Mapping Toolbox™ Release Notes

How to Contact The MathWorks

`www.mathworks.com`	Web
`comp.soft-sys.matlab`	Newsgroup
`www.mathworks.com/contact_TS.html`	Technical Support

`suggest@mathworks.com`	Product enhancement suggestions
`bugs@mathworks.com`	Bug reports
`doc@mathworks.com`	Documentation error reports
`service@mathworks.com`	Order status, license renewals, passcodes
`info@mathworks.com`	Sales, pricing, and general information

508-647-7000 (Phone)

508-647-7001 (Fax)

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

For contact information about worldwide offices, see the MathWorks Web site.

Mapping Toolbox™ Release Notes

Trademarks

MATLAB and Simulink are registered trademarks of The MathWorks, Inc. See `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

The MathWorks products are protected by one or more U.S. patents. Please see `www.mathworks.com/patents` for more information.

Contents

Summary by Version

This table provides quick access to what's new in each version. For clarification, see "Using Release Notes" on page 2.

Version (Release)	New Features and Changes	Version Compatibility Considerations	Fixed Bugs and Known Problems	Related Documentation at Web Site
Latest Version V2.7.1 (R2008b)	Yes Details	Yes Summary	Bug Reports includes fixes	Printable Release Notes: PDF Current product documentation
V2.7 (R2008a)	Yes Details	Yes Summary	Bug Reports includes fixes	No
V2.6 (R2007b)	Yes Details	Yes Summary	Bug Reports includes fixes	No
V2.5 (R2007a)	Yes Details	Yes Summary	Bug Reports includes fixes	No
V2.4 (R2006b)	Yes Details	Yes Summary	Bug Reports includes fixes	No
V2.3 (R2006a)	Yes Details	Yes Summary	Bug Reports includes fixes	No
V2.2 (R14SP3)	Yes Details	Yes Summary	Bug Reports includes fixes	No
V2.1 (R14SP2)	Yes Details	Yes Summary	Bug Reports includes fixes	No
V2.0.3 (R14SP1)	Yes Details	Yes Summary	Fixed bugs	No
V2.0.2 (R14)	Yes Details	No	No bug fixes	No
V2.0.1 (R13SP1+)	Yes Details	No	"Fixed Bugs" on page 74	Version 2.0.1 product documentation

Version (Release)	New Features and Changes	Version Compatibility Considerations	Fixed Bugs and Known Problems	Related Documentation at Web Site
V2.0 (R13SP1+)	Yes Details	Yes Summary	"Fixed Bugs" on page 90	No
V1.3.1 (R13SP1)	Yes Details	No	"Fixed Bugs" on page 92	No
V1.3 (R13)	Yes Details	No	Fixed bugs	No
V1.2 (R12)	Yes Details	No	No bug fixes	No

Using Release Notes

Use release notes when upgrading to a newer version to learn about:

- New features
- Changes
- Potential impact on your existing files and practices

Review the release notes for other MathWorks™ products required for this product (for example, MATLAB® or Simulink®) for enhancements, bugs, and compatibility considerations that also might impact you.

If you are upgrading from a software version other than the most recent one, review the release notes for all interim versions, not just for the version you are installing. For example, when upgrading from V1.0 to V1.2, review the release notes for V1.1 and V1.2.

What's in the Release Notes

New Features and Changes

- New functionality
- Changes to existing functionality

Version Compatibility Considerations

When a new feature or change introduces a reported incompatibility between versions, the **Compatibility Considerations** subsection explains the impact.

Compatibility issues reported after the product is released appear under Bug Reports at the MathWorks Web site. Bug fixes can sometimes result in incompatibilities, so you should also review the fixed bugs in Bug Reports for any compatibility impact.

Fixed Bugs and Known Problems

The MathWorks offers a user-searchable Bug Reports database so you can view Bug Reports. The development team updates this database at release time and as more information becomes available. This includes provisions for any known workarounds or file replacements. Information is available for bugs existing in or fixed in Release 14SP2 or later. Information is not available for all bugs in earlier releases.

Access Bug Reports using your MathWorks Account.

Version 2.7.1 (R2008b) Mapping Toolbox Software

This table summarizes what's new in Version 2.7.1 (R2008b):

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

Using the Map Axes Map Limit Properties with `axesm`, `setm`, and `defaultm`

Changes and enhancements have been made to `axesm`, `setm`, and `defaultm` with respect to map axes properties that affect the fundamental display geometry:

- `MapProjection`
- `Zone`
- `Origin`
- `FLatLimit`
- `FLonLimit`

- `MapLatLimit`
- `MapLonLimit`

The changes result in the following improvements:

- The use of the map limit properties to set up a map axes is more intuitive.
- The way in which `defaultm` resolves possible inconsistencies between these properties is now consistent with the behavior of `axesm` and `setm`.
- The map limit properties (`MapLatLimit` and `MapLonLimit`), the frame limit properties (`FLatLimit` and `FLonLimit`), and the `Origin`, `MapProjection`, and `Zone` properties interact in a more clear and predictable fashion.

For more information, see the section "Using the Map Limit Properties" in the Mapping Toolbox™ User's Guide and bug report 319891 on the MathWorks Web site.

Changing Projection Type of an Existing Map Axes with `setm`

In previous releases, calling the `setm` function to change the `MapProjection` property of a map axes, especially when switching between an azimuthal and non-azimuthal projection (e.g., a conic or cylindrical projection), often resulted in the following types of problems:

- The modified map axes might cover a different part of the Earth.
- The map frame and graticule might fail to update properly.
- Map limit properties changed at the same time as the projection might not have the proper effect.

The `setm` function now more effectively resets the projection, clearing out settings that were specific to the earlier projection, updating the map frame and graticule, and staying in the same general part of the world (even when switching between azimuthal and non-azimuthal projections).

Compatibility Considerations

You may need to change the way in which you reset various map axes properties, such as `Origin`, `FLatLimit`, and `FLonLimit` after changing projections, as discussed in the section "Switching Between Projections" in the Mapping Toolbox User's Guide. In many cases it will no longer be necessary to reset as many properties.

Other Bug Fixes with Compatibility Considerations

- The default `FLatLimit` for `lambert` and `lambertstd` has been changed to `[-45 45]`. In previous releases, `axesm` produced huge map frames, due to the `FLatLimit` default of `[-90 90]`.
- The function `gridm` now returns handles to the line objects used to display the parallels and meridians. In previous releases, a call to `gridm` using `linespec` or property name/property value syntaxes returned empty.
- The function `geotiff2mstruct` no longer sets the `maplatlimit` and `maplonlimit` fields.
- A reference ellipsoid set to a non-default value (via the `geoid` property) no longer reverts to the default when the UTM zone is reset. For more information, see bug report 459353 on the MathWorks Web site.
- The `daspectm` function now works for azimuthal projections and units of radians.

`coast.MAT` Data File Revised

Portions of the global coastline latitude-longitude vectors in the `coast.MAT` data file have been revised to ensure proper polygon topology. The data edits comprise the following:

- Replacing or removing various "bow-tie" and degenerate linear (non-polygonal) island features.
- Opening a "pinched" section in the middle of Lake Balkhash in Central Asia.
- Merging the eastern and western sections of Wrangel Island near the Bering Strait (cut by the 180-degree meridian) into a single polygon with longitudes ranging from slightly less than 180 to slightly greater than 180.

- Eight additional edits to pull apart landmasses with points of contact and remove coastal "spikes."

Map Limit Syntaxes Removed

The following syntaxes are obsolete. An error occurs if you use them.

- `pcolorm(Z)`
- `pcolorm(Z,gratsize)`
- `surfacem(Z)`
- `surfacem(Z,gratsize)`
- `surflm(Z)`
- `surflm(Z,s)`
- `surfm(Z)`
- `surfm(Z,gratsize)`

These syntaxes displayed a data grid with geographic limits that matched the map latitude and longitude limits in the current map axes. Using the old syntaxes correctly involved knowing the latitude and longitude limits of your data and matching them to the values listed under `maplatlimit` and `maplonlimit` in the map axes properties. We have replaced these syntaxes with a more direct approach that requires you to enter the latitude and longitude limits for the data grid.

Compatibility Considerations

The table below suggests alternative code to replace the obsolete syntaxes. In the following table, `Z` is a regular data grid (a 2-D array of class `double`) and `gratsize` is a two-element vector specifying the size of the graticule on which `Z` displays:

```
gratsize = [number_of_parallels number_of_meridians]
```

`h` is a handle to the surface that is displayed. And `latlim` and `lonlim` are the geographic limits of the data grid (in degrees):

```
latlim = [southern_limit northern_limit]
```

```
lonlim = [western_limit eastern_limit]
```

Original Syntax	Replacement Syntax

`h = pcolorm(Z)`
 	
constructs a surface using the regular data grid `Z` and a graticule mesh (using `meshgrat`) with size equal to `size(Z)` and with geographic limits that match the map latitude and longitude limits in the current map axes.
	`[lat,lon] = meshgrat(latlim,lonlim,size(Z));`
	
	`h = pcolorm(lat,lon,Z)`

`h = pcolorm(Z,gratsize)`
 	
uses a graticule mesh with size equal to `gratsize`.
	`[lat,lon] = meshgrat(latlim,lonlim,gratsize);`
	
	`h = pcolorm(lat,lon,Z)`

`h = surfacem(Z)` constructs a surface using the regular data grid `Z` and a graticule mesh (using `meshgrat`) of size 50-by-100. The geographic limits match the map latitude and longitude limits in the current map axes.	`h = surfacem(latlim,lonlim,Z)`

`h = surfacem(Z,gratsize)`
 	
uses a graticule mesh with size equal to `gratsize`.
	`[lat,lon] = meshgrat(latlim,lonlim,gratsize);`
	
	`h = surfacem(lat,lon,Z)`

`h = surflm(Z)` constructs a surface using the regular data grid `Z` and a graticule mesh (using `meshgrat`) with size equal to `size(Z)` and with geographic limits that match the map latitude and longitude limits in the current map axes. It is displayed with a default light source.	`h = surflm(latlim,lonlim,Z)`

Original Syntax	Replacement Syntax
`h = surflm(Z,s)` specifies the direction of the light source. `s` is a two- or three-element vector that specifies the direction from the surface map to the light source as defined in the documentation for `surfl`.	`h = surflm(latlim,lonlim,Z,s)`
`h = surfm(Z)` constructs a surface using the regular data grid `Z` and a graticule mesh (using `meshgrat`) with size equal to `size(Z)` and with geographic limits that match the map latitude and longitude limits in the current map axes.	`h = surfm(latlim,lonlim,Z)`

`h = surfm(Z,gratsize)`
 	
uses a graticule mesh with size equal to `gratsize`.
	`[lat,lon] = meshgrat(latlim,lonlim,gratsize);`
	
	`h = surfm(lat,lon,Z)`

Version 2.7 (R2008a) Mapping Toolbox Software

This table summarizes what's new in Version 2.7 (R2008a):

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

Functions for Working with Geographic Quadrangles

A geographic quadrangle is an area on the surface of a sphere or ellipsoid bounded on the east and west by a pair of meridians and on the north and south by a pair of parallels. In many ways, such an object is similar to a bounding rectangle in the plane, but they can be difficult to work with because of the way longitudes wrap around and the way meridians converge at the poles. For example,

- The western longitude limit can have a larger numerical value than the eastern longitude limit.
- If one of the bounding latitudes is +90 or -90 degrees, the quadrangle has thee sides rather than four.
- As noted below, the intersection of two geographic quadrangles might possibly comprise two separate parts—with the eastern end of the first quadrangle intersecting the western end of the second quadrangle, and vice versa.

Mapping Toolbox software typically represents a geographic quadrangle in terms of its latitude and longitude limits, stored in 1-by-2 vectors having the forms

```
latlim = [southern_limit northern_limit]
lonlim = [western_limit eastern_limit]
```

Vectors like these have been used in various Mapping Toolbox functions since its inception, and can appear in the input or output argument lists of over dozen functions.

In R2008a, three new functions let you query, intersect, and display geographic quadrangles, and account for subtleties such as those described above:

- `ingeoquad` — Returns `true` for points inside or on latitude-longitude quadrangle
- `intersectgeoquad` — Returns intersection(s) of two latitude-longitude quadrangles
- `outlinegeoquad` — Returns sampled polygon vertices for a latitude-longitude geographic quadrangle

Use `ingeoquad`, for example, to check whether a geographic point is located within the area covered by a regular data grid, given the latitude and longitude limits computed by `limitm`.

Use `intersectgeoquad` to compute overlap, if any, between two quadrangles. Interestingly, three general results are possible: no intersection, an intersection that is itself a geographic quadrangle, and an intersection the comprises two distinct geographic quadrangles. (The intersection can have two parts if the input quadrangles wrap around in longitude to overlap on both their eastern and western sides. This case, of course, is not possible for bounding boxes in the plane.)

Use `outlinegeoquad` to generate a pair of latitude and longitude coordinate vectors that define a polygon that traces the outline of a geographic quadrangle. This can be useful for displaying the quadrangle graphically using `geoshow`, for example, especially on a projection where the meridians and/or parallels do not project to straight lines, because in addition to connecting the four corners `outlinegeoquad` lets you interpolate additional vertices along parallels, meridians, or both.

Fixes and Improvements to Function avhrrgoode

Function `avhrrgoode` has been rewritten to improve its efficiency and to remove a number of problems and limitations:

- Fixed a spatial referencing problem when a nonglobal region has been specified which caused locations to be offset by half a pixel.
- The function no longer returns incorrect NaN coordinate values at the equator when given certain latitude limits that cross the equator.
- The function no longer errors when attempting to read a file name with certain legal latitude and longitude limits.
- The new version executes at least five times faster.

Compatability Considerations

- The nonfunctional syntaxes `avhrrgoode` and `avhrrgoode(region)` have been removed from the documentation.

- The function now returns empty when the user-supplied limits are outside data limits.
- The function no longer permits longitude limits to be specified outside the interval [-180 180].
- Parameters other than `region` and `filename` can be specified as empty to use their default values.
- In versions prior to R2008a, when reading from the global data set and a smaller region data set, the size of the outputs differed by one column from each other when given identical latitude and longitude limits. Now the sizes are the same.

Improved Accuracy for the limitm and setpostn Functions

In previous releases, after calculating the latitude and longitude limits of the geographic quadrangle bounding a regular data grid, function `limitm` arbitrarily rounded those limits to the nearest one millionth of a degree (equivalent to about 10 cm in latitude or equatorial longitude). Although it is small, this rounding operation in effect applied an arbitrary shift to points on or very near the edge of the grid. The direction of the shift and its magnitude were arbitrary because rounding can either increase or decrease a value. In any given case, the shift depended on the specific referencing vector and the number of columns and rows in the data grid. This behavior unnecessarily degraded the numerical accuracy of `limitm` and those functions which depend on it, and it has now been removed. For more information, see bug report 420038 on the MathWorks Web site.

In the `setpostn` function , an identical rounding step has been removed. Additional changes eliminate a problem for certain input points near boundaries between grid cells that caused row and column subscripts returned by `setpostn` to be off by 1. For points near the northern and eastern edges of the data grid—but still within the grid—returned subscript values could exceed the corresponding grid size. For more information, see bug report 173338 on the MathWorks Web site.

