data to Python and from Python to MATLAB, for both in-process and out-of-process execution modes. For example, create a file `test.py` containing this code:

```
def returnData(data):  
    return data
```

To pass a complex MATLAB array to `returnData`, type:

```
mc = complex(magic(3));  
c = py.test.returnData(mc)
```

```
c =  
Python memoryview:
```

```
8.0000 + 0.0000i  1.0000 + 0.0000i  6.0000 + 0.0000i  
3.0000 + 0.0000i  5.0000 + 0.0000i  7.0000 + 0.0000i  
4.0000 + 0.0000i  9.0000 + 0.0000i  2.0000 + 0.0000i
```

Use `details` function to view the properties of the Python object.

Use `double` function to convert to a MATLAB array.

To convert the return value to a MATLAB array, type:

```
C = double(c)
```

```
C = 3x3 complex  
8.0000 + 0.0000i  1.0000 + 0.0000i  6.0000 + 0.0000i  
3.0000 + 0.0000i  5.0000 + 0.0000i  7.0000 + 0.0000i  
4.0000 + 0.0000i  9.0000 + 0.0000i  2.0000 + 0.0000i
```

For information about MATLAB to Python data type mapping, see [Pass Matrices and Multidimensional Arrays to Python](#).

## Python: Version 3.9 support

MATLAB now supports CPython 3.9, in addition to existing support for 2.7, 3.7, and 3.8. For more information, see [Versions of Python Compatible with MATLAB Products by Release](#)

## ⚠ WSDL Web Services Documents: Apache CXF version 3.4.2 support

MATLAB supports Apache CXF version 3.4.2 for use with WSDL Web services. For more information, see [Set Up WSDL Tools](#).

## ⚠ Compatibility Considerations

Download the latest version 3.4.2 release of the Apache CXF tool from <https://cxf.apache.org/> download.

## ⚠ Perl 5.32.1: MATLAB support on Windows

As of R2021b, MATLAB on Windows ships with an updated version of Perl, version 5.32.1. See <https://www.perl.org> for a standard distribution of Perl, Perl source code, and information about using Perl.

⚠Compatibility Considerations

If you use the `perl` command on Windows platforms, see <https://www.perl.org> for information about using this version of the Perl programming language.

⚠Functionality being removed or changed

**name=value syntax errors for calls to Python functions using `py.` prefix**

*Behavior change*

Starting in R2021b, MATLAB errors when you use `name=value` syntax for passing keyword arguments to Python functions using the `py.` prefix. In R2021a, MATLAB might silently give the wrong answer. Use `pyargs` to pass keyword arguments.

For example, the Python `print` function has a keyword argument `sep`. This Python statement sets the `sep` argument to a comma followed by a space:

```
print('comma','separated','values',sep=', ')
```

When you call this statement in MATLAB, MATLAB interprets `sep=', '` as a `name=value` argument:

```
py.print('comma','separated','values',sep=', ')
```

| R2021a Behavior                                                                 | R2021b Behavior                                                                                                         | How to Update Your Code                                                                 |
|---------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------|
| <pre>py.print(...<br/>    'comma','separated','values',<br/>    sep=', ')</pre> | <pre>py.print(...<br/>    'comma','separated','values',<br/>    sep=', ')</pre>                                         | <pre>py.print(...<br/>    'comma','separated','values',<br/>    pyargs(sep=', '))</pre> |
| Silent wrong answer:<br><br><pre>comma separated values sep ,</pre>             | Error:<br><br>Error using py.print<br>Using name=value format is not supported.<br>Use pyargs to pass keyword arguments | <pre>comma, separated, values</pre>                                                     |

**`createSoapMessage`, `callSoapService`, and `parseSoapResponse` have been removed**

*Errors*

Consider using `matlab.wsdL.createWSDLCClient` instead of the `createSoapMessage`, `callSoapService`, and `parseSoapResponse` functions to communicate with Web services using Simple Object Access Protocol (SOAP). There is no direct function replacement for the SOAP functions, but when you create a WSDL interface, you have access to the Web service functionality.

**`createClassFromWsdL` has been removed**

*Errors*

The `matlab.wsdL.createWSDLCClient` function replaces the `createClassFromWsdL` function to communicate with Web services from MATLAB using Web Services Description Language (WSDL). `matlab.wsdL.createWSDLCClient` enables you to specify additional information needed to access the WSDL document. For more information, see `weboptions`.

To get started using `matlab.wsdL.createWSDLCClient`, follow these steps.

- 1 Download supported versions of the Java JDK and Apache CXF programs. For more information, see Set Up WSDL Tools.
- 2 Set the paths to these programs, where `jdk` is the path to the JDK installation and `cx` is the path to the CXF program.

```
matlab.wsdL.setWSDLToolPath('JDK',jdk,'CXF',cx)
```

To update your code, replace calls to `createClassFromWsdL` with calls to `matlab.wsdL.createWSDLClient`. For example, for a Web service with this URL:

```
url = 'https://examplesite.com/samplewebservice';
```

replace this call to `createClassFromWsdL`:

```
createClassFromWsdL(strcat(url,'?WSDL'))
```

with:

```
matlab.wsdL.createWSDLClient(url)
```

---

**Note** `matlab.wsdL.createWSDLClient` does not support RPC-encoded WSDL documents.

---



## Hardware Support

### Connect and Control Arduino board using the Arduino Explorer App

The MATLAB Support Package for Arduino Hardware now has an Arduino Explorer app.

Using this app, you can:

- Set up the Arduino board
- Connect to an Arduino board over USB, Bluetooth®, and WiFi
- Configure, read from, and write to Arduino pins
- Visualize data from Arduino pins
- Record and save data from Arduino pins to the MATLAB workspace
- Analyze the recorded data
- Generate equivalent MATLAB code

### Read data from APDS9960 sensor connected to the Arduino hardware

The MATLAB Support Package for Arduino Hardware enables you to read gesture, proximity, clear light and color (RGB) data from APDS9960 sensor connected to Arduino hardware.

### Support for CAN shields on Raspberry Pi Hardware

Use the MATLAB Support Package for Raspberry Pi Hardware to read and write CAN messages from the CAN network on the Raspberry Pi hardware.



# R2021a

---

**Version: 9.10**

**New Features**

**Bug Fixes**

**Compatibility Considerations**

## Environment

### Live Editor Controls: Create dynamic controls in live scripts by linking variables to drop-down items and slider values

When adding a drop-down list to a live script, you can populate the items in it using values stored in a variable. When adding a slider, you can specify the minimum, maximum, and step values using variables.

For example, create a live script and define the variable `lastnames` that contains a list of last names.

```
lastnames = ["Houston","Vega","Obrien","Potter","Rivera","Hanson","Fowler","Tran","Briggs"];
```

Run the live script to create `lastnames` and add it to the workspace. Then, go to the **Live Editor** tab, and in the **Code** section, select **Control > Drop Down**. In the **Items** section of the control configuration menu, select `lastnames` as the **Variable**.

The screenshot shows the configuration menu for a 'Drop down' control. It is organized into several sections: 'LABEL' with a 'Label' field set to 'Drop down'; 'ITEMS' with two lists: 'Item labels' (Houston, Vega, Obrien, Potter) and 'Item values' (lastnames(1), lastnames(2), lastnames(3), lastnames(4)); a section titled 'Select a variable to add its content to drop down' with a 'Variable' dropdown set to 'lastnames'; 'DEFAULTS' with a 'Default value' dropdown set to 'Houston'; and 'EXECUTION' with a 'Run' dropdown set to 'Current section'.

Close the configuration menu to return to the live script. The drop-down list now contains the last names defined in `lastnames`.

The screenshot shows a drop-down list with 'Houston' selected. The list is open, showing all the names from the 'lastnames' variable: Houston, Vega, Obrien, Potter, Rivera, Hanson, Fowler, Tran, and Briggs.

If you add, remove, or edit the values in `lastnames`, the items in the drop-down list update accordingly.

For more information, see [Add Interactive Controls to a Live Script](#).

## Live Editor Fonts: Change the name, style, size, and color of fonts programmatically using settings

You can change the name, style, size, and color of titles, headings, text, and code in the Live Editor using settings.

For example, this code changes the color and style of titles in the Live Editor:

```
s = settings;
s.matlab.fonts.editor.title.Style.PersonalValue = {'bold'};
s.matlab.fonts.editor.title.Color.PersonalValue = [0 0 255 1];
```

This code increases the size and changes the font of normal text in the Live Editor:

```
s = settings;
s.matlab.fonts.editor.normal.Size.PersonalValue = 20;
s.matlab.fonts.editor.normal.Name.PersonalValue = 'Calibri';
```

The Live Editor updates all open live scripts and live functions to show the selected fonts. When you create new live scripts or functions, they use the new fonts as well.

### Plot Random Data

This script plots a vector of random data and draws a horizontal line on the plot at the mean.

```
n = 50;
r = rand(n,1);
plot(r)

m = mean(r);
hold on
plot([0,n],[m,m])
hold off
title('Mean of Random Uniform Data')
```

For more information, see `matlab.fonts`.

## Live Editor Display: Specify where to display output by default

You can change the default location for output in a new live script. Depending on this preference, new live scripts that you create display their output inline or on the right. To change the default, go to the **Home** tab, and in the **Environment** section, select  **Preferences**. Select **MATLAB > Editor / Debugger > Display**, and then select an option for the **Live Editor default view**:

- **Output on right** — Output displays to the right of the code. Each output displays next to the line that creates it. This option is ideal when writing code.

- **Output inline** — Output displays inline with the code. Each output displays underneath the line that creates it. This option is ideal for sharing.

## Live Editor Functions: Run live functions interactively using the Run button in MATLAB Online

In MATLAB Online, you can now run live functions interactively using the  **Run** button.

To run live functions that require input argument values or any other additional setup, configure the  **Run** button by clicking **Run** and adding one or more commands. For more information about configuring the  **Run** button, see [Configure the Run Button for Functions](#).

## Live Editor Bookmarks: Navigate quickly between lines

To navigate quickly between lines, set bookmarks in your live scripts or live functions. Bookmarks are particularly useful in long files when you frequently need to move between sections.

To set a bookmark, put the cursor on the line that you want to set it on. Then, go to the **Live Editor** tab and, in the **Navigate** section, click  **Bookmark**. A bookmark icon  appears to the left of the line. To clear a bookmark, with the cursor on the line with the bookmark, click **Bookmark** and select **Set/Clear**. You also can clear the bookmark by clicking the bookmark icon  to the left of the line.

To navigate to a bookmark, go to the **Live Editor** tab, and in the **Navigate** section, click **Bookmark** and select **Previous** or **Next**.

For more information about navigating within files, see [Go To Location in File](#).

## Live Editor Animation Playback Controls: Interactive interface to control animations

Playback controls now appear within the figure window after an animation is done playing. These playback controls provide the ability to replay the animation and explore individual frames without having to re-run the entire live script. Animation playback controls are not supported for animations generated by the `movie` function.

## Live Editor Performance: Improved performance when saving large live scripts or functions

When saving large live scripts or functions, you can continue using the Live Editor sooner in R2021a than in R2020b. While you continue to use the Live Editor, MATLAB saves the file in the background. When MATLAB finishes saving the file, the asterisk (\*) next to the file name disappears, indicating that the file is saved.

For example, on a Windows 10, Intel Xeon E5-1650 CPU @ 3.60 GHz test system, if you save a live script containing 35,000 lines of code and then click the title bar of another document open in the Live Editor, MATLAB immediately switches to the other open document. In R2020b, there is a noticeable delay before MATLAB switches to the other open document.

## Help Browser: View web documentation by default

Starting in R2021a, when you run MATLAB with an internet connection, the Help browser displays the web documentation by default. When you run MATLAB on a system without an internet connection, or if your internet connection becomes unavailable, the Help browser displays the installed documentation instead.

To change the default documentation location, on the **Home** tab, in the **Environment** section, click  **Preferences**. Select **MATLAB > Help** and change the **Documentation Location**. For more information, see Help Preferences.

## Documentation: View MATLAB documentation in French, Italian, and German

A subset of MATLAB documentation in French, Italian, and German is available on the web to licensed MATLAB users. Only a subset of the full documentation is available. For more information, see Translated Documentation.

## MATLAB Drive: Get the location of your MATLAB Drive root folder programmatically

You can get the location of your MATLAB Drive root folder programmatically, using the `matlabdrive` command, from your desktop or in other MATLAB environments such as MATLAB Online. For example, if you have MATLAB Drive Connector installed on your desktop system, MATLAB returns the location of your MATLAB Drive:

```
md = matlabdrive  
  
md =  
    'C:\Users\username\MATLAB Drive'
```

## MATLAB Drive: Pause and resume syncing in MATLAB Drive Connector

You can temporarily stop the syncing of your MATLAB Drive files, for example, if you are on a metered or slow internet, by pausing and then resuming syncing.

For more information, see “Use MATLAB Connector to Manage Your Files”.

## MATLAB Drive: MATLAB Drive Connector now available in Chinese and Korean

MATLAB Drive Connector now available in Chinese and Korean.

## MATLAB Drive: Change the folder permissions for an invited member of a shared folder in MATLAB Drive Online (May 2021)

After inviting someone to access a shared folder, you can change the folder permissions for that person. For example, if you invite someone to share a folder as a read-only folder, you can change the permissions for that person to allow them to edit the contents of the folder.

For more information, see “Share Folders Using MATLAB Drive”.

## **MATLAB Drive: Share folder by invitation to others who already have access to the folder through a view-only link (May 2021)**

You can now invite others to access a shared folder even if they already have access to the folder through a view-only link. Inviting someone to access a shared folder allows you to give them edit permissions to the folder.

### **⚠️ Functionality being removed or changed**

#### **PNG images in documentation are compressed**

*Behavior change*

Starting in R2021a, all PNG images included in the documentation are compressed, reducing their stored size on disk. The compressed images should be visually identical to the original images.



## Language and Programming

### Name=Value Syntax: Use name=value syntax for passing name-value arguments

MATLAB supports a new syntax for passing name-value arguments. In the new syntax, the name and value arguments are connected by an equal sign, and the name is not enclosed in quotes.

**Name=value syntax:** `plot(x,y,LineWidth=2)`

**Comma-separated syntax:** `plot(x,y,"LineWidth",2)`

MATLAB continues to support the comma-separated name,value syntax. Existing functions and methods support both syntaxes, and the process for writing functions and methods with name-value arguments is unchanged.

Use the new syntax to help identify name-value arguments for functions and to distinguish names from values in lists of name-value arguments. There are some limitations on where and how the name=value syntax can be used:

- The recommended practice is to use only one syntax in any given function call. However, if you do mix name=value and name,value syntaxes in a single call, all name=value arguments must appear after the name,value arguments. For example, `plot(x,y,"Color","red",LineWidth=2)` is a valid combination, but `plot(x,y,Color="red","LineWidth",2)` errors.
- Similarly, name=value arguments must appear after all positional arguments. Calling `myFunction(name=value,posArgument)` errors.
- The name=value syntax can only be used directly in function calls. They cannot be wrapped in a cell array or additional parentheses. Calling `myFunction(a,(name=value))` errors.

### Retrieving Display Format: format function can get and set display format

The `format` function can now output the current Command Window display format. Calling `format` with an output variable returns a `DisplayFormatOptions` object that describes the current numeric and line spacing formats:

```
fmt = format
```

```
fmt =  
    DisplayFormatOptions with properties:
```

```
    NumericFormat: "shortE"  
    LineSpacing: "loose"
```

You can also use a `DisplayFormatOptions` object as an input to `format` to change the display format.

## Capturing disp Output: Use the `formattedDisplayText` function to store disp output as a string

`formattedDisplayText` captures the output of `disp(obj)` as a string. The function also enables you to control the formatting of the captured string, including numeric format and line spacing. For example, `A` is a vector containing three logical values. `formattedDisplayText` displays the vector elements as the words “true” or “false”:

```
out = formattedDisplayText(A, "UseTrueFalseForLogical", true)
```

```
out =  
    '    true    false    true'  
    ,
```

## Virtual File Storage: `mkdir` and `rmdir` will now be able to create and remove files from VFS directories

Starting in R2021a, `rmdir` and `mkdir` are able to create folders in remote locations.

## Function Argument Validation: Debugger and profiler is now supported

The MATLAB debugger will now be usable inside of the arguments blocks of functions. While debugging an arguments block the workspace is read only. The MATLAB profiler will now record lines inside of arguments blocks.

## Class Diagram Viewer: Create graphical class diagrams to explore class details and share designs

Use the Class Diagram Viewer tool to create graphical class diagrams, including details such as:

- Inheritance relationships, including abstract classes, mixins, and multiple inheritances
- Lists of properties, methods, and events
- Class member access levels

You can use the diagrams to explore complex class designs. You can also use the diagrams to share proposed designs with team members, either as static images or editable diagrams.

Use the associated `matlab.diagram.ClassViewer` class for command line access to diagrams.

## Enumeration Comparisons: Use `isequal` to compare enumeration members with text data types

Enumeration classes now have a default implementation of the `isequal` method. You can use `isequal` to compare enumeration members with character vectors, strings, and cell arrays of character vectors or strings. For more information, see `isequal` Method.

## ▲eval function: Context checking to resolve identifiers

MATLAB now resolves identifiers like variable names in an `eval` statement using additional context. For instance, MATLAB recognizes when a call to `eval` uses a variable declared in a function.

## ▲Compatibility Considerations

With the additional context, MATLAB resolves ambiguities differently than in previous releases. Some code now produces errors or warnings.

Code that warns starting in R2021a will error in a future release.

| Example                                                                                                                                                                                                                     | Previous Result                | R2021 Result                         | Notes                                                                                                                                                                                                                                                                                                  |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------|--------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <pre>function myfun(pi)     eval('pi') end  &gt;&gt; myfun</pre>                                                                                                                                                            | <pre>ans =  3.1416</pre>       | Errors                               | MATLAB resolves <code>pi</code> as a variable in the function workspace, so <code>myfun</code> now errors when <code>pi</code> is not defined.                                                                                                                                                         |
| <pre>function myfun     disp(pi)     eval('pi = 1'); end  &gt;&gt; myfun</pre>                                                                                                                                              | <pre>3.1416  pi =  1</pre>     | Same output but warns on assignment. | <p>MATLAB resolves <code>pi</code> as a variable in the function workspace.</p> <p>In a future release, <code>myfun</code> errors. This new behavior is consistent with processing the following code:</p> <pre>function myfun     disp(pi)     pi = 1; end</pre>                                      |
| <pre>% assignLocal.m script local = 1;  % myfun.m file with local function: function myfun     local()     assignLocal     eval("local") end function local     disp("local fx") end  % function call: &gt;&gt; myfun</pre> | <pre>local fx  local =  </pre> | Same output but warns on assignment. | <p>In a future release, MATLAB will give precedence as described in “Function Precedence Order”. In this example, precedence goes to the local function. This new behavior is consistent with processing the following code:</p> <pre>function myfun     local()     assignLocal     local() end</pre> |

## ⚠ Functionality being removed or changed

### **format with no arguments is not recommended**

*Still runs*

The `format` command, by itself, resets the output display format to the default, which is the short, fixed-decimal format for floating-point notation and loose line spacing for all output lines.

```
format
```

For clearer code, explicitly specify the `default` style.

```
format default
```

### **Defining classes and packages: Using `schema.m` will not be supported in a future release**

*Still runs*

Support for classes and packages defined using `schema.m` files will be removed in a future release. Replace existing schema-based classes with classes defined using the `classdef` keyword.

### **compose does not accept an invalid hexadecimal value, octal value, or trailing backslash**

*Errors*

Previously, when the `formatSpec` input to `compose` contained an invalid hexadecimal value, octal value, or trailing backslash it would issue a warning and truncate the output at the point of the invalid value. Starting in R2021a, MATLAB will issue an error instead. With this change, `compose` will error for all invalid `formatSpec` inputs.

### **Using `get` and `set` to access or change display format is not recommended**

*Still runs*

Using `get` and `set` to programmatically access or change the numeric display format and the display line spacing is not recommended. Use settings instead. For example:

```
s = settings;  
myformat = s.matlab.commandwindow.NumericFormat.ActiveValue
```

```
myformat =  
    'short'
```

```
s.matlab.commandwindow.DisplayLineSpacing.TemporaryValue = 'compact';  
myspacing = s.matlab.commandwindow.DisplayLineSpacing.ActiveValue
```

```
myspacing =  
    'compact'
```

For more information, see `matlab.commandwindow` Settings.

### **Using `feature('EightyColumns')` to access and change Command Window display width is not recommended**

*Still runs*

Using the command `feature('EightyColumns')` or `feature('EightyColumns',value)` to programmatically determine or change whether the Command Window display width limit is enabled is not recommended. Use settings instead. For example:

```
s = settings;  
s.matlab.commandwindow.UseEightyColumnDisplayWidth.TemporaryValue = 1;  
limitwidth = s.matlab.commandwindow.UseEightyColumnDisplayWidth.ActiveValue  
  
limitwidth =  
    logical  
         0
```

For more information, see `matlab.commandwindow` Settings.

## Data Analysis

### Data Preprocessing Live Editor Tasks: Operate on multiple table variables and specify output format for table input

When you are working with data in a table or timetable, these Live Editor tasks now allow you to operate on multiple table variables at the same time:

- Clean Missing Data
- Clean Outlier Data
- Smooth Data
- Remove Trends
- Find Local Extrema
- Find Change Points

You can also choose which variable to display when visualizing the results.

In addition, tasks that modify variables provide new output options. You can return a table with all of the variables, or with only the variables that were modified. For tasks that return logical arrays, you can specify the size of the output. The size can match the size of the input table or a table containing only the variables that were used in the calculation.

### Clean Outlier Data Live Editor Task: Visualize results with a histogram

The Clean Outlier Data Live Editor task now offers histogram plots for most detection methods. The histogram can summarize the input data, the outliers, the cleaned data with the outliers filled, and the outlier detection thresholds and center value.

### fillmissing Function: Specify custom fill method

You can now specify a custom method for filling missing values when using the `fillmissing` function. Specify the custom method as a function handle.

### normalize Function: Normalize multiple data sets with same parameters

`normalize` can now return the centering and scaling parameter values used to perform the normalization. You can reuse these parameters to normalize subsequent data sets in the same manner. For example, you can normalize an array of data `A` and then normalize a second array `B` with the same parameters:

```
[Anorm,C,S] = normalize(A);  
Bnorm = normalize(B, 'center',C, 'scale',S);
```

The new outputs `C` and `S` contain the centering and scaling parameter values, respectively. So that you can easily reuse them in a later normalization step, you can also now specify the `'center'` and `'scale'` normalization methods at the same time. These are the only two normalization methods that you can specify together.

