Mapping Toolbox

For Use with MATLAB®

Systems Planning and Analysis, Inc.

Computation

Visualization

Programming

User's Guide

Version 1

How to Contact The MathWorks:

508-647-7000	Phone

508-647-7001	Fax

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

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

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

Mapping Toolbox User's Guide

Printing History: May 1997 First printing - Version 1
October 1998 Second printing - Version 1.1
October 1998 Online update - Version 1.1

Dedication

In memory of John Snyder, whose prior work on Map Projections inspired and made possible the creation of the Mapping Toolbox.

Contents

Preface

1 Mapping Fundamentals

2 Displaying Maps

3 Atlas Data

4 Map Projections

5 Mapping Applications

6 GUI Tools

7 External Data Interface

Geographic Terms

A

Bibliography

B

Projections Reference

C

Index

Preface

Acknowledgments

Systems Planning and Analysis, Inc. (SPA) is proud to have developed the Mapping Toolbox. The Mapping Toolbox is the product of the contributions of numerous SPA employees. Ed Byrns and Eric Brown, who are no longer at SPA, conceived the Toolbox, designed the architecture, and supported the development to completion of version 1.0. Walter Stumpf managed the completion and continued development of the product and also developed the external interface and the atlas data.

Tomi Debole and Andy Kim were essential to the development, testing, and documentation of the Mapping Toolbox. Tom Seaman wrote many of the testing functions. Juliet Crumrine edited the documentation. Wendell Nix helped with the rhumb line calculations and the geometry of track intersections.

The product was developed with long-term support from The MathWorks and many of its key employees, most notably Loren Shure, Jim Tung, Matthew Simoneau, Scott Gray, Kathy Ford, Jennifer French, and Clay Thompson.

1

Mapping Fundamentals

What Is a Map?

The simplest definition of a map is *a representation of geographic data*. To the ancient Egyptians, this representation first took the form of lists of city names in the order they would be encountered when following a given road. Two-dimensional renditions, such as paper road maps, are more familiar to us today. Even classroom globes are maps under this definition.

This toolbox addresses electronic representations of geographic data. It allows the creation, use, and presentation of geographic data in a variety of forms and to a variety of ends.

In the Mapping Toolbox, a *map* is any variable or set of variables representing or assigning values to a geographic location or region, from a single point to an entire planet.

Types of Maps in the Mapping Toolbox

Vector Maps

A series of latitude-longitude coordinate pairs representing, for example, points along the coast of Greenland, or along Interstate 80, or even the two sets together, form a map. In this case, the geographic data is in *vector* format and is referred to as a *vector map*. This format consists of specific points, along with some indication as to how they should or should not be connected to each other. In the Mapping Toolbox, vector data consists of sequentially ordered pairs of latitude and longitude coordinates. The pairs are considered to be connected in sequence; breaks in connectivity must be delineated by the creation of separate variables or by inserting `NaN`s into the sets at the appropriate break points. For vector map data, the connectivity of the data is often only a concern during display.

To illustrate a vector map, enter the following:

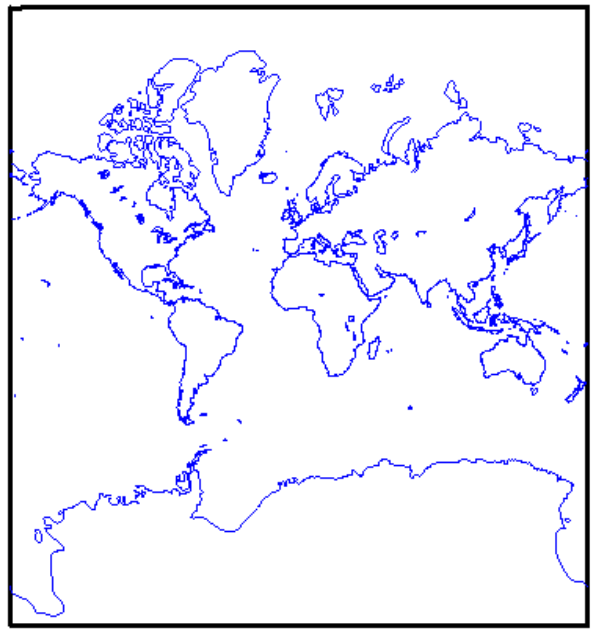
```
load coast
whos
```

```
   Name             Size      Bytes   Class

   lat            9589x1      76712   double array
   long           9589x1      76712   double array
```

```
axesm mercator
framem
plotm(lat,long)
```

The variables `lat` and `long` are vectors in the `coast` MAT-file, which together form a vector map of the coastlines of the world.

We have chosen to view the map in a Mercator projection. A map projection displays the spherical map data in a two-dimensional plane. There are many possible ways to project a map.



Matrix Maps

A matrix in which each element represents a value corresponding to a specific geographic area also forms a map. In this case, the geographic data is in *matrix* format, which is often called *raster* format. However, the Mapping Toolbox makes great use of the powerful matrix manipulation capabilities of MATLAB to fully exploit this map type, so the term *matrix map* is more appropriate.

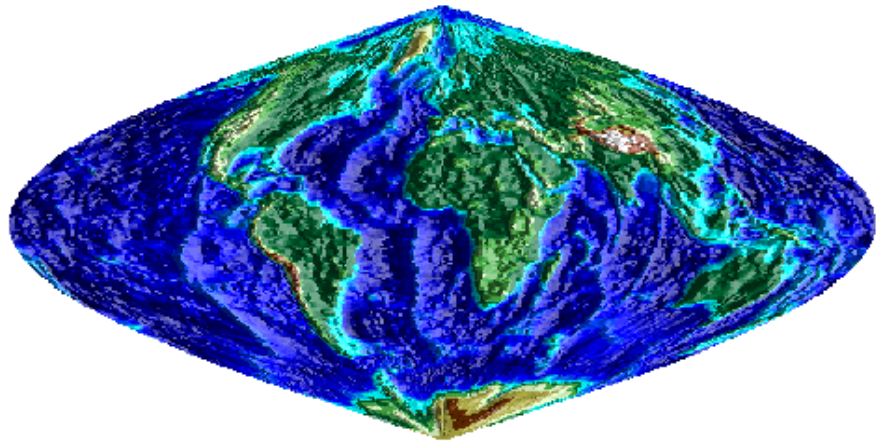
When the data consists of surface elevations, the map can also be referred to as a *digital elevation map* (DEM) or *topographical map*.

As an example of this map type, consider a 180-by-360 matrix. Each row represents one degree of latitude, and each column represents one degree of longitude. Each entry of this matrix is the average elevation, in meters, for the one-degree-by-one-degree region of the Earth to which its row and column correspond. This is, in fact, the `topo` matrix that is provided in the MATLAB product in the workspace of the same name.

To display the matrix map, type the following:

```
clma % clears map
load topo
axesm sinusoid
meshlsrm(topo,topolegend)
```

The `topo` matrix is displayed as a lighted, shaded relief map in a Sinusoidal projection:



Note that the meaning of the map and the projection in which it is displayed are completely independent. The `topo` variable is a map whether it is displayed or not, and it is the same map no matter how it is displayed.

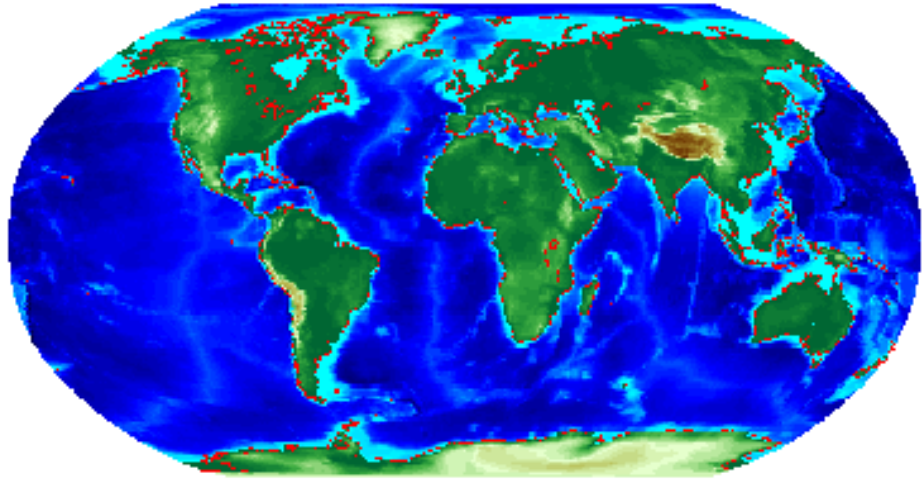
Composite Maps

Many occasions arise in which vector map variables and matrix map variables are used or displayed together. For example, continental coastlines in vector form might be displayed with matrix temperature data to make the latter more useful. When several map variables are used together, regardless of type, they can be correctly referred to as a single map.

Since we have our `coast` and `topo` workspaces available from the previous examples, we can combine the two in a single map and see how well the two types of data correspond:

```
figure; axesm robinson
meshm(topo,topolegend); demcmap(topo)
plotm(lat,long,'r')
```

Here is the resulting map:



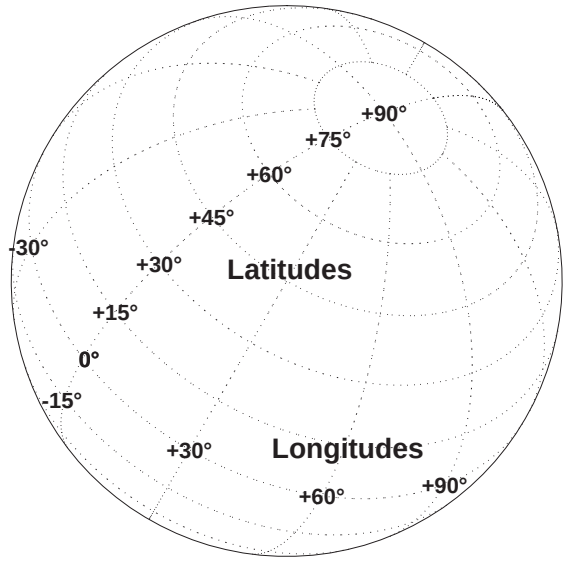
The remainder of this chapter will focus on the fundamental principles of geographic measurement and data manipulation that are a prerequisite for creating map displays. Chapter 2, “Displaying Maps”, takes up the topic of map display and interaction.

Geographic Measurement

Latitude and Longitude

The position of a point on the surface of the Earth, or any other planet, for that matter, can be specified with two angles, *latitude* and *longitude*. These angles can be specified in degrees or radians; however, degrees are far more common in geographic usage.

Latitude is the angle at the center of the planet between the plane of the Equator and a line through the center passing through the surface at the point in question. Latitude is positive in the Northern Hemisphere, reaching a limit of +90° at the North Pole, and negative in the Southern Hemisphere, reaching a limit of -90° at the South Pole. Lines of constant latitude are called *parallels*.



Longitude is the angle at the center of the planet between two planes passing through the center and perpendicular to the plane of the Equator. The first plane also includes the surface point in question; the second plane contains the Prime Meridian, which currently means it includes the point defined by the Royal Observatory in Greenwich, England. Lines of constant longitude are called *meridians*.

Longitudes ranging from -180° to +180° are unique. Longitudes may be given outside this range, but such values can always be renamed to a value within the range. The Mapping Toolbox provides a number of functions to convert data between different latitude and longitude ranges. For example, the command `npi2pi` wraps data to this domain:

```
longitudes = [-560 125 190];
newlongitudes = npi2pi(longitudes)
newlongitudes =
  160.0000  125.0000 -170.0000
```

Sometimes it is more useful to consider longitude as strictly positive, proceeding from the Prime Meridian (0°) eastward around and back to the Prime Meridian (360°). Any longitude data can be converted to this domain using the `zero22pi` command:

```
positivelongs = zero22pi(newlongitudes)
positivelongs =
  160.0000  125.0000  190.0000
```

If you need this data in radians, you can use an angle conversion function:

```
radianlongs = deg2rad(positivelongs)
radianlongs =
  2.7925    2.1817    3.3161
```

Several angle conversion functions are available in this toolbox, supporting degrees, radians, and degrees-minutes-seconds notation.

What is the antipodal point (on the opposite side of the Earth) of Natick, Massachusetts (about 42.3°N, 71.35°W)?

```
[antilat,antilong] = antipode(42.3,-71.35)
antilat =
   -42.3000

antilong =
   108.6500
```

The result (42.3°S,108.65°E) lies in the Indian Ocean southwest of Australia.

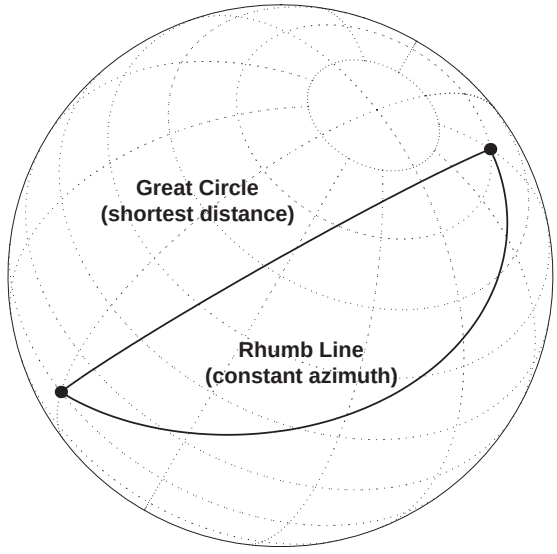
Great Circles and Rhumb Lines

In plane geometry, lines have two important characteristics. A line represents the shortest path between two points, and the slope of such a line is constant. When describing lines on the surface of a spheroid, however, only one of these characteristics may be guaranteed at a time.

In geography, a *great circle* is the shortest path between two points along the surface of a sphere. The precise definition of a great circle is the intersection of the surface with a plane passing through the center of the planet. In general, great circles do not have a constant azimuth, the spherical analog of slope; they cross successive meridians at different angles. When the Earth is taken as a sphere, the Equator and all meridians are great circles. Great circles always bisect the sphere.

A *rhumb line* is a complex curve, the special property of which is that it crosses each meridian at the same angle. This curve is also referred to as a *loxodrome*. In general, a rhumb line is not the shortest path between two points on the rhumb line. All parallels, including the Equator, are rhumb lines, since they cross all meridians at 90°. Additionally, all meridians are rhumb lines. Rhumb lines always terminate at the poles, unless the azimuth is true east or west, in which case the rhumb line closes on itself to form a parallel of latitude.

A description of how to calculate points along great circles and rhumb lines appears later in this document.



Measuring Distance

When you calculate the distance between two points with the Mapping Toolbox, the result will depend upon whether you want a great circle or a rhumb line distance. The `distance` command will return the appropriate distance between two points as an angular arc length, employing the same angular units as the input latitudes and longitudes. The default path type is the shorter great circle, and the default angular units are degrees. The previous figure shows two points at (15°S, 0°) and (60°N, 150°E). The great circle distance between them, in degrees of arc, is as follows:

```
distgc = distance(-15,0,60,150)
distgc =
  129.9712
```

The rhumb line distance is greater:

```
distrh = distance('rh',-15,0,60,150)
distrh =
  145.0288
```

To determine how much longer the rhumb line path is in, say, kilometers, you can use a distance conversion function on the difference:

```
kmdifference = deg2km(distrh–distgc)
kmdifference =
  1.6744e+03
```

Several distance conversion functions are available in the toolbox, supporting degrees, radians, kilometers, meters, statute miles, nautical miles, and feet. Converting distances between angular arc length units and surface length units requires the radius of a planet or spheroid. By default, the radius of the Earth is used.

Measuring Azimuth

Azimuth is the angle a line makes with a meridian, taken clockwise from north. When the azimuth is calculated from one point to another using the Mapping Toolbox, the result will depend upon whether a great circle or a rhumb line azimuth is desired. For great circles, the result will be the azimuth at the starting point of the connecting great circle path. In general, the azimuth along a great circle is not constant. For rhumb lines, the resulting azimuth is constant along the entire path.

Azimuths, or bearings, are returned in the same angular units as the input latitudes and longitudes. The default path type is the shorter great circle, and the default angular units are degrees. In our example, the great circle azimuth from the first point to the second is:

```
azgc = azimuth(-15,0,60,150)
azgc =
  19.0391
```

For the rhumb line, the constant azimuth is:

```
azrh = azimuth('rh',-15,0,60,150)
azrh =
  58.8595
```

One feature of rhumb lines is that the inverse azimuth, from the second point to the first, is the complement of the forward azimuth and can be calculated by simply adding 180° to the forward value:

```
inverserh = azimuth('rh',60,150,-15,0)
inverserh =
  238.8595

difference = inverserh–azrh
difference =
  180
```

This is not true, in general, of great circles:

```
inversegc = azimuth('gc',60,150,-15,0)
inversegc =
  320.9353

difference = inversegc–azgc
difference =
  301.8962
```

The azimuths associated with cardinal and intercardinal compass directions are the following:

North	0° or 360°
Northeast	45°
East	90°
Southeast	135°
South	180°
Southwest	225°
West	270°
Northwest	315°

Reckoning Positions

A common problem in geographic applications is the determination of a destination given a starting point, an initial azimuth, and a distance. In the Mapping Toolbox, this process is called *reckoning*. A new position may be reckoned in a great circle or a rhumb line sense (great circle or rhumb line track).

As an example, an airplane takes off from La Guardia Airport in New York (40.75°N, 73.9°W) and follows a northwestern rhumb line flight path at 200 knots (nautical miles per hour). Where would it be after 1 hour?

```
[rhlat,rhlong] = reckon('rh',40.75,-73.9,nm2deg(200),315)
rhlat =
  43.1054
rhlong =
  -77.0665
```

Notice that the distance, 200 nautical miles, must be converted to degrees of arc length with the `nm2deg` conversion function to match the latitude and longitude inputs. If the airplane had a flight computer that allowed it to follow an exact great circle path, what would the aircraft's new location be?

```
[gclat,gclong] = reckon('gc',40.75,-73.9,nm2deg(200),315)
gclat =
  43.0615
gclong =
  -77.1238
```

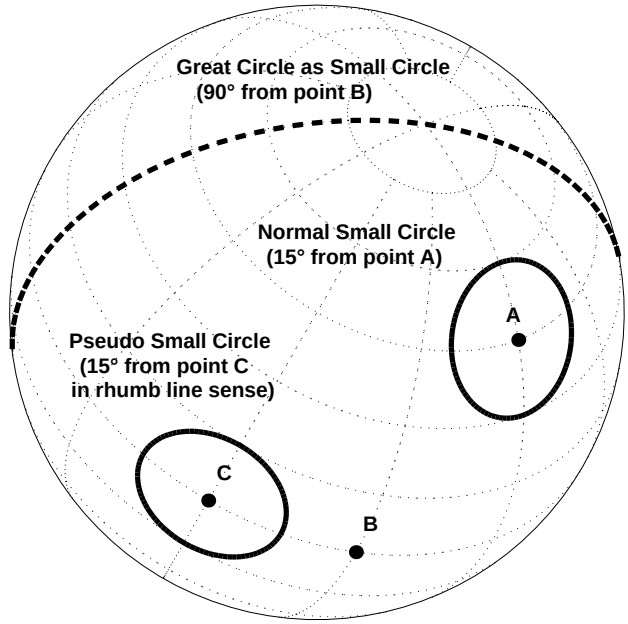
Notice also that for short distances at these latitudes, the result hardly differs between great circle and rhumb line. The two destination points are less than 4 nautical miles apart. Incidentally, after 1 hour, the airplane would be just north of New York's Finger Lakes.

Small Circles

In addition to rhumb lines and great circles, one other geographic construction is significant in geography and the Mapping Toolbox: the *small circle*. The precise definition of a small circle is the intersection of a plane with the surface of the planet. In the Mapping Toolbox, this definition includes planes passing through the center of the planet, so the set of all small circles includes all great circles as limiting cases. This usage is not universal.

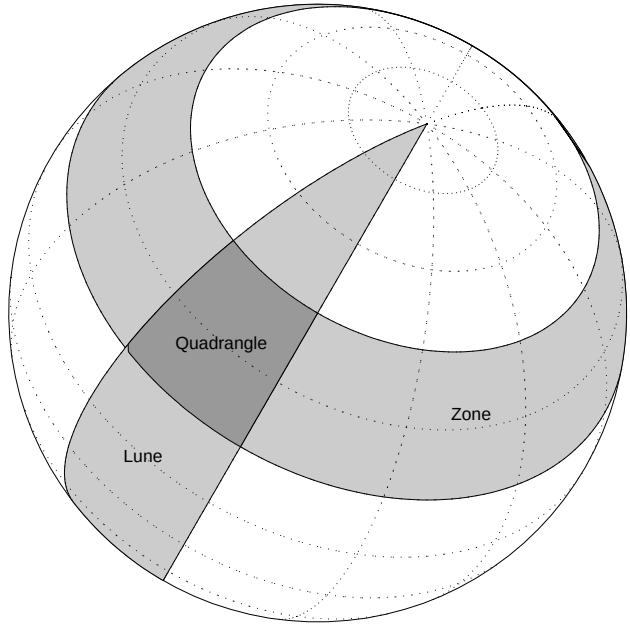
Small circles are most easily defined by distance. *All points 45 nm (nautical miles) distant from (45°N,60°E)* would be the description of one small circle. If degrees of arc length are used as a distance measurement, then a great circle is the set of all points 90° distant from a particular *center* point.

For true small circles, the distance must be defined in a great circle sense, the shortest distance between two points on the surface of a sphere. However, the Mapping Toolbox also allows the calculation of a *pseudo small circle*, for which distances are measured in a rhumb line sense (along lines of constant azimuth). These should not be confused with true small circles.



Measuring Area

In solid geometry, the area of a spherical quadrangle can be exactly calculated. A spherical quadrangle is the intersection of a *lune* and a *zone*. In geographic terms, a *quadrangle* is defined as a region bounded by parallels north and south, and meridians east and west.



In the pictured example, a quadrangle is formed by the intersection of a zone, which is the region bounded by 15°N and 45°N latitudes, and a lune, which is the region bounded by 0° and 30°E longitude. Under the spherical planet assumption, the fraction of the entire spherical surface area inscribed in the quadrangle can be calculated:

```
area = areaquad(15,0,45,30)
area =
  0.0187
```

That is, less than 2% of the planet's surface area is in this quadrangle. To get an absolute figure in, for example, square miles, you must provide the appropriate spherical radius. The radius of the Earth is about 3958.9 miles.

```
area = areaquad(15,0,45,30,3958.9)
area =
      3.6788e+06
```

The surface area within this quadrangle is over 3.6 million square miles for a spherical Earth.

The Geoid Model

Although the Earth is round, it is not exactly a sphere. However, for many users and most applications, the difference is so small that a spherical assumption is sufficient. This section addresses how the Mapping Toolbox handles more accurate models of the shape, or figure, of the Earth and other planets.

What Is a Geoid?

Literally, *geoid* means *Earth-shaped*. The geoid is the precise figure of the Earth. Specifically, it is an equipotential surface with respect to gravity. It is approximately an oblate ellipsoid, but not exactly so because of local variations in gravity.

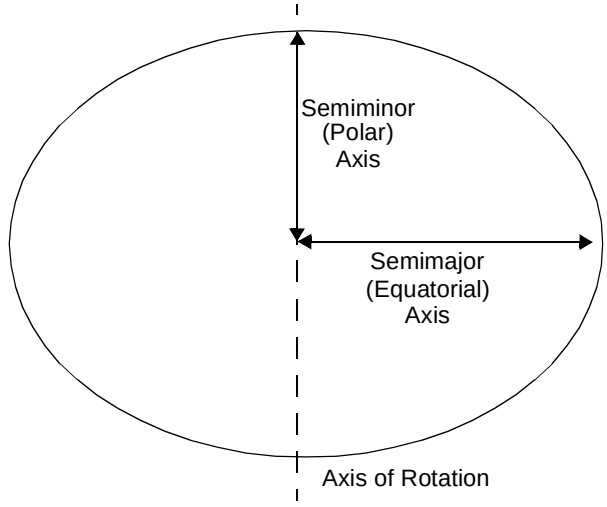
For the purposes of map display, the geoid of the Earth is assumed to be an exact ellipsoid. This is a simplification of the precise definition of geoid; however, it is a good approximation, which allows a great variety of calculations to be performed that would otherwise be prohibitively complicated. Ellipsoidal models of the Earth's figure are considered sufficient for cartography. The use of the term geoid for this ellipsoid is perhaps peculiar to the toolbox. Further, in the Mapping Toolbox, the term geoid is extended to ellipsoidal models for the figures of the Sun, Moon, and planets.

For purposes other than map display, a more detailed model of the figure of the Earth is used. See Chapter 3, "Atlas Data" and Chapter 7, "External Data Interface", for more information on detailed geoid models.

The Geoid Vector

Since the toolbox treats the geoid as an ellipsoid, the geoid can be described with a two-element vector, which is often called the *geoid vector* in this guide. The geoid vector has the form [*semimajor-axis eccentricity*]. The semimajor axis can be in any unit of distance; the choice of these units often drives the units used for distance outputs in the toolbox functions.

An *ellipsoid* is defined as a rotation of an ellipse around its minor axis. Ellipses can be defined with a semimajor and a semiminor axis, *flattening*, the parameter `n`, or the preferred method of a semimajor axis and eccentricity. The toolbox has functions to convert elliptical definitions from any of these forms to the form of the geoid vector. For example, the command `axes2ecc` returns an eccentricity when given the semimajor and semiminor axes.



The geoid vector, along with several other kinds of planetary data for each of the planets and the Earth's moon, can be queried using the `almanac` function provided by the Mapping Toolbox (see “Measuring the Planets”).

For the Earth, the semimajor axis is about 21 kilometers longer than the semiminor axis. Use the `almanac` function to verify this:

```
geoid = almanac('earth','geoid','kilometers')
geoid =
      6378.14          0.08

semiminor = minaxis(geoid)
semiminor =
      6356.75

geoid(1) - semiminor
ans = 21.38
```

When compared to the semimajor axis, which is almost 6400 kilometers, this difference seems insignificant and can be neglected for most purposes. For this reason, most functions in the Mapping Toolbox default to a spherical model of the Earth. The ellipse in the previous diagram is very exaggerated. The difference between the semimajor and semiminor axes of a correct ellipse at this scale would be far smaller than the width of the line used to draw it.

What Is the "Correct" Geoid Vector?

A variety of ellipsoid models have been proposed through the years. They differ because of the surveying information upon which they are based, or because certain regions of interest of the true geoid are best modeled by a particular ellipsoid that may be less accurate for other parts of the world. The Mapping Toolbox default geoid vector (after the sphere) is based on the 1980 Geodetic Reference System ellipsoid. This geoid vector is returned by the statement `almanac('earth','geoid')`. It is also the reference ellipsoid for the 1984 World Geodetic System equipotential model.

Several other geoid vectors are supported, for models ranging from Everest's 1830 ellipsoid (used for India) to the International Astronomical Union ellipsoid of 1965 (used for Australia). These can be referenced by the following statements:

```
geoid1 = almanac('earth','geoid',[],'everest');
geoid2 = almanac('earth','geoid',[],'iau65');
```

The reference page for the `almanac` function has more information on which geoids are available.

You should probably use a spherical Earth unless you have a good reason to use an ellipsoidal model. If you use an ellipsoidal model, you should probably use the default unless you need a specific one. If you cannot find the geoid vector you need, you can create it in the following form:

```
geoidvec = [semimajor-axis eccentricity]
```

Note that the default units for the geoid semimajor axis in the `almanac` function are kilometers, which can be used by simply passing in an empty matrix in place of the input units string, as in the previous example. Eccentricity is dimensionless.

Measuring the Planets

The Mapping Toolbox contains a function that provides almanac data on the major bodies of our solar system. Geophysical data, such as geoid vectors, radii, surface areas, and volumes, can be accessed for the Sun, the Earth's moon, and all of the planets, in any of the supported units of distance measurement.

Many planets have ellipsoidal geoid vectors available. Some planets return spherical geoid vectors only:

```
almanac('earth','geoid','nauticalmiles')
ans =
  3443.92          0.08

almanac('mars','geoid','kilometers')
ans =
  3396.90          0.11

almanac('moon','geoid','statutemiles')
ans =
  1079.97             0
```

When a radius is desired, a scalar is returned representing the radius of the best spherical model of the planet. Notice that for a spherical model, the radius in radians is 1:

```
almanac('mercury','radius','kilometers')
ans =
  2439

almanac('neptune','radius','radians')
ans =
  1
```

Surface areas and volumes are calculated based on a spherical model by default. In most cases, you can use the geoid model instead, and for the Earth, you can specify any of the supported geoid models. You can also request the actual tabulated values of the Earth:

```
almanac('mars','surfarea','kilometers','geoid')
ans =
  1.4441e+08

almanac('earth','volume','kilometers','international')
ans =
  1.0833e+12

almanac('earth','volume','kilometers','actual')
ans =
  1.0832e+12
```

A complete description of available data is provided under the `almanac` reference page in the "Mapping Reference" chapter of the *Mapping Toolbox Reference Guide*.

Angles, Times, and Distances – Units and Notation

Angular Notation

Angles can be represented as variables in the Mapping Toolbox in three ways: degrees, radians, and degrees-minutes-seconds. The toolbox provides functions for converting between these formats.

Degrees

This is the default angular unit notation for the toolbox. Although for most scientific applications, radians are more commonly used, in geographic usage, degrees are much more convenient.

Degree notation is simply decimal notation in terms of degrees. Forty-three and one-half degrees would be 43.5.

Radians

Radian notation is simply decimal notation in terms of radians. Two radians would be 2.0.

Degrees-Minutes-Seconds

Degrees-minutes-seconds, or *dms*, notation is a little more complicated. In text, a dms angle would be *ddd° mm' ss"*. For example, 142°15'27" is 142 degrees, 15 minutes, and 27 seconds. There are 60 seconds in a minute and 60 minutes in a degree. In the Mapping Toolbox, when “dms” angles are represented by a single number, the format is *dddmm.ss*. For example, 142°15'27" is 14215.27.

The real value of this notation is for the input of data already in this format. The toolbox includes the `mat2dms` function for easily entering dms data.

A special case of the dms format is the *dm* format, in which seconds are not included.

If you have a three-column matrix in which the columns are degrees, minutes, and seconds, respectively, `mat2dms` will convert it to dms format:

```
dmsmatrix = [45 13 46; 156 45 01; -7 34 12.1]
dmsmatrix =
 45.0000   13.0000   46.0000
156.0000   45.0000    1.0000
 -7.0000   34.0000   12.1000

dmsformat = mat2dms(dmsmatrix)
dmsformat =
 1.0e+04 *
 0.4513
 1.5645
-0.0734
```

Note: Care must be exercised when working with the dms format; for example, two angles in this format cannot simply be added. It is advisable to convert "dms" data to decimal degrees before working extensively with it.

Converting Between Angle Unit Formats

The toolbox includes a variety of angle unit conversion functions. For example, to convert the dms format values to degrees or to radians, you can use `dms2deg` and `dms2rad`, respectively:

```
degformat = dms2deg(dmsformat)
degformat =
 45.2294
156.7503
 -7.5700
radformat = dms2rad(dmsformat)
radformat =
 0.7894
 2.7358
-0.1321
```

Similar functions include `deg2rad`, `rad2deg`, `deg2dms`, etc. There is also a more general function, `angledim`, which will convert from one format to another. For example, how many degrees are in ¼ radians?

```
degs = angledim(1/4*pi,'radians','degrees')
degs =
  45
```

Time Notation

Times can be represented as variables in the Mapping Toolbox in three ways: hours, seconds, and hours-minutes-seconds. The toolbox provides functions for converting between these formats.

Hours

This is the default time unit notation for the toolbox.

Hour notation is simply decimal notation in terms of hours. Two hours and fifteen minutes would be 2.25.

Seconds

Seconds notation is simply decimal notation in terms of seconds. One hour would be 3600.

Hours-Minutes-Seconds

Hours-minutes-seconds, or *hms* notation, is analogous to dms notation for angles. In text, an hms time would be *hh*:*mm*:*ss*. For example, 12:36:15 is 12 hours, 36 minutes, and 15 seconds. In the Mapping Toolbox, when hms times are represented by a single number, the format is *hhmm.ss*. For example, 12:36:15 is 1236.15.

The real value of this notation is in entering data already in this format. The toolbox includes the `mat2hms` function to easily input hms data, which functions very similarly to the `mat2dms` function described earlier.

Note: Care must be exercised when working with the hms format; for example, two times in this format cannot simply be added. It is advisable to convert hms data to decimal hours before working extensively with it.

Converting Between Time Unit Formats

Time units can be converted using functions similar to those described for angle unit conversions. These include `hr2sec` and `hms2hr`, as well as a general conversion function `timedim`, which works just like `angledim`.

Distance Notation

Distances in the Mapping Toolbox can be measured in a number of different units. Functions are provided to convert between nautical miles (nm), statute miles (sm), feet (ft), kilometers (km), meters (m), degrees of arc length (deg), and radians of arc length (rad). The names of these functions are of the form `sm2km`, `km2rad`, etc. A general distance conversion function, `distdim`, is available as well.

There is no single default unit of distance measurement in the toolbox. Navigation functions use nautical miles as a default, almanac functions default to kilometers, and the `distance` function defaults to degrees of arc length. It is essential that you understand the default units of any function you use.

Note: When distances are given in terms of angular units (degrees or radians), be careful to remember that these are in terms of arc length. While a degree of latitude always subtends a degree of arc length, this is not necessarily the case for degrees of longitude. Also, using arc length as distance assumes a knowledge of the underlying radius.

On the Earth, a degree of arc length is about 60 nautical miles:

```
nauticalmiles = deg2nm(1)
nauticalmiles =
  60.0405
```

The Earth is the default assumption for these conversion functions. Other radii may be used, however:

```
nauticalmiles = deg2nm(1,almanac('moon','radius'))
nauticalmiles =
  30.3338
```

Working with Vector Maps

Data Format

Vector maps are usually contained in a pair of variables, which represent the latitude-longitude pairs for points of interest. For example, the following two variables can be a vector map:

```
lat = [45.6 -23.47 78];
long = [13 -97.45 165];
```

Since this is a made-up example, these points could mean anything. Perhaps they are the locations over which geosynchronous satellites are stationed. If so, you might think of them as three distinct points. Perhaps they represent the starting point, the mid-course marker, and the finish point of an airplane race. Think of them as three points connected by two line segments. Or perhaps they represent the vertices of a triangle bounding a region with a propensity for unexplained phenomena, in which case think of them as describing a polygon.

The Mapping Toolbox provides functionality for each of these interpretations. For many purposes, the distinction is irrelevant; for others, the choice of a function implies the appropriate interpretation. For example, the command `plotm` will display the data as a line, while `fillm` will display it as a filled polygon.

When vector variables are used to represent several segments or polygons, successive objects should be delineated by the insertion of `NaN`s in both variables. For example, if a second segment is to be added to the above map, the two objects can reside in the same pair of variables:

```
lat = [45.6 -23.47 78 NaN 43.9 -67.14 90 -89];
long = [13 -97.45 165 NaN 0 -114.2 -18 0];
```

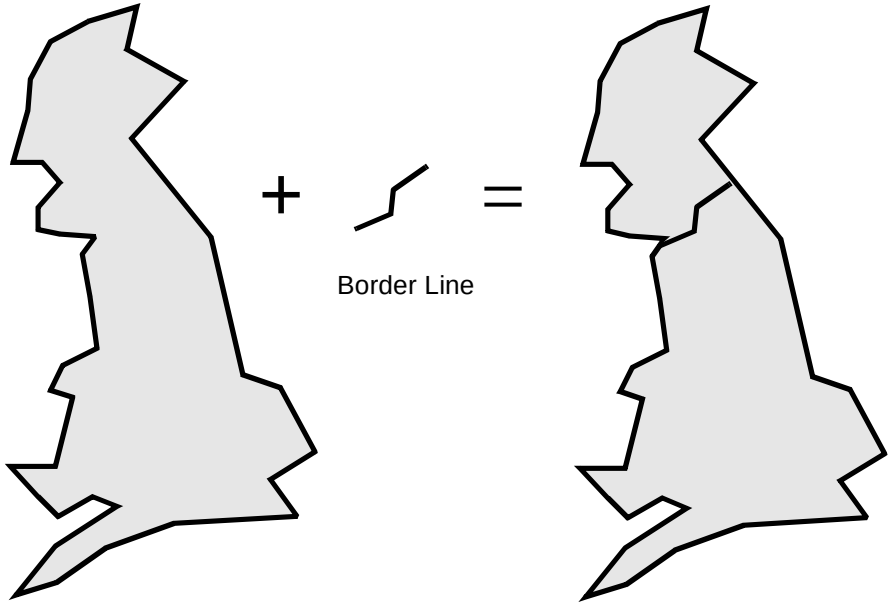
Notice that the `NaN`s must appear in the same locations in both variables. Here we have a segment of three points separated from a segment of four points. The `NaN`s perform two functions; they provide a simple means of identifying break points in the data, and they act as *pen-up* commands when plotting vector maps. The `NaN`s are also used in the Mapping Toolbox to separate non-connected patch faces.

Note: This definition differs from MATLAB, which does not display `NaN` clipped data as patches.

Segments versus Polygons

Geographic objects represented by vector data may or may not be formatted as polygons. Imagine two variables, `latcoast`, and `loncoast`, which correspond to sequential points around the coast of the island of Great Britain. If this data returns to its starting point, then a polygon-formatted map of Great Britain exists. This data might be plotted as a patch or as a line, and it might be logically employed in calculations as either.

Now consider two more variables, `latborder` and `lonborder`, which correspond to points along the Anglo-Scottish border, proceeding from the west coast at Solway Firth to the east coast at Berwick-upon-Tweed. This data can really only be plotted as a line. The two pairs of variables together can form a map. They can be displayed as a patch of Great Britain with a line showing the border. However, you still would not have a polygon-formatted map of Scotland alone.



Creating Vector Data

In addition to the vector map data available in the Mapping Toolbox and from external sources for such things as coastlines, political boundaries, and other more *permanent* objects, you may want to create your own vector data for such things as great circles, rhumb lines, and small circles.

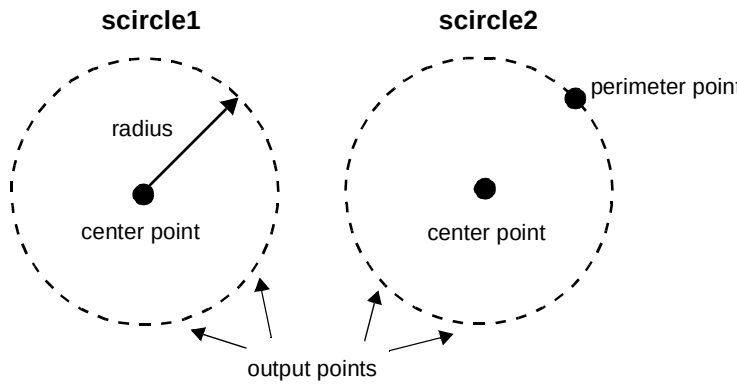
Calculating Small Circles

You can calculate vector data for points along a small circle in two ways. If you have a center point and a known radius, use `scircle1`; if you have a center point and a single point along the circumference of the small circle, use `scircle2`. For example, to get data points describing the small circle at 10° distance from (67°N, 135°W), use the following:

```
[latc,lonc] = scircle1(67,-135,10);
```

To get the small circle centered at the same point that passes through the point (55°N,135°W), use `scircle2`:

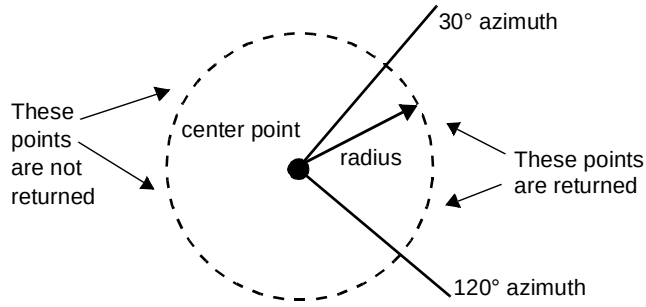
```
[latc,lonc] = scircle2(67,-135,55,-135);
```



The `scircle1` function also allows points along a specific arc of the small circle to be calculated. For example, if you want to know the points 10° in distance and between 30° and 120° in azimuth from (67°N,135°W), simply provide arc limits:

```
[latc,lonc] = scircle1(67,-154,10,[30 120]);
```

scircle1 with arc limits



When an entire small circle is calculated, the data is in polygon format. For all calculated small circles, 100 points are returned unless otherwise specified. You can calculate several small circles at once by providing vector inputs. See the reference pages on `scircle1` and `scircle2` in the *Mapping Toolbox Reference Guide* for more information.

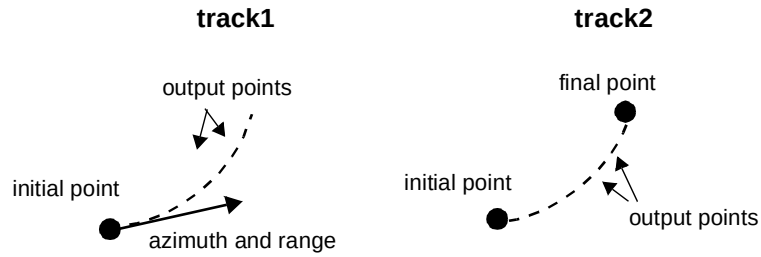
Calculating Tracks – Great Circles and Rhumb Lines

You can generate vector data corresponding to points along great circle or rhumb line tracks using `track1` and `track2`. If you have a point on the track and an azimuth at that point, use `track1`. If you have two points on the track, use `track2`. For example, to get the great circle path starting at (31°S, 90°E) with an azimuth of 45° with a length of 12°, use `track1`:

```
[latgc,longc] = track1('gc',-31,90,45,12);
```

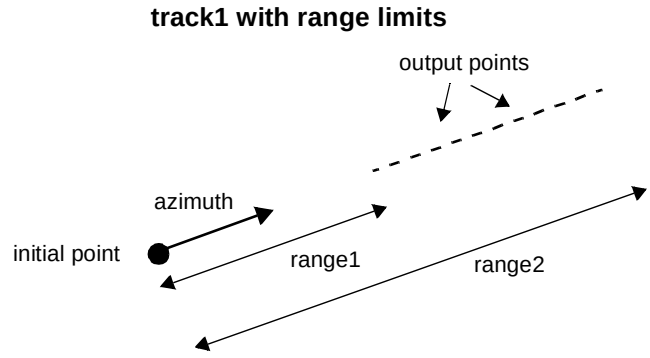
For the great circle from (31°S, 90°E) to (23°S, 110°E), use `track2`:

```
[latgc,longc] = track2('gc',-31,90,-23,110);
```



The `track1` function also allows range endpoints to be specified. For example, if you want points along a rhumb line starting 5° away from the initial point and ending 13° away, at an azimuth of 55°, simply specify the range limits:

```
[latrh,lonrh] = track1('rh',-31,90,55,[5 13]);
```



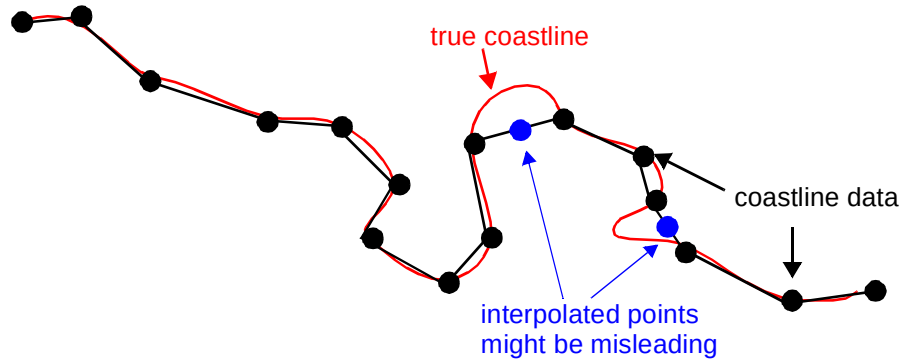
When no range is provided for `track1`, the returned points represent a *complete track*. For great circles, a complete track is 360°, encircling the planet and returning to the initial point. For rhumb lines, the complete track terminates at the poles, unless the azimuth is 90° or 270°, in which case the complete track is a parallel that returns to the initial point.

For calculated tracks, 100 points are returned unless otherwise specified. You can calculate several tracks at one time by providing vector inputs. See the reference pages on `track1` and `track2` in the *Mapping Toolbox Reference Guide* for more information. More vector path calculations are described later in Chapter 5, “Navigation”.

Geographic Interpolation

When using vector data, you must be careful when you make assumptions concerning geographic reality between data points. For instance, when plotting vector data, you might connect each point with a straight line segment. This does not usually indicate any true knowledge about the region between known points. Data consisting of points along a coastline might be sparse; in the absence of other knowledge, filling in data can be misleading.

Interpreting Sparse Vector Data



Despite the dangers of misinterpretation, many circumstances exist in which geographic data interpolation is useful or necessary. Sparser data can be linearly *filled-in* with the `interpm` function.

Consider a set of latitude and longitude points that you want to be no further than one degree in separation in either direction.

```
lats = [1 2 4 5]; longs = [1 3 4 5]; maxdiff = 1;

[newlats,newlongs] = interpm(lats,longs,maxdiff)
newlats =
  1.0000
  1.5000
  2.0000
  3.0000
  4.0000
  5.0000
newlongs =
  1.0000
  2.0000
  3.0000
  3.5000
  4.0000
  5.0000
```

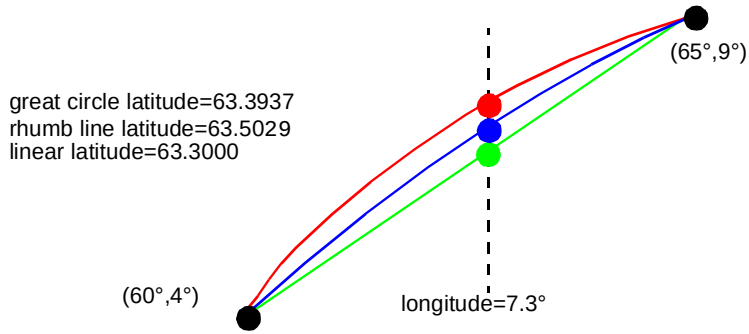
In the original `lats`, there is a gap of 2 degrees between the 2 and the 4. A linearly interpolated point, (3,3.5) was therefore inserted in `newlats` and `newlongs`. Similarly, in the original `longs`, there is a gap of 2 degrees between the 1 and the 3. The point (1.5,2) was therefore interpolated and placed into `newlats` and `newlongs`. Now, no adjacent points in either `newlats` or `newlongs` is greater than `maxdiff` in separation.

The `interpm` function returns the original data with new linearly interpolated points inserted. Sometimes, however, only the interpolated values are desired. The commands `intrplat` and `intrplon` provide a capability similar to MATLAB's `interp1` command, allowing for different methods of interpolation.

Use `intrplat` to interpolate a latitude for a given longitude. Given a monotonic set of longitudes and their matching latitude points, you can interpolate a new latitude for a given longitude in a linear, spline, cubic, rhumb line, or great circle sense.

Here, find the latitude corresponding to a longitude of 7.3° in the following data in a linear, great circle, and rhumb line sense:

```
longs = [1 3 4 9 13]; lats = [57 68 60 65 56]; newlong = 7.3;

newlat = intrplat(longs,lats,newlong,'linear')
newlat =
  63.3000

newlat = intrplat(longs,lats,newlong,'gc')
newlat =
  63.5029

newlat = intrplat(longs,lats,newlong,'rh')
newlat =
  63.3937
```



The `intrplon` function provides the same capability for interpolating new longitudes for given latitudes.

Polygon Area

For vector data in polygon format, geographic areas can be calculated using the command `areaint`. The function performs a numerical integration using Green's Theorem for the area on a surface enclosed by a polygon. Since this is a discrete integration on discrete data, the results are not exact. Nevertheless, the method provides the best means of calculating the area of arbitrarily shaped regions. Better measures result from better data.

The Mapping Toolbox function `areaint` (for area by integration), like the other area functions `areaquad` and `areamat`, returns areas as a fraction of the entire

planet's surface, unless a radius is provided. Here we calculate the area of the continental United States using the usalo workspace. Three areas are returned, because the data contains three polygons: Long Island, Martha's Vineyard, and the rest of the continental U.S.:

```
load usalo
earthradius = almanac('earth','radius');
area = areaint(uslat,uslon,earthradius)
area =
  1.0e+06 *
       7.9256
       0.0035
       0.0004
```

Since the default Earth radius is in kilometers, the area is in square kilometers. From the same workspace, the areas of the Great Lakes can be calculated, this time in square miles:

```
earthradius = almanac('earth','radius','miles');
area = areaint(gtlakelat,gtlakelon,earthradius)
area =
  1.0e+04 *
       8.0124
       1.0382
       0.7635
```

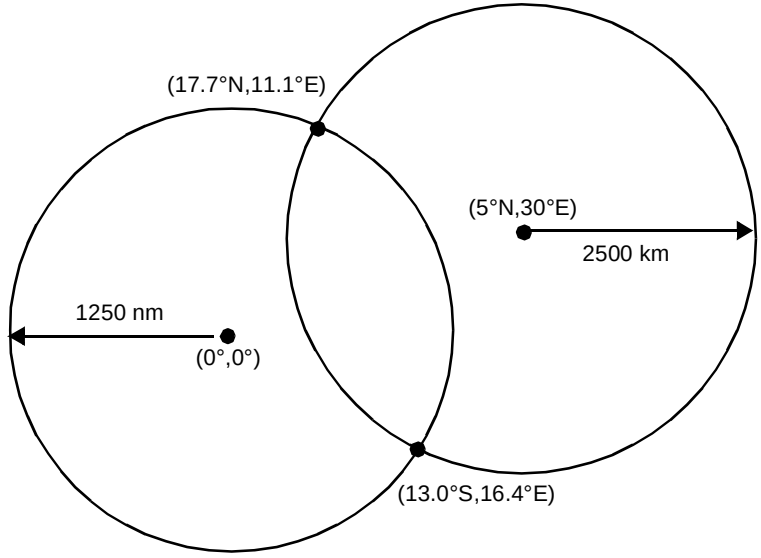
Again, three areas are returned, the largest for the polygon representing Superior, Michigan, and Huron together, the other two for Erie and Ontario.

Vector Calculations – Intersections

The Mapping Toolbox provides a set of functions to perform intersection calculations on vector data computed by the toolbox, which include great and small circles, as well as rhumb line tracks. The functions do not, however, determine intersections of arbitrary vector data.

Compute the intersection of a small circle centered at (0°,0°) with a radius of 1250 nautical miles and a small circle centered at (5°N,30°E) with a radius of 2500 kilometers:

```
[lat,long] = scxsc(0,0,nm2deg(1250),5,30,km2deg(2500))
lat =
  17.7487 -12.9839
long =
  11.0624 16.4170
```

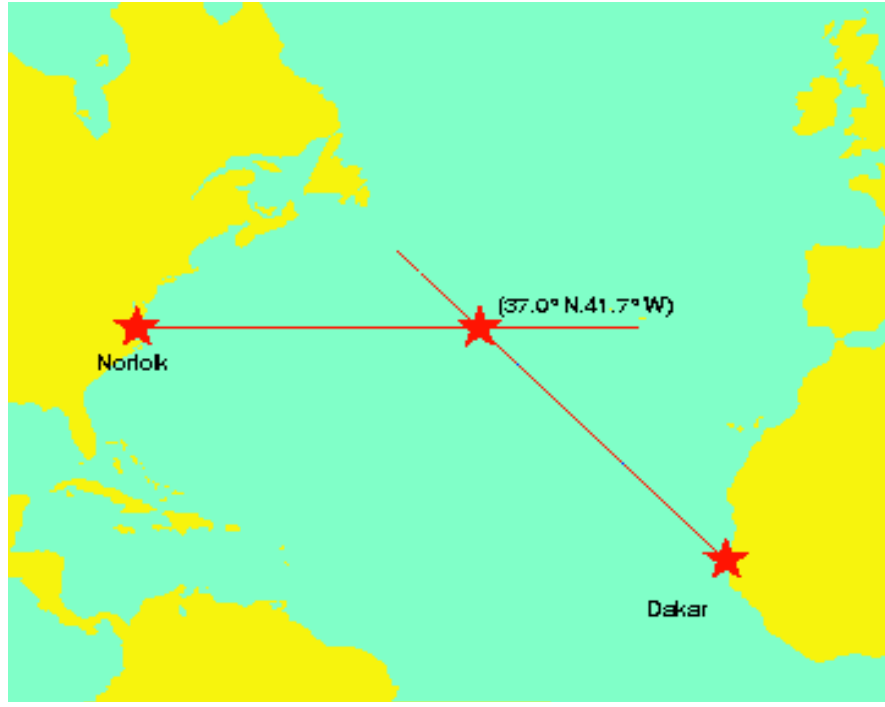


Notice that, in general, small circles intersect twice or never. For the case of exact tangency, `scxsc` returns two identical intersection points. Other similar commands include `rhxrh` for intersecting rhumb lines, `gcxgc` for intersecting great circles, and `gcxsc` for intersecting a great circle with a small circle.

Imagine a ship setting sail from Norfolk, Virginia (37°N,76°W), maintaining a steady due-east course (90°), and another ship setting sail from Dakar, Senegal (15°N,17°W), with a steady northwest course (315°). Where would the tracks of the two vessels cross?

```
[lat,long] = rhxrh(37,-76,90,15,-17,315)
lat =
  37
long =
  -41.7028
```

The intersection of the tracks is at (37°N,41.7°W), which is roughly 600 nautical miles west of the Azores in the Atlantic Ocean.



Trimming Vector Data to a Defined Region

Sometimes vector data may extend beyond the geographic region of interest. For example, if you have world coastline data, but you require a map of Australia only, you may want to create new variables that contain only the data you need. You might also want to trim your data to save memory or to speed up calculations and display.

We can distinguish between trimming line data and patch data. Line data can be trimmed by simply removing points outside the region of interest. Patch data requires a more complicated method to ensure that the patch objects are correctly displayed. For the vector data, two functions are available to achieve this. If the variables are to be treated as line data, the `maptriml` command will return variables containing only those points in the defined region. If polygon format must be retained, the `maptrimp` command will ensure this, although compared to the line formatted data, the patch-trimmed data will be larger and take more time to compute.

Note: For viewing purposes, trimming the data before display is not required. The vector data is automatically trimmed to the region specified by the frame limits (`FLatLimit` and `FLonLimit` map axes properties) for azimuthal projections and frame or map limits (`MapLatLimit` and `MapLonLimit` map axes properties) for non-azimuthal projections. The trimming is done internally in the display routine, so the original data remains intact. See “Map Axes” in Chapter 2 of this document, along with the reference pages for the trimming functions in the “Map Reference” section of the *Mapping Toolbox Reference Guide*, for further information on trimming geographic vector data.

Load the coast MAT-File, and trim the vector data to a region centered on Australia:

```
load coast
whos
   Name             Size      Bytes   Class

   lat            9589x1      76712   double array
   long           9589x1      76712   double array

latlim = [-50 0]; longlim = [105 160];
[linelat,linelong] = maptriml(lat,long,latlim,longlim);
[polylat,polylong] = maptrimp(lat,long,latlim,longlim);

whos
   Name             Size      Bytes   Class

   lat            9589x1      76712   double array
   latlim            1x2         16   double array
   linelat         870x1       6960   double array
   linelong        870x1       6960   double array
   long           9589x1      76712   double array
   longlim           1x2         16   double array
   polylat        1020x1       8160   double array
   polylong       1020x1       8160   double array
```

Simplifying Vector Data

Sometimes you may need to reduce the number of points in your vector data, while still maintaining an accurate representation of the geographical data. A quick and easy method would be to throw out every other element or set of elements in each vector. However, the result may be a very crude representation of the original data. Another option exists. The Mapping Toolbox provides a function implementing a powerful line simplification algorithm that selectively deletes points.

Note: Reduced geographic data may not always be appropriate for display. If all intermediate points in a dataset are reduced, then lines appearing straight in one projection will be incorrectly displayed as straight lines in others.

For example, reduce the number of points in the state of Massachusetts with the `reducem` function:

```
load usahi
whos
   Name            Size      Bytes   Class

   stateline       1x51     837656   struct array
   statepatch      1x51     837758   struct array
   statetext       1x51      51190   struct array
```

The `stateline` structure contains vector data for all 50 U.S. states and the District of Columbia. We are only interested in the state of Massachusetts, the 19th entry in the structure:

```
lat = stateline(19).lat;
long = stateline(19).long;
clear state*
```

Now simplify your vector data:

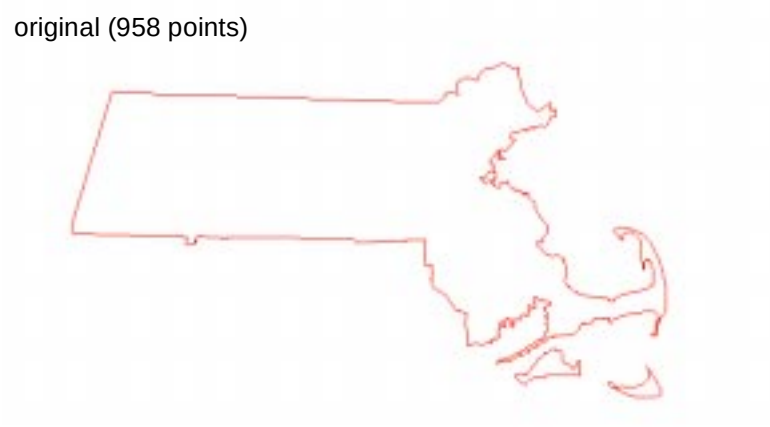
```
[newlat,newlong,cerr,tol] = reducem(lat,long);
whos
   Name            Size      Bytes   Class

   cerr             1x1          8   double array
   lat            958x1       7664   double array
   long           958x1       7664   double array
   newlat         253x1       2024   double array
   newlong        253x1       2024   double array
   tol              1x1          8   double array
```

The Massachusetts vector data has been reduced to almost a quarter of its original size:

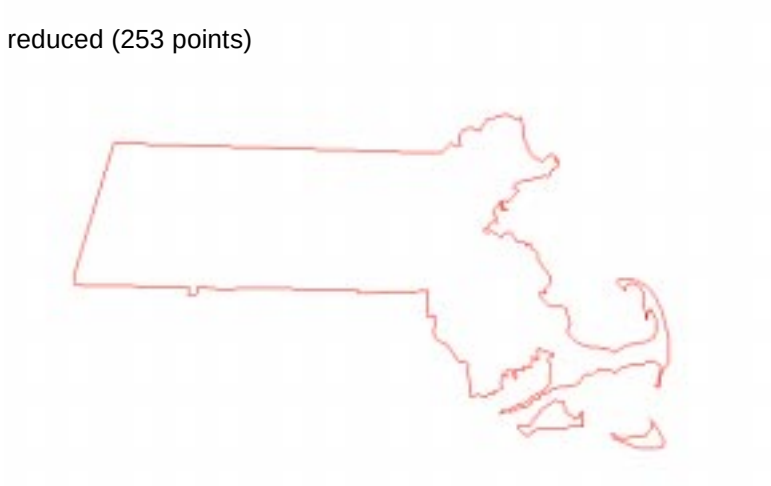
```
size(newlat)/size(lat)
ans =
  0.2641
```

Plots of the datasets, however, show two virtually identical maps:

original (958 points)



reduced (253 points)

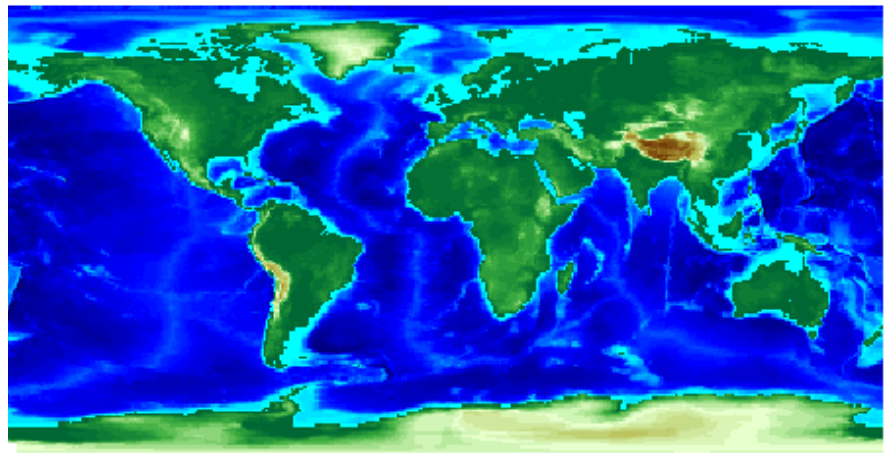


Working with Matrix Maps

One of the most powerful features of the Mapping Toolbox is its capacity for manipulating matrix maps. Two types of matrix maps are defined by the Mapping Toolbox: regular and general. Regular matrix maps are rectangular in appearance and are a subset of general matrix maps, which can be of practically any shape or orientation.