Compatibility Considerations

These corrections can cause subtle changes in the behavior of other functions that work with regular data grids referenced to latitude-longitude, for example, `imbedm`.

If your referencing vector contains approximations to rational numbers that do not have an exact a 64-bit floating point representation (e.g., for cells that are 1.5 degrees wide, `refvec(1)` is 0.666666...), you may still find that certain points that are extremely close to a grid cell boundary cross into a neighboring cell just across the boundary. Such numerical ambiguity is inevitable given how the information in a referencing vector is encoded. Although it cannot be eliminated within `setpostn`, the inexactness only affects points that fall within a few factors of `eps` (very much less than a millionth of a degree) away from a given cell boundary.

New Point Location Demo Data for Tsunami Events

The Mapping Toolbox demo data in the `$MATLABROOT/toolbox/map/mapdemos` directory now includes a global tsunami data set in shapefile format with `'Point'` geometry. The data set comprises four files:

```
tsunamis.dbf
tsunamis.shp
tsunamis.shx
tsunamis.txt
```

`tsunamis.txt` is not part of the shapefile set. It is a text file documenting the data set.

The data includes tidal wave events for which the maximum water height was at least one meter, ranging for the years 1950 to 2006, inclusive. The Global Tsunami Database, U.S. National Geospatial Data Center (NGDC), National Oceanic and Atmospheric Administration (NOAA), available at `http://www.ngdc.noaa.gov/seg/hazard/tsu.shtml`, is the source of the data. All the files consist of U.S. Government information that is in the public domain and is not subject to copyright protection.

The approximate location of each event is a single point in geodetic coordinates (latitude-longitude) with an unspecified datum. The `.dbf` file contains 18

separate text or numeric attributes for most events, including wave height, causes and seismic magnitudes, and location and country names.

The shapefiles were created at The MathWorks™ from querying the online source data, importing the results into the MATLAB workspace, and exporting them using the Mapping Toolbox `shapewrite` function. For more information, type

```
edit tsunamis.txt
```

at the MATLAB prompt.

Better Trimming Benefits fillm Function

The changes described in the Versin 2.6 (R2007b) release note “Improvements to Data Trimming in patchm and patchesm” on page 27 resulting from improved polygon trimming also apply to the `fillm` function.

Restored units Options for Function angl2str

The `angl2str` function once again can format strings for angles in degrees-minutes (DM) and degrees-minutes-second (DMS) notations. These options were removed in Version 2.6 (R2007b), and have now been restored. In addition to the `'degrees'` and `'radians'` *`units`* options, you can now obtain DM- and DMS-formatted strings by specifying

- `'degrees2dm'` — for degrees-decimal minutes formatting
- `'degrees2dms'` — for degrees-minutes-decimal seconds formatting

To use these options, input angles must be in degrees. That is, `angl2str` uses the string *`units`* to indicate both the units in which the `angle` argument is provided *and* to control the output format.

This change restores the behavior of `angl2str` prior to Version 2.6 in a slightly different form. Before V. 2.6, the DM and DMS options were specified by a *`units`* strings of `'dm'` and `'dms'`, respectively. The new strings that replace them signify that the functions `degrees2dm` and `degrees2dms`, introduced in Version 2.5 (R2007a), perform the conversions of inputs given in degrees to DM and DMS notation.

New Longitude-Wrapping Option in the closePolygonParts Utility

The `closePolygonParts` function now accepts an optional third argument, `angleunits`, that must be string-valued and can be either `'degrees'` or `'radians'`. If you include this argument with a value appropriate for the first two `(lat, lon)` arguments, `closePolygonParts` can correctly account for longitude wrapping. For example, a polygon that begins at a given latitude with a longitude of -180 degrees, and ends at the same latitude with a longitude of 180 degrees is regarded as closed and an additional vertex is not added.

Changes to Terminology for Geographic Data Structures

From Version 2.0 onward, the Mapping Toolbox documentation has referred to "version 1 geographic data structures" and "version 2 geographic data structures," using the terms "geostruct1" and "geostruct2" respectively as shorthand for them. To reflect current usage, starting with this version of the toolbox, these terms are obsolete; new terms and distinctions have been defined to help clarify what these structures are and can be used for:

- Geographic data structure arrays, introduced in Version 2.0, contain vector features and are called either
  - *Geostructs*, if they contain geographic coordinates (latitudes and longitudes)
  - *Mapstructs*, if they contain projected map/planar coordinates (*x* and *y*)
- *Display structure arrays*, dating from Version 1, also used to be called geographic data structures, and can contain either vector features or raster geodata.

Due to their greater generality, geostructs and mapstructs are the preferred form in which to represent vector features in the toolbox. The preferred way to package raster geodata is with regular or geolocated data grids (2-D numeric arrays accompanied by referencing matrices or vectors). There are only a few Mapping Toolbox functions that can still generate display structures (by importing data from external file formats):

- `dcwdata` — Returns line/patch display structures

- `dcwgaz` — Returns line/patch display structures
- `demdataui` — Returns "regular"—as in regular data grid, that is—display structures
- `mlayers` — GUI to control plotting of display structure elements
- `tgrline` — Returns line/patch display structures
- `vmap0data` — Returns line/patch display structures
- `vmap0ui` — GUI for selecting data from Vector Map Level 0

Even fewer functions accept display structures as inputs:

- `displaym` — Displays elements of a display structure
- `extractm` — Extracts lat-lon coordinates from line/patch display structure

In addition to `displaym` and `extractm`, the `updategeostruct` function converts a line or patch display structure to a geostruct.

For more information, see "Mapping Toolbox Geographic Data Structures".

Identifiers Provided for all Warnings

All warnings issued from within Mapping Toolbox functions now include identifiers, enabling you to suppress them at your own discretion. Previously, this was possible for only certain warnings, but with the addition of new identifiers in over two dozen functions in R2008a, all warnings are now covered. For example, you can turn off the warning that `setpostn` issues when given a latitude-longitude position outside the limits of the specified data grid. In this case, the warning identifier is

```
'map:setpostn:pointOutsideLimits'
```

You can suppress it using the following statement:

```
warnstate = warning('off','map:setpostn:pointOutsideLimits');
```

Then, after making your call to `setpostn`, you can restore the original warning state with

```
warning(warnstate);
```

See the MATLAB `warning` function reference page for the for more information on turning warnings off and on and managing the warning state.

Documentation for Functions tigermif and tigerp Removed

The reference pages for following functions, which themselves were removed in R2007b, have been removed from the Mapping Toolbox User's Guide:

- `tigerp` — Read TIGER *p* and *pa* thinned boundary files (ArcInfo format)
- `tigermif` — Read the TIGER MIF thinned boundary file (MapInfo format)

Compatibility Considerations

See the R2007b release note "Functions tigermif and tigerp Are Obsolete and Error if Used" on page 35 for alternatives to `tigermif` and `tigerp`.

Removed Syntaxes that Returned Error Messages in Optional Argument

In earlier versions, the following Mapping Toolbox functions supported syntaxes that included an optional output argument called `msg`. If this output argument was included in a call to one of these functions, and certain error conditions were encountered while executing the function, then instead of issuing an error, the function would return the corresponding error message in `msg`. The following functions are affected:

- `axesm`
- `defaultm`
- `displaym`
- `gcm`
- `handlem`
- `lightm`
- `linem`
- `maps`

- meshm
- namem
- patchesm
- roundn
- surfacem
- surflsrm
- textm
- unitstr
- utmzone
- utmzoneui

For example, even with no map axes present, the command

```
[mstruct, msg] = gcm
```

returned without error in R2007b and earlier, but placed an error message string in msg.

These syntaxes have been disabled in R2008a. If you try to use them, a warning is issued. The warning may be followed by an error, depending on whether or not an error condition is encountered within the function. For example, if a map axes is present, the command above results in

```
Warning: Function GCM no longer returns error message strings in
output argument MSG. Instead any errors are thrown where they occur.
You should remove the last output argument (MSG) from your call to
GCM in order to avoid this warning. If you want to handle errors
yourself, call GCM in a try-catch block.
> In mapdisp/private/warnObsoleteMSGSyntax at 6
  In gcm at 20
```

If there is no map axes, it results in

```
Warning: Function GCM no longer returns error message strings in
output argument MSG. Instead any errors are thrown where they occur.
You should remove the last output argument (MSG) from your call to
```

```
GCM in order to avoid this warning. If you want to handle errors
yourself, call GCM in a try-catch block.
> In mapdisp/private/warnObsoleteMSGSyntax at 6
  In gcm at 20
??? Error using ==> gcm>checkaxes at 41
No axes in current figure.
Select a figure with map axes or use AXESM to define one.

Error in ==> gcm at 24
h = checkaxes(varargin{:});
```

Compatibility Considerations

As suggested by this warning, if you have any scripts or functions of your own that depend on the old syntax, you should remove the `msg` argument and place the function call in a `try-catch` block instead.

Version 2.6 (R2007b) Mapping Toolbox Software

This table summarizes what's new in Version 2.6 (R2007b):

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

Exporting Vector Geodata to Earth Browsers

`kmlwrite` is a new function for exporting vector point data to a file in KML format. KML stands for Keyhole Markup Language; it is an XML dialect used to structure geographic data for display in an Earth browser, such as Google® Earth, Google Maps, and Maps for Google Mobile. KML has a hierarchical structure of nested elements and attributes. `kmlwrite` has a simple API that lets Mapping Toolbox users write vector data to a KML file in order to subsequently display the data onto an Earth browser.

When used with Google Earth, files output from `kmlwrite` can be seen immediately in Google Earth, if that application is available to the user. If the files are uploaded to a publicly accessible Web server, they can be viewed by anyone on the Internet via Google Maps or other Web sites and browser utilities that can read and display KML files. Google Maps and Google Maps for mobile do not support the range of KML markup that Google Earth supports (for example, placemark locations must be specified to them as coordinates, not as addresses). See the Google KML documentation at http://code.google.com/apis/kml/documentation/mapsSupport.html for more information.

`kmlwrite` accepts latitude and longitude point vectors, passed either in geostructs or as column arrays. It also accepts addresses, which can be as general as a country's name or as specific as a street address. When geostructs are the input, the attribute data in the geostruct can be formatted as HTML tables and included in the KML output. When latitude-longitude arrays are input, you can pass attributes to `kmlwrite` with strings. When addresses are the input, geostructs are not used.

To customize placemarks, you can control formatting of geostruct attributes in the KML file with an attribute specification, a struct used to format them (for example, to add units to length attributes or to control the number of decimal places for numeric values). A new support function, `makeattribspec`

lets you change the names used as labels in placemarks (geostruct field names are used by default), omit fields from placemarks, and add HTML markup to the attributes displayed in placemark tables.

See "Exporting Vector Geodata" in the Mapping Toolbox User's Guide and the mapexkmlexport demo, "Exporting Vector Point Data to KML" for more information.

Improved Conversion Between Angle Units

The `angledim` function has been replaced by four, more specific, functions: `fromRadians`, `fromDegrees`, `toRadians`, and `toDegrees` (described below in "Four New Angle-Unit Conversion Functions" on page 24). However, `angledim` has been retained in Version 2.6 for backward compatibility. The functions `deg2rad`, `rad2deg`, and `unitsratio` provide additional alternatives.

Because it must resolve both the input and output units, `angledim` is excessive for most applications. It works only for class `double` and it quietly discards the imaginary part of any complex input. You can use any of several more efficient alternatives:

If you are working from the command line, you can often replace `angledim` with `deg2rad` or `rad2deg`. If you are converting angle units within a script or function and you know both the *`from`* and *`to`* unit names at the time of coding, then you can also replace `angledim` with `deg2rad` or `rad2deg`. If you know either *`from`* or *`to`* at the time of coding, then you can use `fromRadians`, `fromDegrees`, `toRadians`, or `toDegrees`. Apply one of the following transformations to your code:

- `angledim(angleIn,'radians',`*`to`*`)` ⇒ `fromRadians(`*`to`*`,angleIin)`
- `angledim(angleIn,'degrees',`*`to`*`)` ⇒ `fromDegrees(`*`to`*`,angleIin)`
- `angledim(angleIn,`*`from`*`,'radians')` ⇒ `toRadians(`*`from`*`,angleIn)`
- `angledim(angleIn,`*`from`*`,'degrees')` ⇒ `toDegrees(`*`from`*`,angleIn)`

Also note that the functions in the `fromRadians` family can convert multiple variables in a single function call. For example, you can replace this code

```
angle1 = angledim(angle1InRadians,'radians',to);
angle2 = angledim(angle2InRadians,'radians',to);
```

with

```
[angle1,angle2] =
fromRadians(to,angle1InRadians,angle2InRadians);
```

If you do not know either *from* or *to* at the time of coding, then you can call `unitsratio` to obtain the correct conversion factor, then multiply the values of one or more variables. For example, you can replace:

```
angle1Out = angledim(angle1In, from, to);
angle2Out = angledim(angle2In, from, to);
```

with

```
r = unitsratio(to, from);
angle1Out = r * angle1In;
angle2Out = r * angle2In;
```

Four New Angle-Unit Conversion Functions

The following functions have been added for efficient conversion of angle units (degrees or radians) when either the target or destination units (but not both) are unknown before run time.

- `toDegrees` — Convert angles to degrees
- `toRadians` — Convert angles to radians
- `fromDegrees` — Convert angles from degrees
- `fromRadians` — Convert angles from radians

If the output units match the inputs units, as in `toDegrees(units, angle1, angle2,...)`, where `units` turns out to equal `'degrees'`, then the input angles are simply copied to the output angles.

Use these functions in place of `angledim`. The new functions are more efficient, especially when the value of either the `from` or `to` argument of `angledim` is known in advance and the value of the other angle-unit argument is not.

Improvements in Handling Length Units

Alternatives to the distdim Function

There are now more efficient ways to convert length and distance units than the `distdim` function. In place of `distdim`, you can use `unitsratio` to compute multiplicative factors to apply when converting between different units of distances and angles, which you can use in subsequent calculations. For other alternatives, see "Replacing distdim" in the `distdim` reference page for details.

The unitstr function Is Obsolete

The `unitstr` function, which validates names and abbreviations for units of distance, angle, and time, is obsolete and will be removed in a future release. The syntax `str = unitstr(str,'times')` has already been removed. Instead, see the documentation for `unitsratio` for a list of valid unit strings.

Compatibility Considerations. There is no replacement for `unitstr`, but `unitsratio` recognizes all the unit strings known to the toolbox.

Interpretation of "Miles" Units has Changed

As of R2007b, the following functions interpret distance units specified as `'miles'` as *International Miles*, not *Statute Miles*:

- `almanac`
- `daspectm`
- `elevation`
- `mapprofile`
- `paperscale`
- `scaleruler`

Compatibility Considerations. This will not materially affect the accuracy of results in most cases; the lengths of the two types of miles only differ by about two parts per million (three millimeters). The `distdim` function's interpretation of miles has not changed. However, there are better alternatives to it; see the release note "Alternatives to the distdim Function" on page 25.