To further support these changes, when method is 'center' or 'scale', the possible values of `methodtype` now include arrays and tables. While these `methodtype` values are intended to work with the new outputs C and S, you also can compute your own normalization parameters to specify.

## ▲ **groupcounts Function: Display percentages of group counts**

`groupcounts` now displays information about the percentage each group count represents.

- For table and timetable inputs, `groupcounts` automatically displays the percentages represented by each group count in the output table.
- For array inputs, `groupcounts` has a new third output argument to return the percentages represented by each group count.

## ▲ **Compatibility Considerations**

When `groupcounts` operates on data in a table or timetable, the output contains an additional table variable for the percentages. The percentages are in the range [0 100] and are included in the table variable `Percent`.

Any code that references specific table variables is unaffected. However, you might need to update code that depends on the number of variables in the output table.

## **ts2timetable Function: Convert timeseries objects to timetables**

To convert `timeseries` objects to timetables, use the `ts2timetable` function.

## **table and timetable Functions: Specify dimension names using the 'DimensionNames' name-value argument**

When you create tables and timetables, you can specify their dimension names by using the 'DimensionNames' name-value argument with these functions:

- `array2table`
- `array2timetable`
- `cell2table`
- `struct2table`
- `table`
- `timetable`

In previous releases, you could change the dimension names only by creating a table or timetable and then changing its `DimensionNames` property.

## ▲ **Functionality being removed or changed**

### **Table dimension names cannot match reserved names**

*Behavior change*

MATLAB raises an error if you assign a dimension name that matches one of these reserved names: 'Properties', 'RowNames', 'VariableNames', or ':'. In previous releases, MATLAB raised a warning and modified the dimension names so that they were different from the reserved names.

For example, if you create a table and then assign 'Properties' as a dimension name, the result is an error.

```
T = array2table(magic(3));  
T.Properties.DimensionNames = {'Rows', 'Properties'}
```

### **'SamplingRate' will be removed**

#### *Warns*

The 'SamplingRate' name-value argument will be removed in a future release. Use 'SampleRate' instead. The corresponding timetable property is also named SampleRate.

For backward compatibility, you still can specify 'SamplingRate' as the name of the name-value argument. However, the value is assigned to the SampleRate property.

This change in behavior affects the timetable functions shown in the list:

- array2timetable
- retime
- synchronize
- table2timetable
- timetable



## Data Import and Export

### XML Files: Read, write, and import XML files using `readtable`, `readtimetable`, and other functions

The `readtable`, `writetable`, `readtimetable`, `writetimetable`, and `detectImportOptions` functions now support reading and writing XML files. This list outlines the added capabilities of each function.

- `readtable` and `readtimetable` — Read XML data into MATLAB as a table or timetable. You can specify optional name-value arguments to control how `readtable` and `readtimetable` treat XML data. For example, specify `'ImportAttributes', false` to ignore attribute nodes.
- `writetable` and `writetimetable` — Write a table or timetable in MATLAB to an XML file. Specify optional name-value arguments to control how `writetable` and `writetimetable` treat XML data. For example, specify `'AttributeSuffix', '_att'` to specify that all table or timetable variables with the suffix `'_att'` should be written as attributes in the output XML file.
- `detectImportOptions` — The `detectImportOptions` function now returns an `XMLImportOptions` object when you call it on an XML file. Its behavior when you call it on other file types has not changed. Use the `XMLImportOptions` object with `readtable` to customize import options. For instance:
  - Import only a subset of data using the `SelectedVariableNames` property.
  - Specify the names and data types of the variables in the input file using the `VariableNames` and the `VariableTypes` properties.
  - Manage the import of specific nodes in the XML file using name-value arguments such as `'TableSelector'`, `'RowSelector'`, or `'VariableSelectors'`.

For more information, see `XMLImportOptions`.

### MATLAB API for Advanced XML Processing: Create, read, write, transform, and query XML

Use the MATLAB API for XML Processing (MAXP) to develop advanced applications that create, read, write, transform, and query XML documents. MAXP consists of these packages:

- `matlab.io.xml.dom` — Classes for creating, reading, and writing XML files and strings following the W3C DOM standard.
- `matlab.io.xml.transform` — Classes for transforming XML documents from one type to another following the XSLT 1.0 standard.
- `matlab.io.xml.xpath` — Classes for querying XML documents using XPath 1.0 expressions.

### XML Files: Register XML namespace prefixes for evaluating XPath expressions using `readtable`, `readstruct`, and other functions

Use the `RegisteredNamespaces` name-value argument to specify namespace prefixes that `readtable`, `readtimetable`, `readstruct`, `XMLImportOptions`, and `detectImportOptions` use when evaluating XPath expressions in an XML file. `RegisteredNamespaces` can be used when you also evaluate an XPath expression specified by a selector name-value argument, such as `StructSelector` for `readstruct`, or `VariableSelectors` for `readtable` and `readtimetable`.

The `readstruct` function automatically detects namespace prefixes to register for use in XPath evaluation, but you can also register new namespace prefixes using the `RegisteredNamespaces` name-value argument. You might register a new namespace prefix when an XML node has a namespace URL, but no declared namespace prefix in the XML file. In that case, you can specify `RegisteredNamespaces` as a string array containing a namespace prefix and the associated URL.

For example, evaluate an XPath expression on an XML file named `example.xml` which does not contain a namespace prefix declaration. Specify `'RegisteredNamespaces'` as `["myprefix", "https://www.mathworks.com"]` to assign the prefix `myprefix` to the URL `https://www.mathworks.com`.

```
data = readstruct("example.xml", "StructSelector", "/myprefix:Data", ...  
    "RegisteredNamespaces", ["myprefix", "https://www.mathworks.com"])
```

In the resulting structure, the namespace prefix and URL will appear as attributes belonging to the element specified in the `StructSelector` name-value argument.

## **Low-level file I/O functions and remote data: Perform read and write operations on remotely stored files**

You can now use low-level file I/O functions, such as `fopen`, `fread`, and `fwrite`, to work with files stored in remote locations. Some supported remote locations include Amazon S3 and Windows Azure Blob Storage.

When reading from or writing to a remote location, you must specify the full path using a uniform resource locator (URL). For example, open a binary file from the Amazon S3 cloud.

```
fid = fopen("s3://bucketname/path_to_file/example.bin");
```

For more information on setting up MATLAB to access your online storage service, see [Work with Remote Data](#).

## **save and load functions and remote data: Save, load, and append data to remotely stored v7.3 MAT-files**

You can now access v7.3 MAT-files stored in remote locations, such as Amazon S3 and Windows Azure Blob Storage, using the `save` and `load` functions.

When saving, loading, or appending data to a remote location, you must specify the full path using a uniform resource locator (URL). For example, load a MAT-file from the Amazon S3 cloud.

```
load("s3://bucketname/path_to_file/example.mat");
```

For more information on setting up MATLAB to access your online storage service, see [Work with Remote Data](#).

## **Reading Online Data: Read files over HTTP and HTTPS using readtable, audioread, and other reading functions**

Read files from an internet URL by specifying `filename` as a string that contains the protocol type `'http://'` or `'https://'`. This lets you read data from their primary online sources.

This functionality is supported by these functions: `audioread`, `audioinfo`, `parquetread`, `parquetinfo`, `readtable`, `readtimetable`, `readvars`, `readstruct`, `readmatrix`, `readcell`, `readlines`, and `detectImportOptions`.

## Parquet Data Format: Use categorical data in parquet data format

Read, write, and analyze parquet data that contain the categorical data type.

This functionality is supported by these functions: `parquetread`, `parquetwrite`, `parquetinfo`, and `parquetDatastore`.

## Datastores: Read all data from a datastore using parallel processing

You can use parallel processing when reading all data from a datastore (requires Parallel Computing Toolbox). Parallel processing results in improved performance when reading data, especially with remote data.

## Data Compression Functions: Improved functionality in zip/unzip and tar/untar

On Windows, macOS, and Linux systems:

- `zip` can compress an individual file of any size.
- `zip` can compress any number of files in a single function call.
- `tar` can compress a group of files of any cumulative size.

Additionally, on Windows systems, `unzip` and `untar` replace invalid characters with underscores if they occur in entry path names of the original file.

## imfinfo function: Get information about all Adobe Digital Negative (DNG) file tags

The `imfinfo` function returns information on all DNG file tags as individual named fields in the output structure. For a complete list of DNG file tags, see Chapter 4 of the Adobe® Digital Negative (DNG) Specification.

## jsonencode: Add indentation to JSON text

Use the `jsonencode` 'PrettyPrint' option to display JSON text with an indentation of two spaces.

```
s.Width = 800;
s.Height = 600;
s.Title = 'View from the 15th Floor';
s.Animated = false;
s.IDs = [116, 943, 234, 38793];
jsonencode(s, 'PrettyPrint', true)
```

```
ans =
    '{
      "Width": 800,
      "Height": 600,
```

```
        "Title": "View from the 15th Floor",
        "Animated": false,
        "IDs": [
            116,
            943,
            234,
            38793
        ]
    },
    ...
}
```

**▲ Functionality being removed or changed**

**The `H5I.get_name` function only accepts named HDF5 datatypes as input arguments.**  
*Behavior change*

Starting in R2020a, the `H5I.get_name` function only accepts committed (previously called named) HDF5 datatypes as input arguments, and will error if you pass other datatypes as input. In releases R2019b and earlier, `H5I.get_name` does not error if you pass other datatypes as input.

To verify that the input is a committed HDF5 datatype, call the `H5T.committed` function on it. The `H5T.committed` function returns a value of 1 if the input is a committed HDF5 datatype, and a value of 0 if it is not.

**`imfinfo` now returns Adobe DNG tags belonging to versions 1.2 through 1.5 in individual named fields in the output structure**  
*Behavior change*

When you call the `imfinfo` function on an Adobe DNG file, it now returns tags belonging to versions 1.2 through 1.5 as individual named fields in the output structure. Previously, tags belonging to these versions were stored in the 'UnknownTags' field of the output structure. For a complete list of DNG file tags, see Chapter 4 of the Adobe Digital Negative (DNG) Specification.

**`imread` reads the first frame in a GIF file by default**  
*Behavior change*

Starting in R2021a, when you read a GIF file without specifying additional arguments, the `imread` function reads the first frame by default. Previously, `imread` read all the frames in the file by default.

To read all the frames in the order that they appear in the GIF file, specify the value of the 'Frames' name-value argument as 'all'.

**`serial` function will be removed**  
*Still runs*

`serial` and its object properties will be removed. Previously, `serial` and its object properties were not recommended. Use `serialport` and its properties instead.

This example shows how to connect to a serial port device using the recommended functionality.

| Functionality                                                                                  | Use This Instead                            |
|------------------------------------------------------------------------------------------------|---------------------------------------------|
| <code>s = serial("COM1");</code><br><code>s.BaudRate = 115200;</code><br><code>fopen(s)</code> | <code>s = serialport("COM1",115200);</code> |

See [Transition Your Code to serialport Interface](#) for more information about using the recommended functionality.

## Mathematics

### Graph Algorithms: Compute all paths, all cycles, and cycle basis

graph and digraph objects have new functions to compute paths and cycles:

- `allpaths` — Compute all paths between two nodes in a graph or digraph object.
- `allcycles` — Compute all cycles in a graph or digraph object.
- `cyclebasis` — Compute the fundamental cycle basis of a graph object.
- `hascycles` — Determine whether a graph or digraph object contains cycles.

### griddedInterpolant Object: Use multivalued interpolation to interpolate multiple data sets simultaneously

`griddedInterpolant` can now interpolate multiple data sets on the same grid at the same query points. For example, if you specify a 2-D grid, a 3-D array of values at the grid points, and a 2-D collection of query points, then `griddedInterpolant` returns the interpolated values at the query points for each 2-D page in the 3-D array of values.

Previously, this functionality was available in `interp1` for 1-D interpolation, but this improvement to `griddedInterpolant` adds support for N-D multivalued interpolation.

### eig Function: Improved algorithm for skew-Hermitian matrices

`eig` now has an improved algorithm for input matrices that are skew-Hermitian. With the function call `[V,D] = eig(A)`, where `A` is skew-Hermitian, `eig` now guarantees that the matrix of eigenvectors `V` is unitary and the diagonal matrix of eigenvalues `D` is purely imaginary.

### cdf2rdf Function: Improved algorithm for all inputs

`cdf2rdf` has an improved algorithm for all input matrices that reduces floating-point round-off errors in the calculation.

### ⚠ Functionality being removed or changed

#### Line Continuation: Ellipsis following operator treated as a space

*Behavior change*

Previously, if an ellipsis followed an operator inside of a matrix or cell array, it caused the operator to be treated as unary. Ellipses will now be treated as a space in all cases.

| Old Behavior                                                                                                                                   | New Behavior                                                                                                                                                                     |
|------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <pre>x = [1 - ...<br/>2]<br/><br/>x =<br/>  1 -2</pre> <p>Previously, this code was equivalent to the expression, <code>x = [1 -2]</code>.</p> | <pre>x = [1 - ...<br/>2]<br/><br/>x =<br/>  -1</pre> <p>Now, the ellipsis will be treated as a space so this code is equivalent to the expression, <code>x = [1 - 2]</code>.</p> |

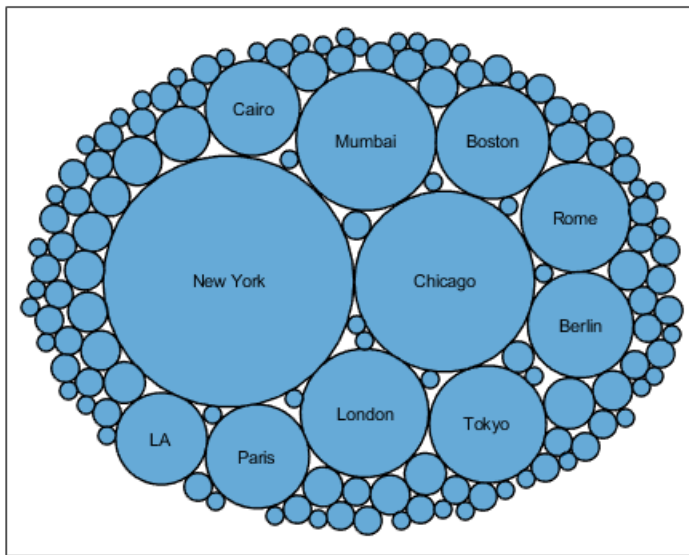
## Graphics

### Create Plot Live Editor Task: Create plots interactively and generate code

The Create Plot Live Editor Task makes generating and exploring visualizations for data simple and interactive. With this task you can select the data you wish to visualize and choose the plot type that best represents that data. Alternatively, you can explore the visualizations available in MATLAB to find the desired plot type and add your data. This task creates labels for the visualization based on the data and can be used to add or adjust the optional parameters of the visualization.

### bubblecloud Function: Visualize part-to-whole relationships

Use the `bubblecloud` function to illustrate the relationship between elements in your data set and the set as a whole. For example, you can visualize data collected from different cities, and represent each city as a bubble whose size is proportional to the value for that city.



### tilelayout Function: Control the tile indexing scheme

Control whether the tile indices increase across the rows or down the columns of a layout by setting the `TileIndexing` property of a `TiledChartLayout` object. Select one of the following options:

- `'rowmajor'` — Increment the tile indices across the first row before moving to the next row. This is the default behavior.
- `'columnmajor'` — Increment the indices down the first column before moving to the next column. This indexing scheme is the same as linear indexing for arrays.

The `nexttile` function populates tiles according to the indexing scheme. If you change the tile indexing of a populated layout, the tile positions change to match the new scheme.

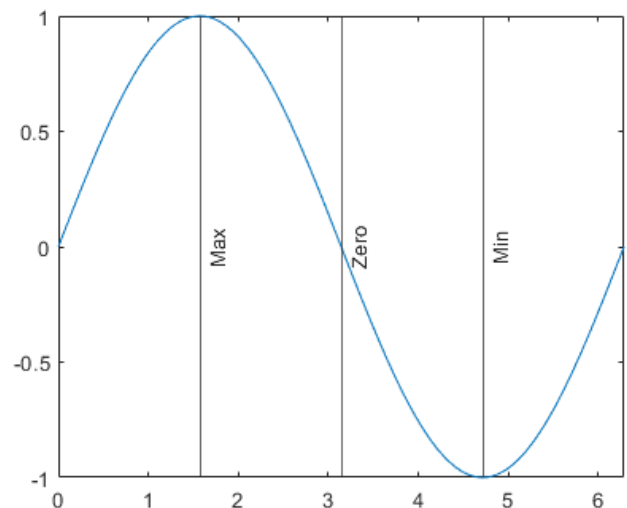
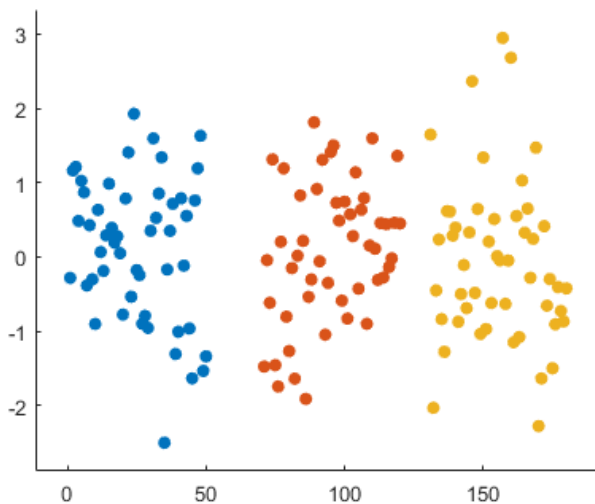


## PolarAxes Objects: Use the CurrentPoint property or call ginput to get the cursor location within polar axes

Query the `CurrentPoint` property of a `PolarAxes` object to get the location of the last click within the axes. The `ginput` function also supports querying coordinates of polar axes.

## Scatter Plots and Constant Lines: Create multiple scatter plots or constant lines at once

- The `scatter`, `polarscatter`, and `swarmchart` functions now accept the same combinations of matrices and vectors as the `plot` function does. As a result, you can visualize multiple data sets at once rather using the `hold` function between plotting commands.
- The `xline` and `yline` functions now accept vectors of values for creating multiple vertical or horizontal reference lines. You can also specify separate labels for each line using a string array or a cell array.



## Axis Limits: Define LimitsChangedFcn callback that executes when the limits of an axis change

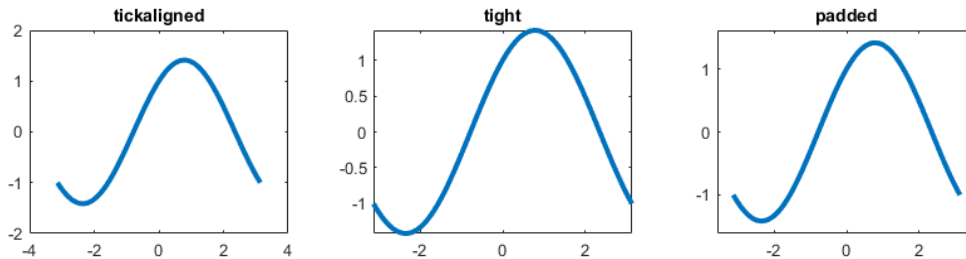
Define the `LimitsChangedFcn` callback function on any type of ruler object such as a numeric ruler. The callback function executes when the limits of the corresponding axis change. For example, you can define the callback in an app to update another aspect of the UI when the user pans within the axes.

## Axis Limits: Control axis limits

Control the axis limits for your plots by setting the `XLimitMethod`, `YLimitMethod`, or `ZLimitMethod` on the axes. Select one of the following property values:

- `'tickaligned'` — Align the edges of the axes box with the tick marks that are closest to your data without excluding any data. This is the default option.

- `'tight'` — Fit the axes box tightly around the data by setting the axis limits equal to the range of the data.
- `'padded'` — Fit the axes box around the data with a thin margin of padding on each side. The width of the margin is approximately 7% of your data range.



## exportgraphics and copygraphics Functions: Specify RGB, CMYK, or grayscale output

Choose a color space when exporting or copying graphics. Specify the `Colorspace` name-value argument when you call the `exportgraphics` or `copygraphics` functions. Select one of the following options:

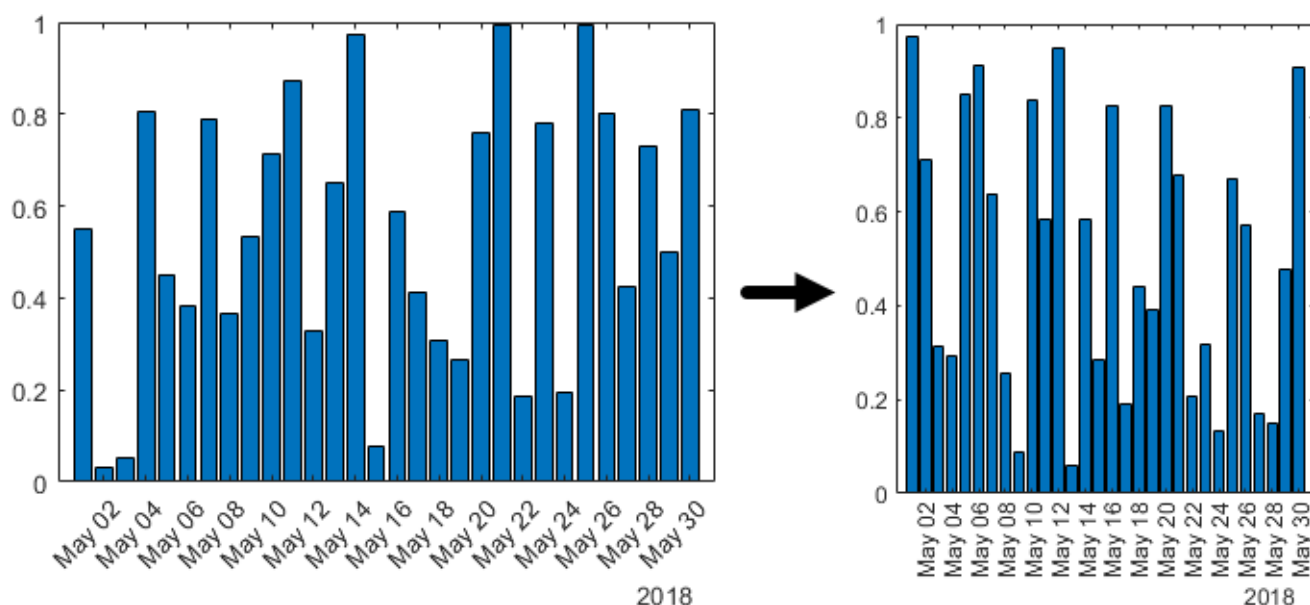
- `'rgb'` — Capture truecolor RGB content. This is the default color space.
- `'gray'` — Convert the content to grayscale.
- `'cmyk'` (`exportgraphics` only) — Convert the content to cyan, magenta, yellow, and black (CMYK) before exporting the content as an EPS file.