Regular Matrix Maps

A *regular matrix map* is one in which columns of data run south to north, rows of data run west to east, and each matrix element represents the same angular step in each direction, for all rows and columns. For example, if each row represents one degree of latitude, and each column one degree of longitude, and the matrix is aligned north and south, then it is a regular matrix map. The topo map provided with MATLAB is such a regular matrix map:



The Map Legend

Many regular matrix maps do not cover the entire planet. It is therefore necessary to identify the scale and placement of the map with a special data structure, the *map legend*. The map legend is a three-element vector that identifies the size of each matrix entry and the coordinates on the globe of the northwestern corner of the map. The three entries are given in terms of angular units of degrees only. The specific format of the structure is as follows:

```
maplegend = [cells/angleunit north-latitude west-longitude]
```

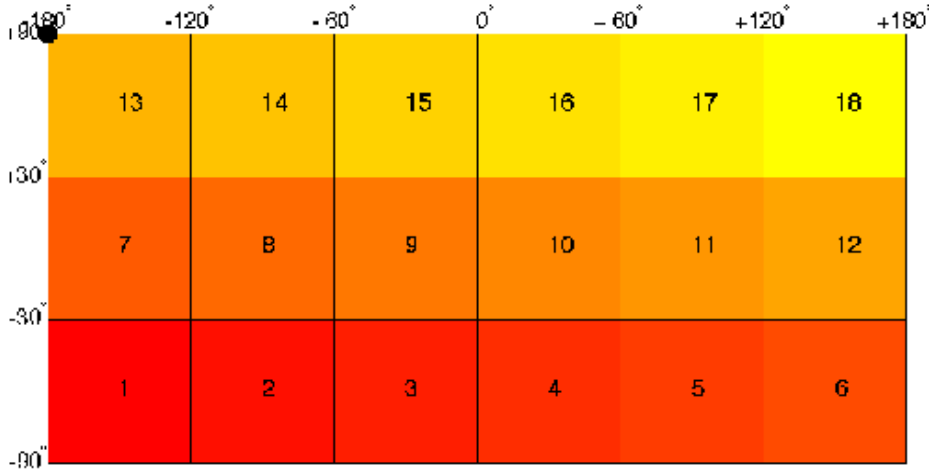
For the `topo` matrix map, each matrix cell represents one degree, the northern edge is the North Pole, and the western edge is the Prime Meridian. Therefore, the map legend for `topo` is:

```
topolegend = [1 90 0]
```

This map legend structure is stored in the `topo` workspace. Imagine an extremely coarse map of the world in which each cell represents 60°. Such a map matrix would be 3-by-6, and its map legend would be:

```
maplegend = [1/60 90 -180] = [0.0167 90 -180]
```

In this case, the map's western edge is the International Date Line, at 180°W:



Note that the first row of the matrix is displayed as the bottom of the map, while the last row is displayed as the top. All regular matrix maps in the Mapping Toolbox, as well as regular surfaces in MATLAB, are displayed in this fashion.

As an example of a matrix map that does not encompass the entire world, load the Cape Cod image using the loadcape script.

```
loadcape
maplegend
maplegend =
  120    44   -72
```

The maplegend indicates that there are 120 cells per angular degree. This has a horizontal resolution 120 times finer than that of the topo matrix map, which was one cell per degree. The cape region covered here extends from 41°N to 44°N, and from 72°W to 69°W.

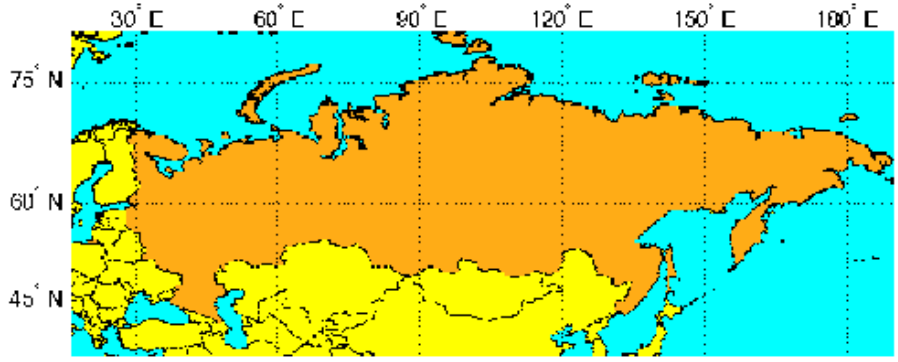
```
[latlimits,longlimits] = limitm(map,maplegend)
latlimits =
  41    44
longlimits =
  -72   -69
```

The Geographic Meaning

A matrix map and its associated map legend can provide a host of geographic information regarding the map and its entries. First, as was just demonstrated, the north, south, east, and west limits of the mapped area can be determined as follows:

```
load russia
[latlim,longlim] = limitm(map,maplegend)
latlim =
  35    80
longlim =
  15   190
```

The map in the `russia` workspace extends over the International Date Line (180° longitude). You could use the previously described command `npi2pi` to rename the eastern limit to be -170, or 170°W.



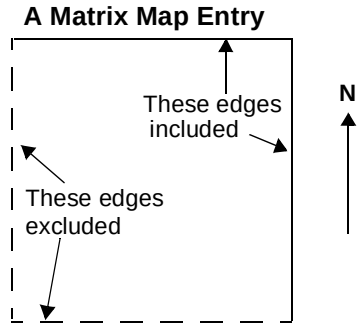
The command `setltln` allows you to determine the geographic coordinates of a particular matrix element. The returned coordinates actually show the center of the geographic area represented by the matrix entry:

```
row = 23; col = 79;
[lat,long] = setltln(map,maplegend,row,col)
lat =
  39.5000
long =
  30.7000
```

You can also determine the row and column of the matrix map entry containing a given point:

```
[r,c] = setpostn(map,maplegend,lat,long)
r =
  23
c =
  79
```

Each matrix entry can be thought of as an angular *square* which includes its northern and eastern edges, but not its western and southern edges:



The exceptions to this are that the southernmost row (row 1) also contains its southern edge, and the westernmost column (column 1) contains its western edge, except when the map encompasses the entire 360° of longitude. In that case, the westernmost edge of the first column is not included, because it is identical to the easternmost edge of the last column. These exceptions ensure that all points on the globe can be represented exactly once in a regular matrix map.

Although each map matrix entry represents an area, not a point, it is often convenient to assign singular coordinates for reference. An entry is defined by the point in the center of the area represented by the entry. For example, reference the center cell coordinate for the 3rd row, 17th column, of the Russia map:

```
row = 3; col = 17;
[lat,long] = setltln(map,maplegend,row,col)
lat =
  35.5000
long =
  18.3000
```

Since the cells in the Russia matrix map represent 0.2° squares (5 cells per degree), the cell in question extends from 35.6°S to 35.4°S and from 18.2°E to 18.4°E.

Accessing Matrix Map Elements

The actual values contained within the map matrix entries are important as well. The Mapping Toolbox provides several functions for accessing and altering the values of matrix map entries.

If the actual row and column of a desired entry are known, then a simple matrix index can return the appropriate value. Use the row and column from the last example (3rd row, 17th column) to determine the value of that cell:

```
value = map(row,col)
value =
  2
```

More often, the geographic coordinates are known:

```
value = ltln2val(map,maplegend,lat,long)
value =
  2
```

The latitude-longitude coordinates associated with particular values in a matrix map can be found with `findm`, which is similar to the standard MATLAB command `find`. Here we find the coordinates in the `topo` map with values greater than 5500 meters:

```
load topo
[lats,longs] = findm(topo>5500,topolegend);
[lats longs]
ans =
  34.5000   79.5000
  34.5000   80.5000
  30.5000   84.5000
  28.5000   86.5000
```

You can change all instances of a given value in a matrix map to a new value in one step. Here is an example using the map of Russia:

```
oldcode = ltln2val(map,maplegend,37,79)
oldcode =
  4
newmap = changem(map,5,oldcode);
newcode = ltln2val(newmap,maplegend,37,79)
newcode =
  5
```

All entries in `newmap` corresponding to 4s in `map` now have the value 5. You can also define a mask to determine map entries to change. A mask is a matrix the same size as the map matrix, with 1s everywhere the entry is to take on the new value. A mask is often generated by a logical operation on a map variable, a topic which is described in greater detail below. The map in the `russia` workspace contains 3s for Russia. If you want to set every non-Russia matrix entry to zero, you can use `maskm`:

```
nonrussia = (map~=3);
newmap = maskm(map,nonrussia,0);
newcode = ltln2val(newmap,maplegend,37,79)
newcode =
  0
```

Finally, if you know the latitude and longitude limits of a region, you can calculate the required matrix size and an appropriate map legend for a desired map scale. Before making a large, memory-taxing matrix map, you should first determine its size. For a map of the continental United States at a scale of 10 cells per degree, do the following:

```
scale = 10;
[r,c,maplegend] = sizem([25 55],[-130 -60],scale)
r =
  300
c =
  700
maplegend =
  10    55  -130
```

This matrix map would be 300-by-700. What if the scale were reduced to 5 rows/columns per degree?

```
scale2 = 5;
[r,c,maplegend] = sizem([25 55],[-130 -60],scale2)
r =
  150
c =
  350
maplegend =
  5    55  -130
```

A 150-by-300 matrix might be more manageable.

Valued Maps and Indexed Maps

A *valued map* is a matrix map in which the entries represent some value or measurement. A simple example of a valued map is the topo matrix in the topo workspace. Each entry of topo is an average elevation in meters for that piece of the Earth. The entries of a valued map answer the question, *how much is here?*

An *indexed map* is a matrix map in which the entries are an index value indicating where more information might be found. A simple example of an indexed map is the `map` variable in the `worldmtx` workspace. Each entry of `map` is an index to the `nations` structure indicating which nation contains that piece of the Earth. The entries of an indexed map answer the question, *what is this?*

To illustrate the difference between valued maps and indexed maps, consider two geographic points, (45°N,105°E) and (47°N,108°E). The valued map `topo` contains a different kind of information for these points from what is found in the indexed world `map`:

```
load topo
ltln2val(topo,topolegend,45,105)
  1270
ltln2val(topo,topolegend,47,108)
ans =
  1513
```

As might be expected, the two regions represented by the appropriate entries of `topo` have different average elevation values. However, they have the same index value in `map`, because they are in the same country:

```
load worldmtx
ltln2val(map,maplegend,45,105)
ans =
  119
ltln2val(map,maplegend,47,108)
ans =
  119

nations(119).name
ans =
Mongolia
```

Matrix Maps as Logical Variables

A powerful use of matrix maps is the application of logical conditions to create *logical maps*. Logical maps are matrix maps consisting entirely of 1s and 0s. They can be created by performing logical tests on matrix map variables. The resulting binary matrix is the same size as the original map(s) and can use the same map legend, as the following illustrates:

```
logicalmap = (realmap>0)
```

Multiple conditions can operate on the same map:

```
logicalmap = (realmap>-100)&(realmap<100)
```

Or, if several maps are the same size and share the same map legend, a logical map can be created by testing conditions on several *layers* of maps:

```
logicalmap = (population>10000)&(elevation<400)&...
             (country==nigeria)
```

The Mapping Toolbox provides functions that enable the creation of logical maps. For example, the 5-cell-per-degree maps covering the continental United States of all 1s and all 0s can be easily constructed:

```
latlims = [25 55]; longlims = [-130 -60]; scale = 5;
onesmap = onem(latlims,longlims,scale);
zeroesmap = zerom(latlims,longlims,scale);
```

Maps of all `NaN`s and sparse maps of all 0s can be made in a similar fashion with the commands `nanm` and `spzerom`.

A useful way of analyzing the results of logical map manipulations is to determine the area satisfying the conditions (i.e., with a logical value of 1). The command `areamat` can provide the fractional surface area on the globe associated with 1s in a logical map. Each matrix element is a quadrangle, and the sum of the areas meeting the logical condition provides the total area. For example, according to the `topo` map, what fraction of the Earth lies above sea level?

```
load topo
a = areamat((topo>0),topolegend)
a =
  0.2890
```

The answer is about 30%. If a planetary radius is provided in, for example, kilometers, the total physical area above 0 elevation can be returned in square kilometers:

```
a = areamat((topo>0),topolegend,almanac('earth','radius'))
a =
  1.4739e+08
```

To interpret this graphically, remember that the logical matrix map (topo>0) is binary. The following is a display of the map with black where the condition is true (i.e., the logical map entry is “1”) and white where it is false (“0”). It is displayed in an Equal-Area Cylindrical projection, which vertically shrinks rows near the poles to show their true relative area. Similarly, the `areamat` command *shrinks* the contribution of these rows to the total area.

General Matrix Maps

In addition to regular matrix maps, the Mapping Toolbox provides another matrix map format: *general matrix maps*. These maps can be displayed, and their values and coordinates can be queried, but much of the functionality available for regular matrix maps cannot be exploited for general matrix maps.

The examples thus far have shown maps that covered simple, regular quadrangles, that is, geographically rectangular. General matrix maps, in addition to these rectangular orientations, can have other shapes as well.

The Map Format

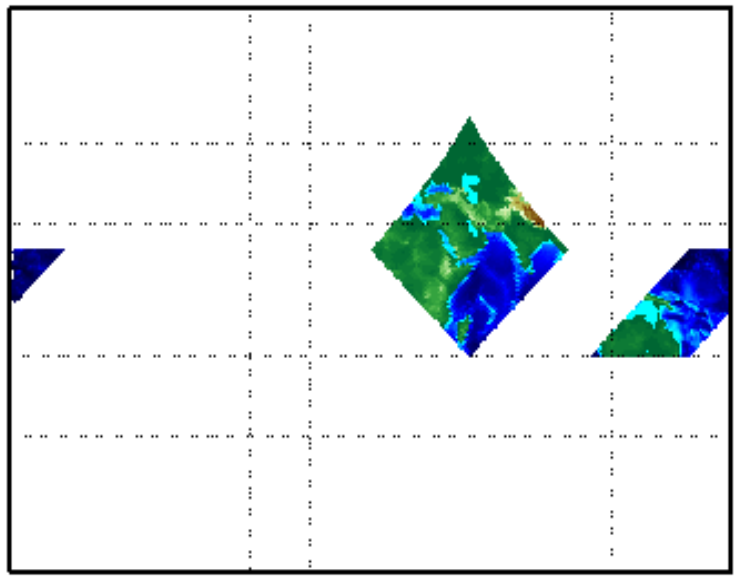
To define a general matrix map, you need three variables: the matrix of indices or values associated with the mapped region, a matrix giving cell-by-cell latitude coordinates, and a matrix giving cell-by-cell longitude coordinates.

An example of an irregularly shaped general matrix map is available in the Mapping Toolbox for examination:

```
load mapmtx
whos
   Name            Size      Bytes   Class

   lg1            50x50      20000   double array
   lg2            50x50      20000   double array
   lt1            50x50      20000   double array
   lt2            50x50      20000   double array
   map1           50x50      20000   double array
   map2           50x50      20000   double array
```

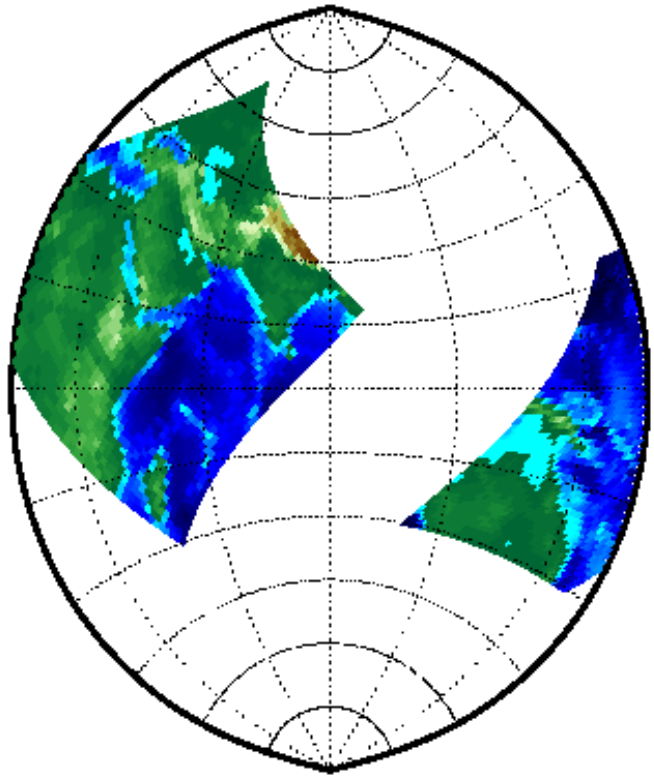
Two general matrix maps are in this workspace, each requiring three variables. The values contained in `map1` correspond to the latitude and longitude coordinates, respectively, in `lt1` and `lg1`. Notice that all three matrices are the same size. Similarly, `map2`, `lt2`, and `lg2` together form a second general matrix map. These datasets were extracted from the `topo` matrix map shown in previous examples. Neither of these maps is regular in that their columns do not run north-and-south.

To get an idea of their geography, display them together:



Notice that neither map is a regular rectangle. One looks like a diamond geographically, the other like a trapezoid. The trapezoid is displayed in two pieces because it crosses the edge of the map. These shapes can be thought of as the geographic organization of the data, just as rectangles are for regular matrix maps. But, just as for regular matrix maps, this organizational logic does not mean that displays of these maps are necessarily a specific shape.

Here you can view these maps together in a Polyconic projection:



Since the Polyconic is limited to a 150° range in longitude, those portions of the maps outside this region are trimmed automatically.

Geographic Interpretations

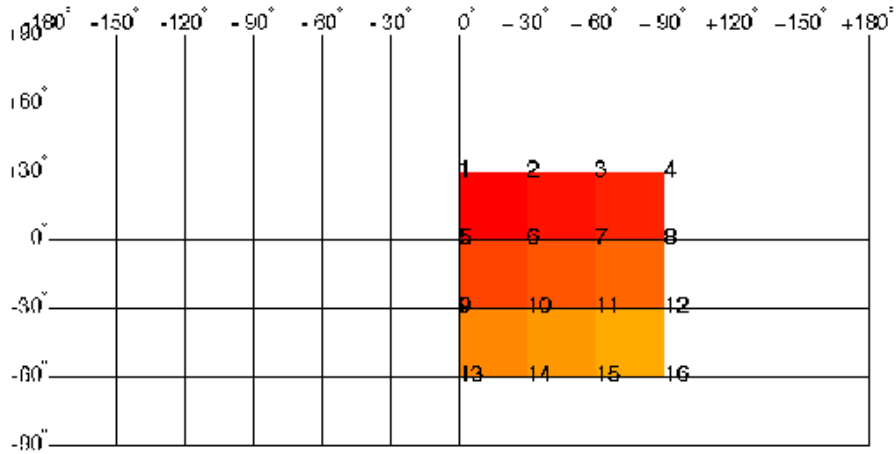
The Mapping Toolbox supports several different interpretations of general matrix maps. First, a map matrix having the same size as the latitude and longitude coordinate matrices represents the values of the map data at the corresponding geographic points. Next, a map matrix having one fewer row and column than the geographic coordinate matrices represents the values of the map data within the area formed by the four adjacent latitudes and longitudes. Finally, if the latitude and longitude matrices are still smaller than the map matrix, you can interpret them as describing a coarse *graticule*, or mesh of latitude and longitude cells into which the blocks of map data are warped.

This section discusses the first two interpretations of general matrix maps. Chapter 2, "Displaying Maps" has more information on the use of graticules.

As an example of the first interpretation, consider a 4-by-4 map matrix whose cell size is 30-by-30 degrees, along with its corresponding 4-by-4 latitude and longitude matrices:

```
map = [ 1  2  3  4;...
  5  6  7  8;...
  9 10 11 12;...
  3 14 15 16];

lat = [ 30  30  30  30;...
  0   0   0   0;...
  -30 -30 -30 -30;...
  -60 -60 -60 -60];

long = [0 30 60 90;...
  0 30 60 90;...
  0 30 60 90;...
  0 30 60 90];
```

This general matrix map is displayed with the values of `map` shown at the associated latitudes and longitudes:

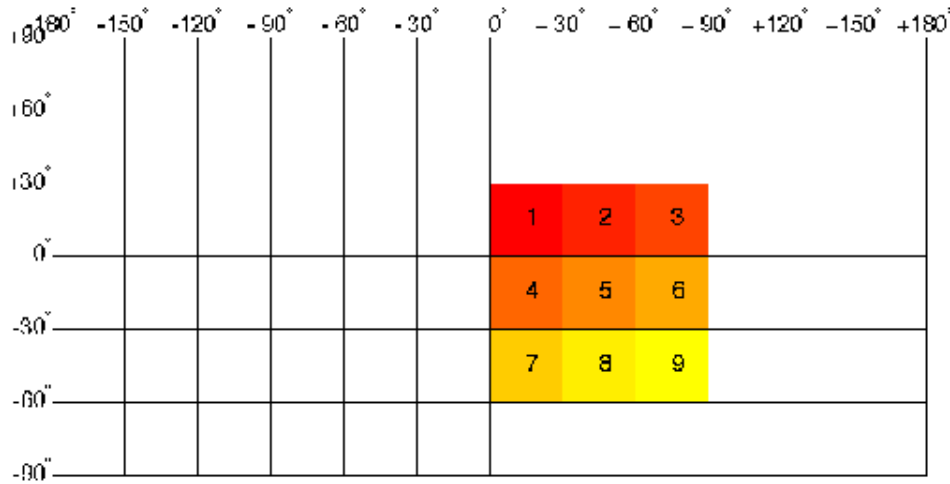


Notice that only 9 of the 16 total cells are displayed. The value displayed for each cell is the value at the upper left corner of that cell, whose coordinates are given by the corresponding `lat` and `long` elements. By MATLAB convention, the last row and column of the map matrix is not displayed, although they exist in the `CData` property of the surface object.

For the second interpretation, consider a 3-by-3 map matrix with the same `lat` and `long` variables:

```
map = [1 2 3;...
       4 5 6;...
       7 8 9];
```

Here is a surface plot of the map matrix, with the values of map shown at the center of the associated cells:



All of the map data is displayed for this general matrix map. The value of each cell is the value at the center of the cell, and the latitudes and longitudes in the coordinate matrices are the boundaries for the cells.

You may have noticed that the first row of the matrix is displayed as the top of the map, whereas for a regular matrix map, the opposite was true; the first row corresponded to the bottom of the map. This difference is due to the `lat` and `long` matrices and how they are ordered. The `lat` and `long` coordinate matrices determine the arrangement of the general matrix map.

Of course, a regular matrix map can also be considered a general matrix map. All that are required are the graticule or coordinate matrices, which can be produced from the regular matrix map itself using the `meshgrat` function:

```
load topo
[lat,lon] = meshgrat(topo,topolegend);
whos
   Name            Size      Bytes   Class

   lat          180x360     518400   double array
   lon          180x360     518400   double array
   topo         180x360     518400   double array
   topolegend       1x3         24   double array
   topomap1        64x3       1536   double array
   topomap2       128x3       3072   double array
```

A regular matrix map can also be constructed from a general matrix map. The coordinates and values can be “imbedded” in a new regular matrix map with a somewhat coarser resolution. This ensures that every cell in the new map receives a value from the old one. Convert the diamond-shaped general matrix map back to a regular matrix map.

```
load mapmtx

latlim = [min(lt1(:)) max(lt1(:))];
lonlim = [min(lg1(:)) max(lg1(:))];

[map,maplegend] = nanm(latlim,lonlim,1/2);
% original was about 1 cell per degree

map = imbedm(lt1,lg1,map1,map,maplegend);
```

2

Displaying Maps

Introduction to Mapping Graphics

One of the fundamental concepts underlying the Mapping Toolbox is the idea that maps are variables that exist independently of their display. However, for visual geographic data analysis, application development, and presentation of results, quality display capabilities are a must.

With the Mapping Toolbox, you can display geographic information almost as easily as you can plot regular data in MATLAB. Most mapping commands are similar to MATLAB commands, except they deal with latitude and longitude spherical coordinates instead of x and y Cartesian coordinates. Mapping commands typically have the same names as their MATLAB counterparts, with the addition of an `'m'`, for maps, at the end. For example, the Mapping Toolbox analog to MATLAB's `plot` command is `plotm`.

The Mapping Toolbox manages most of the steps in displaying a map. It will project your data, cut and trim it to desired limits, and display the resulting map for you. The toolbox also provides the means to add other elements to your displayed map, such as a frame, gridlines, coordinate labels, and text labels. If you change your projection properties, or even the projection itself, the Mapping Toolbox will redraw the map with the new settings, undoing any cuts or trims if necessary. See Chapter 4, "Map Projections" for information on how to project data without displaying it.

The toolbox also makes it easy to modify and manipulate maps. The map display and mapped objects can be modified either from the command line or through graphical user interfaces and property editing tools that can be invoked by clicking on the display. Most mapping display commands have graphical interfaces. See Chapter 6, "GUI Tools" for more on these capabilities.

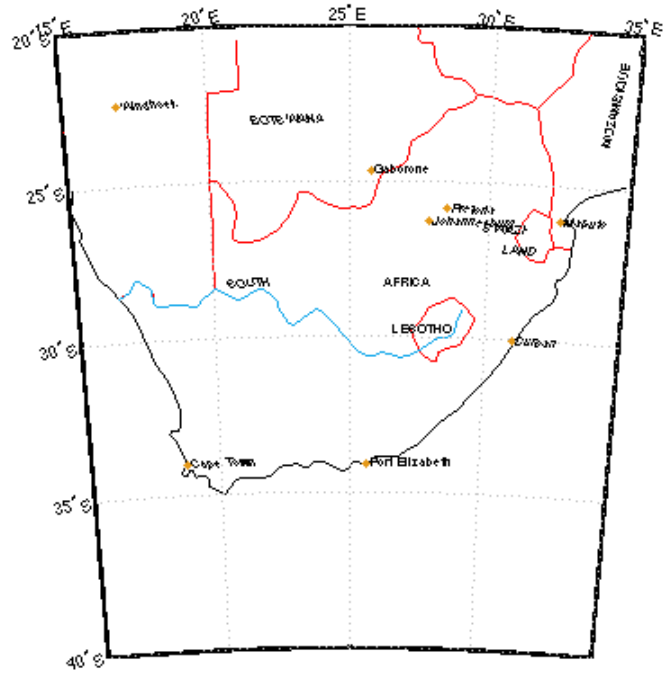
Note: An important restriction accompanies this ease of use. The Mapping Toolbox manages the map display with the `UserData` property field in the Axes control. The `UserData` property of mapped objects is also used by the toolbox. This may cause conflicts with other functions that use the `UserData` property field.

Map Displays Made Easy

The Mapping Toolbox allows you to control many aspects of a map display. The majority of this chapter is devoted to the elements of a map display and the means to control them. However, the ability to control numerous properties is accompanied by the need to set them appropriately. You can trade control for simplicity by using the worldmap and usamap functions. These functions create maps of regions of the world or the United States using appropriate map projections and settings. You can then modify or add to the map displays using the methods described later in this chapter. You can also use these functions to prepare a map display without any atlas data, adding your own instead.

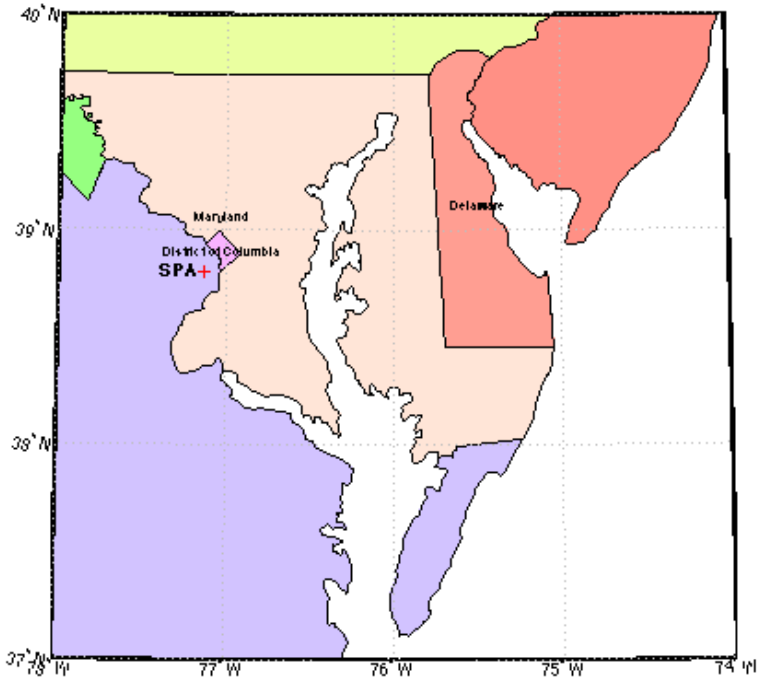
Here are some examples. Create a base map of South Africa.

```
figure
worldmap 'south africa'
hidem(gca)
```



Create a map of the Chesapeake Bay using geographic limits, and add some annotation.

```
latlim = [37 40]; lonlim = [-78 -74];
figure

usamap(latlim,lonlim)

plotm(38.8063,-77.1312,'r+')
textm(38.8063,-77.1312,' SPA ',...
'fontweight','bold','HorizontalAlignment','r')
```



Map Axes

To display maps of any kind in the Mapping Toolbox, the user must first define a map axes. A *map axes* is a normal axes that has been prepared for mapping using the `axesm` command that defines the map projection properties. Before you execute any mapping command, you need to define a map axes in the current MATLAB axes and choose a map projection using `axesm`.

Note: To successfully employ any map display function, such as `plotm`, `fillm`, or `surfacem`, the current axes must be a map axes as created by `axesm`. An axes is considered a valid map axes if its `UserData` property contains a structure defining a map projection.

A *projection* is the systematic means of representing three-dimensional geographic data in two dimensions, a topic that is covered more extensively in Chapter 4, “Map Projections”. Here, it is sufficient to understand that many different projections are available, each with different features. Some projections maintain correct relative area between objects, for example, or correct angular relationships. Some are well-suited to displaying features at certain latitudes or with certain orientations. The command `maps` displays a list of available projections.

When creating a map axes object, you must define its projection by using an appropriate ID string. All other properties can be set by default. For example, the following are all valid commands for creating a Miller projection map axes:

```
axesm miller
axesm('miller')
axesm('MapProjection','miller')
```

Other properties can be set for the map axes during creation as well:

```
axesm('MapProjection','mercator',...
      'Frame','on',...
      'AngleUnits','degrees',...
      'FontName','courier',...
      'parallellabel','on',...
      'Origin',[0 180 0])
```

If you specify some map axes parameters with axesm, the Mapping Toolbox determines appropriate values for other related map axes properties. As an example, create a map axes with an Equal-Area Conic projection for part of the world:

```
axesm('MapProjection','eqaconic',...
      MapLatLimit',[30 60],...
      MapLonLimit',[-15 45],...
      MapParallels',[])
```

The axesm function will automatically set the origin and frame limits for a map centered on the limited geographic region, with the results of subsequent mapping display commands shown only within the specified limits. Empty matrices are used to force recalculation of a property or a return to the default value. For example, setting the map parallels to an empty matrix will automatically determine map parallels appropriate for the current map limits, minimizing distortion in the projection.

Since the MapProjection property is the only property that must be explicitly set, it can be specified without the property name, as long as the projection name is the first argument in axesm. For instance:

```
axesm('miller','PropertyName1',PropertyValue1,...
      'PropertyName2',PropertyValue2,...)
```

The axesm command without any arguments activates a GUI that allows for easy manipulation of the map axes properties.

For proper map appearance and ease of use, the Mapping Toolbox changes some of the default MATLAB axes properties. The altered properties are:

- The `NextPlot` property is set to `'add'`.
- The `UserData` property contains the map projection structure that defines the specific map axes properties.
- The `Box` property is set to `'on'`.
- The `ButtonDownFcn` property is set to the function `uimaptbx`, which processes callbacks for Mapping Toolbox objects.
- The command `showaxes off` is executed to remove axes ticks from the MATLAB Cartesian axes (i.e., the `XTick`, `YTick`, and `ZTick` properties are set to empty matrices).

More information on MATLAB axes properties can be found in the "Axes" section of *Using MATLAB Graphics*. The map axes properties are defined under the `axesm` heading in the "Mapping Reference" chapter in the *Mapping Toolbox Reference Guide*, while information on the use of the `axesm` GUI can be found in the "GUI Reference" chapter in the *Mapping Toolbox Reference Guide*.

Accessing and Manipulating Map Axes Properties

Just as the properties of the underlying standard axes can be accessed and manipulated using the MATLAB commands `set` and `get`, map axes properties can also be accessed and manipulated using the commands `setm` and `getm`.

Note: Use the `axesm` command to *create* a map axes. Use the `setm` command to *modify* the map axes.

As an example, create a map axes:

```
axesm('MapProjection','miller','Frame','on')
```

Suppose you would like to make the frame lines bordering the map thicker. First you need to determine the current line width of the frame. Access the property value by typing:

```
getm(gca,'FLineWidth')
ans =
  2
```

Try resetting the line width to four points:

```
setm(gca,'FLineWidth',4)
```

You can set any number of properties simultaneously with `setm`. Reduce the line width, change the projection to an Equidistant Cylindrical, and verify the changes:

```
setm(gca,'FLineWidth',3,'MapProjection','eqdcylin')

getm(gca,'FLineWidth')
ans =
  3

getm(gca,'MapProjection')
ans =
eqdcylin
```

To inspect the entire set of map axes properties at their current settings, use the following command:

```
getm(gca)
ans =
     mapprojection: 'eqdcylin'
              zone: []
        angleunits: 'degrees'
            aspect: 'normal'
       falseeasting: []
      falsenorthing: []
       fixedorient: []
             geoid: [1 0]
       maplatlimit: [-90 90]
       maplonlimit: [-180 180]
      mapparallels: 30
        nparallels: 1
            origin: [0 0 0]
       scalefactor: []
           trimlat: [-90 90]
           trimlon: [-180 180]
             frame: 'on'
             ffill: 100
        fedgecolor: [0 0 0]
        ffacecolor: 'none'
         flatlimit: [-90 90]
        flinewidth: 3
         flonlimit: [-180 180]
              grid: 'off'
        galtitude: Inf
           gcolor: [0 0 0]
        glinestyle: ':'
        glinewidth: 0.5000
    mlineexception: []
         mlinefill: 100
        mlinelimit: []
     mlinelocation: 30
      mlinevisible: 'on'
    plineexception: []
         plinefill: 100
```

```
    plinelimit: []
 plinelocation: 15
  plinevisible: 'on'
     fontangle: 'normal'
     fontcolor: [0 0 0]
      fontname: 'helvetica'
      fontsize: 9
     fontunits: 'points'
    fontweight: 'normal'
   labelformat: 'compass'
    labelunits: 'degrees'
 meridianlabel: 'off'
mlabellocation: 30
mlabelparallel: 90
   mlabelround: 0
 parallellabel: 'off'
plabellocation: 15
plabelmeridian: -180
   plabelround: 0
```

Similarly, the possible property values and their defaults can be displayed with the `setm` command alone:

```
setm(gca)

AngleUnits[ {degrees} | radians | dms | dm ]
Aspect[ {normal} | transverse ]
FalseEasting
FalseNorthing
FixedOrientFixedOrient is a read-only property
Geoid
MapLatLimit
MapLonLimit
MapParallels
MapProjection
NParallelsNParallels is a read-only property
Origin
ScaleFactor
TrimLatTrimLat is a read-only property
TrimLonTrimLon is a read-only property
Zone
Frame[ on | {off} ]
FEdgeColor
FFaceColor
FFill
FLatLimit
FLineWidth
FLonLimit
Grid[ on | {off} ]
GAltitude
GColor
GLineStyle[ - | -- | -. | {:} ]
GLineWidth
MLineException
MLineFill
MLineLimit
MLineLocation
MLineVisible[ {on} | off ]
PLineException
PLineFill
PLineLimit
```

```
PLineLocation
PLineVisible[ {on} | off ]
FontAngle[ {normal} | italic | oblique ]
FontColor
FontName
FontSize
FontUnits[ inches | centimeters | normalized | {points} | pixels ]
FontWeight[ {normal} | bold ]
LabelFormat[ {compass} | signed | none ]
LabelUnits[ {degrees} | radians | dms | dm ]
MeridianLabel[ on | {off} ]
MLabelLocation
MLabelParallel
MLabelRound
ParallelLabel[ on | {off} ]
PLabelLocation
PLabelMeridian
PLabelRound
```

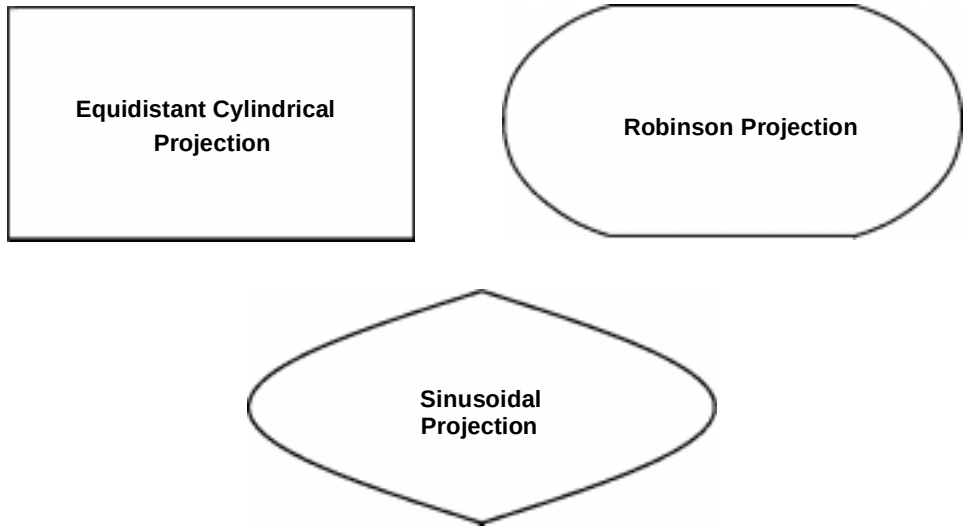
Properties can also be displayed individually:

```
setm(gca,'AngleUnits')
AngleUnits[ {degrees} | radians | dms | dm ]

setm(gca,'MapProjection')
An axes's "MapProjection" property does not have a fixed set
  of property values.

setm(gca,'Frame')
Frame[ on | {off} ]

setm(gca,'FixedOrient')
FixedOrientFixedOrient is a read-only property
```

The Map Frame

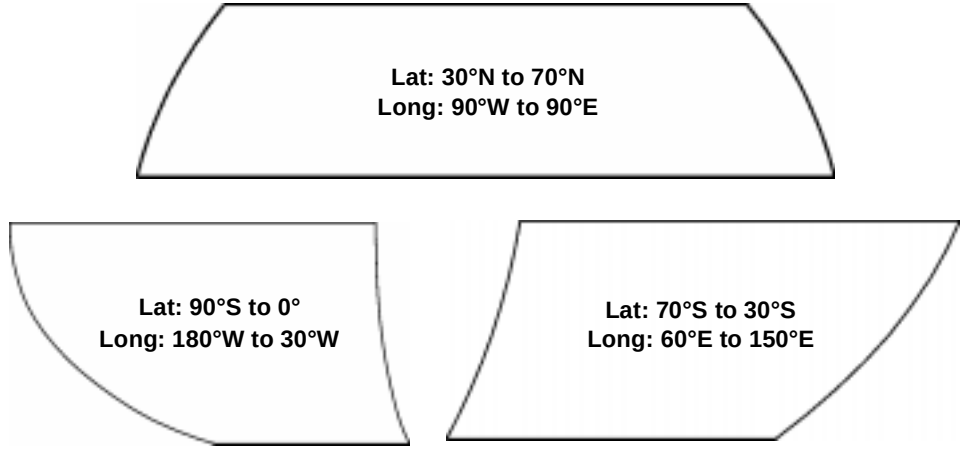
In the Mapping Toolbox, the *map frame* is a *box* enclosing the geographic display. The frame is displayed if the map axes property `Frame` is set to `'on'`. This can be accomplished upon map axes creation with `axesm`, with `setm`, or with the direct command `framem on`. The frame is geographically defined as a latitude-longitude quadrangle that is projected appropriately. For example, on a map of the world, the frame might extend from pole to pole and a full 360° in longitude. In appearance, the frame would take on the characteristic shape of the projection. The examples below are full-world frames shown in three very different projections:

Full-World Map Frames



As a map object, each of the previously displayed frames is identical; however, the selection of a display projection has varied their appearance. Since each of the examples shows the entire world, `FLatLimit` is `[-90 90]`, and `FLonLimit` is `[-180 180]` for each case. The frame quadrangle can encompass smaller regions, as well:

Frame Quadrangles Shown in the Robinson Projection (Symmetric about Prime Meridian)



For the frames shown above, the projection is centered on the Prime Meridian, or 0 longitude. Such a frame would be the result of creating a map axes with the defaults for the Robinson projection and then resetting the frame limits to cover just part of the world.

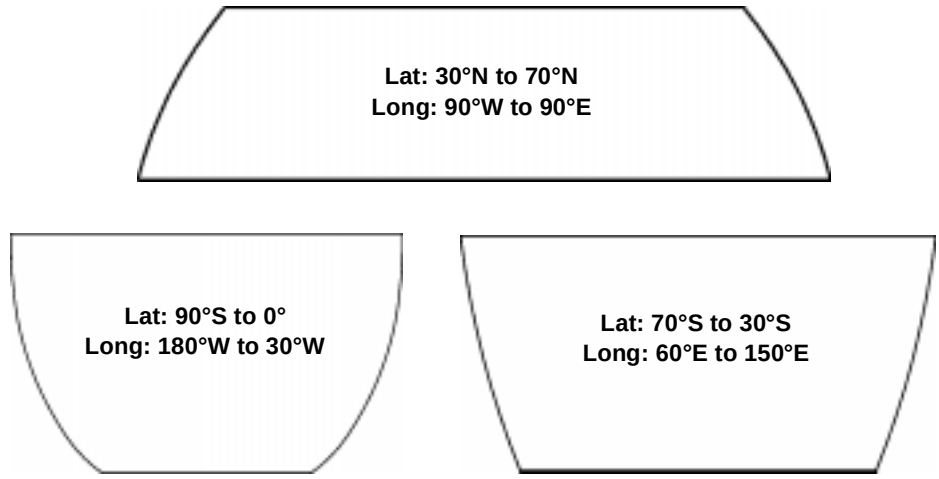
For example, to view the asymmetric frame in the lower right of the previous figure, type:

```
axesm robinson
setm(gca,'FLatLimit',[-70 -30],...
     'FLonLimit',[60 150],...
     'Frame','on')
```

When you want your frame to be symmetric about the region of interest, have `axesm` determine the proper settings for you. If you specify the map limits

without specifying the map origin and frame limits, axesm will automatically set the appropriate values for a proper symmetric frame.

Frame Quadrangles Shown in the Robinson Projection (Symmetric about Map Limits)



For example, to view the symmetric frame in the lower right of the previous figure, set the map limits with axesm:

```
axesm('MapProjection','robinson',...
      'MapLatLimit',[-70 -30],...
      'MapLonLimit',[60 150],...
      'Frame','on')
```

In addition to the latitude and longitude limits of the frame, other properties can be manipulated. The frame is actually a patch with a default face color set to `'none'` and a default edge color of black. These map axes properties can be altered by manipulating the `FFaceColor` and `FEdgeColor` properties. For example, the command:

```
setm(gca,'FFaceColor','cyan')
```

can make the background region of your display resemble water. Since the frame patch is always the lowest layer of a map display, other patches, perhaps representing land, will appear above the "water." If an object is subsequently plotted "below" the frame patch, the frame altitude can be recalculated to lie below this object with the command `framem reset`. The frame is replaced and not reprojected.

The line width of the edge, which is 2 points by default, can be set using the `FLineWidth` property.

The primary advantage of displaying the map frame is that it can provide positional context for other displayed map objects. For example, when vector data of the coasts is displayed, the frame provides the "edge" of the world.

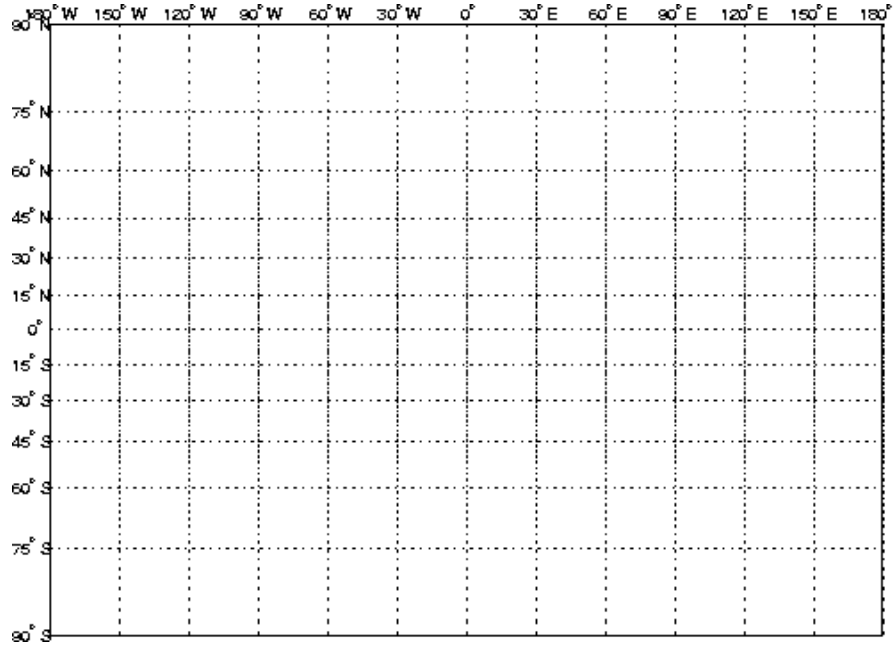
In addition to establishing frame properties upon map axes creation, or through the `setm` command, you can use the `framem` command. The command `framem` alone is a toggle for the `Frame` property, which controls the visibility of the frame. You can also call `framem` with property names and values to alter the appearance of the frame:

```
framem('FlineWidth',4,'FEdgeColor','red')
```

The Map Grid

In The Mapping Toolbox, the *map grid* is the set of displayed meridians and parallels. Display the grid by setting the map axes property `Grid` to `'on'`. This can be accomplished upon map axes creation with `axesm`, with `setm`, or with the direct command `gridm on`.

To control display of meridians and parallels, set a scalar meridian spacing or a vector of desired meridians in the `MLineLocation` property. The corresponding property `PLineLocation` serves the same purpose for parallels. The default values are every 30° for meridians and every 15° for parallels.

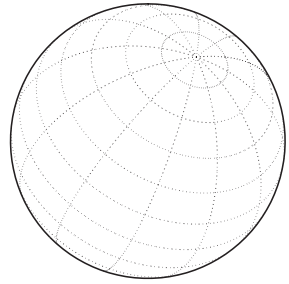


Default Grid on a Miller Projection

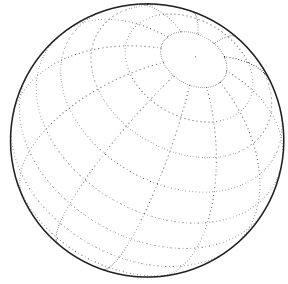
By default, the grid is placed as the top layer of any display. You can alter this by changing the `GAltitude` property, so that other map objects can be placed "above" the grid. The new grid is drawn at its new altitude.

To reposition the grid back to the top of the display, use the command `gridm reset`. You can also control the appearance of grid lines with the `GLineStyle` and `GLineWidth` properties, which are `':'` and `0.5`, respectively, by default.

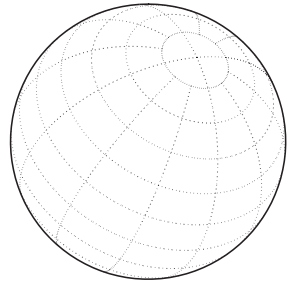
The Miller projection is an example in which all the meridians can extend to the poles without appearing to be cluttered. Look, however, at the following Orthographic projections. When all the meridians extend to the poles, the look is "congested." The `MLineLimit` property allows you to specify a pair of latitudes at which to terminate the meridians. For example, setting `MLineLimit` to `[-75 75]` completely clears the polar region of displayed meridians. Sometimes you might like to allow some of the meridians to ignore the limit and extend to the pole. If, for example, `MLineException` is set to `[-90 0 90 180]`, then the meridians corresponding to those four longitudes will continue past the limit on to the pole:



Default grid allows all displayed meridians to extend to the poles.



The property `MLineLimit` can truncate meridians at a given latitude, here at 75°N and S.



The property `MLineException` allows certain meridians to extend to the poles despite the `MLineLimit`. Here, four meridians 90°W, 0°, 90°E, and 180° are excepted.

There are also corresponding map axes properties for controlling the extent of displayed parallels: `PLineLimit` and `PLineException`.

Users can label displayed parallels and meridians. `MeridianLabel` and `ParallelLabel` are on-off properties for the display of labels on the meridians and parallels, respectively. They are both `'off'` by default. Initially, the label locations coincide with the default displayed grid lines, but you can alter this by using the `PLabelLocation` and `MLabelLocation` properties. These grid lines are labeled across the north edge of the map for meridians and along the west edge of the map for parallels. However, the property `MLabelParallel` allows you to specify `'north'`, `'south'`, `'equator'`, or a specific latitude at which the meridian labels will be displayed, and the `PLabelMeridian` allows the choice of `'west'`, `'east'`, `'prime'`, or a specific longitude for the parallel labels. Refer to the `axesm` reference page in the *Mapping Toolbox Reference Guide* for complete descriptions of all map axes properties.

Map and Frame Limits

In the Mapping Toolbox, the map and frame limits are two related map axes properties that limit the map display to a defined region. The map latitude and longitude limits restrict the displayed parallels and meridians, while the frame limits control the extent of the frame around the displayed data. Any object that extends outside the frame limits is automatically trimmed.

The frame limits are also specified differently from the map limits. The map limits are in Greenwich coordinates referenced to an origin at the intersection of the Prime Meridian and the Equator, while the frame limits are in coordinates referenced to the center of the frame, which is the latitude and longitude of the map axes origin.

For all non-azimuthal projections, frame limits are specified as quadrangles ([*latmin latmax*] and [*longmin longmax*]) in the frame coordinate system. In the case of azimuthal projections, the frames are circular and are described by a polar coordinate system. One of the frame latitude limits must be a negative infinity (-Inf) to indicate an azimuthal frame (think of this as the center of the circle), while the other limit determines the radius of the circular frame (*rlatmax*). The longitude limits of azimuthal frames are inconsequential, since a full circle is always displayed.

If you are uncertain about the correct format for a particular projection frame limit, you can reset them to the default values using empty matrices.

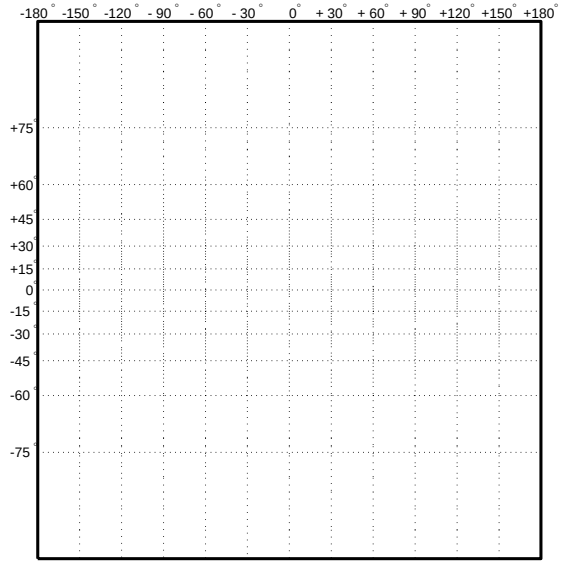
Note: For non-azimuthal projections in the normal aspect, the map is limited to the minimum of the map and frame limits; hence, the two limits will coincide after evaluation. When changing one set of limits, you may want to clear the other set to get consistent limits.

Switching Between Projections

When switching between projections using `setm` or the graphical interfaces, you may need to change some of the map axes properties for proper appearance. Settings that are suitable for one projection may not be appropriate for another. Some projections have default properties that define *that* particular projection and may not be altered; for example, the Balthasart Cylindrical projection is defined to have standard parallels (`MapParallels`) at 50°. Other projections have default properties that are initially set for proper world display; for example, the Mercator projection limits the latitude range to ±86° to avoid “blowing up” at the poles.

Although similar projections may share the same set of properties (Miller Cylindrical and Plate Carree Cylindrical), others may be drastically different (Polyconic and Stereographic azimuthal). The classification of the map projections is often a good indicator of whether changes need to be made. For instance, switching from a cylindrical to an azimuthal projection requires a few modifications:

```
axesm mercator
framem on; gridm on; mlabel on; plabel on
setm(gca,'LabelFormat','signed')
```



To display the default map and frame latitude limits for the Mercator projection, type the following:

```
[getm(gca,'MapLatLimit'); getm(gca,'FLatLimit')]
ans =
      -86    86
      -86    86
```

Both the frame and map latitude limits are set to ±86° for the Mercator projection to maintain a safe distance from the singularity at the poles.

Now switch the projection to an Orthographic azimuthal:

```
setm(gca,'MapProjection','ortho')
```

What happened to the map frame and labels? If you recall, the frame latitude limits have not been changed and still correspond to the default values for a Mercator projection, as do all the other properties. Only those properties that *must* have particular values are updated for the current projection.

```
getm(gca,'FLatLimit')
ans =
      -86    86
```

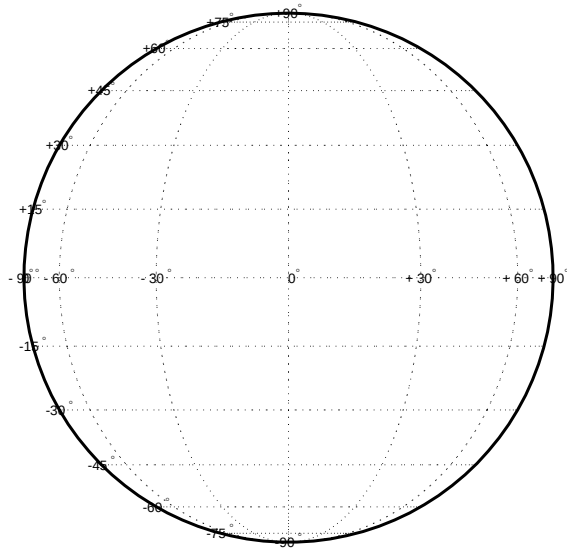
You must manually reset the frame and map limits to appropriate values for an Orthographic projection so that the circular frame is displayed. If you don't know the default or appropriate values, provide an empty matrix for any of the property values:

```
setm(gca,'FLatLimit',[],'MapLatLimit',[])
[getm(gca,'MapLatLimit'); getm(gca,'FLatLimit')]
ans =
      -90     90
      -Inf    89
```

You will also need to manually specify the locations of the meridian and parallel labels:

```
setm(gca,'MLabelParallel',0,'PLabelMeridian',-90)
```

Now the map is displayed correctly, with the frame:



Default property values can be accessed by the use of empty matrices, as shown in the last example, and the properties can be reset according to the "new" values. The entire set of properties can be reset to their default values by pressing the **Reset** button on the `axesmui` GUI.

For complete descriptions of all map axes properties, consult the `axesm` reference page in the "Mapping Reference" section of the *Mapping Toolbox Reference Guide*. For more information on the use of `axesmui`, refer to its reference page in the "GUI Reference" section of the *Mapping Toolbox Reference Guide*.

Projected and Unprojected Objects

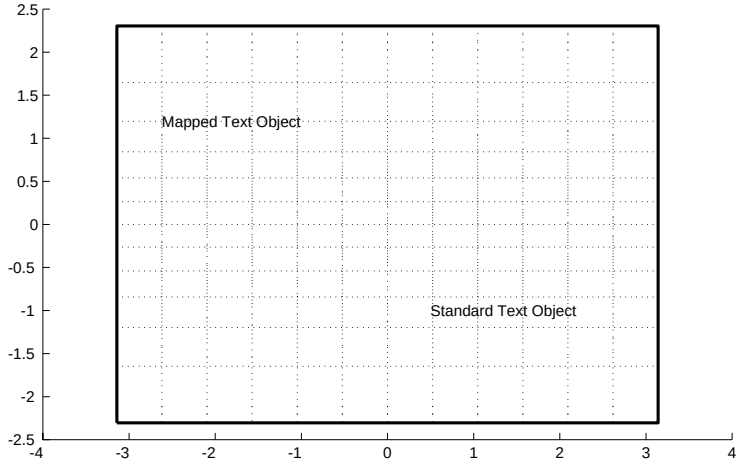
All objects displayed using Mapping Toolbox commands are projected on the map axes based on their designated latitude-longitude positions. The latitudes and longitudes are mathematically transformed to x and y positions using the formulas for the current map projection. Objects displayed using standard MATLAB commands (like `plot`, `fill`, `surface`, and `text`) are placed in the designated *x-y* location. If the projection changes for a map axes (for example, by using the `setm` command to alter the `MapProjection` property), those objects displayed with standard commands are unaffected, but those displayed using mapping commands are reprojected into new positions. Here is an example:

```
axesm miller; framem on; gridm on;
showaxes; grid off;
```

This set of commands creates a map axes, a map frame enclosing the region of interest, and geographic grid lines. The x-y axes, which are normally hidden, are displayed, and the MATLAB x-y grid is turned off. It should be noted that the MATLAB x-y `grid` is separate from the Mapping Toolbox geographic version `gridm`.

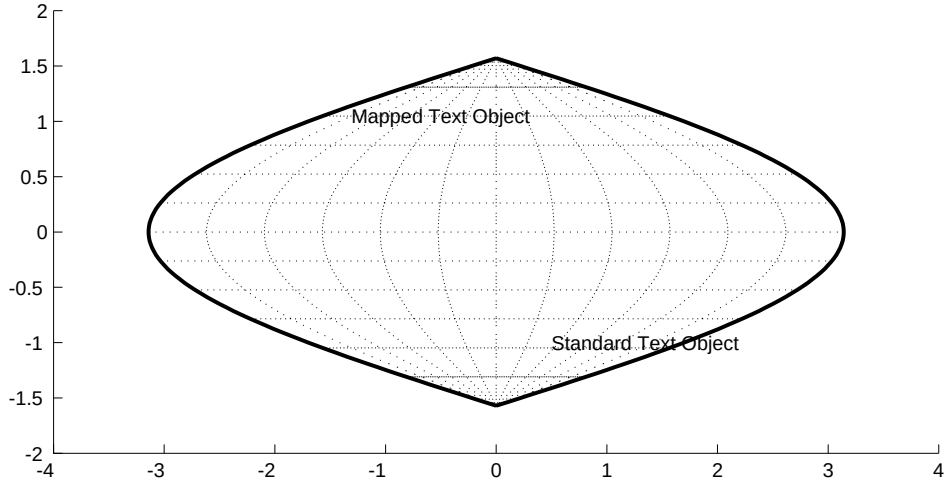
```
text(.5,-1,'Standard Text Object')
textm(60,-150,'Mapped Text Object')
```

Here, a standard text object is placed at x=0.5 and y=-1, while a mapped text object has been placed at (60°N,150°W) in the Miller projection.