New Angle Wrapping Functions

Four new low-level functions have been added that force longitudes, azimuths, or phase angles to span intervals of `[0 360]` or `[-180 180]` degrees or `[0 2*pi]` or `[-pi pi]` radians.

- `wrapTo180` — Wrap angle in degrees to [-180 180]
- `wrapTo360` — Wrap angle in degrees to [0 360]
- `wrapToPi` — Wrap angle in radians to [-pi pi]
- `wrapTo2Pi` — Wrap angle in radians to [0 2*pi]

The first two functions work in degrees, the next two in radians. None of them perform argument checking.

You can use the new wrapping and functions in place of `npi2pi` and `zero22pi` for greater efficiency. The older functions will eventually be removed from the toolbox.

New Function to Unwrap Sequences of Angles

The new `unwrapMultipart` function unwraps vectors of angles similarly to the MATLAB function `unwrap`, except that it handles vectors that include NaN separators, unwrapping each section separately. Use it to remove discontinuities from vectors of longitudes, azimuths, or phase angles that contain NaN-delimited sequences and as a replacement for the obsolete function `smoothlong`.

Improvements to Data Trimming in patchm and patchesm

The `patchm` and `patchesm` functions now completely trim away polygons and parts of polygons that fall outside your current map limits. This improvement also affects `fillm`, which calls `patchm`. Previously the patch functions simply shifted coordinates inward so that vertices collected at the edge of the limits, where they would appear as lines along map borders, unless obscured by the map frame. This change allows OpenGL to better render the patch objects constructed by `patchm` and `patchesm`, making them more compatible with the use of `AlphaData` to achieve transparency. See the release note "Map Axes Now Display Transparent Objects More Easily" on page 29 for more details.

Compatibility Considerations

The more complete trimming in `patchm` and `patchesm` means that there are circumstances under which automatic reprojection can no longer display all the data provided to these functions. Automatic reprojection causes map objects created with `plotm`, `linem`, `patchm`, `patchesm`, and certain other display functions (but not `geoshow`) to be removed, projected, and redisplayed whenever a call to `setm` changes certain map axes properties, including the map limits and projection type. In the case of `patchm`, a set of polygons will become unavailable for automatic reprojection if *all* of the polygons are trimmed away completely. In the case of `patchesm`, which constructs a separate object for each polygon, *any* polygon that is trimmed away completely will be unavailable for reprojection, even if it would lie within newly defined map limits. In either of these cases, you should delete the handle(s) returned by `patchm` or `patchesm`, then repeat the original calls after changing your map axes properties.

Other potential compatibility issues:

- `patchm` and `patchesm` exhibit greater sensitivity to incomplete or incorrect polygon topology.
- You might need to manually set the renderer for proper display of some patch data

See the release note "Map Axes Now Display Transparent Objects More Easily" on page 29 for information about rendering and the Mapping Toolbox

demo mapexgshhs for an example of a situation where polygon topology necessitates manual setting of the renderer.

Higher Quality boston.tif GeoTIFF Satellite Image

The original `boston.tif` GeoTIFF satellite image has been replaced by a higher resolution image, created by and provided courtesy of GeoEye™. The new image has the same name as the old one, `boston.tif`. The new `boston.tif` file, and an overview image in JPEG format, `boston_ovr.jpg`, include material copyright © by GeoEye, all rights reserved. The new image is 2881-by-4481 pixels, with a ground pixel size of 3.2808333333 U.S. survey feet (one meter). The original image was 720-by-1120 pixels and had a ground pixel size of four meters. Both images cover the downtown section of Boston, Massachusetts, the Charles River, and parts of Cambridge. The new image is a "pan-sharpened" multispectral image with visible red, green, and blue bands, and is stored in RGB form. The original image was also multispectral, but was a simple composite of red, green, and blue bands, and it was written to the GeoTIFF file as an indexed-color image. One additional change is that rather using meters, the new image is spatially referenced to the Massachusetts State Plane Mainland coordinate system with units of U.S. survey feet. The overview image, `boston_ovr.jpg`, is referenced to latitude-longitude, with a ground pixel size of approximately 16 meters. For further information, refer to the text files `boston.txt`, `boston_ovr.txt`, and `boston_metadata.txt` in `toolbox/map/mapdemos`.

Compatibility Considerations

Older satellite images of Boston and a demo have been removed from Mapping Toolbox directories. The new `boston.tif` and `boston_ovr.jpg` images replace the images having the same names previously included in `toolbox/map/mapdemos`. In addition, several older images related to `boston.tif` have been removed:

- `boston_red.tif`
- `boston_green.tif`
- `boston_blue.tif`
- `boston_pan.tif`
- `boston_enhanced_pan.tif`

The `mapexenhance` demo (“Enhancing Multispectral GeoTIFF Images”), which used several of these images, has also been removed.

Map Axes Now Display Transparent Objects More Easily

It is now much easier to achieve transparency effects from the toolbox by setting the `AlphaData` property of an object. Previously, functions `axesm`, `lightm`, `contourm`, and `contour3m` set the figure's `Renderer` property: `axesm` and `lightm` set it to `'zbuffer'`, while `contourm` and `contour3m` set it to `'painters'`. You then had to manually reset `Renderer` to `'opengl'` in order for transparency to take effect.

Now the `RendererMode` of the figure retains the default MATLAB value of `'auto'`, causing MATLAB to select the most appropriate renderer for you; it will use OpenGL when appropriate, given your `AlphaData` settings. Using OpenGL not only enables transparency effects, it also can make use of hardware graphics acceleration capabilities should they be available.

Compatibility Considerations

If you need a particular map display to look the same as it did in Mapping Toolbox Version 2.5 (R2007a), in most cases you can just issue the command

```
set(gcf,'Renderer','zbuffer')
```

after you construct your map axes. If you are calling `contourm` or `contour3m`, issue the command

```
set(gcf,'Renderer','painters')
```

after you call the contouring function.

The consequence of doing this is that you will not be able to use transparency with that map figure until you reset its renderer to `'opengl'` or set its `'RendererMode'` back to `'auto'`, which is its default state.

The arcgridread Function Now Imports Noninteger Data Grids

In previous releases of the toolbox, `arcgridread` could only import data grids that had integer values (often of meters or feet). This limitation has now been removed, such that input grids can contain arbitrary values in decimal notation.

Change to avhrrlambert Function Behavior When No Data Is Available

In previous releases of the toolbox, `avhrrlambert` would error if the quadrangle defined by `latlim` and `lonlim` (when projected to form a polygon in the appropriate Lambert Equal Area Azimuthal projection) failed to intersect the bounding box of the data in the projected coordinates. In this release, `avhrrlambert` does not error when this occurs but returns empty matrices.

Compatibility Considerations

If you depend on `avhrrlambert` to error when there is no data in your quadrangle, you will need to change your code.

Enhancements to Mapping Toolbox User's Guide

Several sections of the chapter "Understanding Geospatial Geometry" have been rewritten and new material has been added to better explain critical topics such as ellipsoid models, units of and notations for angles and length, and the conversions that are possible between various units. There is also a new section, "Exporting Vector Geodata", explaining and illustrating how to use the new `kmlwrite` and `makeattribspec` functions.

Functions deg2rad and rad2deg No Longer Convert Complex to Real

In prior versions, when given complex inputs, functions and `rad2deg` issued a warning and then converted their inputs to real. Now they no longer do either of these things; in the unlikely event of complex input, these functions simply scale the imaginary part by the same factor as the real part. For example, in R2007a and earlier releases, they behave as follows:

```
>> deg2rad(180i)
Warning: Imaginary parts of complex ANGLE argument ignored
> In deg2rad at 16
ans =
             0
```

Going forward from this release, the result is

```
>> deg2rad(180i)
ans =
        0 + 3.1416i
```

Degrees-Minutes-Seconds Conversion Functions Are Obsolete and Error if Used

The following functions, which accepted or produced double scalars to represent degrees, minutes, and seconds now error when used, and will be removed completely from the toolbox in a future release:

- `deg2dm`
- `deg2dms`
- `dms2deg`
- `dms2dm`
- `dms2mat`
- `dms2rad`
- `mat2dms`
- `rad2dm`
- `rad2dms`

The scalar DM and DMS encodings are being eliminated from the toolbox because they were never used for internal computations, and always had the potential to generate serious numerical errors if passed accidentally to functions that expected normal latitude-longitude tuples. They also made the functions that accepted them less efficient due to the need to convert from DM or DMS to fractional latitudes and longitudes before processing the input data.

In every case, an alternative that does not use the old degrees-minutes-seconds scalar encoding exists. See the following section on compatibility for replacements and "New Functions for Degrees-Minutes-Seconds Conversions" on page 38 in the V2.5 Release Notes for descriptions of replacement functions, and the compatibility considerations below for descriptions of alternative syntaxes and expressions you can use for degrees-minutes-seconds conversions.

Compatibility Considerations

DM and DMS representations are widely used in published reports and can occur in geodata that you want to read into the MATLAB workspace. You can still import and export DM and DMS data, but Mapping Toolbox functions no longer accepts the old encodings as alternatives to floating-point representations of latitude and longitude for internal manipulations.

The following functions (which all use scalar DMS encoding) are being retired. They remain in the product for R2007b, but now generate errors when used. They will be removed completely in the next version. Use the alternative suggested in lieu of these functions.

- `deg2dm` — Instead use `degrees2dm` to convert degrees to degrees-minutes vector.
- `deg2dms` — Instead use `degrees2dms` to convert degrees to degrees-minutes-seconds vector.
- `dms2deg` — Instead use `dms2degrees` to convert degrees-minutes-seconds vector to degrees.
- `dms2dm` — Instead combine `dms2degrees` and `degrees2dm`, as in `degrees2dm(dms2degrees([-29 42 18.7]))` to remove the seconds component from a degree-minutes-second vector.
- `dms2mat` — Instead use `degrees2dms` to convert degrees to degrees-minutes-seconds vector.
- `dms2rad` — Instead use `dms2degrees` to convert degrees-minutes-seconds vector to degrees and call `deg2rad` or multiply by `pi/180`.
- `mat2dms` — Instead use `dms2degrees` to convert degrees-minutes-seconds vector to degrees.

- `rad2dm` — Instead, call `rad2deg` or multiply input arguments by `180/pi`, and then call `degrees2dm`.
- `rad2dms` — Instead, call `rad2deg` or multiply input arguments by `180/pi`, and then call `degrees2dms`.

In addition, the `axesm` and `setm` functions no longer accept the strings `'dms'` and `'dm'` for setting either the *`AngleUnits`* or *`LabelUnits`* properties of a map axes.

Many other Mapping Toolbox functions optionally accept angle strings for their `units` parameter; the following 57 functions now only accept `'degrees'` and `'radians'`, whereas in prior versions they would also accept `'dm'` and `'dms'` as values for *`units`*:

`angl2str`	`distance`	`histr`	`putpole`	`stdist`
`angledim`	`eastof`	`imbedm`	`rcurve`	`stdm`
`antipode`	`elevation`	`interpm`	`reckon`	`timezone`
`areaint`	`ellipse1`	`intrplat`	`rhxrh`	`track`
`areamat`	`epsm`	`intrplon`	`rotatem`	`track1`
`areaquad`	`eqa2grn`	`mapprofile`	`rsphere`	`track2`
`axesm`	`gc2sc`	`meanm`	`scaleruler`	`unitstr`
`azimuth`	`gcxgc`	`meshgrat`	`scircle1`	`westof`
`convertlat`	`gcxsc`	`neworig`	`scircle2`	`zero22pi`
`crossfix`	`gradientm`	`newpole`	`scxsc`	
`daspectm`	`grn2eqa`	`npi2pi`	`setm`	
`departure`	`hista`	`org2pol`	`smoothlong`	

These functions now error when provided `'dm'` or `'dms'` for their `units` argument.

Time Conversion Functions Are Obsolete and Error if Used

The following functions, which converted time representations, now error when used and will be removed completely from the toolbox in a future release:

- `hms2hm`
- `hms2hr`
- `hms2mat`
- `hms2sec`
- `hr2hm`
- `hr2hms`
- `hr2sec`
- `mat2hms`
- `sec2hm`
- `sec2hms`
- `sec2hr`
- `time2str`
- `timedim`

Compatibility Considerations

These functions now raise errors when they are invoked. They will be completely removed in a future version of the toolbox. No substitutes have been provided, as no operations of the toolbox have ever depended on them.

cmapui GUI is now Obsolete

The `cmapui` GUI now errors if you attempt to use it. It will be completely removed from the next Mapping Toolbox version. Use the MATLAB `colormapeditor` GUI instead, which provides better functionality. You can also use the **Colormap** drop-down menu in the Property Editor (part of the MATLAB plotting tools and available via the `propedit` command) to select a built-in colormap; the `custom` option on that drop-down menu opens `colormapeditor`. To set up a colormap for terrain displays, you can use the

`demcmap` function. To generate an appropriate (but random) colormap for political maps, use the `polcmap` function.

Functions tigermif and tigerp Are Obsolete and Error if Used

The following functions error and issue an error message when you attempt to use them:

- `tigerp` — Read TIGER p and pa thinned boundary files (ArcInfo format)
- `tigermif` — Read the TIGER MIF thinned boundary file (MapInfo format)

Compatability Considerations

In place of these format readers, download U.S. Census cartographic boundary files in shapefile format and use `shaperead` to import them.

Version 2.5 (R2007a) Mapping Toolbox Software

This table summarizes what's new in Version 2.5 (R2007a):

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

New features and changes introduced in this version are

Performance Improvements for los2 and viewshed

This release includes a faster `los2` function (which computes intervisibility between locations on or above a terrain grid). The `viewshed` function (which computes the portions of a terrain grid that can be seen from a given viewpoint) has also been accelerated as a result.

Utility Functions for Computing Distance and Position Along Meridians

Two functions that reckon position and distance along a meridian on the ellipsoid are now available:

- `meridianarc` — Computes distance along a meridian between two latitudes
- `meridianfwd` — Reckons position along meridian given a starting point and distance

Some GUIs Are No Longer Available from the Command Line

In prior releases, when you typed certain Mapping Toolbox function names with no argument list, a specialized GUI appeared that enabled you to interactively set parameters related to the function. This feature was seldom used and sometimes raised errors when users attempted to operate the GUIs. Starting in this release, a GUI will no longer appear when you issue the following commands:

- `comet3m`
- `cometm`
- `contourfm`
- `contour3m`
- `contourm`
- `demcmap`
- `fill3m`
- `fillm`
- `lightm`
- `limitm`
- `linem`
- `meshlsrm`
- `meshm`
- `patchesm`

- `patchm`
- `pcolorm`
- `plot3m`
- `plotm`
- `quiver3m`
- `quiverm`
- `scatterm`
- `stem3m`
- `surfacem`
- `surflm`
- `surfm`
- `surflsrm`
- `symbolm`
- `textm`

Compatibility Considerations

Use the above functions with arguments to avoid raising errors. Their GUIs will continue to be available via `maptool` (which places menus on a figure containing map axes), but they are not being actively supported and will be eliminated in a future release.