## colororder Function: Control colors in stacked plots

The `colororder` function now supports charts created with the `stackedplot` function.

## Tick Labels: Automatically rotate tick labels

When you manually specify the ticks or the tick labels for a chart, the tick labels automatically rotate to give the best possible presentation given the size of the figure and the number of the tick labels.



## patch and errorbar Functions: Expanded data type support

The `patch` and `errorbar` functions now support more data types:

- The `patch` function accepts numeric, datetime, duration, and categorical values for the x-, y-, and z-coordinates.
- The `errorbar` function accepts numeric, datetime, duration, and categorical values for the x- and y-coordinates. It also accepts numeric and duration values for the error bar lengths above, below, and on either side of the data points.

## Geographic Plots: Access basemaps using additional proxy server authentication types

You can now access basemaps for geographic axes and charts using additional proxy server authentication types.

- On Windows, you can use Basic, Digest, NTLM, Negotiate (SPNEGO), and Kerberos authentication.
- On Linux and macOS, you can use Basic, Digest, and NTLM authentication.

Prior to R2021a, geographic axes and charts supported only types without authentication or with Basic authentication. For more information about specifying proxy server settings, see [Use MATLAB Web Preferences For Proxy Server Settings](#).

## ▲ Functionality being removed or changed

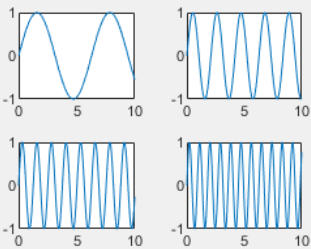
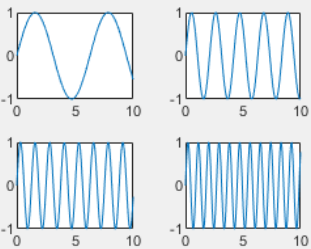
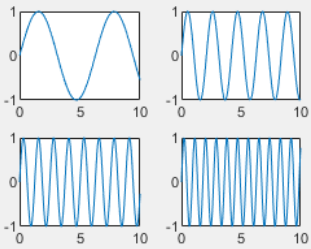
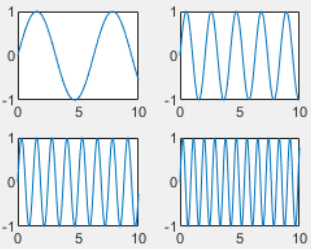
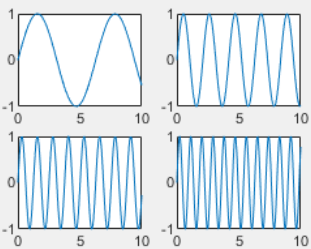
### Tile spacing and padding options for tiled chart layouts have changed

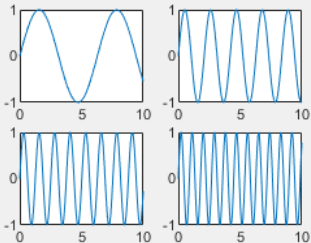
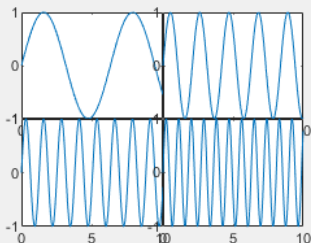
*Behavior change*

When you create a tiled chart layout, some of the `TileSpacing` and `Padding` properties provide a different result or have new names.

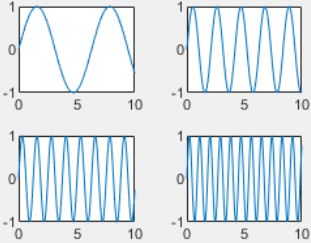
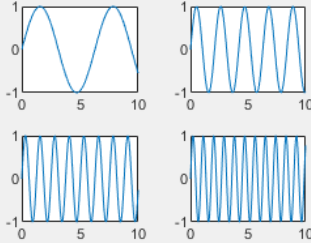
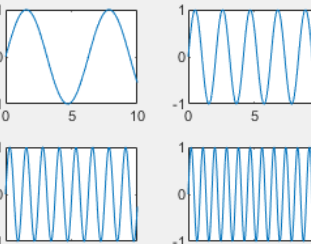
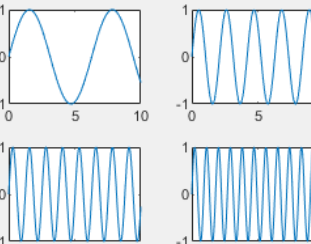
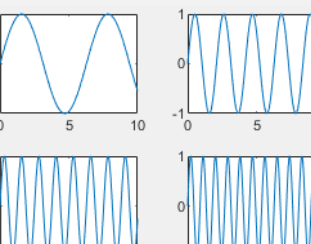
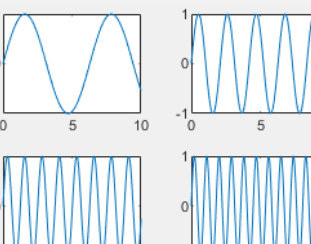
The new `TileSpacing` options are `'loose'`, `'compact'`, `'tight'`, and `'none'`. The new `Padding` options are `'loose'`, `'compact'`, and `'tight'`. The following tables describe how the previous options relate to the new options.

**TileSpacing Changes**

| Previous TileSpacing Option                                                                                       | R2021a TileSpacing Option                                                                                         | How to Update Your Code                                                                                                                                                                                               |
|-------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <p><code>'normal'</code></p>     | <p><code>'loose'</code></p>      | <p>Consider changing instances of <code>'normal'</code> to <code>'loose'</code>.</p> <p>The <code>'normal'</code> option is no longer recommended, but it continues to work, and there are no plans to remove it.</p> |
| <p><code>'compact'</code></p>  | <p><code>'compact'</code></p>  | <p>No changes needed.</p>                                                                                                                                                                                             |
| <p>Not Applicable</p>                                                                                             | <p><code>'tight'</code></p>    | <p><code>'tight'</code> is a new option. It provides the same result as <code>'none'</code> does in previous releases.</p>                                                                                            |

| Previous TileSpacing Option                                                                     | R2021a TileSpacing Option                                                                       | How to Update Your Code                                                                                                                                                                                                            |
|-------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <p>'none'</p>  | <p>'none'</p>  | <p>The 'none' option removes all the space between adjacent plot boxes, and the tick labels overlap with neighboring plot boxes.</p> <p>To preserve the spacing between the plot boxes, change instances of 'none' to 'tight'.</p> |

### Padding Changes

| Previous Padding Option                                                                              | R2021a Padding Option                                                                                | How to Update Your Code                                                                                                                                                        |
|------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <p>'normal'</p>    | <p>'loose'</p>     | <p>Consider changing instances of 'normal' to 'loose'.</p> <p>The 'normal' option is no longer recommended, but it continues to work, and there are no plans to remove it.</p> |
| <p>'compact'</p>  | <p>'compact'</p>  | <p>No changes needed.</p>                                                                                                                                                      |
| <p>'none'</p>     | <p>'tight'</p>    | <p>Consider changing instances of 'none' to 'tight'.</p> <p>The 'none' option is no longer recommended, but it continues to work, and there are no plans to remove it.</p>     |

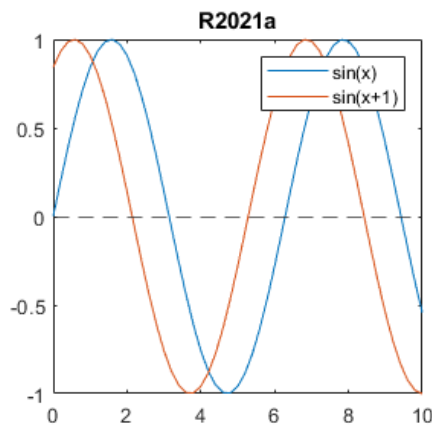
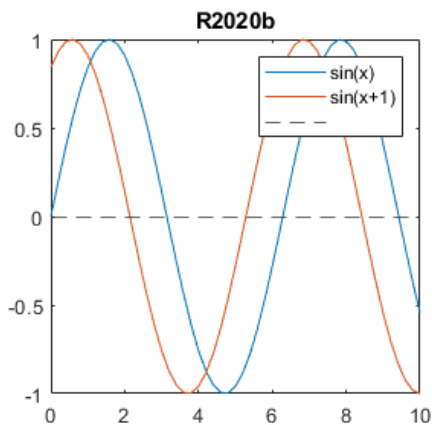
## Passing an empty label to the legend function omits the entry from the legend

### *Behavior change*

When you call the `legend` function and specify a label as an empty character vector, an empty string, or an empty element in a cell array or string array, the corresponding entry is omitted from the legend. In R2020b and earlier releases, the entry appears in the legend without a label.

For example, this code plots two sine waves and a reference line at  $y=0$ . Then it creates a legend with three labels, where the last label is empty. In R2020b, the third line appears in the legend without a label. In R2021a, the third line is omitted from the legend.

```
x = 0:0.2:10;
plot(x,sin(x),x,sin(x+1));
hold on
yline(0,'--')
legend('sin(x)', 'sin(x+1)', '')
```



To keep an entry in the legend without a label, include a space character in the label. For example, to update the preceding code, specify the last label as a character vector containing a space (' ').

```
legend('sin(x)', 'sin(x+1)', ' ')
```

Alternatively, if you do not want to display a space character, you can pass the individual line objects to the `legend` function with an array of labels. To get the individual line objects, call each plotting function with an output argument.

```
x = 0:0.2:10;
p = plot(x,sin(x),x,sin(x+1));
hold on
line0 = yline(0,'--');
legend([p(1) p(2) line0], {'sin(x)', 'sin(x+1)', ''});
```

## The XData, YData, and ZData properties on Patch objects created with the fill and fill3 functions return values of the original data type

### *Behavior change*

The `XData`, `YData`, and `ZData` properties on a `Patch` object created by the `fill` or `fill3` functions return the coordinates using the original input data type, rather than returning them as double values.

In previous releases, `datetime`, `duration`, and categorical coordinates are converted to double values when they are stored in the `XData`, `YData`, and `ZData` properties.

For example, this code creates a filled polygon with `datetime` x-coordinates. Then it calculates `x2` using the values stored in the `XData` property. In R2020b, `h.XData` and `x2` are double arrays. In R2021a, `h.XData` and `x2` are `datetime` arrays.

```
x = datetime('01-Jan-2018') + days([0 1 1 0]);
y = [0 0 1 1];
h = fill(x,y,'red');
x2 = h.XData + 1;
```

To preserve the double values in your code, get the double values from the `Vertices` property of the `Patch` object. The x-, y-, and z-coordinates are stored as double values in the first, second, and third columns of the `Vertices` array.

```
x2 = h.Vertices(:,1) + 1;
```

Alternatively, use the `ruler2num` function. Pass the coordinate values and the corresponding axis ruler to the `ruler2num` function.

```
ax = gca;
x2 = ruler2num(h.XData,ax.XAxis) + 1;
```

### Graphics objects and UI components sized using 'points', 'inches', and 'centimeters' units will increase in size on macOS platforms

*Behavior change in future release*

In a future release, graphics and UI objects that have `Units` or `FontUnits` properties set to `'points'`, `'inches'`, or `'centimeters'` will use a conversion value of 1 pixel = 1/96th inch on macOS platforms. The current conversion value is 1 pixel = 1/72nd inch. As a result, these objects and text elements will display 1.33 times larger than their previous size. This change will provide a more readable default font size and will ensure a consistent object size across Windows and macOS platforms.

The following objects use a unit value of `'pixels'` by default and will not be affected by this change:

- UI components in App Designer or in apps created with the `uifigure` function
- UI components in apps created in GUIDE and migrated to App Designer using the GUIDE to App Designer Migration Tool for MATLAB
- Axes objects created using the `uiaxes` function

For the most control when sizing and laying out your graphics objects and UI components, use a value of `'pixels'` for `Units` and `FontUnits` properties. To maintain object sizes in current and future releases, make these updates in your code:

- Objects with `Units` or `FontUnits` set to `'points'` — Update the value of the property from `'points'` to `'pixels'`.
- Objects with `Units` or `FontUnits` set to `'inches'` — Update the value of the property from `'inches'` to `'pixels'` and multiply all `Position` values by 72.
- Objects with `Units` or `FontUnits` set to `'centimeters'` — Update the value of the property from `'centimeters'` to `'pixels'` and multiply all `Position` values by 72/2.54.

For example, this code creates a push button in a figure window, with its position specified in inches:

```
uicontrol('Units','inches','Position',[0.6 0.1 1.75 0.5]);
```

In a future release, the push button created by this code will display 1.33 times larger on macOS platforms. To maintain the size and position of the push button in current and future releases, update the code to:

```
uicontrol('Units','pixels','Position',[50 10 126 36]);
```

### **Align Distribute Tool will be removed in a future release**

#### *Warns*

The Align Distribute Tool will be removed in a future release.

To control the arrangement of multiple plots in a figure, create a tiled chart layout using the `tiledlayout` function instead.

To align or distribute graphics objects within a figure, select **Tools > Align** or **Tools > Distribute** from the figure toolbar instead.



## App Building

### **uihyperlink Function: Add and configure clickable links in apps and on the App Designer canvas**

To create a link, call the `uihyperlink` function or, in App Designer, drag a hyperlink UI component from the **Component Library** onto the canvas.

Hyperlink UI components are supported only in App Designer apps and in figures created with the `uifigure` function.

### **uitree Function: Add and configure check box trees in apps and on the App Designer canvas**

A check box tree is a styled with check boxes to the left of every item. You can now create check box trees in apps and on the App Designer canvas. Check box trees allow for easier selection of multiple tree nodes.

In apps created programmatically with the `uifigure` function, create a check box tree using the `uitree` function by specifying the style `'checkbox'`.

In App Designer, create a check box tree by dragging it from the **Component Library** onto the canvas.

### **Interpreter Property: Style text and display equations in labels with HTML and LaTeX markup**

Use the Interpreter property on Label objects (created with the `uicontrol` function) to enable markup in the label text. To add HTML markup, set the Interpreter property to `'html'`. To add LaTeX markup, set the Interpreter property to `'latex'`.

For more information, see `Label` Properties.

### **WindowState Property: Create UI figures that remain in the foreground**

To keep a specific UI figure window in front of other windows, set the `WindowState` property to `'alwaysontop'`. Unlike modal figures, UI figure windows with this property setting do not restrict keyboard and mouse interactions.

The `'alwaysontop'` property value is available only in figures created using the `uifigure` function.

For more information, see `UI Figure` Properties.

### **scroll Function: Scroll to a location within a table UI component programmatically**

Use the `scroll` function to scroll within a table UI component programmatically. Specify the scroll location as the top, bottom, left, or right side of the table, or as a specific row, column, or table cell.

## UI Component Accessibility: Select ListBox items, Table cells, ColorPicker colors, and DatePicker menus using the keyboard

You can now use keyboard shortcuts to change the focus and make selections in various UI components. These shortcuts are supported for components in figures created using the `uifigure` function.

| Component                                      | Action                                                  | Keyboard Shortcut  |
|------------------------------------------------|---------------------------------------------------------|--------------------|
| ListBox with Multiselect property set to 'off' | Toggle list box selection.                              | <b>Space</b>       |
| ListBox with Multiselect property set to 'on'  | Select all items.                                       | <b>Ctrl+A</b>      |
|                                                | Move focus to previous item and toggle selection.       | <b>Shift+Up</b>    |
|                                                | Move focus to next item and toggle selection.           | <b>Shift+Down</b>  |
|                                                | Move focus to previous item without toggling selection. | <b>Ctrl+Up</b>     |
|                                                | Move focus to next item without toggling selection.     | <b>Ctrl+Down</b>   |
|                                                | Toggle selection of item currently in focus.            | <b>Ctrl+Space</b>  |
| Table                                          | Select table cell in top-left corner.                   | <b>Ctrl+Home</b>   |
|                                                | Select table cell in bottom-right corner.               | <b>Ctrl+End</b>    |
| ColorPicker gradient selector                  | Move gradient selector.                                 | Arrow keys         |
| ColorPicker hue slider                         | Move hue slider.                                        | <b>Up and Down</b> |
| DatePicker                                     | Cycle between drop down menus, buttons, and calendar.   | <b>Tab</b>         |

## App Designer: Use custom UI components in App Designer

You can now configure custom UI component classes to appear in the App Designer **Component Library** and to be used interactively in **Design View**.

To configure a custom UI component class for use in App Designer, follow these steps:

- Define your own UI component class by creating a subclass of the `matlab.ui.componentcontainer.ComponentContainer` base class.
- Call the `appdesigner.customcomponent.configureMetadata` function and specify the path to the component class `.m` file.
- Use the resulting dialog box to configure the metadata associated with the component, including the component name, icon, author, and category.

This creates a `resources` folder with the App Designer metadata. To view the component in the App Designer **Component Library**, add the folder containing the component class file and the generated `resources` folder to the MATLAB path.

To share your custom UI component for others to use in App Designer, share both the component class file and the `resources` folder.

For more information, see [Configure Custom UI Components for App Designer](#).

## ▲ App Designer: Zoom and pan in the canvas, and zoom in the Code View editor

To zoom in or out in the App Designer canvas and in the **Code View** editor, hold **Ctrl** and move the scroll wheel, or press **Ctrl+Plus** and **Ctrl+Minus**. To return to the default scale, press **Ctrl+Alt+0**. Zooming does not affect the **Component Library**, **Component Browser**, or **Code Browser**.

To pan in the App Designer canvas, use one of the following:

- **Click** and drag with the middle mouse button.
- Hold **Space** while clicking and dragging with the left mouse button.

## ▲ Compatibility Considerations

In previous releases, the **Ctrl+Plus** and **Ctrl+Minus** keyboard shortcuts zoomed the entire App Designer desktop.

## App Designer: Control color and tab settings in Code View using MATLAB preferences

Color and tab preferences applied to the MATLAB Editor are now also applied to the App Designer Editor.

Additionally, you can now change the background color of the App Designer read-only code. Access this setting in the App Designer tab of MATLAB preferences.

For more information, see [Personalize Code View Appearance](#).

## App Designer: Customize split-screen layouts in the App Designer editor

To view your document in horizontal or vertical split-screen mode in the App Designer **Code View** editor, select a layout in the App Designer toolbar.

## App Testing Framework: Perform gestures on panels and tables

The app testing framework supports gestures on more UI components:

- Perform `press`, `hover`, and `chooseContextMenu` gestures on panels.
- Perform `choose`, `type`, and `chooseContextMenu` gestures on table UI components.

## App Testing Framework: Close alert dialog box in front of figure window

You can now use the `dismissAlertDialog` method as part of a test case to programmatically close an alert dialog box in front of a figure window. For example, create a modal alert dialog box and close it by calling the method.

```
fig = uifigure;
uiaalert(fig, 'File not found', 'Invalid File')

tc = matlab.uitest.TestCase.forInteractiveUse;
tc.dismissAlertDialog(fig)
```

## Web Apps and Standalone Applications: Datatips supported in graphics

Graphics created in web apps and standalone applications support datatips. Use datatips in these applications just as you would in MATLAB figures.

## ⚠️ Functionality Being Removed or Changed

### GUIDE will be removed in a future release

#### *Warns*

The GUIDE environment and the `guide` function will be removed in a future release.

After GUIDE is removed, existing GUIDE apps will continue to run in MATLAB but will not be editable using the drag-and-drop GUIDE environment. To continue editing an existing GUIDE app and help maintain its compatibility with future MATLAB releases, use one of the suggested migration strategies listed in the table.

| App Development                 | Migration Strategy                                                                                 | How to Migrate                                                                                                     |
|---------------------------------|----------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------|
| Frequent or ongoing development | Migrate your app to App Designer.                                                                  | Use the GUIDE to App Designer Migration Tool for MATLAB on <a href="https://www.mathworks.com">mathworks.com</a> . |
| Minimal or occasional editing   | Export your app to a single MATLAB file to manage your app layout and code using MATLAB functions. | Open the app in GUIDE and select <b>File &gt; Export to MATLAB-file</b> .                                          |

To create new apps, use App Designer and the `appdesigner` function instead. App Designer is the recommended app development environment in MATLAB.

To learn more about migrating apps, see [GUIDE Migration Strategies](#).

For more information about App Designer, go to [Comparing GUIDE and App Designer](#) on [mathworks.com](https://www.mathworks.com).

### GUIDE templates have been removed

All GUIDE templates other than the blank GUI have been removed. To create new apps interactively, use App Designer and the `appdesigner` function instead.

### **Graphics objects and UI components sized using 'points', 'inches', and 'centimeters' units will increase in size on macOS platforms**

*Behavior change in future release*

In a future release, graphics and UI objects that have `Units` or `FontUnits` properties set to `'points'`, `'inches'`, or `'centimeters'` will use a conversion value of 1 pixel = 1/96th inch on macOS platforms. The current conversion value is 1 pixel = 1/72nd inch. As a result, these objects and text elements will display 1.33 times larger than their previous size. This change will provide a more readable default font size and will ensure a consistent object size across Windows and macOS platforms.

The following objects use a unit value of `'pixels'` by default and will not be affected by this change:

- UI components in App Designer or in apps created with the `uifigure` function
- UI components in apps created in GUIDE and migrated to App Designer using the GUIDE to App Designer Migration Tool for MATLAB
- Axes objects created using the `uiaxes` function

For the most control when sizing and laying out your graphics objects and UI components, use a value of `'pixels'` for `Units` and `FontUnits` properties. To maintain object sizes in current and future releases, make these updates in your code:

- Objects with `Units` or `FontUnits` set to `'points'` — Update the value of the property from `'points'` to `'pixels'`.
- Objects with `Units` or `FontUnits` set to `'inches'` — Update the value of the property from `'inches'` to `'pixels'` and multiply all `Position` values by 72.
- Objects with `Units` or `FontUnits` set to `'centimeters'` — Update the value of the property from `'centimeters'` to `'pixels'` and multiply all `Position` values by 72/2.54.

For example, this code creates a push button in a figure window, with its position specified in inches:

```
uicontrol('Units','inches','Position',[0.6 0.1 1.75 0.5]);
```

In a future release, the push button created by this code will display 1.33 times larger on macOS platforms. To maintain the size and position of the push button in current and future releases, update the code to:

```
uicontrol('Units','pixels','Position',[50 10 126 36]);
```

## Performance

### Sparse Matrix Multiplication: Improved performance multiplying large sparse matrices

Matrix multiplication performance has improved when multiplying sparse matrices. The performance improvement arises from added support for multithreading in the operation, and therefore the speedup gets better as the matrix size and number of nonzeros increase.