If we change the projection to a Sinusoidal, the standard text object remains at the same Cartesian position, so its latitude-longitude position is altered. The mapped text object remains at the same geographic location, so its x-y position is altered. Also, the frame and gridlines are replaced according to the new map projection:

```
setm(gca,'MapProjection','sinusoid')
showaxes; grid off;
```



Similarly, vector and matrix data can be displayed using either mapping or standard commands.

Displaying Vector Maps as Lines

The Mapping Toolbox lets you display vector map data as line objects much like the line display commands in MATLAB. The Mapping Toolbox line graphics functions have MATLAB analogs, the names of which can usually be determined by appending an `'m'` to the MATLAB function name. For instance, the Mapping Toolbox version of `plot` is `plotm`.

The following table lists the available Mapping Toolbox line display functions:

Function	Used to Create
`contorm`	Contour plot of map data
`contor3m`	Contour plot of map data in 3-D space
`linem`	Line objects projected on map axes
`plotm`	Lines projected on map axes
`plot3m`	Lines projected on map axes in 3-D space

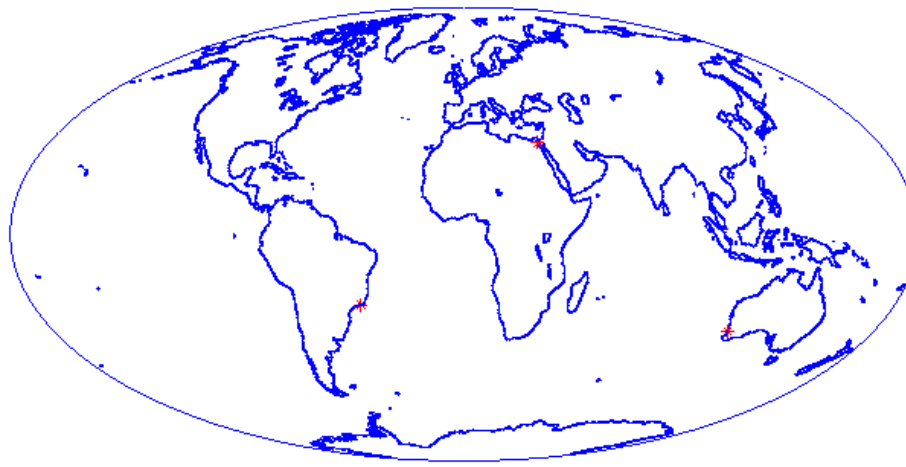
Plot the `coast` vector data using `plotm`. Just as with `plot`, you can specify line property names and values in the command:

```
load coast
axesm mollweid
framem('FEdgeColor','blue','FLineWidth',0.5)
plotm(lat,long,'LineWidth',1,'Color','blue')
```

Sometimes vector data represent specific points. Suppose you have variables representing the locations of Cairo (30°N,32°E), Rio de Janeiro (23°S,43°W), and Perth (32°S,116°E), and you want to plot them as markers only, without connecting line segments.

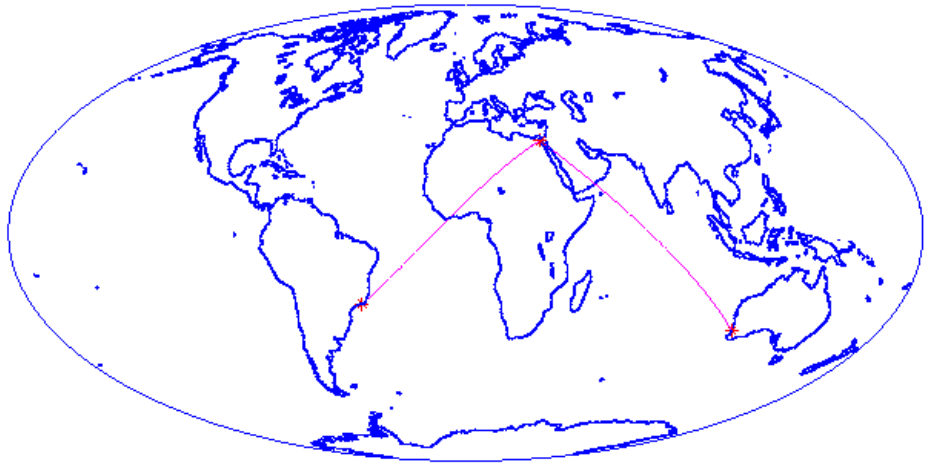
Here is the map:

```
citylats = [30 -23 -32]; citylongs = [32 -43 116];
plotm(citylats,citylongs,'r*')
```



In addition to these sorts of “permanent” geographic data, you can also display calculated vector data. Calculate and plot a great circle track from Cairo to Rio de Janeiro, and a rhumb line track from Cairo to Perth:

```
[gclat,gclong] = track2('gc',citylats(1),citylongs(1),...
                            citylats(2),citylongs(2));
[rhlat,rhlong] = track2('rh',citylats(1),citylongs(1),...
                            citylats(3),citylongs(3));
plotm(gclat,gclong,'m-'); plotm(rhlat,rhlong,'m-')
```



Displaying Vector Maps as Patches

Vector map data that is properly formatted can be displayed as a patch, or filled-in polygon. Like many Mapping Toolbox functions, the names of the patch map display functions can be determined by appending an `'m'` to its MATLAB counterpart (e.g., add an `'m'` to `patch` to get `patchm`).

Note: The Mapping Toolbox patch display commands differ from their MATLAB equivalents by allowing you to display patch vector data that use `NAN`s to separate faces.

The following table lists the available Mapping Toolbox patch display functions:

Function	Used to Create
`fillm`	Filled 2-D map polygons
`fill3m`	Filled 3-D map polygons in 3-D space
`patchm`	Patch objects projected on map axes
`patchesm`	Patches projected as individual objects on map axes

The `usalo` MAT-file is useful for illustrating patches:

```
load usalo
who
Your variables are:

conus          gtlakelon      statelat       uslon
greatlakes     state          statelon
gtlakelat      stateborder    uslat
```

The variables `uslat` and `uslon` together describe three polygons (separated by `NaNs`) representing the continental United States. The smaller polygons are necessary to display Long Island and Martha's Vineyard. The variables `gtlakelat` and `gtlakelon` describe three polygons (separated by `NaNs`) for the Great Lakes. The variables `statelat` and `statelon` contain line-segment data (separated by `NaNs`) for the borders between states, which is not formatted for patch display.

Conic projections make good displays of the United States, so create a map axes using an Albers Equal-Area Conic projection. By specifying map limits that contain the region of interest, you automatically center the projection on an appropriate longitude, and the frame encloses just the mapping area, not the entire globe.

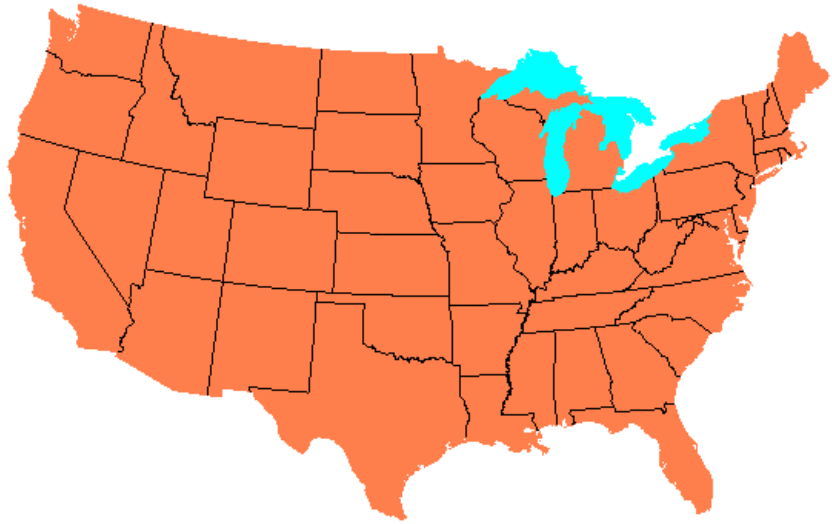
A good rule-of-thumb for conic projections is to set the standard parallels at one-sixth of the distance from both latitude limits. You can do this automatically by providing an empty matrix as the standard parallel:

```
axesm('MapProjection','eqaconic','MapParallels',[],...
      'MapLatLimit',[23 52],'MapLonLimit',[-130 -62])
```

When patch data is displayed, layering can become important as objects above can hide objects below. You can control the visibility of objects by the order of their display and by their "altitude" position. For this reason, the primary patch display function in the Mapping Toolbox, `fillm`, allows you to control the z-axis level of displayed patches. Here the land data is displayed at the default level, z=0. The Great Lakes data is assigned the altitude of z=1. To plot the line segment data between these layers, use the `plot3m` command and specify an altitude of 0.5:

```
fillm(uslat,uslon,'FaceColor',[1 .5 .3],'EdgeColor','none')
fillm(gtlakelat,gtlakelon,1,...
      'FaceColor','cyan','EdgeColor','none')
plot3m(statelat,statelon,0.5,'k')
```

Here is the result:



The `fillm` function makes use of the low-level function `patchm`. The Mapping Toolbox provides another patch drawing function called `patchesm`. The optimal use of either depends on the application and user preferences. The `patchm` function creates one displayed object and returns one handle for a patch, which may contain multiple, not necessarily connecting, faces. The Mapping Toolbox uses `NaN`s to separate non-connected patch faces, unlike MATLAB, which does not handle `NaN` clipped data for patches. For the `patchesm` function, each face is a separate object with its own handle. In general, `patchm` requires more memory but is faster than `patchesm`.

Displaying Matrix Maps

The Mapping Toolbox provides functions for the display and enhancement of both regular and general matrix maps in a variety of formats. Recall that regular matrix maps require a map legend vector that describes the scale and location of the map, while general matrix maps require matrices of latitude and longitude coordinates.

The matrix map display functions are geographic analogies to the MATLAB surface drawing commands. Like the line plotting functions discussed in the previous chapter, the Mapping Toolbox function names are usually identical to their MATLAB counterparts, except for the addition of an 'm.' For instance, the Mapping Toolbox functions, `surfacem` and `surfm`, are analogous to MATLAB's `surface` and `surf` commands. You will most likely find a Mapping Toolbox version of a MATLAB surface drawing function by appending an 'm' to the MATLAB function name.

Note: In the Mapping Toolbox, the set of functions beginning with '`mesh`' are used for regular matrix maps, while those with '`surf`' are reserved for general matrix maps. It should be noted here that this usage differs entirely from MATLAB's definition; that is, `mesh` plots are used for colored, wire-frame views of the surface, while `surf` displays colored, faceted surfaces.

Surface map objects can be displayed in a variety of different ways. You can assign colors from the figure colormap to surfaces according to the value of their data. You can also display images where the matrix data consists of indices into a colormap or display the matrix as a three-dimensional surface, with the z coordinates given by the map matrix. You can use monochrome surfaces that reflect a “pseudo-light” source, thereby producing a three-dimensional, shaded relief model of the surface. Finally, you can use a combination of color and light shading to create a lighted shaded relief map.

The following table lists the available Mapping Toolbox surface map display functions:

Function	Used to Create
`meshm`	Regular matrix map warped to projected graticule mesh
`surfm`	General matrix map projected on map axes
`imagem`	Regular matrix map displayed as image
`pcolorm`	Projected matrix map in z = 0 plane
`surfacem`	Matrix map warped to projected graticule mesh
`surflm`	3-D shaded surface with lighting projected on map axes
`meshlsrm`	3-D lighted shaded relief of regular matrix map
`surflsrm`	3-D lighted shaded relief of general matrix map

The Graticule

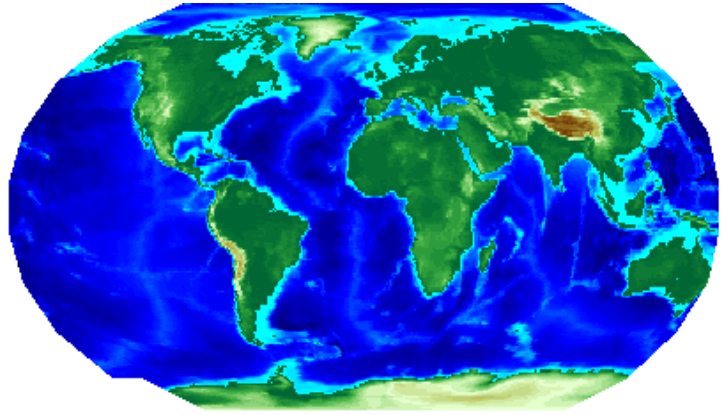
The Mapping Toolbox projects surface objects in a manner very similar to the traditional methods of map making. The cartographer traditionally lays out a grid of meridians and parallels, called the *graticule*. Each graticule cell is therefore a geographic quadrangle. The cartographer calculates the appropriate *x-y* locations for every vertex in the graticule grid and draws the projected graticule by "connecting the dots." Finally, the cartographer draws the map data in "freehand," attempting to account for the shape of the graticule cells. Similarly, the Mapping Toolbox calculates the x-y locations of the four vertices of each graticule cell and warps the matrix data to fit the resulting quadilateral.

In the toolbox, as in traditional cartography, the finer the graticule mesh (i.e., the more meridians and parallels used), the greater the precision of the projected map display, at the cost of greater effort and time. Graticules for regular matrix maps are defined in the toolbox as a two-element vector, of the form [*number-of-parallels*, *number-of-meridians*]. For general matrix maps, the graticule is related to the size of the latitude and longitude coordinate matrices, where *number-of-parallels*=*mrows*-1 and *number-of-meridians*=*ncols*-1.

While the graticule cell for a regular matrix map is restricted to equal-angle quadrangles (i.e., length of cell in latitude must equal length of cell in longitude), general matrix maps have no such constraint. Their graticule cells can be of any size.

The `topo` regular matrix map can be displayed quickly using a coarse graticule. The cost is in precision:

```
load topo
figure; axesm robinson
graticule = [10 20];
h = meshm(topo,topolegend,graticule);
demcmap(topo)
```



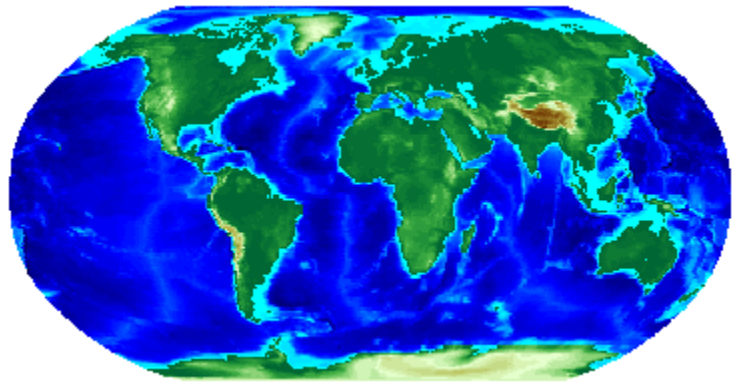
Notice that for this coarse graticule, the edges of the map do not appear as smooth curves. What may not be as obvious is that the easternmost column of graticule cells and the southwesternmost cell are sometimes invisible on displayed matrix maps. This is necessary for the proper projection of the

surface object and is not a concern except with the coarsest graticules. Previous displays used the default [50 100] graticule, for which this effect is negligible.

Regardless of the graticule resolution, the matrix map data is unchanged. In this case, the matrix map is the 180-by-360 `topo` matrix, and regardless of projection fidelity, the resolution of its value data is unchanged.

Map objects displayed as surfaces have all the properties of any MATLAB surface, which can be set at object creation or by using the MATLAB `set` command. The mapping `setm` command allows the `MeshGrat` graticule property to be manipulated for regular matrix surfaces. Since you saved the handle of the last displayed map, reset its graticule to a very fine grid. Remember, the trade-off is between resolution and time, so this could take a while:

```
setm(h,'MeshGrat',[200 400])
```



You'll probably notice that the result does not appear to be any better than the original display with the default [50 100] graticule, but it took much longer to produce. Actually, the 200-by-400 graticule grid is finer than the 180-by-360 data resolution. There is really no reason to ever use a graticule finer than your data. In practice, you will probably find that coarse graticules are good for development tasks and fine graticules serve well for final graphics production.

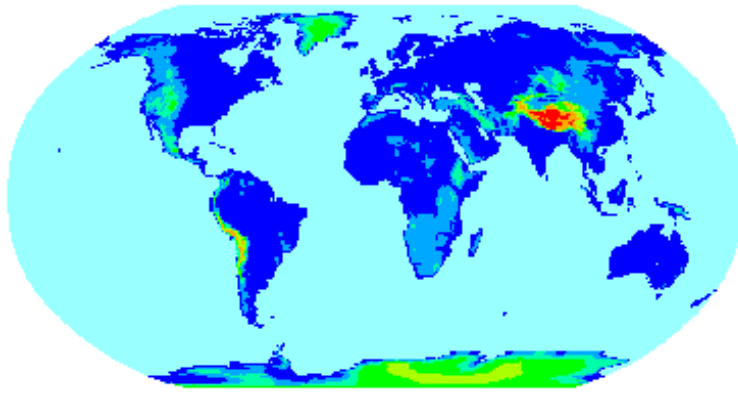
Coloring Matrix Maps

In this tutorial, you may have noticed the use of the function `demcmap` in several digital elevation map (DEM), or topographic, display examples. This function creates colormaps appropriate for the display of DEMs, although it is certainly not limited to just DEMs.

The colormaps, by default, have atlas-like colors varying with elevation or depth that properly preserve the land-sea interface. The demcmap function can also be used to create colormaps very different from the default atlas colors.

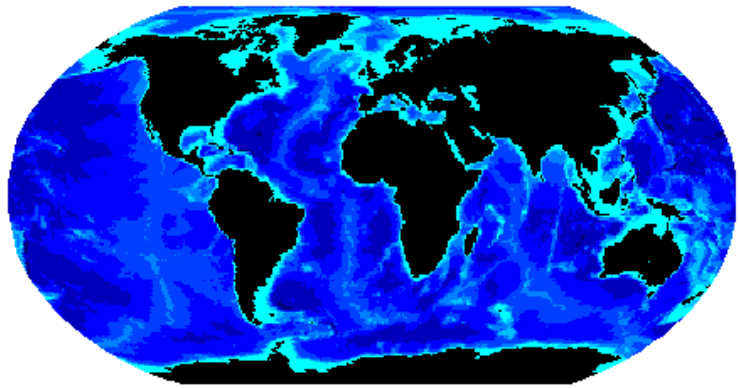
Suppose you want to display the topo map using 16 total colors, coloring the oceans a uniform light-cyan and the land similar to MATLAB's hsv colormap:

```
load topo
landclr = [0 0 1; 0 1 0; 1 0 0];
axesm robinson
meshm(topo,topolegend)
demcmap(topo,16,[.7 1 1],landclr)
```



Instead of specifying the size of the colormap, you can make all values within an elevation band the same color. Modify the colormap to use a surface data interval of 100, coloring all land areas black, and using the default sea colors to take a closer look at the bathymetry data:

```
demcmap('inc',topo,1000,[],[0 0 0])
```



Use the string `'inc'` as the first input argument to indicate that a desired elevation interval or increment is to be used to build the colormap. You can also employ the default colors for the sea by entering an empty matrix in place of an RGB color vector.

Data Representation

Image and Surface Coloring

The simplest way to display a matrix map is to assign colors to matrix elements according to the value of their data. The map can be displayed either as a 2-D image or a 3-D surface using the map values as the z data.

Consider the matrix map located in the korea workspace. It contains a matrix of land elevations and bathymetry data for the region around the Korean peninsula, along with a map legend variable, which indicates a regular matrix map format. Calculate a graticule for the regular matrix map using the meshgrat function so that a general matrix map is also available. Also, convert the units for the map matrix from meters to degrees, so they are consistent with the latitude and longitude coordinate matrices.

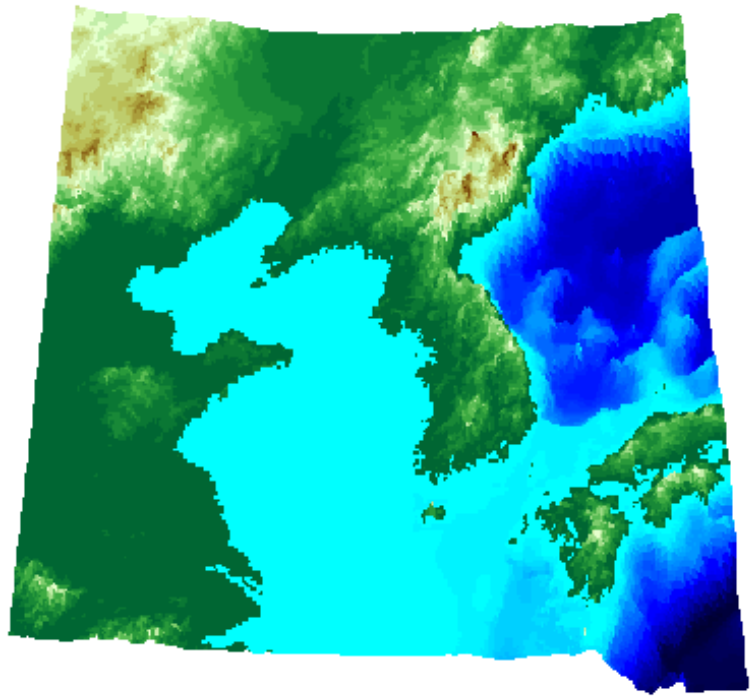
```
load korea
[lat,lon] = meshgrat(map,maplegend);
map = km2deg(map/1000);
whos
   Name            Size      Bytes    Class

   lat          180x240     345600    double array
   lon          180x240     345600    double array
   map          180x240     345600    double array
   maplegend        1x3         24    double array
```

Notice that the lat and lon coordinate matrices, or graticule, are the same size as the map matrix, a requirement for constructing 3-D surfaces.

You have already seen the meshm command used to display regular matrix maps, so display the korea matrix map using the surfm command:

```
axesm('MapProjection','eqaconic','MapParallels',[],...
      'MapLatLimit',[30 45],'MapLonLimit',[115 135])
surfm(lat,lon,map,map); demcmap(map)
set(gca,'DataAspectRatio',[1 1 .5])
view(0,75)
```

Here is the result:

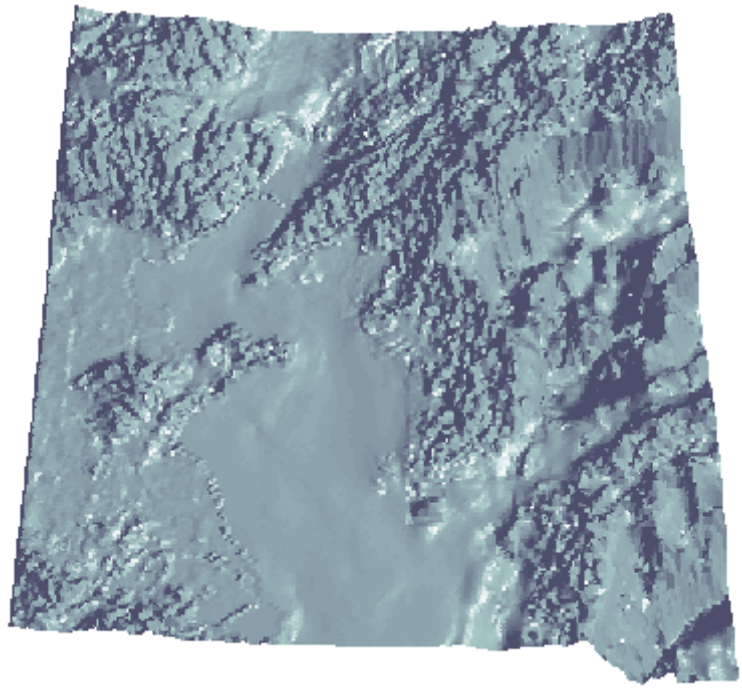


Surface Light Shading

A monochrome three-dimensional, shaded-relief map can be displayed with the function `surflm`, which is analogous to the MATLAB `surfl` command. This functionality is available only for general matrix maps. Remember that regular matrix maps are a subset of general matrix maps and can be easily converted.

To display a map in this format, enter the following:

```
clma
surflm(lat,lon,map); colormap(bone)
```



This is the same region as the previous map, viewed in three dimensions. Notice the dramatic dip in the lower-right portion of the map caused by the deep ocean trenches of the North Pacific Ocean. The view is from the south, at an elevation angle of 75°. The light source in this case is the default 45° counterclockwise from the view direction.

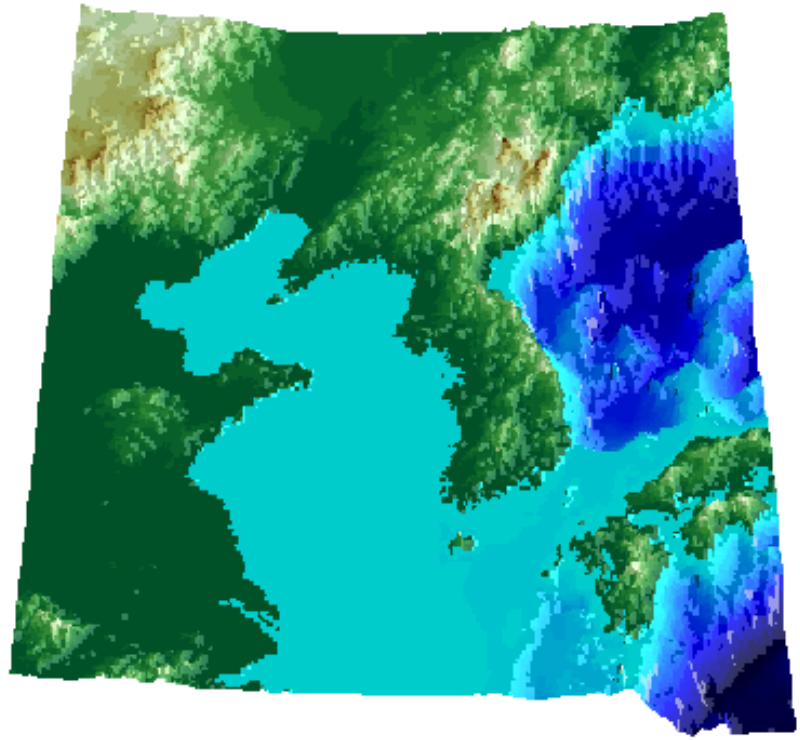
This kind of representation shows much more of the fine structure of the land and sea floor, but because of the lack of color, it is difficult to distinguish land from sea.

Surface Lighted Shaded Relief

The commands `meshlsrm` and `surflsrm` display maps in a lighted shaded relief format and can be thought of as extensions to `surflm` that combine surface coloring and surface light shading. Again, `meshlsrm` is used for regular matrix maps and `surflsrm` for general matrix maps. Although there are no analogous MATLAB functions, you can obtain similar results using MATLAB light objects. However, unlike MATLAB lighted surfaces that do not modify the surface's `CData`, these mapping commands construct a new colormap and associated `CData` matrix by using grayscales to lighten or darken a matrix component based on its calculated surface normal to a light source.

To display a lighted shaded relief map, type the following:

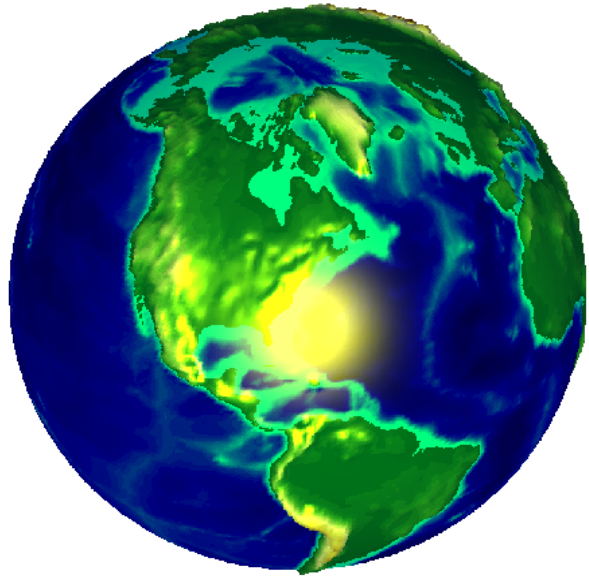
```
clma
surflsrm(lat,lon,map)
```



Mapped Light Objects

The Mapping Toolbox allows for the mapping of light objects using the `lightm` command, which is similar to the `light` command in MATLAB. Here, place a lighted source at an infinite distance, representing the Sun at the summer solstice (23.5°N) at local apparent noon in Natick, Massachusetts (71°W):

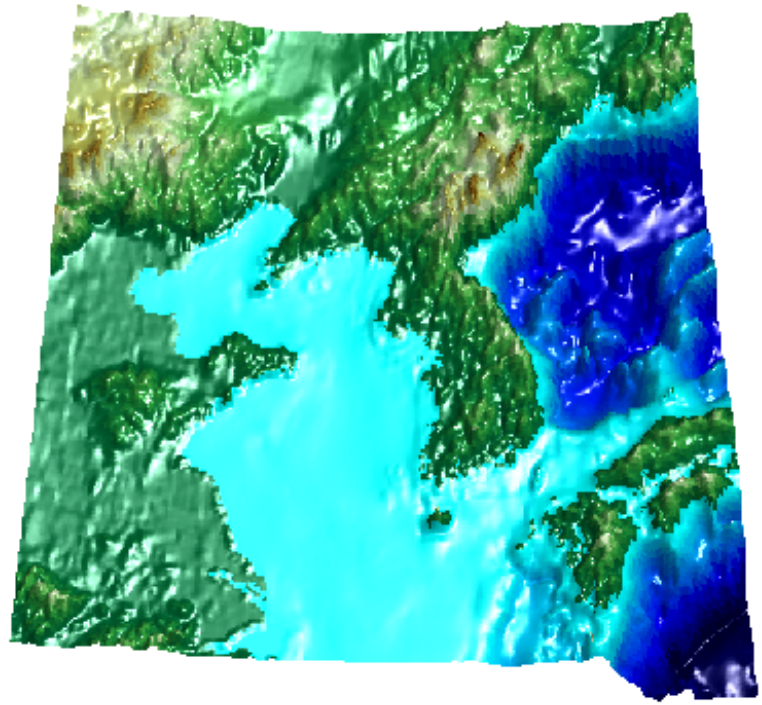
```
load topo
ptopo = topo; ptopo(topo<0) = 0;
ptopo = 100*ptopo/(almanac('earth','radius')*1000);

axesm globe; view([15 38])
meshm(topo,topolegend,size(topo),ptopo); demcmap(topo)
shading interp
lightm(23.5,-71,'Color','y')
material([.5 .7 1.5]); lighting phong
```



The topo surface had been modified to show the 3-D features of land only, which is vertically exaggerated by a factor of 100. The ocean depths are shown by color only.

As a comparison to the lighted shaded relief example shown earlier, add a light source to the surface colored matrix map of the Korean peninsula region shown earlier:

```
figure; load korea
axesm('MapProjection','eqaconic','MapParallels',[],...
      'MapLatLimit',[30 45],'MapLonLimit',[115 135])
meshm(map,maplegend,size(map),map); demcmap(map)
light('Position',[1 0 1])
material([.5 .5 1]); lighting phong
daspectm('meters',125)
view(0,75)
```



The light effects employed here are different and separate from those used by `surfm`, `meshlsrm`, and `surflsrm`. Here, a light object is added directly to the map surface. Like all displayed surfaces, the light object is a separate MATLAB graphic object with its own object handle. For the functions `meshlsrm` and `surflsrm`, light effects are simulated within the functions by modifying the colormap with bands of light and dark. The map matrix is then converted to indices for the new "shaded" colormap based on calculated surface normals.

Using light objects allows for a wide range of different light effects, such as ambient light, light color, and surface reflectance properties, but will ultimately require more memory and time to display.

The `meshlsrm` and `surflsrm` functions provide a simple means of displaying maps with simulated light effects. Both shaded relief formats represent the best available method of displaying matrix maps, since both large (surface coloring) and small (surface shading) features can be examined on one map.

For more information, consult the reference pages for `meshlsrm`, `surflsrm`, and `lightm` in the *Mapping Toolbox Reference Guide*, along with the section on lighting in the MATLAB document, *Using MATLAB Graphics*.

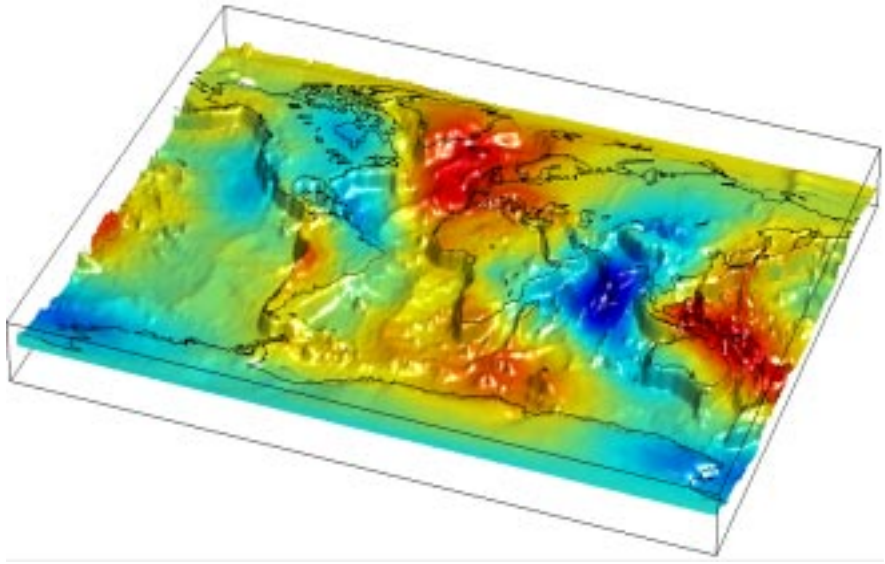
Draping Data on Elevation Maps

Lighting effects are also helpful when elevation maps are combined with other kinds of data. The shading resulting from lighting effects provides the necessary visual cues when you drape satellite data on a grid of elevations. This kind of display is commonly used to combine topography from digital elevation models with land use images from remote sensing satellites such as LANDSAT and SPOT. Such displays can also be created with the Mapping Toolbox.

If the elevation and color matrices correspond pixel-for-pixel to the same geographic locations, such a display can be created using the optional altitude arguments in the surface display commands. The following example shows the Earth's geoid draped on the corresponding topography. The geoid can be described as the surface of the ocean in the absence of waves, tides, or land obstructions. The geoid is influenced by the gravitational attraction of denser or lighter materials in the Earth's crust and by the shape of the crust. The geoid is also important in surveying because it defines the direction of "up," from which is measured the geodetic latitude. The geoid heights vary from a minimum of about 105 meters below sea-level, shown in blue, to a maximum of about 85 meters above sea-level, shown in red. You can see that the geoid mirrors the topography of the major mountain chains such as the Andes, the Himalayas, and the Mid-Atlantic Ridge. You can also see that large areas of high or low geoid heights are not simply a result of topography.

```
load topo
load geoid
axesm gstereo

meshm(geoid,geoidlegend,size(geoid),topo)

plotm(coast,'k')
zdatam(handlem('allline'),1000) % keep lines above surface

daspectm('m',200); tightmap
view(20,35)
camlight; lighting phong

set(gca,'projection','perspective') % allow axes to converge
```

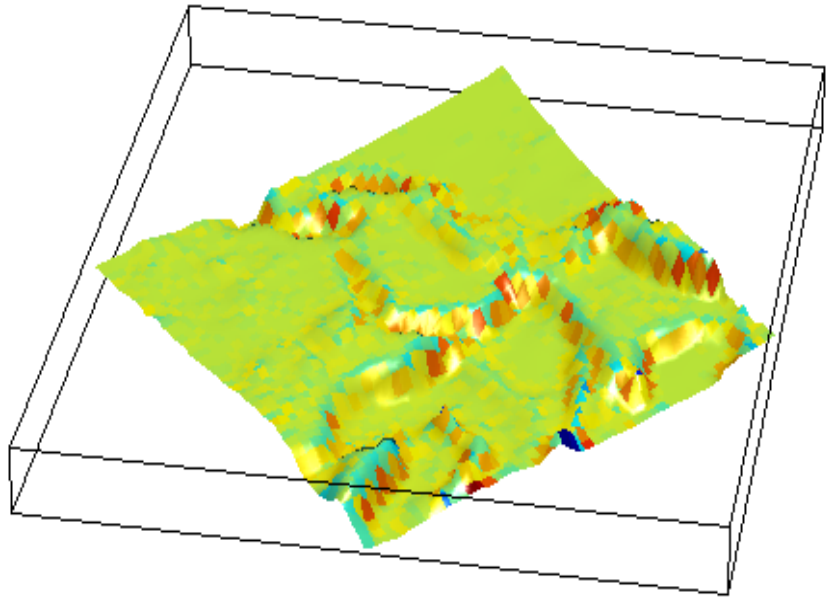
Here is the displayed map:



If your elevation and color data is based on different graticules, you must construct a new version of one matrix that is consistent with the other. Suppose you have elevation data that is in a regular matrix map and color data in a general matrix map. One approach is to interpolate elevation values for the general matrix map graticule. For example, if your elevation map is `topo` and `topolegend`, and the color data is the curvature of the diamond-shaped general matrix map in the `mapmtx` MAT-file, `lt1`, `lg1`, and `del2(map1)`, you can create a diamond-shaped map of elevations using `ltln2val`. When you use `ltln2val` for this purpose, it is important that the regular matrix map of elevations completely covers the area of the general matrix map. The four matrices can then be combined in a surface display. Notice how the greatest curvature values, shown in red, are found at the continental margins of Africa and where the Indian plains meet the Himalayas.

```
load topo
load mapmtx

map1z = ltln2val(topo,topolegend,lt1,lg1);

figure
axesm miller
surfm(lt1,lg1,del2(map1),map1z)
view(10,30); daspectm('m',100)
material shiny;camlight;lighting phong
axis tight; set(gca,'Projection','Perspective')
```

Here is the result:

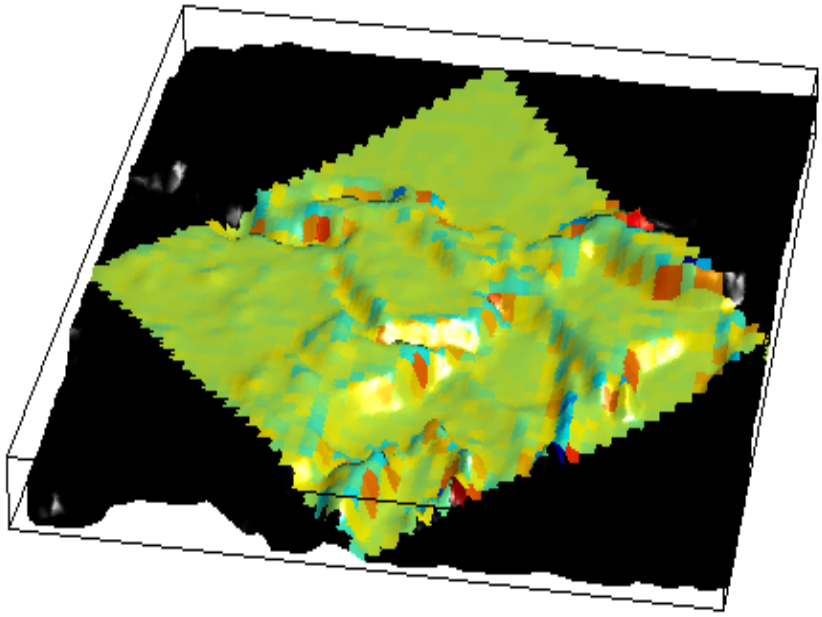


Alternatively, you can construct a regular matrix map version of your general matrix map color data. First, create a new regular matrix map that covers the region of the general matrix map, then imbed the color data values into the new matrix. The new matrix may need to be somewhat lower in resolution than the original, to ensure that every cell in the new map receives a value. The advantage of this approach is that a host of computational functions can be used with the resulting regular matrix map. This example also illustrates the fact that the color and elevation matrices do not have to be the same size. If the resolution of the two is different, you can create the surface as a three-dimensional elevation map and later apply the colors. You do this by setting the surface `Cdata` property to contain the color matrix, and setting the surface face color to '`TextureMap`'.

Try the following:

```
load topo; load mapmtx

latlim = [min(lt1(:)) max(lt1(:))];
lonlim = [min(lg1(:)) max(lg1(:))];

[smalltopo,smalltopoleg] = maptrims(topo,topolegend,...
       latlim,lonlim);
[smalltopocurv,smalltopocurvleg] = nanm(latlim,lonlim,.5);
% 1/.5 cells per degree scale
smalltopocurv = imbedm(lt1,lg1,del2(map1),...
       smalltopocurv,smalltopocurvleg);

figure; axesm miller
h = meshm(smalltopo,smalltopoleg,size(smalltopo),smalltopo);
view(10,30); daspectm('m',100)
material shiny; camlight; lighting phong
axis tight; set(gca,'Projection','Perspective')
set(h,'Cdata',smalltopocurv,'FaceColor','TextureMap')

ltln2val(smalltopocurv,smalltopocurvleg,39,76.3)
ans =
   1.9238e+03
```

Here is the result:



The Geographic Data Structure

In the previous examples, all of the map data was in the form of individual variables and had to be displayed using different mapping commands according to the type of data at hand (i.e., line, patch, matrix, text, etc.). The Mapping Toolbox also provides an easy means of displaying, extracting, and manipulating collections of all types of map objects that have been organized in a specially defined and formatted *geographic data structure*. Note that this structure is different from the *map projection structure*, which defines a map projection along with its mapping properties. The geographic data structure contains the information required for the proper display of the graphic object. The following table lists the six object types along with their required fields of information:

fields	light	line	patch	regular	surface	text
type	•	•	•	•	•	•
tag	•	•	•	•	•	•
lat	•	•	•		•	•
long	•	•	•		•	•
map				•	•	
maplegend				•		
meshgrat				•		
string						•
altitude	•	•	•	•	•	•
otherproperty	•	•	•	•	•	•

Some fields may contain empty entries, but the fields themselves must exist for the object to be displayed correctly. For instance, the `altitude` field can be an empty matrix and `otherproperty` can be an empty cell array. The `tag` field must be a string not equal to the name of the object type (i.e., line, surface, text, etc.). For complete descriptions of these fields and the object types, consult the `geographic data structure` reference page in the *Mapping Toolbox Reference Guide*.

The `usalo` workspace contains several variables in the geographic data structure format:

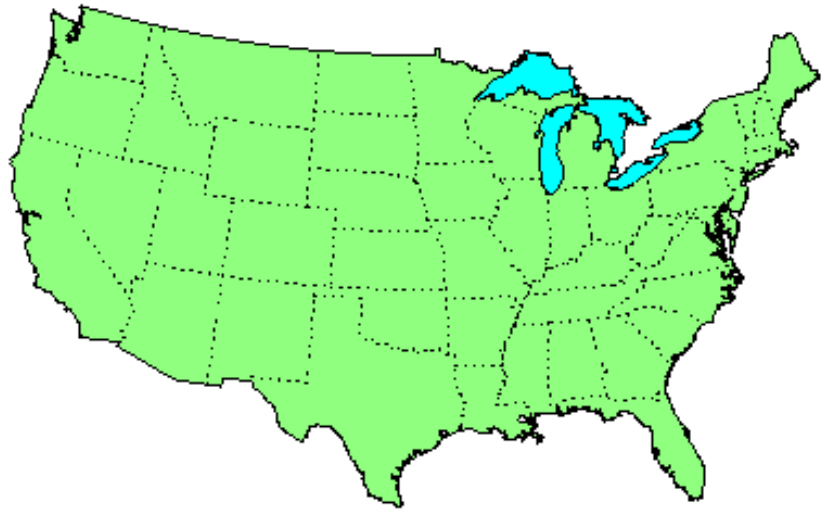
```
load usalo
conus
conus =
              lat: [4392x1  double]
             long: [4392x1  double]
             type: 'patch'
              tag: 'ContinentalUnitedStates'
    otherproperty: []
         altitude: []

stateborder
stateborder =
              lat: [2345x1 double]
             long: [2345x1 double]
             type: 'line'
              tag: 'StateBorder'
    otherproperty: {   1x1 cell  }
         altitude: []

greatlakes
greatlakes =
1x3 struct array with fields:
    type
    tag
    lat
    long
    altitude
    otherproperty
```

The structure conus contains a patch object of the continental U.S.; greatlakes consists of three patches representing the Great Lakes, and the structure stateborder contains line vector data for the state boundaries.

These can be displayed with the displaym function:

```
axesm('MapProjection','lambert','MapParallels',[],...
      'MapLatLimit',[23 52],'MapLonLimit',[-130 -62])
displaym(conus)
displaym(greatlakes)
displaym(stateborder)
```

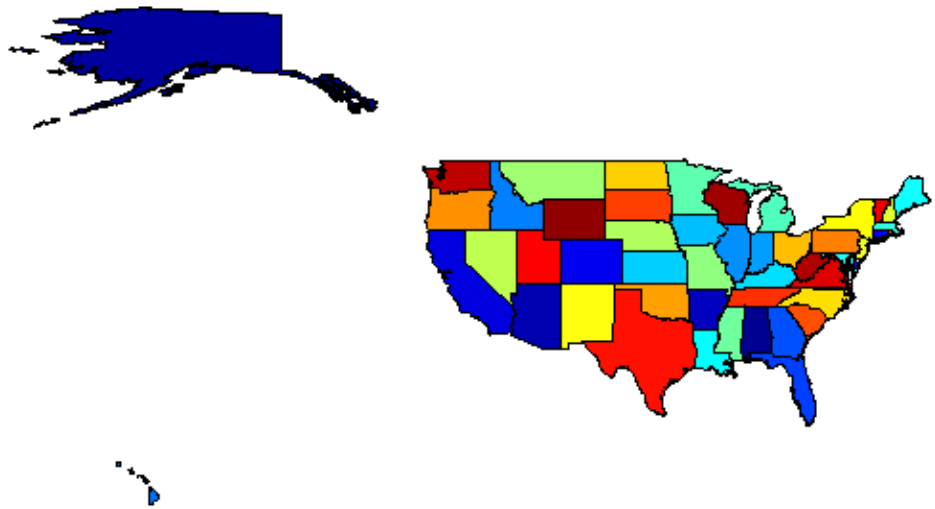
The map shown here is virtually identical to that of a previous example in which the vector variables uslat, uslon, gtlakelat, gtlakelon, statelat, and statelon were used:



Now display all 50 states and the District of Columbia using the patches located in the geographic data structure variable state:

```
figure
axesm trystan
displaym(state)
```

Here is the map in a Trystan Edwards Cylindrical projection:



To view individual states that reside in the `state` structure, you can either use the `displaym` function:

```
displaym(state,'mass');
```

Or you can extract and display the data from the formatted structure yourself:

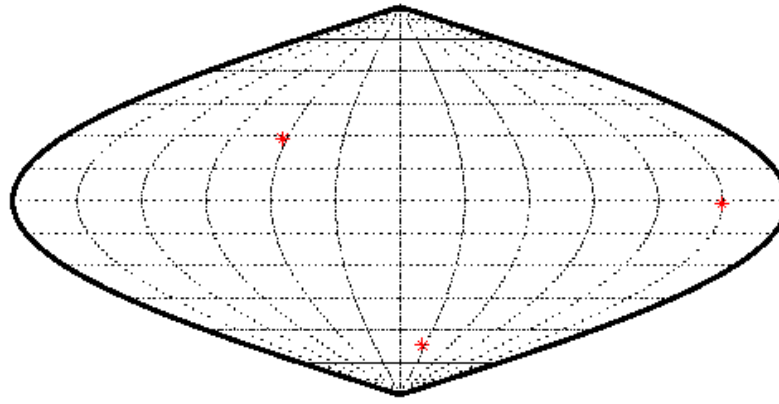
```
[lat,long] = extractm(state,'mass');
plotm(lat,long)
```

Note: Most of the Mapping Toolbox atlas data and data from public sources processed by Mapping Toolbox external interface functions enters the MATLAB workspace as geographic data structures. You can also format your own data as geographic data structures.

Interacting with Displayed Maps

The Mapping Toolbox provides the means to interact with displayed maps. The `inputm` command allows you to get the latitude-longitude position of a mouse-click; the `gcpmap` function returns the current mouse position, again in latitude and longitude. As an example, display a map axes with its grid. The lines of code below allow you to click on the map three times and store the geographic coordinates for the three points selected in `lat` and `long`. The final `plotm` command will plot the points you clicked on. The display shown here corresponds to the points chosen for the example:

```
axesm sinusoid
framem on; gridm on
points=inputm(3)
points =
      28.9373  -62.1631
      -1.1575  149.3467
      -67.1344   26.8096

plotm(points,'r*')
```



Your points will probably be different. You can also use special interactive commands to interactively place text, small circles, and tracks. For more information on these capabilities, consult the `gtextm`, `scircleg`, and `trackg` entries in the *Mapping Toolbox Reference Guide*, along with Chapter 6, "GUI Tools" of this manual.

The Mapping Toolbox also allows you to manipulate displayed objects by name. Many mapping commands assign descriptive names to the `Tag` property of the objects they create. The name functions allow you to control the display of groups of similarly named objects, determine the names and change them if so desired, and use the name in the Handle Graphics `set` and `get` commands. There is also a Mapping Toolbox graphical user interface, `mobjects`, to help you manage the display and control of objects.

Some mapping display functions like `framem`, `gridm`, and `contorm`, assign object tags by default. You can also set the name upon display by assigning a string to the `Tag` property in mapping display functions that allow property-value pairs. If the `Tag` does not contain a string, the name falls back to an object's `Type` property, such as `'line'` or `'text'`.

Display a vector map of the world:

```
figure; axesm fournier
framem on; gridm on;
plabel on; mlabel('MLabelParallel',0)

load coast
plotm(lat,long,'k','Tag','Coastline')
```

You can view the names of the objects in the current axes using `namem`:

```
namem
ans =

Coastline
PLabel
MLabel
Meridian
Parallel
Frame
```

Change the line width of the coastline:

```
set(handlem('Coastline'),'LineWidth',2)
```

Use a graphical user interface to pick objects and change their colors. Be sure to use the right property name. Patches, for example, have a `'FaceColor'` and `'EdgeColor'`, while most other objects simply have `'Color'`.

```
set(handlem,'Color',uisetcolor)
```

Change the color of any objects with tags containing the substring `'line'`.

```
set(handlem('line',gca,'findstr'),'Color','r')
```

Hide, then show all of the objects of type `'text'`, regardless of their tags:

```
hidem('alltext')
showm('alltext')
```

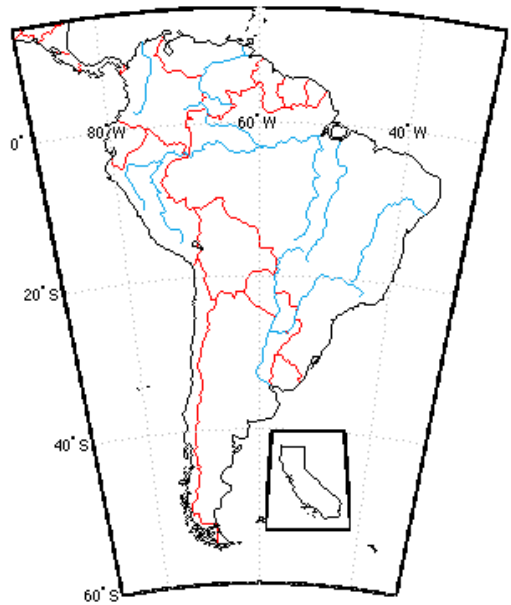
For more information on the use of these functions and tools, consult the *Mapping Toolbox Reference Guide* under the headings for `handlem`, `hidem`, `showm`, `clmo`, `namem`, `tagm`, and `mobjects`.

Specialized Map Functions

Inset Maps

Inset maps are often used to display widely separated areas, generally at the same scale. You can create inset maps by adding multiple axes to a figure and defining appropriate map projections for each. If you want to ensure that the scale of each of the maps is the same, use `axesscale` to resize them. Here is an example of an inset map of California at the same scale as the map of South America, graphically relating the size of the continent to what may be a more familiar region.

```
h1 = worldmap('south america');
h2 = axes('pos',[.5 .2 .1 .1]);
usamap('californiaonly','lineonly')
setm(h2,'FFaceColor','w')
mlabel; plabel; gridm  % toggle off
axes(h2)
axesscale(h1)
hidem([h1 h2])
```

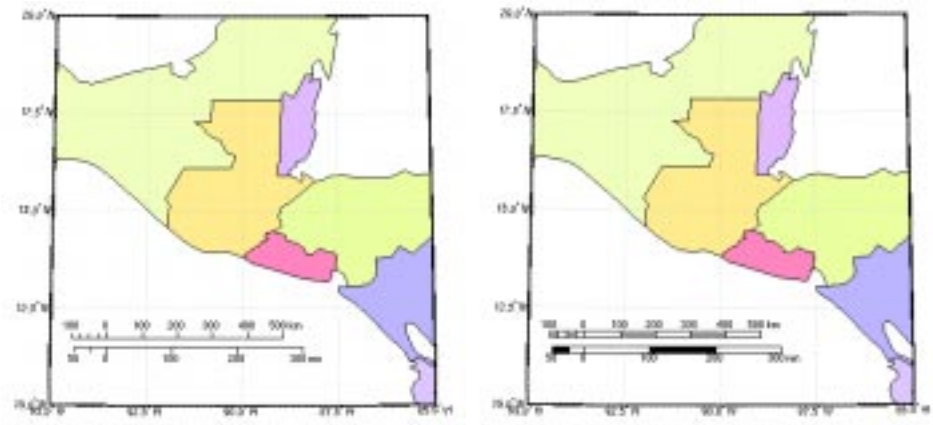


Graphic Scale

Another element used to provide an indication of size is the graphic scale. This is a ruler-like object that shows distances on the ground at the correct size for the projection. You can use the `scaleruler` function to add a graphic scale to the current map. The `scaleruler` settings can be checked or modified using `getm` and `setm`. You can also move the graphic scale to a new position by dragging its baseline.

As an example, create a map, add a graphic scale with the default settings, and shift it up a bit. Add a second scale showing nautical miles, and change the tick marks and direction. Show the same map with different ruler styles.

```
worldmap('guatemala','patchonly')

scaleruler
setm(handlem('scaleruler1'),'YLoc',.205)

scaleruler('units','nm')
setm(handlem('scaleruler2'),'MajorTick',0:100:300,...
     'MinorTick',0:25:50,'TickDir','down',...
     'MajorTickLength',km2nm(20),...
     'MinorTickLength',km2nm(12.5))

setm(handlem('scaleruler1'),'RulerStyle','lines')
setm(handlem('scaleruler2'),'RulerStyle','patches')
```



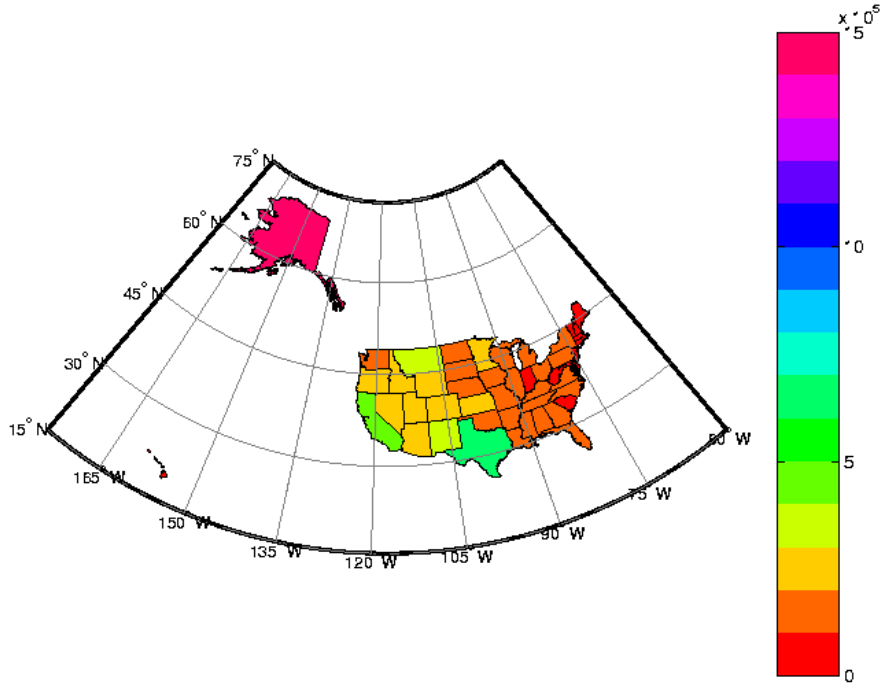
Choropleth Maps

A common type of map display has patch faces colored proportional to some specified data. The data generally represents the value of something per unit area, such as population density, electoral votes, or incidence of disease. Maps of this type are generally called *choropleth* maps. It is easy to construct choropleth maps with MATLAB and the Mapping Toolbox. Simply assign the data values to the `CData` property of the displayed patches.

In the following example, patches representing the 50 states of the U.S. (and the District of Columbia) are displayed and colored according to the surface areas calculated by the `areaint` function. An equal-area projection is appropriate for this kind of display.

```
load usalo
axesm('MapProjection','eqaconic','MapParallels',[],...
      'MapLatLimit',[15 75],'MapLonLimit',[-175 -60],...
      'MLineLocation',15,'MLabelParallel','south',...
      'MeridianLabel','on','ParallelLabel','on',...
      'GLineStyle','-','GColor',0.5*[1 1 1],...
      'Grid','on','Frame','on')
displaym(state)

geoid = almanac('earth','geoid');
tags = {state.tag};
for i=1:length(state)
      lat = state(i).lat; long = state(i).long;
      surfarea = sum(areaint(lat,long,geoid));
      set(handlem(tags{i}),'CData',surfarea);
end

caxis([0 1.5E6])
colormap(hsv(15))
colorbar
```

The coloring scheme of the map makes it easy to distinguish which states are comparable in area. The color scale to the right of the map shows the surface areas in square kilometers:

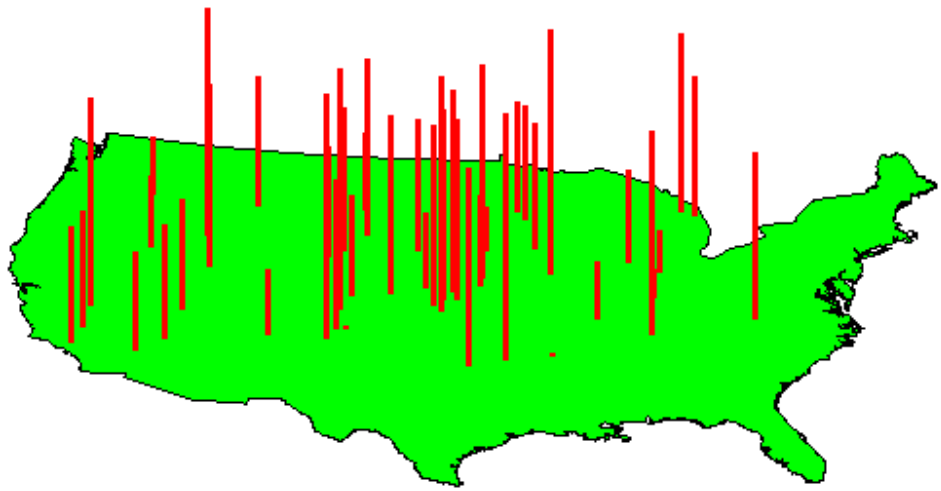


The largest state is Alaska (about 1.5 million square kilometers), while Rhode Island stands as the smallest state (about 2500 square km). The District of Columbia, of course, is the smallest entity of the dataset, at roughly 200 square km.