New Functions for Degrees-Minutes-Seconds Conversions

Four new functions have been added to convert to and from decimal degrees and degrees-minutes-seconds (DMS):

- `dms2degrees` — Convert degrees-minutes-seconds to degrees
- `dm2degrees` — Convert degrees-minutes to degrees

- `degrees2dms` — Convert degrees to degrees-minutes-seconds
- `degrees2dm` — Convert degrees to degrees-minutes

The DMS inputs and outputs of these functions are vectors of one row and three columns for each row in the decimal degrees input or output. The first column contains the "degrees" element and is integer-valued. The second column contains the "minutes" element and is integer-valued. The third column contains the "seconds" element, and may have a nonzero fractional part. Similarly, DM inputs and outputs are two-column vectors with integer degrees and fractional minutes parts.

The new conversion functions dispense with the DMS encoding used in prior versions of the toolbox. These represented DMS angles by a single real number, the format of which is `dddmm.ss`. Such an encoding is no longer used internally by Mapping Toolbox functions, as it is not self-documenting and can lead to erroneous computations. For example, two DMS-encoded real numbers cannot be added to obtain a meaningful result.

Compatibility Considerations

DM and DMS representations are widely used in published reports and can occur in geodata that you want to read into the MATLAB workspace. You can still import and export DM and DMS data, but Mapping Toolbox functions no longer accepts the old encodings as alternatives to floating point representations of latitude and longitude for internal manipulations.

The scalar DM and DMS encodings are being eliminated from the toolbox because they were never used for internal computations, and always had the potential to generate serious numerical errors if passed accidently to functions that expected normal latitude-longitude tuples. They also made the functions that accepted them less efficient due to the need to convert from DM or DMS to fractional latitudes and longitudes before processing the input data.

The following existing functions (which all use scalar DMS encoding) are being retired. They remain available but now issue warnings that they are obsolete when used:

- `deg2dm` — Instead use `degrees2dm` to convert degrees to degrees-minutes vector

- `deg2dms` — Instead use `degrees2dms` to convert degrees to degrees-minutes-seconds vector
- `dms2deg` — Instead use `dms2degrees` to convert degrees-minutes-seconds vector to degrees
- `dms2mat` — Instead use `degrees2dms` to convert degrees to degrees-minutes-seconds vector
- `dms2rad` — Instead use `dms2degrees` to convert degrees-minutes-seconds vector to degrees and call `deg2rad` or multiply by `pi/180`
- `mat2dms` — Instead use `dms2degrees` to convert degrees-minutes-seconds vector to degrees
- `rad2dm` — Instead, call `rad2deg` or multiply input arguments by `180/pi`, and then call `degrees2dm`
- `rad2dms` — Instead, call `rad2deg` or multiply input arguments by `180/pi`, and then call `degrees2dms`

In addition, the `axesm` and `setm` functions no longer accept the strings `'dms'` and `'dm'` for setting either the `AngleUnits` or `LabelUnits` properties of a map axes.

Many other Mapping Toolbox functions optionally accept angle strings for their `units` parameter; the following 57 functions now only accept `'degrees'` and `'radians'`, whereas in prior versions they would also accept `'dm'` and `'dms'` as values for *`units`*:

`angl2str`	`distance`	`histr`	`putpole`	`stdist`
`angledim`	`eastof`	`imbedm`	`rcurve`	`stdm`
`antipode`	`elevation`	`interpm`	`reckon`	`timezone`
`areaint`	`ellipse1`	`intrplat`	`rhxrh`	`track`
`areamat`	`epsm`	`intrplon`	`rotatem`	`track1`
`areaquad`	`eqa2grn`	`mapprofile`	`rsphere`	`track2`
`axesm`	`gc2sc`	`meanm`	`scaleruler`	`unitstr`
`azimuth`	`gcxgc`	`meshgrat`	`scircle1`	`westof`

`convertlat`	`gcxsc`	`neworig`	`scircle2`	`zero22pi`
`crossfix`	`gradientm`	`newpole`	`scxsc`	
`daspectm`	`grn2eqa`	`npi2pi`	`setm`	
`departure`	`hista`	`org2pol`	`smoothlong`	

These functions now issue warnings when provided `'dm'` or `'dms'` for their `units` argument.

Time Conversion Functions to be Removed

The following functions to convert between time units and encodings will be removed from a future release of the toolbox:

- `hms2hr`
- `hms2hm`
- `hms2mat`
- `hms2sec`
- `hr2hm`
- `hr2hms`
- `hr2sec`
- `mat2hms`
- `sec2hm`
- `sec2hms`
- `sec2hr`
- `time2str`
- `timedim`

Compatibility Considerations

These functions remain available, but when they are invoked now issue warnings that they are obsolete.

Version 2.4 (R2006b) Mapping Toolbox Software

This table summarizes what's new in Version 2.4 (R2006b):

New Features and Changes	Version Compatibility Considerations	Fixed Bugs and Known Problems	Related Documentation at Web Site
Yes Details below	Yes — Details in **"Changes in Behavior of mapshow and geoshow" on page 43** below. See also Summary.	Bug reports at Web site	Printable Release Notes: PDF Current product documentation

New features and changes introduced in this version are

Standard Formulations of Five Major Map Projections

New formulations of five conic map projections are provided. The existing implementations remain available under their old names. The new versions use the same names as the ones they supplement, appended with "std":

- Cassini Transverse Cylindrical (`cassinistd`)
- Albers Equal-Area Conic (`eqaconicstd`)
- Equidistant Conic (`eqdconicstd`)
- Lambert Conformal Conic (`lambertstd`)

- Polyconic (`polyconstd`)

Computations used for the new versions differ from the old ones only when the latitude origin (the first element of the *origin vector*) is nonzero. In this case, the old versions shift the origin off the equator through a solid body rotation of the sphere (or, for an ellipsoidal earth model, a suitable auxiliary sphere). This is technically correct, but differs from accepted industry standards for these projections. The new versions use the standard formulations and give results that are consistent with projection results from other software packages, regardless of the latitude origin. The old versions are retained in the toolbox, with no change in behavior, to ensure backward compatibility.

See the Projections Reference documentation for more information.

Two New Geodetic/Geocentric Latitude Conversion Functions

Two new functions provide a more direct route to functionality already available via the `convertlat` function:

- `geocentric2geodeticlat` converts an array of geocentric latitude in radians to geodetic latitude in radians on a reference ellipsoid given a first eccentricity
- `geodetic2geocentriclat` converts an array of geodetic latitude in radians to geocentric latitude in radians on a reference ellipsoid given a first eccentricity.

Accelerated Performance for geoshow, mapshow, and bufferm

Functions `geoshow`, `mapshow`, and `bufferm` run substantially faster in many cases, especially when vector display is being controlled via symbol specs in `mapshow` and `geoshow`.

Changes in Behavior of mapshow and geoshow

In addition to operating faster, the `mapshow` and `geoshow` functions now behave slightly differently regarding their defaults, handles returned, warnings issued, and several other aspects:

Default Symbols and Colors

- Point marker type changes from `'X'` to `'+'`
- Point marker color changes from `'black'` to `'red'`
- Line color changes from `'black'` to `'blue'`
- Polygon facecolor changes from `'black'` to pale yellow

Polygon edgecolor remains `'black'`

Contour DisplayType Behavior Changes

- The `DisplayType` option `'contour'` now returns an hggroup handle. The children of the hggroup are patches. In prior versions, an array of line handles was returned.
- You can specify any contourgroup property as a parameter value pair. In previous versions, `mapshow` allowed you to set the `LineStyle` property, but no other contour properties.
- Both `mapshow` and `geoshow` might return a different number of contour levels by default than in previous versions, in which you could not specify contour intervals; in R2006b, you can control contour intervals and levels via the `LevelStep` or `LevelList` contourgroup properties, among others.
- In R2006b, when plotting contours on a regular axes (not a map axes), geoshow projects the contour lines using a Plate Carree projection; in previous versions it simply displayed longitudes as *x* and latitudes as *y* without doing any trimming or longitude wrapping.

Graphic Objects and Return Values for Vector Inputs

- Vector coordinate array input (*x*-*y* or latitude-longitude pairs) with a `DisplayType` of `'Line'` or `'Point'` now generates an ordinary line object instead of a map graphics line.
- For geostruct input, an hggroup object is constructed; its handle is returned instead of an array of handles to map graphic objects:
  - For polygon geostructs, map graphics polygon objects are still constructed, but become children of the hggroup.

- For point, multipoint, and line geostructs, the children of the hggroup are ordinary line objects; map graphics objects are no longer constructed.

In both cases each child of the hggroup, rather than each element in an array of handles, corresponds to a distinct feature in the geostruct.

Handles Returned for Graphic Objects

- Geostruct inputs result in an hggroup handle containing either line objects (for point, multipoint, and line inputs) or modified patch objects (for polygon inputs) as their children.
- Coordinate arrays (*x*,*y* pairs) displayed as lines now result in ordinary line objects.
- Geostructs containing lines result as ordinary line objects within hggroups.

New Warnings Issued

- `mapshow` and `geoshow` now warn when given a geostruct within which the `Geometry` field differs from a specified `'DisplayType'` parameter.
- `mapshow` will warn if it is given a geostruct containing `Lat` and `Lon` fields instead of `X` and `Y` fields.
- `geoshow` will warn if it is given a geostruct containing `X` and `Y` fields instead of `Lat` and `Lon` fields.

geoshow Supplies Default Projection

`geoshow` now projects vector and raster inputs using a default projection (Plate Carree) if the parent axes is not a map axes. The axes itself is unchanged (it is not modified to become a map axes), but the scale factor of the projection is set such that latitudes and longitudes in degrees can be read directly from the axes ticks and grid lines.

Duplicate Parameter/Value Pair Inputs Allowed

`mapshow` no longer errors or warns if given duplicate Parameter/Value pair inputs; in such circumstances, `mapshow` now uses the last value (even with SymbolSpecs)

geoshow Supports True Surface Display

`geoshow` now creates a true 3-D surface if given a `'surface' DisplayType` rather than setting the `ZData` values to `0`.

Texturemap DisplayType Behavior Changes

The `'texturemap' DisplayType` now uses the pixel edges to create `XData` and `YData` grids rather than using the pixel centers, which correctly registers the display to map coordinates. The `ZData` contains an array of zeros having the same dimensions as the `XData` and YData `arrays`, which exceed the input grid in size by one in both the *x* and *y* dimensions.

You should use`'texturemap'` displays when the attribute being displayed is coded by color (i.e. 2-D displays); use `'surface'` displays when you need to show data with relief (nonzero `ZData`).

More General Support for Graphics Properties

All Handle Graphics® patch properties are now supported for polygon inputs.

All Handle Graphics line properties are now supported for point and line inputs, except that `'linestyle'` is ignored for point inputs.

Limitations on Referencing Matrices for Geoshow Removed

`geoshow` is now capable of accepting any referencing matrix. Previously it could only accept those referencing matrices that were convertible to referencing vectors.

mapshow and geoshow Ignore Empty Inputs Rather than Erroring

In previous versions, `mapshow` and `geoshow` would throw errors when provided with empty (`[]`) arrays. This behavior could be inconvenient when running these functions via scripts. The new behavior is also more consistent with that of MATLAB plotting functions such as `plot`, `surf`, `mesh`, and `contour`.

dted Automatically Fixes Incorrectly Specified Longitude Directions in DTED Data

Some DTED level 0 files available via the National Geospatial-Intelligence Agency's (NGA) web interface may have minor errors. Specifically, Level 0 data for cells just to the east of the prime meridian may have longitude coordinate strings with 'W' substituted for 'E'. The `dted` function now detects and automatically corrects this data error.

Version 2.3 (R2006a) Mapping Toolbox Software

This table summarizes what's new in Version 2.3 (R2006a):

New Features and Changes	Version Compatibility Considerations	Fixed Bugs and Known Problems	Related Documentation at Web Site
Yes Details below	Yes — Details in **"GeoTIFF Compatibility Considerations" on page 49** below. See also Summary.	Bug reports at Web site	Printable Release Notes: PDF Current product documentation

New features and changes introduced in this version are

Full Support for 64-Bit Windows

Version 2.3 adds support for the mex- and library-based functions `geotiffinfo`, `geotiffread`, `sdtsinfo`, and `sdtsdemread` on this new MATLAB platform via library upgrades (described below) and a custom port of STDS++.

Third-Party Library and Code Upgrades

Third-party libraries and software packages have been upgraded to their current versions to ensure best performance and compatibility with external geospatial data sources and applications software:

- General Polygon Clipper (GPC) upgraded to Version 2.32
- PROJ.4 library upgraded to Version 4.4.9
- SDTS++ library upgraded to Version 1.5.1
- GeoTIFF library upgraded to Version 1.2.2

Support for 32-Bit Floating-Point GeoTIFF Images

The MATLAB function `imread` can now import TIFF images containing 32-bit floating-point data. As a result, `geotiffread` now reads the corresponding variety of GeoTIFF.

GeoTIFF Compatibility Considerations

The structure returned by `geotiffinfo` in V. 2.3 has changed. The following table describes the differences between the current and previous versions:

Version 2.3	Previous Versions
The `TiePoints` structure contains two substructures, `ImagePoints` and `WorldPoints`. `ImagePoints` contains `[1-by-N]` arrays `Row` and `Col`; `WorldPoints` contains `[1-by-N]` arrays `X` and `Y`.	The `TiePoints` structure contained two `[3-by-1]` arrays, `ImagePoints` and `WorldPoints`.
The `CornerCoords` structure contains six `[1-by-4]` row vectors, respectively, `X`, `Y`, `Col`, `Row`, `Lat`, and `Lon`.	The `CornerCoords` structure contained six `[4-by-1]` column vectors: `PCSX`, `PCSY`, `X`, `Y`, `LAT`, and `LON`.
The `Zone` field contains `[]` if the UTM zone is not applicable or was missing from the metadata.	The `Zone` field contained `32767` if the UTM zone was not applicable or was missing from the metadata.

Utility Functions for NaN-Separated Polygons and Lines

closePolygonParts

Closes all rings in a multipart polygon to ensure proper analysis and rendering.

isShapeMultipart

Boolean-valued function that returns `true` if a polygon or line has multiple parts.

removeExtraNanSeparators

Eliminates redundant `NaN` separators that might exist in polygons and lines.

Standardized Vector Topology in coast.mat

Polygons in the low-resolution coastline sample data file `coast.mat` now follow the convention used by `geoshow`, `mapshow`, and `mapview` to display polygons with "holes" (inner rings representing lakes, inland seas, and islands within them). Outer contours now always run clockwise and inner contours run counterclockwise. These edits, which reversed the order of vertices in some rings, enable the display functions to fill outer rings properly while leaving inner rings blank.

Three New Demos

If you are viewing these release notes using the Help browser, clicking any of the demo links below will open the demo in a browser window. Click the links at the top of that window to view or run the M-code for the demo.

Converting Coastline Data (GSHHS) to Shapefile Format

Shows how to extract coastlines from the Global Self-consistent Hierarchical High-resolution Shorelines (GSHHS) data set, manipulate the polygon features, and save the result to a polygon shapefile.