For example, if you multiply two 1e5-by-1e5 random sparse matrices with approximately two million nonzeros, performance in R2021a is about 4.4x faster than in R2020b on a machine with 6 physical cores.

```
function timingSparseMult
rng default
A = sprand(1e5,1e5,0.0002);
B = sprand(1e5,1e5,0.0002);
tic
C = A*B;
toc
end
```

The approximate execution times are:

**R2020b:** 2.2 s

**R2021a:** 0.5 s

The code was timed on a Windows 10, Intel Xeon W-2133 CPU @ 3.60 GHz test system by calling the function `timingSparseMult`.

### Sparse Linear Systems: Improved performance solving sparse linear systems $A*X = B$ with multicolumn B

Solving a linear system of the form  $A*X = B$  by executing  $X = A \setminus B$  shows improved performance when A is a sparse square matrix and B is a matrix with two or more columns. The speedup applies to the solving step of the calculation but not the factorization step. The performance improvement arises from added support for multithreading, and therefore the speedup gets better as the number of columns in B increases.

For example, if you solve  $A*X = B$  using a 1e4-by-1e4 sparse coefficient matrix with approximately 40,000 nonzeros and a B matrix with 100 columns, performance in R2021a is about 5x faster than in R2020b on a machine with 6 physical cores. This code uses `decomposition` to factor the coefficient matrix, so only the solving process is timed. If you use  $X = A \setminus B$  instead, you still see a speedup, but the time required to factor the matrix is included and has not changed.

```
function timingSparseBackslashMultRHS
rng default
A = sprand(1e4,1e4,0.0003) + speye(1e4);
B = sprand(1e4,100,0.002);
dA = decomposition(A);
tic
x = dA \ B;
```

```
toc
end
```

The approximate execution times are:

**R2020b:** 1.5 s

**R2021a:** 0.3 s

The code was timed on a Windows 10, Intel Xeon W-2133 CPU @ 3.60 GHz test system by calling the function `timingSparseBackslashMultRHS`.

## vecnorm Function: Improved performance operating on data with multiple columns

The performance of the `vecnorm` function has improved for all norm types when the data has 16 or more columns and at least  $2^{17}$  elements.

The improvement also applies to N-D array data that can be permuted into a matrix with the requisite number of columns. The speedup varies depending on the type of norm being calculated.

For example, if you calculate the 2-norm of a 1000-by-1000-by-3 array along the third dimension, performance in R2021a is about 7.3x faster than in R2020b.

```
function timingVecnorm
rng default
N = 1000;
A = rand(N,N,3);
for k = 1:200
    D = vecnorm(A,2,3);
end
end
```

The approximate execution times are:

**R2020b:** 8.8 s

**R2021a:** 1.2 s

The code was timed on a Windows 10, Intel Xeon W-2133 CPU @ 3.60 GHz test system using the `timeit` function:

```
timeit(@timingVecnorm)
```

## ismember Function: Improved performance for cell inputs

The `ismember` function shows improved performance operating on cell inputs. The speedup depends on the size and layout of the data, with the largest speedup when the input has many cells that contain few elements in each cell.

For example, if you use `ismember` to compare two 1000-by-1 cell arrays with 10 elements in each cell, performance in R2021a is about 4.7x faster than in R2020b.

```
function timingIsmember
a = num2cell(char(randi(127,[1000 10])),2);
```

```
b = num2cell(char(randi(127,[1000 10])),2);
tic
for ii = 1:1e4
    ismember(a,b);
end
toc
end
```

The approximate execution times are:

**R2020b:** 6.6 s

**R2021a:** 1.4 s

The code was timed on a Windows 10, Intel Xeon W-2133 CPU @ 3.60 GHz test system by calling the function `timingIsMember`.

## unique Function: Improved performance for numeric, logical, char, and cell inputs

The `unique` function shows improved performance operating on numeric, logical, char, and cell inputs. The speedup generally gets better as the size of the inputs increases, and the improvement applies when using any optional flags except the `'legacy'` flag.

For example, if you operate on a 100-by-1 cell array with 10 elements in each cell, performance in R2021a is about 3.3x faster than in R2020b.

```
function timingUniqueCell
a = num2cell(char(randi(127,[100 10])),2);
tic
for ii = 1:1e5
    b = unique(a);
end
toc
end
```

The approximate execution times are:

**R2020b:** 3.9 s

**R2021a:** 1.2 s

Also, if you use `unique` on a numeric input with 10,000 elements and specify three outputs with the `'stable'` option, performance in R2021a is about 3.5x faster than in R2020b.

```
function timingUniqueNumeric
a = rand(10000,1);
tic
for ii=1:1e4
    [C,ia,ic] = unique(a,'stable');
end
toc
end
```

The approximate execution times are:

**R2020b:** 9.4 s



**R2021a:** 2.7 s

In both cases, the code was timed on a Windows 10, Intel Xeon W-2133 CPU @ 3.60 GHz test system by calling the functions `timingUniqueCell` and `timingUniqueNumeric`.

## Graph Functions: Improved performance modifying node and edge lists

`graph` and `digraph` functions that modify the node and edge lists of the graph show improved performance. This applies to the functions `addedge`, `rmedge`, `addnode`, `rmnode`, `subgraph`, and `reordernodes`. The improvement applies to graphs that have no node properties (or only node names), and graphs with no edge properties (or only edge weights). The improvement is most noticeable when one of these functions is called many times in a loop, and the largest improvement applies to graphs that have both node names and edge weights.

For example, if you use `addedge` in a loop to add new edges with node names and edge weights to an empty graph, performance in R2021a is about 13x faster than in R2020b.

```
function timingAddedge
names = string('A':'Z') + (1:10);
names = names(1:100);
rng default
g = graph;
for ii = 1:1e3
    g = addedge(g, names(randi(100)), names(randi(100)), randn);
end
end
```

The approximate execution times are:

**R2020b:** 2.6 s

**R2021a:** 0.2 s

The code was timed on a Windows 10, Intel Xeon W-2133 CPU @ 3.60 GHz test system by using the `timeit` function:

```
timeit(@timingAddedge)
```

## Axes Toolbar: Appears without delay when axes are ready

Prior to R2021a, when hovering the cursor over figure axes, there was delay before the axes toolbar appeared. Now the toolbar will appear as soon as the axes are ready.

## Rearranging UI Components: Improved performance when rearranging UI components in a UI figure

Programmatically rearranging existing UI components in a figure created with the `uifigure` function shows improved performance.

For example, if you sort 50 panels within a grid layout, performance in R2021a is approximately 1.9x faster than in R2020b.

```
function timingSortComp
    % Create components
    panels = {};
    fig = uifigure;
    g = uigridlayout(fig,[1,1],'RowHeight',40);
    g.Scrollable = true;
    num = 50;
    for i = 1:num
        p = uipanel(g);
        uilabel(p,'Text',[ 'Panel ', num2str(i)],'Position',[10 10 70 22]);
        g.RowHeight(end) = 40;
        panels{end+1} = p;
    end
    drawnow;

    % Rearrange components
    tic
    order = length(panels):-1:1;
    for i = 1:length(order)
        panels{i}.Layout.Row = order(i);
    end
    drawnow;
    toc
end
```

The approximate execution times are:

**R2020b:** 0.70 s

**R2021a:** 0.36 s

The code was timed on a Windows 10, Intel Xeon E5-1650 CPU @ 3.60 GHz test system by calling the function `timingSortComp`.

## UI Figure Interactions: Faster responses to scroll, pointer movement, and resize interactions in UI figures

In figures created with the `uifigure` function, the following interactions have improved performance:

- Scrolling in a figure with a `WindowScrollWheelFcn` callback or an object with a predefined scroll behavior
- Resizing a visible figure with a `SizeChangedFcn` callback or an object with a predefined resize behavior
- Moving the mouse pointer in a figure with a `WindowButtonMotionFcn` callback when the figure contains any UI components except axes components

This performance increase is more noticeable when using a trackpad to interact with the figure.

For example, scrolling to zoom in on the plot created by the code below is smoother and more responsive in R2021a than in R2020b.

```
t1 = datetime(2019,1,1);
t2 = datetime(2020,1,1);
dates = linspace(t1,t2,10000);
data = rand(10000,10);
fig = uifigure;
stackedplot(fig,dates,data);
```

On a Windows 10, Intel Xeon W-2133 CPU @ 3.60 GHz test system, the responses to the scroll action are:

**R2020b:** When you zoom in on the plot by scrolling for approximately two seconds, the plot has about a five second delay in completing the zoom animation.

**R2021a:** The zoom animation completes immediately after you finish the scroll action.

## Plots in Apps: Improved performance for polar plots, volume visualizations, plots with more than 16 axes, and older systems

Displaying polar plots, volume visualizations, or more than 16 axes in an app have improved performance. The improvement affects plots displayed in apps:

- Plots that are displayed in an app created with App Designer
- Plots displayed in a figure created with the `ui figure` function

Systems with older graphics drivers might experience the improvement for *all* types of plots that are created within the apps and figures listed above. For example, Intel drivers earlier than version 10.0.0.0 for Windows systems will experience additional improvements.

This code creates a polar plot and executes a `for` loop that changes the theta values at every iteration. The `for` loop executes about 2x faster than in R2020b.

```
function timingPolar
f = uifigure;
pax = polaraxes('parent',f);
theta = 0:0.01:2*pi;
rho = sin(2*theta).*cos(2*theta);
pp = polarplot(pax, theta,rho);
pax.FontSize = 12;
drawnow

tic;
for i=1:100
    pp.ThetaData = pp.ThetaData + .02*pi;
    drawnow
end
toc
end
```

The approximate execution times are:

**R2020b:** 10.15 s

**R2021a:** 5.30 s

The code was timed on a Windows 10, Intel Xeon W-2133 CPU @ 3.60 GHz test system by calling the function `timingPolar`.

## Plots in Apps: Improved performance for plots with large numbers of markers

Performance is improved for modifying certain types of plots in apps. You can observe the improvement when the following conditions are true:

- The plots are displayed in an App Designer app, or they are displayed in a figure created with the `uifigure` function.
- Your system is running a locally installed version of MATLAB on a modern Windows or macOS system.
- You run the code either from the MATLAB command window or within a program file (.m file).

Note that plots created in live scripts do not show this improvement.

The plots typically contain large numbers of markers, and your code updates an aspect of those markers, such as their positions. The improvement is more significant as you increase the number of markers. For example, if you create a scatter plot with 10 million markers, and change the marker positions 10 times, the performance in R2021a is about 1.3x faster than in R2020b.

```
function timingScatter
f = uifigure;
a = axes(f);
x = rand(1e7, 2);
s = scatter(a,x(:,1),x(:,2), 'Marker', '*');
drawnow;

tic;
for i=1:10
    x = rand(1e7,2);
    s.XData = x(:,1);
    s.YData = x(:,2);
    drawnow
end
toc
end
```

The approximate execution times are:

**R2020b:** 19.43 s

**R2021a:** 14.88 s

The code was timed on a macOS 10.14.6, Intel Core i9 CPU @ 3.60 GHz test system by calling the function `timingScatter`.

## Live Editor: Improved performance when saving large live scripts or functions

When saving large live scripts or functions, you can continue using the Live Editor sooner in R2021a than in R2020b. While you continue to use the Live Editor, MATLAB saves the file in the background. When MATLAB finishes saving the file, the asterisk (\*) next to the file name disappears, indicating that the file is saved.

For example, on a Windows 10, Intel Xeon E5-1650 CPU @ 3.60 GHz test system, if you save a live script containing 35,000 lines of code and then click the title bar of another document open in the

Live Editor, MATLAB immediately switches to the other open document. In R2020b, there is a noticeable delay before MATLAB switches to the other open document.

## Software Development Tools

### Projects: List all referenced projects of the current project

You can now use `listAllProjectReferences` to programmatically list all projects in the reference hierarchy of a specified project.

### Projects: List impacted project files

You can now use `listImpactedFiles` to programmatically list all project files impacted by changes to a specified file.

### Dependency Analyzer: Find required add-ons

Starting in R2021a, the Dependency Analyzer detects and lists required add-ons, including apps and toolboxes, for the whole project or for selected files. The Dependency Analyzer might not detect required support packages. For more details, see [Find Required Products and Add-Ons](#).

### Unit Testing Framework: Create test runners using alternative syntax

You can now use the `testrunner` function to create a runner for tests authored using the MATLAB unit testing framework or Simulink Test. In previous releases, you can explicitly create a runner only by calling one of the static methods of the `matlab.unittest.TestRunner` class.

Use the `testrunner` function to create a default runner, a minimal runner with no plugins installed, or a runner configured for text output. For example, create a default runner to run the tests in a test class.

```
suite = testsuite('MyTestClass');  
runner = testrunner;  
results = run(runner,suite);
```

### Unit Testing Framework: Initialize parameterization properties at suite creation time

Starting in R2021a, you can specify parameterization properties that do not have a default value. This feature is useful when parameters cannot be determined at the time MATLAB loads the test class definition. To initialize a parameterization property at test suite creation time, use a static method with the `TestParameterDefinition` attribute. For more information, see [Define Parameters at Suite Creation Time](#).

### Unit Testing Framework: Run tests in parallel on thread-based pool

You can now run your tests on a thread-based parallel pool (requires Parallel Computing Toolbox). To run tests using thread workers, start a thread-based pool and then call the `runInParallel` method or the `runtests` function with the `UseParallel` name-value pair argument.

Thread-based parallel pools support only a subset of MATLAB and testing framework functionality. For more information, see `runInParallel` or `runtests`.

## Unit Testing Framework: Run tests in MATLAB Online interactively

Starting in R2021a, you can run tests in MATLAB Online interactively. When you open a .m file defining a function-based or class-based test in MATLAB Online, the toolstrip lets you run all tests in the file or run the current test in the file. Also, you can customize the test run with options, such as running tests in parallel (requires Parallel Computing Toolbox) or running tests with a specified level of output detail.

The **Run Tests** section in the **Editor** tab of the toolstrip provides an alternative to programmatically running tests with the `runtests` function. For more information, see [Run Tests in Editor](#).

## App Testing Framework: Perform gestures on panels and tables

The app testing framework supports gestures on more UI components:

- Perform `press`, `hover`, and `chooseContextMenu` gestures on panels.
- Perform `choose`, `type`, and `chooseContextMenu` gestures on table UI components.

## App Testing Framework: Close alert dialog box in front of figure window

You can now use the `dismissAlertDialog` method as part of a test case to programmatically close an alert dialog box in front of a figure window. For example, create a modal alert dialog box and close it by calling the method.

```
fig = uifigure;
uialert(fig, 'File not found', 'Invalid File')

tc = matlab.uitest.TestCase.forInteractiveUse;
tc.dismissAlertDialog(fig)
```

## ▲ Functionality being removed or changed

### Character data is not supported in custom examples demos.xml file

*Behavior change*

Starting in R2021a, when creating custom examples, character data is not supported in the description of the `demos.xml` file. If you have a `demos.xml` file that contains character data such as `<`, `>`, `'`, `"`, and `&`, the description does not appear correctly in the Help browser.

To patch an existing `demos.xml` that contains character data, use the `patchdemoxmlfile` function. `patchdemoxmlfile` patches the specified `demos.xml` file, replacing character data with non-character data.

For example, patch the `demos.xml` file in the folder `D:\Work\mytoolbox\help`:

```
patchdemoxmlfile D:\Work\mytoolbox\help
```

## External Language Interfaces

### C++ Interface: Support for C++ language features

The C++ interface supports these additional C++ language features.

- Support for `std::vector` values containing `std::string` values and C++ arrays containing C-style strings. For more information, see [String and Character Types in the C++ to MATLAB Data Type Mapping](#) topic.
- Support for `void*` values as input and output arguments. For more information, see [Use void\\* Arguments](#), [void\\* Argument Types](#), and [addOpaqueType](#).
- Pass C++ functions to function pointers. For more information, see [Use Function Type Arguments](#) and [addFunctionType](#).
- Support for function and member function template instantiations. Publishers can modify function names. For more information, see [Customize Function Template Names](#).

### C++ Interface: Publisher options and analysis

The C++ interface supports these build configuration features.

- Generate an interface from header and source (.cpp) files. Pass a .cpp or .hpp file in the `clibgen.generateLibraryDefinition SourceFiles` argument.
- Generate an interface from a header and a .dll file for Microsoft Visual Studio compilers. Pass a .dll file in the `clibgen.generateLibraryDefinition LibraryFiles` argument.
- Improved troubleshooting messages.

### ⚠Java Packages to be removed

Java packages and subpackages that currently ship with MATLAB will not be available in MATLAB in a future release.

### ⚠Compatibility Considerations

To continue using a Java package, install its JAR file and add the JAR file to the static path in MATLAB using the instructions in [Static Path](#).

### Java Engine: MATLAB value object support

To work with MATLAB value objects in a Java engine application, use the `com.mathworks.matlab.types.ValueObject` class in the [Java Engine API Summary](#). You can create a value object in MATLAB, return it to Java, and call its methods. For information about mapping Java data types to MATLAB data types, see [Java Data Type Conversions](#).

### ⚠Python Interface and Engine: Version 3.6 support discontinued

Support for Python version 3.6 is discontinued.



## ⚠ Compatibility Considerations

To ensure continued support for your applications, upgrade to a supported version of Python, either version 3.7 or 3.8. For more information, see [Versions of Python Compatible with MATLAB Products by Release](#).

## ⚠ Perl 5.32.0: MATLAB support on Windows

As of R2021a, MATLAB on Windows ships with an updated version of Perl, version 5.32.0.

- See [www.perl.org](http://www.perl.org) for a standard distribution of Perl, Perl source, and information about using Perl.
- See <https://metacpan.org/pod/HTML::Parser> for a standard distribution of `HTML::Parser`, source code, and information about using `HTML::Parser`.
- See <https://metacpan.org/pod/HTML::Tagset> for a standard distribution of `HTML::Tagset`, source code, and information about using `HTML::Tagset`.

## ⚠ Compatibility Considerations

If you use the `perl` command on Windows platforms, see [www.perl.org](http://www.perl.org) for information about using this version of the Perl programming language.

## Hardware Support

### Support added for IMU sensors

The MATLAB Support Package for Raspberry Pi Hardware now provides code generation and connected IO support to Raspberry Pi functions for these IMU sensors:

- HTS221
- LPS22HB
- LSM303C
- LSM6DSL
- LSM9DS1
- MPU-6050
- MPU-9250

### New functionalities added to Raspberry Pi Resource Monitor app

The Raspberry Pi Resource Monitor App from the MATLAB Support Package for Raspberry Pi Hardware has been improved to:

- Display peripherals used in a MATLAB or Simulink application deployed on the Raspberry Pi hardware
- Enable or disable peripherals
- Check for missing libraries and packages
- Display all processes currently running on the Raspberry Pi hardware

# R2020b

---

**Version: 9.9**

**New Features**

**Bug Fixes**

**Compatibility Considerations**

## Environment

### **MATLAB Online Accessibility: Use a screen reader to interact with the Command Window and create scripts and functions**

In MATLAB Online, you can use a screen reader to interact with the Command Window, create and edit scripts and functions in the Editor, and navigate through the MATLAB desktop tools. Using a screen reader is not supported in the Live Editor.

For more information, see [Use a Screen Reader in MATLAB Online](#).

### **Live Editor Images: Add alternative text to images**

You can add alternative text to an image in a live script or function to make it accessible to individuals using a screen reader. To add alternative text, right-click the image and select **Edit Image...** from the context menu. Add text to the **Alt Text** edit field.

### **Live Editor Images: Change the size of images**

To change the size of an image in a live script or function, right-click the image and select **Edit Image...** from the context menu. Then, to specify a size relative to the original image size, select **Relative (%)** and specify the width and height of the image as a percentage of the original image. To specify an absolute size, select **Absolute (px)** and specify the width and height of the image in pixels. To return to the original image size, right-click the image and select **Reset Image**.

### **Live Editor Hyperlinks: Navigate to existing files from a live script or live function using links**

Use hyperlinks to navigate to existing files from a live script or live function. To insert a hyperlink to an existing file, select the text to link in the current file, go to the **Insert** tab, and click  **Hyperlink**. Edit your display text (optional), select **Existing File**, and then enter or browse for the file path.

For more information about adding hyperlinks into live scripts and live functions, see [Format Files in the Live Editor](#).

### **Live Editor Export: Export all live scripts and live functions in a folder to a standard format**

You can export all of the live scripts and live functions in a folder to a standard format. Available formats include PDF, Microsoft Word, HTML, and LaTeX. To export the contents of a folder, on the **Live Editor** tab, select **Save > Export Folder...** Then, specify the export format, the location of the folder to be exported, and the location for the exported files.



For more information about exporting live scripts and live functions, see [Share Live Scripts and Functions](#).

## matlabRelease Object: Query MATLAB Release Information

`matlabRelease` is a new object that makes it possible to quickly query the MATLAB information of your system. `isMATLABReleaseOlderThan` will allow you to determine if your current MATLAB release is older than a specified release. Use these functions to quickly verify release versions.

## Query Parallel Functionality: Determine if support for Parallel Computing Toolbox functionality is available

You can now query if support for GPU and parallel pool functionality is available in your MATLAB installation using the following functions:

- `canUseGPU`
- `canUseParallelPool`

Use these functions to check supported functionality and avoid executing code that relies on specific hardware constraints.

## Comparison Tool: Compare text files in MATLAB Online

Starting in R2020b, you can compare text files in MATLAB Online using the `visdiff` function. To compare text files, in the Command Window, type:

```
visdiff('filename1.m','filename2.m');
```

## MATLAB Drive: Decreased password confirmation requests in MATLAB Drive Connector

Decreased password confirmation requests in MATLAB Drive Connector.

## **MATLAB Drive: Faster shut down of MATLAB Drive Connector when syncing is in progress**

When syncing is in progress, MATLAB Drive Connector shuts down faster than in previous versions.

## **MATLAB Drive: Integration with OS-specific password management systems**

MATLAB Drive Connector now integrates with OS-specific password management systems.

## **MATLAB Drive: MATLAB Drive Connector available in Japanese**

MATLAB Drive Connector is now available in Japanese.

## **MATLAB Drive: Download folders from MATLAB Drive online to your desktop (December 2020)**

You can now download folders in addition to downloading individual files from MATLAB Drive online. To download a folder, in MATLAB Drive online, click the **Download** button or right-click the folder and select **Download**.

For more information, see “Access Files in Your MATLAB Drive”

## **MATLAB Drive: Preview the contents of a shared folder in MATLAB Drive online before accepting invitation (December 2020)**

When you receive an invitation or a link to a shared folder, you can now navigate the structure of the shared folder and preview individual files in MATLAB Drive online before accepting the invitation.

For more information, see “Share Folders Using MATLAB Drive”.