Thematic Map Functions

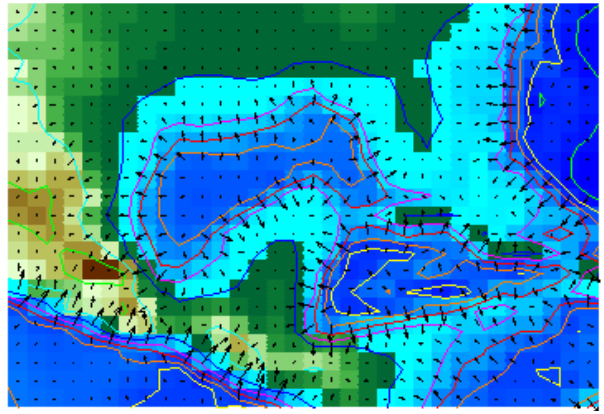
The Mapping Toolbox provides a wide variety of other display and symbology functions. The `cometm` and `quiverm` functions operate like their MATLAB counterparts, `comet` and `quiver`. The `stem3m` command allows you to display geographic bar graphs. Like the MATLAB `scatter` command, the `scatterm` function allows you to display a thematic map with proportionally sized symbols. The `tissot` function calculates and displays Tissot Indicatrices. For more information on these capabilities, consult the descriptions of these functions in the "Mapping Reference" section of the *Mapping Toolbox Reference Guide*.

Here is an example of a stem plot over a map of the continental United States. The bars could represent anything from selected city populations to the number of units of a product purchased at each location:

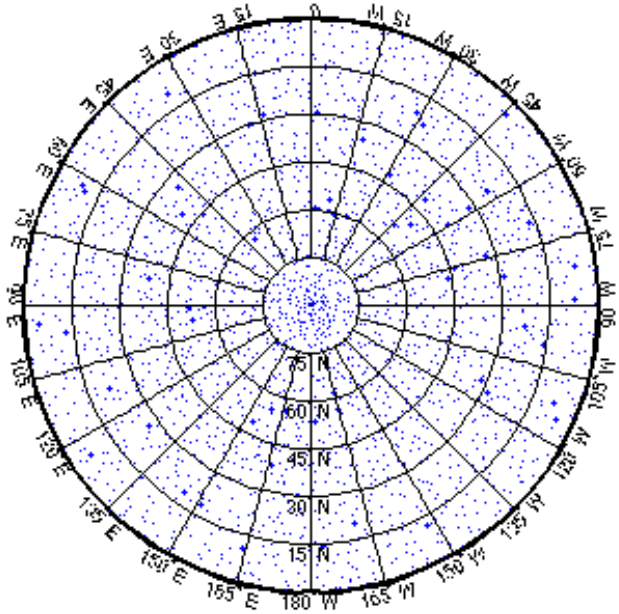


Contour and quiver plots can be extremely useful in analyzing matrix data. In the following example, contour elevation lines have been drawn over a topographical map. The region displayed is the Gulf of Mexico, obtained from the `topo` matrix. Quiver plots have been added to visualize the gradient of the topographical matrix.

Here is the displayed map:



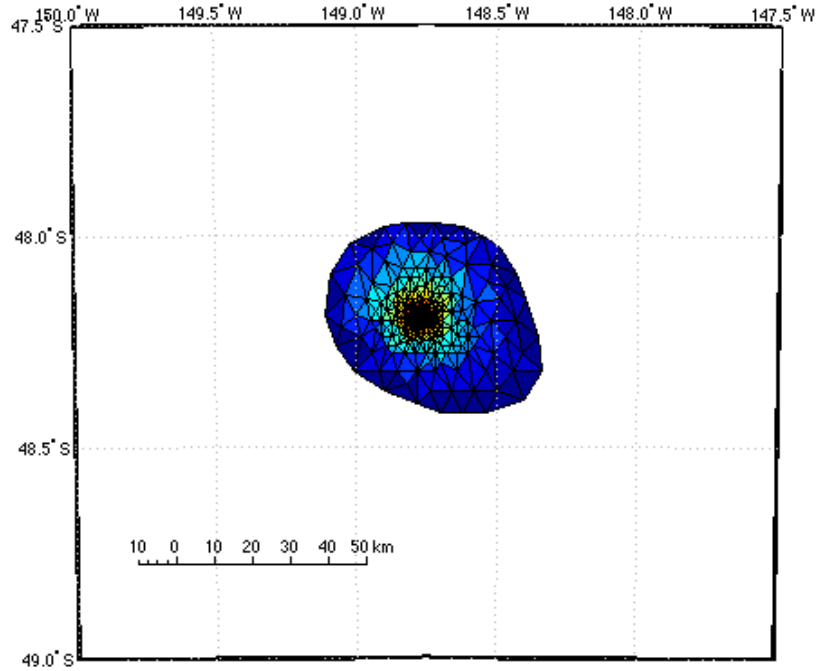
Here is an example of using the `scatterm` function to create a star chart of the northern sky. The stars are represented by filled circles whose size is proportional to visual magnitude. Star gazers may recognize the Big Dipper constellation, or Ursa Major, located at 180° W, 60° N:



Cartesian MATLAB Display Functions

If you cannot find a Mapping Toolbox display function that does what you need, you may be able to use a non-mapping MATLAB function. You can use the MATLAB function to add the graphic objects to the display, using latitude and longitude as x and y, and then project the data afterwards. For example, the Mapping Toolbox does not have a function that displays a triangulated surface from random data, but MATLAB does. Create such a map of the seamount data provided with MATLAB.

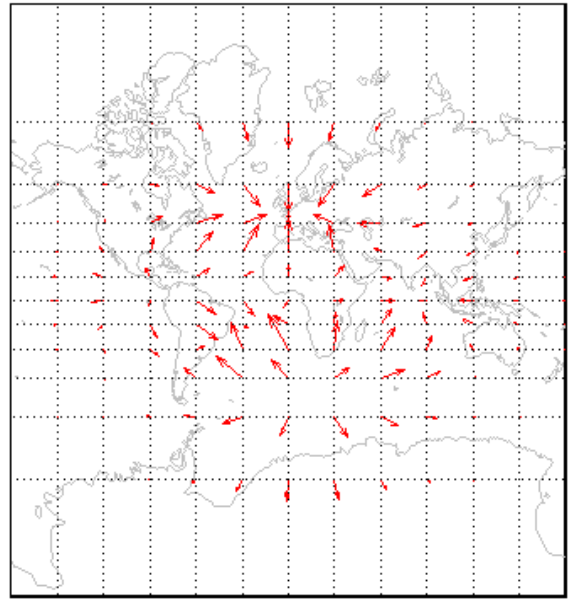
```
load seamount
worldmap([-49 -47.5],[-150 -147.5],'none')

tri = delaunay(y,x);
h = trisurf(tri,y,x,z);
project(h,'yx')

scaleruler
```



If the displayed objects are already in the right place and do not need to be projected, you can trim them to the map frame and convert them to mapped objects using `trimcart` and `makemapped`. They can then be manipulated as if they had been created with map display functions. This is useful when you want to get back the geographic coordinates.

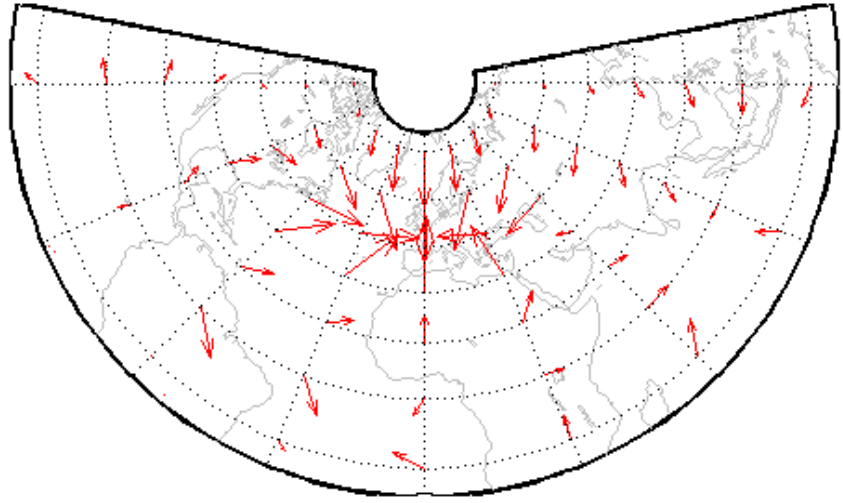
You can also combine Mapping Toolbox and MATLAB commands to mix spherical and Cartesian coordinates. An example would be a quiver plot in which the locations of the vectors are geographic, but the lengths are not. In that case, you can use Mapping Toolbox projection calculations and MATLAB graphics commands. Cylindrical projections are the simplest. North is up and east is right, so just project the locations.

```
figure; axesm mercator; framem; gridm
plotm(coast,'color',[.75 .75 .75])

[u,v] = gradient(peaks(13)/10);
[lat,lon] = meshgrat(-90:15:90,-180:30:180);
[x,y] = mfwdtran(lat,lon);

h = quiver(x,y,u,v,.2,'r');
trimcart(h)
```



An extra step may be required for non-cylindrical projections. In these projections, the directions of north and east vary with location. To make the directions agree with the map grid, vectors should be rotated to bring them into alignment. This can be done with the vector transformation function `vfwdtran`. Consider the same data displayed on a conic projection.

```
figure
axesm('lambert','MapLatLimit',[-20 80])
framem; gridm
plotm(coast,'color',[.75 .75 .75])

[x,y] = mfwdtran(lat,lon);
thproj = deg2rad(vfwdtran(lat,lon,90*ones(size(lat))));
[th,r] = cart2pol(u,v);
[uproj,vproj] = pol2cart(th+thproj,r);

h = quiver(x,y,uproj,vproj,0,'r') ;
trimcart(h)
```



Conformal projections are often the best choice for displays like this. They preserve angles, ensuring that the difference between north and east will always be 90 degrees in projected coordinates.

Printing Maps

Maps are often printed at a size that makes objects on paper a particular fraction of their real size. The ratio of the mapped and real object sizes is called scale, and it is usually notated with a colon as “1:1,000,000” or “1:24,000”. Another way of specifying scale is to call out the printed and real lengths, for example “1 inch = 1 mile”.

You can specify the printed scale using the `paperscale` function. It modifies the size of the printed area on the page to match the scale. If the resulting dimensions are larger than your paper, you can reduce the amount of empty space around the map using `tightmap`, `zoom`, or `panzoom`, and by changing the axes position to fill the figure. This also reduces the amount of memory needed to print with the `zbuffer` (raster image) renderer. Be sure to set the paper scale last. For example:

```
set(gca,'Units','Normalized','Position',[0 0 1 1])
tightmap
paperscale(1,'in', 5,'miles')
```

You can also check the size and extent of text and the relative position of axes using `previewmap`. This function resizes the figure to the printed size. For more information on printing see the “Printing MATLAB Graphics” section of *Using MATLAB Graphics*.

3

Atlas Data

Types of Data

The Mapping Toolbox includes a number of datasets for global and regional displays. Map data is available in both vector and matrix format, covering the world and the United States. The Mapping Toolbox also provides astronomical data for simple stellar cartography.

More detailed data is available over the Internet or on CD-ROM and can be imported to MATLAB using the data interface functions described in Chapter 7, “External Data Interface”.

World Vector Data

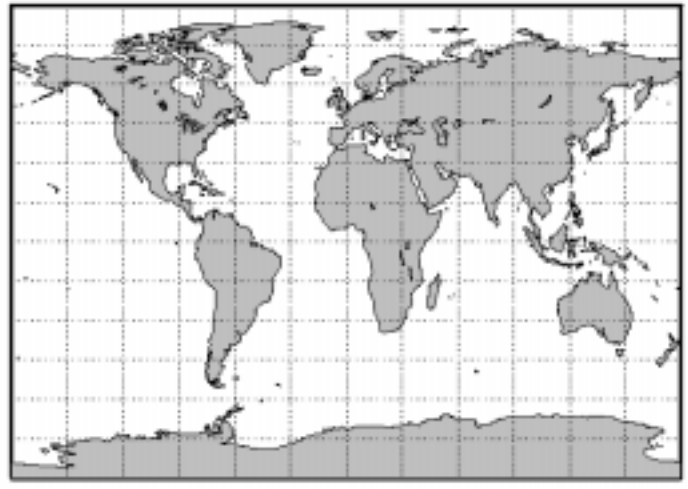
Coastlines

The coast MAT-file contains a set of vector shorelines intended for global displays in which political boundaries are not required. The data is stored in vectors of latitude and longitude, consisting of nearly 10,000 points. While this a considerable quantity of data, it is of very low resolution by cartographic standards.

```
load coast
whos
   Name            Size      Bytes    Class

   lat           9589x1      76712    double array
   long          9589x1      76712    double array
```

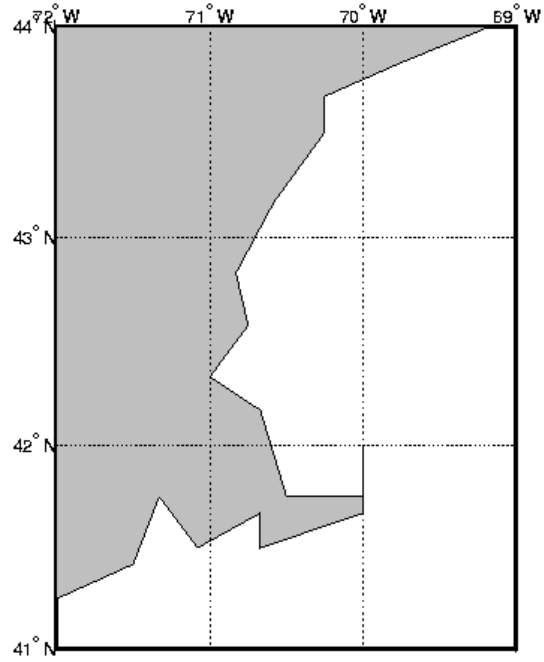
The dataset is in patch format, meaning it has polygon segments that return to their starting points and can therefore be usefully displayed using patch commands. The polygon segments are separated by `NaN`s into about 240 faces. The data can also be displayed as lines or points.

```
axesm giso; framem; gridm
patchesm(lat,long,'FaceColor',0.75*[1 1 1])
```



The resolution for this dataset makes it ideal for global-scale displays but inappropriate for significantly smaller regions, as can be seen when the region around Cape Cod in the northeastern United States is displayed. This is, incidentally, the area covered by the cape MAT-file provided with MATLAB.

```
figure
axesm('MapProjection','mercator',...
  'MapLatLimit',[41 44],'MapLonLimit',[-72 -69])
framem
gridm('MLineLocation',1,'PLineLocation',1);
mlabel('MLabelLocation',1)
plabel('PLabelLocation',1)
patchesm(lat,long,'FaceColor',.75*[1 1 1])
```



World Atlas Data

The `worldlo` global atlas data contains a set of very small-scale (approximately 1:30,000,000) data for use in global displays. The data is provided in the form of Mapping Toolbox geographic data structures containing national boundaries, rivers, lakes, and cities. Here is a list of the structures in the `worldlo` workspace:

```
load worldlo
whos
   Name             Size      Bytes    Class

   DNline            1x2     112546    struct array
   DNpatch          1x39      59884    struct array
   POline            1x2     456600    struct array
   POpatch         1x206     682454    struct array
   POtext          1x194     188132    struct array
   PPpoint         1x319     400426    struct array
   PPtext          1x318     296988    struct array
   description      4x75        600    char array
   gazette         1x513     353504    struct array
```

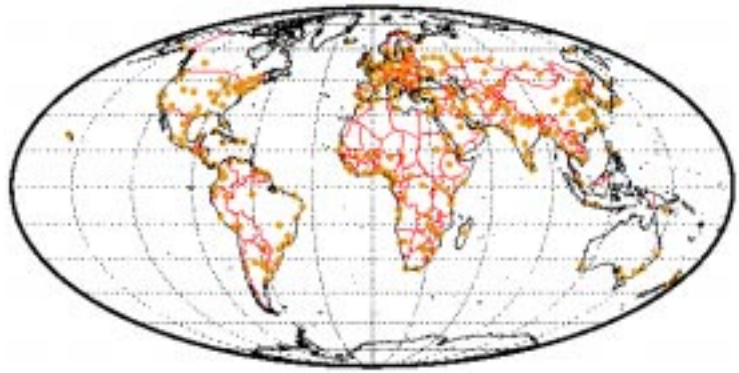
The data has been classified into Drainage (DN), Political/Ocean boundaries (PO), and Populated Places (PP). The data is further separated into structures containing patches, lines, points, or text.

In addition to containing more kinds of data than `coast`, this dataset is also more detailed. The drainage data consists of about 7,000 points, while the political lines and patches each have about 30,000 points. There are more than 200 political units in the political data and more than 300 major cities in populated places. Because of the quantity of data, complicated displays may require more than MATLAB's default memory size and may take longer to project and render. Displays of patches require the greatest memory and computing time.

The contents of the structures can be displayed from the command line using `displaym`. For example, type the following:

```
axesm mollweid; framem; gridm
displaym(POline)
displaym(PPpoint)
```

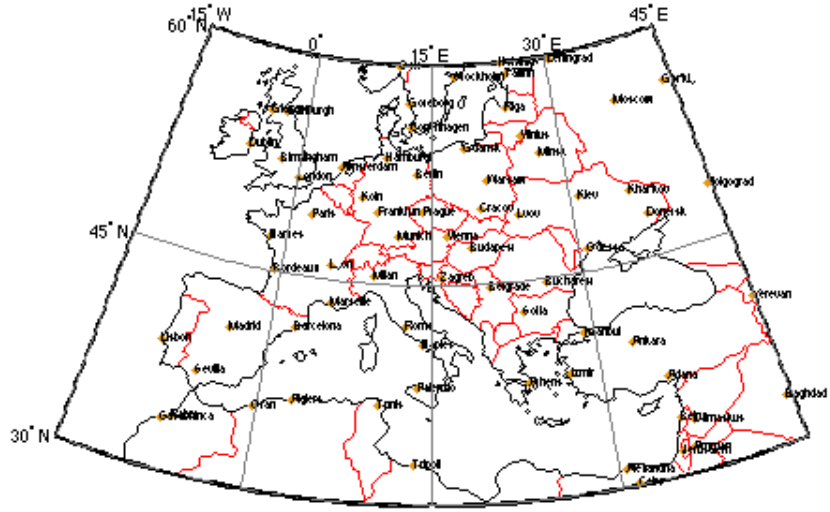
Here is the result:



The map displays the coastlines, international borders, and major cities of the world. We could also label the cities with data from the `PPtext` structure, but at this scale, the result would be unreadable. When we restrict ourselves to a smaller geographic region, more information can be shown:

```
figure
axesm('Mapprojection','eqaconic','MapParallels',[],...
  'MapLatLimit',[30 60],'MapLonLimit',[-15 45],...
  'MLabelLocation',15,'MLineLocation',15,...
  'PLabelLocation',15,'PLineLocation',15,...
  'GColor',.5*[1 1 1],'GLineStyle','-')
framem; gridm; mlabel; plabel
displaym(POline)
displaym(PPpoint)
h = displaym(PPtext); trimcart(h)
```

The zoomed-in region of Europe allows the names of the major cities to be read more easily:

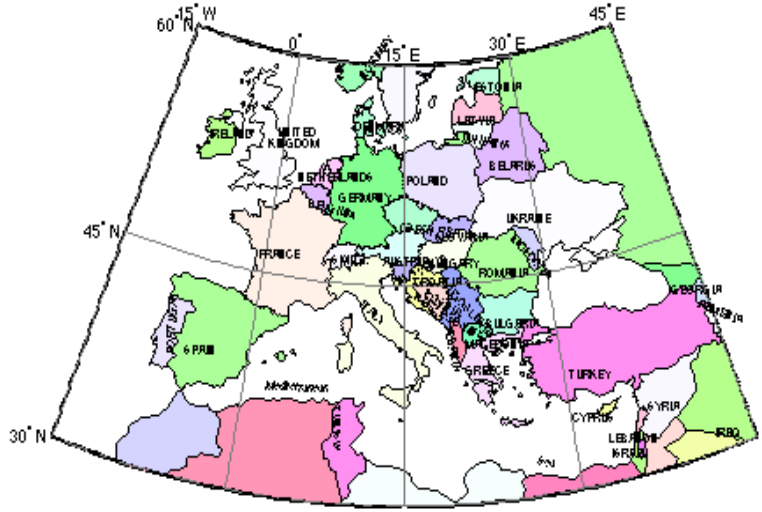


Note: Unlike vector and matrix data, text objects are not automatically trimmed when they fall outside the map frame limits. You can remove them using the `trimcart` function or some other method (e.g., click-on-text and `deletem(gco)`, or use click-and-drag tools).

There are other ways to represent political data. We can display the countries as patches, label them with strings from the `POtext` structure, and pick random colors for the patch faces. Clear the previous line map, and redraw the patch map:

```
clma
displaym(POpatch); polcmap
h = displaym(POtext); trimcart(h); zdatam(h,1)
```

Here is the result:

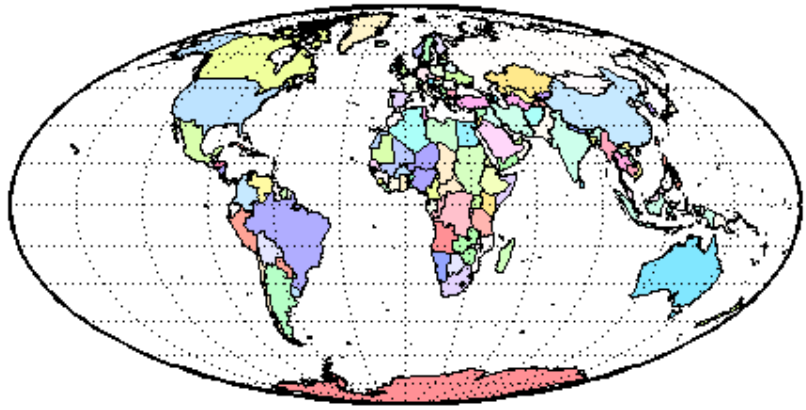


Note how the separated parts of countries are filled with the same colors. Sardinia and Sicily, for example, are shaded the same color as the boot of Italy. This is accomplished automatically because each of the components is tagged as “Italy.”

Every element in a `worldlo` structure has a tag field. Drainage data has been tagged to distinguish between rivers and lakes. The political/ocean lines are tagged as coastlines or international boundaries, and the political patches have been tagged with the names of the country. Both of the functions `displaym` and `mlayers` use the same colors for similarly tagged objects if no `otherproperty` is specified in the geographic data structure.

In the following example, it is evident that the many islands of Indonesia and Canada are similarly colored, as are all parts of the United States:

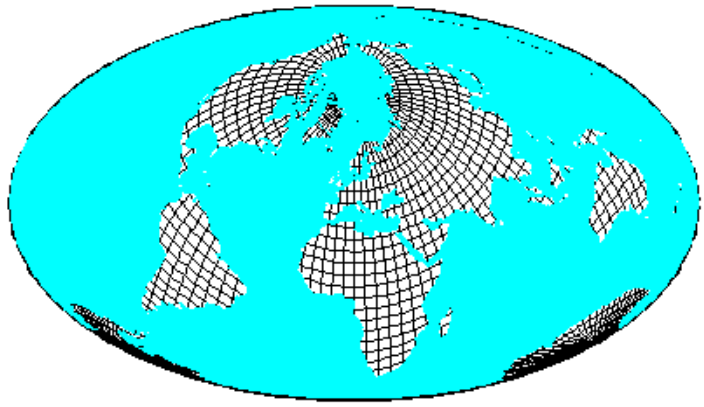
```
figure
axesm mollweid; framem; gridm
displaym(POpatch)
```



There is also a corresponding set of patches for the oceans. These can be used to *blank out* objects in ocean areas. The data is stored separately in the `oceanlo.mat` file, but it can be accessed using the `worldlo` atlas function. The patches are in large tiles with few points along the edges, which may result in

visible seams for non-cylindrical projections. `interpm` can be used to fill in points at a sufficiently fine spacing.

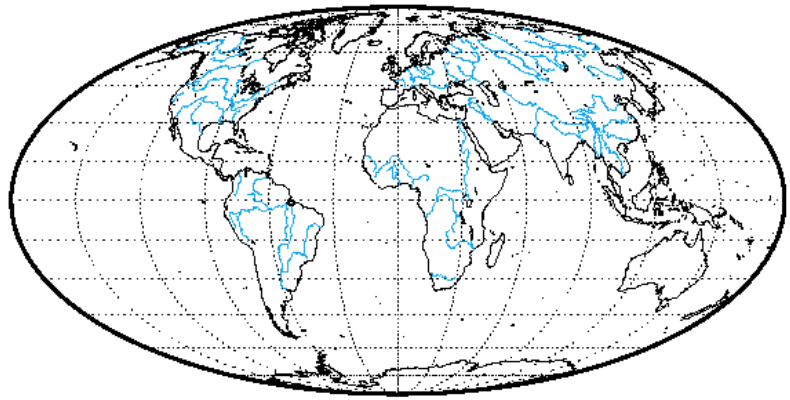
```
[lat,lon] = extractm(worldlo('oceanmask'));
[lat,lon] = interpm(lat,lon,5);

axesm('bries','GAlt',-1,'MLineLocation',5,'PLineLocation',5,...
      'GLineStyle','-')
framem; gridm
patchesm(lat,lon,'c','EdgeColor','none')
```



You can use the tags to reduce the amount of data in the display by selectively deleting data. This can be done by displaying all of the data in a structure and then using the tags to get the handles of the displayed objects you wish to remove. The first European example made use of this technique, as does the next one. It shows rivers, lakes, and coastlines. International boundaries have been removed using their tags:

```
clma
axesm mollweid; framem; gridm
displaym(POline)
delete(handlem('International Boundary'))
displaym(DNline)
```

Here is the map of world coastlines and drainage:



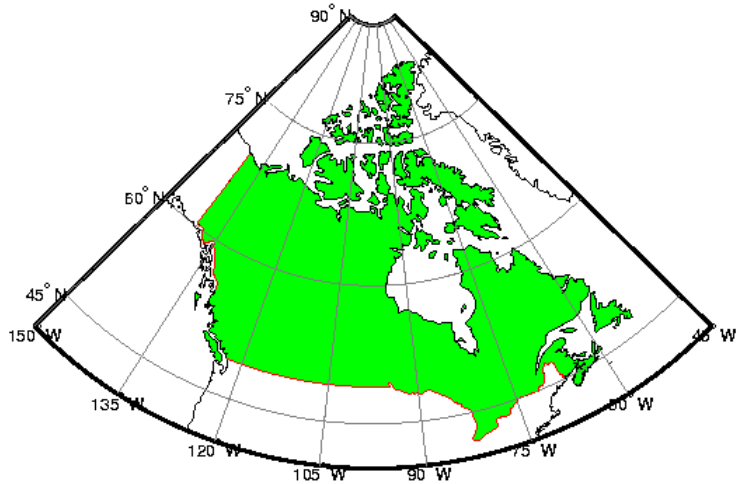
There are a number of other ways to work with the tags associated with this data. We can view the tags in the workspace by working directly with the geographic data structure.

```
unique(strvcat(DNline.tag),'rows')
ans =
Inland shorelines
Streams,rivers,channelized rivers
```

Or we can use `mlayers` and `mobjects` to plot the structures, view the tags, and manipulate similarly tagged objects as a group. See Chapter 6, “GUI Tools” for some examples.

The tags can also be used to extract data from the structures. The following commands extract the patch data for Canada and plot it with the political line data:

```
figure
axesm('MapProjection','eqaconic','MapParallels',[],...
  'MapLatLimit',[40 90],'MapLonLimit',[-150 -45],...
  'MLabelLocation',15,'MLineLocation',15,...
  'PLabelLocation',15,'PLineLocation',15,...
  'GColor',.5*[1 1 1],'GLinestyle','-',...
  'MLabelParallel','south')
framem; gridm; mlabel; plabel
displaym(POline)
[clat,clon] = extractm(POpatch,'canada');
patchesm(clat,clon,'g')
```



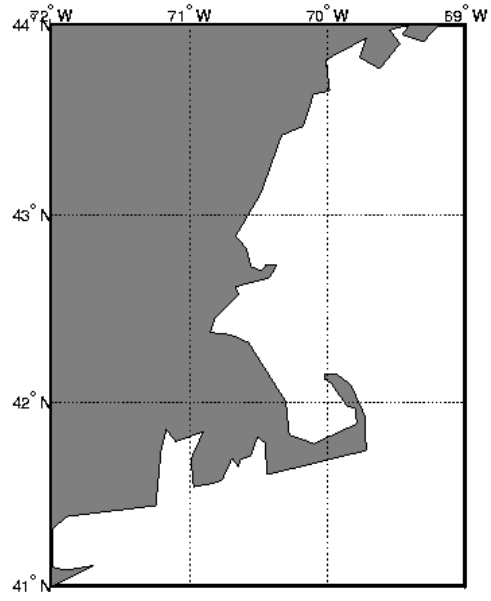
Note: Extracting and displaying only the desired data can reduce the time and memory required to display a map.

The Mapping Toolbox automatically trims and saves data outside the current map latitude and longitude limits. If you use `displaym` to show a small part of the world, the rest of the world data is retained within the map axes structure.

Every time the projection parameters are changed, all of the world data is restored, trimmed, and projected. This overhead can be significant for a dataset as large as `worldlo`.

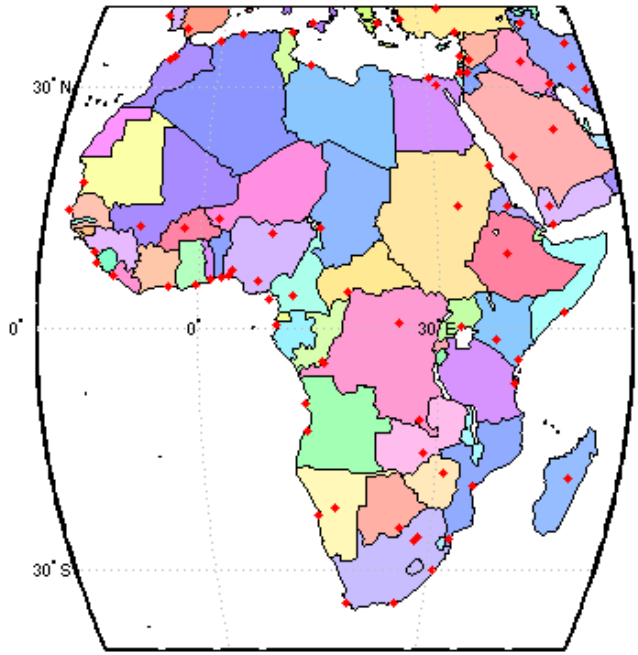
Apply the extraction technique to generate a display of Cape Cod. As was shown in the earlier examples, this data contains somewhat generalized shapes for the countries and rivers. The level of detail is great enough for global and regional maps, but the distinct data points become noticeable when displayed for smaller regions. This data should be considered accurate to no better than 10 or 20 kilometers. For more accurate data, see the `usahi` atlas data or the high resolution data accesible throught the external interface described in Chapter 7.

```
figure
axesm('MapProjection','mercator','Frame','on',...
  'MapLatLimit',[41 44],'MapLonLimit',[-72 -69])
gridm('MLineLocation',1,'PLineLocation',1)
mlabel('MLabelLocation',1); plabel('PLabelLocation',1)
h = displaym(POpatch,'united states');
set(h,'FaceColor',.5*[1 1 1])
```



There are some more convenient ways of working with the `worldlo` atlas data. The `worldmap` function makes it easy to create map displays. Just specify the region or country you wish to map. This function will handle the cartographic details like selecting a projection, setting the map limits, and setting the origin and the grid and label spacings. You can also use the `worldlo` interface function to load one of the atlas data structures as an argument to a function, or load just that structure into the workspace. Make a map of Africa with filled patches, and add the locations of major cities to the display.

```
worldmap('africa','patch'); polcmap(256)
h = displaym(worldlo('PPpoint')); setm(h,'Color','r')
```



The `worldlo` atlas data can also be used to look up names and locations. All of the names mentioned in the `POtext` and `PPtext` structures are collected in a gazette structure, which can be queried using `extractm`. For example, the location of the city of Antananarivo on the island of Madagascar can be found by using the following commands:

```
[lat,long,indx] = extractm(gazette,'antan');
gazette(indx)
ans =
             type: 'line'
    otherproperty: []
              tag: 'Antananarivo'
           string: 'Populated Place Name'
         altitude: []
              lat: -18.6900
             long: 47.4480
```

A much more extensive gazette feature can be found in the Digital Chart of the World (DCW), containing more than 10,000 names, compared to about 500 in the `worldlo` MAT-file. See Chapter 7, “External Data Interface” for more information on the DCW.

World Matrix Data

Political

The Mapping Toolbox includes a set of matrix data coded with political information in the MAT-file `worldmtx`.

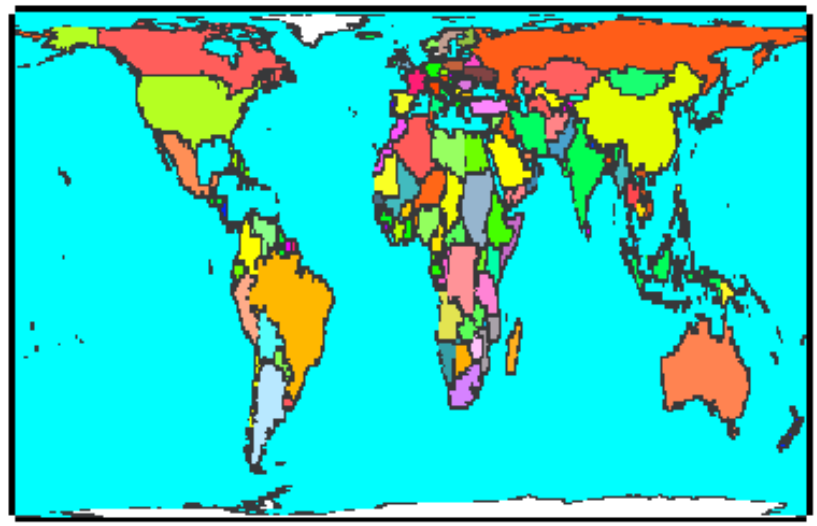
```
load worldmtx
whos
   Name            Size      Bytes    Class

   clrmap          195x3      4680    double array
   map           180x360    518400    double array
   maplegend         1x3        24    double array
   nations         1x195     21324    struct array
```

The variable `map` is a regular matrix map at the same resolution (180 by 360) as the `topo` data, but in this case, it is an indexed map of political entities, as opposed to a matrix of elevations and bathymetric data. The `nations` structure relates the indices in the map to the names of countries, and the variable `clrmap` is a colormap that provides a good political display of the world.

Display the political regular matrix map using the `meshm` function and the provided colormap:

```
axesm gortho
meshm(map,maplegend); colormap(clrmap)
```

The map is shown in a Gall Orthographic projection, more recently made familiar as the Peters projection. This projection is equal-area, a property that is often desirable in representing competing political units.



The low resolution of this data makes it suitable only for global displays. Larger scale, regional maps are better made with vector data such as that in the `worldlo` database. More detailed political matrix maps can be created from vector data using the techniques described in Chapter 6, “GUI Tools” of this document.

The primary use of this dataset is to determine which country contains a point known by its latitude and longitude. What country claims sovereignty over the point at 5° E and 13° N?

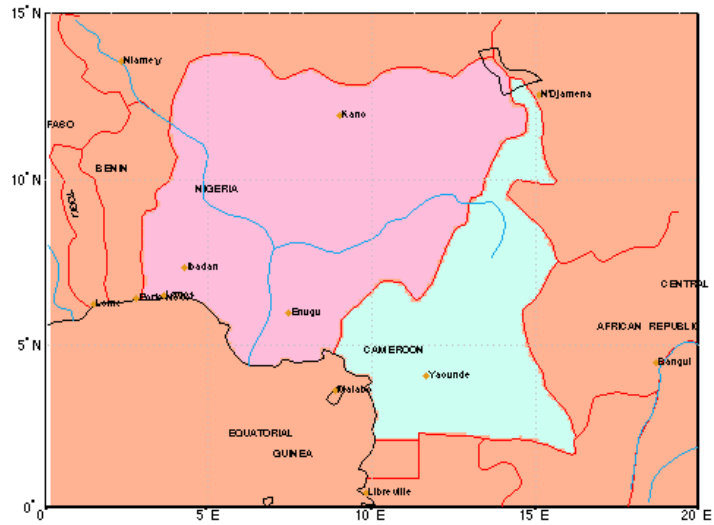
```
code = ltln2val(map,maplegend,5,13)
code =
  34

nations(code)
ans =
  name: 'Cameroon'
```

Work through this example to see how the result was obtained. The matrix `map` contains integer country codes. By calling the function `ltln2val`, we extracted the country code of the geographic point. The country code is an index into the `nations` structure, which returns the name of the country.

The coarseness of this dataset imposes some limitations. Just like any other matrix entry, the coastline and borders between nations are one degree quadrangles that may cover more than 100 kilometers on a side. The code for a border is often returned when you query areas with small nations. The actual shape of small countries is also poorly represented by such large cells. For instance, the area represented by the map in the `cape` MAT-file would cover just 9 cells for this dataset.

You can create your own matrix maps at higher resolutions. If the worldlo atlas data is good enough, use country2mtx. If you have higher resolution data, use vec2mtx. Here is a matrix map of Cameroon and Nigeria at a resolution of 10 cells per degree, or about 10 kilometers.

```
latlim = [0 15]; lonlim = [0 20];

[cmap,cmaplegend] = country2mtx('cameroon',10,latlim,lonlim);
[nmap,nmaplegend] = country2mtx('nigeria',10,latlim,lonlim);

newmap = zeros(size(cmap)); newmaplegend = cmaplegend;

newmap(cmap==0) = 34;  % find cells in the interior
newmap(nmap==0) = 130; % use codes from nations structure

worldmap(latlim,lonlim)
meshm(newmap,newmaplegend,size(newmap))
polcmap
```



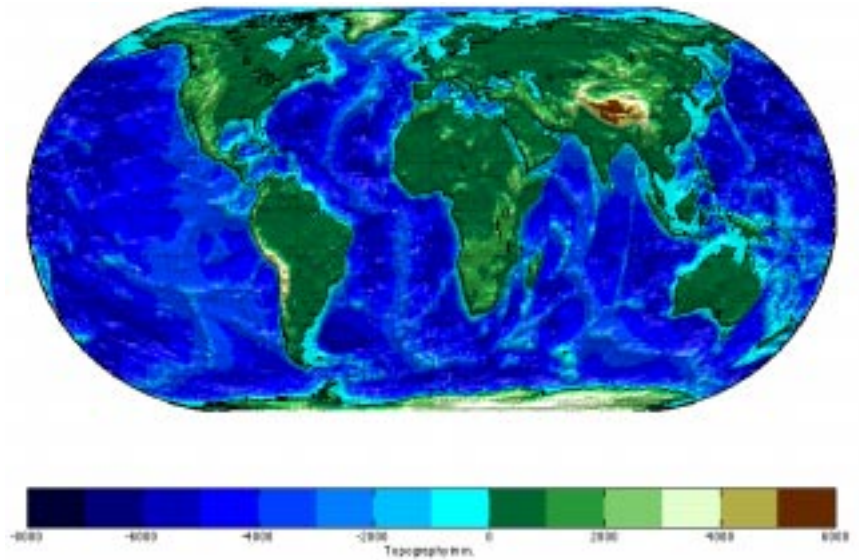
Terrain

A low resolution set of global elevation data is provided with MATLAB in the topo workspace. It is a regular matrix of average elevations and depths (in meters) for a constant grid spacing of one degree by one degree.

```
load topo
whos
   Name              Size      Bytes    Class

   topo           180x360     518400    double array
   topolegend         1x3         24    double array
   topomap1          64x3       1536    double array
   topomap2         128x3       3072    double array
```

Here it is displayed in an equal-area Eckert IV projection as a shaded-relief surface, which allows both large and small features to be seen:

```
axesm eckert4; framem; gridm
plotm(coast,'k'); zdatam(handlem('line'),max(topo(:)))

meshm(topo,topolegend,size(topo),topo,'FaceColor','interp')
demcmap('inc',topo,1000)

camlight(0,80);; material([0.7 0.8 .4]); daspectm('m',300)
hcb = colorbar('horiz')
set(get(hcb,'XLabel'),'String','Topography in m.')
```

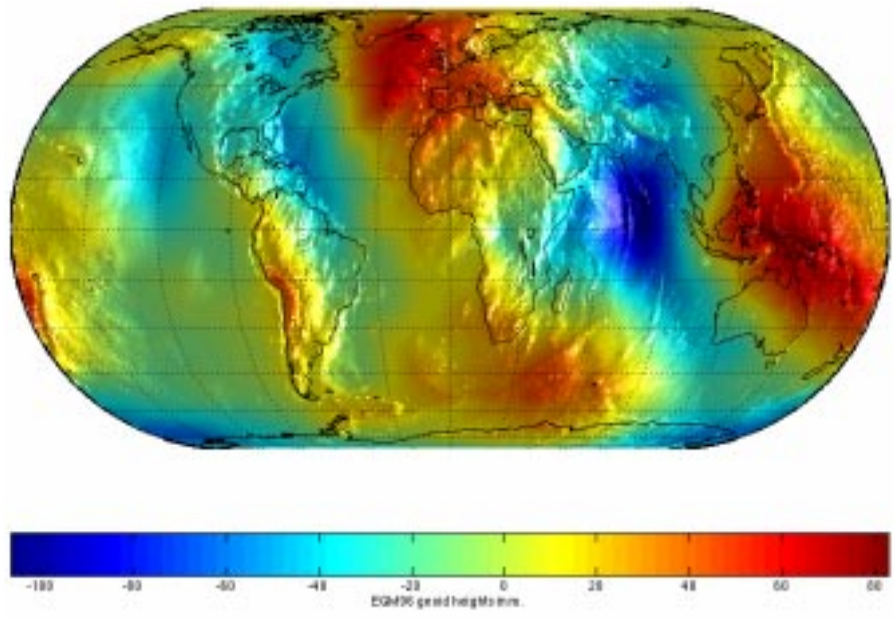


Geoid

While the `topo` matrix contains the heights and depths of the Earth's crust, the `geoid` matrix gives the shape of its gravitational field. The geoid can be regarded as the ocean surface formed in the absence of waves, currents, tides, and land obstructions. In technical terms, it is a gravitational equipotential surface. Elsewhere, this document refers to the ellipsoidal shape of the Earth as the geoid. Although this is sufficient for cartography, the ellipsoidal approximation is not accurate enough for applications like surveying and geodesy, and more detailed models are used. The EGM96 geoid model is a widely used spherical harmonic model of the geoid complete to degree and order 360. A 1-degree grid of geoid heights is provided in the `geoid.mat` atlas file. The full resolution version of the matrix of geoid heights with a grid spacing of 15 minutes can be read using the `egmgeoid96` external interface function described in Chapter 7. See the External Data Reference Guide for more information on the EGM96 geoid model.

```
load geoid
figure; axesm eckert4; framem; gridm

plotm(coast,'k')
zdatam(handlem('line'),max(geoid(:)))
meshm(geoid,geoidlegend,size(geoid),geoid,'Facecolor','interp')
light; material(0.6*[ 1 1 1])

set(gca,'dataaspectratio',[ 1 1 200])
hcb = colorbar('horiz')
set(get(hcb,'Xlabel'),'String','EGM96 geoid heights in m.')
```

Here is the result:



```
z = ltln2val(geoid,geoidlegend,-11.1,130.22,'bicubic')
z =
       47.369
```

Interpolating the full resolution grid gives a result that is 5 meters higher.

United States Vector Data

Low Resolution Data

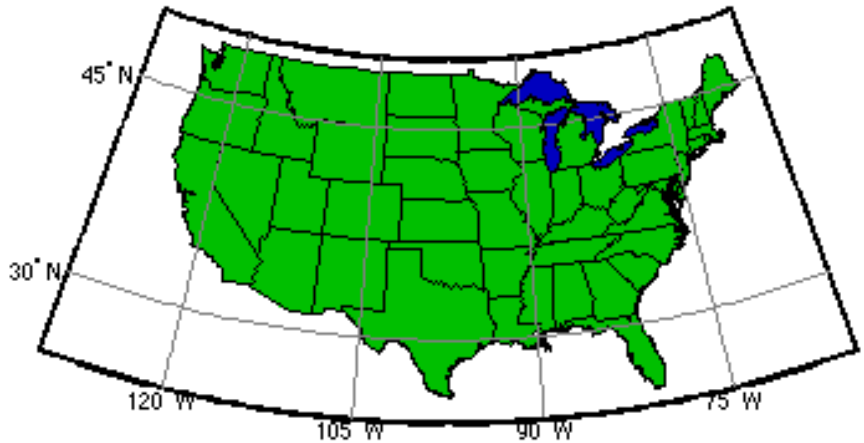
Vector political and coastline data for the United States are provided in a number of useful formats in the `usalo` workspace.

```
load usalo
whos
   Name              Size      Bytes    Class

   conus              1x1      71072    struct array
   greatlakes         1x3      22220    struct array
   gtlakelat       1229x1       9832    double array
   gtlakelon       1229x1       9832    double array
   state             1x51     256986    struct array
   stateborder        1x1      38390    struct array
   statelat        2345x1      18760    double array
   statelon        2345x1      18760    double array
   uslat           4339x1      34712    double array
   uslon           4339x1      34712    double array
```

There are several types of data here. The double arrays are vectors of latitude and longitude points. The pair `uslat` and `uslon` form the outline of the coast and political borders of the continental United States. The polygon closes on itself, so it may be filled as a patch. Next, the `statelat` and `statelon` form the borders between the states for display as lines. Finally, `gtlakelat` and `gtlakelon` are patchable outlines of the Great Lakes. The commands suitable for displaying vector data of this type are `plotm`, `linem`, `fillm`, `patchm`, and `patchesm`.

Display the data vectors as patches and lines:

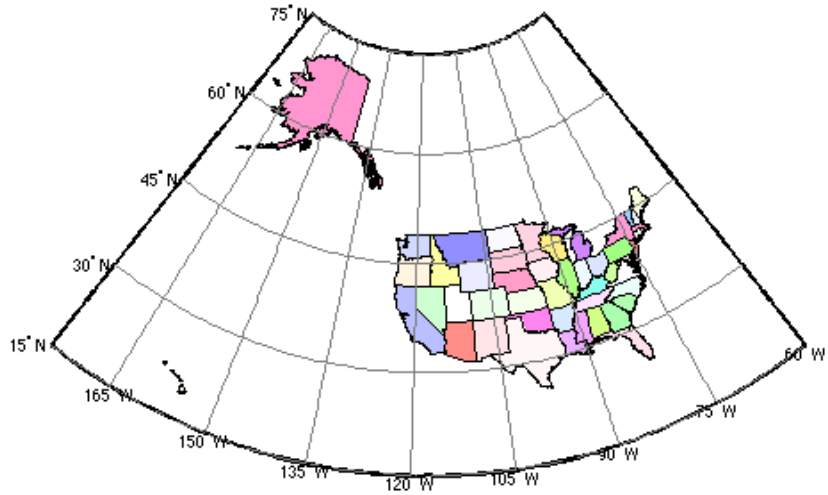
```
axesm('MapProjection','eqaconic','MapParallels',[29.5 49.5],...
  'MapLatLimit',[24 50],'MapLonLimit',[-130 -65],...
  'MLabelLocation',15,'MLineLocation',15,...
  'PLabelLocation',15,'PLineLocation',15,...
  'GColor',.5*[1 1 1],'GLineStyle','-',...
  'MLabelparallel','south')
framem; gridm; mlabel; plabel
patchm(uslat,uslon,[0 .75 0])
patchm(gtlakelat,gtlakelon,1,[0 0 .75])
plotm(statelat,statelon,'k')
```



The structure arrays are Mapping Toolbox geographic data structures and can be displayed using `mlayers` or `displaym`. The data in `stateborder` is identical to that in `statelat` and `statelon`. Similarly, `conus` contains the same outline of the continental United States as `uslat` and `uslon` but is defined to be displayed as a patch. The 50 state and District of Columbia patches are located in the `state` structure.

Display the `state` structure using `displaym`. Notice the only one command is needed to plot 51 different patches.

```
figure
axesm('MapProjection','eqaconic','MapParallels',[],...
  'MapLatLimit',[15 75],'MapLonLimit',[-175 -60],...
  'MLabelLocation',15,'MLineLocation',15,...
  'PLabelLocation',15,'PLineLocation',15,...
  'GColor',.5*[1 1 1],'GLineStyle','-',...
  'MLabelParallel','south')
framem; gridm; mlabel; plabel
displaym(state); polcmap
```

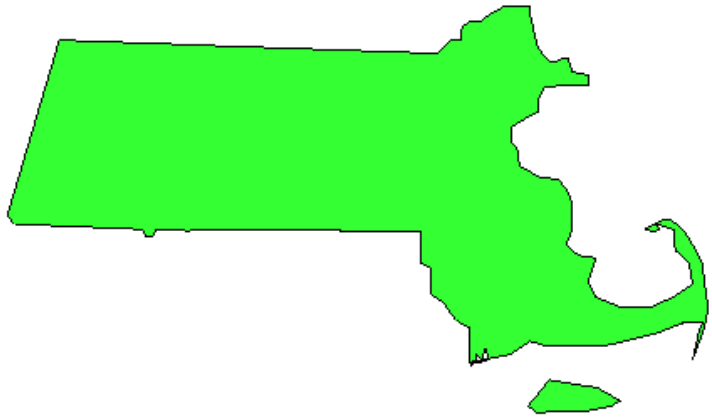


The states are all tagged, allowing them to be manipulated or to be extracted by name. You can see the available state names by entering the following:

```
strvcat(state.tag)
```

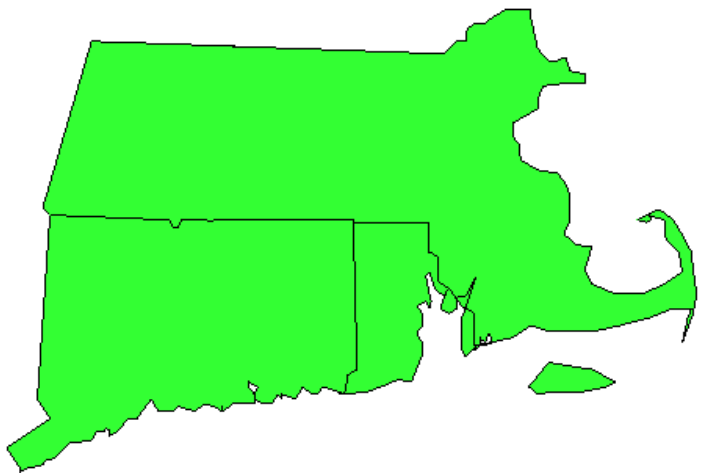
The state of Massachusetts can be displayed by itself:

```
displaym(state,'mass');
```



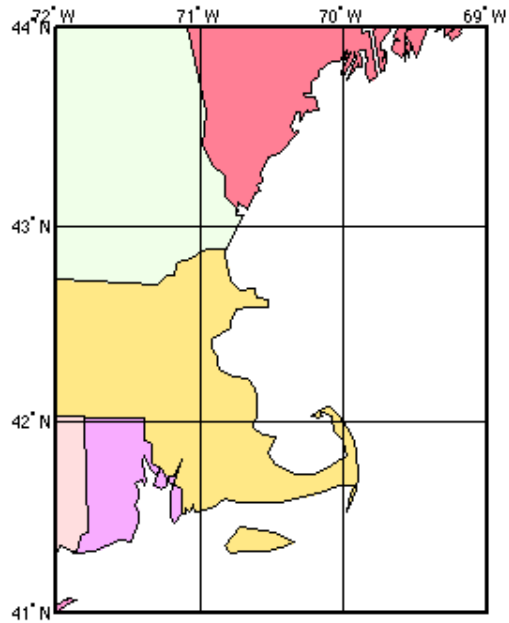
Or with its neighboring southern states, Connecticut and Rhode Island:

```
displaym(state,strvcat('mass','conn','rhode'));
```



Look at the region around Cape Cod again, to get an impression of the detail of the data:

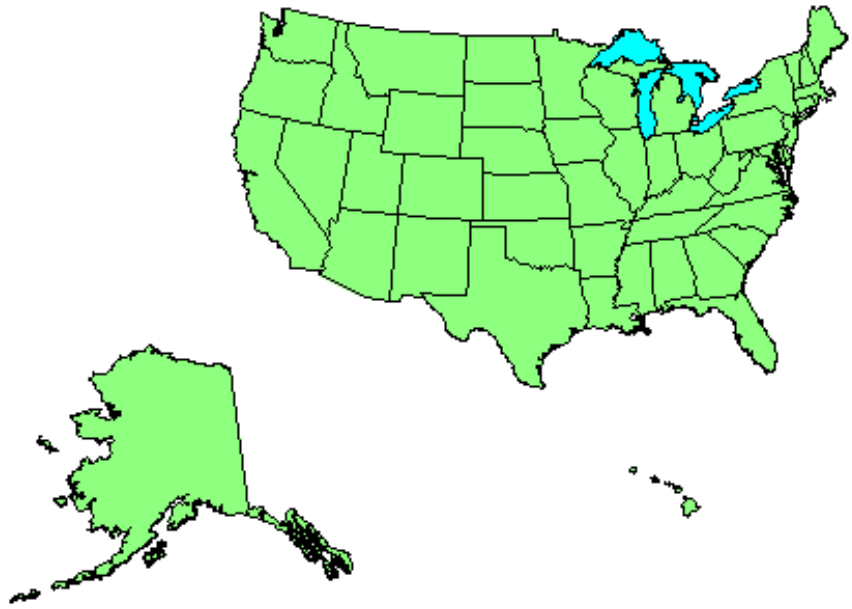
```
figure
axesm('MapProjection','mercator','Frame','on',...
  'MapLatLimit',[41 44],'MapLonLimit',[-72 -69])
gridm('MLineLocation',1,'PLineLocation',1,...
  'GLineStyle','-')
mlabel('MLabelLocation',1)
plabel('PLabelLocation',1)
displaym(state); polcmap
```



There is an obvious compromise between the size of a dataset and its accuracy and detail. This data can be displayed relatively quickly for displays of all the United States, but it is not suitable for mapping smaller regions. Note the generalized character of the coastline and the absence of small islands. More detailed and accurate data is available in the `usahi` workspace and through the Mapping Toolbox External Data Interface functions.

An easy way to display the `usalo` data is with the `usamap` function. The `usamap` function takes care of the cartographic details. The inset maps of the entire

United States are made with the `usalo` data. Here are the fifty states displayed at the same scale.



Medium Resolution State Outlines

Another more detailed set of state outlines is available in the Mapping Toolbox. The data in the `usahi` MAT-file is similar in format to the `state` structure found in `usalo` but contains more detailed coastlines and islands. The resolution of `stateline` in the `usahi` workspace is about three times greater than the corresponding `state` in the `usalo` workspace.

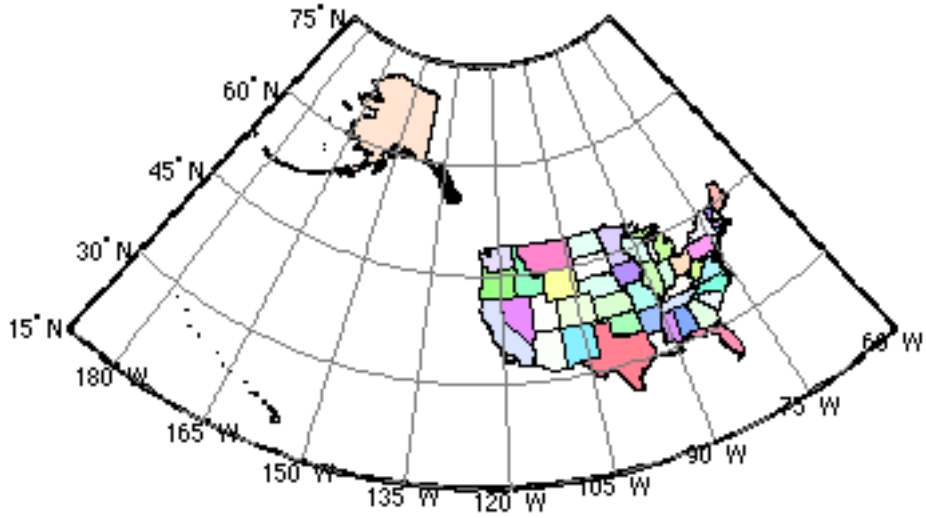
```
load usahi
whos
   Name             Size      Bytes    Class

   statelin         1x51     837656    struct array
   statepatch       1x51     837758    struct array
   statetext        1x51      51190    struct array
```

The listed variables are all formatted geographic data structures containing lines, patches, or text. The data in the `stateline` structure is the same as that in `statepatch` but has been defined to display as lines rather than patches. The structure `statetext` contains the corresponding names of the states. Note that because of the higher resolution of this data, it may require more than the default memory size to display and will take longer to project and render. This difference is particularly marked for patches.

Display the map in an Equal-Area Conic projection:

```
axesm('MapProjection','eqaconic','mapparallels',[],...
  'MapLatLimit',[15 75],'MapLonLimit',[-188 -60],...
  'MLabelLocation',15,'MLineLocation',15,...
  'PLabelLocation',15,'PLineLocation',15,...
  'GColor',.5*[1 1 1],'GLineStyle','-',...
  'MLabelParallel','south')
framem; gridm; mlabel; plabel
displaym(statepatch); polcmap
```

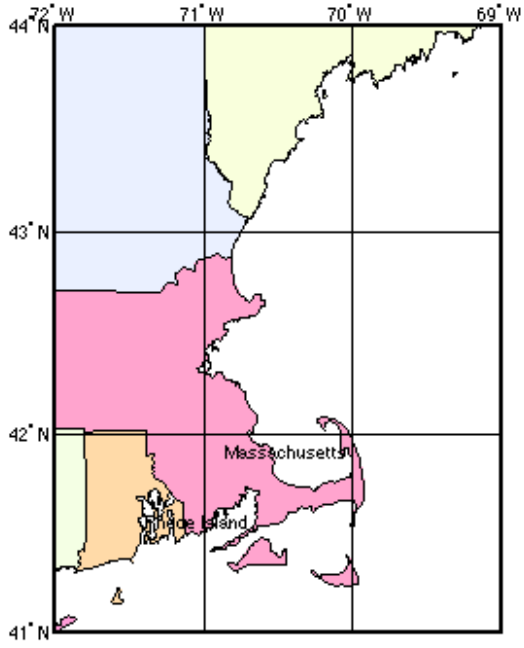
Here is the result:



At this scale, it is difficult to see the difference between this data and the much smaller usalo workspace. By focusing on the Cape Cod region, the higher resolution becomes more apparent:

```
figure
axesm('MapProjection','mercator','Frame','on',...
  'MapLatLimit',[41 44],'MapLonLimit',[-72 -69])
gridm('MLineLocation',1,'PLineLocation',1,'GLineStyle','-')
mlabel('MLabelLocation',1)
plabel('PLabelLocation',1)
displaym(statepatch)
h = displaym(statetext); trimcart(h)
polcmap
```

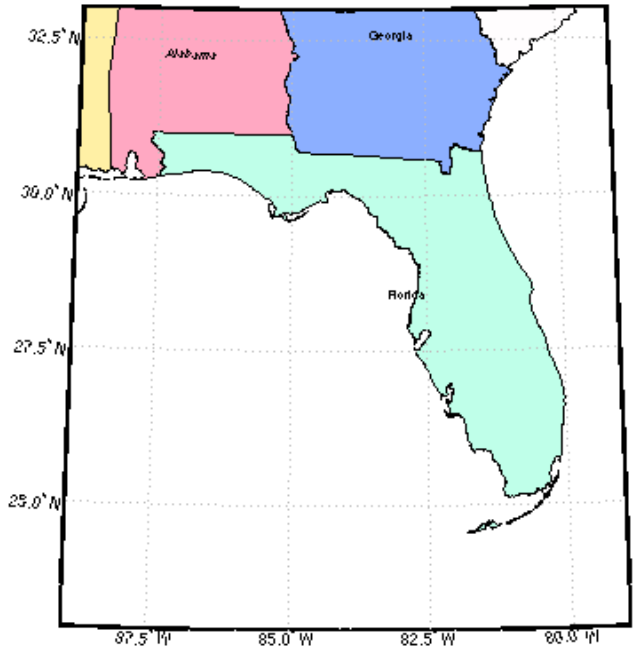
Here is the zoomed-in region of Cape Cod. The level of detail is noticeably greater than that for the data in the usalo MAT-file.