Plotting a 3-D Dome as a Mesh Over a Globe

Illustrates how to construct a 3-D feature in a system of local vertical coordinates, then transform and combine it with a globe display in Earth-Centered, Earth-Fixed (ECEF) coordinates.

Unprojecting a Digital Elevation Model (DEM)

Shows how to unproject a georeferenced terrain elevation grid from Universal Transverse Mercator (UTM) into a regular latitude-longitude grid having comparable spatial resolution.

Version 2.2 (R14SP3) Mapping Toolbox Software

This table summarizes what's new in Version 2.2 (R14SP3):

New Features and Changes	Version Compatibility Considerations	Fixed Bugs and Known Problems	Related Documentation at Web Site
Yes Details below	Yes—Details in **"Compatibility Considerations" on page 53** below. See also Summary.	Bug reports at Web site	Current product documentation

New features and changes introduced in this version are

Geodetic-Geocentric Coordinate Conversion Functions

New three-dimensional coordinate conversion functions (`geodetic2ecef`, `ecef2geodetic`, `ecef2lv`, `lv2ecef`) transform 3-D point locations between geodetic (latitude, longitude, height), geocentric Cartesian (Earth Centered, Earth Fixed), and local vertical Cartesian coordinate systems.

Additional User Control Over Shapefile Content

Function `shapewrite` now allows user control over field names, lengths, and decimal precision when writing feature attributes to the DBF file, via a "DBF specification." The new function `makedbfspec` constructs a default DBF specification from a geographic data structure (geostruct2) array. Users can customize the output and pass it to `shapewrite`.

Shapefile Read/Write Efficiency Enhanced

Improved implementations of functions `shaperead` and `shapewrite` process data substantially faster (about four times faster for a 10-MB shapefile of major roads in Massachusetts).

Improved Rendering of Polygons with Inner Rings

The Map Viewer (function `mapview`) and functions `mapshow` and `geoshow` now properly render polygons containing inner rings (e.g., lakes and inland seas within a continent, islands within a pond). Features in underlying layers "show through" inner rings because they are not obscured by the patch faces used to render the polygons.

Compatibility Considerations

Polygon Vertex Ordering Is Now Significant for Properly Rendering Filled Polygons. The map display functions `geoshow`, `mapshow`, and `mapview` now require that coordinate vectors representing polygons have consistent directionality, such that

- Vertices defining outer rings (to be filled) be encoded in a *clockwise* direction.
- Vertices defining inner rings (often termed "lakes" or "islands," to be rendered as transparent holes) be encoded in a *counterclockwise* direction.

If you have vector map data sets that violate these conditions, the map display functions `geoshow`, `mapshow`, and `mapview` might not be capable of rendering them as filled polygons. To determine the directionality of polygon vertices, use the logical function `ispolycw`, which returns a separate result for each `NaN`-delimited polygon in an array of vertices. If you find inner rings which are clockwise or outer rings which are counterclockwise, use the utility functions `poly2ccw` or `poly2cw`, respectively, to reverse the direction of those rings.

Map Viewer Now Georeferences Images It Saves

When the Map Viewer saves the visible or selected area as a raster map (an image file), it now also writes a worldfile to georeference the image.

TIGER/Line File Support Upgraded

Function `tgrline` now supports the most recent (2003/2004) TIGER/Line data sets from the U.S. Bureau of the Census.

Version 2.1 (R14SP2) Mapping Toolbox Software

This table summarizes what's new in Version 2.1 (R14SP2):

New Features and Changes	Version Compatibility Considerations	Fixed Bugs and Known Problems	Related Documentation at Web Site
Yes Details below	Yes — Details labeled as **Compatibility Considerations** below. See also Summary.	Bug reports at Web site	Current product documentation

New features and changes introduced in this version are

New Function Reads Both 5-Minute and 2-Minute ETOPO Data

The new function `etopo` reads from either the 5-minute (ETOPO5) or the 2-minute (ETOPO2) global terrain data set. This function supersedes function `etopo5` and fixes several significant bugs.

Function gshhs Now Returns a Version 2 Geostruct

Function `gshhs`, which reads the Global Self-consistent Hierarchical High-resolution Shoreline data set, has been upgraded. It now returns a Version 2 geographic data structure (geostruct2) array instead of a Version 1 geostruct. Polygons returned from `gshhs` now follow the shapefile vertex-ordering convention (supported by functions `polybool`, `shaperead`, and `shapewrite`, for example). Under this convention the coordinates of outer rings (e.g., continent outlines) are given in clockwise order, while counterclockwise ordering is used for inner rings (e.g., lakes and inland seas within a continent). Note that function `gshhs` does not yet support Version 1.3 of the data set, released on Sept. 27, 2004.

Geodata Can Now Be Exported in Shapefile Format

The new function `shapewrite` writes a geographic data structure to a shapefile. It exports a Version 2 geographic data structure array (geostruct2), creating `.shp`, `.shx`, and `.dbf` files. Like `shaperead`, the function supports the Point, MultiPoint, PolyLine, and Polygon shape types. The contents of string-valued attribute fields and scalar numerical attribute fields are written to the dBase (`.dbf`) file.

Accessing Geodata Resources on the Internet

Links and URLs to documentation and data files for various Internet sources of digital map data are now collected in the following technical note on the MathWorks Web site:

```
http://www.mathworks.com/support/tech-notes/2100/2101.html
```

This technical note replaces many individual links formerly scattered across the User's Guide, reference pages, and M-file help. Collecting this information on a Web page rather than on product CDs or printed documentation should

substantially mitigate recurrent problems with stale links. Please report any stale links that you might find in the technical note to MathWorks Technical Support (`http://www.mathworks.com/contact_TS.html`), so that it can be updated promptly.

Compatibility Considerations for Atlas Data and Associated Functions

Through Version 2.0.3, the toolbox included a set of *atlas data* with global geopolitical data embedded as MATLAB arrays in four MAT-files: `worldlo`, `worldhi`, `worldmtx`, and `worldmtxmed`. However, geopolitical data is difficult to keep current, and is subject to inaccuracies and interpretations that can cause contention. Therefore, starting with Version 2.1, Mapping Toolbox demo data now excludes geopolitical data that would specify national sovereignty over specific regions of the Earth. The only exceptions are the boundaries of the 50 U.S. states and the District of Columbia.

This change means that the `worldlo`, `worldhi`, `worldmtx`, and `worldmtxmed` MAT-files are no longer part of the toolbox. However, the nonpolitical data on global coastlines, major lakes and inland seas, major rivers, and major cities and populated places that was in `worldlo.mat` has been retained in the toolbox and transformed into shapefile format. This includes the addition of name attributes for many previously unnamed features. There are four new shapefiles in this category: `landareas.shp`, `worldlakes.shp`, `worldrivers.shp`, and `worldcities.shp`.

For consistency, the atlas data for the United States that was originally stored in the `usalo` and `usahi` MAT-files has also been transformed, although none has been removed. These data sets now reside in the following shapefiles and MAT-files: `usastatelo.shp`, `usastatehi.shp`, `conus.mat`, and `greatlakes.mat`.

The toolbox originally included four functions dedicated to extracting data from the atlas data MAT-files: `worldlo`, `worldhi`, `usalo`, and `usahi`. With the data removal/transformation described above, these functions are no longer needed and have been removed from the toolbox in Version 2.1. You can easily access the new shapefiles using the `shaperead` function, which includes powerful and flexible options for selecting features and even controlling which attributes are read. In addition, function `country2mtx`, whose sole purpose

was to rasterize the country boundary polygons in `worldlo.mat`, has been removed.

Related changes extend to the `worldmap` function, which formerly combined two purposes:

- Select an appropriate map projection and parameters with which to display a given latitude-longitude area.
- Automatically display atlas data for that area.

In Version 2.1, `worldmap` supports only the first of these actions. A call to `worldmap` constructs a map axes object and can easily be followed with a variety of Mapping Toolbox commands to display the map data of your choice. Because the `usamap` function is so similar to `worldmap`, corresponding changes have been made there as well.

To help those who have relied heavily on `worldmap` and `usamap` to plot base maps with automatically selected vector map data, examples throughout the User's Guide, reference pages, and M-file Help entries have been updated to illustrate the new behavior of `worldmap` and `usamap`, and to show how to create maps including vector shapefile data layers. These examples cover a wide variety of ways to read and subset data with `shaperead` and display data with `geoshow` and other Mapping Toolbox display functions. A good place to start is with the examples for the `worldmap` and `usamap` functions. Also see example code in "Changes to worldmap and usamap Display Types" on page 59.

To help you update commands, scripts, and data for constructing and maintaining base maps, a recently published technical note on the MathWorks Web site provides links to data and documentation for many sources of vector and raster digital map data that you can access over the Internet:

`http://www.mathworks.com/support/tech-notes/2100/2101.html`

Compatibility Considerations for worldmap and usamap

`worldmap` and `usamap` have been simplified to construct appropriate map axes for a given area without displaying any map data.

In all cases, map frames, latitude-longitude grid lines, meridian labels, and parallel labels are turned on. You can use the following command sequence to remove them:

```
framem off; gridm off; mlabel off; plabel off
```

Other changes include the following:

- `usamap` now accepts two-letter U.S. Postal Service abbreviations for state names (e.g., AL, AK, AR, etc.).
- The following input options are now obsolete (if used, a warning is issued):
  - A first argument equal to `'lo'` or `'hi'`
  - The `regiononly` and `stateonly` syntax: a state or country name with the string `'only'` appended
  - All `type` options: `'none'`, `'line'`, `'lineonly'`, `'patch'`, `'patchonly'`, `'mesh'`, `'meshonly'`, `'dem'`, `'demonly'`, `'dem3d'`, `'dem3donly'`, `'lmesh3d'`, `'lmesh3donly'`, `'ldem3d'`, and `'ldem3donly'` (the new behavior matches the `'none'` option)

Changes to worldmap and usamap Display Types

As of this release, the `worldmap` and `usamap` functions no longer supports the *`type`* input argument. This argument provided an easy way to control display behavior.

The *`type`* option in `worldmap` was a single argument that could be one of the following strings: `'none'`, `'line'`, `'lineonly'`, `'patch'`, `'patchonly'`, `'mesh'`, `'meshonly'`, `'dem'`, `'demonly'`, `'dem3d'`, `'dem3donly'`, `'lmesh3d'`, `'lmesh3donly'`, `'ldem3d'`, and `'ldem3donly'`. In `usamap`, *`type`* was a subset of the above names (the 3-D options were not supported).

In the current release, the various *`type`* display options can be simulated by following a call to `worldmap` or `usamap` with an appropriate set of Mapping Toolbox commands. The following table specifies how you can achieve the effects of the old `worldmap` *`type`* argument using such auxiliary methods:

Mapping 1.x to 2.0.x Usage	Mapping 2.1 Usage

`load topo`
`worldmap(topo, ...`
`topolegend, 'dem')`
	`load topo`
	`worldmap(topo, topolegend)`
	`meshm(topo, topolegend)`
	`demcmap(topo)`
	`land = shaperead('landareas.shp',...`
	`'UseGeoCoords', true);`
	`geoshow([land.Lat], [land.Lon])`

`load topo`
`worlmdap(topo, ...`
`topolegend, 'demonly')`
	`load topo`
	`worldmap(topo, topolegend)`
	`meshm(topo, topolegend)`
	`demcmap(topo)`

`load topo`
`worldmap(topo, ...`
`topolegend, 'dem3d')`
	`load topo`
	`worldmap(topo, topolegend)`
	`meshm(topo, topolegend,size(topo),topo)`
	`da = daspect;`
	`pba = pbaspect;`
	`da(3) = 7.5*pba(3)/da(3);`
	`daspect(da);`
	`demcmap(topo)`
	`land = shaperead('landareas.shp',...`
	`'UseGeoCoords', true);`
	`geoshow([land.Lat], [land.Lon])`

Mapping 1.x to 2.0.x Usage	Mapping 2.1 Usage

`load topo`
`worldmap(topo, ...`
`topolegend, 'dem3donly')`
	`load topo`
	`worldmap(topo, topolegend)`
	`meshm(topo, topolegend,size(topo),topo)`
	`demcmap(topo)`

`load korea`
`worldmap (map, refvec,...`
`'mesh')`
	`load korea`
	`worldmap(map, refvec)`
	`meshm(map, refvec)`
	`land = shaperead('landareas.shp',...`
	`'UseGeoCoords', true);`
	`geoshow([land.Lat], [land.Lon])`
	(Text North Korea and South Korea will be missing)

`load korea`
`worldmap(map, refvec,...`
`'meshonly')`
	`load korea`
	`worldmap(map, refvec)`
	`meshm(map,refvec)`

`load topo`
`worldmap(topo,...`
`topolegend, 'mesh3d')`
	`load topo`
	`worldmap(topo, topolegend)`
	`meshm(topo, topolegend,size(topo),topo)`
	`da = daspect;`
	`pba = pbaspect;`
	`da(3) = 7.5*pba(3)/da(3);`
	`daspect(da);`

Mapping 1.x to 2.0.x Usage	Mapping 2.1 Usage

`load topo`
`worldmap...`
`(topo,topolegend,...`
`'mesh3donly')`
	`load topo`
	`worldmap(topo, topolegend)`
	`meshm(topo, topolegend,size(topo),top`

`load topo`
`worldmap(topo,...`
`topolegend,`
`'ldem3d')`
	`load topo`
	`worldmap(topo, topolegend)`
	`meshm(topo, topolegend,size(topo),topo)`
	`da = daspect;`
	`pba = pbaspect;`
	`da(3) = 7.5*pba(3)/da(3);`
	`daspect(da);`
	`demcmap(topo)`
	`camlight(90,5);`
	`camlight(0,5);`
	`lighting phong`
	`material([0.25 0.8 0])`

`load topo`
`worldmap(topo, ...`
`topolegend,'ldem3donly')`
	`load topo`
	`worldmap(topo, topolegend)`
	`meshm(topo, topolegend,size(topo),topo)`
	`da = daspect;`
	`pba = pbaspect;`
	`da(3) = 7.5*pba(3)/da(3);`
	`daspect(da);`
	`demcmap(topo)`

Mapping 1.x to 2.0.x Usage	Mapping 2.1 Usage

`load topo`
`worldmap(topo,...`
`topolegend,`
`'lmesh3d'`
	`load topo`
	`worldmap(topo, topolegend)`
	`meshm(topo, topolegend,size(topo),topo)`
	`da = daspect;`
	`pba = pbaspect;`
	`da(3) = 2*pba(3)/da(3);`
	`daspect(da);`
	`camlight(90,5);`
	`camlight(0,5);`
	`lighting phong`
	`material([0.25 0.8 0])`

`load topo`
`worldmap(topo,...`
`  topolegend,...`
`  'lmesh3donly')`
	`load topo`
	`worldmap(topo, topolegend)`
	`meshm(topo, topolegend,size(topo),topo)`
	`da = daspect;`
	`pba = pbaspect;`
	`da(3) = 2*pba(3)/da(3);`
	`daspect(da);`

Mapping 1.x to 2.0.x Usage	Mapping 2.1 Usage

`load korea`
`worldmap(map, refvec,...`
	`load korea`
	`worldmap(map, refvec)`
	`land =...`
	`shaperead('landareas.shp',...`
	`'UseGeoCoords', true);`
	`geoshow([land.Lat], [land.Lon])`
	(Text North Korea and South Korea will be missing. Land area boundaries resolution is lower.)