## Language and Programming

### pattern Object and Functions: Match patterns in text functions

Patterns are an intuitive alternative to regular expressions for matching patterns in text. Pattern functions can be combined together in expressions in order to build complex patterns that can then be used as inputs for text-searching functions. For instance, to define a pattern for MATLAB release names, which start with "R", followed by the four-digit year, and then either "a" or "b":

```
pat = "R" + digitsPattern(4) + ("a"|"b");
```

Match that pattern in a string:

```
str = "String arrays were introduced in R2016b. Patterns were added in R2020b.";
extract(str,pat)
```

```
ans =
    2x1 string array
    "R2016b"
    "R2020b"
```

For more information, see `pattern`.

### extract Function: Extract substrings from strings

To extract substrings from strings, use the `extract` function. You can specify the substring to be extracted as literal text or by using a pattern object to match the text in a substring.

### Functions: New validation functions for arguments and properties

The following functions are designed for use in function argument and property validation.

- `mustBeA` Validate that value comes from one of specified classes
- `mustBeNonmissing` Validate that value is not missing
- `mustBeFloat` Validate that value is floating-point array
- `mustBeScalarOrEmpty` Validate that value is scalar or empty
- `mustBeVector` Validate that value is vector
- `mustBeInRange` Validate that value is in the specified range
- `mustBeFile` Validate that path refers to file
- `mustBeFolder` Validate that input path refers to folder
- `mustBeValidVariableName` Validate that input name is valid variable name
- `mustBeText` Validate that value is a string array, character vector, or cell array of character vectors
- `mustBeTextScalar` Validate that value is a single piece of text
- `mustBeNonzeroLengthText` Validate that value is string array, character vector, or cell array of character vectors that has non-zero length

## **underlyingType, isUnderlyingType, and mustBeUnderlyingType Functions: Query the underlying data type of classes**

The `class` function is useful to determine the class of a variable. However, some classes in MATLAB can contain underlying data that has a different type compared to what `class` returns. Example classes include `gpuArray`, `dlarray`, and distributed arrays. The `underlyingType`, `isUnderlyingType`, and `mustBeUnderlyingType` functions now provide a simple way to query the underlying data types of those classes.

For most classes, `class(X)` and `underlyingType(X)` return the same answer. However, for classes that can contain underlying data of a different type, the return values are different. For a `gpuArray X` that contains data of type `double`, for example, `class(X)` returns `'gpuArray'`, whereas `underlyingType(X)` returns the underlying MATLAB data type, `'double'`.

## **height and width Functions: Return number of rows or columns in an array**

The `height` and `width` functions now work with arrays in addition to tables. `height` returns the number of rows in the array and `width` returns the number of columns.

## **⚠ Class conversions: Assignment operations convert more classes into built-in data types**

In assignment statements such as `A(1:k) = C`, where `A` has a built-in data type such as `double`, MATLAB attempts to convert `C` to be the same data type as `A` using a series of conversions. That conversion behavior has changed.

- Previously, only one conversion was attempted. This conversion attempted to use a class conversion function, such as `logical(C)` or `double(C)`, to obtain a variable with the same data type as `A`.
- Now, if the first conversion fails, or if it returns a type that differs from `A`, then MATLAB attempts a secondary conversion with `cast(C, "like", A)`. If `C` belongs to a class that defines a `cast` method that supports the "like" flag, and that method returns a value with the same type as `A`, then MATLAB uses the resulting value to perform the assignment into `A`.

If both conversion attempts fail, then MATLAB throws a `MATLAB:invalidConversion` error.

An example of code that used to error, but now executes, is

```
a = 1:6;
g = distributed([11 12 13]);
a(1:3) = g

a =

    11    12    13     4     5     6
```

## **⚠ Compatibility Considerations**

Some assignment statements that used to throw an error now execute. If your code relied on the errors that MATLAB threw for those conversions, such as within a `try/catch` block, then your code might no longer catch those errors.



## ▲ Functionality being removed or changed

### **hex2dec, bin2dec, and base2dec Functions: Issue warning when inputs are greater than or equal to flintmax**

*Behavior change*

Starting in R2020b, the `hex2dec`, `bin2dec`, and `base2dec` functions issue a warning when their inputs have values greater than or equal to `flintmax`. In previous releases, these functions did not issue a warning when their inputs had such values.

These functions now issue a warning because inputs representing integers greater than or equal to `flintmax` might not be represented exactly as double-precision floating-point values.

To convert values greater than `flintmax` exactly, consider one of these alternatives:

- In place of scalar text inputs, use hexadecimal or binary literals representing the same values. When you write a value as a literal, MATLAB stores it as an integer that represents the value exactly. For more information, see [Hexadecimal and Binary Values](#).
- To convert hexadecimal inputs greater than `flintmax`, you can use the `sscanf` function with the `%lx` operator. When you use `%lx`, the converted values are integers that have the `uint64` data type. These integers have enough storage to represent values greater than `flintmax` exactly.

### **Defining classes and packages: Using `schema.m` will not be supported in a future release**

*Still runs*

Support for classes and packages defined using `schema.m` files will be removed in a future release. Replace existing schema-based classes with classes defined using the `classdef` keyword.

## Data Analysis

### **Implicit Expansion: For calendarDuration, categorical, datetime, and duration arrays, automatically expand dimensions of length 1 when applying element-wise operations and functions**

calendarDuration, categorical, datetime, and duration arrays now support implicit expansion for certain operations and functions. In previous releases, these operations and functions only supported implicit expansion for numeric and logical arrays.

Implicit expansion is a generalization of scalar expansion. With scalar expansion, a scalar expands to be the same size as another array to facilitate element-wise operations. With implicit expansion, the element-wise operators and functions listed in the table can implicitly expand their inputs to be the same size, as long as the arrays have compatible sizes. Two arrays have compatible sizes if, for every dimension, the dimension sizes of the inputs are either the same or one of them is 1. See Compatible Array Sizes for Basic Operations and Array vs. Matrix Operations for more information.

For each operator or function, the table shows the data types that gained support for implicit expansion in R2020b.

| Operator or Function | Data Types with New Support for Implicit Expansion in R2020b |
|----------------------|--------------------------------------------------------------|
| +                    | calendarDuration, datetime, and duration                     |
| -                    |                                                              |
| .*                   | calendarDuration, categorical, and duration                  |
| ./                   | duration                                                     |
| .\                   |                                                              |
| .^                   | None                                                         |
| ==                   | categorical, datetime, and duration                          |
| ~=                   |                                                              |
| <                    | categorical (ordinal arrays only), datetime, and duration    |
| <=                   |                                                              |
| >                    |                                                              |
| >=                   |                                                              |
| max                  | categorical (ordinal arrays only), datetime, and duration    |
| min                  |                                                              |
| mod                  | duration                                                     |
| rem                  |                                                              |
| isbetween            | datetime and duration                                        |

## ⚠ Compatibility Considerations

If your code uses any of the element-wise operators or functions listed above and relies on the errors that MATLAB returns from mismatched sizes, particularly within a `try/catch` block, then your code might no longer catch those errors. This change is because some combinations of input sizes that previously caused errors are now valid.

For example, in older releases of MATLAB, you could not add a row and a column vector that are `duration` vectors, but those operands are now valid for addition with implicit expansion. In other words, an expression like `seconds([1 2]) + seconds([1; 2])` previously returned a size mismatch error, but now it executes.

### normalize Function: Scale data by interquartile range

You can use the `'medianiqr'` method with the `normalize` function to center data on a median of 0 and scale it to have an interquartile range of 1. Additionally, when using the `'scale'` method, the new `'iqr'` option allows you to scale the data by the interquartile range.

### groupsummary Function: Summarize data using functions that require multiple input arguments

When specifying a summary computation, `groupsummary` now supports function handles and anonymous functions that require more than one input argument.

### fillmissing Function and Clean Missing Data Live Editor Task: Specify maximum gap size to fill

When filling missing data using the `fillmissing` function, you can use the `'MaxGap'` name-value pair to fill only gaps of missing data up to a specified size. When using the Clean Missing Data Live Editor task, enter a gap size next to **Max gap to fill**.

### Clean Outlier Data Live Editor Task: Define outliers based on percentile thresholds

The Clean Outlier Data Live Editor task now offers the option to define outliers in data as points outside of a percentile range. To use this option, select **Percentiles** as the **Detection method**, and specify values for the **Lower threshold** and **Upper threshold**.

## ⚠ Functionality being removed or changed

### table and timetable Data Types: Change to dimension name selection when combining or joining tables and timetables

*Behavior change*

In R2020b, when you combine tables and timetables, the dimension names of the output table come from the first nondefault dimension names in the input table and timetables. In previous releases, the dimension names always came from the first input, even when that table or timetable had default dimension names.

For example, if TT1 and TT2 are timetables, and TT2 has the nondefault name 'Start' as the name of its first dimension, then when you concatenate the timetables the output also has 'Start' as the name of its first dimension. The name of the first dimension of TT1 is 'Time', but that is a default dimension name. In previous releases, the name of the first dimension of the output was 'Time' because that name was in the first input timetable.

```
TT1 = timetable(seconds(1:2)',[1;2]);
TT2 = timetable(seconds(1:2)',[3;4]);
TT2.Properties.DimensionNames{1} = 'Start';
TT = [TT1 ; TT2]
```

TT =

4×1 timetable

| Start | Var1 |
|-------|------|
| 1 sec | 1    |
| 2 sec | 2    |
| 1 sec | 3    |
| 2 sec | 4    |

This change in behavior affects the functions in this list.

- `cat`
- `horzcat`
- `innerjoin`
- `outerjoin`
- `setxor`
- `union`
- `vertcat`

### **Retime Timetable Live Editor Task: Use linear interpolation as default general rule for adjusting data**

#### *Behavior change*

In R2020b, linear interpolation is the default general rule for adjusting data in the Retime Timetable live editor task. In previous releases, the default general rule is to fill gaps in the output timetable with missing values.

However, even in R2020b the default rule reverts to filling gaps with missing values if either of these conditions is met:

- The row times of the input timetable are not sorted.
- The input timetable has at least one variable whose data type is not numeric, `datetime`, or `duration`.

## Data Import and Export

### **readstruct and writestruct functions: Read and write structured data in XML files**

Read and write structured data stored in XML files using the `readstruct` and `writestruct` functions:

- Use the `readstruct` function to read structured data stored as XML files.
- Use the `writestruct` function to write MATLAB structures to XML files.

### **readlines function: Read the lines in a text file as a string array**

Use the `readlines` function to read each line in a text file as an N-by-1 string array, where N is the number of lines in the text file. For more information, see `readlines`.

### **Spreadsheet files: Customize formatting when writing data to spreadsheet files with `PreserveFormat` and `AutoFitWidth`**

The `writecell`, `writetable`, `writematrix`, and `writetimetable` functions have two new name-value pairs that enable you to format spreadsheet files: `'PreserveFormat'` and `'AutoFitWidth'`.

- `'PreserveFormat'` specifies whether to preserve the existing cell formatting of the original data.
- `'AutoFitWidth'` specifies whether to automatically adjust the column width of the spreadsheet file to fit the data.

### **imread function and Tiff object: Read images from Aperio SVS and TIFF files containing JPEG2000 compression**

The `imread` function and the Tiff object can now read Aperio SVS microscopy image files and TIFF image files with JPEG2000 compression.

### **ArrayDatastore object: Create datastores from in-memory data**

The `arrayDatastore` function creates a datastore from in-memory data. You can combine `ArrayDatastore` objects with datastores that contain on-disk data (such as `ImageDatastore` and `TabularTextDatastore` objects) using the `combine` and `transform` functions. To choose the ways that an `ArrayDatastore` object reads and returns data, specify these properties: `'IterationDimension'`, `'ReadSize'`, and `'OutputType'`. For more information, see `arrayDatastore`.

### **Datastore: Transform multiple datastores using the `transform` function**

The `transform` function now accepts multiple datastores as input. Specify each datastore as another input argument.

```
ds = transform(@fc, ds1, ds2, ds3);
```

The `transform` function creates one `TransformedDatastore` object from the resulting transformation. For more information on the `TransformedDatastore` object, see `TransformedDatastore`.

## **FileDatastore object: Shuffle and create subsets of a FileDatastore**

You can use the `shuffle` and `subset` functions to shuffle and create subsets of a `FileDatastore` object. You must use the `'ReadMode', 'file'` name-value pair argument to use `subset` and `shuffle` on a `FileDatastore` object.

## **writeall function: Write data from text and spreadsheet files to different row groups in Parquet files**

In R2020b, when writing to parquet files using the `writeall` function, you do not need to set the `ReadSize` property of `TabularTextDatastore`, `SpreadsheetDatastore`, or `ParquetDatastore` objects to `'file'`. `writeall` will write the amount of data specified by the `ReadSize` property of the datastore to a separate row group in the Parquet file.

## **fileparts function: Parse file names specified as cell arrays of character vectors and string arrays**

The `fileparts` function now accepts cell arrays of character vectors and string arrays as the input file names.

## **Audio devices: Refresh the available audio devices using the audiodevreset function**

You can use the `audiodevreset` function to refresh the available audio devices after adding or removing a device from your machine. Use the `audiodevinfo` function to view the updated list of audio devices.

## **Audio files and web-based data: Read and write remotely stored audio files using audioread, audiowrite, and audiointro**

You can access audio files stored in remote locations, such as Amazon S3, Windows Azure Blob Service, and HDFS, using the `audioread`, `audiowrite`, and `audiointro` functions.

When reading data from a remote location, you must specify the full path using a uniform resource locator (URL). For example, read a `.wav` file from the Amazon S3 cloud.

```
audio = audioread('s3://bucketname/path_to_file/sample_audio.wav');
```

For more information on setting up MATLAB to access your online storage service, see [Work with Remote Data](#).

## HDF5 files and web-based data: Read and write remotely-stored HDF5 files using existing HDF5 functions

MATLAB HDF5 functions can now work with remote files. Use the existing high-level and low-level functions to read and write files stored in remote locations, such as Amazon S3, Windows Azure Blob Service, and HDFS.

When reading data from a remote location, you must specify the full path using a uniform resource locator (URL). For example, display the metadata of an HDF5 file from the Amazon S3 cloud.

```
h5disp('s3://bucketname/path_to_file/my_file.h5');
```

For more information on setting up MATLAB to access your online storage service, see [Work with Remote Data](#).

## HDF5 files: Read and write file names encoded using Unicode characters

You can now read and write file names encoded as Unicode characters using the high-level and low-level HDF5 functions in MATLAB.

## Scientific File Format Libraries: NetCDF library upgraded to 4.7.3

The NetCDF library is upgraded to version 4.7.3.

## Image File Format Libraries: LibTIFF library upgraded to version 4.1.0

The LibTIFF library is upgraded to version 4.1.0.

## Bluetooth Interface: Support for communicating with Bluetooth devices

You can now use MATLAB to connect to Bluetooth devices and transmit and receive ASCII and binary data.

Get started with the new interface by viewing a list of all Bluetooth devices on your computer using `bluetoothlist`. In this example, an HC-06 Bluetooth module is paired to the computer.

```
list = bluetoothlist
```

```
list=1x4 table
      Name      Address      Channel      Status
    _____  _____  _____  _____
    "HC-06"    "98D331FB3B77"    1      "Ready to connect"
```

Then, create a `bluetooth` object, write data to the device, and read from it. In this example, the HC-06 is configured as a loopback device.

```
b = bluetooth("HC-06");
write(b,1:10);
read(b,10);
```

For more information, see [Bluetooth Communication](#).

## TCP/IP Client Interface: New functions and properties

The TCP/IP client interface has a new set of functions and properties.

You can still perform the following operations using existing functions:

- Create a TCP/IP client connection with a TCP/IP server using the `tcpclient` function.
- Read data from a remote host using the `read` function.
- Write data to a remote host using the `write` function.

You can now perform the following operations using new functions:

- Start an echo TCP/IP server using the `echotcpip` function.
- Read a line of ASCII string data from a remote host using the `readline` function.
- Write a line of ASCII string data to a remote host using the `writeline` function.
- Set a terminator for ASCII string communication with a remote host using the `configureTerminator` function.
- Set a callback function and trigger condition for communication with a remote host using the `configureCallback` function.
- Flush buffers for communication with a remote host using the `flush` function.

Get started with the TCP/IP client interface by creating a `tcpclient` object connected to a TCP/IP echo server, writing data to it, and reading from it.

```
echotcpip("on",3030)
t = tcpclient("localhost",3030)
write(t,1:5,"uint8")
read(t,5);
```

For more information, see TCP/IP Communication.

## Serial Port Interface: Improved performance

The `serialport` interface shows improved performance over the `serial` interface. For example, this code for writing and reading data with the `serialport` object is about 1.1x faster than the code for writing and reading data with the `serial` object with the default baud rate of 9600.

```
% s is a serial object
function timingTest(s,bytecount)
fwrite(s,1:bytecount,"uint8");
fread(s,bytecount,"uint8");
end

% s is a serialport object
function timingTest(s,bytecount)
write(s,1:bytecount,"uint8");
read(s,bytecount,"uint8");
end
```

The approximate execution times for different baud rates follow:

|  |            |
|--|------------|
|  | s.BaudRate |
|--|------------|



|            | 9600   | 19200 | 56000 | 115200 |
|------------|--------|-------|-------|--------|
| serial     | 120 ms | 68 ms | 31 ms | 23 ms  |
| serialport | 109 ms | 55 ms | 21 ms | 11 ms  |

The code was timed on a Windows 10, Intel(R) Xeon(R) CPU E5-1650 v4 @ 3.60 GHz test system using the `timeit` function:

```
bytecount = 100;
timeit(@()timingTest(s,bytecount))
```

The tests were done using a serial loopback connector.

For more information, see [Serial Port Devices](#).

## ⚠️ Functionality being removed or changed

### The 'PreserveVariableNames' property of `readtable` is no longer recommended

*Behavior change*

The 'PreserveVariableNames' property of `readtable` has been removed. Use the 'VariableNamingRule' name-value pair instead. Specify one of the following values:

- 'preserve' preserves variable names that are not valid MATLAB identifiers, such as variable names that include spaces and non-ASCII characters.
- 'modify' converts invalid variable names to valid MATLAB identifiers.

### The `UnderlyingDatastore` property of `TransformedDatastore` is no longer recommended

*Behavior change*

The `UnderlyingDatastore` property of `TransformedDatastore` has been removed. Use the `UnderlyingDatastores` property instead. Calling the `UnderlyingDatastores` property returns a cell array containing zero or more datastores. For more information, see [TransformedDatastore](#).

### The `partition` function adds remaining observations to the first existing partitions

*Behavior change*

If you specify a number of partitions in the `partition` function that is not a numerical factor of the number of files in the datastore, the `partition` function will place each of the remaining observations in the existing partitions, starting with the first partition.

The number of existing partitions that contain an additional observation is equal to the remainder obtained when dividing the number of files in the datastore by the number of partitions. For example, if your datastore object contains 23 files that you wish to partition into 3 parts, the first two partitions that `partition` creates will contain 8 files, and the last partition will contain 7 files.

### Server certificate and hostname validation are enabled by default when accessing OPeNDAP servers with the NetCDF interface

*Behavior change*

In R2020b, the MATLAB NetCDF interface connects only to trusted data access protocol (DAP) endpoints by default. Previously, when you accessed an OPeNDAP server, both the server certificate and hostname validation were disabled by default.

To learn how to disable server certificate and hostname validation when accessing an OPeNDAP server, see [Import NetCDF Files and OPeNDAP Data](#).

### **isSingleReadPerFile has been removed**

The `isSingleReadPerFile` function of the `matlab.io.datastore.FileWritable` class has been removed.

## Mathematics

### Optimize Live Editor Task: Solve optimization problems interactively

The Optimize Live Editor task lets you optimize multivariable functions or solve scalar equations interactively. The task provides a visual way to access the `fminbnd`, `fminsearch`, `fzero`, and `lsqnonneg` solvers and set their options. For an example, see [Optimize Live Editor Task](#).

### pagetimes Function: Perform matrix multiplication on pages of N-D arrays

Use the `pagetimes` function to perform batched matrix multiplication on the pages of N-D arrays. In this context, the N-D array is treated as a container for several 2-D matrices.

### pagetranspose and pagectranspose Functions: Transpose pages of N-D arrays

Use the `pagetranspose` and `pagectranspose` functions to transpose the pages of an N-D array. In this context, the N-D array is treated as a container for several 2-D matrices.

### svdsketch Function: Compute SVD factors of low-rank matrix sketch

`svdsketch` computes the singular value decomposition (SVD) factors of a low-rank sketch of the input matrix. This operation preserves the most important features of the matrix based on a specified tolerance. For large matrices, `svdsketch` can typically compute an approximate SVD faster than `svds`.

### ⚠️ Functionality being removed or changed

#### 'cubic' method of `interp1` performs cubic convolution

*Behavior change*

In R2020b, the 'cubic' interpolation method of `interp1` performs cubic convolution. The 'v5cubic' and 'cubic' interpolation methods now perform the same type of interpolation, which is consistent with the behavior of `interp2`, `interp3`, and `interp4`. The cubic convolution interpolation method is intended for uniformly-spaced data, and it falls back to 'spline' interpolation for irregularly-spaced data.

In previous releases, 'cubic' was the same as 'pchip', and only 'v5cubic' performed cubic convolution.

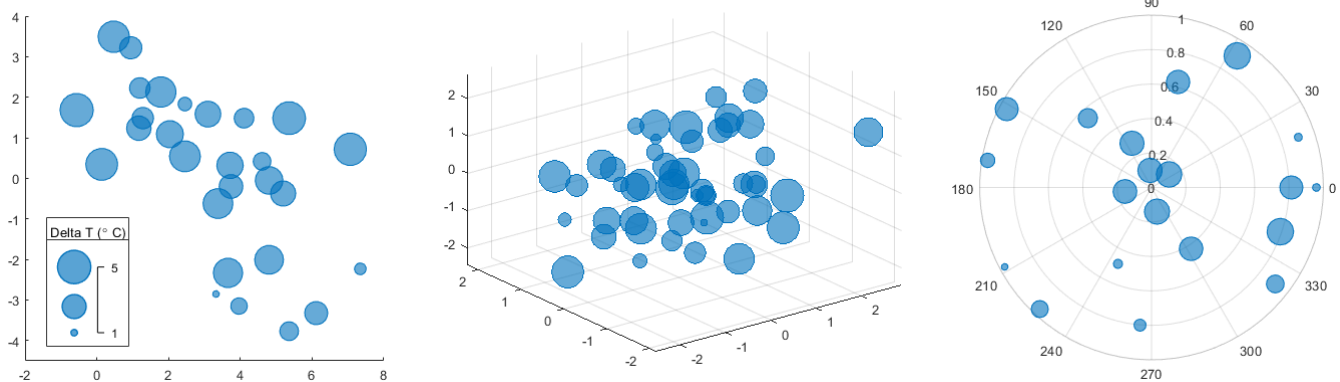
## Graphics

### **bubblechart, bubblechart3, and polarbubblechart Functions: Create bubble charts in 2-D, 3-D, and in polar coordinates**

Use the `bubblechart` and `bubblechart3` functions to create bubble charts in 2-D and 3-D Cartesian spaces, respectively. Use the `polarbubblechart` function to create bubble charts in polar coordinates.