You can use usamap to quickly create base maps with this data. Just provide the state name or geographic limits. This function will handle the cartographic

details like selecting a projection, setting the map limits, and setting the origin and the grid and label spacings. You can also use the `usahi` interface function to load one of the atlas data structures as an argument to a function, or load just that structure into the workspace.

```
figure
usamap florida
```



United States Matrix Data

Political

The Mapping Toolbox also provides a set of political matrix data for the continental United States located in the usamtx MAT-File.

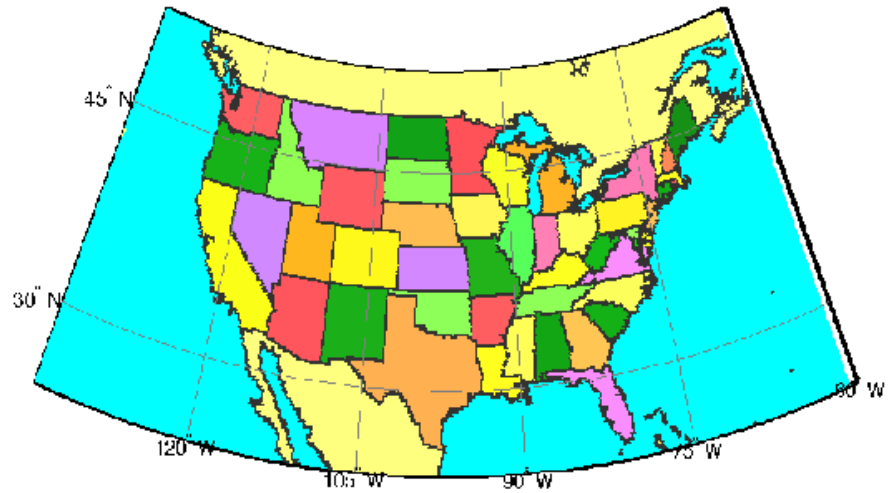
```
load usamtx
whos
   Name            Size      Bytes    Class

   clrmap          54x3       1296    double array
   map           140x370     414400    double array
   maplegend       1x3          24    double array
   states         54x15       1620    char array
```

The data is in a regular matrix map format, similar to that discussed in the "World Matrix Data" section. The variable map is at a resolution of 5 cells per degree, or roughly 20 kilometers or better. Recall that the resolution of the worldmtx data is 1 cell per degree; hence, this dataset is five times finer. A colormap has been provided, along with a vector of state names, both corresponding to the political code, or *index*, used in the map matrix.

Display the matrix map in an Albers Equal-Area Conic projection:

```
axesm('MapProjection','eqaconic','MapParallels',[],...
  'MapLatLimit',[24 52],'MapLonLimit',[-134 -60],...
  'MLabelLocation',15,'MLineLocation',15,...
  'PLabelLocation',15,'PLineLocation',15,...
  'GColor',.5*[1 1 1],'GLinestyle','-',...
  'MLabelParallel','south')
framem; gridm; mlabel; plabel
meshm(map,maplegend); colormap(clrmap)
```

Here is the result:



As with the world matrix data, the low resolution of this data makes it suitable for only large area displays. Vector maps are more appropriate where more detail is required. Chapter 6, “GUI Tools” describes how to create higher resolution matrix maps from vector maps.

The primary use of this dataset is to determine which state contains a particular geographic point. In what state is Alamogordo (32.8990° N and 105.9570° W)?

```
code = ltln2val(map,maplegend,32.8990,-105.9570)
code =
  34

states(code,:)
ans =
New Mexico
```

You can create your own higher resolution political matrix maps using `vec2mtx`.

Terrain

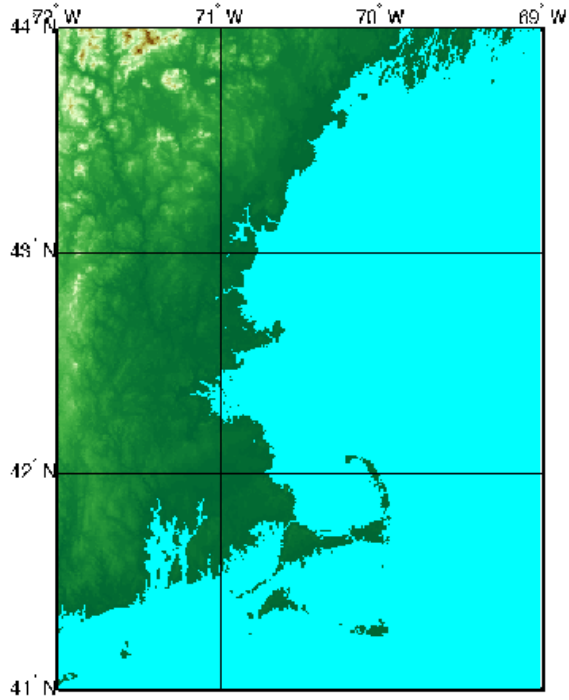
As an example of a higher resolution digital elevation map, MATLAB provides the cape workspace, containing an image of elevation data for the northeastern United States on a 30 arc-second grid (resolution of about one kilometer or better on the ground). The data can be defined as a regular matrix map using the loadcape function script, which rearranges the data and provides the necessary map legend:

```
loadcape
whos
   Name            Size      Bytes    Class

   caption          2x55        220    char array
   cmap            192x3       4608    double array
   map           360x360    1036800    double array
   maplegend         1x3         24    double array
```

Here we display the elevation data using a conformal Mercator projection, so shapes of small regions suffer little distortion, while the distortion in relative areas is scarcely noticeable for such a small region.

```
axesm('MapProjection','mercator',...
  'MapLatLimit',[41 44],'MapLonLimit',[-72 -69])
framem
gridm('MLineLocation',1,'Plinelocation',1,'GLineStyle','-')
mlabel('MLabelLocation',1)
plabel('PLabelLocation',1)
meshm(map,maplegend); demcmap(map)
```

Here is the result:



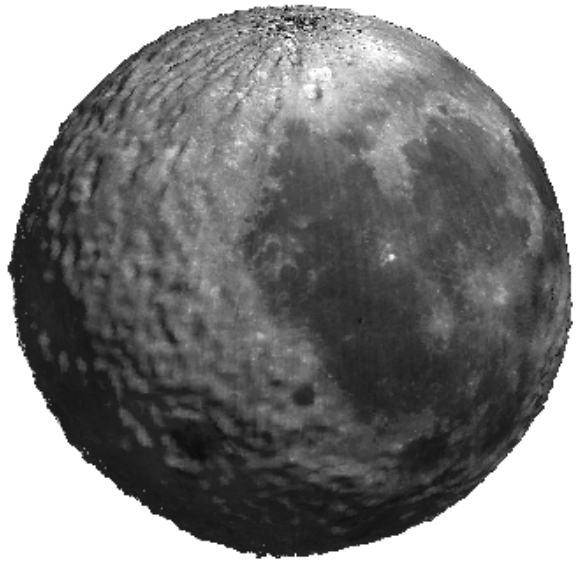
Global coverage of digital terrain and bathymetry at this resolution is provided through the Mapping Toolbox External Data Interface. A variety of freely available digital elevation maps is available over the Internet for import into MATLAB. These maps range in resolution from about 10 km to 30 meters. See Chapter 7, “External Data Interface” for more information on the data and import functions.

Astronomical Data

Although the Earth may be the most commonly mapped object, the same cartographic techniques are used to map the stars and planets. As an example of such astronomical data, the Mapping Toolbox includes map data for the Earth's moon and the stars.

The moon data is low resolution topography and albedo (reflectance) from the Clementine spacecraft. Topography, found in the `moontopo.mat` file, is a 1-degree regular matrix map in units of meters. Albedo is stored as a 3-cells-per-degree integer image in the `moonalb.mat` file. The `loadmoonalb` script reads the albedo image and converts it to double precision numbers for display. Here is the albedo image texture mapped onto the topography, vertically exaggerated by a factor of 10.

```
load moontopo; loadmoonalb
axesm('globe','geoid',almanac('moon','radius','m'))

h = meshm(moontopo,moontopolegend,size(moontopo),10*moontopo);
set(h,'CData',moonalb,'FaceColor','texturemap')
colormap(gray); view(20,20)
camlight; material dull; lighting phong
```



Star positions and magnitudes are found in the stars MAT-file.

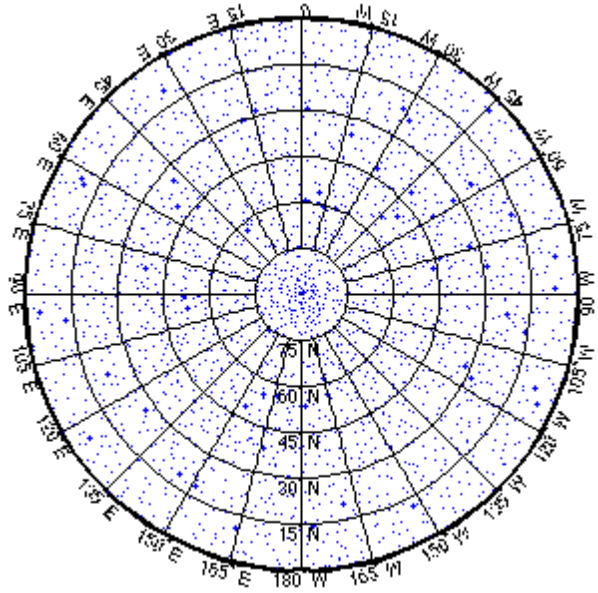
```
load stars
whos
   Name              Size      Bytes    Class

   glat            4652x1      37216    double array
   glong           4652x1      37216    double array
   lat             4652x1      37216    double array
   long            4652x1      37216    double array
   relintensity    4652x1      37216    double array
   vmag            4652x1      37216    double array
```

This is a set of more than 4500 star locations and visual magnitudes derived from the Fifth Fundamental Catalog of Stars parts I and II (FK5 and FK5e). Other star data such as mean errors, proper motions, spectral types, parallaxes, radial velocities, and cross identifications to other catalogues can be obtained from the catalog using the interface function described in Chapter 7, “External Data Interface”.

The positions of stars in equatorial coordinates are given in the vectors lat and long. These are latitudes and longitudes in degrees for the stars as seen from within the celestial sphere. The visual magnitude of each star is given in the vector vmag. Stars are typically plotted with diameters proportional to the relative intensity. The vector relintensity contains modified vmag data suitable for display with scatterm to exhibit this proportional diameter. All intensities are positive, and brighter stars have a larger relintensity number.

Display the stars of the northern sky in the Equidistant azimuthal projection. Scale the realive intensities to:

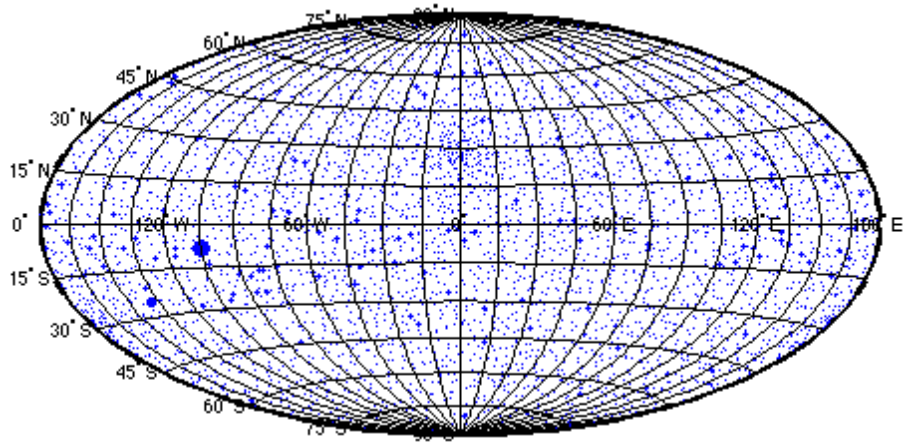
```
axesm eqdazim
framem; gridm; mlabel; plabel
setm(gca,'Origin',[90 180 0],'FLatLimit',[-inf 90],...
  'MLabelLocation',-180:15:165,'MLineLocation',15,...
  'MLineLimit',[-75 75],'MLabelParallel','equator',...
  'PLabelMeridian',180,'PLabelLocation',15:15:75,...
  'LabelRotation','on''GLineWidth',.01,...
  'GLineStyle','-')
set(handlem('alltext'),...
  'VerticalAlignment','top','HorizontalAlignment','center')
h = scatterm(lat,long,sqrt(relintensity)*30,'filled');
```



You may be able to identify the Big Dipper, or the constellation of Ursa Major, located at 180° E, 60° N.

Another system used to describe the positions of heavenly bodies is galactic coordinates. This coordinate system puts the center of our galaxy at the origin and places the north pole so that the Milky Way is aligned with the galactic equator. The positions of the stars in this coordinate system are given in the vectors `glat` and `glong`. Stars in galactic coordinates are typically plotted in a pseudo-cylindrical projection like the Hammer:

```
figure
axesm hammer
framem; gridm; mlabel; plabel
setm(gca,'GLineWidth',.01,'GLineStyle','-',...
  'MLineLocation',15,'MLabelParallel','equator',...
  'MLabelLocation',-120:60:180)
scatterm(glat,glong,sqrt(relintensity)*30,'filled')
```



See Chapter 7, “External Data Interface” for more information on how to convert between these two systems.

4

Map Projections

What Is a Projection?

It has long been established that the shape of the Earth resembles a sphere and not a flat surface. If the world were indeed flat, cartography would be a trivial matter and map projections would be quite unnecessary.

To create a two-dimensional representation of a curved surface such as the Earth, you must determine how to transform that surface to a plane. The method of this transformation is called a *map projection*, from the geometric methods that were traditionally used to construct maps. While many map projections no longer rely on physical projections, it is useful to think of map projections in geometric terms.

Quantitative Properties

A sphere, unlike a cone or cylinder, cannot be simply reformed into a plane. In order to view the surface of a round body on a two-dimensional flat plane, you must first define a developable surface (i.e., one that can be *cut* and *flattened* onto a plane without stretching) and devise a means of systematically representing all or part of the spherical surface onto the plane. This inevitably leads to distortions of one kind or another. Three characteristic properties of map projections are subject to distortion: area, shape, and scale. A given projection cannot retain more than one of these properties over a large area of the Earth.

An *equal-area* projection is one in which equal spatial units at different points on the map represent the same true area measurement. For example, consider a coin of any size as our spatial unit. The area of the Earth covered by the coin on one part of the map display is exactly equal to that covered by the coin on any other part. Shape and scale must be distorted, although there are some equal-area maps designed to minimize these distortions for at least parts of the map and perhaps preserve shape or scale at specific points.

Other terms for equal-area projections include *equivalent*, *homolographic* or *homalographic* (from the Greek *homalos* or *homos*, same, and *graphos*, write), *authalic* (from the Greek *autos*, same, and *ailos*, area), and *equireal*.

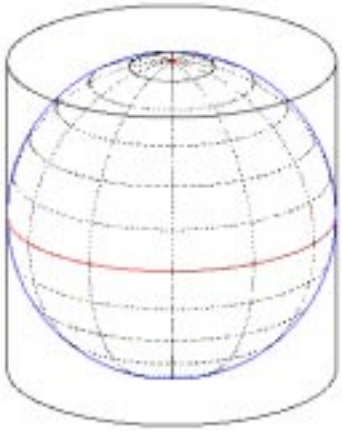
A *conformal* projection is one that preserves the relative local angles (shape) about every point on the map, except for one or more possible *singular* points. Since relative local angles are correct, meridians intersect parallels at right angles (90 degrees). No map can be both equal-area and conformal. Another term for conformal is *orthomorphic* (from the Greek *orthos*, straight, and *morphe*, shape).

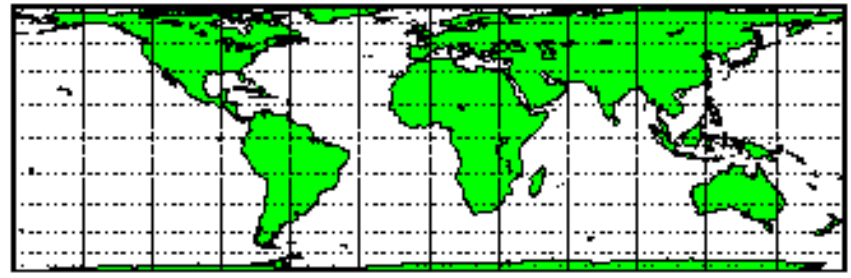
An *equidistant* projection shows true scale between one or two points and every other point on the map, or along every meridian. No map projection shows scale correctly in all directions throughout the entire map.

Geometric Surfaces

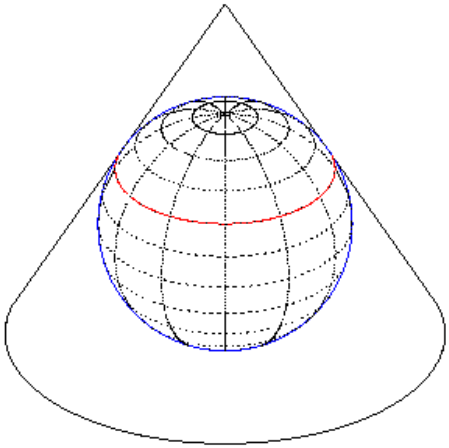
Three standard types of geometric surfaces are used to develop map projections: the cylinder, the cone, and the plane. A few projections, however, cannot be categorized as such, or are combinations of these. The three classifications are used for a wide variety of projections, including some that are not geometrically constructed.

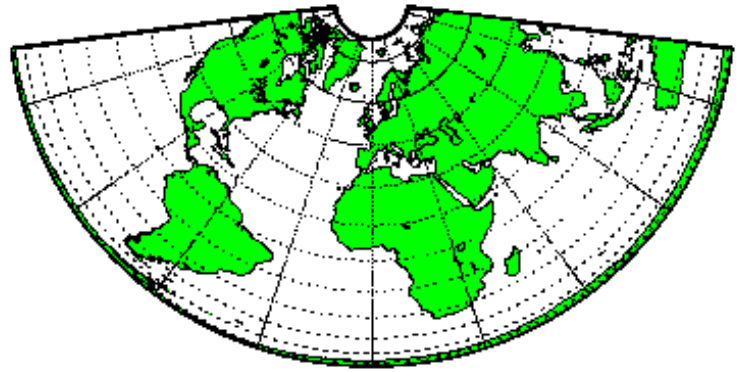
A *cylindrical* projection is produced by wrapping a cylinder around a globe representing the Earth. The map projection is the image of the globe projected onto the cylindrical surface, which is then unwrapped into a flat surface. Parallels appear as horizontal lines and meridians as vertical lines. The following shows a regular cylindrical or *normal aspect* orientation in which the cylinder is tangent to the Earth along the Equator and the projection is calculated horizontally from the axis.



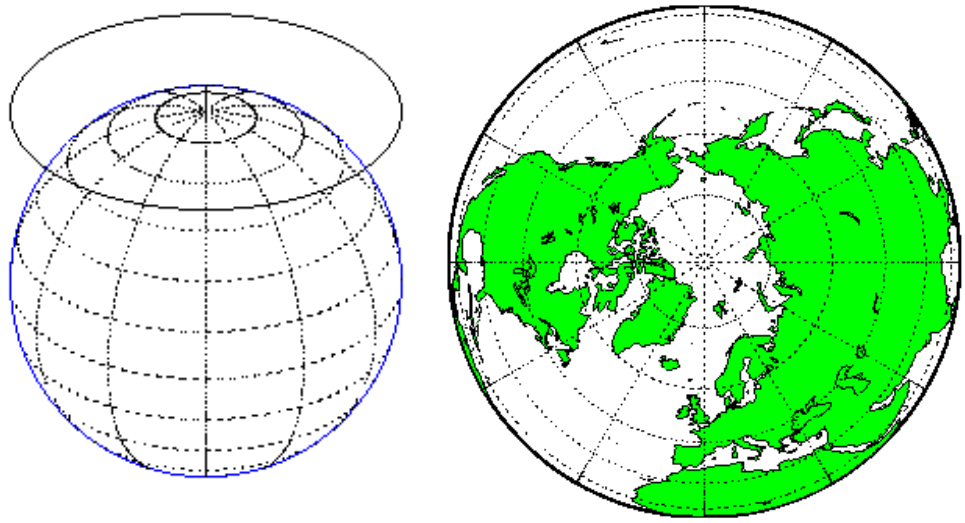


A *conic* projection is derived from the projection of the globe onto a cone placed over it. For the *normal aspect*, the apex of the cone lies on the polar axis of the Earth, and the surface of the cone touches the globe along a particular parallel of latitude. A *polyconic* projection considers each band of parallels as part of separate cones, each tangent to the globe at that particular parallel latitude.





An *azimuthal* projection is a projection of the globe onto a plane. For a polar azimuthal projection, the plane is tangent to the Earth at one of the poles, with meridians projected as straight lines radiating from the pole, and parallels shown as complete circles centered at the pole.



Each of these projection types can be further modified:

- Instead of being tangent to one standard parallel, the cylindrical or conic surface may cut the globe at two parallels. In like manner, the plane for an azimuthal projection may intersect the globe instead of being tangent.
- The vantage point of the projection onto the geometric surface may be changed. The perspective may be chosen from the center of the Earth or along the polar axes (*central* or *gnomonic*), from the opposite side of the Earth's surface (*stereographic*), or from an infinite point in space (*orthographic*).

Projection Aspect

Until now, the discussion of map displays has focused on the *normal* aspect, by far the most commonly used. This section discusses the use of *transverse*, *oblique* and, *skew-oblique* aspects.

Projection aspect is primarily of interest in the display of maps. However, this section will also discuss how the idea of projection aspect as a coordinate system transformation can be applied to map variables for analytical purposes.

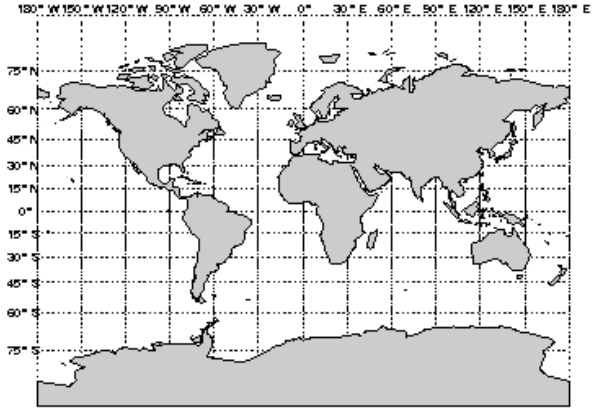
The Origin Vector

A map axes `Origin` property is a vector describing the geometry of the displayed projection. The form of this vector is of the following:

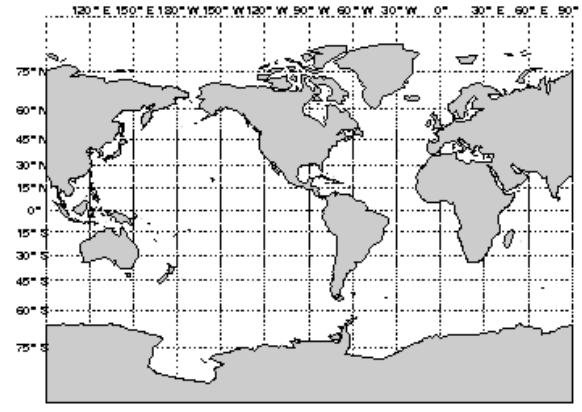
```
origvec = [latitude longitude orientation]
```

The latitude and longitude represent the geographic coordinates of the center point of the display from which the projection is calculated. The orientation refers to the angle from *straight up* at which the North Pole bears from this center point. The default origin vector is [0 0 0], that is, the projection is centered on the geographic point (0°,0°) and the North Pole is *straight up* from this point. Such a display is in a *normal* aspect. In fact, changes to only the longitude value of the origin vector do not change the aspect; a *normal* aspect is one centered on the Equator in latitude with an orientation of 0°.

Both of these Miller projections have normal aspects, despite having different origin vectors:



Origin at (0°,0°), with a 0° Orientation.
(origin vector = [0 0 0])



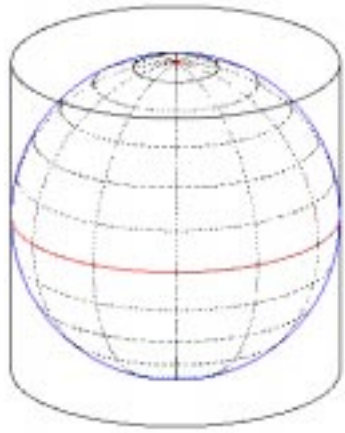
Origin at (0°,90°W), with a 0° Orientation.
(origin vector = [0 -90 0])

This makes sense if you think about a simple, true cylindrical projection. This is the projection of the globe onto a cylinder wrapped around it. For normal aspects, this cylinder is tangent to the globe at the Equator, and changing the origin longitude simply corresponds to rotating the sphere about the longitudinal axis of the cylinder. If we continue with the wrapped-cylinder model, we can understand the other aspects as well.

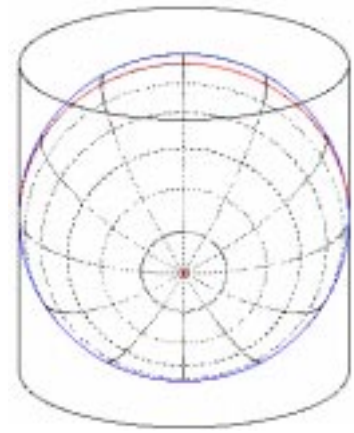
Following this description, a *transverse* projection can be thought of as a cylinder wrapped around the globe tangent at the poles and along a meridian. Finally, when such a cylinder is tangent along an arbitrary great circle, the result is an *oblique* projection.

Here are diagrams of the four cylindrical map orientations, or aspects:

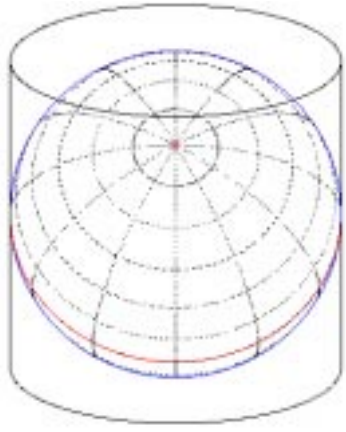
Normal



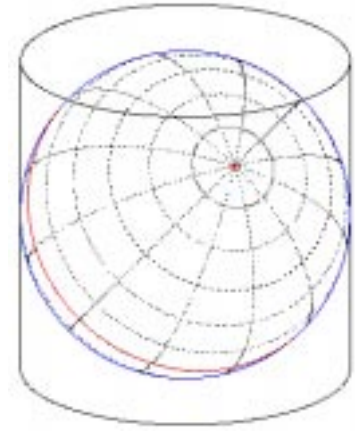
Transverse



Oblique

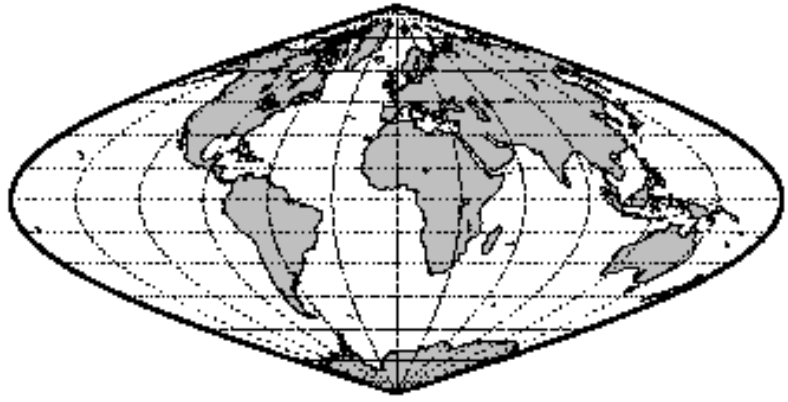


Skew-Oblique



Of course, few projections are actually true cylindrical projections, but the concept of the wrapped cylinder is nonetheless a convenient way to think about it. Perhaps the best way to gain an appreciation for projection aspect is to look at a few examples.

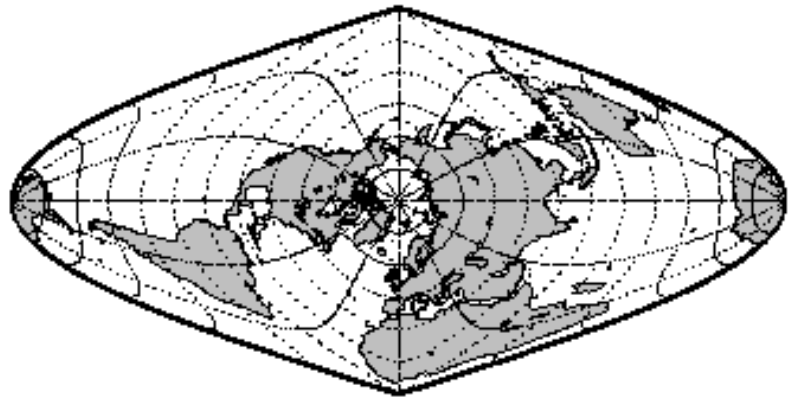
Here, a psuedocylindrical projection, the Sinusoidal, is used. First, look at the normal aspect of this projection:



Normal Aspect: Origin at (0°,0°), Orientation 0°
(origin vector = [0 0 0])

In the normal aspect, the North Pole is at the *top* of the image. To create a transverse aspect, imagine *pulling* the North Pole down to the center of the image, which was originally occupied by the point (0°,0°).

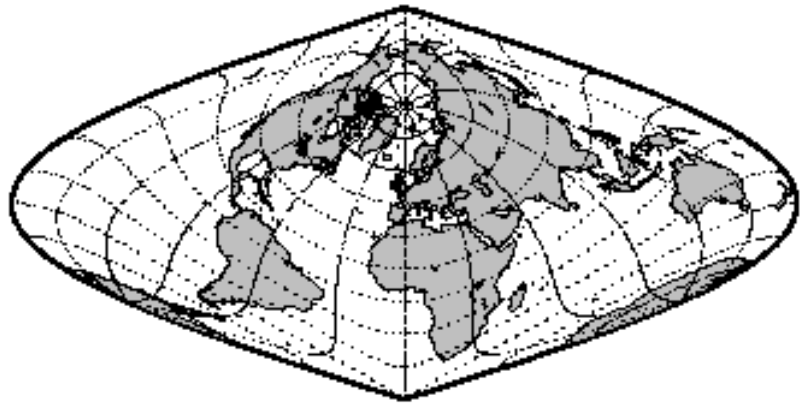
The shape of the frame is unaffected; this is still a Sinusoidal projection:



Transverse Aspect: Origin at (90°N,0°), Orientation 0°
(origin vector = [90 0 0])

The normal and transverse aspects can be thought of as limiting conditions. Anything else is an oblique aspect. Conceptually, if you push the North Pole halfway back to its original position, that is, to the position originally occupied by the point (45°N, 0°) in the normal aspect, the result will be a simple oblique aspect.

Here is the oblique aspect Sinusoidal projection:

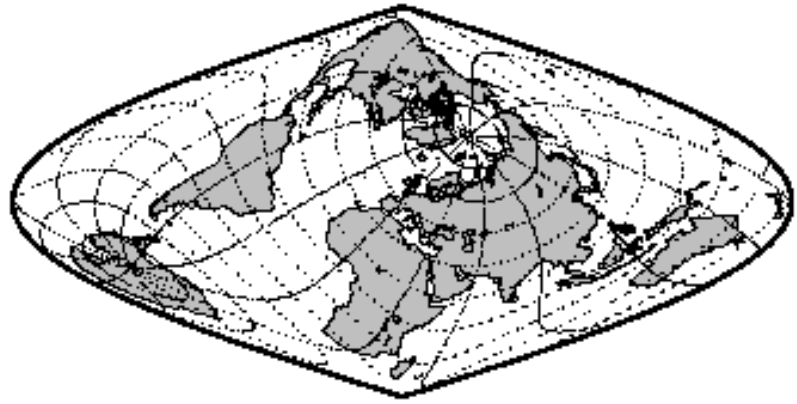


Oblique Aspect: Origin at (45°N,0°), Orientation 0°
(origin vector = [45 0 0])

Another way of understanding this is to imagine the new origin point (45°N,0°) being pulled to the center of the image, to the point originally occupied by (0°,0°) in the normal aspect.

The examples so far have left the aspect orientation at 0°. If the orientation is altered, an oblique aspect becomes a *skew-oblique*. Imagine the previous example with an orientation of 45°. Conceptually, the skew-oblique aspect corresponds to pulling the new origin point, (45°N,0°) down to the center of the projection and then rotating the projection until the North Pole lies at an angle of 45° clockwise from straight up with respect to the new origin.

Here is the result:



Skew-Oblique Aspect: Origin at (45°N,0°), Orientation 45°
(origin vector = [45 0 45])

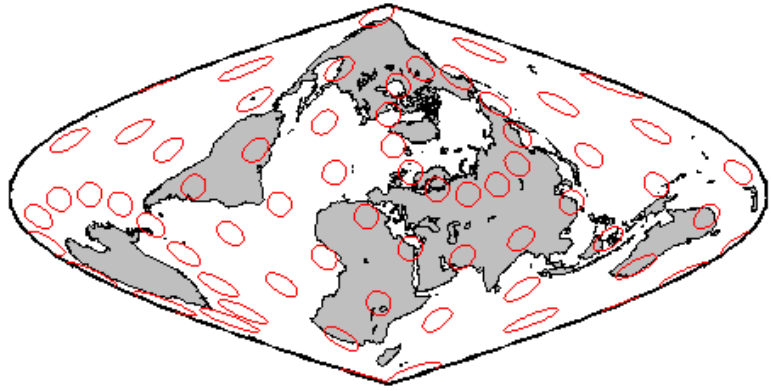
Any projection can be viewed in alternate aspects. Some of these are quite useful. For example, the transverse aspect of the Mercator projection is widely used in cartography, especially for mapping regions with predominantly north-south extent. One candidate for such handling might be Chile. Oblique Mercator projections might be used to map long regions that run neither north and south nor east and west, such as New Zealand.

Note: The projection aspect discussed in this section is different from the map axes `Aspect` property. The map axes `Aspect` property controls the orientation of the figure axes. For instance, if a map is in a `'normal'` setting with a *landscape* orientation, a switch to a `'transverse'` aspect rotates the axes by 90°, resulting in a *portrait* orientation. To display a map in the transverse aspect, combine the `'transverse'` aspect property with a -90° skew angle. The skew angle is the last element of the `Origin` vector. For example, a `[0 0 -90]` vector would produce a transverse map.

The features of a projection are maintained in any aspect, *relative to the base projection*. As we have seen, the *outline*, or frame doesn't change. Non-directional features also do not change. For example, the Sinusoidal projection is equal-area, and is so in any aspect. Directional features must be considered carefully, however. In the normal aspect of the Sinusoidal projection, scale is true along every parallel and the central meridian. This is not the case for the skew-oblique aspect; however, scale is true along the lines where these parallels and meridian would be in the corresponding base projection.

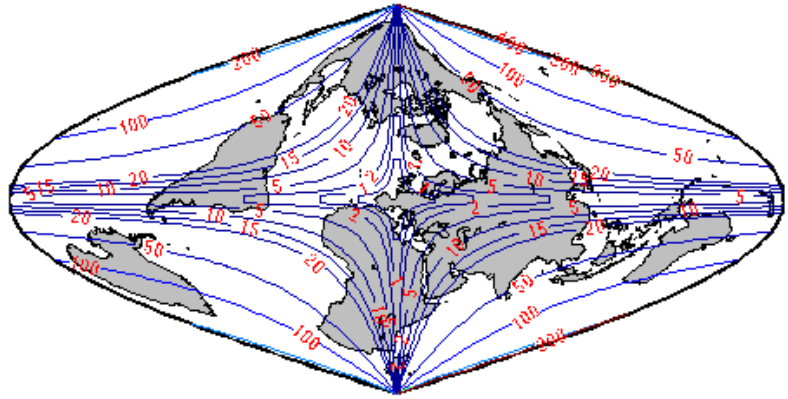
The base projection can be thought of as a standard coordinate system, and the normal aspect conforms to it. The other aspects can be thought of as coordinate transformations.

A standard method of visualizing the distortions introduced by the map projection is to display small circles at regular intervals across the globe. After projection, the small circles appear as ellipses of various sizes, elongations, and orientations. The sizes and shapes of the ellipses reflect the projection distortions. Conformal projections have circular ellipses, while equal area projections have ellipses of the same area. This method was invented by Tissot, and the ellipses are called Tissot indicatrices in his honor. You can add the indicatrices to a map display with the command `tissot` and remove them with `clmo Tissot`.



For a more quantitative look at the projection distortion, use `mdistort`. This function draws contour lines of constant scale, area, and angular distortion on the map. This is useful in selecting projections and projection parameters to keep distortion within limits. Here are the lines of percent scale distortion for

the skew-oblique Sinusoidal projection. The lines of zero distortion along the axes of the base projection are apparent in the contour plot.



Coordinate Transformations

Previously, we discussed the concept of altering the display aspect of a map projection. One way to think of this is to consider redefining the coordinate system, and then displaying a projection based on the new system. For example, think about redefining your coordinate system so that your hometown is the origin. If you calculated a map projection in a normal aspect with respect to this *transformed* coordinate system, the resulting display would look like an oblique aspect with respect to the *true* coordinate system.

This transformation of coordinate systems can be useful independent of map displays. If a transformed coordinate system is selected so that your home town is the new *north pole*, then the transformed coordinates of all other points will provide interesting information.

The distance of each point from your hometown is then simply 90° minus the transformed latitude. For example, the point antipodal to your hometown would be the transformed *south pole*. -90°. Its distance from your hometown is then 90°-(-90°), or 180°, as expected. Points 90° distant from your hometown all have a transformed latitude of 0°. In fact, these points make up the transformed *equator*. The transformed longitudes correspond to the great circle azimuths of those points from your hometown.

When regular matrix maps are transformed in this manner, distance and azimuth calculations with the map variable reduce to row and column operations.

Vector Data – rotatem

As an example of transforming a coordinate system, suppose you live in Midland, Texas, at (32°N,102°W). You have a brother in Tulsa (36.2°N,96°W) and a sister in New Orleans (30°N,90°W).

```
midl_lat = 32;   midl_lon = -102;
tuls_lat = 36.2; tuls_lon = -96;
newo_lat = 30;   newo_lon = -90;
```

What can we say about their homes, relative to yours?

```
dist2tuls = distance(midl_lat,midl_lon,tuls_lat,tuls_lon)
dist2tuls =
  6.5032

dist2newo = distance(midl_lat,midl_lon,newo_lat,newo_lon)
dist2newo =
  10.4727
```

Tulsa is about 6.5 degrees distant, New Orleans about 10.5 degrees distant. The great circle azimuths are the following:

```
az2tuls = azimuth(midl_lat,midl_lon,tuls_lat,tuls_lon)
az2tuls =
  48.1386

az2neworl = azimuth(midl_lat,midl_lon,newo_lat,newo_lon)
az2neworl =
  97.8644
```

The absolute difference in these azimuths is 49.7258°, a fact we will use later.

Today, you feel on top of the world, so make Midland, Texas, the *north pole* of a transformed coordinate system. To do this, first determine the origin required to put Midland at the pole:

```
origin = newpole(midl_lat,midl_lon)
origin =
  58    78     0
```

The origin of the new coordinate system will be (58°N,78°E). Midland is now at a *new latitude* of 90°.

What are the *new coordinates* of Tulsa and New Orleans? To calculate these, use the `rotatem` command. Since it defaults to radians, be sure to include the units input `'degrees'`:

```
[tuls_lat1,tuls_lon1] = rotatem(tuls_lat,tuls_lon,...
                                origin,'forward','degrees')
tuls_lat1 =
  83.4968
tuls_lon1 =
  -48.1386

[newo_lat1,newo_lon1] = rotatem(newo_lat,newo_lon,...
                                origin,'forward','degrees')
newo_lat1 =
  79.5273
newo_lon1 =
  -97.8644
```

Now, test the new coordinate system for the advertised properties. The distance to Tulsa was 6.5032° distant. Its colatitude in the new coordinate system is 90°-83.4968°, or 6.5032°. Similarly, the new colatitude of New Orleans, 90°-79.5273°, is the distance 10.4727°.

Additionally, the absolute difference in the azimuths of the two cities from Midland was 49.7258°. The difference in their *new longitudes* is -48.1386° - (-97.8644°), which is also 49.7258°, as advertised.

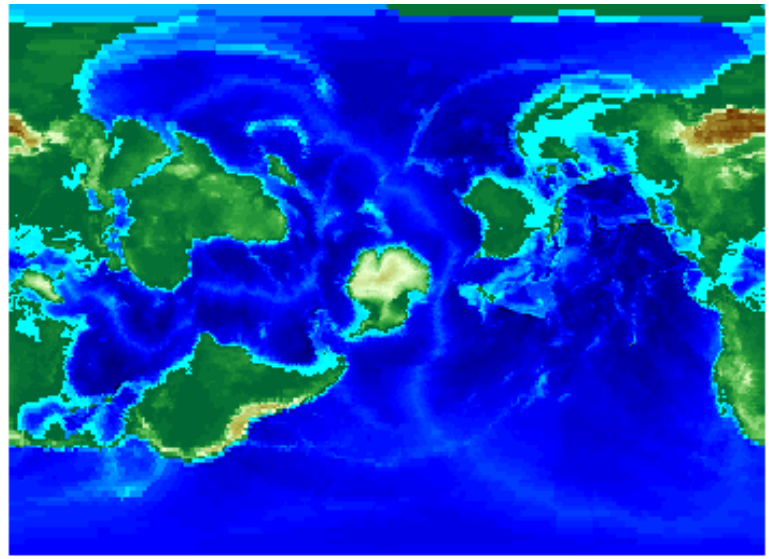
Matrix Data – neworig

A regular matrix map can be used to create a new general matrix map with its data rearranged to correspond to a new coordinate system. Suppose you want to transform the topo data to a new coordinate system in which Sri Lanka (7°N,80°E) is the *North Pole.*

```
load topo
origin = newpole(7,80);
[map,lat,lon] = neworig(topo,topolegend,origin);
```

Display the new map:

```
axesm miller
surfm(map,[30 30]); demcmap(topo)
```

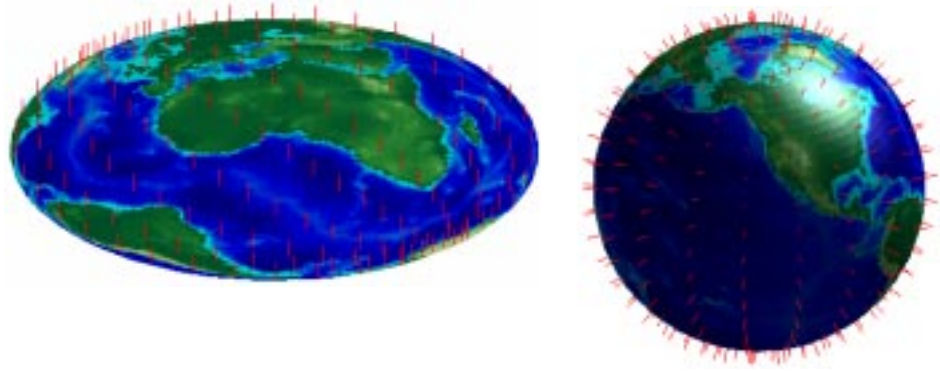


An interesting feature of this new matrix map is that every cell in its first row is 0°-1° distant from the Sri Lankan point (7°N,80°E), and every cell in its second row is 1°-2° distant, etc. Another feature is that every cell in a particular column has the same great circle azimuth from the point.

Working with the Globe Projection

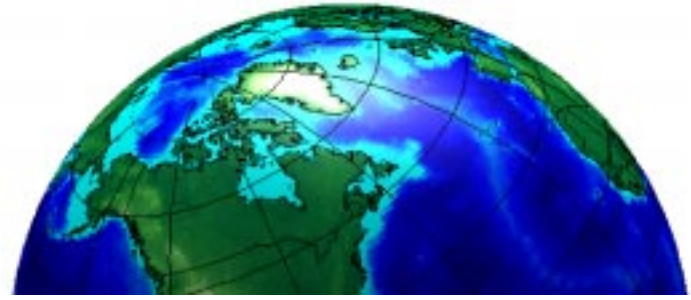
Projection is the process of transforming coordinates from the round spherical coordinate system to flat Cartesian coordinates. The basic assumption of most projections in the Mapping Toolbox is that latitudes and longitudes map to x- and y-coordinates. The single exception is the globe projection, which maps latitudes and longitudes to points on a sphere in Earth-centered Cartesian (x-,y-,z-) coordinates. This makes the globe useful in applications where the proper three-dimensional relationship between objects is important. The following examples illustrate the difference between the two-dimensional orthographic projection, which looks like a globe but is really flat, and the three-dimensional globe

```
load topo
axesm ortho; framem
meshm(topo,topolegend);demcmap(topo)
view(3); daspectm('m',1)

[latgrat,longrat] = meshgrat(topo,topolegend,[20 20]);
stem3m(latgrat,longrat,500000*ones(size(latgrat)),'r')

figure
axesm('globe','Geoid',almanac('earth','radius','m'))
meshm(topo,topolegend); demcmap(topo)
view(3)

[latgrat,longrat] = meshgrat(topo,topolegend,[20 20]);
stem3m(latgrat,longrat,500000*ones(size(latgrat)),'r')
light
```



Because the globe is three-dimensional, having an opaque surface underneath vector data helps to hide lines on the other side of the globe. The following example shows how you can display lines on an underlying surface and use camera position functions to control the view. To make the surface of the globe one color, change the `FaceColor` property of the `topo` surface.

```
figure; axesm('globe','galt',0)
gridm('GLineStyle','-')

load topo
hs = meshm(topo,topolegend,size(topo));
hl = displaym(worldlo('POline')); set(hl,'color','k')
demcmap(topo); camlight; material(0.6*[ 1 1 1])

[tlat,tlon] = extractm(worldlo('gazette'),'Moscow');
[plat,plon] = extractm(worldlo('gazette'),'Washington');

camtargm(tlat,tlon,0); camposm(plat,plon,3)
camupm(tlat,tlon); camva(20)

hidem(gca)
```



Projection Computations

Most of the examples in this document assume that the end product of a map projection is a graphical representation of the map. While this is true for most cases, there may be times when you need access to the non-geographic coordinates of your map data in the projected space.

An easy way to retrieve the projected coordinates of a map that has already been displayed is with the MATLAB `get` command. The projected coordinates are stored in the object's `XData` and `YData` properties. As an example, display a map frame for the Mollweide projection, and extract its x-y coordinates:

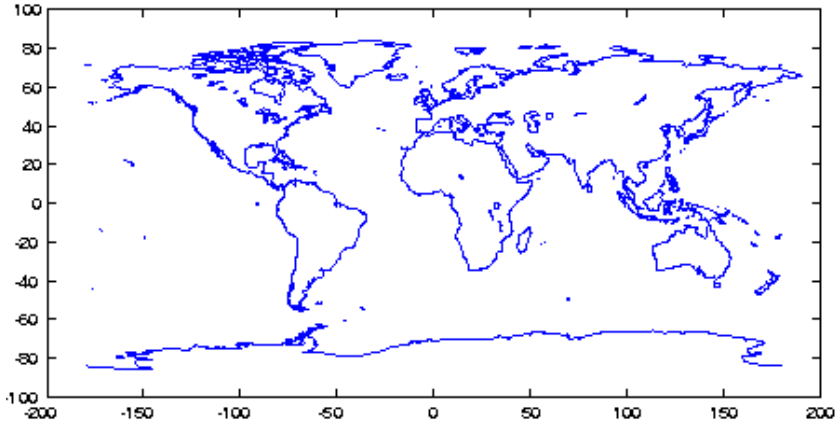
```
axesm mollweid
h = framem;
x = get(h,'XData');
y = get(h,'YData');
```

Of course, you do not need to display a map object to get its projected coordinates. You can perform the same projection computations that are done within the Mapping Toolbox display commands.

Begin by displaying a map of the `coast` vector data:

```
figure
load coast
plot(long,lat); axis equal
set(gca,'XLim',[-200 200],'YLim',[-100 100])
```

Here is the result. Note that some of the data extends beyond 180° longitude.



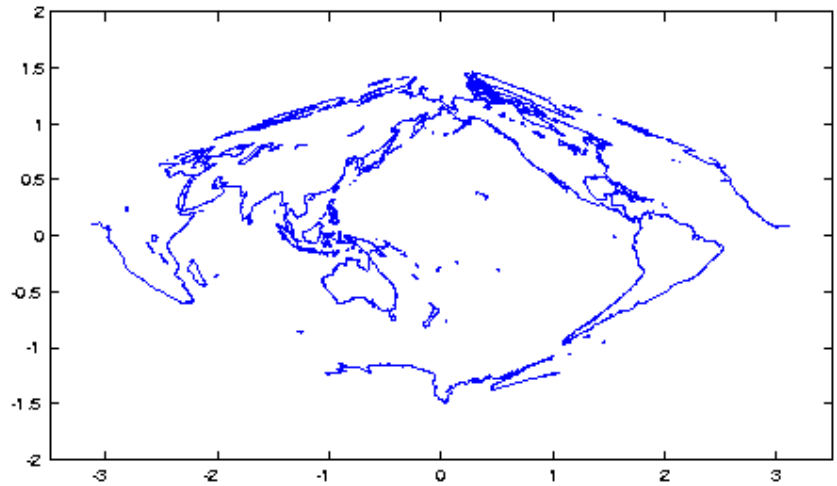
Before projecting the data, you must define the projection parameters, just as you would prepare a map axes with `axesm` before displaying a map. The projection parameters are stored in a map projection structure that normally resides in the `UserData` property of a MATLAB axes, but you can use it directly for the projection computation.

The following commands create an empty map projection structure for a Sinusoidal projection, change the map origin, and fill in the rest of the structure fields with default property values. Just as you can change the property settings of a map axes with `setm`, you can use the entries of the map projection structure to control the projection properties.

```
mstruct = defaultm('sinusoid');
mstruct.origin = [0 180 0];
mstruct = defaultm(sinusoid(mstruct));
```

Having defined the map projection parameters, you can project the latitude and longitude vectors into the proper coordinates of the Sinusoidal projection and display the result using non-mapping, MATLAB commands.

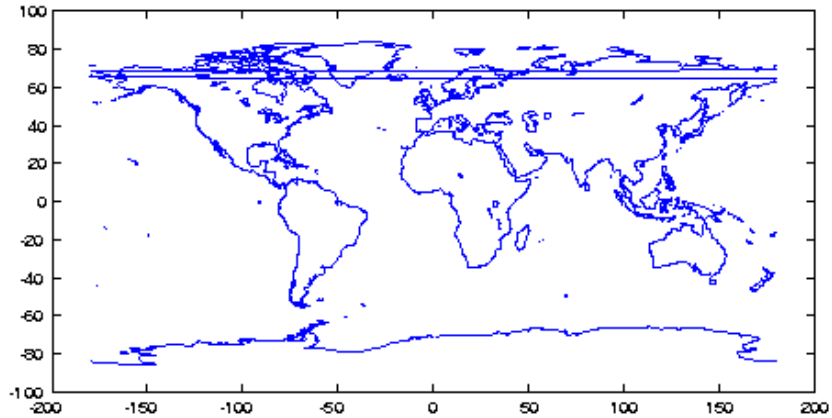
```
[x,y] = mfwdtran(mstruct,lat,long,[],'line');
plot(x,y); axis equal
set(gca,'XLim',[-3.5 3.5],'YLim',[-2 2])
```



For the transformation function, an empty matrix is used to take the place of altitude data, and the string `'line'` identifies the object type and allows the vector data to be clipped and trimmed. Notice that the International Date Line is now at the center of the map (180°), as specified earlier.

You can transform the projected x-y data back into the Greenwich geographic coordinates with the inverse transformation function. Plot the result:

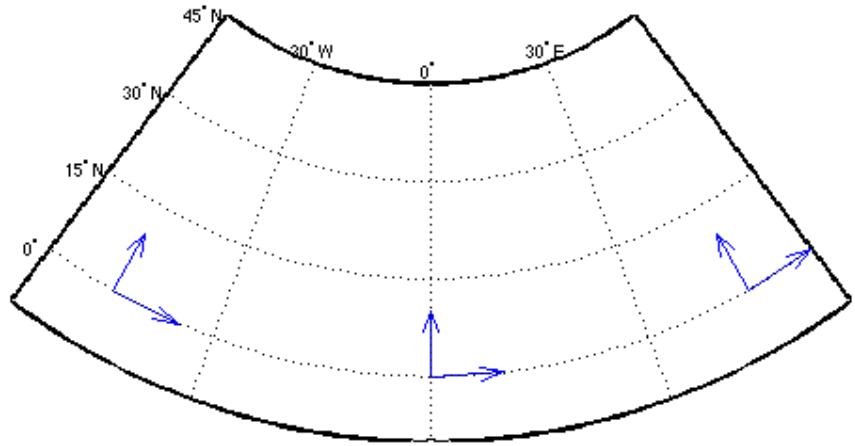
```
[lat2,long2] = minvtran(mstruct,x,y);
plot(long2,lat2); axis equal
set(gca,'XLim',[-200 200],'YLim',[-100 100])
```



What happened? Recall that the original data extended beyond 180° longitude. In the projection transformation process, longitude data outside `[-180 180]` degrees is projected back into this range since angles differing by 360° are geographically equivalent. The data from the inverse transformation process therefore jumps from 180° to -180°, as depicted by the straight horizontal lines in the figure above. The `smoothlong` function can be used to remove discontinuities in longitude.

In addition to projecting geographic positions into Cartesian coordinates, you can project angles between the sphere and the plane. For cylindrical projections, north maps to up on the y-axis, and east maps to right on the x-axis. This is not necessarily true of other projections. In conic projections, for example, north may be to the left or right of the y-axis, depending on the geographic coordinates.

What are the angles of north and east in projected coordinates for an equidistant conic? Geographic angles are measured clockwise from north, while projected angles are measured counterclockwise from the x-axis.




```
vfwdtran([0 0 0],[-45 0 45],[0 0 0])
ans =
       59.614           90       120.39

vfwdtran([0 0 0],[-45 0 45],[90 90 90])
ans =
      -30.385    0.0001931       30.386
```

Summary Guide

Literally an infinite number of possible map projections can be constructed. The Mapping Toolbox provides 60 different map projections. To view all the projections in the toolbox, use more than the command `maps`.

Cartographers choose a map projection by determining which property distortions they want to minimize or eliminate, if any. They also determine which of the three projection types (cylindrical, conic, or azimuthal) best suits their purpose, or they may decide upon special desired features, such as straight rhumb lines or great circles, true direction, or some other interesting trait. The following table lists the available projections along with their properties. Detailed information on all map projections provided by the Mapping Toolbox can be found in Appendix C, "Projections Reference".

Projection	Syntax	Type	Equal-Area	Conformal	Equidistant	Special Features
Balthasart	`balthsrt`	Cylindrical	•			
Behrmann	`behrmann`	Cylindrical	•			
Bolshoi Sovietskii Atlas Mira	`bsam`	Cylindrical				
Braun Perspective	`braun`	Cylindrical				
Cassini	`cassini`	Cylindrical			•	
Central	`ccylin`	Cylindrical				
Equal-Area Cylindrical	`eqacylin`	Cylindrical	•			
Equdistant Cylindrical	`eqdcylin`	Cylindrical			•	
Gall Isographic	`giso`	Cylindrical			•	
Gall Orthographic	`gortho`	Cylindrical	•			
Gall Stereographic	`gstereo`	Cylindrical				

Projection	Syntax	Type	Equal-Area	Conformal	Equidistant	Special Features
Lambert Equal-Area Cylindrical	`lambcyln`	Cylindrical	•			
Mercator	`mercator`	Cylindrical		•		1
Miller	`miller`	Cylindrical				
Plate Carree	`pcarree`	Cylindrical			•	
Trystan Edwards	`trystan`	Cylindrical	•			
Universal Transverse Mercator (UTM)	`utm`	Cylindrical		•		
Wetch	`wetch`	Cylindrical				
Apianus II	`apianus`	Pseudocylindrical				
Collignon	`collig`	Pseudocylindrical	•			
Craster Parabolic	`craster`	Pseudocylindrical	•			
Eckert I	`eckert1`	Pseudocylindrical				
Eckert II	`eckert2`	Pseudocylindrical	•			
Eckert III	`eckert3`	Pseudocylindrical				
Eckert IV	`eckert4`	Pseudocylindrical	•			
Eckert V	`eckert5`	Pseudocylindrical				
Eckert VI	`eckert6`	Pseudocylindrical	•			
Fournier	`fournier`	Pseudocylindrical	•			
Goode Homolosine	`goode`	Pseudocylindrical	•			
Hatano Assymetrical Equal-Area	`hatano`	Pseudocylindrical	•			
Kavraisky V	`kavrsky5`	Pseudocylindrical	•			

Projection	Syntax	Type	Equal-Area	Conformal	Equidistant	Special Features
Kavraisky VI	kavrsky6	Pseudocylindrical	•			
Loximuthal	loximuth	Pseudocylindrical				2
McBryde-Thomas Flat-Polar Parabolic	flatplrp	Pseudocylindrical	•			
McBryde-Thomas Flat-Polar Quartic	flatplrq	Pseudocylindrical	•			
McBryde-Thomas Flat-Polar Sinusoidal	flatplrs	Pseudocylindrical	•			
Mollweide	mollweid	Pseudocylindrical	•			
Putnins P5	putnins5	Pseudocylindrical				
Quartic Authalic	quartic	Pseudocylindrical	•			
Robinson	robinson	Pseudocylindrical				
Sinusoidal	sinusoid	Pseudocylindrical	•			
Tissot Modified Sinusoidal	modsine	Pseudocylindrical	•			
Wagner IV	wagner4	Pseudocylindrical	•			
Winkel I	winkel	Pseudocylindrical				
Albers Equal-Area Conic	eqaconic	Conic	•			
Equidistant Conic	eqdconic	Conic			•	
Lambert Conformal Conic	lambert	Conic		•		
Murdoch I Conic	murdoch1	Conic			•	3
Murdoch III Minimum Error Conic	murdoch3	Conic			•	3
Bonne	bonne	Pseudoconic	•			

Projection	Syntax	Type	Equal-Area	Conformal	Equidistant	Special Features
Werner	werner	Pseudoconic	•			
Polyconic	polycon	Polyconic				
Van Der Grinten I	vgrint1	Polyconic				
Breusing Harmonic Mean	breusing	Azimuthal				
Equidistant Azimuthal	eqdazim	Azimuthal			•	
Gnomonic	gnomonic	Azimuthal				4
Lambert Azimuthal Equal-Area	eqaazim	Azimuthal	•			
Orthographic	ortho	Azimuthal				
Stereographic	stereo	Azimuthal		•		5
Universal Polar Stereographic (UPS)	ups	Azimuthal		•		5
Vertical Perspective Azimuthal	vperspec	Azimuthal				
Wiechel	wiechel	Pseudoazimuthal	•			
Aitoff	aitoff	Modified Azimuthal				
Briesemeister	bries	Modified Azimuthal	•			
Hammer	hammer	Modified Azimuthal	•			
Globe	globe	Spherical	•	•	•	6

1. Straight rhumb lines.
2. Rhumb lines from central point are straight, true to scale, and correct in azimuth.
3. Correct total area.
4. Straight line great circles.
5. Great and small circles appear as circles or lines.
6. Three-dimensional display.