`load korea`
`worldmap(map, refvec,...`
`'lineonly')`
	`load korea`
	`worldmap(map, refvec)`
	`land = ...`
	`shaperead('landareas.shp',...`
	`'UseGeoCoords', true);`
	`geoshow([land.Lat], [land.Lon])`

`load korea`
`'line')`
`worldmap(map, refvec,...`
`'none')`
	`load korea`
	`worldmap(map, refvec)`

Mapping 1.x to 2.0.x Usage	Mapping 2.1 Usage

`load korea`
`worldmap(map, refvec,...`
`  'patch')`
	`load korea`
	`worldmap(map, refvec)`
	`land = ...`
	`shaperead('landareas.shp',...`
	`'UseGeoCoords', true);`
	`faceColors = ...`
	`makesymbolspec('Polygon',...`
	`{'INDEX', [1 numel(land)],...`
	`'FaceColor', ...`
	`polcmap(numel(land))});`
	`geoshow(land,'SymbolSpec',...`
	`makesymbolspec('Polygon',...`
	`faceColors)`
	(Text North Korea and South Korea will be missing. Country coloring will be missing.)

`load korea`
`worldmap(map, refvec,...`
`'patchonly')`
	`load korea`
	`worldmap(map, refvec)`
	`land = ...`
	`shaperead('landareas.shp',...`
	`'UseGeoCoords', true);`
	`faceColors =...`
	`{'INDEX', [1 numel(land)],...`
	`'FaceColor', ...`
	`polcmap(numel(land))});`
	`geoshow(land,'SymbolSpec',...`
	`faceColors)`
	(Country coloring will be missing.)

Data Files Added in This Release

The following files were added to the `mapdemos` directory, for use in toolbox demos and examples:

- `landareas` — Polygon shapefile: global coastlines, both exterior and interior, including names for larger land masses
- `worldlakes` — Polygon shapefile: coastlines and names of major lakes and inland seas worldwide
- `worldrivers` — PolyLine shapefile: major world rivers and their names
- `worldcities` — Point shapefile: locations and names of major cities and populated places worldwide
- `usastatelo` — Polygon shapefile: low-resolution outlines and names of the 50 U.S. states plus D.C.
- `usastatehi` — Polygon shapefile: moderate-resolution outlines and names of the 50 U.S. states plus D.C.
- `conus` — MAT-file: Low-resolution latitudes and longitudes, in degrees, for the perimeter of the conterminous United States (CONUS), the Great Lakes, and interstate borders
- `greatlakes` — MAT-file: A Version 1 geographic data structure (geostruct1) with outlines and names for the Great Lakes of North America

Atlas Data MAT-Files Removed in This Release

MAT-files containing Atlas Data have been removed in Version 2.1. Some of the data has been retained in a different form. The disposition of these data sets and variables is described below.

World MAT-File Data

- `worldlo.mat`, which contained the following variables:
  - `DNline` — Data moved to `worldrivers.shp`
  - `DNpatch` — Data moved to `worldlakes.shp`
  - `POline` — Data removed from toolbox
  - `POtext` — Data removed from toolbox

  - `PPpoint` — Data moved to `worldcities.shp`
  - `PPtext` — Data moved to `worldcities.shp`
  - `gazette` — Data removed from toolbox
- `worldhi.mat` — Data removed from toolbox
- `worldmtx.mat` — Data removed from toolbox
- `worldmtxmed.mat` — Data removed from toolbox

United States MAT-File Data

- `usalo.mat`, which contained the following variables (all retained):
  - `conus` — Data moved to `conus.mat`
  - `greatlakes` — Data moved to `greatlakes.mat`
  - `gtlakelat` — Data moved to `conus.mat`
  - `gtlakelon` — Data moved to `conus.mat`
  - `state` — Data moved to `usastatelo.shp`
  - `stateborder` — Data moved to `conus.mat`
  - `statelat` — Data moved to `conus.mat`
  - `statelon` — Data moved to `conus.mat`
  - `uslat` — Data moved to `conus.mat`
  - `uslon` — Data moved to `conus.mat`
- `usahi.mat` — Data moved to `usastatehi.shp`

Functions Deleted in This Release

The following functions, which performed specific operations on Atlas Data sets, have been removed in Version 2.1:

- `country2mtx` — Create a raster map grid of a country from `worldlo` atlas data
- `usahi` — Return high-resolution vector atlas data for the United States
- `usalo` — Return vector atlas data for the United States

- `worldhi` — Return high-resolution vector atlas data for the world
- `worldlo` — Return vector atlas data for the world or oceans

Functions Made Obsolete in This Release

- `etopo5` — Replaced by `etopo`. Use `etopo` instead.
- `tigerp` — Download U.S. Census cartographic boundary files in shapefile format and use `shaperead` instead.
- `tigermif` — Download U.S. Census cartographic boundary files in shapefile format and use `shaperead` instead.

Version 2.0.3 (R14SP1) Mapping Toolbox Software

This table summarizes what's new in Version 2.0.3 (R14SP1):

New Features and Changes	Version Compatibility Considerations	Fixed Bugs and Known Problems	Related Documentation at Web Site
Yes Details below	Yes Details in **"Compatibility Considerations" on page 70** below. See also Summary.	Fixed bugs	Current product documentation

The new features and changes introduced in this version are

Polygon Vertex Ordering Functions

Four new functions have been added to Mapping 2.0.3. These are called by or can be used in conjunction with the `polybool` function (see "polybool Revised, Has New Helper Functions" on page 70.) Three of the functions check or change the ordering of vertices that define a polygon, and the fourth one splits polygons with holes in a consistent fashion.

These functions are

- `ispolycw` — `True` if vertices of polygonal contour are clockwise ordered
- `poly2cw` — Convert polygonal contour to clockwise ordering
- `poly2ccw` — Convert polygonal contour to counterclockwise ordering
- `poly2fv` — Convert polygonal region to face-vertex form for use with `patch` in order to properly render polygons containing holes

Compatibility Considerations

axesm Does Not Reorder Longitude Limits

Function `axesm` now requires that `'MapLonLimit'` values have the form `[west east]`, but east can be less than west numerically (e.g., `[120 -60]`). The form `[east west]` is no longer supported.

contourm Returns Contourgroup Handle

Function `contourm` now returns a handle to a contourgroup object instead of an array of handles to line objects.

elevation Has New Azimuth Output

Function `elevation` can now return a third output, the line-of-sight azimuth corresponding to the elevation angle (output 1) and slant range (output 2). This eliminates the need to use a geodesic or rhumb line azimuth, computed with functions `azimuth` or `distance`, as an approximate substitute.

gtopo30 Has New Options

The functionality of `gtopo30` has been extended so that:

- Data can be read from a more general directory structure.
- Empty may be supplied in place of any input argument.
- Latitude and longitude limits are no longer restricted to lie within a tile. Instead, areas not covered by available data are automatically filled with `NaN`s.

inputm Has New Button Output

`inputm` now has an optional third output argument, `'button'`, which functions like the third output of `ginput`.

polybool Revised, Has New Helper Functions

Mapping 2.0.3 fixes major problems with the `polybool` function, which computes set operations on polygonal regions. The fixes include incompatible data structure and syntax changes.

- The `'cutvector'` option has been removed from the syntax. This option facilitated plotting of polygonal regions containing holes, but the cutvector computation was not robust, nor was it an effective way to represent polygonal regions containing multiply nested holes and/or discontiguous regions. To replace the cutvector functionality, a new function, `poly2fv`, has been added. `poly2fv` converts a polygonal region, possibly including holes and discontiguous regions, into a faces and a vertices matrix that can be used with the patch function to display the region. The M-file help for `polybool` and `poly2fv` contain examples illustrating how to use `poly2fv`.
- The `'cell'` and `'vector'` options have been removed from the syntax. `polybool` now returns the output polygonal region using the same format as the input polygonal region. Use `polysplit` and `polyjoin` to convert from one format to the other.
- `polybool` now uses the "clockwise" rule to distinguish between external polygonal contours and internal polygonal contours. This is the same rule used by ESRI shapefiles. `polybool` assumes that a polygonal contour whose vertices are arranged in clockwise order is an external contour, and that a polygonal contour whose vertices are arranged in counterclockwise order is an internal contour. Use `ispolycw` to determine whether a polygonal contour is clockwise ordered, and use `poly2cw` and `poly2ccw` to convert polygonal contours to be clockwise or counterclockwise ordered, respectively. If a polygonal region input has no external contours according to this rule, `polybool` issues a warning message.

An input polygonal region can either take the form of a pair of `NaN`-separated vectors, or it can take the form of a pair of cell arrays. In the cell array format, each cell must contain the vertices of a single polygonal contour. Representing a polygonal region as a cell array whose cells can contain `NaN`-separated vectors is no longer supported, and an error message will be issued.

Summary of new polybool-related helper functions. You can use the helper functions that support `polybool` for your own purposes. They are

- `ispolycw` — True if vertices of polygonal contour are clockwise ordered
- `poly2cw` — Convert polygonal contour to clockwise ordering
- `poly2ccw` — Convert polygonal contour to counterclockwise ordering
- `poly2fv` — Convert polygonal region to face-vertex form for use with `patch`

Summary of known problems addressed. The issues solved by the new implementation of `polybool` are

- `polybool` no longer errors out or produces incorrect results for proper inputs.
- `polybool` now produces consistent output when computing intersections, with results no longer depending on the order of its inputs.
- `polybool` now operates correctly on polygons with collinear edges.

polymerge Output Vertex Order Changes

Improvements in the algorithm that `polymerge` uses to chain together adjacent polygon or line segments, required to address a bug and improve efficiency, in some cases also change the ordering of the output vertices.

Version 2.0.2 (R14) Mapping Toolbox Software

This table summarizes what's new in Version 2.0.2 (R14):

New Features and Changes	Version Compatibility Considerations	Fixed Bugs and Known Problems	Related Documentation at Web Site
Yes Details below	No	No bug fixes	Current product documentation

New features and changes introduced in this version are

Mapping Toolbox Compilability Enhanced

Almost all Mapping Toolbox command line functions can be compiled using the new MATLAB Compiler. However, large Atlas Data MAT-files need to be added explicitly to use them in applications.

SDTS Functions Now Available on HP-UX

The input functions `sdtsinfo` and `sdtsdemread` are now available on the HP-UX platform.

Version 2.0.1 (R13SP1+) Mapping Toolbox Software

This table summarizes what's new in Version 2.0.1 (R13SP1+):

New Features and Changes	Version Compatibility Considerations	Fixed Bugs and Known Problems	Related Documentation at Web Site
Yes Details below	No	Fixed bugs, below	V2.0.1 product documentation

The new feature introduced in this version is

mapexreg Demo (Georeferencing an Image to an Orthotile Base Layer) Generalized and Clarified

Calls to `imtransform` and `makerefmat` are no longer hard-coded for an output pixel size of one meter. The choice of output pixel size now is explained in detail.

Fixed Bugs

Version 2.0.1 contains the following bug fixes:

almanac

Function `almanac` now gives a precise inverse flattening (1/f) value for Bessel 1841 ellipsoid. Previously, `almanac` used a truncated value (299.1528) instead of the full precision value (299.1528128).

axesm

Function `axesm` now accepts the `'meridianlabel'` parameter. Previously, `axesm` would throw an error if the `'meridianlabel'` parameter was specified.

distance

Function `distance` now computes correct results for rhumb line distances along a parallel (including the equator) on a (nonspherical) ellipsoid.

distance

Function `distance` now uses the robust "aversine formula" to compute great circle distances on a sphere. For certain inputs roundoff sensitivities in the previous implementation resulted in a small, but nonzero, distance between identical points.

distance

Function `distance` now correctly obtains correct results for rhumb lines that follow the equator, or any other parallel, and cross the 180-degree meridian; it gives the distance along the short arc of the parallel. Previously the length of the long arc could be returned instead.

distance, azimuth

Functions `distance` and `azimuth` now produce accurate results even for long geodesics on an ellipsoid, up to near-antipodal distances.

projfwd, projinv

Functions `projfwd` and `projinv` now use `NaN` separators on all platforms. Previously, under certain circumstances, `projfwd` and `projinv` returned coordinate arrays with `Inf` separators rather than `NaN`s.

reckon

Function `reckon` now supports geodesics on an ellipsoid, in addition to rhumb lines on an ellipsoid and both geodesics and rhumb lines on a sphere.

Version 2.0 (R13SP1+) Mapping Toolbox Software

This table summarizes what's new in Version 2.0 (R13SP1+):

New Features and Changes	Version Compatibility Considerations	Fixed Bugs and Known Problems	Related Documentation at Web Site
Yes Details below	Yes — Details in **"Compatibility Considerations" on page 88** below. See also Summary.	"Fixed Bugs" on page 90, below	V2.0.1 product documentation

New features and changes introduced in this version are

Release Summary

Mapping Toolbox Version 2.0 provides a comprehensive set of functions and graphical user interfaces for building map displays and performing geospatial data analysis in the MATLAB environment. In this major release of the toolbox, new functions and capabilities include:

- Support for standard GIS and geospatial file formats:
  - ESRI shapefile; GeoTIFF; Arc ASCII Grid; SDTS Raster Profile (read only)
  - "Worldfiles" for spatially registered images (read and write)
- New display functions for vector features, georeferenced imagery, and data grids in *x-y* map coordinates (as well as unprojected latitude-longitude)
- New functions for vector symbolization based on feature attributes
- Transverse Mercator projection and PROJ.4 projection library support
- A new interactive Map Viewer with the following features:
  - Support for multiple raster and vector layers
  - Data import from file or workspace
  - Zooming/panning/map navigation tools
  - Cursor coordinate and map-scale readout
  - Data tip and info tools
  - Graphical overlays
  - Map exporting/printing
- A major update of the Mapping Toolbox User's Guide, focused on clarifying concepts and terminology, explaining features and functions, adding new and enhancing existing examples, and organizing and cross-referencing material to make the documentation easier to access

New Demos and Sample Data

Six new Mapping Toolbox demos are now, available from the **Demos** tab of the Help Browser. You can also access this demos page by typing

```
mapdemos
```

The demos consist of the following:

- mapexenhance — Enhancing Multispectral GeoTIFF Images
- mapexfindcity — Interactive Global City Finder
- mapexgeo — Creating Maps using `geoshow` (for latitude, longitude data)
- mapexmap — Creating Maps using `mapshow` (for x, y data)
- mapexrefmat — Creating and Using Referencing Matrices
- mapexreg — Georeferencing an Image to an Orthotile Base Layer
- viewmaps — GUI Demonstrating Map Projections (not new in 2.0)

Note that the above commands run the demo scripts to produce figures, whereas `mapdemos` describes and illustrates the demos in the Help Browser.