Use the following functions to customize different aspects of your bubble charts:

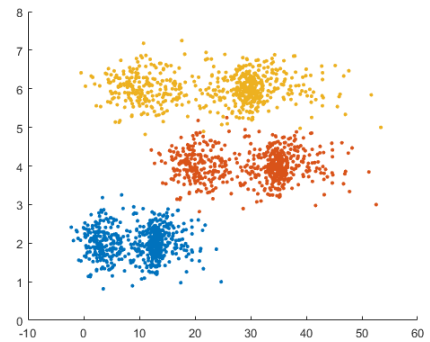
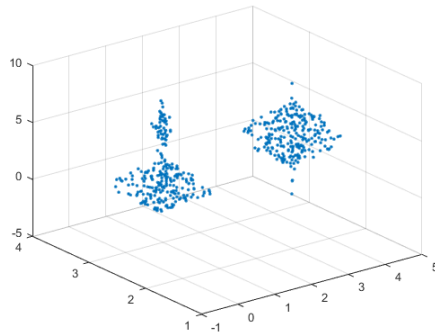
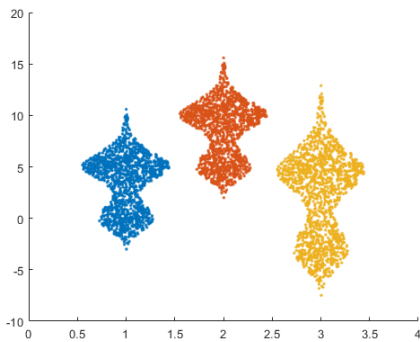
- `bubblelegend` function — Illustrate the correspondence between the bubble sizes and the data values.
- `bubblelim` function — Control the correspondence between the relative bubble sizes and the data values.
- `bubblesize` — Control the absolute bubble sizes in points, where one point equals 1/72 inch.



### **Swarm charts and Scatter objects: Visualize distributions of discrete data**

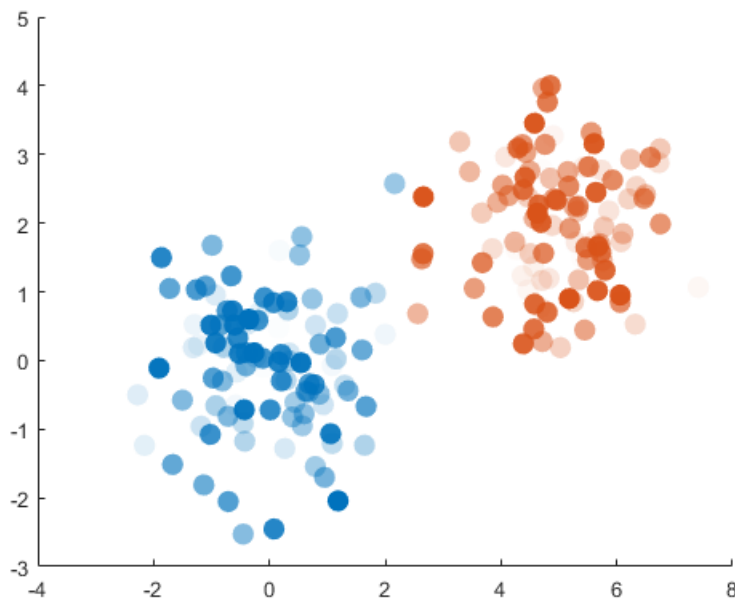
Use the `swarmchart` function to visualize distributions of scattered points at discrete values of  $X$ . Use the `swarmchart3` function to visualize distributions for combinations of discrete  $X$  and  $Y$ . Both functions return `Scatter` objects that have these new properties:

- `XJitter`, `YJitter`, and `ZJitter` — Control the algorithm for spreading (jittering) the points along the  $x$ -,  $y$ -, or  $z$ -dimensions.
- `XJitterWidth`, `YJitterWidth`, and `ZJitterWidth` — Control the maximum amount of jitter.



## scatter Function: Vary the transparency across all points

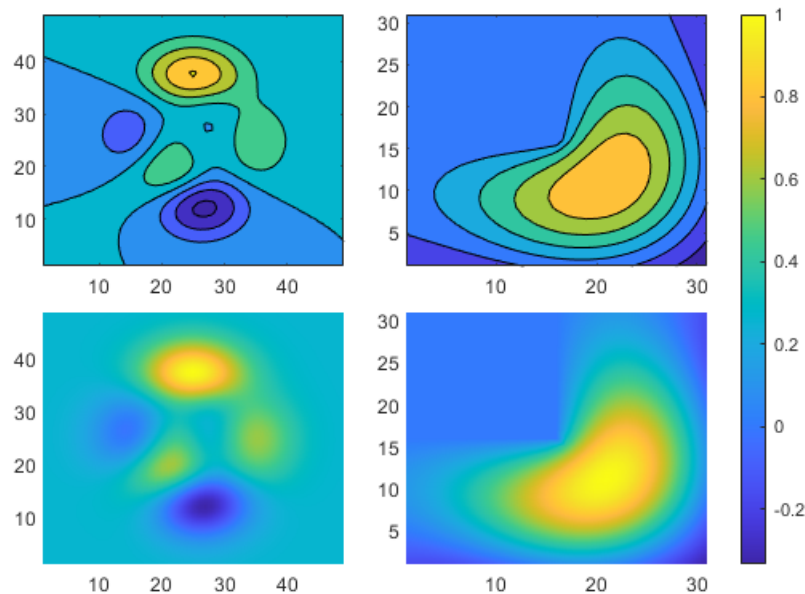
When you create a scatter plot using the `scatter` function, you can vary the level of transparency across the points by setting the `AlphaData` and `AlphaDataMapping` properties of the `Scatter` object.



## tiledlayout and nexttile Functions: Improved placement of legends, and colorbars, and shared decorations

### Improved Legend and Colorbar Placement

- You can now place shared legends, shared colorbars, or additional axes into any of four outer tiles around the perimeter of a tiled chart layout. For more information, see `nexttile`.



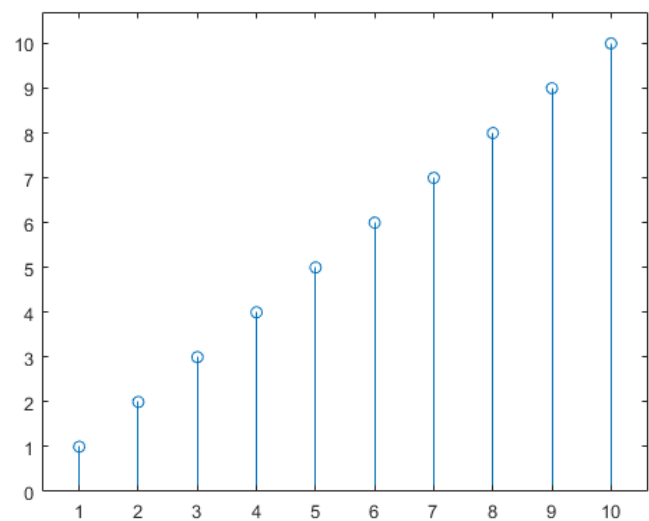
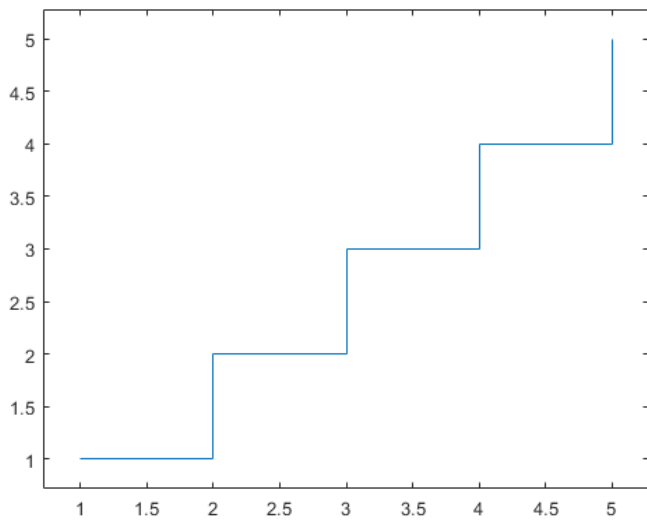
- Tiled chart layouts also provide more support for positioning legends. When you add a legend to a plot, the layout provides better support for the 'best', 'bestoutside', and 'none' location options.

#### **More Consistent Layout for Shared Decorations and Polar Axes**

- When you include shared decorations (such as shared titles and colorbars) and manually adjust the axes aspect ratios, the presentation is better and more consistent with other layouts.
- The presentation is also better for layouts containing polar axes with the 'flow' tile arrangement.

#### **axis Function: Pad axis limits to show plotted data near the limits more clearly**

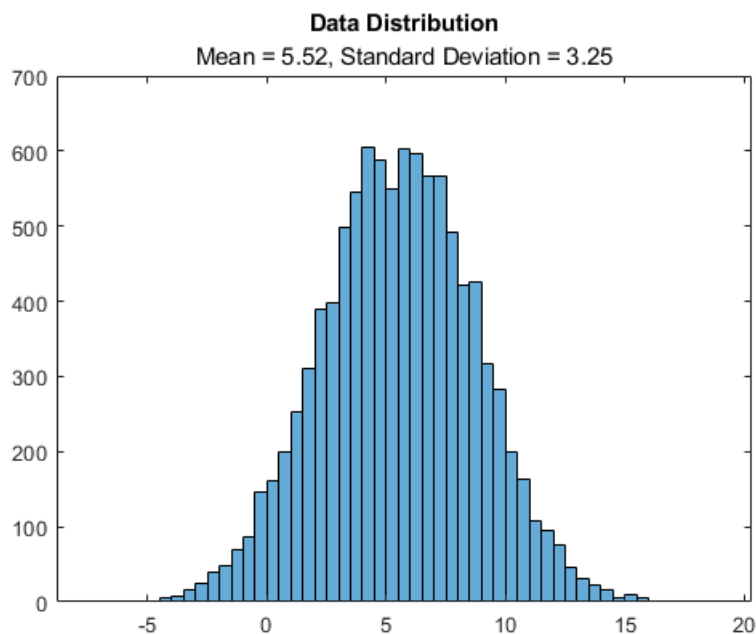
Specify the 'padded' option when calling the `axis` function to pad the upper and lower limits of the x- and y-axis. This option is helpful when your plot displays data near one or more of the axis limits.



## Titles, Subtitles, and Axis Labels: Add subtitles to plots, and align titles and axis labels with the plot box

### Create Subtitles

- Create a title and subtitle for a plot by calling the `title` function with two character vector or string arguments.
- For more control over the appearance of each text object, call the `title` and `subtitle` functions separately.

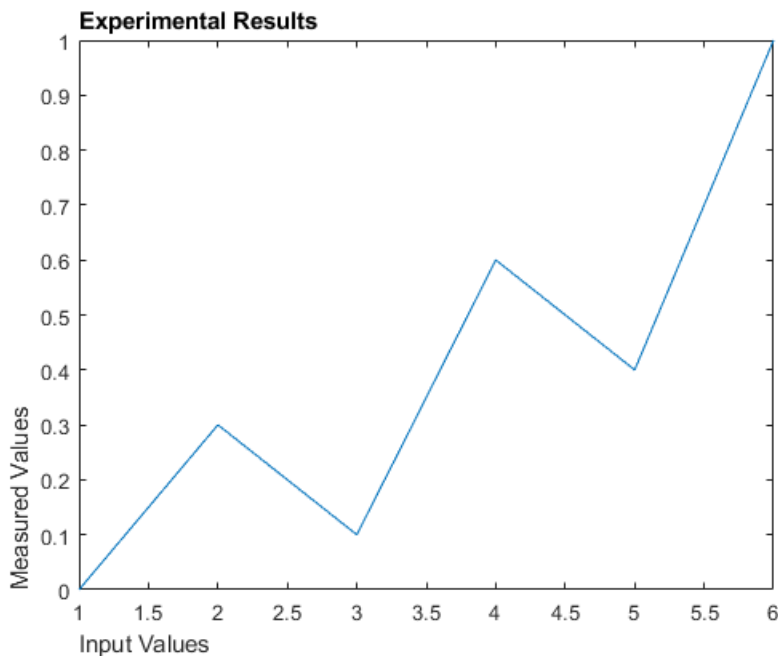


### Align Titles and Axis Labels to the Plot Box

- Align a title by setting the `TitleHorizontalAlignment` property of the axes to `'left'`, `'right'`, or `'center'`.
- Align an axis label by setting the `LabelHorizontalAlignment` property of the ruler object to `'left'`, `'right'`, or `'center'`. Use the `XAxis`, `YAxis`, or `ZAxis` property on the axes object to access the ruler object.

For example, this code creates a plot with a left-aligned title and axis labels.

```
plot([0 3 1 6])
title('Experimental Results')
xlabel('Input Values')
ylabel('Measured Values')
ax = gca;
ax.TitleHorizontalAlignment = 'left';
ax.XAxis.LabelHorizontalAlignment = 'left';
ax.YAxis.LabelHorizontalAlignment = 'left';
```



### Data Tips: Customize data tip content on standalone visualizations

Add or remove rows from data tips on standalone visualizations that you create from tables. You can customize data tips for these standalone visualizations:

- Heatmap charts created using the `heatmap` function
- Scatter plots with marginal histograms created using the `scatterhistogram` function
- Geographic bubble charts created using the `geobubble` function
- Parallel coordinates plots created using the `parallelplot` function

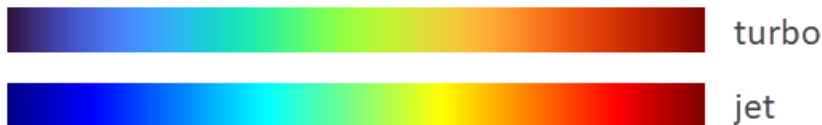
To add or remove rows, right-click on the chart and hover over **Modify Data Tips**. Then, select or deselect table variables.



## turbo Colormap: jet colormap alternative with more perceptually uniform transitions

Use the `turbo` colormap to display visualizations using a colormap that is similar to `jet`, but with more perceptually uniform transitions along the color scale. Perceptually uniform colormaps help you to visualize the differences in your data more accurately.

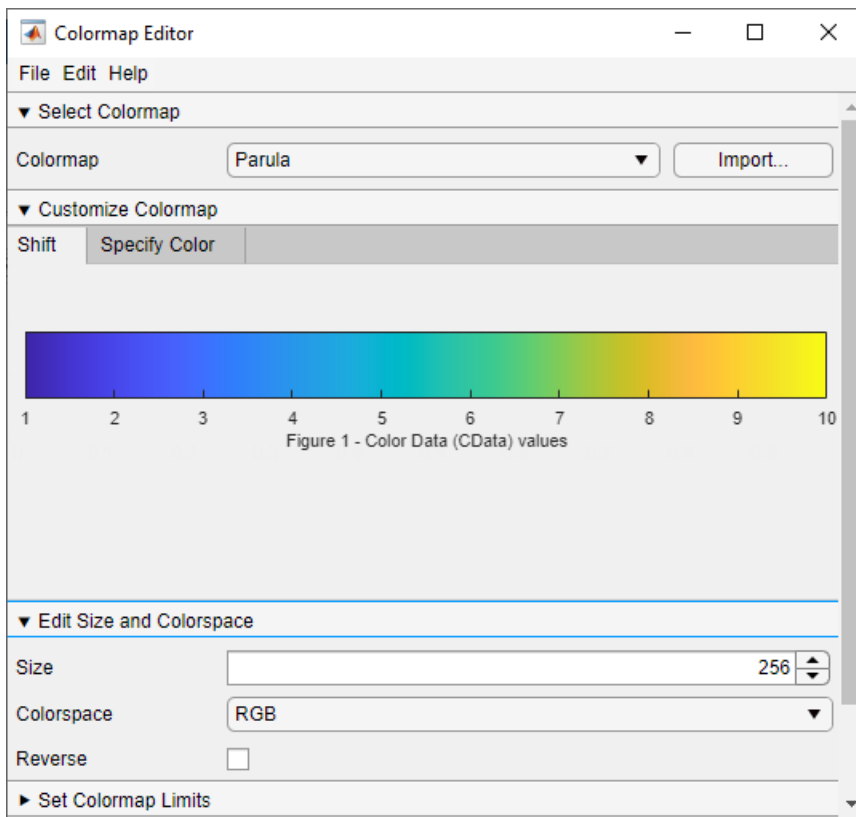
These colorbars show the difference between the `turbo` and `jet` colormaps. The `turbo` colormap is on top, and the `jet` colormap is on the bottom.



## Colormap Editor: Customize colormaps using modernized interface

Customize colormaps using the modernized interface of the Colormap Editor. Starting in R2020b, you can use the Colormap Editor to:

- Import a colormap as an  $m$ -by-3 array from the workspace.
- Save a customized colormap as a workspace variable.
- Shift the placement of colors in the colormap.
- Reverse the order of colors in the colormap.
- Change the number of elements in the colormap.



## boxchart Function: Use color to differentiate between box charts

Use color to differentiate between box charts by specifying the `'GroupByColor'`, `cgroupdata` name-value pair argument of `boxchart`. The function creates a box chart for each group of data and assigns the same color to groups with the same `cgroupdata` value. You can specify the color grouping variable with or without a positional grouping variable. For an example, see [Use Positional and Color Grouping Variables](#).

## im2gray and cmap2gray: Convert images and colormaps to grayscale

Use the `im2gray` function to convert colored images to grayscale. Use the `cmap2gray` function to convert a colormap to grayscale.

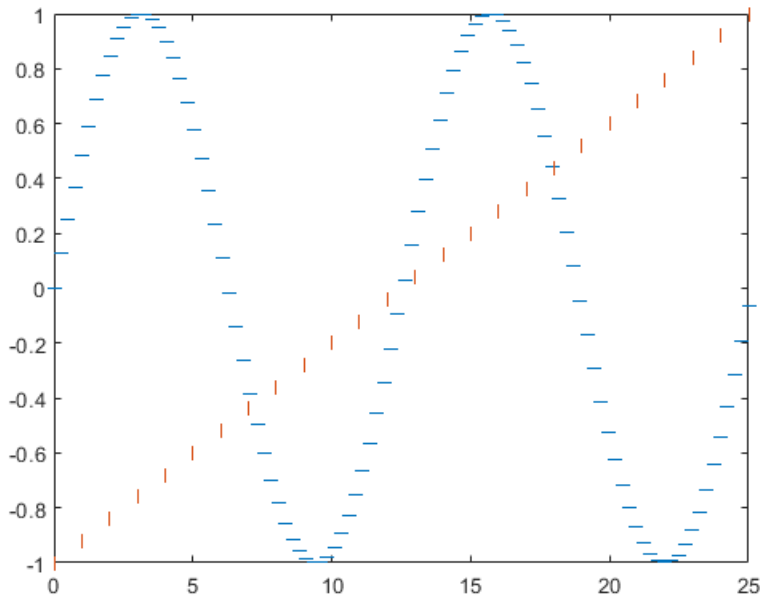
## validatecolor Function: Calculate normalized RGB triplets for color names, hexadecimal color codes, or integer values

Use the `validatecolor` function to calculate the normalized RGB triplet values for colors specified in any of the following ways:

- Color names, such as `'red'` or `'green'`
- Hexadecimal color codes, such as `'#FF0000'` or `'#F00'`
- Integer RGB triplets, such as `uint8([255 0 0])`

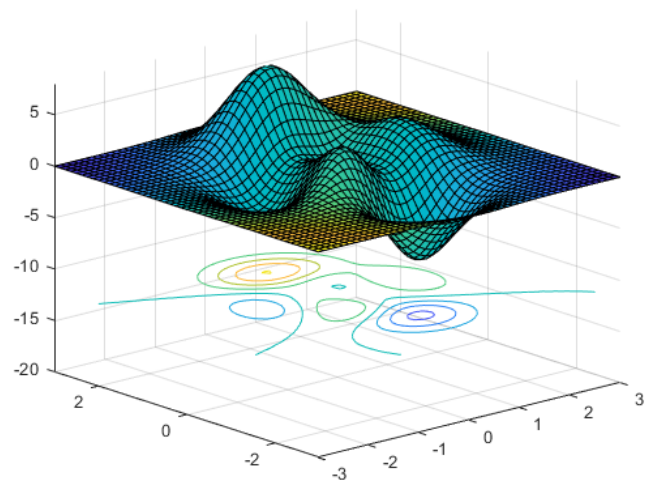
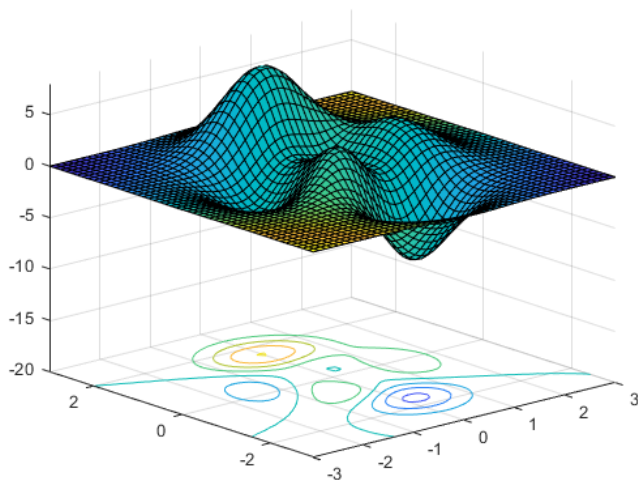
## Markers: Specify horizontal or vertical line markers for plots

Create plots with horizontal or vertical line markers when you call plotting functions such as `plot` or `scatter`. Specify horizontal line markers using the underscore ('\_') symbol. Specify vertical line markers using the pipe ('|') symbol.



## surf and meshc Functions: Specify Z-level for contours on surface and mesh plots

When you create a surface or a mesh plot with contour lines using the `surf` or `meshc` functions, you can display the contours at any z-level by specifying the `ZLocation` property of the `Contour` object. The `ZLocation` property can have a value of 'ZMin', 'ZMax', or a numeric value. The default value is 'ZMin', which corresponds to the lowest level shown in the plot box.



## animatedline Function: Create animated lines in polar plots

Create animated polar plots by calling the `animatedline` function and specifying a `PolarAxes` object as the first input argument.