5

Mapping Applications

Introduction to Geographic Statistics and Navigation

The Mapping Toolbox provides a set of functions for applying mapping computations to geographic statistical analysis and navigation. The geographic statistics functions are necessary for proper analysis of geographical data as opposed to simple Cartesian (x-y) data. The navigation functions provide such useful navigational tasks as establishing position and planning routes or tracks.

Geographic Statistics

Most statistical functions implicitly assume that the data resides in a Cartesian coordinate system. Simply using geographic data in a Cartesian manner can give statistically inappropriate results. While the Cartesian assumption may be insignificant for small geographic regions, for larger areas it can lead to incorrect conclusions due to faulty distance measures, faulty area assumptions, or both. The Mapping Toolbox provides several functions for analyzing geographic data statistically.

Geographic Means

Consider the problem of calculating the mean position of a collection of geographic points. You might be inclined to simply take the arithmetical mean of the latitudes and longitudes using the standard MATLAB mean command, but this could be very misleading.

Take two points at the same latitude, 180° apart in longitude, for example (30°N,90°W) and (30°N,90°E). The *mean* latitude is (30+30)/2=30, which seems right. However, the mean longitude must be (90+(-90))/2=0. From the point of view at the Prime Meridian this seems fine, too. At the 0° meridian, each point has the same longitude difference from the mean. What about the point of view at the Date Line? At the 180° meridian, each point also has the same longitude difference. Why isn't 180° the mean longitude?

This problem is further complicated when some points are at different latitudes. Remember, a degree of longitude at the Arctic Circle covers a much smaller distance than a degree at the Equator.

In fact, in the first example, is 30°N the right mean latitude, either? The mean position of two points should be equidistant from those two points, and further, it should be the equidistant point that minimizes this distance. Is (30°N,0°) a reasonable mean point?

```
dist1 = distance(30,90,30,0)
dist1 =
  75.5225
dist2 = distance(30,-90,30,0)
dist2 =
  75.5225
```

Here is another point, call it (lat,lon), that is also equidistant from both points, and the distance is much shorter:

```
dist1 = distance(30,90,lat,lon)
dist1 =
  60.0000
dist2 = distance(30,-90,lat,lon)
dist2 =
  60.0000
```

What is this mystery point? The lat is 90°N, and any lon will do. The North Pole is the true geographic mean of these two points. Note that the great circle containing both points also runs through the North Pole, and a great circle represents the shortest path between two points.

The Mapping Toolbox includes the function meanm to determine the geographic mean of any number of points. This is accomplished through the three-dimensional vector addition of all the points. For example, try the following:

```
lats = [30 30];
longs = [-90 90];
[latbar,longbar] = meanm(lats,longs)
latbar =
  90
longbar =
  0
```

This is the answer we now expect. This geographic mean can result in one oddity; if the vectors all cancel each other, the mean is the center of the planet. In this case, the returned mean point is (NaN,NaN) and a warning is displayed. This phenomenon is highly improbable in *real* data, but can be easily constructed. For example, it occurs when all the points are equally spaced along a great circle. Try taking the geographic mean of (0°,0°), (0°,120°), and (0°,240°), which trisect the Equator.

Geographic Standard Deviation

As you might now expect, the Cartesian definition of standard deviation provided in the standard MATLAB function `std` is also inappropriate for most geographic data. Depending upon your purpose, you may want to use the separate geographic deviations for latitude and longitude provided by the function `stdm`, or the single standard distance provided in `stdist`. Both methods measure the deviation of points from the mean position calculated by `meanm`.

The Meaning of stdm

The `stdm` function handles the latitude and longitude deviations separately.

```
[latstd,lonstd] = stdm(lat,lon)
```

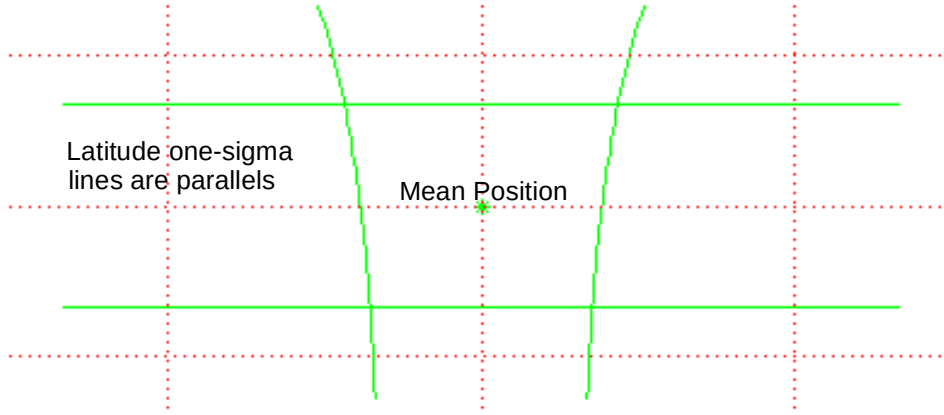
The function returns two deviations, one for latitudes and one for longitudes.

Latitude deviation is a straightforward standard deviation calculation from the mean latitude (mean parallel) returned by `meanm`. This is a reasonable measure for most cases, since a degree of latitude always has the same arc length.

Longitude deviation is another matter. Simple calculations based on sum-of-squares angular deviation from the mean longitude (mean meridian) are misleading. The arc length represented by a degree of longitude at extreme latitudes is significantly smaller than that at low latitudes.

The term *departure* is used to represent the arc length distance along a parallel of a point from a given meridian. For example, assuming a spherical planet, the departure of a degree of longitude at the Equator is a degree of arc length, but the departure of a degree of longitude at a latitude of 60° is one-half a degree of arc length. The `stdm` function calculates a sum-of-squares departure deviation from the mean meridian.

If you want to plot the one-sigma lines for `stdm`, the latitude sigma lines are parallels. However, the longitude sigma lines are not meridians; they are lines of constant departure from the mean parallel.



This handling of deviation has its problems. For example, its dependence upon the logic of the coordinate system can cause it to break down near the poles. For this reason, the standard distance provided by `stdist` is often a better measure of deviation. The `stdm` handling is useful for many applications, especially when the data is not global. For instance, these potential difficulties would not be a danger for data points confined to the country of Mexico.

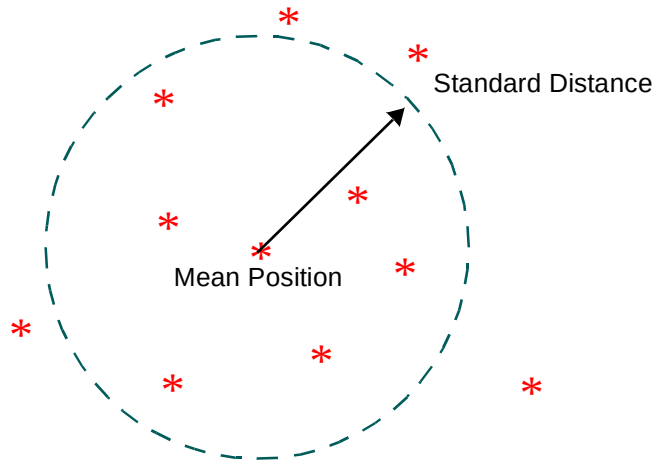
The Meaning of stdist

The standard distance of geographic data is a measure of the dispersion of the data in terms of its distance from the geographic mean. Among its advantages are its applicability anywhere on the globe and its single value:

```
dist = stdist(lat,lon)
```

In short, the standard distance is the average, norm, or *cubic-norm* of the distances of the data points in a great circle sense from the mean position. It is probably a superior measure to the two deviations returned by `stdm` except

when a particularly latitude- or longitude-dependent feature is under examination



Equal-Areas in Geographic Statistics

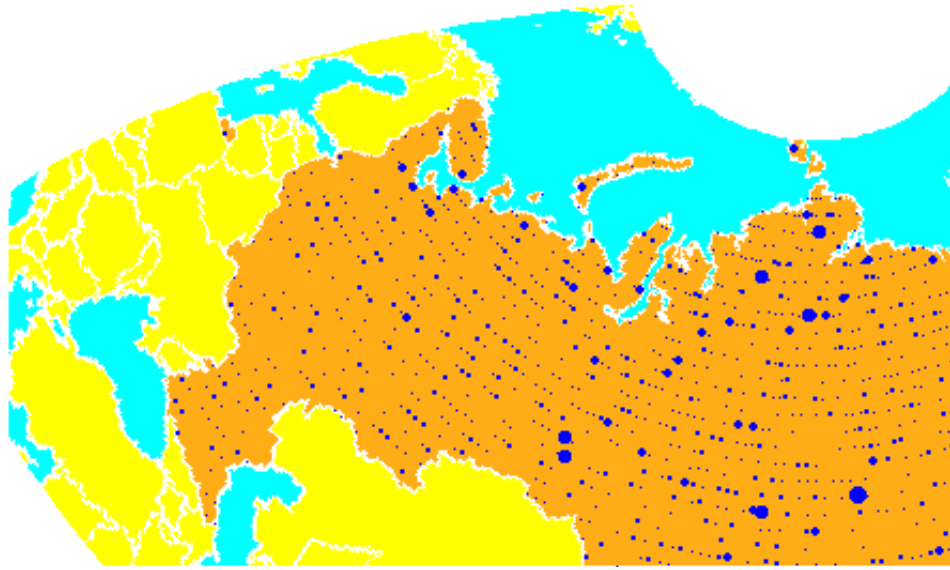
A common error in applying two-dimensional statistics to geographic data lies in ignoring equal-area treatment. It is often necessary to *bin-up* data to statistically analyze it. In a Cartesian plane, this is easily done by dividing the space into equal *x-y* squares. The geographic equivalent of this is to bin up the data in equal latitude-longitude *squares*. Since such squares at high latitudes cover smaller areas than their low-latitude counterparts, the observations in these regions are underemphasized. The result can be conclusions that are biased towards the Equator.

Geographic Histograms

The geographic histogram command `histr` allows you to display *binned-up* geographic observations. The `histr` command results in equirectangular binning. Each bin has the same angular measurement in both latitude and longitude, with a default measurement of 1 degree. The center latitudes and longitudes of the bins are returned, as well as the number of observations per bin:

```
[binlat,binlon,num] = histr(lats,lons)
```

As previously noted, these equirectangular bins result in counting bias towards the Equator. Here is a display of the one-degree-by-one-degree binning of approximately 5,000 random data points in Russia. The relative size of the circles indicates the number of observations per bin:



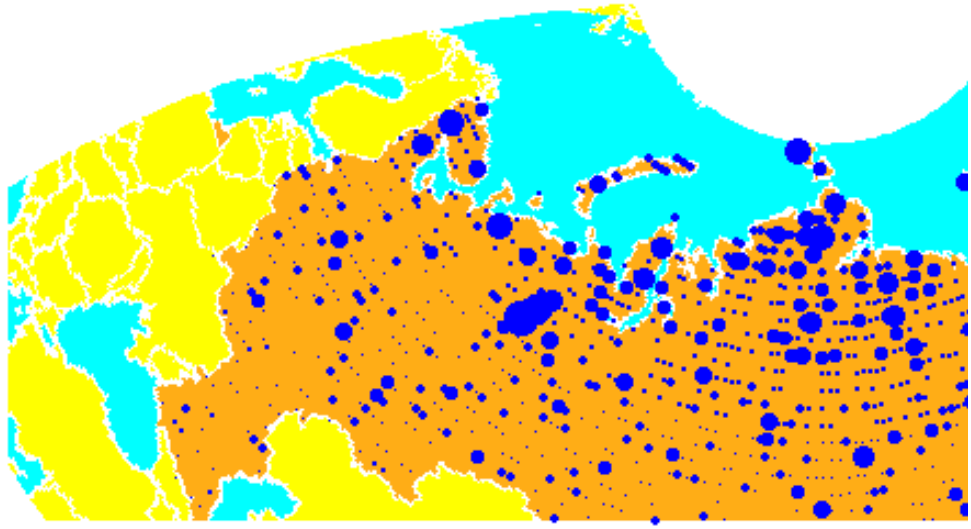
This is a portion of the whole map, displayed in an equal-area Bonne projection. The first step in creating data displays without area bias is to choose an equal-area projection. The proportionally sized symbols are a result of the specialized display function `scatterm`.

The area bias can be eliminated by adding a fourth output argument to `histr`, which will be used to weight each bin's observation by that bin's area:

```
[binlat,binlon,num,wnum] = histr(lats,lons)
```

The fourth output is the weighted observation count. Each bin's observation count is divided by its normalized area. Therefore, a high-latitude bin will have a larger weighted number than a low-latitude bin with the same number of

actual observations. The same data and bins look much different when they are area-weighted:



Notice that there are larger symbols to the north in this display. The previous display suggested that the data was relatively uniformly distributed. When equal-area considerations are included, it is clear that the data is skewed to the north. In fact, the data used is northerly skewed, but a simple equirectangular handling failed to demonstrate this.

The `histr` command, therefore, does provide for the display of area-weighted data. However, the actual bins used are of varying areas. Remember, the one-degree-by-one-degree bin near a pole is much smaller than its counterpart near the Equator.

The `hista` command provides for actual equal-area bins.

Converting to an Equal-Area Coordinate System

Finally, on the topic of equal-area data handling, the actual data itself can be converted to an equal-area coordinate system for analysis with other statistical functions. It is easy to convert a collection of geographic latitude-longitude points to an equal-area *x-y* Cartesian coordinate system. The `grn2eqa` command applies the same transformation used in calculating the Equal-Area Cylindrical projection:

```
[x,y] = grn2eqa(lat,lon)
```

For each Greenwich `lat - long` pair, an equal-area `x - y` is returned. The variables x and y can then be operated on under the equal-area assumption, using a variety of two-dimensional statistical techniques. Tools for such analysis can be found in the Statistics Toolbox and elsewhere. The results can then be converted back to Greenwich coordinates using the `eqa2grn` command:

```
[lat,lon] = eqa2grn(x,y)
```

Remember, when converting back and forth between systems, latitude corresponds to y and longitude corresponds to x.

Geographically Filtering Datasets

Often, a set of data will contain unwanted data mixed in with the desired values. For example, your data might include points for the entire United States, but you only want to work with those points falling in Alabama, or perhaps the dataset is *untidy*– out of 4,000 points, you notice that 3 or 4 obviously fall outside of reality (for example, one of your city-points is in the middle of the ocean). It can be quite a chore to look at each dataset element individually. Perhaps selecting a portion of the data is part of your analysis.

The `filterm` command works with a matrix map to filter a vector dataset. The form is of the following:

```
[flats,flons] = filterm(lats,lons,map,maplegend,allowed]
```

In short, each location defined by `lats` and `lons` is compared to the value at that point in `map`. If the value is `allowed`, that point is included in `flats` and `flons`.

The map might be a politically indexed map, and the allowed values might be the code or codes corresponding to the states or countries desired (e.g., Alabama). The map might also be a valued map, or a logical condition thereon, and the allowed value might be 1 for *true*. Here's what an example might look like using the `topo` map:

```
[flats,flons] = filterm(lats,lons,topo>0,topolegend,1)
```

The result would be those points corresponding to land.

Navigation

One field that makes extensive use of geographic information is navigational science and practice. The Mapping Toolbox includes a variety of specialized functions for navigation.

The practice of navigation includes a variety of tasks for the operation of both watercraft and aircraft. One task is establishing position, using known, fixed landmarks (piloting), employing the stars, Sun, and Moon (celestial navigation), utilizing technological fixing systems (radio and satellite navigation, including GPS), or deducing net movement from a past known position (dead reckoning).

Another navigational task involves planning a voyage or flight, which includes determining a short route (great circle approximation), weather avoidance (optimal track routing), and setting out a plan of intended movement (track laydown).

The Mapping Toolbox contains functions for some of these navigational activities.

Conventions for Navigational Functions

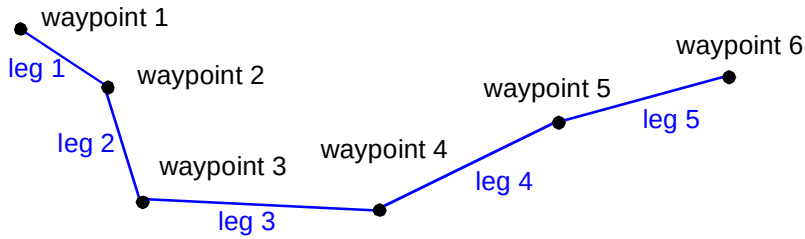
Units

The Mapping Toolbox is, in general, very flexible in allowing a variety of angular and distance measurement units. To make the strictly navigational functions easy to handle and to conform to common navigational practice, *for these specific functions*, angles are always in degrees, distances are always in nautical miles, and speeds are always in knots (nautical miles per hour). The functions for which these conventions hold are `dreckon`, `gcwaypts`, `legs`, and `navfix`. Related functions that do not carry this restriction include `rhxrh`, `scxsc`, `gcxgc`, `gcxsc`, `track`, `timezone`, and `crossfix`, because of their potential for application outside of navigation.

Navigational Track Format

Navigational track format requires column-vector variables for the latitudes and longitudes of track waypoints. A *waypoint* is a point through which a track passes, usually corresponding to a course (or speed) change. Navigational tracks are made up of the line segments connecting these waypoints, which are

called *legs*. In this format, therefore, *n* legs are described using *n+1* waypoints, because an endpoint for the final leg must be defined.



Here, five track legs require six waypoints. In navigational track format, the waypoints would be represented by two 6-by-1 vectors, one for the latitudes and one for the longitudes.

Fixing Position

The fundamental objective of navigation is to determine at a given moment how to proceed to your destination, avoiding hazards on the way. The first step in accomplishing this is to establish your current position. Early sailors kept within sight of land to facilitate this. Today, navigation within sight (or radar range) of land is called *piloting*. Positions are fixed by correlating the bearings and/or ranges of landmarks. In real-life piloting, all sighting bearings are treated as rhumb lines, while in fact they are actually great circles.

Over the distances involved with visual sightings (up to 20 or 30 nautical miles), this assumption causes no measurable error and it provides the significant advantage of allowing the navigator to plot all bearings as straight lines on a Mercator projection.

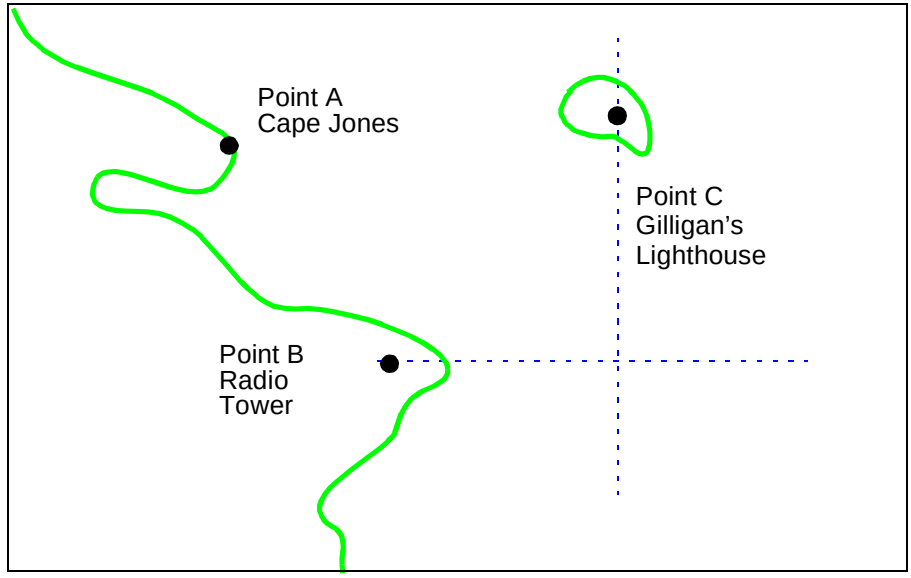
The Mercator was designed exactly for this purpose. Range circles, which might be determined with a radar, are assumed to plot as true circles on a Mercator chart. This allows the navigator to manually draw the range arc with a compass.

These assumptions also lead to computationally efficient methods for fixing positions with a computer. The Mapping Toolbox includes the `navfix` function, which mimics the manual plotting and fixing process using these assumptions.

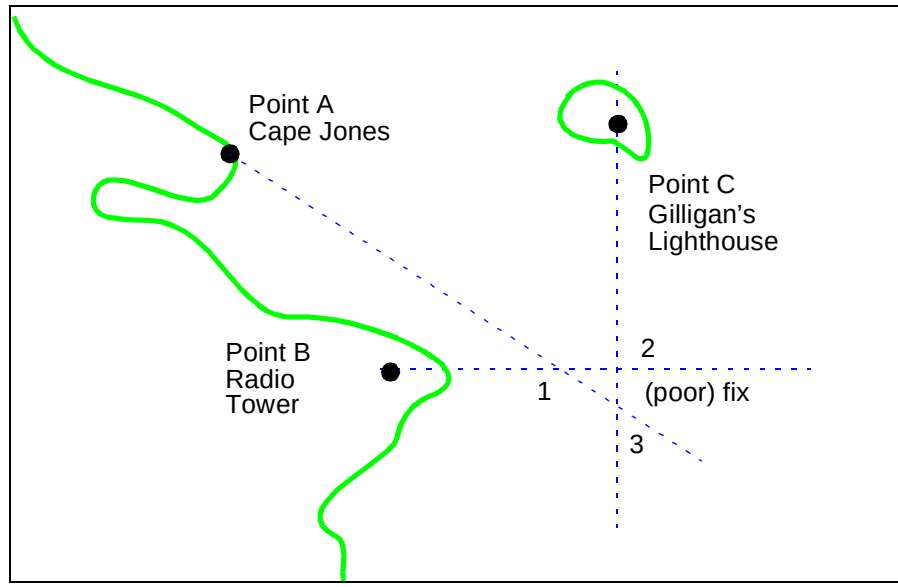
To obtain a good navigational fix, your relationship to at least three known points is considered necessary. A questionable or poor fix can be obtained with two known points.

Some Possible Situations

In this imaginary coastal region, you take a visual bearing on the radio tower of 270°. At the same time, Gilligan's Lighthouse bears 0°. If you plot a 90°-270° line through the radio tower, and a 0°-180° line through the lighthouse on your Mercator chart, the point at which the lines cross is a fix. Since you have used only two lines, however, its quality is questionable.



But wait; your port lookout says he took a bearing on Cape Jones of 300°. If that line exactly crosses the point of intersection of the first two lines, you will have a perfect fix.



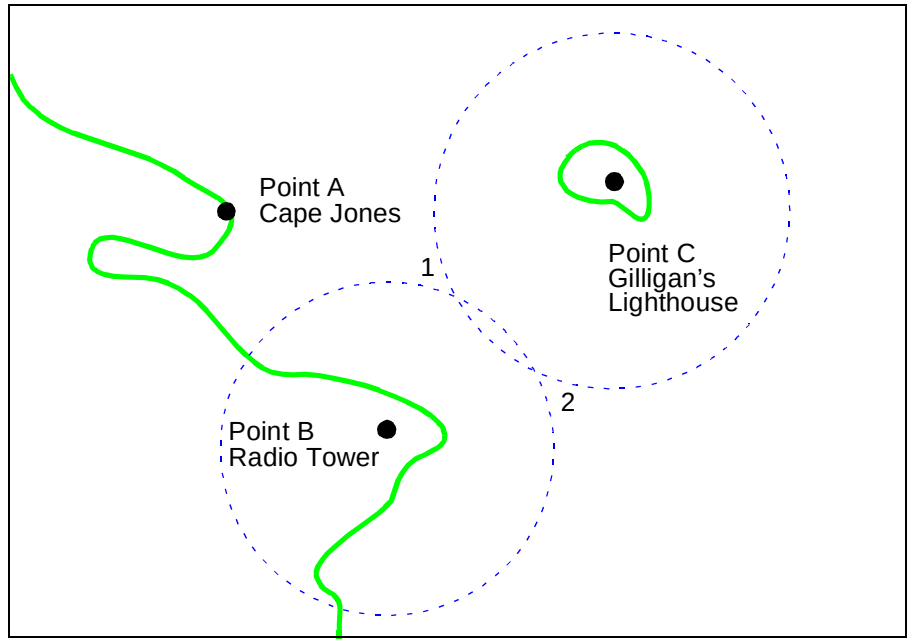
Whoops. What happened? Is your lookout in error? Possibly, but perhaps one or both of your bearings was slightly in error. This happens all the time. Which point, 1, 2, or 3, is correct? As far as you know, they are all equally valid.

In practice, the little triangle is plotted, and the fix position is taken as either the center of the triangle or the vertex closest to a danger (like shoal water). If the triangle is large, the quality is reported as *poor*, or even as *no fix*. If a fourth line of bearing is available, it can be plotted to try to resolve the ambiguity. When all three lines appear to cross at exactly the same point, the quality is reported as *excellent* or *perfect*.

Notice that three lines resulted in three intersection points. Four lines would return six intersection points. This is a case of combinatorial counting. Each intersection corresponds to choosing two lines to intersect from among *n* lines, so the total number of intersection points will be *n*-choose-2.

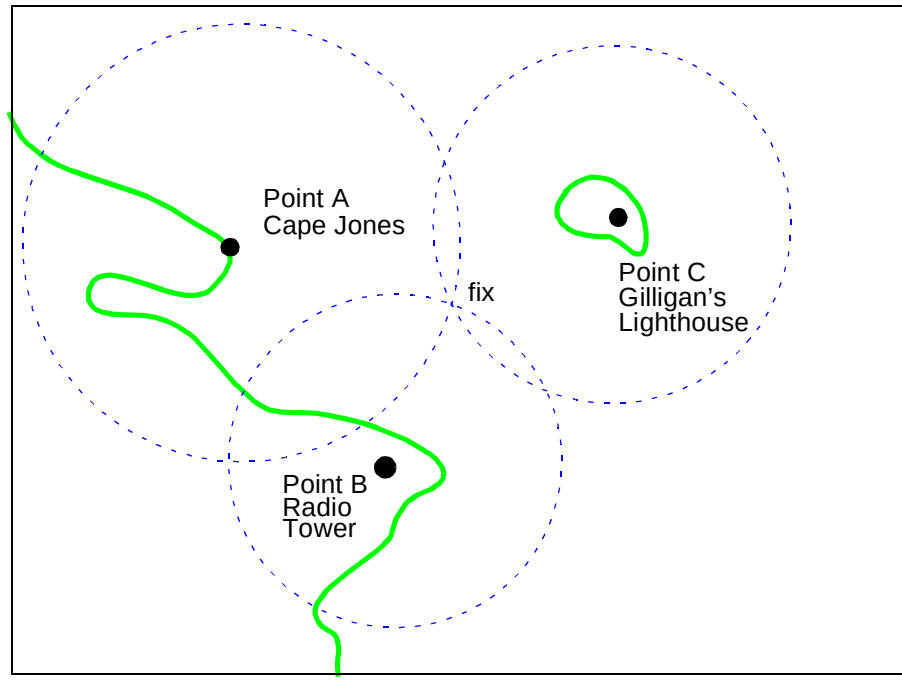
The next time you traverse these straits, it is a very foggy morning. You can't see any landmarks, but luckily, your navigational radar is operating. Each of these landmarks has a good radar signature, so you're not worried. You get a

range from the radio tower of 14 nautical miles and a range from the lighthouse of 15 nautical miles.



Now what? You took ranges from only two objects, and yet you have two possible positions. This ambiguity arises from the fact that circles can intersect twice.

Luckily, your radar watch reports that he has Cape Jones at 18 nautical miles. This should resolve everything.

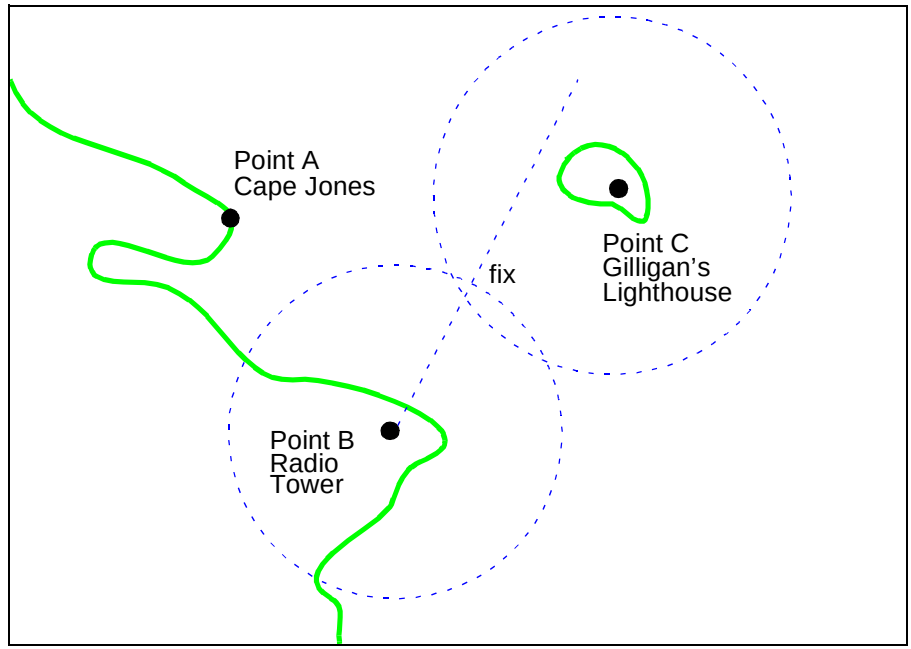


You were lucky this time. The third range resolved the ambiguity and gave you an excellent fix. Three intersections practically coincide. Sometimes the ambiguity is resolved, but the fix is still poor because the three closest intersections form a sort of circular triangle.

Sometimes the third range only adds to the confusion, either by bisecting the original two choices, or by failing to intersect one or both of the other arcs at all. In general, when *n* arcs are used, 2x(*n*-choose-2) possible intersections result. In this example, it is easy to tell which ones are *right*.

Bearing lines and arcs can be combined. If instead of reporting a third range, your radar watch had reported a bearing from the radar tower of 20°, the ambiguity could have also been resolved. Note, however, that in practice, lines of bearing for navigational fixing should only be taken visually, except in

desperation. A radar's beamwidth can be a degree or more, leading to uncertainty.



As you begin to wonder whether this manual plotting process could be automated, your first officer shows up on the bridge with a laptop and the Mapping Toolbox.

Using navfix

The navfix command can be used to determine the points of intersection between any number of lines and arcs. Be warned, however, that due to the combinatorial nature of this process, the computation time grows rapidly with the number of objects. To illustrate this function, assign positions to the landmarks. Point A, Cape Jones, is at (latA,lonA). Point B, the radio tower, is at (latB,lonB). Point C, Gilligan's Lighthouse, is at (latC,lonC).

For the bearing-lines-only example, the syntax is:

```
[latfix,lonfix] = navfix([latA latB latC],[lonA lonB lonC],...
                         [300 270 0])
```

This defines the three points and their bearings as taken *from the ship*. The outputs would look something like this, with actual numbers of course:

```
latfix =
  latfix1       NaN          % A intersecting B
  latfix2       NaN          % A intersecting C
  latfix3       NaN          % B intersecting C
lonfix =
  lonfix1       NaN          % A intersecting B
  lonfix2       NaN          % A intersecting C
  lonfix3       NaN          % B intersecting C
```

Notice that these are two-column matrices. The second column consists of NaNs because it is used only for the two-intersection ambiguity associated with arcs.

For the range-arcs-only example, the syntax is:

```
[latfix,lonfix] = navfix([latA latB latC],[lonA lonB lonC],...
                         [16 14 15],[0 0 0])
```

This defines the three points and their ranges as taken from the ship. The final argument indicates that the three cases are all ranges.

The outputs have the following form:

```
latfix =
  latfix11  latfix12          % A intersecting B
  latfix21  latfix22          % A intersecting C
  latfix31  latfix32          % B intersecting C
lonfix =
  lonfix11  lonfix12          % A intersecting B
  lonfix21  lonfix22          % A intersecting C
  lonfix31  lonfix32          % B intersecting C
```

Here, the second column is used, because each pair of arcs has two potential intersections.

For the bearings and ranges example, the syntax requires the final input to indicate which objects are lines of bearing (indicated with a 1) and which are range arcs (indicated with a 0):

```
[latfix,lonfix] = navfix([latB latB latC],[lonB lonB lonC],...
                         [20 14 15],[1 0 0])
```

The resulting output is mixed:

```
latfix =
  latfix11       NaN          % Line B intersecting Arc B
  latfix21  latfix22          % Line B intersecting Arc C
  latfix31  latfix32          % Arc B intersecting Arc C
lonfix =
  lonfix11       NaN          % Line B intersecting Arc B
  lonfix21  lonfix22          % Line B intersecting Arc C
  lonfix31  lonfix32          % Arc B intersecting Arc C
```

Only one intersection is returned for the line from B with the arc about B, since the line originates inside the circle and intersects it once. The same line intersects the other circle twice, and hence it returns two points. The two circles taken together also return two points.

Usually, you have an idea as to where you are before you take the fix. For example, you might have a dead reckoning position for the time of the fix (see below). If you provide `navfix` with this estimated position, it will choose from each pair of ambiguous intersections the point closest to the estimate. Here's what it might look like:

```
[latfix,lonfix] = navfix([latB latB latC],[lonB lonB lonC],...
                         [20 14 15],[1 0 0],drlat,drlon)
latfix =
  latfix11                    % the only point
  latfix21                    % the closer point
  latfix31                    % the closer point
lonfix =
  lonfix11                    % the only point
  lonfix21                    % the closer point
  lonfix31                    % the closer point
```

A Numerical Example of Using navfix

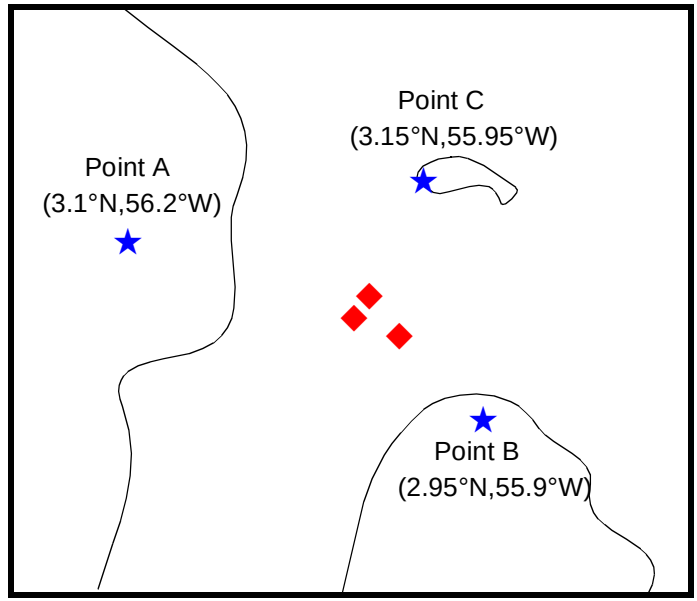
For a numerical example, define some specific points in the middle of the Atlantic Ocean. These are strictly arbitrary; perhaps they correspond to points in Atlantis:

```
lata = 3.1;  lona = -56.2;
latb = 2.95; lonb = -55.9;
latc = 3.15; lonc = -55.95;
```

Plot them on a Mercator projection:

```
axesm('MapProjection','mercator','Frame','on',...
  'MapLatLimit',[2.8 3.3],'MapLonLimit',[-55.8 -56.3])
plotm([lata latb latc],[lona lonb lonc],...
  'LineStyle','none','Marker','pentagram',...
  'MarkerEdgeColor','b','MarkerFaceColor','b',...
  'MarkerSize',12)
```

Here's what it looks like (the labeling and imaginary coastlines are added after the fact for illustration):

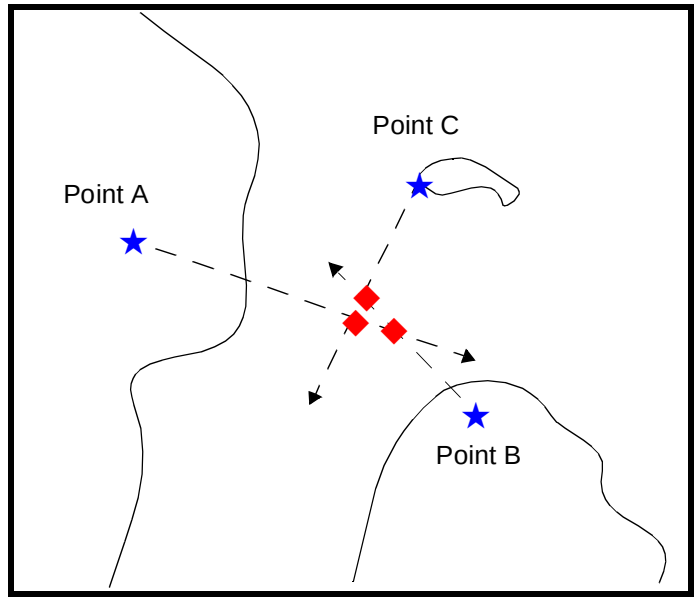


Take three visual bearings: Point A bears 289°, Point B bears 135°, and Point C bears 026.5°. Calculate the intersections:

```
[newlat,newlong] = navfix([lata latb latc],[lona lonb lonc],...
                          [289 135 26.5],[1 1 1])
newlat =
   3.0214       NaN
   3.0340       NaN
   3.0499       NaN
newlong =
   -55.9715       NaN
   -56.0079       NaN
   -56.0000       NaN
```

Add the bearing lines and intersection points to the map:

```
plotm(newlat,newlong,'LineStyle','none',...
  'Marker','diamond','MarkerEdgeColor','r',...
  'MarkerFaceColor','r','MarkerSize',9)
```

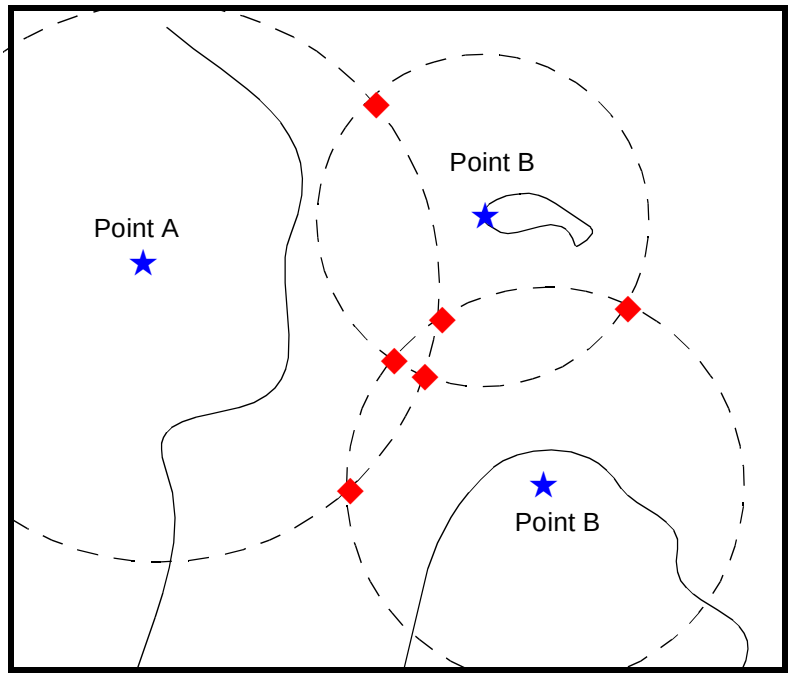


Notice that each pair of objects results in only one intersection, since all are lines of bearing.

What if instead, you had ranges from the three points, A, B, and C, of 13nm, 9nm, and 7.5nm, respectively.

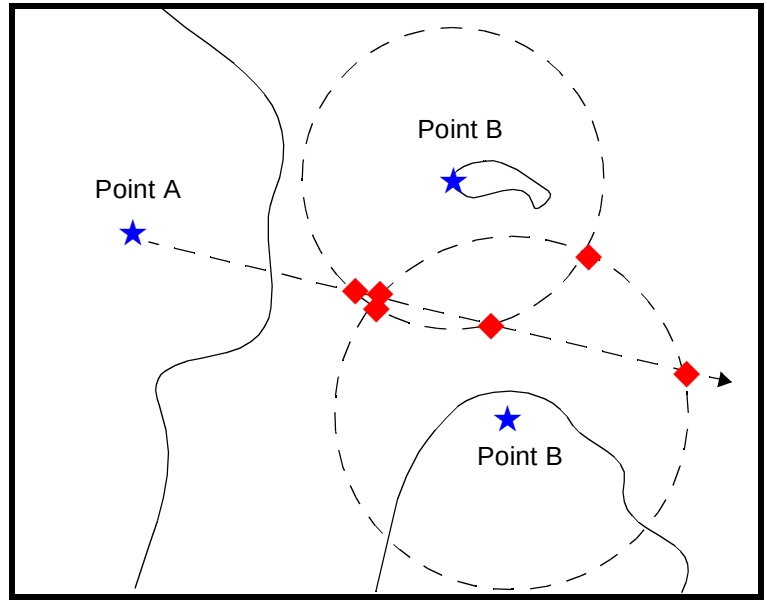
```
[newlat,newlong] = navfix([lata latb latc],[lona lonb lonc],...
                          [13 9 7.5],[0 0 0])
newlat =
   3.0739    2.9434
   3.2413    3.0329
   3.0443    3.0880
newlong =
  -55.9846  -56.0501
  -56.0355  -55.9937
  -56.0168  -55.8413
```

Here's what these points look like:



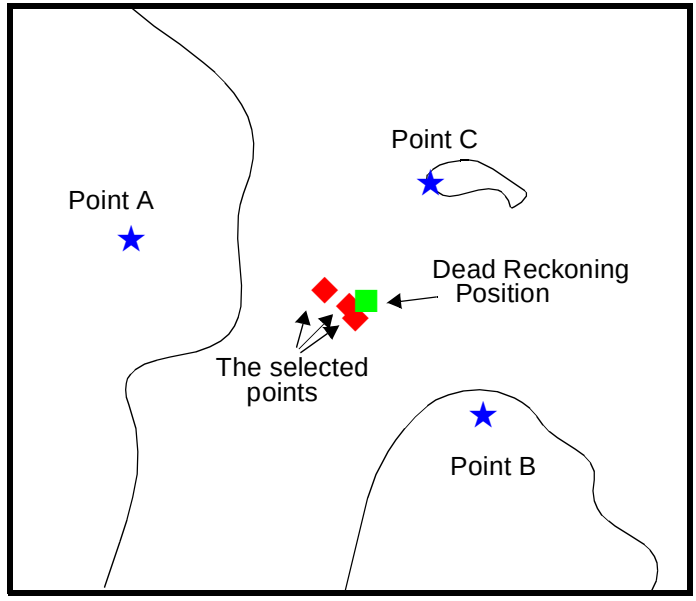
Three of these points look reasonable, three do not. What if, instead of a range from Point A, you had a bearing to it of 284°?

```
[newlat,newlong] = navfix([lata latb latc],[lona lonb lonc],...
                          [284 9 7.5],[1 0 0])
newlat =
  3.0526    2.9892
  3.0592    3.0295
  3.0443    3.0880
newlong =
  -56.0096  -55.7550
  -56.0360  -55.9168
  -56.0168  -55.8413
```



Again, visual inspection of the results indicates which three of the six possible points seem like *reasonable* positions. When using the dead reckoning position (3.05°N,56.0°W), the closer, more reasonable candidate from each pair of intersecting objects is chosen:

```
drlat = 3.05; drlon = -56;
[newlat,newlong] = navfix([lata latb latc],[lona lonb lonc],...
                          [284 9 7.5],[1 0 0],drlat,drlon)
newlat =
   3.0526
   3.0592
   3.0443
newlong =
   -56.0096
   -56.0360
   -56.0168
```



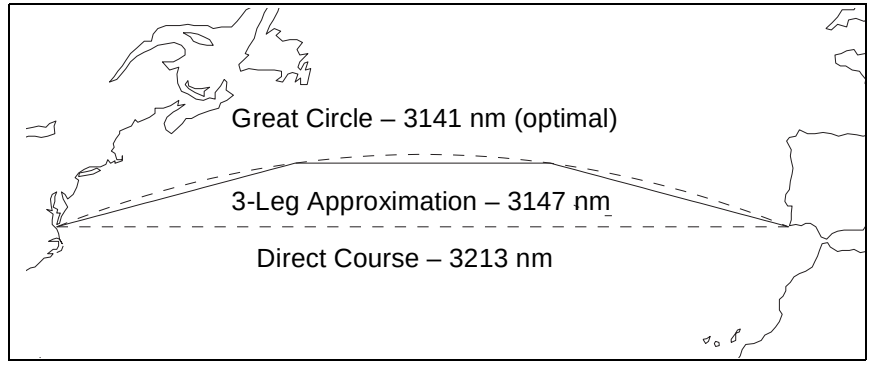
Planning

We know that the shortest path between two geographic points is a great circle. Sailors and aviators are interested in minimizing distance travelled, and hence delay. We also know that the rhumb line is a path of constant heading, the *natural* means of travelling. In general, to follow a great circle path, one would have to continuously alter course. This is impractical. However, a great circle path can be approximated by rhumb line segments so that the added distance is minor and the number of course changes minimal.

Surprisingly, very few rhumb line *track legs* are required to closely approximate the distance of the great circle path.

Consider the voyage from Norfolk, Virginia (37°N,76°W), to Cape St. Vincent, Portugal (37°N,9°W), one of the most heavily trafficked routes in the Atlantic. A due-east rhumb line track is 3,213 nautical miles in length, while the optimal great circle distance is 3,141 nautical miles.

Although the rhumb line path is only a little more than 2% longer, this is an additional 72 miles over the course of the trip. For a 12-knot tanker, this results in a 6-hour delay, and in shipping, time is money. If just three rhumb line segments are used to approximate the great circle, the total distance of the trip is 3,147 nautical miles. Our tanker would suffer only a half-hour delay over optimal.



The Mapping Toolbox provides the function gcwaypts to quickly calculate waypoints in navigation track format in order to approximate a great circle with rhumb line segments. The syntax is simple:

```
[latpts,lonpts] = gcwaypts(lat1,lon1,lat2,lon2,numlegs)
```

All of the inputs for this command are scalars. The numlegs input is the number of equal length legs desired, which is 10 by default. The outputs are column vectors representing waypoints in navigational track format. The size of each of these vectors will be [(numlegs+1) 1]. Here are the points for the example illustrated above:

```
[latpts,lonpts] = gcwaypts(37,-76,37,-9,3)
latpts =
  37.0000
  41.5076
  41.5076
37.0000
lonpts =
  -76.0000
  -54.1777
  -30.8223
  -9.0000
```

These points represent waypoints along the great circle between which the approximating path follows rhumb lines. Four points are needed for three legs, because the final point at Cape St. Vincent must be included.

Track Laydown – Displaying Navigational Tracks

Navigational tracks are most useful when graphically displayed. Traditionally, the navigator identifies and plots waypoints on a Mercator projection and then connects them with a straightedge, which on this projection results in rhumb line tracks. In the previous example, waypoints were chosen to approximate a great circle route, but they may be selected for a variety of other reasons.

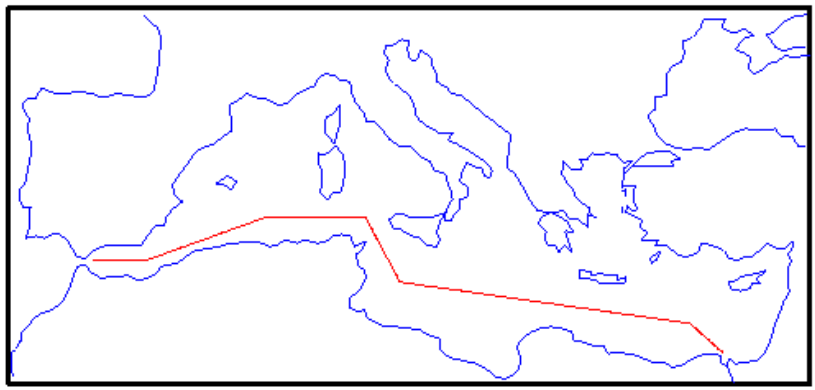
Let's say that after arriving at Cape St. Vincent, your tanker must traverse the Straits of Gibraltar and then travel on to Port Said, the northern terminus of the Suez Canal. On the scale of the Mediterranean Sea, following great circle paths is of little concern compared to ensuring that the many straits and passages are safely transited. The navigator selects appropriate waypoints and plots them.

To do this with the Mapping Toolbox, you can display a map axes with a Mercator projection, select appropriate map latitude and longitude limits to isolate the area of interest, plot coastline data, and interactively mouse-select the waypoints with the `inputm` command. The `track` command will generate points to connect these waypoints, which can then be displayed with `plotm`.

For illustration, assume that the waypoints are known (or were gathered using `inputm`). See the *Mapping Toolbox Reference Guide* or "Interacting with Displayed Maps" in Chapter 2 to learn about using `inputm`.

```
waypoints = [36 -5; 36 -2; 38 5; 38 11; 35 13; 33 30; 31.5 32]
waypoints =
  36.0000   -5.0000
  36.0000   -2.0000
  38.0000    5.0000
  38.0000   11.0000
  35.0000   13.0000
  33.0000   30.0000
  31.5000   32.0000

load coast
axesm('MapProjection','mercator',...
'MapLatLimit',[30 47],'MapLonLimit',[-10 37])
framem
plotm(lat,long)

[lttrk,lntrk] = track(waypoints);
plotm(lttrk,lntrk,'r')
```

Although these track segments are straight lines on the Mercator projection, they will be curves on others:



The segments of a track like this are called *legs*. Each of these legs can be described in terms of course and distance. The function legs will take the waypoints in navigational track format and return the courses and distances required for each leg. Remember, the order of the points in this format determines the direction of travel. Courses are therefore calculated from each waypoint to its successor, not vice-versa.

```
[courses,distances] = legs(waypoints)
courses =
  90.0000
  70.3132
  90.0000
  151.8186
  98.0776
  131.5684
distances =
  145.6231
  356.2117
  283.6839
  204.2073
  854.0092
  135.6415
```

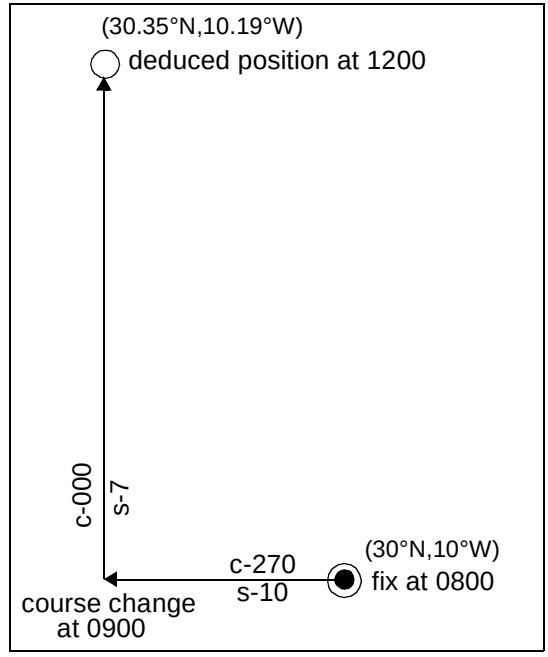
Since this is a navigation function, the courses are all in degrees and the distances are in nautical miles. From these distances, speeds required to arrive

at Port Said at a given time can be calculated. Southbound traffic is allowed to enter the canal only once per day, so this information might be economically significant, since unnecessarily high speeds can lead to high fuel costs.

Dead Reckoning

When sailors first ventured out of sight of land, they faced a daunting dilemma. How could they find their way home if they didn't know where they were? The practice of *dead reckoning* is an attempt to deal with this problem. The term is derived from *deduced reckoning*.

Briefly, dead reckoning is vector addition plotted on a chart. For example, if you had a fix at (30°N,10°W) at 0800, and you proceeded due west for 1 hour at 10 knots, and then you turned north and sailed for 3 hours at 7 knots, you should be at (30.35°N,10.19°W) at 1200. All's well, ring eight bells on time, Mr. Bowditch, right?



However, Mr. Bowditch *shoots the sun* at local apparent noon and discovers that the ship's latitude is actually 30.29°N. What's worse, he lives before the invention of a reliable chronometer, and so he cannot calculate his longitude at all from this sighting. What happened?

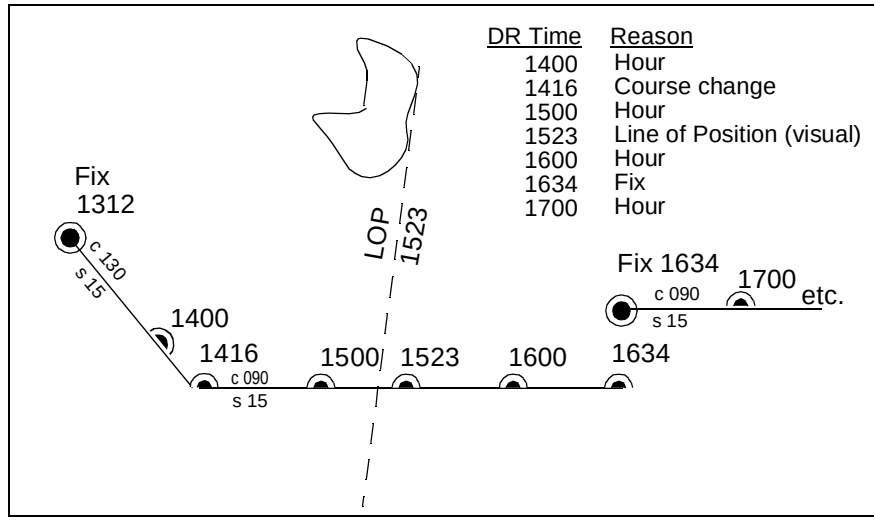
Leaving aside the difficulties in speed determination and the need to tack off course, even modern craft have to contend with winds and currents. However, despite these limitations, dead reckoning is still used for determining position between fixes and for forecasting future positions. This is because dead reckoning provides a certainty of assumptions that estimations of wind and current drift cannot.

When navigators establish a fix from some source, be it from piloting, celestial, or satellite observations, they plot a dead reckoning (DR) track, which is a plot of the intended positions of the ship forward in time. In practice, dead reckoning is usually plotted for 3 hours in advance, or for the time period covered by the next three expected fixes. In open ocean conditions, hourly fixes are sufficient; in coastal pilotage, three-minute fixes are common.

Specific DR positions, which are sometimes called *DR*s, are plotted according to the *Rules of DR*:

- DR at every course change
- DR at every speed change
- DR every hour on the hour
- DR every time a fix or running fix is obtained
- DR three hours ahead or for the next three expected fixes
- DR for every line of position (LOP), either visual or celestial

For example, the navigator plots these DRs:



Notice that the 1523 DR does not coincide with the LOP at 1523. Although note is taken of this variance, one line is insufficient to calculate a new fix.

The Mapping Toolbox includes the function `dreckon`, which calculates the DR positions for a given set of courses and speeds. The function provides DR positions for the first three rules of dead reckoning. The approach is to provide a set of waypoints in navigational track format corresponding to the plan of intended movement.

The time of the initial waypoint, or fix, is also needed, as well as the speeds to be employed along each leg. Alternatively, a set of speeds and the times for which each speed will apply can be provided. `dreckon` will return the positions and times required of these DRs:

- `dreckon` calculates the times for positions of each course change, which will occur at the waypoints
- `dreckon` calculates the positions for each whole hour
- if times are provided for speed changes, `dreckon` calculates positions for these times, if they do not occur at course changes

Imagine you have a fix at midnight at the point (10°N,0°):

```
waypoints(1,:) = [10 0]; fixtime = 0;
```

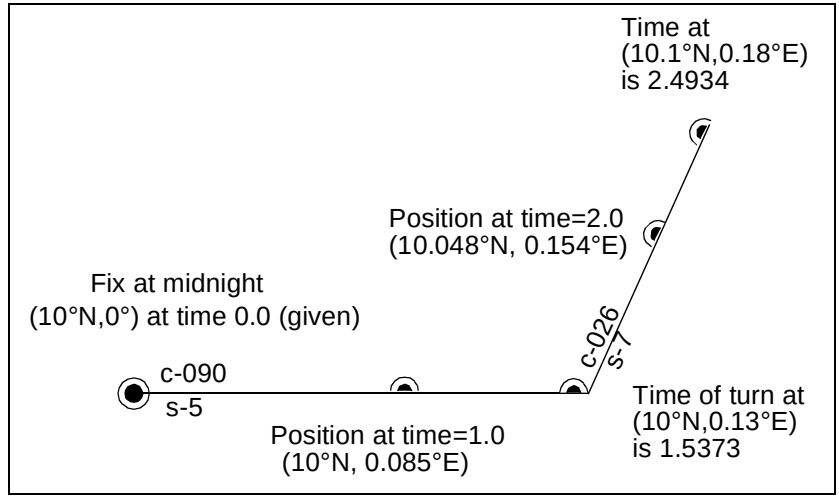
You intend to travel east and alter course at the point (10°N,0.13°E) and head for the point (10.1°N,0.18°E). On the first leg, you will travel at 5 knots, and on the second leg you will speed up to 7 knots.

```
waypoints(2,:) = [10 .13];
waypoints(3,:) = [10.1 .18];
speeds = [5;7];
```

To determine the DR points and times for this plan, use dreckon:

```
[drlat,drlon,drtime] = dreckon(waypoints,fixtime,speeds);
[drlat drlon drtime]
ans =
  10.0000    0.0846    1.0000     % Position at 1 am
  10.0000    0.1301    1.5373     % Time of course change
  10.0484    0.1543    2.0000     % Position at 2 am
  10.1001    0.1801    2.4934     % Time at final waypoint
```

Here is an illustration of this track and its DR points:

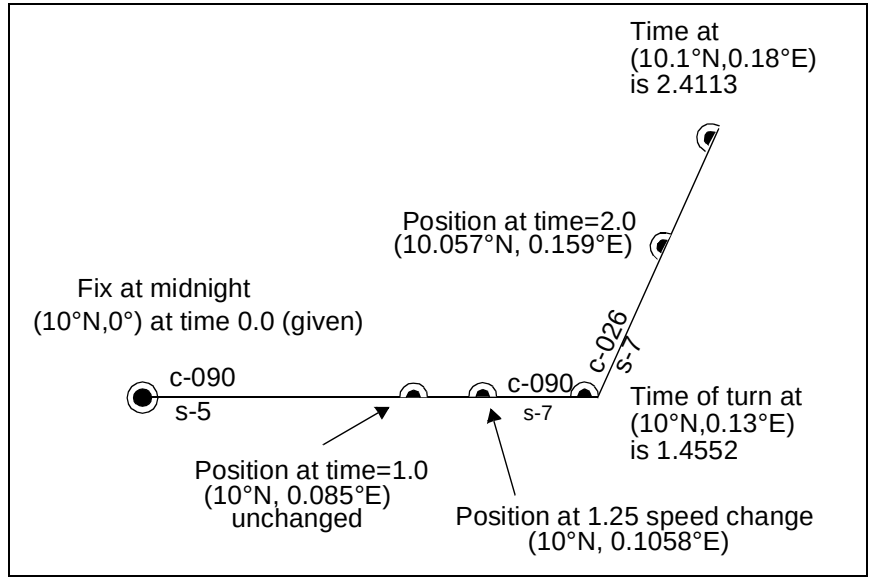


However, you would like to get to the final point a little earlier to make a rendezvous. You decide to recalculate your DRs based on speeding up to 7 knots a little earlier than planned. The first calculation tells you that you were going to increase speed at the turn, which would occur at a time 1.5373 hours after midnight, or 1:32 a.m. (at time 0132 in navigational time format). What time would you reach the rendezvous if you increased your speed to 7 knots at 1:15 a.m. (0115, or 1.25 hours after midnight)?