In addition, a number of new sample data sets containing vector features, digital elevation models, and georeferenced images have been added for use in demos and examples. The new data includes satellite images of the Boston area, topographic grids of the White Mountains in New Hampshire, and vector data for roads and hydrographic features in the Boston area. The data sets are provided in standard geospatial and GIS formats: GeoTIFF, TIFF images with worldfiles, SDTS Raster Profile, Arc ASCII Grid, and ESRI shapefiles. They are accompanied by ASCII text files (with suffix `.txt`), containing descriptive metadata.

The new data files are in addition to existing sample data sets stored in MAT-files, such as `geoid`, `russia`, and `korea`. In addition, metadata indicating the source and describing the contents of most sample data and atlas data has been added to their respective MAT-files.

To read the metadata for a Mapping Toolbox MAT-file data set, `load` it and inspect the `source` and `description` workspace variables.

To read brief descriptions of the demos and sample data sets, type

```
help mapdemos
```

New Map Viewer

The toolbox includes a new interactive tool for displaying and examining map data, called the Map Viewer. The Map Viewer helps you work with data that is already in a projected map coordinate system, which is the case for many high-resolution satellite and aerial imagery products, as well as many vector map data sets that cover small areas of the Earth in substantial detail. The Map Viewer gives you a view (or views) of an *x-y* map coordinate plane where, for example, *x* and *y* may correspond to the easting and northing coordinates of a given UTM or State Plane zone. If some of your data is referenced to geographic (latitude-longitude) coordinates, you first need to use an appropriate projection to transform it to map coordinates.

Key features of the Map Viewer include:

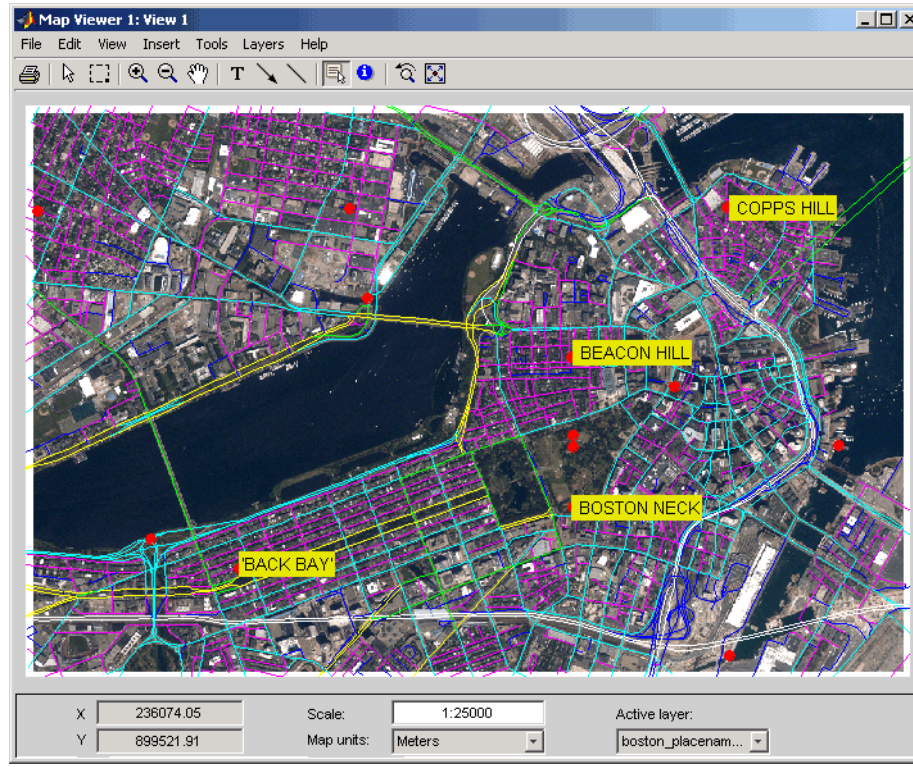
- A two-dimensional view of the *x-y* map plane with interactive tools for navigating in that plane: magnification, de-magnification, panning, setting the map scale, and zooming the view to the extent of a given data set, or all data sets
- Vector and image data import from files or from the MATLAB workspace
- Management of each data set in a separate layer. You can control layer ordering and visibility to bring what you need into view.
- Multiple, simultaneous views (differing in scale or view extent) of the same data layers
- A data tip tool to identify vector shapes or query a feature attribute that you specify
- An info tool to display, in a separate window, all the attributes of a selected feature
- Tools to annotate the map with lines, arrows, and text
- Ability to export a raster snapshot of the map view

To start the Map Viewer, use the `mapview` function. Typing

```
mapview
```

will open a Map Viewer window. You can then import standard format files or load workspace data into the Map Viewer. The following figure shows the

Map Viewer's main window after loading image, line, and point data sets, also illustrating user-specified symbology for streets and landmarks, and Data Tip labels for selected point features:



You can print maps as they appear in the Map Viewer window, to either a printer or a file. Either click the **Print** button on the left end of the toolbar or select **Print** in the **File** menu. A standard figure print dialog appears. You should consider selecting a driver appropriate to your printer; this may be necessary if you attempt to print in color.

Spatially Referenced Image Formats

The toolbox adds support for several industry-standard formats for spatially referenced images and data grids: Arc ASCII Grid files, GeoTIFF files, and worldfiles. You do this by using the following new functions: `arcgridread`,

`geotiffinfo`, `geotiffread`, `getworldfilename`, `worldfileread`, and `worldfilewrite`.

The grid and image file-reading functions construct referencing matrices, enabling a general and consistent approach to managing the relationship between map coordinates and pixel columns and rows (see below).

Working with Images Referenced to Map Coordinates

The toolbox includes a family of functions supporting images and gridded data sets that are referenced to two-dimensional projected map coordinates. These functions represent the relationship between a point at position (`row`,`col`) in an image and point in map coordinates (x,y) with a *referencing matrix*. A referencing matrix is a 3-by-2 matrix, `R`, that represents a general affine transformation such that

```
[x y] = [row col 1] * R
```

Most often the image (or data grid) consists of square pixels and has edges that align with the map coordinate axes, but you are not limited to this situation.

The new function `makerefmat` provides several options for constructing referencing matrices from scratch. See its M-file help for further information. However, you can most often obtain a referencing matrix from a world file, via `worldfileread`, or from a GeoTIFF file via `geotiffread`. If you have created a new referencing matrix for an image in a generic image format such as TIFF or JPEG, you can save this information in a world file using `worldfilewrite`.

New functions `pix2map` and `map2pix` implement the basic transformation defined above, along with its inverse. With functions `mapbbox` and `mapoutline` you can calculate the image bounds or perimeter in map coordinates. You can use the function `pixcenters` to calculate the coordinates of the center of each image pixel. This is helpful if you have a data grid that you want to display using `surface`.

The new function `mapshow` accepts a referencing matrix in order to correctly position an image in a standard Handle Graphics axes (i.e., not a map axes).

Because the affine transformation is extremely general, it can equally well represent the registration of an image to geographic coordinates, with longitude taking the place of map `x` and latitude replacing map `y`:

```
[lon lat] = [row col 1] * R
```

A referencing matrix is a more general version of the 1-by-3 map legend vector, perhaps better characterized as a *referencing vector*, that many toolbox functions already use. To convert a referencing vector to a referencing matrix, you can use function `refvec2mat`. The inverse transformation exists only for referencing matrices having a special form, but if it does you can compute it with `refmat2vec`. Functions `latlon2pix` and `pix2latlon` support transformations for referencing matrices in geographic coordinates, allowing for longitude-wrapping differences.

Use the new function `geoshow` along with map axes to display data grids referenced to latitude and longitude via either a referencing vector or a referencing matrix. Or, you can use the new function `grid2image` to quickly display an unprojected grid with latitude as the ordinate (*x*-axis) and longitude as the abscissa (*y*-coordinate).

SDTS Terrain Data Access

Mapping Toolbox functions can now import terrain grids stored in United States Spatial Data Transfer Standard (SDTS) raster profile format, giving you access to a wide selection of public domain terrain data sets, principally those prepared by the U.S. Geological Survey (USGS). USGS DEM data is no longer distributed from government Internet sites. Instead, these and other USGS files are now available from a commercial server at no charge. For further information, see `http://edc.usgs.gov/products/elevation/dem.html` and `http://edc.usgs.gov/geodata/`.

Two new toolbox functions support SDTS data handling:

- `sdtsinfo` returns a structure containing metadata for an SDTS data set.
- `sdtsdemread` returns elevation data and a referencing matrix from an SDTS DEM data set.

For details, type `help sdtsinfo` or `help sdtsdemread` at the MATLAB prompt.

Shapefiles and Vector Features

The toolbox now reads the shapefile format defined by Environmental Systems Research Institute (ESRI) and widely used as a data exchange format for nontopological GIS data. You can query a shapefile for its shapetype, attribute names and types, number of records, etc., using `shapeinfo`, and read 2-D shapes using `shaperead`. The `shaperead` function constructs a *geostruct2* geographic data structure array (see "Geographic Data Structure Enhancements" on page 83), tailored to the contents read in from shapefiles.

The shape data is represented in the MATLAB workspace as a 1-D structure array, with one element per feature. Attribute values are stored in structure fields, or in a separate, parallel structure array. Because attributes are defined according to the needs of the shapefile author and might be unknown until the time a given shapefile is read, the new Mapping Toolbox geographic data structures (described below) are necessarily more general than the existing toolbox geographic data structures.

Geographic Data Structure Enhancements

Certain Mapping Toolbox functions introduced in Version 2 use an enhanced geographic data structure layout (called a *geostruct2*) to store and manipulate vector geodata. This layout has the flexibility to store any kind and number of attributes, and handles either geographic (latitude and longitude) or plane (x and y) coordinates. In contrast, a Version 1 geographic data structure (*geostruct1*) — which is still supported — is limited to a fixed set of fields and can contain geographic coordinates only.

One way to create a Version 2 geographic data structure is to input vector geodata to the workspace from a shapefile. The function `shaperead` returns a geostruct2 that encapsulates some or all of the data stored in a shapefile and its supporting index and dBASE files. To determine what kinds of data a shapefiles contains, you can use the `shapeinfo` function to query it.

The new functions `mapshow`, `geoshow`, and `mapview` each display vector data in geostruct2 form. Use `geoshow` when `Lat` and `Lon` coordinate fields exist in the geostruct, and use `mapshow` or `mapview` when `X` and `Y` fields are present.

You can transform a geostruct1 into a geostruct2 (but not the reverse). Use the function `updategeostruct` for this purpose. See "Mapping Toolbox

Geographic Data Structures" in the Mapping Toolbox documentation for descriptions of geostruct formats and uses.

Another new function, `extractfield`, lets you conveniently combine all the values of a given geostruct field into a single array. For example, given a shapefile with a `'Name'` attribute, `extractfield` can output a cell array of feature names from the geostruct returned by `shaperead`.

Attribute-Driven Vector Data Symbolization

When you display shape features as points, lines, or polygons using `mapview`, `mapshow`, and `geoshow`, you can make feature attributes control how they are symbolized. This lets the data control graphic attributes such as color, line width, marker symbol, and visibility on a per-feature basis. In prior versions of the toolbox, attribute-specific symbology was difficult to accomplish, but now a general mechanism exists for this purpose. The new function `makesymbolspec` generates a structure called a *symbolspec* that maps specific attribute values (or ranges) to symbology parameters, and lets you specify default parameters for unspecified values. For example, the following symbolspec colors all roads of `class` 1 red, all roads of `class` 2 green, all roads of `class` 3 blue, and defaults other classes to black (where `class` is an attribute of the road layer):

```
roadColors = makesymbolspec('Line',{'CLASS',1,'Color','r'},...
                                   {'CLASS',2,'Color','g'},...
                                   {'CLASS',3,'Color','b'},...
                                   {'Default','Color','k'});
```

To see an example of what you can do with symbolspecs, look at the illustration for "New Map Viewer" on page 79. For details, type `help shaperead`, `help mapshow`, `help geoshow`, `help mapview`, or `help makesymbolspec` at the MATLAB prompt.

Access to PROJ.4 Map Projections

Mapping Toolbox software now incorporates the PROJ.4 cartographic projections library, originally written by Gerald Evenden of the USGS. Currently the toolbox uses this library primarily to support the GeoTIFF format. You can access selected PROJ.4 projections and their inverses using new functions `projfwd` and `projinv`. Use the `projlist` function to learn

about which PROJ.4 projections are available. In addition to accepting GeoTIFF `info` structures, `projfwd` and `projinv` also work with a standard map projection structure (mstruct) used to define projections within map axes.

Minor Enhancements

In addition to the major new features just described, this release includes the following enhancements and new functions, among others.

almanac

Now accepts `'ellipsoid'` as a synonym for `'geoid'` as a parameter or refbody value.

convertlat

A new function that unifies the latitude conversions previously dispersed among functions `geod2aut`, `aut2geod`, etc. For a list of the files it replaces, see "Functions Obsolete in Version 2.0" on page 88.

geoloc2grid

New function for converting a geolocated data grid (general matrix map) to a regular gridded data set (matrix map). Easier to use for this purpose than `imbedm`, and produces a smoother output with 2-D resampling.

grid2image

New function to display a regular data grid, with a referencing matrix or referencing vector, as an image. The grid can contain `double`, `uint8`, or `uint16` values. `grid2image` replaces the obsoleted function `imagem`.

tranmerc

A new map projection, Transverse Mercator, has been added. This is a general Transverse Mercator, as opposed to the more specific Universal Transverse Mercator supported by the `utm` function.

unitsratio

Provides conversion factors among units of length, and between radians and degrees. Supports a wider range of length units than `distdim`.

For information on using any of these functions, type `help` *`function`* at the MATLAB prompt.

Summary of Functions Added in Version 2.0

Version 2.0 features many new functions, all of which are itemized below. Several existing functions have been deprecated or deleted. These are described in "Functions Obsolete in Version 2.0" on page 88 and "Functions Deleted in Version 2.0" on page 89.

Geospatial Data Import and Access

- `arcgridread` — Read a gridded data set in Arc ASCII Grid Format.
- `geotiffinfo` — Information about a GeoTIFF file
- `geotiffread` — Read a georeferenced image from GeoTIFF file.
- `getworldfilename` — Derive a worldfile name from an image file name.
- `sdtsdemread` — Read data from an SDTS raster/DEM data set.
- `sdtsinfo` — Information about an SDTS data set
- `shapeinfo` — Information about a shapefile
- `shaperead` — Read vector feature coordinates and attributes from a shapefile.
- `worldfileread` — Read a worldfile and return a referencing matrix.
- `worldfilewrite` — Construct a worldfile from a referencing matrix.

Vector Map Data and Geographic Data Structures

- `extractfield` — Extract the field values from a structure.
- `updategeostruct` — Update a geographic data structure.