## colororder Function: Control colors in geographic bubble charts

The `colororder` function now supports charts created with the `geobubble` function.

## ⚠️ Functionality being removed or changed

### Calling the `alpha` function with the `alphadata`, `facealpha`, or `alphadatamapping` arguments changes `Scatter` objects in the axes

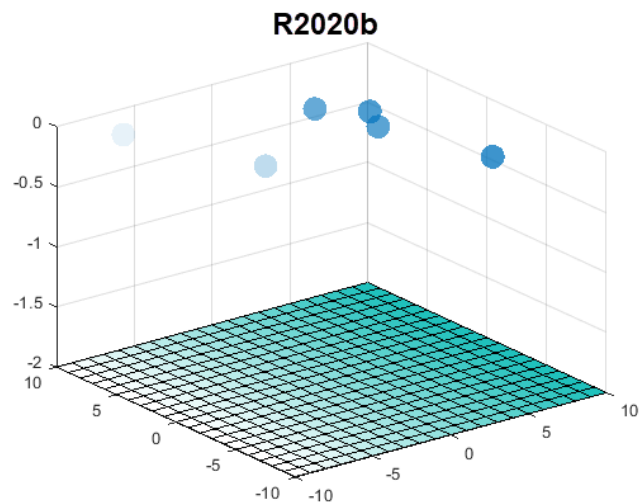
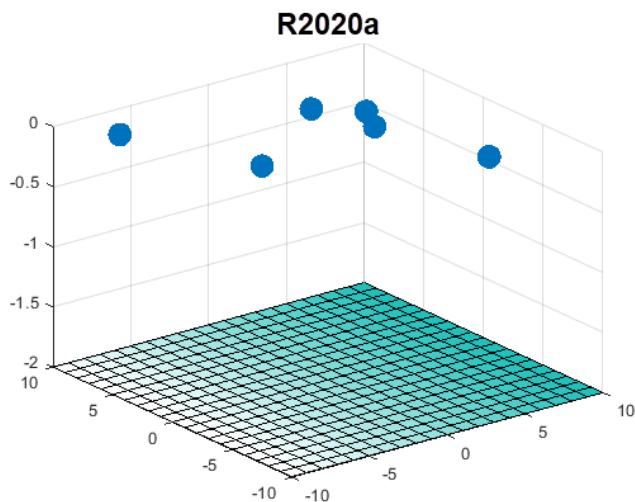
*Behavior change*

`Scatter` objects in the axes are affected when you call the `alpha` function with the `alphadata`, `facealpha`, or `alphadatamapping` arguments without specifying a particular object within the axes to modify.

In R2020a and earlier releases, the `alphadata`, `facealpha`, and `alphadatamapping` arguments have no effect on `Scatter` objects in the axes.

For example, this code creates a surface plot and a scatter plot, and then calls the `alpha` function to vary the transparency along the x-dimension. In R2020a, only the surface plot changes when you call the `alpha` function. In R2020b, both plots are affected.

```
[X,Y] = meshgrid(-10:10);
Z = ones(21,21) * -2;
surf(X,Y,Z)
hold on
scatter([-8 2 4 -5 5 3],[7 4 2 -1 -7 0],200,'filled')
alpha('x')
```



To prevent `Scatter` objects from changing, specify the object to modify as the first argument to the `alpha` function. For example, to update the preceding code, call the `surf` function with an output argument `s`. Then pass `s` to the `alpha` function to modify only the Surface object.

```
[X,Y] = meshgrid(-10:10);
Z = ones(21,21) * -2;
s = surf(X,Y,Z)
hold on
scatter([-8 2 4 -5 5 3],[7 4 2 -1 -7 0],200,'filled')
alpha(s,'x')
```

### **Colormap 'default' option for heatmap displays the blue colormap instead of parula**

#### *Behavior change*

Setting the colormap on a heatmap chart to 'default' sets the chart's colormap to the default blue colormap for heatmap charts. In R2020a and previous releases, the 'default' option changes the colormap to `parula`.

To specify the default colormap for a heatmap chart, pass the chart to the `colormap` function.

```
h = heatmap(rand(10));
colormap(h,'default')
```

Only heatmap charts are affected by this change.

### **rbbox and dragrect do not display rectangles outside of the figure window**

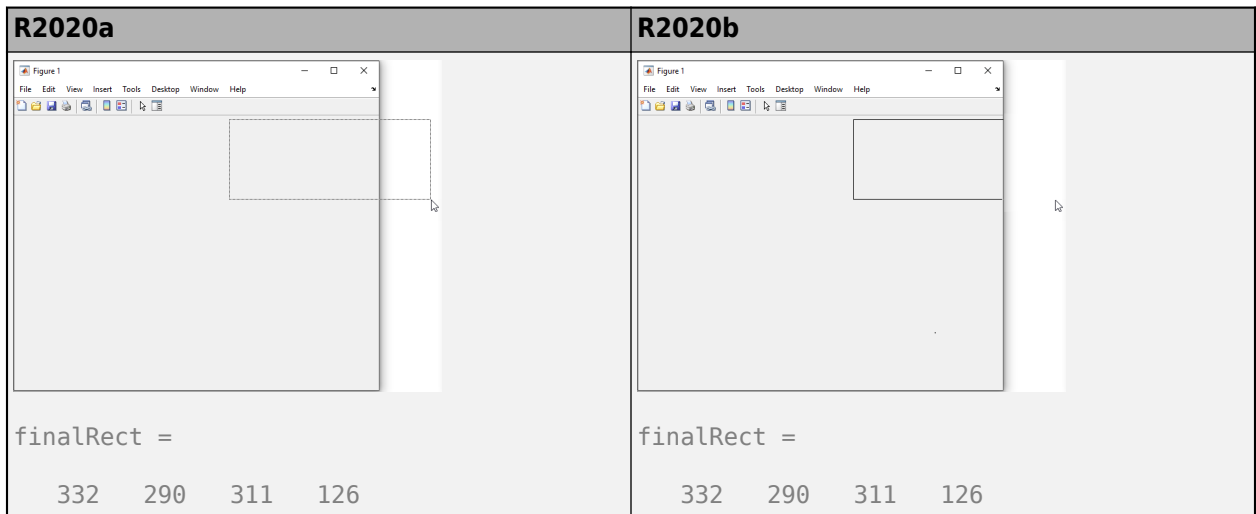
#### *Behavior change*

If you begin dragging a rectangle using the `rbbox` or `dragrect` functions and then move part of it past the edge of the figure window, only the part of the rectangle that is inside the window is visible. In R2020a and earlier releases, the whole rectangle is visible, even the part that is outside of the figure window.

This change does not affect the values returned by `rbbox` or `dragrect`.

This code and table compares the behavior of `rbbox` in R2020a and R2020b.

```
waitforbuttonpress
[finalRect] = rbbox
```



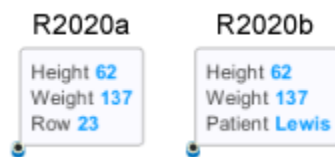
### Data tips on charts created from tables display the first dimension name and row names *Behavior change*

When you create a chart from a table, the data tips include the row information you provide. For instance, if you change the first dimension name of the table, then the data tips display that name instead of **Row**. In addition, if you specify row names for the table, then the data tips display the row names instead of the row numbers.

By default, the first dimension name of table arrays is 'Row' and the row names are empty. This means the behavior in R2020b only differs from earlier releases when you change the `DimensionNames` or `RowNames` properties of the table array.

For example, load the `patients` data set and create a table. Change the dimension names from `{'Row', 'Variables'}` to `{'Patient', 'Data'}` and the row names from `{}` to `LastName`. Then, create a scatter plot with marginal histograms. The image shows the differences in the data tips for R2020a and R2020b.

```
load patients
tbl = table(LastName, Age, Gender, Height, Weight);
tbl.Properties.DimensionNames = {'Patient', 'Data'};
tbl.Properties.RowNames = LastName;
scatterhistogram(tbl, 'Height', 'Weight')
```



The charts affected by this change are:

- Heatmap charts created using the `heatmap` function
- Scatter histogram charts created using the `scatterhistogram` function
- Geographic bubble charts created using the `geobubble` function
- Parallel coordinates plots created using the `parallelplot` function

## App Building

### **uitable Function: Configure column widths to use weighted variable or to automatically adjust to fit data**

Configure the column widths of table UI components. You can use a weighted variable width or you can automatically adjust the width to fit the data and column names.

To specify a weighted variable width, set the `ColumnWidth` property value to a number paired with an 'x' character (for example, '2x'). To configure columns to automatically adjust to column names and data, set the `ColumnWidth` property value to 'fit'.

The 'fit' and 'Nx' `ColumnWidth` property values are supported only in App Designer and `uitable`-based apps.

For more information, see Table Properties.

### **scroll Function: Scroll to the top or bottom of a text area programmatically**

Use the `scroll` function to scroll to the top or bottom of a text area component programmatically.

### **WindowStyle Property: Create modal UI figures**

To restrict keyboard and mouse interactions to a specific UI figure window, set the `WindowStyle` property to 'modal'. For more information see, UI Figure Properties.

### **Icon Property: Specify custom icons for UI figure windows and toolbar push and toggle tools**

You can now add custom icons to figures created with the `uitable` function. You can also specify custom icons as image files for toolbar push and toggle tools in App Designer or `uitable`-based apps. To specify a custom icon, set the `Icon` property value to an image file or an m-by-n-by-3 truecolor array.

For push and toggle tools created in App Designer and `uitable`-based apps, use the `Icon` property to specify icons instead of the `CData` property.

For more information, see UI Figure Properties, PushTool Properties or ToggleTool Properties.

### **WordWrap Property: Wrap long text to fit the width of certain UI components**

Use the `WordWrap` property to prevent text from getting clipped horizontally when the width of your UI component is smaller than the text you want to display. `Label`, `CheckBox`, `RadioButton`, `Button`, `StateButton`, `ToggleButton`, and `TextArea` objects support the `WordWrap` property.

Setting the `WordWrap` property value to `'on'` breaks the text into new lines so that each line fits within the component. It avoids breaking words when possible. When the value is set to `'off'` the text does not wrap.

For more information, see [Label Properties](#).

## **Enable Property: Turn interaction off and on for buttons and panel groups**

To control whether a panel or button group responds to user interaction, use the `Enable` property.

For example, when the `Enable` property of a panel is set to `'on'` you can interact with it, and with UI components within it as long as they are enabled. When the `Enable` property is set to `'off'`, you cannot interact with the panel or its content. Changing the value of the `Enable` property for a panel does not change the value of the `Enable` property for UI components contained within it. The `Enable` property for button groups behaves in the same way.

The `Enable` property is supported only for panels and button groups in App Designer and `uifigure`-based apps.

For more information, see [Panel Properties](#).

## **BackgroundColor Property: Set the background color for grid layouts**

Set the `BackgroundColor` property on `GridLayout` objects (created with the `uigridlayout` function). The default background color is the same as the default color for all parent containers, such as figures and panels.

## **Custom Components: Develop your own class of UI components**

Define your own class of UI components by creating a subclass of the `matlab.ui.componentcontainer.ComponentContainer` base class. Create a class to make composite UI components that use multiple MATLAB UI components or graphics objects.

Creating a class has these benefits:

- A convenient interface — When users want to customize an aspect of your UI component, they can set a property rather than having to modify and rerun your code. Users can modify properties at the command line or inspect them in the Property Inspector.
- Encapsulation — Organizing your code in this way allows you to hide implementation details from your users. You implement methods that perform calculations and manage the underlying graphics objects.

For more information, see [UI Component Development Overview](#).

## **App Designer: Allow only one running instance of your app at a time**

In App Designer, you can select whether your app can run multiple instances at a time or only a single instance.

To change the run behavior of your app, select the App node from the **Component Browser**. Then, from the **Code Options** section of the **Inspector** tab, select or clear **Single Running Instance**.



When **Single Running Instance** is selected and you run the app multiple times, MATLAB reuses the existing instance and brings it to the front rather than creating a new one. When **Single Running Instance** is not selected, then a new app instance is created each time you run it, while existing instances of the app also continue to run.

Selecting or clearing **Single Running Instance** does not change the App Designer run behavior of the app. Regardless of the **Single Running Instance** option, App Designer always closes the existing app instance before creating a new one.

For more information, see [Manage Code in App Designer Code View](#).

## App Designer: Change the stacking order of UI components

To change the stacking order (also known as *z-order*) of components in your app, select a component. Then, in the App Designer toolstrip, expand the **Reorder**  drop-down menu from the **Arrange** section of the **Canvas** tab and select a reorder option. Alternatively, right-click a component and select an option from the **Reorder** menu.

## App Designer: Add and configure toolbar components on the App Designer canvas

Add a toolbar component to your app by dragging one from the **Component Library** onto the canvas. Configure the toolbar by adding push or toggle tools to it. Then, add icons to the tools and configure their callbacks.

## App Designer: Draw UI components on the App Designer canvas

In App Designer, you can now draw UI components from the **Component Library** on the canvas.

To draw a component on the canvas, select it from the **Component Library** and then move your cursor over the canvas. The cursor changes to a crosshair. Click your mouse to add the component to the canvas in its default size, or click and drag to size the component as you add it.

## App Designer: Find differences and merge apps

You can use the Comparison Tool to find differences between apps. The Comparison Tool highlights differences in the code of two apps. To start a comparison of two apps, go to the **Home** tab, and in the **File** section, click **Compare**, and then select the files that you want to compare. To start a comparison from the Current Folder browser, select a file, right-click, and select **Compare Against**.

You also can merge changes in callback and utility functions from one file to the other. Merging changes can be useful when resolving conflicts between different versions of an app. To merge two apps, in the Comparison Tool, select the  **Merge Mode** button to start the merge. Use the  button to replace content in the right pane with content from the left pane. The right pane contains the merged result. To save the result, click the  **Save Result** button.

For more information, see [Compare and Merge Apps](#).

## Graphics Support: Create more plots in apps with full support for any type of axes

All plotting workflows are now supported in App Designer apps and in apps you create programmatically with the `uifigure` function. For example:

- You can add more types of plots to an app, including `pareto`, `plotmatrix`, and `boxplot`.
- You can place any type of axes or standalone visualization (such as a heatmap) directly into a scrollable container or in a `GridLayout`.
- You can place a tiled chart layout directly into a scrollable container or in a `GridLayout`.

Also, `UIAxes` objects now have the same properties and options as `Axes` objects have for customizing their appearance and behavior:

- You can use the `InnerPosition` property to align UI components with the plot box.
- You can use the `PositionConstraint` property to control the space around the plot box when you add or modify decorations such as titles and axis labels.
- The `Units` property of a `UIAxes` object now supports all units: `'pixels'`, `'normalized'`, `'inches'`, `'centimeters'`, `'points'`, and `'characters'`. The default value is `'pixels'`.
- You can use the `ButtonDownFcn`, `PickableParts`, and `HitTest` properties to make the `UIAxes` object capture and respond to mouse clicks.

## Graphics Support: Identify coordinates and display text by clicking or tapping

You can now identify coordinates or display text at specified locations by clicking or tapping within App Designer figures or figures created with the `uifigure` function. To identify axes coordinates, call the `ginput` function. Then, select the locations to identify by clicking or tapping. To display text, call the `gtext` function and specify the text as an argument. Then, select the locations for the text by clicking or tapping. For more information about using the `ginput` and `gtext` functions within App Designer, see [Display Graphics in App Designer](#).

Additionally, you can now migrate GUIDE apps that use `ginput` or `gtext`. For more information, see [GUIDE Migration Strategies](#).

## App Capture: Capture user interfaces using `exportapp` and `getframe`

Capture the all the graphical content in an app, including UI components, as an image file, a PDF file, or an image array in your MATLAB workspace.

- Call the `exportapp` function to capture the content as a JPEG, PNG, or TIFF file, or as a PDF file containing vector graphics.
- Call the `getframe` function with a return argument to capture the content as an image array in your workspace. To ensure all the UI components are included, specify the figure as the first input argument of the `getframe` function. The figure can be one created with the `uifigure` function, or it can be the figure in an App Designer app. By default, App Designer stores the figure in the `UIFigure` property of the app.

## App Testing Framework: Perform choose gestures on context menu items

The `chooseContextMenu` method enables you to test choose gestures on context menu items for UI components. For example, create a context menu with two menu items in a UI figure and choose one of the menu items.

```
fig = uifigure;
cm = uicontextmenu(fig);
m1 = uimenu(cm,'Text','Menu1');
m2 = uimenu(cm,'Text','Menu2');
fig.ContextMenu = cm;

tc = matlab.uiTest.TestCase.forInteractiveUse;
tc.chooseContextMenu(fig,m1)
```

## App Testing Framework: Perform drag gestures on axes and UI axes

The app testing framework supports drag gestures on axes and UI axes. When you test a drag gesture on an axes or UI axes, the framework mimics a user manipulating the component and adjusts the axes limits based on the difference between the start and stop values. For example, create an axes with a plot and then drag from the point (3, 2) to the point (4, 2).

```
f = uifigure;
ax = axes(f);
plot(ax,1:10)

tc = matlab.uiTest.TestCase.forInteractiveUse;
tc.drag(ax,[3 2],[4 2])
```

## App Testing Framework: Perform gestures on push tools and toggle tools

The app testing framework supports gestures on more UI components.

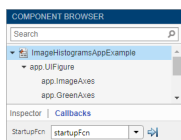
- Perform press gestures in tests on push tools.
- Perform press and choose gestures in tests on toggle tools.

## ⚠️ Functionality Being Removed or Changed

### App Designer `StartupFcn` is now a callback of the app instead of the figure *Behavior change*

The **StartupFcn** callback is now a callback of the app instead of the UI figure. Also, the app is now the root node in the App Designer **Component Browser** hierarchy. The name of the app node is the same as the name of your MLAPP file.

When you load apps created with MATLAB R2020a or earlier into R2020b, the app node is automatically added to the hierarchy. To access the **StartupFcn** callback, select the app node from the **Component Browser**. Then, select the callback from the **Callbacks** tab. The code in your `startupFcn` callback function is not changed.



**App Designer Component Browser hierarchy now uses the z-order of components***Behavior change*

The App Designer **Component Browser** hierarchy now lists components according to their stacking order (z-order) instead of their creation order.

**GridLayout background is no longer transparent***Behavior change*

GridLayout objects now have a `BackgroundColor` property and they are no longer transparent. The default background color is the default color for all containers (for example, figures and panels).

If your app has a grid layout in a container that has a nondefault color, then set the `BackgroundColor` property of the GridLayout object to that color to preserve the appearance of your app.

If your app has objects behind the grid that you want to remain visible, move those objects into the grid by making them children of the GridLayout object.

**BackgroundColor property of UIAxes has no effect***Behavior change*

Setting the `BackgroundColor` property on a UIAxes object no longer has any effect. The area around the plot box is transparent regardless of the value of the `BackgroundColor` property.

To produce the same effect as setting the background color in previous releases, create a panel with the desired `BackgroundColor` value, and then place the UIAxes in the panel.

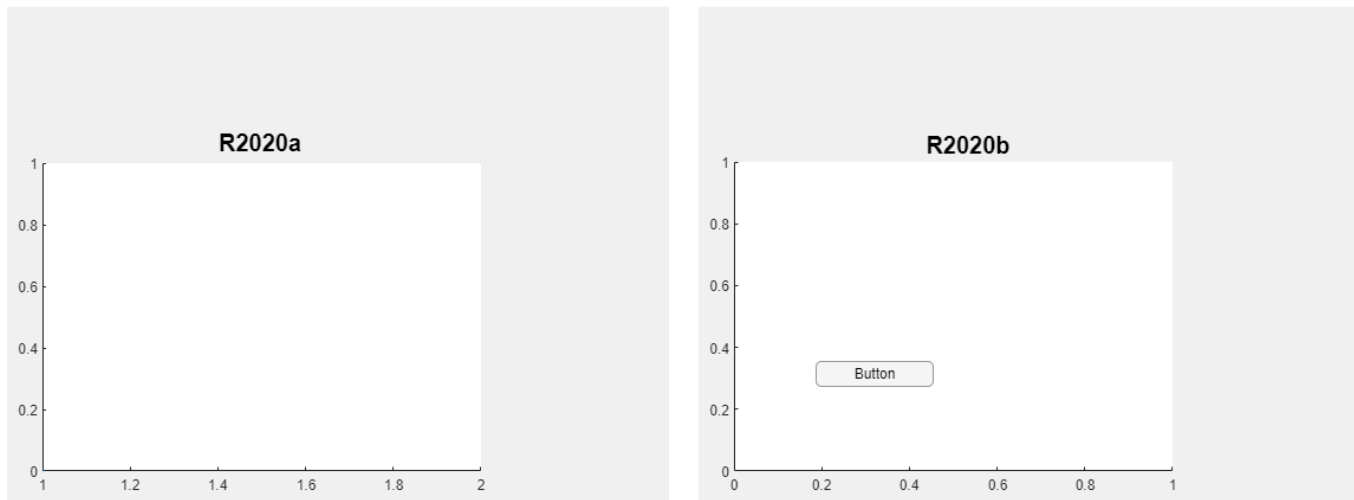
**UIAxes appear behind all other objects in the figure***Behavior change*

The stacking order (also called the z-order) of objects in the figure has changed so that UIAxes objects and their contents appear behind other UI components in the figure. This behavior is consistent with the behavior of other types of axes.

For example, this code creates a figure, a button, and then a UIAxes object.

```
fig = uifigure;  
b = uibutton(fig);  
uax = uiaxes(fig);
```

In R2020a, executing the preceding code displays the UIAxes in front of the button, as shown in the figure on the left. The figure on the right shows the behavior in R2020b, where the UIAxes appears behind UI components regardless of the order of creation.