To indicate times for speed changes, another input is required, providing a time interval after the fix time at which each ordered speed is to end. The first speed, 5 knots, is to end 1.25 hours after midnight. Since you don't know when the rendezvous will be made under these circumstances, set the time for the second speed, 7 knots, to end at infinity. No DRs will be returned past the last waypoint.

```
spdtimes = [1.25; inf];
[drlat,drlon,drtime] = dreckon(waypoints,fixtime,...
                               speeds,spdtimes);
[drlat,drlon,drtime]
ans =
  10.0000    0.0846    1.0000   % Position at 1 am
  10.0000    0.1058    1.2500   % Position at speed change
  10.0000    0.1301    1.4552   % Time of course change
  10.0570    0.1586    2.0000   % Position at 2 am
  10.1001    0.1801    2.4113   % Time at final waypoint
```

This following illustration shows the difference:



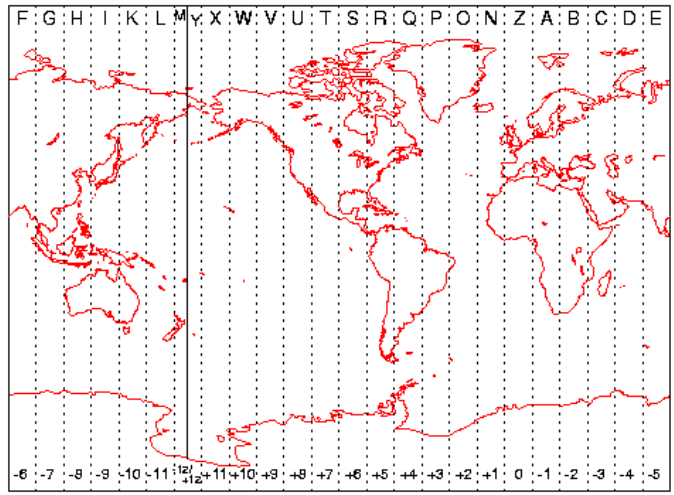
The times at planned positions after the speed change are a little earlier; the position at the known time (2 a.m) is a little farther along. With this plan, you will arrive at the rendezvous about 4-1/2 minutes earlier, so you may want to consider a more authoritative speed change.

Time Zones

Early in this manual, we gave an example of a time zone calculation. The `timezone` function returns a navigational time zone, that is, one based solely on longitude with no regard for statutory divisions. So, for example, Chicago, Illinois, lies in the statutory U.S. Central time zone, which has irregular boundaries devised for political or convenience reasons. However, from a navigational standpoint, Chicago's longitude places it in the *S* (Sierra) time zone. The zone's *description* is +6, which indicates that 6 hours must be added to local time to get Greenwich, or *Z* (Zulu) time. So, if it is noon, standard time in Chicago, it is 12+6, or 6 p.m. at Greenwich.

Each navigational time zone is 15° of longitude in width and has a distinct description and designating letter. The exceptions to this are the two zones on either side of the Date Line, *M* and *Y* (Mike and Yankee). These zones are only

7-1/2° wide, since on one side of the Date Line, the description is +12, and on the other, it is -12. Navigational time zones are very important for celestial navigation calculations. Although the Mapping Toolbox does not contain any functions designed specifically for celestial navigation, a simple example can be devised.



It is possible with a sextant to determine *local apparent noon*. This is the moment when the Sun is at its zenith from your point of view. At the exact center longitude of a time zone, the phenomenon occurs exactly at noon, local time. Since the Sun traverses a 15° time zone in 1 hour, it crosses one degree every 4 minutes. So, if you observe local apparent noon at 11:54, you must be 1.5° east of your center longitude.

You must know what time zone you are in before you can even attempt a fix. This concept has been understood since the spherical nature of the Earth was first accepted, but early sailors had no ability to keep accurate time on ship, and so were unable to determine their longitude. The engineering exploitation of this scientific truism had to wait until the invention of the chronometer.

The `timezone` function is quite simple. It returns the description, `zd`, an integer for use in calculations, a string `zltr` of the zone designator, and a string fully naming the zone. For example, the information for a longitude 123°E is the following:

```
[zd,zltr,zone] = timezone(123)
zd =
   -8
zltr =
H
zone =
   -8 H
```

Returning to our simple celestial navigation example, the center longitude of this zone is:

```
-(zd*15)
ans =
   120
```

This means that at our longitude, 123°E, we should experience local apparent noon at 11:48 a.m., 12 minutes early.

Time Notation

Navigational practice has its own peculiar notation for times. Time labels on navigation plots are always in a special format. Times are given in four digits, hours from 00 to 23 followed by minutes from 00 to 59. So, one minute before noon is 1159, or 1159Z or 1159Q, etc., based on time zone. Similarly, one minute after midnight is 0001. When more precision is required, the seconds are rounded to the nearest quarter minute and zero, one, two or three apostrophes are suffixed to the time, one for each 15-second block. So, 15 seconds before noon would be 1159'"; 14 seconds before noon would have the exact same notation.

The Mapping Toolbox includes the function `time2str` that returns a string in a variety of formats corresponding to a given time. These strings can then be plotted on map displays as desired. Two other clock formats are also allowed – the 12-hour and the 24-hour digital clock readouts. Consider some string notations for the time 13.21 hours after midnight. The default 24-hour clock is:

```
time2str(13.21)
ans =
13:12:36
```

The 12-hour clock reads:

```
time2str(13.21,'12')
ans =
01:12:36 PM
```

And the navigation format for this time is:

```
time2str(13.21,'nav')
ans =
1312''
```

Each of these can be rounded to the nearest minute with the third argument `'hm'` (for hours-minutes – the default is `'hms'`):

```
time2str(13.21,'nav','hm')
ans =
1313
```

6

GUI Tools

An Overview

The Mapping Toolbox includes a number of graphical user interface (GUI) tools that provide an alternative to entering toolbox commands from the MATLAB command window. The Mapping Toolbox GUIs cover most, but not all, aspects of the mapping process.

The Mapping Toolbox GUI system is a collection of independent GUIs, most of which can be activated from the command line and/or from a map display window at any time during a MATLAB session. The Mapping Toolbox also provides `maptool`, a comprehensive tool that allows access to most of the toolbox's independent GUIs through a system of figure window menus and buttons.

Most of the Mapping Toolbox GUIs include a **Help** button used to enter help mode for that particular GUI. Once in help mode, pressing any enabled control will display the help text for that control in a separate help box. Pressing the **Done** button will close the help box and end help mode.

This chapter is a short tutorial on several of the Mapping Toolbox GUI tools. For a more detailed description of each tool, consult the "GUI Reference" section of the *Mapping Toolbox Reference Guide*.

The Complete GUI Environment: maptool

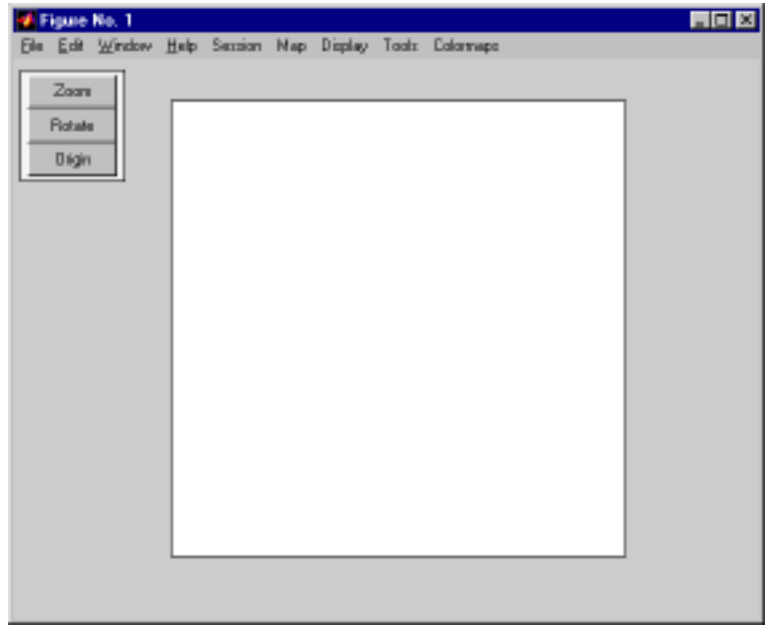
In this section, `maptool` is used to access many of the Mapping Toolbox GUIs. It should be noted, however, that most of the Mapping Toolbox GUIs can be activated independently at any time during a mapping session without the use of `maptool`. The "GUI Reference" section of the *Mapping Toolbox Reference Guide* provides details on the methods for activating each tool.

Starting maptool

You can start `maptool` in one of three ways:

```
maptool
maptool MapProjectionName
maptool('MapPropertyName','MapPropertyValue')
```

If a figure does not already exist, starting maptool creates a new figure window with additional pull-down menus and mouse buttons. If a figure window already exists, starting maptool adds the menus and mouse buttons to the existing figure.



If an initial map projection is not provided at the command line, the **Projection Control** dialog box will appear, with the Robinson projection as the default. This dialog is used to define the map axes settings. All mapping functions activated through maptool will be associated with this map axes. If maptool is applied to a figure that already has a map axes, the menus and mouse buttons are simply added to the existing figure window, and the **Projection Control** box does not appear.

In the upper left corner of the figure is a set of three mouse tool buttons that allow interactive display manipulation.

The Menus

There are five pull-down menus associated with `maptool`.

- The **Session** menu manipulates the workspace and sets the figure window renderer for the `maptool` session.
- The **Map** menu command activates GUIs that are used to project objects onto the map axes.
- The **Display** menu edits map axes properties, manipulates display settings, calculates and displays navigational tracks, small circles, surface distances and map distortion, and provides print preview.
- The **Tools** menu provides a variety of miscellaneous mapping tools, including means to hide and delete mapped objects, and adjust axes display properties.
- The **Colormaps** menu allows for manipulation of the colormap for the current figure.

The Mouse Tool Buttons

There are three mouse tool buttons associated with `maptool`.

- The **Zoom** button toggles *Zoom* mode on and off. Zoom mode zooms in on a two-dimensional map display.
- The **Rotate** button toggles *Rotate 3-D* mode on and off. Rotate 3-D interactively rotates the view of a three-dimensional plot.
- The **Origin** button toggles *Origin* mode on and off. Origin mode interactively modifies the map origin.

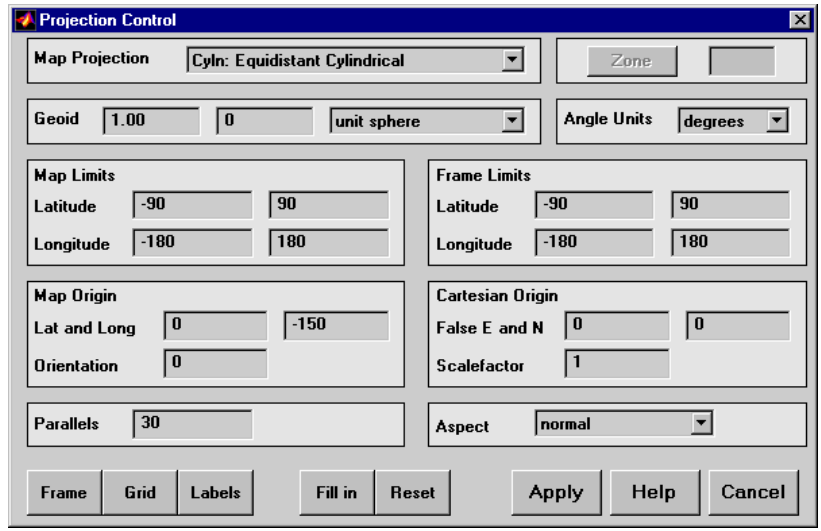
For more details on the `maptool` menus and buttons, see the "GUI Reference" section of the *Mapping Toolbox Reference Guide.*

Example One: Creating a World Map

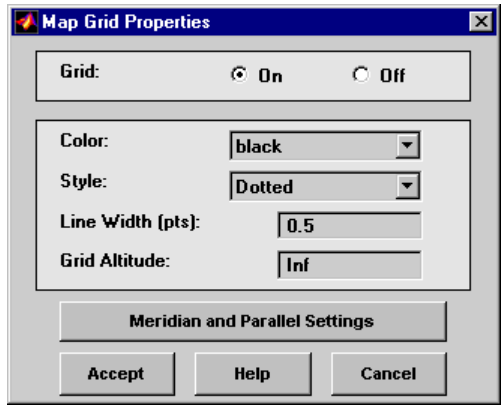
The Mapping Toolbox comes with world atlas data in the `worldlo` workspace. This data will be used in the following example to create a simple line map of the world. The first step is to create a map axes. From the MATLAB command prompt, type:

```
maptool
```

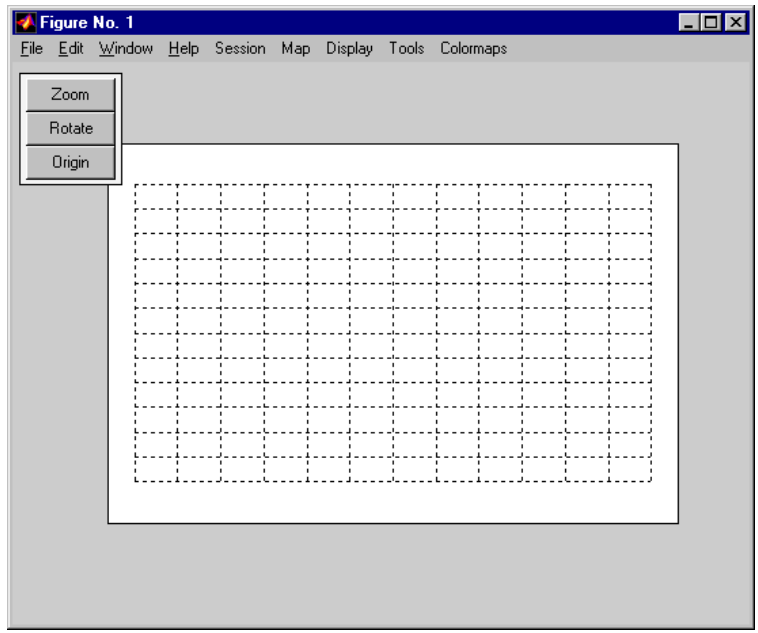
This creates a new figure window and brings up the **Projection Control** dialog box. Use the Map Projection pull-down menu to define an Equidistant Cylindrical projection and change the map origin longitude to center the map at 150 degrees West:



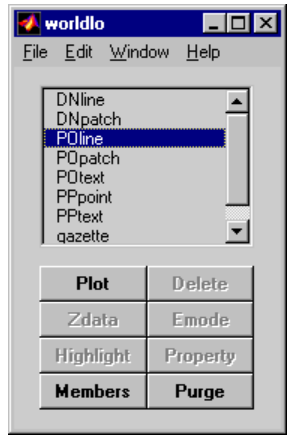
Press the **Grid** button to activate the **Map Grid Properties** dialog box and turn the grid on:



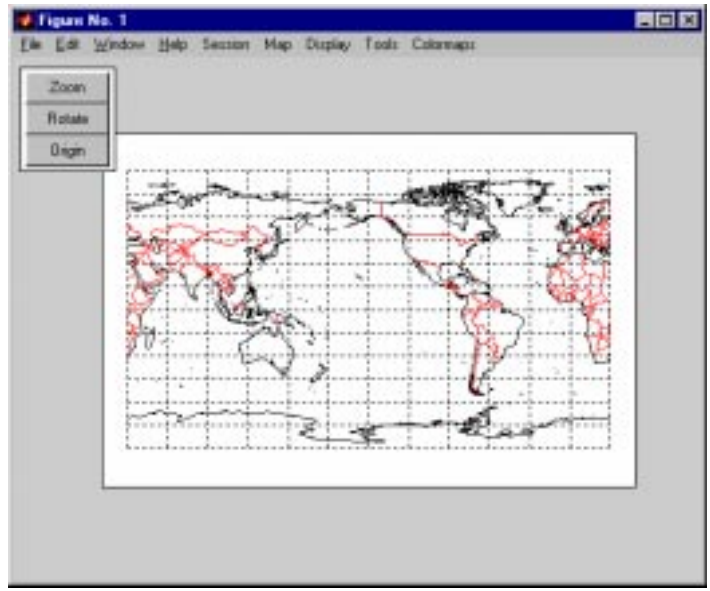
Press **Accept** to return to the **Projection Control** box, and press **Apply** to apply the settings to the map axes:



Select **Session⇒Layers⇒World LoRes** from the menu bar to activate the `mlayers` GUI with the `worldlo` workspace. This brings up the **worldlo** dialog box, which lists all the structures in the `worldlo` workspace:

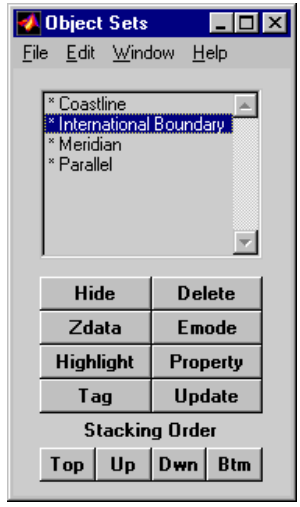


Select `POline` in the list box, and press the **Plot** button to display political/ ocean boundaries. Your figure now displays coastlines and international boundaries:



Close the **worldlo** dialog box. The `worldlo` variables associated with the dialog box have been removed from the workspace, but the plotted objects remain on the map.

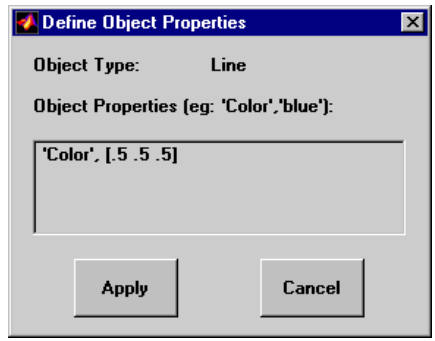
To view and edit properties of currently mapped objects, use the `mobjects` GUI. Select **Tools⇒Objects** from the menu bar. The **Object Sets** dialog box appears, which lists all the objects on the current map axes. An asterisk(*) next to an object indicates that it is visible, and an *h* next to an object indicates that it is currently hidden.



Select `International Boundary` and press the **Delete** button. A **Confirm Deletion** dialog box appears. Deleting an object set removes it from the current map axes. Press **Yes** to continue the deletion process.

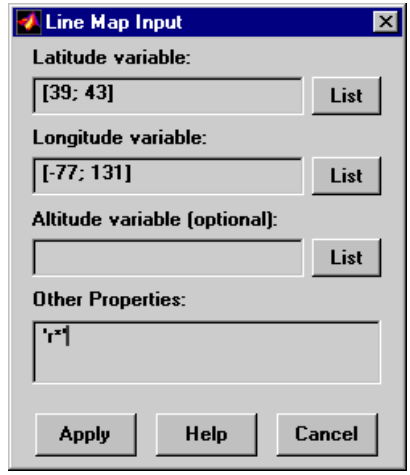
The figure now displays only coastlines. Change the coastlines from black to gray by selecting `Coastline` from the list box and pressing the **Property** button. The **Define Object Properties** box appears. Multiple pairs of property names and values can be entered in this dialog box, but you would simply like to change the color of your coastlines from black to gray.

Type `'Color',[.5 .5 .5]` in the **Object Properties** edit box, and press **Apply**:

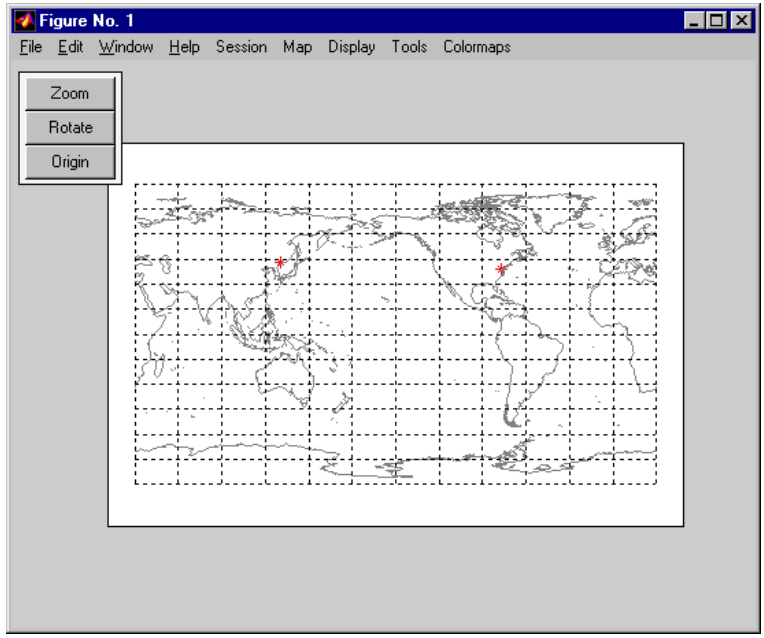


The coastlines change to gray. Close the **Objects Sets** dialog box.

Next we will plot two cities, Washington, D.C. and Vladivostok, Russia. Choose **Map⇒Lines** from the menu bar. The **Line Map Input** box appears. Enter the latitude vector `[39; 43]` and the longitude vector `[-77; 131]`. Note that you may also enter workspace variables as the latitude and longitude values. In the **Other Properties** edit box, type `'r*'` to plot a red marker at each city location. Press **Apply**.



The city markers now appear on the map:

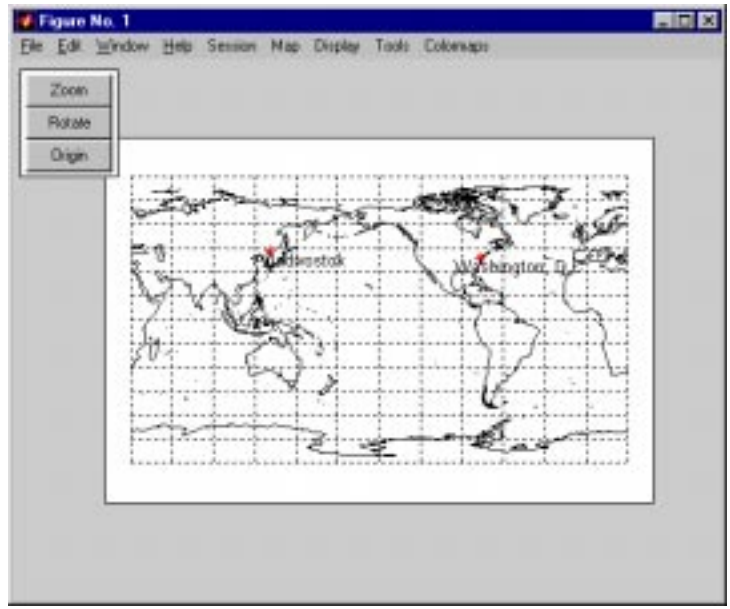


Select **Session⇒Command** from the menu bar. The **Workspace Commands** dialog box appears. This dialog box allows command line statements to be executed directly from `maptool`, rather than from the MATLAB command window. To plot city labels on the map, use the `gtextm` interactive tool, which will position strings at locations on the map specified by a mouse click. Enter the following in the **Workspace Commands** edit box:

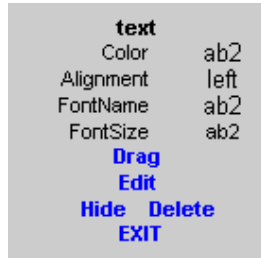
```
gtextm(strvcat('Washington, D.C.','Vladivostok'))
```

Press **Apply**. A crosshair appears on the map. The first input string is `'Washington, D.C.'`, so click in the vicinity of the Washington, D.C. city marker on the map. The string will be placed at that location. Next click near the Vladivostok city marker, and the second string will be placed there.

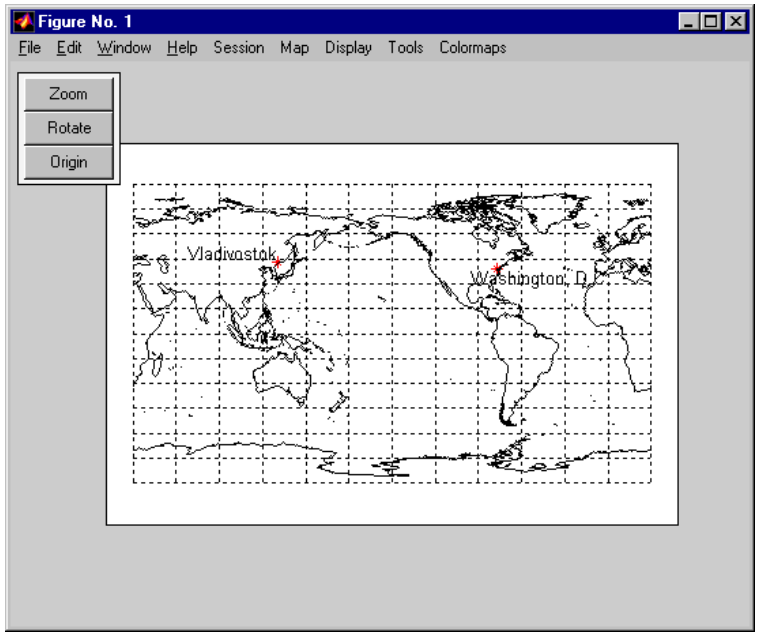
A **Status Report** box appears after each workspace command that is successfully completed. Here is the result of the `gtextm` operation:



The properties of many mapped objects can be edited with the **Click-and-Drag Property Editor** tool. Right-clicking on an object activates this functionality and allows for property manipulation through simple mouse clicks and drags. To reposition the Vladivostok label to make it easier to read, right-click on the text string. A **Click-and-Drag Property Editor** box for text objects appears:



Click **Drag**, and the label will now follow the mouse pointer as it is moved across the map display. (The **Click-and-Drag** tool can be moved out of the way by holding down the **Shift** key and dragging the box aside.) Position the label where you like, and click to place the text:



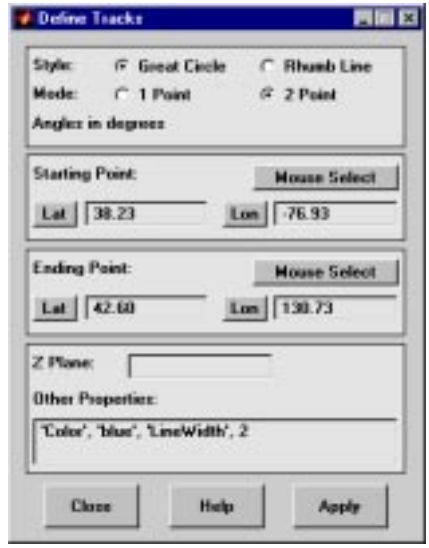
For more information on using the **Click-and-Drag Property Editor** tool, see the "GUI Reference" section of the *Mapping Toolbox Reference Guide*.

A great circle is the shortest distance between two places and can be used to determine approximate airplane flight routes. To plot a route between Washington, D.C. and Vladivostok, you can use the `trackui` tool. Select **Display⇒Tracks** from the menu bar. The **Define Tracks** dialog box appears.

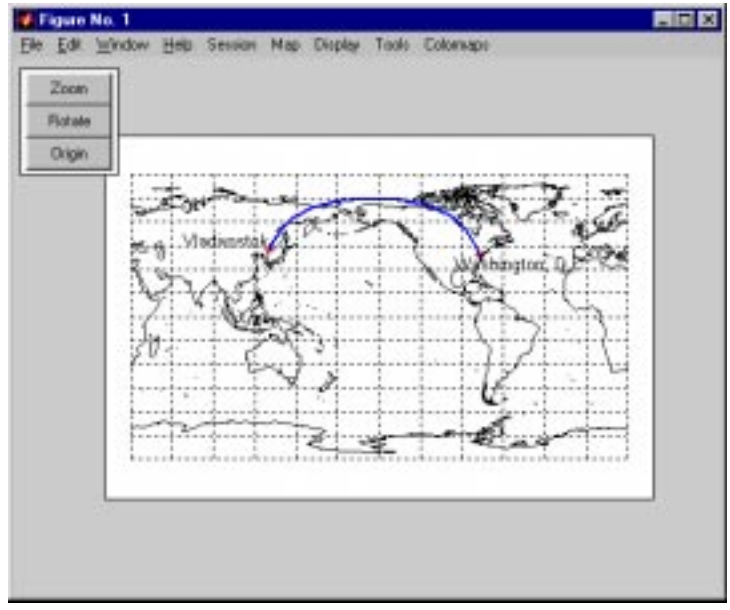
Select **Great Circle, 2 Point** mode. To specify Washington, D.C. as the starting point for your first track, press the **Mouse Select** button.

A crosshair appears on the map display. Position it over the city marker and click. The coordinates appear in the **Starting Point Lat and Lon** boxes. Using the **Ending Point Mouse Select** button, select Vladivostok as your ending point.

In the **Other Properties** edit box, type `'Color','blue', 'Linewidth',2`:



Press Apply, and the flight track is displayed:



If you would like to save your map, select **File⇒Save** from the menu bar to save the figure window in an M-file. The **Zoom** and **Rotate** buttons should be turned off before saving. When you execute the M-file, the figure and uicontrols will be recreated to their original appearance.

Example Two: Creating a Regional Map

Another way to easily plot `worldlo` atlas data is by using the `worldmap` tool. `worldmap` plots a specified region using appropriate default projection parameters and map axes settings. `worldmap` can be used to plot individual countries or entire regions. Here are some examples of valid `worldmap` commands:

```
worldmap france
worldmap asia
worldmap 'north pole'
```

`worldmap` also accepts latitude and longitude limits to define the area to be plotted. The following example uses `worldmap` to create a map of North and South Korea. The maptool menu will then be added to the `worldmap` figure and GUIs will be used to add a digital elevation map and plot the locations of earthquake epicenters. The `korea` and `koreaEQdata` workspaces used in this example are shipped with the Mapping Toolbox. The first step is to load the data we will be using:

```
load korea
load koreaEQdata
who

Your variables are:
kdata       map         maplegend
```

To plot North and South Korea, first define the geographic limits of the area with the `limitm` tool. Enter the following at the MATLAB command prompt.

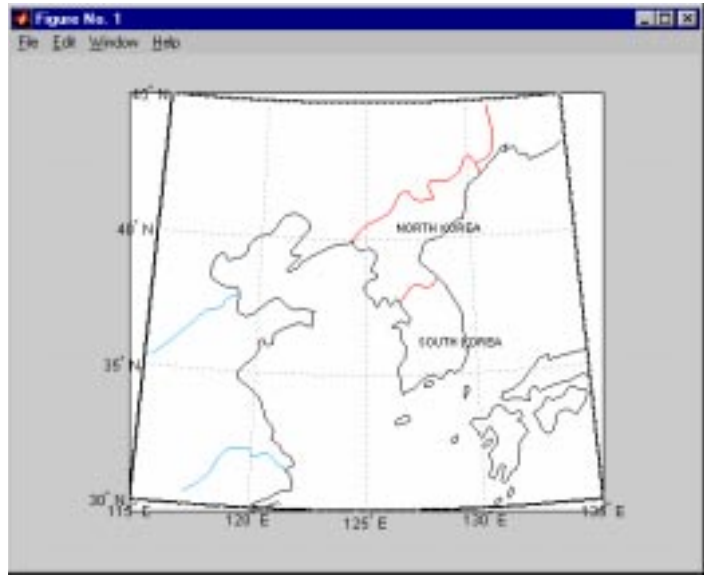
```
limitm
```

The **Map Limit Input** dialog box appears.

Accept the default variable names and press **Apply**. The `latlim` and `lonlim` variables now contain the latitude and longitude limits of the `korea` data. Now `worldmap` can be used to plot the `worldlo` data within these limits. At the MATLAB command prompt, enter:

```
worldmap(latlim, lonlim)
```

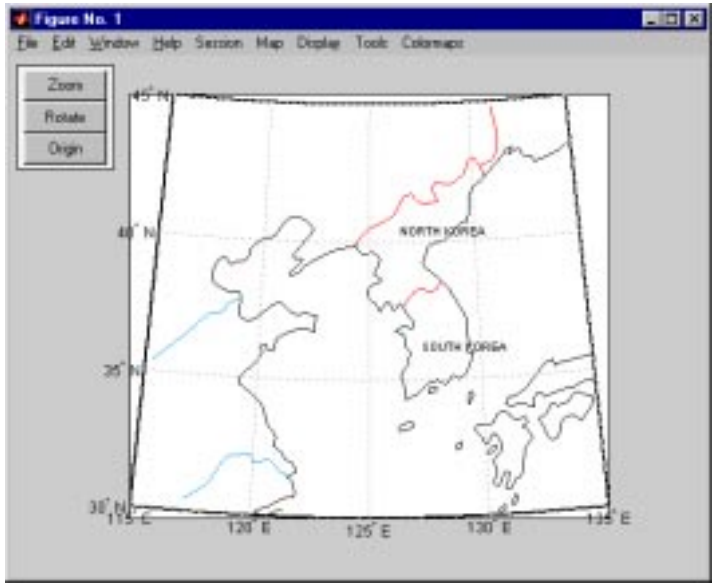
The following map is created:



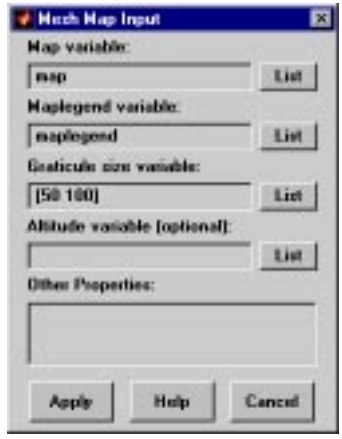
For easy access to the Mapping Toolbox GUIs, we can now add the `maptool` menu to the `worldmap` figure. At the command prompt, enter:

```
maptool
```

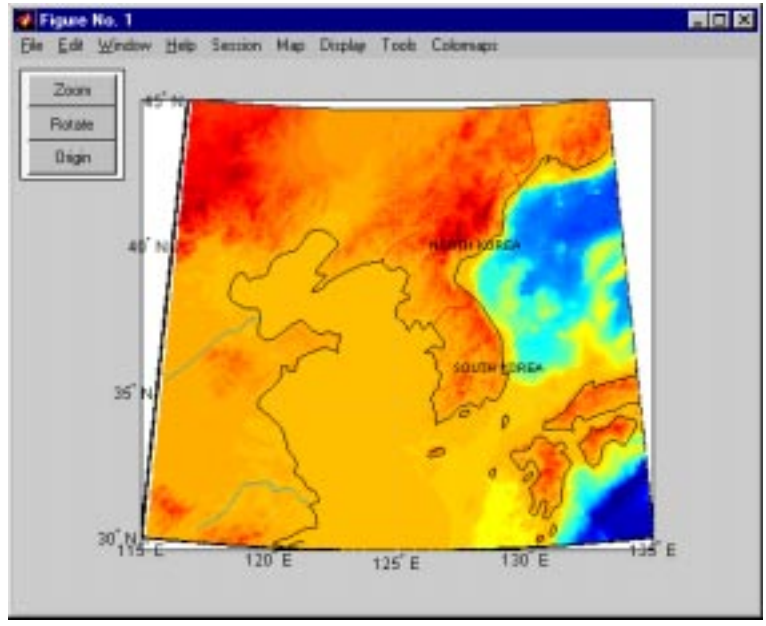
The `maptool` menu and mouse buttons are automatically added to the figure window:



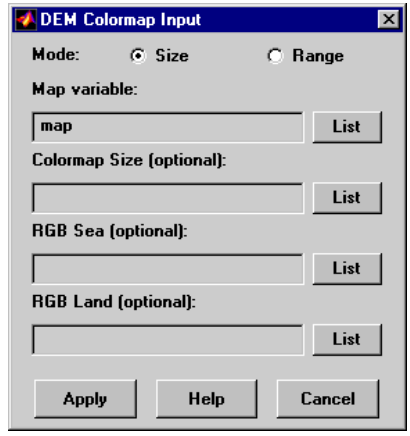
Next we will plot the digital elevation data contained in the korea variables `map` and `maplegend`. Select **Map⇒Regular Surfaces** from the menu bar. The **Mesh Map Input** dialog box appears:



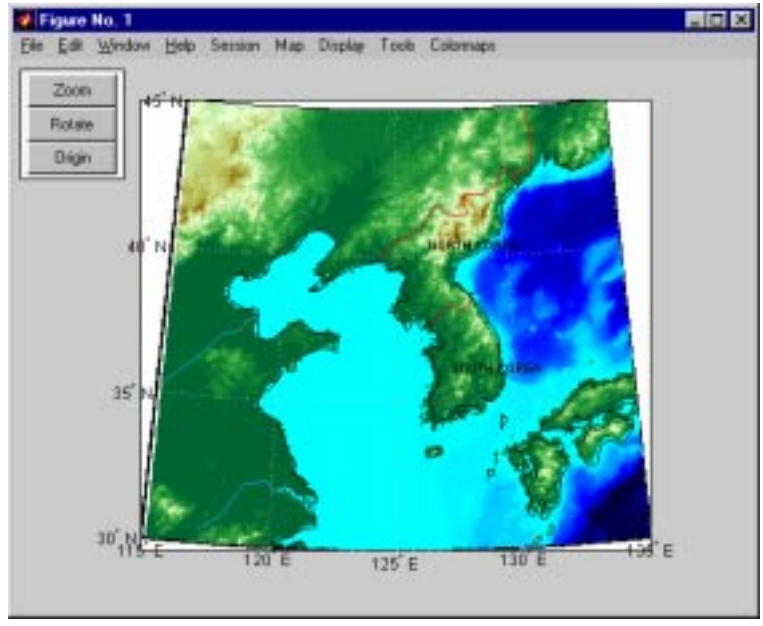
The default `map` and `maplegend` variable names match the `korea` variable names, so just press **Apply**. The surface object is added to the map:



To change the colormap, select **Colormaps⇒Digital Elevation** from the menu bar. The **DEM Colormap Input** box appears. Enter the map variable `map` and press **Apply**.

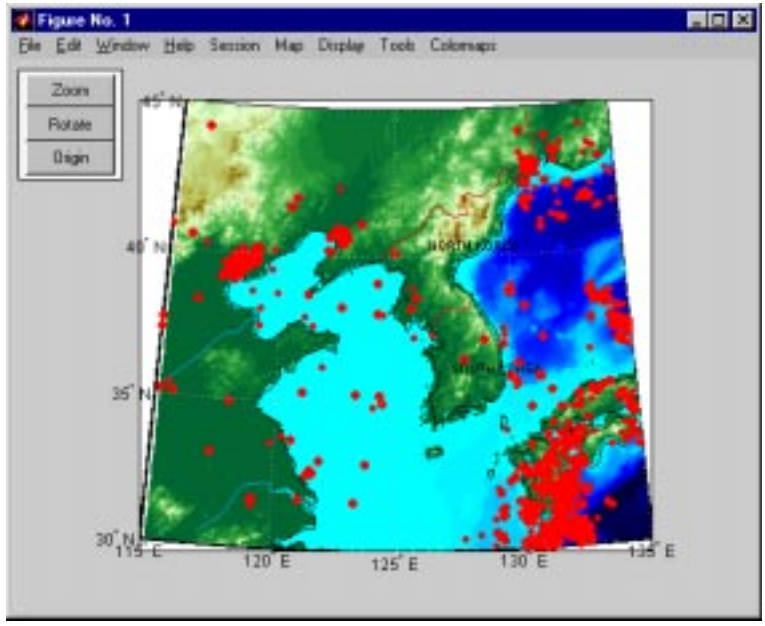


The new colormap is applied:



To plot the locations of earthquakes, we will use the `scatterm` GUI. Select **Map⇒Scatter** from the menu bar. The **Scatter Map Input** box appears. The variable containing the earthquake data is `kdata`. For the **Latitude variable**, enter `kdata(:,1)`. For the **Longitude variable**, enter `kdata(:,2)`. And for the **Marker size**, enter `kdata(:,3).^2`, to display the markers in proportion to the earthquake magnitude. Squaring the magnitudes increases the difference in marker sizes, making them easier to distinguish. Change the **Marker color** to `'r'`, and check the **Filled** box.

Press **Apply** and the earthquake markers are displayed:



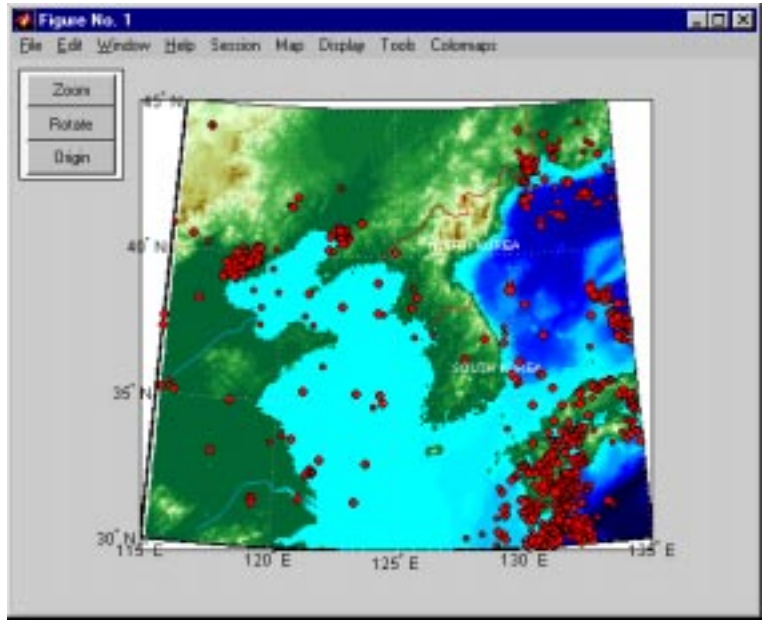
To hide the coastlines, select **Tools⇒Hide⇒Object** from the menu bar. The **Select Object** dialog box appears. Select `Coastline` and press **Ok**.

The coastlines are no longer displayed.

To make the country names a little bit easier to read, we will change the color from black to white. Select **Tools⇒Objects** from the menu bar. The **Object Sets** box appears. Select `Political Names` and press **Property**. In the **Define Object Properties** edit box, type `'Color','white'` and press **Apply**.

Next, we will change the edge color of the earthquake markers to better distinguish each earthquake site. The earthquake markers are patch objects, and are indicated in the **Object Sets** box as `patch`. Select `patch` and press **Property**. Enter `'MarkerEdgeColor','black'` and press **Apply**.

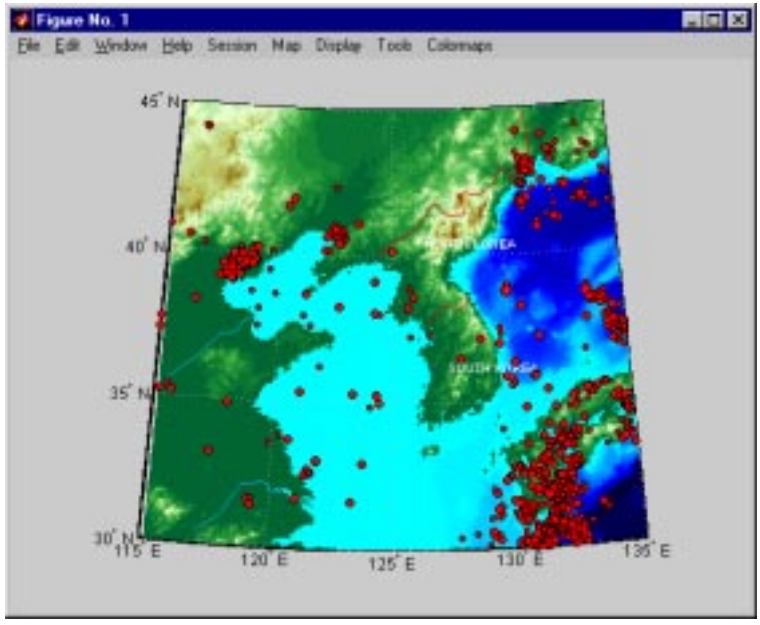
The map now appears as follows:



When we first plotted this map of korea, `worldmap` selected the Lambert Equidistant Conic projection as appropriate for the location and size of the region. Suppose you would now like to switch to an equal area projection. Select **Display⇒Projection** from the menu bar. From the **Map Projection** drop-down list, select the Equal Area Conic (Albers) projection. Press **Apply**, and all of the mapped objects are now projected onto the new projection.

Select **Tools⇒Hide** from the menu bar to hide the **Zoom**, **Rotate**, and **Origin** buttons on the figure window. Select **Tools⇒Axes⇒Invisible** from the menu bar to hide the axes box.

Here is the final map of Korean earthquake locations:



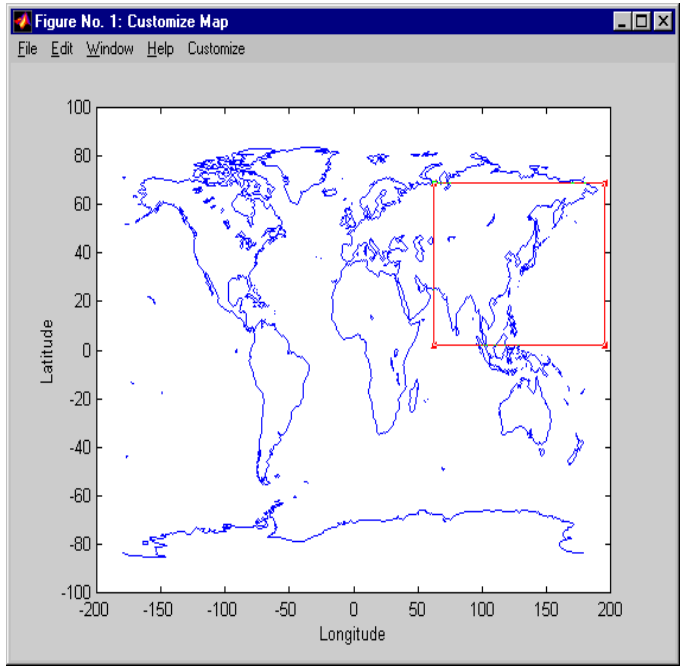
Constructing Personalized Map Data with GUIs

Trimming an Existing Data Set

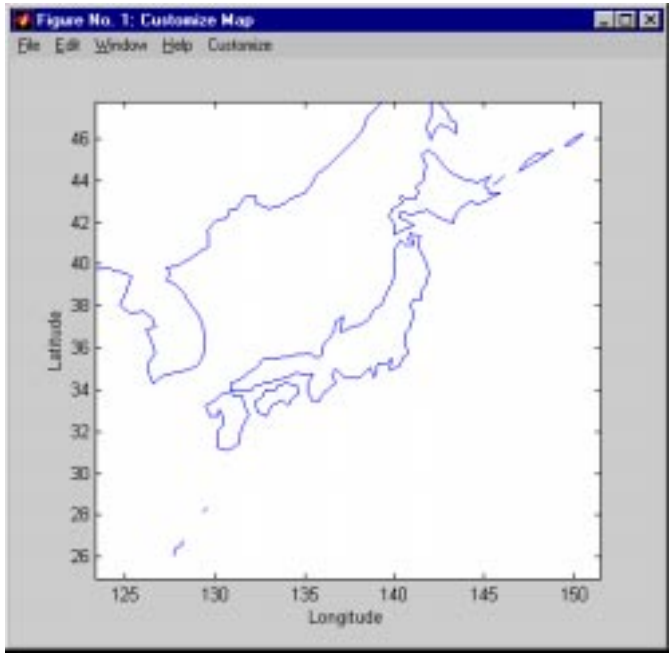
You can use the `maptrim` tool to trim a map dataset. For example, you want to work with just a portion of the coastline data found in the `coast` workspace, say the region around Japan. Use the `maptrim` tool to trim the coastline data to this region, then save the trimmed dataset as a new vector or matrix map.

```
load coast
maptrim(lat,long)
```

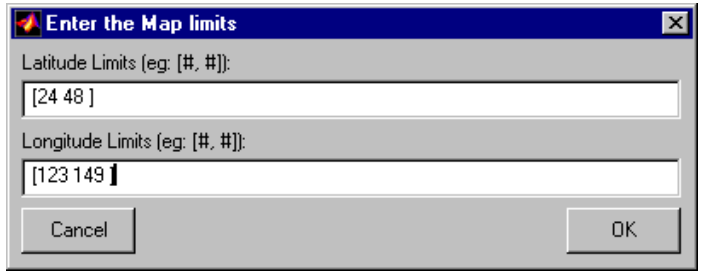
A new figure window displays the unprojected coastline data. Drag the zoom box to the region around Japan.



Double-click in the box to zoom in. Repeat this process until the area of interest fills the figure window.

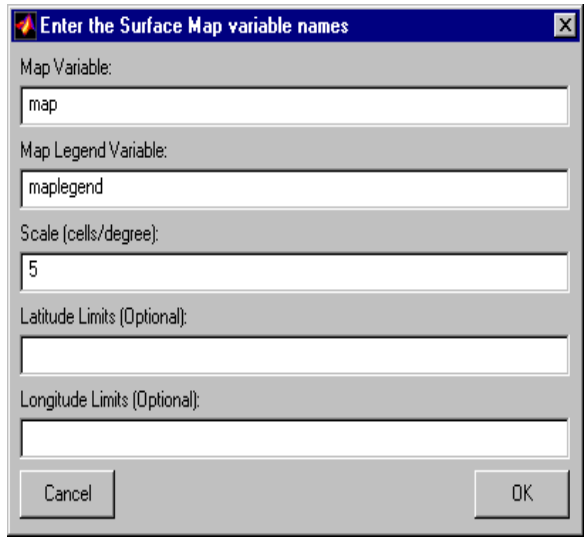


Select **Customize⇒Limits** to specify the exact limits of the region of interest. A dialog box appears with the latitude and longitude limits of the current display. Set the latitude limits to [24 48] and the longitude limits to [123 149]. Press **OK**.



To save this trimmed region as vector data, choose **Customize⇒Save As⇒Line** or **Customize⇒Save As⇒Patch** from the menu bar. A dialog box will appear prompting you to enter variable names for your new dataset.

The `maptrim` tool can also be used to convert data from vector format to matrix format. The command line equivalent function is `rec2mtx`. To convert the trimmed region around Japan, choose **Customize⇒Save As⇒Regular Surface** from the menu bar. Save the trimmed map data under the variable name `map`, and the map legend under variable name `maplegend`. Enter a scale of 5 cells per degree. Press **OK**.



You now have a regular matrix map of Japan in your workspace. The **Customize Map** figure window can now be closed.

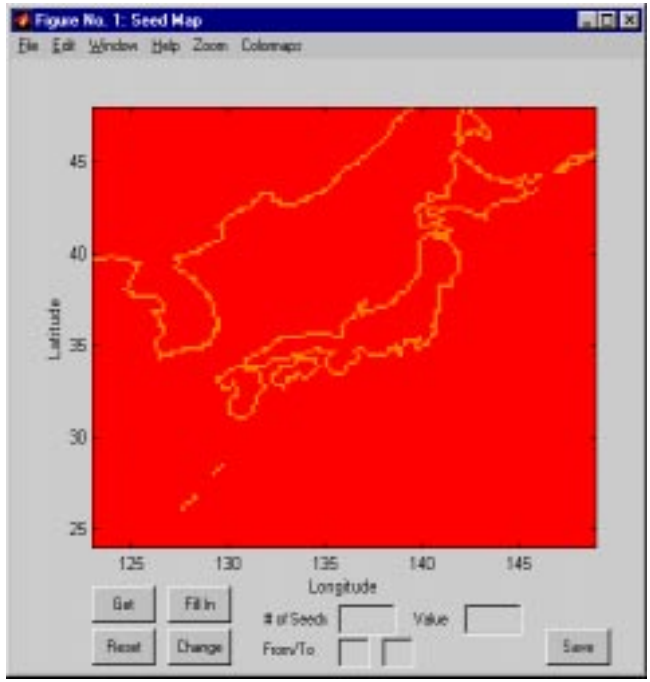
Encoding a Regular Surface Map

Encoding is the process of filling in specific values in regions of a matrix map up to specified boundaries. For example, you can fill in the interior of an island, bounded by its coastline, with the value six. Encoding entire regions at one time allows indexed maps to be created quickly. Some parts of this process are automated in the `vec2mtx` and `country2mtx` commands.

A regular matrix map can be encoded using the `seedm` tool. To encode the map of Japan just created, activate the `seedm` tool by typing the following:

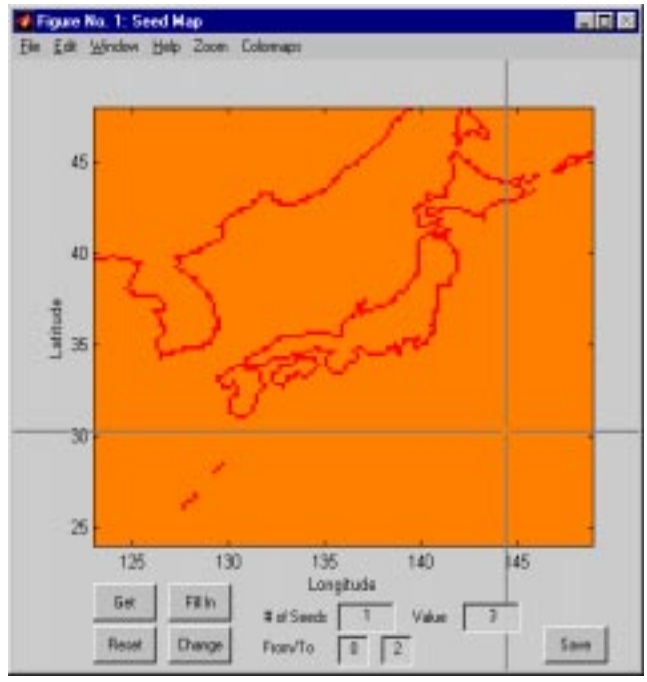
```
seedm(map,maplegend);
```

A new figure window displays the surface map of Japan.



Currently, your map consists of 1's where there is a coastline and 0's elsewhere. Suppose you would like different values for four types of data: coastlines = 1, Japan = 2, water = 3, and other countries = 4. Enter the values `0` and `2` in the **From/To** boxes and press **Change**. This will change all values of 0 to 2.

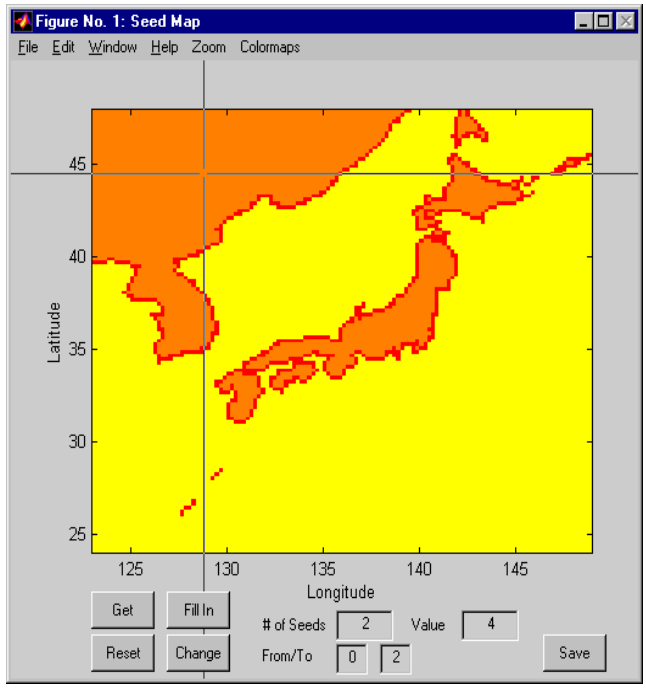
To encode the regions of water, enter a 1 in the **# of Seeds** box and a 3 in the **Value** box. Press **Get**, and a cross hair will appear on the map. Position the cross hair over a region of water and click.



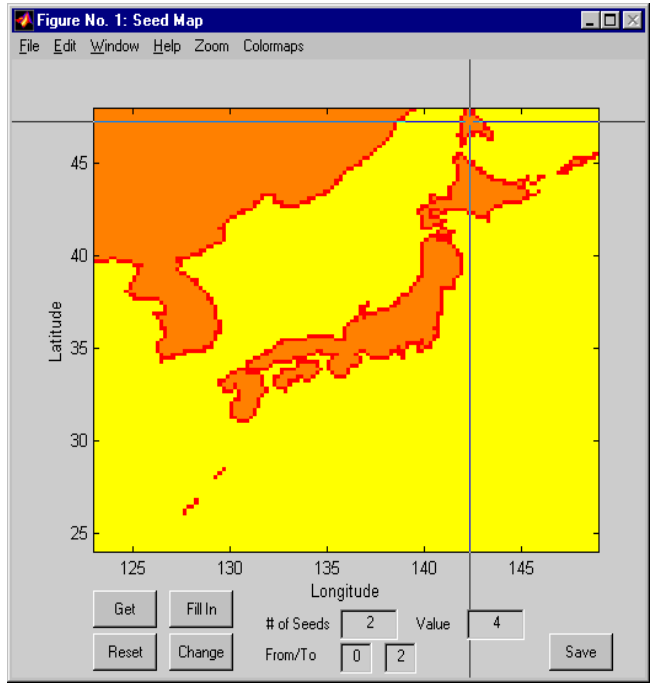
Press **Fill In**, and all values corresponding to water will be changed to 3.

Encode the other countries, there are three regions that will need to be filled with the value 4. Encode the large region of Russia, China, North Korea, and South Korea in the upper left of the map and the Russian Island of Sakhalin at the top of the map in a single step by using two seeds. Enter a 2 in the **# of Seeds box**, and a 4 in the **Value** box.

Press **Get** and the cross hair appears. Click in the first region at the upper left of the map.

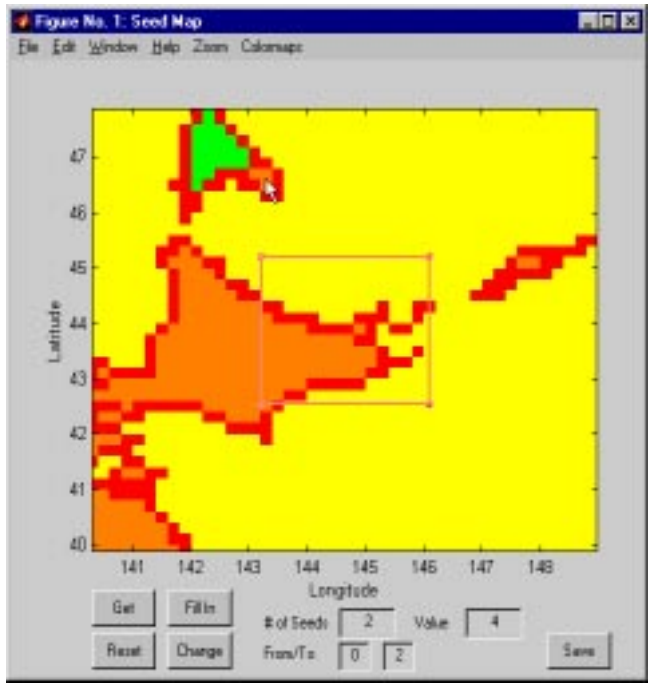


For the second seed, click inside Sakhalin Island.