Spatial Referencing of Georeferenced Images and Data Grids

- `latlon2pix` — Convert latitude-longitude coordinates to pixel coordinates.
- `makerefmat` — Construct an affine spatial-referencing matrix.
- `map2pix` — Convert map coordinates to pixel coordinates.
- `mapbbox` — Compute bounding box of a georeferenced image or data grid.
- `mapoutline` — Compute outline of a georeferenced image or data grid.
- `pix2latlon` — Convert pixel coordinates to latitude-longitude coordinates.
- `pix2map` — Convert pixel coordinates to map coordinates.
- `pixcenters` — Compute pixel centers for georeferenced image or data grid.
- `refmat2vec` — Convert a referencing matrix to a referencing vector.
- `refvec2mat` — Convert a referencing vector to a referencing matrix.
- `geoloc2grid` — Convert a geolocated data array to a regular data grid.

Map Projection Properties and Transformations

- `geotiff2mstruct` — Convert GeoTIFF info to a map projection structure.
- `projlist` — List map projections supported by `projfwd` and `projinv`.
- `projfwd` — Forward map projection using the PROJ.4 library
- `projinv` — Inverse map projection using the PROJ.4 library
- `tranmerc` — Transverse Mercator Projection

Map Display and Interaction

- `geoshow` — Display map latitude and longitude data.
- `grid2image` — Display a regular data grid as an image.
- `mapshow` — Display map data.
- `mapview` — Interactive map viewer
- `makesymbolspec` — Construct a vector symbolization specification.

Geographic Calculations

- `convertlat` — Convert between geodetic and auxiliary latitudes.

Utilities

- `ind2rgb8` — Convert an indexed image to a `uint8` RGB image.
- `unitsratio` — Unit conversion factors

Compatibility Considerations

The following changes affect compatibility between Version 2.0 and previous versions of Mapping Toolbox software.

Functions Obsolete in Version 2.0

The following functions are still available but should no longer be used:

- `imagem`, which displayed a regular matrix map as an image, has been replaced by `grid2image`.
- The following 12 latitude conversion functions have been replaced by a single utility function, `convertlat`:
  - `aut2geod`
  - `cen2geod`
  - `cnf2geod`
  - `iso2geod`
  - `par2geod`
  - `rec2geod`
  - `geod2aut`
  - `geod2cen`
  - `geod2cnf`
  - `geod2iso`
  - `geod2pa`

- `geod2rec`

This improves the stability of numerical results in certain projections.

Functions Deleted in Version 2.0

The following Mapping Toolbox functions are no longer available.

- `coast` — This function simply loaded the `coast` MAT-file containing world coastlines. Instead of calling it, type

  ```
  load coast
  ```

- `loadmoonalb` — This function simply loaded the `moonalb` MAT-file of the Moon's albedo. Instead of calling it, type

  ```
  load moonalb
  ```

- `maskm` — This function, which reassigned a scalar value to an array based on a Boolean mask, has been removed. Ordinary MATLAB command syntax does the same thing.
- `movescale` — This undocumented function now is a subfunction of `scaleruler`.

Changes in Nomenclature

To achieve greater consistency with the literature on geospatial data handling, Mapping Toolbox documentation for Version 2.0 has changed the usage of certain terms and names of variables provided as sample data. The primary changes are described below.

Geoid — Where it is used to describe the geometric shape of the Earth (an equipotential surface), this term has been retained. Where it was used as a synonym for ellipsoid, occurrences of *geoid* have been changed to *ellipsoid*. This includes changing references to *geoid vector* to *ellipsoid vector*. You also now specify ellipsoid models for `almanac` data using the keyword '`ellipsoid`' (however, `'geoid'` still works). Note that a similar update has *not* been made for `axesm`, `getm`, or `setm`, which also use `'geoid'` as a keyword to identify the `geoid` field of the map projection structure (mstruct).

Map — Where this term referred to data sets (either raster or vector), occurrences of *map* have been changed to *data grid*, *data set*, or some other appropriate term. Where it refers to a cartographic presentation of geodata, *map* has been retained.

Map legend — This term, which used to refer to a three-element vector that georeferenced a data grid, has been replaced with *referencing vector*. In some contexts (when more degrees of freedom are involved), the term *referencing matrix* replaces it.

Most sample data sets that contain `map` and `maplegend` as variables (data grids and referencing vectors, respectively) have been updated. For example, the `geoid` MAT-file now has variables `geoid` (the data grid), and `geoidrefvec` and `geoidlegend` (referencing vectors; the second is a copy of the first, provided to maintain compatibility). This file, along with most other sample data except for `topo` and `coast`, also now contains metadata in the form of `source` and `description` strings.

Fixed Bugs

Mapping Toolbox Version 2.0 includes the following bug fixes:

sizem

`sizem` now respects the order of its longitude limits. Now, for example, `[r,c] = sizem([-5 5],[170 -170],1)` sizes a 10-by-20 grid that crosses the 180-degree meridian, rather than a 10-by-340 grid that extending from -170 all the way to 170. `maptriml`, `maptrimp`, and `maptrims` now respect the order of their longitude limits as well.

Fixed Error in Inverse UTM Projection

This release corrects an error in the inverse UTM projection. This error caused mislocations that ranged from negligible near the central meridian to several meters at the east-west zone boundaries.

This release also removes the rounding that decreased precision of the latitude-longitude outputs.

Improved Numerical Behavior of Geodetic-to-Conformal Latitude Conversion

A new formulation, using the new `convertlat` function, fixes subtle inaccuracies near the poles that could cause unexpected, hard-to-explain behavior in the stereographic projection.

Line-of-Sight Computation Corrected for Observer at Zero Elevation

A problem that caused `los2` and `viewshed` to report some visible points as invisible has been fixed.

Version 1.3.1 (R13SP1) Mapping Toolbox Software

This table summarizes what's new in Version 1.3.1 (R13SP1):

New Features and Changes	Version Compatibility Considerations	Fixed Bugs and Known Problems	Related Documentation at Web Site
No	No	Fixed Bugs, below	No

Fixed Bugs

Mapping Toolbox Version 1.3.1 contains the following bug fixes:

almanac

Minor typographical errors in defining the parameters for the Clarke 1866 and Clarke 1880 ellipsoids have been fixed.

areaint

Every simple polygon divides the Earth's surface into two regions. Now `areaint` consistently returns the area of the smaller region.

avhrrgoode

No longer fails due to an illegal assignment of `NaN` to class logical (mitigated side effect of new logical class in Release 13)

axesmui

Now updates frame limits when the projection is changed

demdataui

No longer disables zoom mode

dted

The calculation of directory names is now correct, even for longitudes between -10 and +10.

The calculation of file names for tiles close to the equator no longer includes an extra period.

`dted` now handles the case of missing tiles that result in nonrectangular tile configurations.

Transparently handles a kind of error found in some data tiles just north of the equator (affecting some tiles with file names ending in `n00.dt0`). In these files, the NW and NE corners are incorrectly designated with south instead of north latitudes.

Directory and filename calculations are more efficient.

km2sm, nm2sm, sm2km, sm2nm

These functions now use precise factors for conversion to/from statute miles.

These changes also correct `distdim` results for both statute miles and feet (assuming U.S. Survey Foot).

tigerp

Terminates file read properly when an empty data point is encountered (signaling end of data)

worldmap

No longer generates errors when there is no data to display within the specified latitude/longitude limits; it now displays complete mesh, even for data grids (matrix maps) that span all longitudes

Version 1.3 (R13) Mapping Toolbox Software

This table summarizes what's new in Version 1.3 (R13):

New Features and Changes	Version Compatibility Considerations	Fixed Bugs and Known Problems	Related Documentation at Web Site
Yes Details below	Yes — Details in **"Compatibility Considerations" on page 95** below. See also Summary.	Fixed Bugs	No

New features and changes introduced in this version are

New northarrow Annotation Function

The new `northarrow` function lets you add a north arrow symbol to a map, pointing to geographic north. You can reposition a north arrow by clicking and dragging it, or adjust other properties such as position, color, and size via alternate-clicking.

New mlabelzero22pi Axes Label Command

The new `mlabelzero22pi` command converts meridian labels to use the range 0 to 360 degrees, instead of the default -180 to 180 degrees.

Enhanced clegendm Contour Annotation Function

The `clegendm` function has been enhanced with a new, optional syntax that accepts a string indicating the contour line units. Alternatively, text label strings can be supplied for each and every contour level via a cell array.

New Interactive Interface

The new `lightmui` function provides an interactive, graphical user interface to control the position of lights on a globe or 3-D map.

Updated Atlas Data

Political boundaries and country names have been updated in both the `worldlo` and `worldhi` atlas files.

Ellipses Now Drawn More Smoothly

The `ellipse1` function has been modified to create smoother ellipses. A weighted distribution of azimuth points is used instead of the uniform distribution between the starting and ending points. More points are added at locations near the semi-major and semi-minor axes and fewer points at the other intermediate locations.

Compatibility Considerations

Functions contorm and contor3m Obsoleted

Functions `contorm` and `contor3m` are now obsolete. You should now use `contourm` and `contour3m` instead, which are functionally identical. The older functions still exist, but have been removed from the Mapping Toolbox documentation.

Version 1.2 (R12) Mapping Toolbox Software

This table summarizes what's new in Version 1.2 (R12):

New Features and Changes	Version Compatibility Considerations	Fixed Bugs and Known Problems	Related Documentation at Web Site
Yes Details below	No	No	No

New features and changes introduced in this version are

Higher Resolution Atlas Data

There are now high-resolution country outlines and more city locations available through the `worldhi` database. The `worldmap` command automatically chooses this high-resolution data if the region's area is small enough.

The `worldlo` atlas file has been updated to make it coincide more closely with high-resolution coastlines and boundaries.

The `worldmtxmed` MAT-file provides a medium-resolution political world matrix map.

External Data Interface Improved

Importing high-resolution atlas data is now much easier with these two visual interfaces:

- Digital Elevation Map Data user interface (invoked with the `demdataui` function)
- Vector Map Level 0 user interface (invoked with the `vmap0ui` function)

Many of the matrix map data interface functions now automatically concatenate data across separate files.

The external interface now supports the GLOBE digital elevation map data, a product similar to GTPO30. Use the `globedem` function for working with that data.

New Interactive Interfaces

You can now adjust interactively, on a map display, geographic lines such as great circle tracks, small circles and sectors of small circles. Use the `trackg`, `scircleg`, and `sectorg` functions to make these interactive adjustments. While in an edit mode, you can drag the lines around on the map, modify the lines in a control panel, or read measurements.

There is a new visual interface to create colormaps. Use the `cmapui` function to invoke this new interface.

New Analysis Functions for Geographic Data

You can use the new `elevation` function to find the elevation angle of a geographic point.

The new `gradientm` function performs matrix map data calculations, including gradient, slope, and aspect.

You can use the new `los2` and `viewshed` functions with terrain data to check the line of sight visibility between points or the visibility of entire regions.

Several new functions have been added to support polygon operations:

Function	Description
`bufferm`	Compute polygon buffer regions
`polybool`	Perform polygon Boolean operations

Function	Description
`polyjoin`	Combine polygon segments into a NaN-clipped polygon
`polymerge`	Merge polygon segments with abutting ends
`polysplit`	Separate NaN-clipped polygon segments into cell arrays
`polyxpoly`	Polygon intersections

Other New Functions

Several new functions have been added to support polygon operations.

Function	Description
`contourcmap`	Contour colormap and colorbar for surfaces
`driftcorr`	Heading to correct for wind or current drift
`driftvel`	Wind or current from heading, course, and speeds
`flatearthpoly`	Insert points along date line to pole
`lcolorbar`	Labeled colorbar
`mapprofile`	Interpolated lines connecting waypoints on a surface
`str2angle`	Convert formatted DMS angle strings to numbers

Compatibility Summary for Mapping Toolbox

This table summarizes new features and changes that might cause incompatibilities when you upgrade from an earlier version, or when you use files on multiple versions. Details are provided with the description of the new feature or change.

Version (Release)	New Features and Changes with Version Compatibility Impact
Latest Version V2.7.1 (R2008b)	See compatibility consideration subsections for • "Map Limit Syntaxes Removed" on page 7
V2.7 (R2008a)	See compatibility consideration subsections for • "Fixes and Improvements to Function avhrrgoode" on page 12 • "Improved Accuracy for the limitm and setpostn Functions" on page 13 • "Documentation for Functions tigermif and tigerp Removed" on page 18 • "Removed Syntaxes that Returned Error Messages in Optional Argument" on page 18

Version (Release)	New Features and Changes with Version Compatibility Impact
V2.6 (R2007b)	See compatibility consideration subsections for • "Degrees-Minutes-Seconds Conversion Functions Are Obsolete and Error if Used" on page 31 • "Time Conversion Functions Are Obsolete and Error if Used" on page 34 • "The unitstr function Is Obsolete" on page 25 • "Interpretation of "Miles" Units has Changed" on page 25 • "Functions deg2rad and rad2deg No Longer Convert Complex to Real" on page 30 • "Improvements to Data Trimming in patchm and patchesm" on page 27 • "Higher Quality boston.tif GeoTIFF Satellite Image" on page 28 • "Map Axes Now Display Transparent Objects More Easily" on page 29 • "cmapui GUI is now Obsolete" on page 34 • "Functions tigermif and tigerp Are Obsolete and Error if Used" on page 35 • "Change to avhrrlambert Function Behavior When No Data Is Available" on page 30

Version (Release)	New Features and Changes with Version Compatibility Impact
V2.5 (R2007a)	See compatibility considerations for • "Some GUIs Are No Longer Available from the Command Line" on page 37 • "New Functions for Degrees-Minutes-Seconds Conversions" on page 38 • "Time Conversion Functions to be Removed" on page 41
V2.4 (R2006b)	See compatibility consideration • "Changes in Behavior of mapshow and geoshow" on page 43
Latest Version V2.3 (R2006a)	See compatibility consideration • "GeoTIFF Compatibility Considerations" on page 49
V2.2 (R14SP3)	See compatibility consideration • "Improved Rendering of Polygons with Inner Rings" on page 53
V2.1 (R14SP2)	See compatibility considerations • "Compatibility Considerations for Atlas Data and Associated Functions" on page 57 • "Compatibility Considerations for worldmap and usamap" on page 58 • "Functions Deleted in This Release" on page 67 • "Functions Made Obsolete in This Release" on page 68

Version (Release)	New Features and Changes with Version Compatibility Impact
V2.0.3 (R14SP1)	See compatibility considerations • "axesm Does Not Reorder Longitude Limits" on page 70 • "contourm Returns Contourgroup Handle" on page 70 • "elevation Has New Azimuth Output" on page 70 • "gtopo30 Has New Options" on page 70 • "inputm Has New Button Output" on page 70 • "polybool Revised, Has New Helper Functions" on page 70 • "polymerge Output Vertex Order Changes" on page 72
V2.0.2 (R14)	None
V2.0.1 (R13SP1+)	None
V2.0 (R13SP1+)	See compatibility considerations • "Functions Obsolete in Version 2.0" on page 88 • "Functions Deleted in Version 2.0" on page 89 • "Changes in Nomenclature" on page 89
V1.3.1 (R13SP1+)	None
V1.3 (R13SP1)	See compatibility consideration • "Functions contorm and contor3m Obsoleted" on page 95
V1.2 (R12)	None