The order of the objects listed in the `Children` property of the figure also reflects this change. The `UIAxes` object is always after UI components in the list.

```
fig.Children
```

```
ans =
```

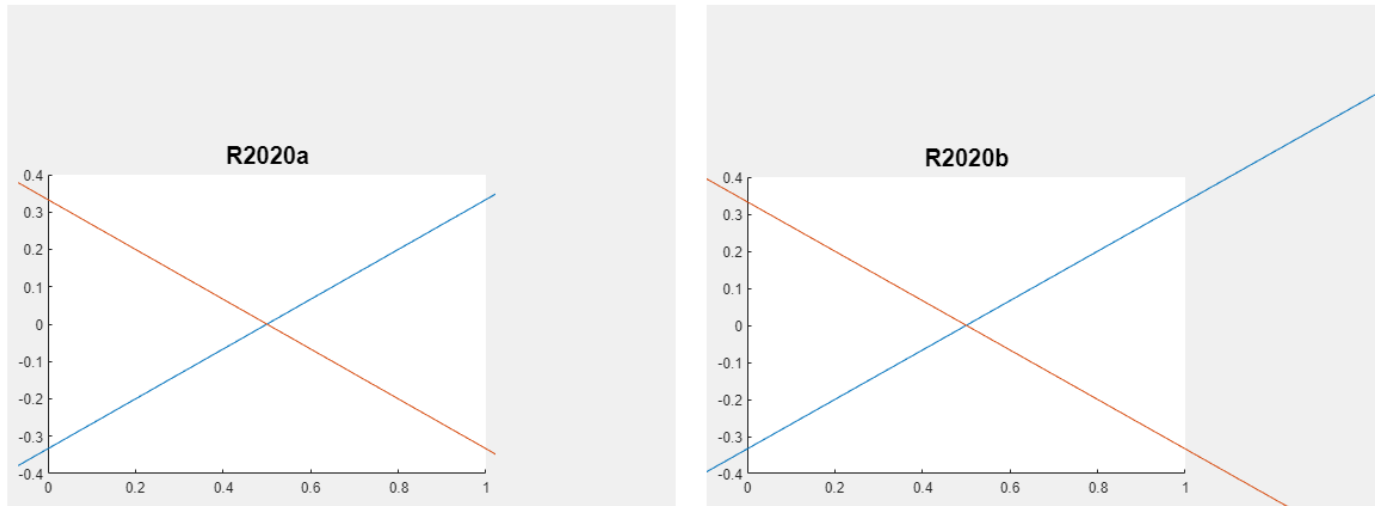
```
2x1 graphics array:
```

```
Button      (Button)
UIAxes
```

### Plots in `UIAxes` might extend outside the bounds of the axes

#### *Behavior change*

Plot objects such as lines might not clip to the bounds defined by the `OuterPosition` property of the `UIAxes`. A line extends beyond the bounds when its `Clipping` property is set to 'off'. In previous releases, lines clip to the `OuterPosition` regardless of the value of the `Clipping` property. For example, the plot on the left shows the R2020a behavior, and the plot on the right shows the R2020b behavior. In both cases, the `Clipping` properties of the lines are set to 'off'.



To prevent the axes content from overlapping with components in your app, set the `Clipping` property of each object in the axes to `'on'`.

### **Colorbars and legends displayed with UIAxes have the same parent as UIAxes**

#### *Behavior change*

When you create a plot in a `UIAxes` object, and then create a colorbar or legend for that plot, the parent object of the colorbar or legend is the same as the parent object of the `UIAxes` object. In previous releases, the parent object of the colorbar or legend is the `UIAxes` object.

### **UIAxes SizeChangedFcn callback has been removed**

#### *Errors*

The `SizeChangedFcn` callback for `UIAxes` objects has been removed. If your app requires a callback that executes when the size of the axes changes, create a `SizeChangedFcn` callback for the parent figure or another container.

## Performance

### ▲ **sum Function: Improved performance summing the first dimension of numeric arrays**

The performance of the `sum` function has improved when operating on the first dimension of numeric inputs:

- `sum(A,1)` and `sum(A,'all')` are faster for matrix or N-D array `A`.
- `sum(v)` is faster for row or column vector `v`.

This performance improvement was added for single inputs in R2017b, and it has now been added for all other numeric types in R2020b: `double`, `logical`, and integer data types (`int8`, `int16`, ...).

For example, if you sum the elements of a vector with 1e9 elements in a loop, performance in R2020b is about 1.4x faster than in R2020a.

```
function timingSum
x = 1/100;
v = x*ones(1e9,1);

tic
for i = 1:10
    b = sum(v);
end
toc
end
```

The approximate execution times are:

**R2020a:** 2.0 s

**R2020b:** 1.4 s

The code was timed on a Windows 10, Intel Xeon W-2133 CPU @ 3.60 GHz test system by calling the function `timingSum`.

### ▲ **Compatibility Considerations**

This performance improvement arises from code changes that also lead to a different set of round-off behaviors in `sum`. The new algorithm reduces the amount of round-off error in calculations, which leads to a more accurate result in general. Therefore, the output of `sum` might change slightly in R2020b compared to R2020a when operating on numeric inputs, even though the results between the two versions are numerically equivalent.

### ▲ **polyfit Function: Improved performance fitting data**

The performance of the `polyfit` function has improved for input data of all sizes and fits of any order.

For example, if you fit a quadratic polynomial to a 500-element vector in a loop, performance in R2020b is about 25x faster than in R2020a.

```
function timingPolyfit
rng default
x = 1:500;
y = -0.3*x + 2*randn(1,500);

for i = 1:1e4
    p = polyfit(x,y,2);
end
end
```

The approximate execution times are:

**R2020a:** 3.5 s

**R2020b:** 0.14 s

The code was timed on a Windows 10, Intel Xeon W-2133 CPU @ 3.60 GHz test system using the `timeit` function:

```
timeit(@timingPolyfit)
```

## ⚠ Compatibility Considerations

The performance improvement arises from changes to the `polyfit` algorithm that also produce more accurate results. Therefore, the output of `polyfit` might change slightly in R2020b compared to R2020a, even though the results between the two versions are numerically equivalent.

## accumarray Function: Improved performance with fill values and certain function handles

The performance of the `accumarray` function has improved in these cases:

- When you use the default `@sum` accumulation function and specify the `fillval` input.
- When you specify the accumulation function as `@min` or `@max`.

For example, when you accumulate a vector with 10,000 elements into an output vector with 100 elements and specify a fill value, performance in R2020b is about 14x faster than in R2020a.

```
function timingAccumarrayFillVal
rng default
subs = randi(1e2,1e4,1);
val = randn(1e4,1);
sz = [1e2 1];
fillval = 1;

for k = 1:2e4
    A = accumarray(subs,val,sz,[],fillval);
end
end
```

The approximate execution times are:

**R2020a:** 11.2 s

**R2020b:** 0.8 s



Also, when you accumulate a vector with 10,000 elements into an output vector with 100 elements and specify the accumulation function as `@min` or `@max`, performance in R2020b is about 8.4x faster than in R2020a.

```
function timingAccumarrayFcnHandle
    rng default
    subs = randi(1e2,1e4,1);
    val = randn(1e4,1);
    sz = [1e2 1];

    for k = 1:2e4
        A = accumarray(subs,val,sz,@max);
    end
end
```

The approximate execution times are:

**R2020a:** 7.6 s

**R2020b:** 0.9 s

In each case, the code was timed on a Windows 10, Intel Xeon W-2133 CPU @ 3.60 GHz test system using the `timeit` function:

```
timeit(@timingAccumarrayFillVal)
timeit(@timingAccumarrayFcnHandle)
```

## spdiags Function: Improved performance constructing sparse banded matrices

The performance of the `spdiags` function has improved by a factor of up to 3x when constructing sparse banded matrices.

For example, if you create a large banded sparse matrix of order  $1e7$  with 11 nonzero diagonals, performance in R2020b is about 2.6x faster than in R2020a.

```
function timingSpdiags
    n = 1e7;
    B = randn(n,11);
    d = -5:5;

    tic
    A = spdiags(B, d, n, n);
    toc
end
```

The approximate execution times are:

**R2020a:** 11.1 s

**R2020b:** 4.2 s

The code was timed on a Windows 10, Intel Xeon W-2133 CPU @ 3.60 GHz test system by calling the function `timingSpdiags`.

## uilibox: Improved performance when setting multiple items in a list box

Setting property values on a `uilibox` shows improved performance when operating on a large number of selected items. For example, this code is about 117x faster than in the previous release:

```
function timingTest
lb = uilibox;
lb.Items = "Item " + (1:1000);
lb.ItemsData = 1:1000;
lb.Multiselect = true;
lb.Value = 1:800;
end
```

The approximate execution times are:

**R2020a:** 4.7 s

**R2020b:** 0.04 s

The code was timed on a Windows 10, Intel Xeon W-2133 CPU @ 3.60 GHz test system using the `timeit` function:

```
timeit(@timingTest)
```

## uitree: Improved performance when creating many nodes in a tree

The `uitreenode` function shows improved performance when creating a large number of nodes parented to a `uitree`. For example, this code that creates a tree with 3,000 nodes is about 2.4x faster than in the previous release:

```
function timingTest
tlto3000 = [];
fig = uifigure;
tree = uitree(fig);
drawnow
for i=1:3000
    treeNodeText = ['t' num2str(i)];
    tlto3000(i) = uitreenode('Parent',tree,'Text',treeNodeText);
end
drawnow
end
```

The approximate execution times are:

**R2020a:** 81.4 s

**R2020b:** 33.7 s

The code was timed on a Windows 10, Intel Xeon W-2133 CPU @ 3.60 GHz test system using the `timeit` function:

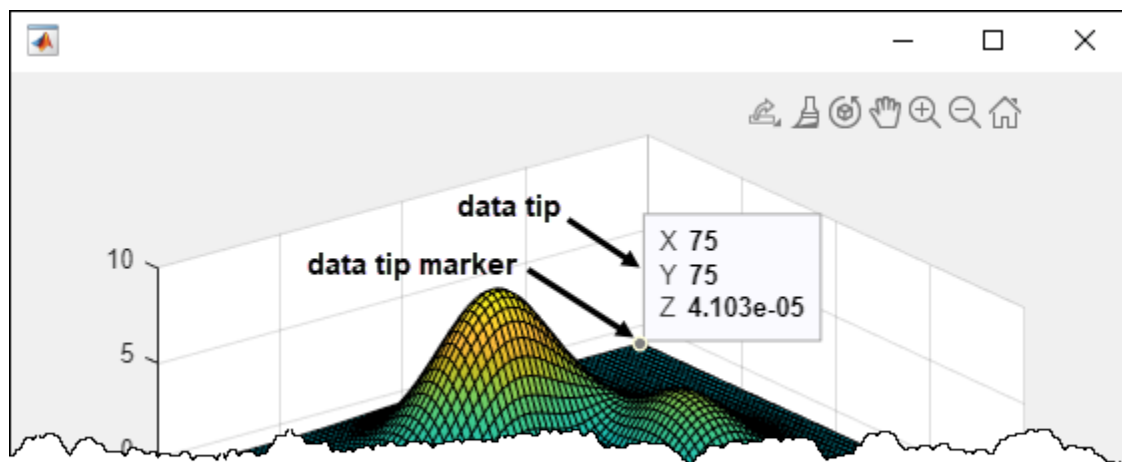
```
timeit(@timingTest)
```

## Data Tip Markers: Improved rendering performance of data tip markers in surface plots of large data sets created in UI figures and MATLAB Online

In figures created with the `uifigure` function and in MATLAB Online, data tip markers for surface plots of large data sets render faster and move more continuously in R2020b than in R2020a. This improvement can be seen when the axes are created with either the `axes` or `uiaxes` function.

For example, on a *Windows 10, Intel Xeon W-2133 CPU @ 3.6 GHz* test system with an *NVIDIA Quadro P600* graphics card, when you move your mouse quickly over the plot lines created by this code, the data tip marker moves more smoothly and tracks your mouse motion more closely in R2020b than in R2020a.

```
fig = uifigure;  
ax = axes(fig);  
surf(ax,peaks(75))
```



## Software Development Tools

### Code Compatibility Report: Unsupported Functionality Will Now Issue Warning

The code compatibility report will now check and notify you of some unsupported functionality within the code.

### Dependency Analyzer: Export to archive and generate a dependency report

Dependency analysis for projects is improved and includes new capabilities:

- Export files in the dependency graph to an archive.
- Save the dependency analysis results in a printable report.
- Investigate where in your files the dependency to a product is introduced.
- Determine which files, if any, use functionalities shared among different MathWorks products.

For more information, see [Analyze Project Dependencies](#).

### Source Control: Improved workflow to set up Git source control

The workflow to set up Git is improved and simplified.

- You do not need to install command-line Git to fully use Git with MATLAB. You can now merge branches using the built-in Git integration.
- For newly created projects using Git and for projects that switched to Git from another source control, MATLAB automatically generates a `.gitattributes` file and populates it with a list of common binary files. You do not need to manually register binary files to prevent corruption.

For more information, see [Set Up Git Source Control](#).

### Projects: Change project definition file type and preserve source control history

You can now use `matlab.project.convertDefinitionFiles` to programmatically change the project definition file management from the type selected when you first created the project to a new type. `matlab.project.convertDefinitionFiles` preserves the source control history of your project. To avoid any merging issues, make sure to convert the definition file type only once for your project.

### Unit Testing Framework: Run tests in parallel on clusters and clouds

You can now run your tests on clusters and clouds using MATLAB Parallel Server. To run tests on a remote parallel pool, use the same API the framework offers for running tests on a local parallel pool. In other words, use the `runInParallel` method or the `runtests` function with the `'UseParallel'` name-value pair argument.

## Unit Testing Framework: Run tests in parallel with standalone applications

Starting in R2020b, you can create standalone applications that support running tests in parallel. To ensure that MATLAB Compiler can locate and package all of the components required for running tests in parallel, use this directive in your program:

```
%#function parallel.Pool
```

For more information, see [Compile MATLAB Unit Tests](#).

## Unit Testing Framework: Report the validity of shared test fixtures

Starting in R2020b, you can report the validity of shared test fixtures to the testing framework. A shared test fixture is valid if the test environment state, configured by the fixture, is maintained throughout the test session.

To report the validity of a shared fixture, override the `needsReset` method of the `matlab.unittest.fixtures.Fixture` class. When running multiple test classes as a single suite, the framework tears down a shared test fixture and sets it up for the subsequent classes if `needsReset` returns `true`.

## App Testing Framework: Perform choose gestures on context menu items

The `chooseContextMenu` method enables you to test choose gestures on context menu items for UI components. For example, create a context menu with two menu items in a UI figure and choose one of the menu items.

```
fig = uifigure;
cm = uicontextmenu(fig);
m1 = uimenu(cm,'Text','Menu1');
m2 = uimenu(cm,'Text','Menu2');
fig.ContextMenu = cm;

tc = matlab.unittest.TestCase.forInteractiveUse;
tc.chooseContextMenu(fig,m1)
```

## App Testing Framework: Perform drag gestures on axes and UI axes

The app testing framework supports drag gestures on axes and UI axes. When you test a drag gesture on an axes or UI axes, the framework mimics a user manipulating the component and adjusts the axes limits based on the difference between the start and stop values. For example, create an axes with a plot and then drag from the point (3, 2) to the point (4, 2).

```
f = uifigure;
ax = axes(f);
plot(ax,1:10)

tc = matlab.unittest.TestCase.forInteractiveUse;
tc.drag(ax,[3 2],[4 2])
```

## App Testing Framework: Perform gestures on push tools and toggle tools

The app testing framework supports gestures on more UI components.

- Perform press gestures in tests on push tools.
- Perform press and choose gestures in tests on toggle tools.

**⚠️Functionality being removed or changed**

**SVN cleanup no longer removes unversioned or ignored files**

*Behavior change*

Starting in R2020a Update 5, SVN cleanup only removes stale locks and unfinished transactions. It does not remove unversioned or ignored files.

You can manually remove unversioned and ignored files.

- 1 In the Current Folder browser, click the **SVN** header to sort files by their SVN status.
- 2 Select the **Not Under Source Control** files.
- 3 Right-click and select **Delete**.

For more information, see [Get SVN File Locks](#).

**ProfileReport has been removed**

*Errors*

The ProfileReport class has been removed. Use CoverageReport or CoberturaFormat instead. Compared to ProfileReport, these two classes generate more accurate code coverage reports.

To update your code, change instances of ProfileReport to CoverageReport or CoberturaFormat. This table shows an example of how you can update your code.

| Before                                                                                                                                                                                                                                                                                 | After                                                                                                                                                                                                                                                                                                        |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <pre>import matlab.unittest.TestRunner import matlab.unittest.plugins.CodeCoverage import matlab.unittest.plugins.codecoverage  runner = TestRunner.withNoPlugins; plugin = CodeCoveragePlugin.forFolder('myTests', ...     'Producing',ProfileReport); runner.addPlugin(plugin)</pre> | <pre>import matlab.unittest.TestRunner import matlab.unittest.plugins.CodeCoveragePlugin import matlab.unittest.plugins.codecoverage.CoverageReport  runner = TestRunner.withNoPlugins; plugin = CodeCoveragePlugin.forFolder('myTests', ...     'Producing',CoverageReport); runner.addPlugin(plugin)</pre> |

To create a MATLAB Profiler Coverage Report without specifying a ProfileReport format, see [Determine Code Coverage Using the Profiler](#).

**Indexing order has changed in choose gestures within button groups**

*Behavior change*

With the app testing framework, you can choose a radio button or toggle button using the button label or its index inside the button group. Starting in R2020b, when you choose a radio button or toggle button using an index, the framework indexes into the Buttons property of the ButtonGroup object. In previous releases, the framework indexes into the Children property of the ButtonGroup object. With this change, the index corresponds to the order in which buttons are created. For example, create a button group that has six toggle buttons.

```
f = uifigure;
bg = uibuttongroup(f);
```

```

tb1 = uitogglebutton(bg, 'Position', [11 165 140 22], 'Text', 'One');
tb2 = uitogglebutton(bg, 'Position', [11 140 140 22], 'Text', 'Two');
tb3 = uitogglebutton(bg, 'Position', [11 115 140 22], 'Text', 'Three');
tb4 = uitogglebutton(bg, 'Position', [11 90 140 22], 'Text', 'Four');
tb5 = uitogglebutton(bg, 'Position', [11 65 140 22], 'Text', 'Five');
tb6 = uitogglebutton(bg, 'Position', [11 40 140 22], 'Text', 'Six');

```

This table shows the outcome of the choose gesture on a toggle button that is specified with index 2.

| Test                                                    | Starting in R2020b                                                                                                                                                                          | R2020a and Earlier                                                                                                                                                                            |
|---------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <pre>tc = matlab.uitest.TestCase; tc.choose(bg,2)</pre> | <p>MATLAB chooses toggle button <b>tb2</b>, which is the second element of the <b>bg.Buttons</b> array. The <b>tc.choose(bg,2)</b> syntax is equivalent to <b>tc.choose(bg, 'Two')</b>.</p> | <p>MATLAB chooses toggle button <b>tb5</b>, which is the second element of the <b>bg.Children</b> array. The <b>tc.choose(bg,2)</b> syntax is equivalent to <b>tc.choose(bg, 'Five')</b>.</p> |

If this behavior change impacts your code, update the index or replace it with the appropriate button label.

## External Language Interfaces

### **C++ Interface: Support for nullptr**

You can now pass `nullptr` to C++ functions. For more information, see `nullptr` Argument Types.

### **C++ Interface: Create interface with C++ source files**

You can build a MATLAB interface to a C++ library or algorithm from source files that contain complete implementations for the library. Use the `SupportingSourceFiles` name-value pair in the `clibgen.generateLibraryDefinition` function. For an example, see [Publish Interface to C++ Library Using Source Files](#).

### **Python: Version 3.8 support**

MATLAB now supports CPython 3.8, in addition to existing support for 2.7, 3.6, and 3.7. For more information, see [Configure Your System to Use Python](#).

### **Python: Terminate Python interpreter and start new one in same MATLAB session**

When you run the Python interpreter in out-of-process mode, you can terminate the interpreter and start a new one without restarting MATLAB. For more information, see [Reload Out-of-Process Python Interpreter](#).

### **mxCreateString C Matrix API functions: UTF-8 support**

The C Matrix API `mxCreateString` function now accepts UTF-8 encoded data in addition to supporting local code page (LCP) encoded strings for backwards compatibility. The `mxArrayToUTF8String` function returns UTF-8 encoded data.

Also, the `mexPrintf`, `mexEvalString`, `mexEvalStringWithTrap`, `mexErrMsgIdAndTxt`, and `mexWarnMsgIdAndTxt` C functions now accept UTF-8 encoded data in addition to supporting LCP encoded strings for backwards compatibility.

### **MATLAB Data API: Create `matlab::data::Object` arrays**

Use the `matlab::data::ArrayFactory` `createArray` method to create a `matlab::data::ObjectArray` and the `createScalar` method to create a scalar `matlab::data::ObjectArray`. You can use `[]` indexing and a `matlab::data::TypedIterator<T>` to access the elements of a `matlab::data::ObjectArray`.



## ⚠ Compiler support changed for building C++ interfaces, MEX files, and standalone MATLAB engine and MAT-file applications

| Support | Compiler                                                                                        | Platform |
|---------|-------------------------------------------------------------------------------------------------|----------|
| Added   | gcc versions 5.x and higher. If you use version 4.0 or earlier, then MATLAB displays a warning. | Linux    |

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

## ⚠ Functionality being removed or changed

### Java packages to be removed

*Behavior change in future release*

Certain Java libraries (JAR files) used by MathWorks products might not be included in future releases. To continue using functionality from these libraries, install the JAR files and add them to the class path used by MATLAB. For information about adding JAR files to the static class path, see [Static Path](#).

### Undocumented com.mathworks Java packages to be removed

*Warns*

The MATLAB Code Compatibility Report lists com.mathworks Java classes as **Unsupported functionality that might cause errors**. Except for documented interfaces and classes in the com.mathworks.engine and com.mathworks.matlab.types packages (the MATLAB Engine API for Java) and the Java Client API for use with MATLAB Compiler SDK™ and MATLAB Production Server™, all other com.mathworks Java interfaces and classes are undocumented and might be modified or removed without warning in future releases. For code stability, avoid using any of these undocumented Java interfaces or classes.

## Hardware Support

### **Live Editor Task: Interactively capture images from USB Webcam interactively and generate MATLAB code in a live script.**

Use the **Acquire Webcam Image** Live Editor Task in the MATLAB Support Package for USB Webcams, to connect to a webcam, set properties, and capture images without writing MATLAB code. The task provides you with controls that help you set webcam resolution values and other image-specific and device-specific properties. The task also automatically generates MATLAB code that becomes part of your live script.

To use tasks in the Live Editor, on the **Live Editor** tab, in the **Task** menu, select a task. Alternatively, in a code block in a live script, begin typing the task name and select the task from the suggested command completions. For more information about Live Editor tasks, see [Add Interactive Tasks to a Live Script \(MATLAB\)](#).

### **⚠️ Functionality being removed or changed**

#### **mkrMotorCarrier function will be removed in a future release**

*Warns*

Starting R2020b, using the `mkrMotorCarrier` function will throw a warning message. This function will be removed in a future release. Use the `motorCarrier` instead of `mkrMotorCarrier` to connect to the MKR Motor Carrier hardware.

#### **MKRMotorCarrier library will be removed in a future release**

*Warns*

Starting R2020b, using the `'MKRMotorCarrier'` library will throw a warning message. This library will be removed in a future release. Use the `'MotorCarrier'` library instead of `'MKRMotorCarrier'` when creating a connection to the Arduino, hardware using the `arduino` object.