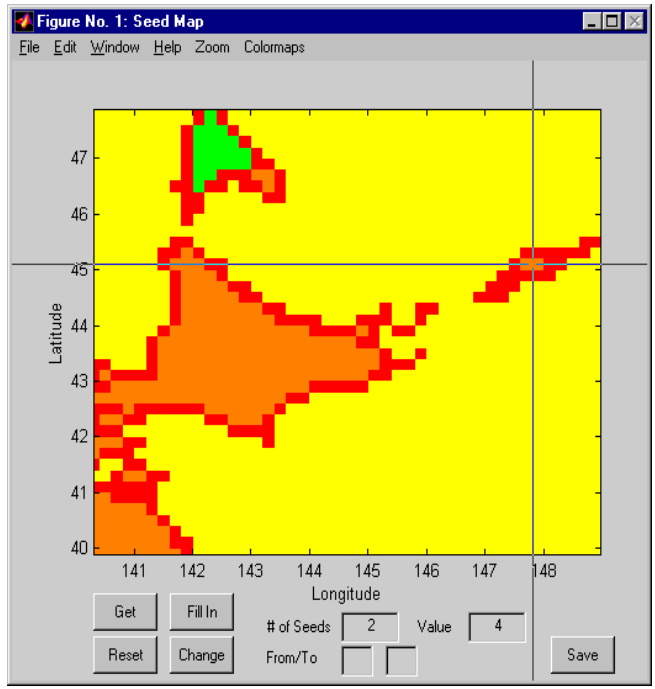
Press **Fill In**, and the values corresponding to those two regions will be changed to 4. The last region you need to encode is the small Kuril island that appears to the east of Hokkaido in the upper right corner of the map.

Drag the zoom box over the island and double-click to zoom in.

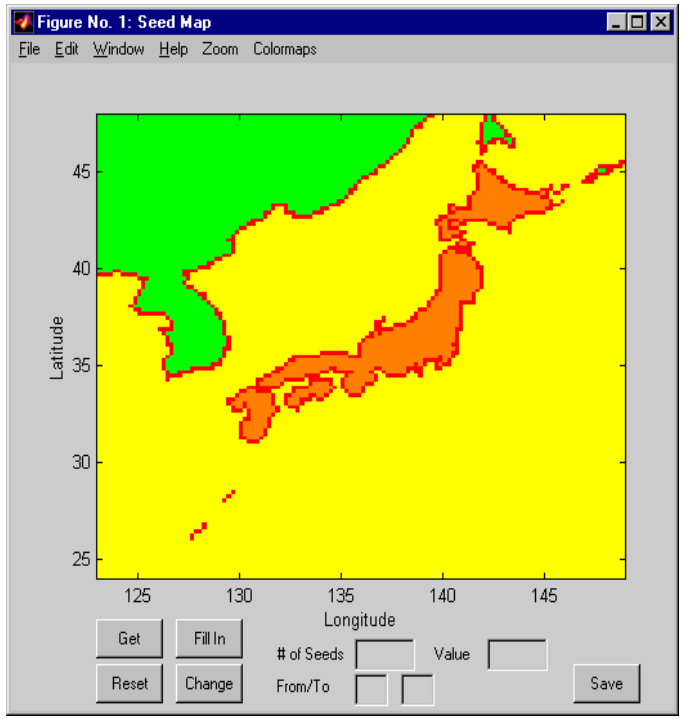


Because the island is so small, and much of its shape consists of coastline, this zoomed-in view will make it much easier to select the interior of the island, rather than its coast.

From this view, you can also see that you missed a spot on Sakhalin Island (see mouse pointer in the picture above), so you will need two seeds for this step. Enter a 2 in the **# of Seeds** box and a 4 in the **Value** box. Press **Get** and click inside the missed region of Sakhalin Island and inside the Kuril island.



Press **Fill In** and the values corresponding to those two regions will be changed to 4. The final map appears as follows:



Press the **Save** button and enter the variable name `map` to replace the old matrix map of Japan with this new encoded map. The **Seed Map** dialog box can now be closed.

Creating a Personalized Colormap

Now you can use the `colorm` tool to make a custom colormap for your map of Japan. To activate the `colorm` GUI, type:

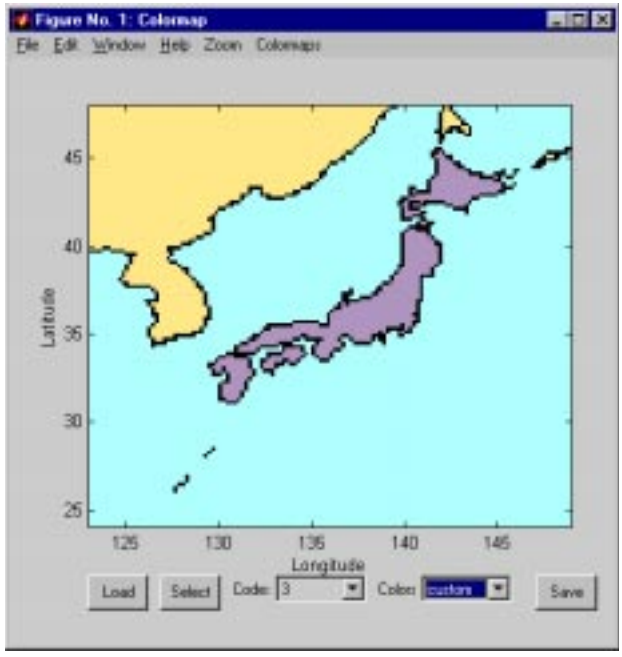
```
colorm(map,maplegend)
```

A new figure window displays the map. Recall that the values for this matrix map are as follows: coastlines = 1, Japan = 2, water = 3, and other countries = 4.

To change the coastlines to black, select `1` from the **Codes** menu, and choose `black` from the **Color** menu. Next, make Japan a light purple by selecting `2` from the **Codes** menu and choosing `custom` from the **Color** menu. The **Custom Color** GUI appears.

Select a custom color, and choose **OK**. Japan will change to the new color. Next, change the color of all other countries to a light beige by selecting `4` from the **Codes** menu, and `custom` from the **Color** menu. Use the **Custom Color** GUI to choose a light beige color, then press **OK**.

Finally, to change the color of water to a light blue, select 3 from the **Codes** menu and custom from the **Color** menu. Choose a light blue from the **Custom Color** GUI and press **OK**. The map of Japan now appears:



Press the **Save** button, then press **OK** to accept the default name `clrmap` for the new colormap variable. The **Color Map** dialog box can now be closed.

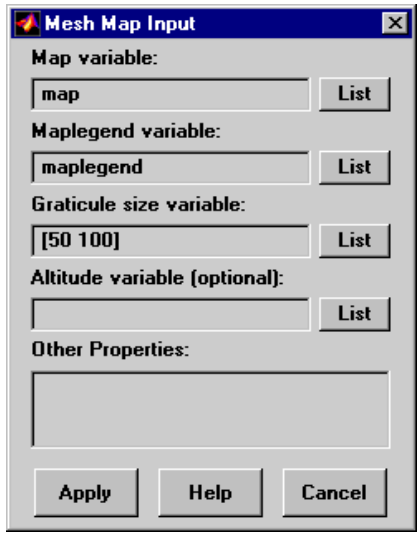
To display the map of Japan, type `axesm`.

In the **Projection Control** box, choose a Miller Cylindrical projection and define the map latitude and longitude limits to be [24 48] and [123 149], respectively.

Use the **Grid** button to turn the grid on, then press the **Parallel and Meridian Settings** button and set the longitude locations and latitude locations to 5 to display grid lines every 5 degrees.

Use the **Label** button to turn the meridian and parallel labels on. Change the label format to none. Press **Parallel and Meridian Settings** button and change the longitude and latitude label locations to 5 to display labels every five degrees. Press **Apply** to apply the settings to the map axes.

To plot the map of Japan using the **Mesh Map Input** dialog box, type `meshm`.

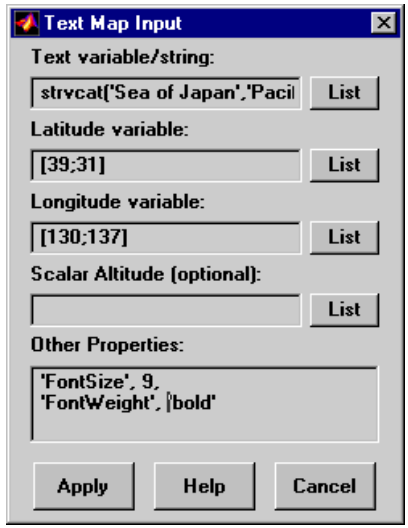


Your variable names are the same as the defaults, so press **Apply** to plot the map. After the map is plotted, type `colormap(clrmap)`. This will set the current colormap to the customized one you created. To annotate the major bodies of water in the display, type `textm`.

The **Text Map Input** dialog box appears. In the **Text variable/string** edit box, enter the following:

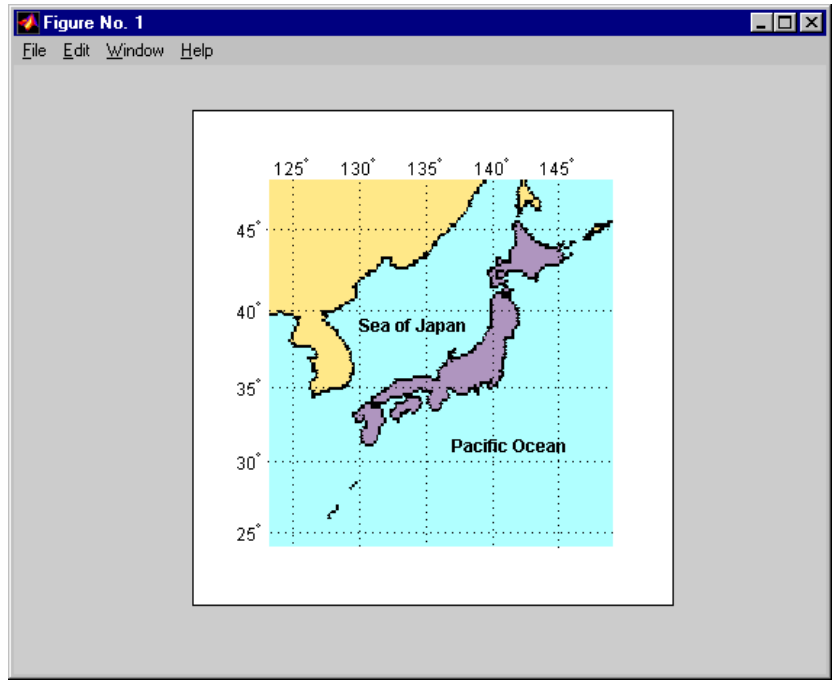
```
strvcat('Sea of Japan','Pacific Ocean')
```

In the **Latitude variable** edit box enter `[39;31]`, and in the **Longitude variable** edit box enter `[130;137]`. In the **Other Properties** edit box, enter `'FontSize',9` and `'FontWeight','bold'`.



Press **Apply** to plot the text.

The final display of your customized map of Japan appears as follows:



The primary utility of such matrix maps is not display, but analysis. You can use this map to quickly identify whether a particular geographic location belongs to Japan, for example. See the political matrix maps section in Chapter 3, “Atlas Data” for more examples.

7

External Data Interface

Working With External Data

While the Mapping Toolbox atlas data described in Chapter 3 is suitable for world or regional maps, more detailed data is usually required for larger scale maps. The best sources of such data are often national mapping agencies, who make their data available to the public at little or no charge, often over the Internet or on CD-ROM. United States government publications are generally free of copyright, allowing free re-use of the data.

Although the Mapping Toolbox can easily display such geographic data once it is in the MATLAB environment, importing it is often difficult because of complicated data file formats. To make these detailed datasets readily accessible, The Mapping Toolbox provides a number of external data interface functions. These functions allow you to read the data into Mapping Toolbox data types such as regular and general matrix maps and geographic data structures.

A wide variety of data is accessible through these external interface functions. There are vector map data of political entities ranging from national and local government boundaries, infrastructure such as roads, railroads, airports and utilities, and geographic data such as coastlines, rivers, elevation contours, and land use. The Earth's surface and ocean elevations are available on regular grids at a number of different resolutions. Astronomical data in the form of star databases is also available.

Many other types of data can be imported to MATLAB beyond the specialized data file formats accessed by the external interface functions. The most important are mapped scientific data in the form of HDF files and JPEG files. See the MATLAB documentation for the `imfinfo` and `imread` functions for more information on how to read these file formats.

Global Vector Data

The Digital Chart of the World/VMAP0

The most detailed, publicly available set of global vector data with consistent coverage of important map features is the Digital Chart of the World (DCW) and the revised successor to the DCW, Vector MAP level 0. Both contain vector data scanned from printed Operational Navigation Charts (ONC) and Jet Navigation Charts (JNC). The ONC charts were compiled at a scale of 1:1,000,000 and the JNC at 1:2,000,000. The publisher of the DCW was the U. S. Defense Mapping Agency, now incorporated into the U. S. National Imagery and Mapping Agency (NIMA). NIMA publishes the VMAP0.

Both contain data for the entire world in separate layers of coverage, including political and ocean features, drainage, hypsography, land cover and vegetation, populated places, roads, railroads, airports, transportation structure, and cultural landmarks. The features are formatted as patches, lines, points, or text.

The DCW was published in 1992 on four CD-ROMs. A revised version, called VMAP0, also distributed on CD-ROMs, was released in 1997. The DCW is now out of print. For more information on purchasing VMAP0 CDs, see the `vmap0data` entry in the "External Data Reference" section of the *Mapping Toolbox Reference Guide*.

The Mapping Toolbox provides high-level functions that compile and extract information from the DCW and VMAP0 into Mapping Toolbox geographic data structures. The following sections introduce these functions and show how to work effectively with the data. Since the DCW is no longer available, these examples focus on VMAP0, but working with the DCW is very similar. The exception is the gazetteer feature, which was dropped in VMAP0. This data can be retrieved from The MathWorks FTP site.

Looking Up Names and Locations in the DCW Gazetteer

One of the best sources of place name information is the Digital Chart of the World Gazetteer. This is an extensive collection of place names and types within a library. The `NOAMER` library, containing the data for North America, has 37,696 names in its gazette. The entire DCW gazette has more than 100,000 names. It can be used to quickly find the locations of places and can help to pick the latitude and longitude limits for other DCW functions.

The Mapping Toolbox interface to the DCW gazette is the dcwgaz function. If the EURNASIA CD is available and mounted on the desktop of a Macintosh computer, you can search for the geographic location of the city of Apatin in Yugoslavia by typing:

```
dcwgaz('EURNASIA','apatin')
APATIN
ans =
             type: 'text'
    otherproperty: {1x2 cell}
              tag: 'Built up area'
           string: 'APATIN'
         altitude: []
              lat: 45.6660
             long: 18.9830
```

On other computers, you must provide a device name as the first argument. The string you provide will depend on how the CD-ROM is mounted. For a Windows PC, the command might be:

```
dcwgaz('F:','EURNASIA','apatin')
```

and on a UNIX computer with the CD-ROM mounted as '/cdrom', you should type:

```
dcwgaz('/cdrom/','EURNASIA','apatin')
```

See the documentation that came with your CD-ROM drive for more information on how disks are mounted.

If you do not have access to the DCW CDs, you can retrieve just the gazette data from The MathWorks FTP site. The gazette data for all four of the libraries has been collected into one directory called DCW_GAZETTE. If you use this version of the data, your device name will need to be the full path down to and including the DCW_GAZETTE directory.

Matches are displayed on screen during the search. The output is returned in a Mapping Toolbox geographic data structure, with the tag identifying the database layer name within the DCW. If more than one entry matches the name, the results are contained in an arrayed structure.

For example, the following command prints to the screen the 95 names in the North American library that begin with the word *port* and returns the structures `txtstruc` and `ptstruc` that can be displayed using `displaym` or `mlayers`.

```
[txtstruc,ptstruc] = dcwgaz(devicename,'NOAMER','port ');
```

Here is a map that shows the ports around the Great Lakes:

There are also databases of geographic names that you can search over the Internet. For global data, try the National Imagery and Mapping Agency GEOnet Name Server at <http://164.214.2.59/gns/html/index.html>. For the United States, try the U. S. Geological Survey Geographic Names Information System at <http://www-nmd.usgs.gov/www/gnis/>.

Importing VMAP0 Themes

VMAP0 is designed as a database of geographic information for general mapping applications. The `vmap0data` function is used to access the map data. The volume of data on the VMAP0 is so large that you must limit your requests to a reasonable amount. First, select the geographical area of interest by giving the name of one of the libraries in the VMAP0. There is one CD for each library: `'NOAMER'` (North America), `'EURNASIA'` (Europe and Northern Asia), `'SOAMAFR'` (South America and Africa), and `'SASAUS'` (Southern Asia and Australia).

You may also need to specify the device name of the CD, just as you did for the `dcwgaz` function. The device name depends on your operating systems and how it is configured. If more than one CD can be mounted at once, there will

probably be a different device name for each CD. With the correct device name, you should see the following for the SASAUS CD-ROM and library:

```
cd(devicename)
ls
FGDC_DAT.TXT README1.TXT  VIEW         VMAPLV0

cd vmaplv0
ls
DHT      LAT      RFERENCE SASAUS
```

The geographic region of interest is further restricted by latitude and longitude limits, given in units of degrees.

You also need to specify which *theme*, or type of data, you want to extract. Themes are layers of data related to a kind of feature, like roads and drainage, and are given as strings. If you enter a code that is not recognized, a list of valid codes and their descriptions are displayed.

Finally, you need to indicate what type of graphic objects you want (patches, lines, points, or text). These are called *topology levels*.

You can see the list of available themes by entering an incorrect one, like '?'. The latitude and longitude limits correspond to the Cape Cod region:

```
Struc = vmap0data(devicename,'NOAMER',...
                  [41 44],[-72 -69],'?','line')
??? Error using ==> vmap0data
Theme not present in library NOAMER

Valid theme identifiers are:
libref : Library Reference
tileref: Tile Reference
bnd    : Boundaries
dq     : Data Quality
elev   : Elevation
hydro  : Hydrography
ind    : Industry
phys   : Physiography
pop    : Population
trans  : Transportation
util   : Utilities
veg    : Vegetation
```

If you are using a Windows PC, with the 'NOAMER' CD-ROM in the 'F:' drive, you can read the Political/Ocean line features for the same Cape Cod region by entering the following:

```
BNDline = vmap0data('F:','NOAMER',...
                    [41 44],[-72 -69],'bnd','line')
BNDline =
1x8 struct array with fields:
    type
    otherproperty
    altitude
    lat
    long
    tag
```

The result is a Mapping Toolbox geographic data structure. Look at the geographic limits by concatenating the data with `extractm`.

```
[lat,long] = extractm(BNDline);
[max(lat) max(long); min(lat) min(long)]
ans =
    45   -66
    40   -72
```

Notice that this region is larger than was requested. VMAP0 tiles the world into quadrangles of various sizes. When a tile overlaps the requested region, `vmap0data` extracts the data for the entire tile. Similarly, if you request a point location, the entire quadrangle is returned. All objects are cut at the edges of the tiles.

The kind of information in an element of the structure is described by the `tag` field. There are eight kinds of boundary lines in this region:

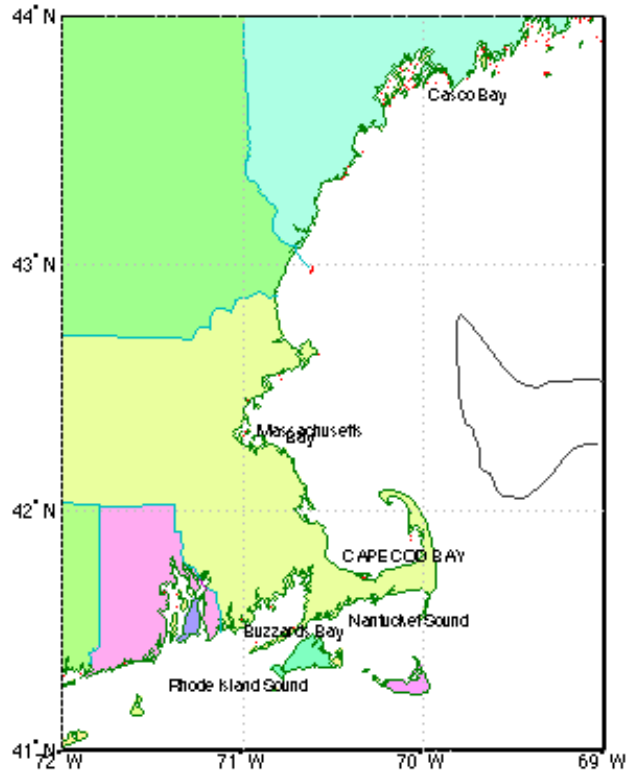
```
unique(strvcat(BNDline.tag),'rows')
ans =
Accurate; Definite; Coastline/Shoreline
Accurate; Definite; International; Administrative Boundary
Accurate; Definite; Primary/1st Order; Administrative Boundary
Approximate; Definite; International; Administrative Boundary
Depth Contour; 1000
Depth Contour; 200
Depth Contour; 2000
Depth Contour; 600
```

You can extract more than one topology level using a cell array of level names. Extracting patches and points is more time-consuming, so this will take longer than the previous example. The following example assumes that the 'NOAMER' CD-ROM is mounted on a UNIX computer as '/cdrom':

```
[BNDpatch,BNDpoint,BNDtext] = vmap0data('/cdrom','NOAMER',...
          [41 44],[-72 -69],'bnd',{'patch' 'point' 'text'});
BNDpatch(1)
ans =
             type: 'patch'
    otherproperty: {}
         altitude: []
              lat: [44x1  double]
             long: [44x1  double]
              tag: 'BAY OF FUNDY; Water (except Inland)'

BNDpatch(end).tag
ans =
North America; United States; VERMONT; Administrative Area
```

The following commands plot the political and ocean features on a Mercator projection, hiding the ocean patches. They also override the default colormap used for the land patches and remove the edge color to hide any tile joints:

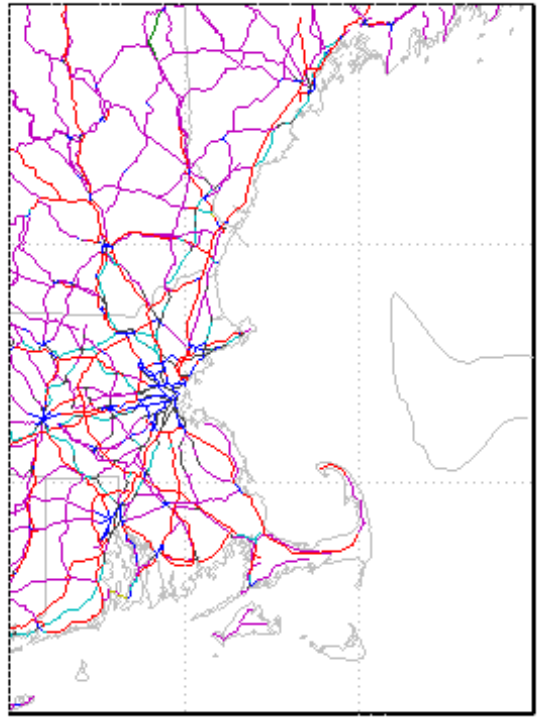
```
worldmap([41 44], [-72 -69],'none')
setm(gca,'MapProjection','mercator'); tightmap

hBNDpatch = displaym(BNDpatch); hBNDline = displaym(BNDline);
hBNDpoint = displaym(BNDpoint); hBNDtext = displaym(BNDtext);
set(hBNDpatch,'EdgeColor','None')
hidem(handlem('NORTH ATLANTIC OCEAN; Water (except Inland)'));
colormap(polcmap)
zdatam('allpatch',-1)
```



Notice how much more detailed this map is than even the best of the atlas data provided in the Mapping Toolbox. The highest resolution atlas data is in the `usahi` workspace, which is limited to the United States. VMAP0 covers the entire world at this level of detail. Unlike the earlier DCW, islands are not tagged with a state name.

The VMAP0 does not contain all topology levels in each theme. Roads, for example, have no patches. If data is requested for topology levels that are not available, the function issues a warning and returns an empty matrix. Here is a display of the road and railroad line data for the Cape Cod map.

```
[TRpatch,TRline,TRpoint,TRtext] = vmap0data('cdrom','NOAMER',...
          [41 44],[-72 -69],trans',{'all'});
clma
hBNDline = displaym(BNDline);
set(hBNDline,'Color',[1 1 1]*0.75,'LineWidth',1)
hTRline = displaym(TRline);
```



You can use the `handlem` graphical user interface to select only the roads or railroads. This would let you change, for example, all of the railroad lines to a dashed style.

The network of roads around Boston gives away its location. VMAP0 also includes a populated places theme, containing the boundaries of built-up areas. Extract the 'pop' theme, reduce the displayed area to Boston and Cape Cod, and plot the populated places:

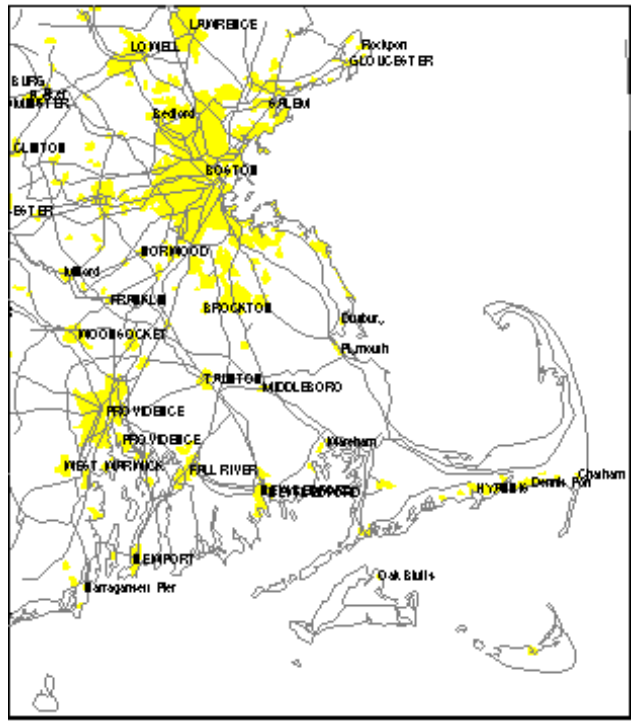
```
PPpatch = vmap0data('NOAMER',[41 44],[-72 -69],'pop','patch');

PPpatch = vmap0data('/cdrom','NOAMER',[41 44],[-72 -69],...
                    'pop','patch');

setm(gca,'MapLatLimit',[41.13 42.75],...
         'MapLonLimit',[-71.7 -69.8])
tightmap

hPPpatch = displaym(PPpatch);
set(hPPpatch,'FaceColor','y','Edgecolor','None')
zdatam(hPPpatch,-1)
```

The names of many of the cities are stored in the tag field of populated place patches and points. Unlike the DCW, VMAP0 does not provide text objects with city names, but they can be extracted from the tags.

Create a new geographic data structure of named cities using the geographic mean of the populated place outlines.

```
k = 0;
for i=1:length(PPpatch)
   str = PPpatch(i).tag;
   if isempty(strmatch('No entry present;',str))
      k=k+1;
      lat = PPpatch(i).lat;
      lon = PPpatch(i).long;
      [lat,lon] = meanm(lat(~isnan(lat)), lon(~isnan(lon)) );
      indx = findstr(';',str);
      PPtext(k).string = str(1:indx(1)-1);
      PPtext(k).type = 'text';
      PPtext(k).tag = 'Place Name';
      PPtext(k).otherproperty = [];
      PPtext(k).lat = lat;
      PPtext(k).long = lon;
      PPtext(k).altitude = [];
   end
end

hPPtext = displaym(PPtext); set(hPPtext,'FontSize',6)
set(handlem('allline'),'Color',[.5 .5 .5])
```

Here is the result:

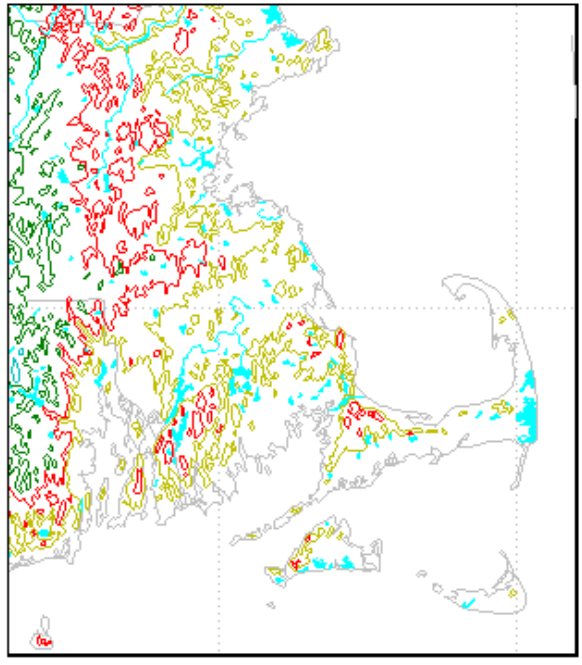


There is more data describing man-made features, including airports, utilities and cultural landmarks. The DCW also contains data for the natural world.

The drainage and elevation contours are usually the most extensive data themes.

```
clma
[DNpatch,DNline,DNtext] = vmap0data(devicename,'NOAMER',...
     [41 44],[-72 -69],'hydro',{'patch','line','text'});
ELline = vmap0data(devicename,'NOAMER',...
     [41 44],[-72 -69],'elev','line');

hBNDline = displaym(BNDline);
hDNline = displaym(DNline);
hDNpatch = displaym(DNpatch);
hELline = displaym(ELline);

set(hBNDline,'Color',[1 1 1]*0.75)
set(hDNline,'Color','c');
set(hDNpatch,'FaceColor','c','EdgeColor','none')
```

Here is the displayed map:



Reading VMAP0 Files Directly

The high-level `vmap0data` function is the most convenient way to read data from the VMAP0 because it integrates information from many different files in different directories. Individual VMAP0 files are less useful because few of them contain complete information. Most people have little need to read the raw VMAP0 data files. Still, the Mapping Toolbox can read most VMAP0 files into MATLAB structures using the `vmap0read` function. The "External Data Reference" section of the *Mapping Toolbox Reference Guide* has more information on directly accessing individual VMAP0 files.

More information on the relationships between the files and the meaning of the data in the files can be found in the U. S. Government "Military Specifications and Standards."

Global Self-consistent Hierarchical High-resolution Shoreline

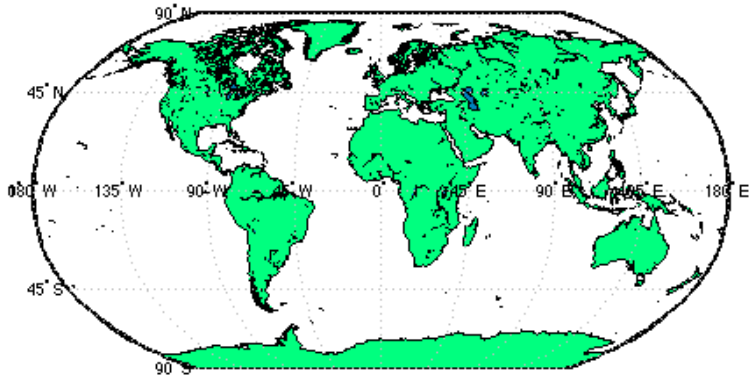
Another source of high-resolution coastline data is the Global Self-consistent Hierarchical High-resolution Shoreline (GSHHS). The GSHHS contains coastlines, major rivers, and lakes for the entire globe. The data requires 85 megabytes of storage at full resolution, but four successively lower resolution versions are also available. The GSHHS was developed by Paul Wessel of the University of Hawaii and Walter H. F. Smith of the NOAA Geosciences. The sources of data were the CIA World Data Bank II, designed for a scale of about 1:2,000,000, and the NOAA World Vector Shoreline, which is at a scale of 1:250,000.

The GSHHS data in the various resolutions is available over the Internet from <ftp://ftp.ngdc.noaa.gov/MGG/shorelines> and <ftp://kiawe.soest.hawaii.edu/pub/wessel/gshhs/>. These are binary files compressed using gzip. You should download the files in binary mode and decompress them. No byte swapping or line ending conversion is needed.

You can read any of the GSHHS data files with the `gshhs` function. GSHHS files have filenames of the form '`gshhs_X.b`', where `X` is one of the letters `c`, `l`, `i`, `h`, and `f`, corresponding to increasing resolution (and file size).

Read the entire crude resolution version and display it.

```
s = gshhs('gshhs_c.b');
worldmap('world','none')
displaym(s)
colormap(winter)
```

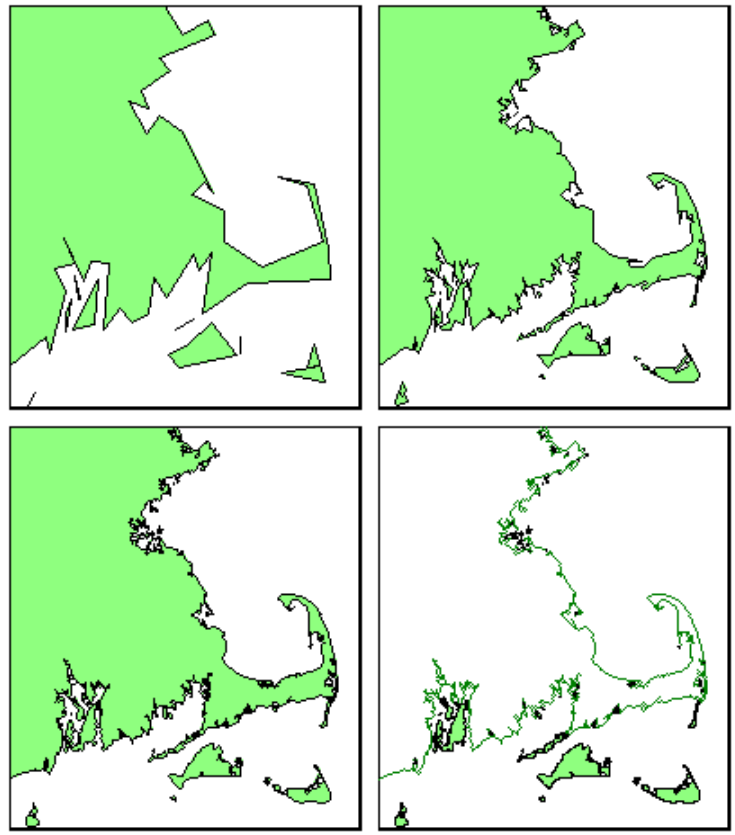


The higher resolution versions of the data are better suited for maps covering smaller regions. Here is a comparison of the other four data files. Before extracting the data, be sure to create an index file using the 'createindex' option. This option writes a file that allows gshhs to read much faster. You only need to do this once.

```
latlim = [41.13 42.75]; lonlim = [-71.7 -69.8];
figure

subplot('position',[.01 .51 .48 .48])
axesm('mercator','MapLatLimit',latlim,'MapLonLimit',lonlim,...
      'Frame','on')
gshhs('gshhs_l.b','createindex')
s = gshhs('gshhs_l.b',latlim,lonlim); displaym(s); tightmap

subplot('position',[.51 .51 .48 .48])
axesm('mercator','MapLatLimit',latlim,'MapLonLimit',lonlim,...
      'Frame','on')
gshhs('gshhs_i.b','createindex')
s = gshhs('gshhs_i.b',latlim,lonlim); displaym(s); tightmap
```

```
subplot('position',[.01 .01 .48 .48])
axesm('mercator','MapLatLimit',latlim,'MapLonLimit',lonlim,...
      'Frame','on')
gshhs('gshhs_h.b','createindex')
s = gshhs('gshhs_h.b',latlim,lonlim); displaym(s); tightmap

subplot('position',[.51 .01 .48 .48])
axesm('mercator','MapLatLimit',latlim,'MapLonLimit',lonlim,....
      'Frame','on')
gshhs('gshhs_f.b','createindex')
s = gshhs('gshhs_f.b',latlim,lonlim); displaym(s); tightmap
```

Here are the results:



The data is returned as geographic data structure of filled patches with the tags 'land', 'lake', 'island' for an island in a lake, or 'pond' for a pond on an island in a lake. The full resolution version displays the continent as a line rather than as a filled patch, because of the memory required to trim a polygon with more than a million points. The high resolution version is virtually indistinguishable in shape from the full version and requires only 20 megabytes to store.

USGS On-Line Coastline Extractor

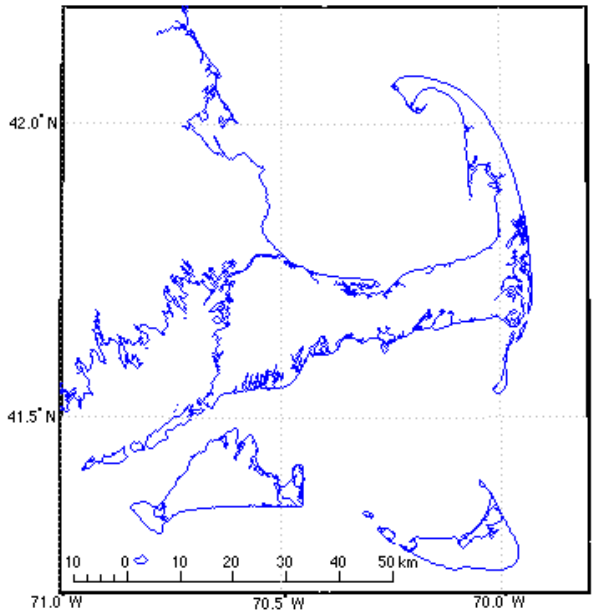
There is also an on-line source for a variety of coastline data. The U. S. Geological Survey has a Web page that allows you to extract coastlines, political boundaries, rivers, and lakes for a selected quadrangle, and download the data in a MATLAB compatible text file. You can extract data from the NOAA/NOS Medium Resolution Coastline (U. S. coverage only, designed for a scale of 1:70,000), the World Vector Shoreline (designed for 1:250,000), the CIA World Data Bank II (designed for 1:2,000,000), and the World Coast Line (designed for 1:5,000,000). The location of the Web page is

<http://crusty.er.usgs.gov/coast/getcoast.html>.

Here is the NOAA/NOS Medium Resolution Coastline for Cape Cod. The data was saved in the file, 15162.dat.

```
load 15162.dat
whos
  Name         Size          Bytes  Class

  X15162   21087x2          337392  double array

usamap([41.2 42.2],[-71 -69.8],'none')
plotm(X15162(:,2),X15162(:,1)); scaleruler
```

Here is the displayed map:



U.S. Vector Data

One of the best sources of public vector data for the United States is the U.S. Census Bureau. This organization has developed a digital database and system for automated mapping in support of the United States decennial census. The Topographically Integrated Geographic Encoding and Referencing (TIGER) system is used within the Bureau to carry out and analyze the census. TIGER contains hydrographic features, roads, railroads, and other transportation features scanned from 1:100,000 U.S. Geological Survey (USGS) topographic maps. The database also contains census tracts and political boundaries for states, counties, and indian reservations; it does not include topography. The Census Bureau periodically publishes extracts from the TIGER database for public use in the form of TIGER/Line and TIGER thinned boundary files. The Mapping Toolbox provides interfaces to both formats.

TIGER/Line

TIGER/Line files are extracts from the U.S. Census Bureau's TIGER map database. Each set of files covers one municipal district and contains transportation features such as roads and railroads, cultural features, hydrographic features, and political boundaries. The data is distributed on CD-ROMs by the Census Bureau. TIGER/Line data is not copyrighted, but the Bureau does charge for the production of the CDs. Ordering and other information can be found from the U.S. Census Bureau's Web site at <http://www.census.gov/>, along with sample TIGER/Line files. CD-ROMs may also be accessible over the Internet through various organizations.

You can import TIGER/Line data into MATLAB using the `tgrline` function. Files can be several thousand lines long, and reading them may require considerable time and memory. Once the data has been read and processed, you may want to save the results into a MATLAB workspace MAT-file for future use.

The following example reads the Washington, D.C., data from the 1994 TIGER/Line edition. These files are available over the Internet from The MathWorks at <ftp://ftp.mathworks.com/> and are delivered as text files in a compressed PC Zip archive. When extracting, make sure the text files are formatted for your computer platform. Generally, the names of the TIGER/Line datasets begin with the string 'tgr' and contain the 5-digit Federal Information Processing Standards (FIPS) code for the state and county.

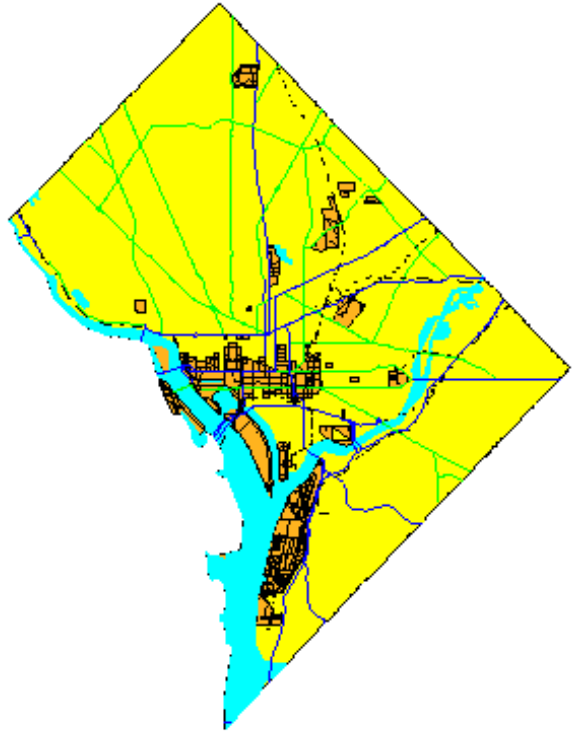
The dataset for the District of Columbia is 'TGR11001':

```
[CL,PR,SR,RR,H,AL,PL] = tgrline('TGR11001');
whos
   Name            Size      Bytes    Class

   AL              45x1     112952    struct array
   CL               1x1       7444    struct array
   H               16x1      25030    struct array
   PL              64x1      59730    struct array
   PR              1x32      53534    struct array
   RR               1x8      16632    struct array
   SR              1x40      76870    struct array
```

The `tgrline` function returns Mapping Toolbox geographic data structures for the county line (CL), primary roads (PR), secondary roads (SR), railroads (RR), hydrography (H), area landmarks (AL), and point landmarks (PL). This is a subset of the total available data in the TIGER/Line files, which contain more detailed local roads and other geographic reference data, such as Zip codes and census tract numbers, that are not handled by the function.

Since the data is now formatted into geographic data structures, you can display the objects using `displaym`. A Mercator projection is appropriate for small-scale, regional maps:

```
axesm mercator
displaym(CL)
hh = displaym(H); set(hh,'EdgeColor','None')
displaym(AL); displaym(PL)
displaym(PR); displaym(SR); displaym(RR)
```



You can also produce the same map with graphical user interfaces. Using the `maptool` application, invoke the `mlayers` tool by selecting **Session⇒Layers⇒Workspace** from the menu bar, or invoke `mlayers` from the command line by typing `rootlayr; mlayers(ans)`.

The amount of detail in the data becomes apparent when you zoom into the federal district of Washington, D.C. While the map has much detail, the street

patterns contain displaced points, and the number of points in the paths of the streets is not enough to give smooth representations of slowly curving streets.

If you hold down the mouse button on an object, you will see its name appear in the lower left part of the figure window. All objects have been tagged with a name. Here are the names of places in the hydrography layer:

```
unique(strvcat(H.tag),'rows')

ans =
Anacostia River
Channel
Dalecarlia Reservoir
Georgetown Reservoir
Kenilworth Aquatic Gardens
Kingman Lake
Lagoon
Mc Millan Reservoir
Potomac River
Reflecting Pool
Tidal Basin
Washington Channel
```

The other layers have tags with street names, interstate highway numbers, railroad names, and landmarks names. The names can be useful in finding the location of a site or in the selective display of data. What is the latitude and longitude of the Washington Monument? Mark the spot:

```
[lat,long,indx] = extractm(PL,'washington mon');
PL(indx)
ans =
              lat: 38.8891
             long: -77.0354
              tag: 'Washington Monument'
             type: 'line'
         altitude: 0.4000
    otherproperty: {1x3 cell}

h = displaym(PL(indx)); set(h,'Marker','x','Color','r')
```

You can also use the `mobjects` tool to browse the names of objects that have already been plotted.

TIGER Thinned Boundary

The U.S. Census Bureau has published lower resolution, thinned extracts of the TIGER database in two data file formats: ARC/INFO and MapInfo Interchange Format (MIF). ARC/INFO and MapInfo are geographic information systems (GIS) developed by the Environmental Systems Research Institute, Inc. (ESRI) and MapInfo Corporation, respectively. Data files for both formats are provided by the U.S. Census Bureau via the Internet <http://www.census.gov/>, along with further descriptions and documentation.

TIGER ArcInfo Format

The following boundary types are available in the ArcInfo format: state, county, minor civil division, census tract/block numbering area, American Indian reservation/Alaska native village statistical area, Alaska native regional corporation, urbanized area, metropolitan area, and congressional district.

The ArcInfo data files have names ending in 'p' and 'pa'; the first represents polygon coordinate data, and the latter polygon attributes. You will also need the '_name.dat' file, or *names* file, containing the FIPS codes and corresponding county names.

Plot the counties in the state of Alaska as an example. You will need the following three files: 'co_name.dat,' 'co_02_p.dat,' and 'co_02_pa.dat.' For these files, 'co' indicates county boundaries, and '02' is the code for Alaska. The code '99' is used for files within the continental United States, and '15' corresponds to Hawaii. The files are compressed in a PC Zip format. When extracting, make sure the text files are formatted for your computer platform.

Read all U.S. county names and FIPS code from the names file. There are more than 3000 entries, so reading the file may take a while:

```
namestruc = fipsname('co_name.dat')
namestruc =
1x3248 struct array with fields:
  name
  id
```

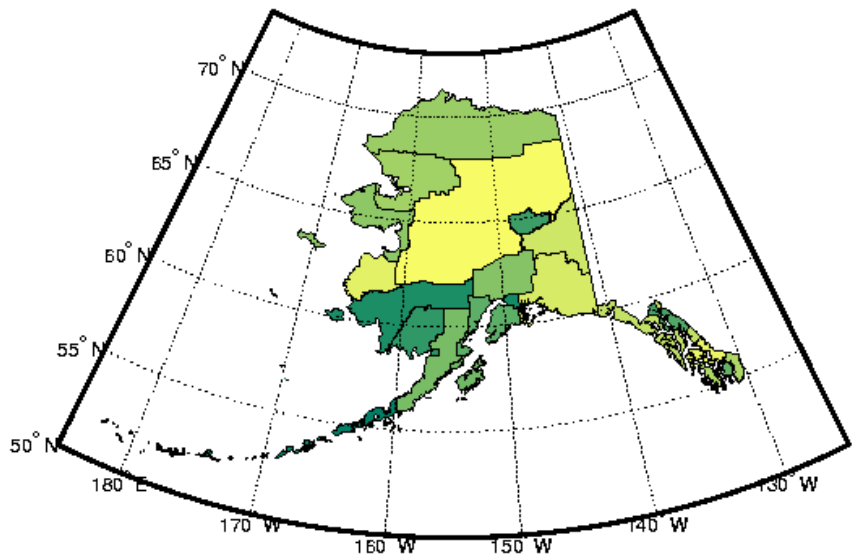
You can query the FIPS code of a particular city or county in Alaska, Anchorage, for example, by the following commands:

```
indx = strmatch('Anchorage',{namestruc.name})
indx =
  70
code = namestruc(indx).id
code =
  2020
```

Use the `tigerp` function to read and process Alaska's thinned county boundaries data into Mapping Toolbox geographic data structures.

Plot the data in a Lambert Conformal Conic projection:

```
[COpatch,COtext] = tigerp(namestruc,'co_02_p.dat');
axesm('MapProjection','lambert',...
  'MapLatLimit',[50 73],'MapLonLimit',[175 235] ,...
  'MLabelLocation',10,'MLineLocation',10,...
  'PLabelLocation',5,'PLineLocation',5,...
  'MLabelParallel','south','MapParallels',[]);
framem; gridm; mlabel; plabel
displaym(COpatch)
colormap(summer)
```



There is one entry in the `COpatch` structure for each county, with the county name in the tag field. The `COtext` structure contains the text labels for the counties.

You can also append new data to existing structures or extract only the counties you want. The following example reads the county boundaries for Nome and then adds the boundaries for the Aleutians East and Aleutians West to the output structure. You can look in the `namestruc` structure to find the codes you need.

```
COpatch = tigerp(namestruc,'co_02_p.dat',[],[],2180);
COpatch = tigerp(namestruc,'co_02_p.dat',COpatch,[],...
  [2013 2016])
COpatch =
1x3 struct array with fields:
  lat
  long
  type
  otherproperty
  altitude
  tag
```

TIGER MapInfo Interchange Format

The U.S. Census Bureau provides thinned boundary files in the MapInfo Interchange Format. Both U.S. state and county boundaries are available in this format.

The MapInfo data files have the 'mif' and 'mid' extensions. You will also need the '_name.dat' file, or names file, containing the Federal Information Processing Standard (FIPS) code numbers and corresponding county names.

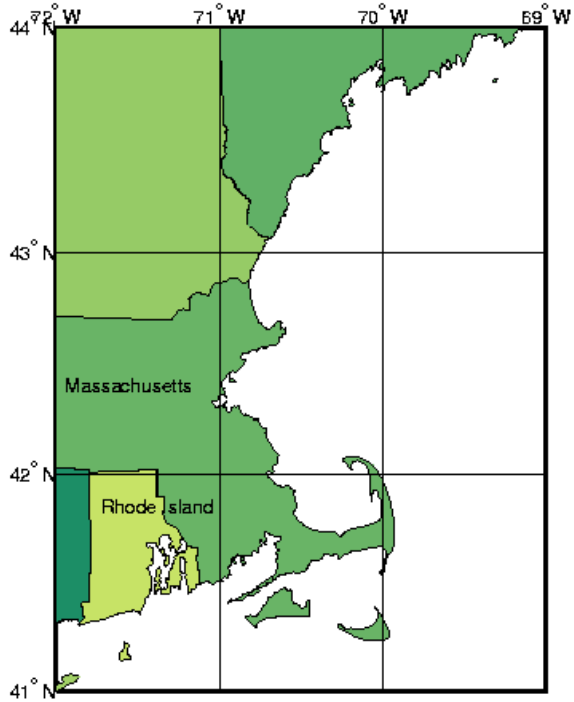
The following example reads the state boundaries of Alaska and Hawaii. Five files are needed: 'ST02.MID,' 'ST02.MIF,' 'ST15.MID,' 'ST15.MIF,' and 'st_name.dat.' You must first read the names file for the names and FIPS codes for the U.S. states and territories:

```
namestruc = fipsname('st_name.dat')
namestruc =
1x57 struct array with fields:
  name
  id
```

Read the file containing Hawaii's county boundaries into a geographic data structure, and append it with Alaska's state boundaries:

```
STpatch = tigermif(namestruc,'ST15.MIF');
STpatch = tigermif(namestruc,'ST02.MIF',STpatch)
STpatch =
1x2 struct array with fields:
   lat
   long
   type
   otherproperty
   tag
   altitude
```

Finally, to compare the resolution and quality of this data to other sources, look at the region around Cape Cod. This map may look familiar to you; it is the same data in the `usahi` workspace, plotted in a Mercator projection. The `usahi` data was derived from the Census Bureau's TIGER Thinned Boundary files.



Global Matrix Data

Some types of geographic data are provided as matrices of values on a geographic grid. The Mapping Toolbox provides interfaces to gridded elevation and bathymetry at resolutions ranging from 10 kilometers to 30 meters, geoid heights, and vegetation index and classification data. There are also functions to help you write your own interface functions.

ETOPO5 and TerrainBase

ETOPO5 and TerrainBase are digital elevation models with worldwide coverage at a resolution of 5 minutes (10 kilometers or better). They are both compilations of data from a variety of different sources, including the U.S. Naval Oceanographic Office, U.S. Defense Mapping Agency, U.S. Navy Fleet Numerical Oceanographic Center, Bureau of Mineral Resources in Australia, and the Department of Industrial and Scientific Research in New Zealand. The ETOPO5 dataset was assembled by Margo Edwards at Washington University, in St. Louis, Missouri.

An overview of this dataset can be found at several Internet locations, including the U.S. Geological Survey Web site located at <http://edcwww.cr.usgs.gov/> and the U.S. National Geophysical Data Center home page at <[http://www.ngdc.noaa.gov/>.

The ETOPO5 data is read using the `etopo5` external interface function for the version of the database contained in the two text files, 'etopo5.southern.bat' and 'etopo5.northern.bat.' These files occupy about 60 megabytes of total disk space when uncompressed and are provided via FTP by The MathWorks, Inc. at <ftp://ftp.mathworks.com/>.

ETOPO5 has been superseded by TerrainBase, which is similar in coverage to ETOPO5, but corrects many of its flaws, including discontinuities at joins between datasets, systematic shifts in location, and lack of data in some regions. TerrainBase is a compilation of the best available, public domain data from almost 20 different sources, including ETOPO5 and DCW-DEM, another digital elevation model that has a Mapping Toolbox interface, covered in the next section.

The TerrainBase dataset was created by the National Geophysical Data Center and World Data Center-A for Solid Earth Geophysics in Boulder, Colorado. The model is currently under development and will be updated as new data sources become available. Further information on the TerrainBase DEM can be found at the U.S. National Geophysical Data Center home page at <http://www.ngdc.noaa.gov/>.

TerrainBase data is read using the `tbase` function. The data files are available on CD-ROM and via Internet. NOAA/NGDC provides ordering information and downloadable files at their Web site. The files require about 18 megabytes of disk space when uncompressed.

You should use the TerrainBase dataset unless you have a good reason to want ETOPO5. TerrainBase corrects some errors, takes up less space, and can be read faster than ETOPO5. The following examples will read TerrainBase data, but you can substitute ETOPO5 data by simply using the `etopo5` function instead of `tbase`.

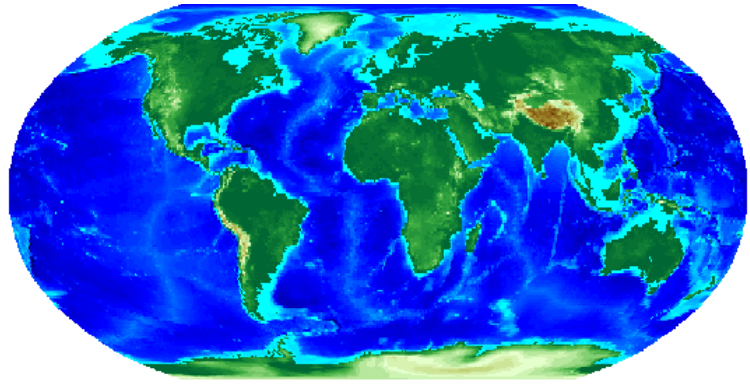
A 10 kilometer resolution DEM has so much data that it is usually impractical to read the entire file into memory. The full matrix has 2160 by 4320 elements. Both the `etopo5` and `tbase` functions allow you to reduce the resolution by discarding data points.

Read every twelfth point in the global data base.

```
[map,maplegend] = tbase(12);
whos
   Name            Size      Bytes    Class

   map          180x360     518400    double array
   maplegend        1x3         24    double array
```

Plot the map on a Robinson projection:

```
axesm robinson
meshm(map,maplegend); demcmap(map);
```



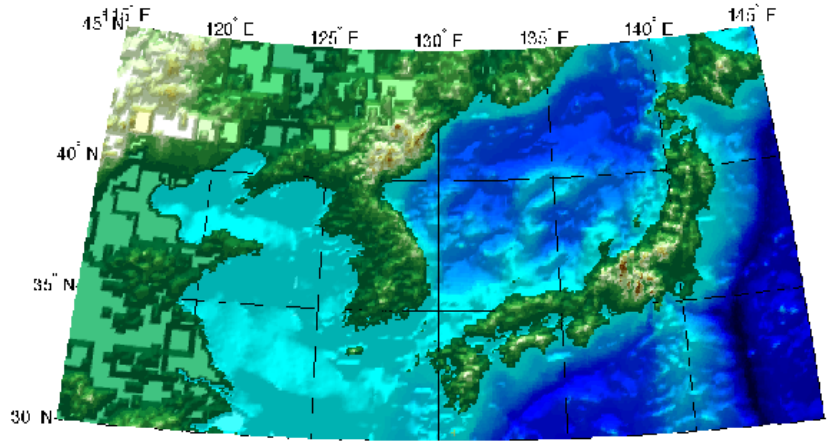
This is a regular matrix map at the same resolution as the `topo` map. It differs from `topo` in that the values represent nearest neighbor samples, rather than averages over the matrix cell, resulting in a rougher appearance.

Elevations and depths are given in meters above or below mean sea level. Some parts of the world are represented by data with a horizontal resolution as coarse as 1 degree by 1 degree. The vertical resolution varies from 1 meter for Australia and New Zealand to as much as 150 meters for parts of Africa, Asia, and South America. Oceanographic data in areas shallower that 200 meters contains little detail because of the way depth contours were converted to gridded depths.

The areas with 1 degree horizontal data can be clearly seen when you display data for a small region at the full resolution of the database.

Display the Korean peninsula and Japan:

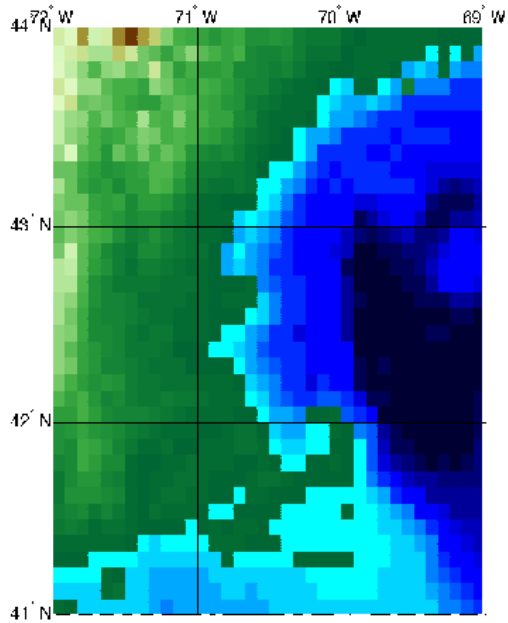
```
[map,maplegend] = tbase(1,[30 45],[115 145]);
axesm('MapProjection','polycon',...
  'MapLatLimit',[30 45],'MapLonLimit',[115 145],...
  'MLabelLocation',5,'MLineLocation',5,...
  'PLabelLocation',5,'PLineLocation',5,...
  'GLineStyle','-','GAltitude',inf) ;
gridm; mlabel; plabel
meshm(map,maplegend,size(map),map); demcmap(map);
lightm(37.5,130); material([.7 .7 1.5]); lighting gouraud
```



Note the discontinuities in some areas, particularly the regions in China.

You can get a sense for the resolution of this dataset by comparing a TerrainBase map of the Cape Cod region in the Northeastern United States with maps of the same region using other datasets. TerrainBase and ETOPO5 have a horizontal resolution of 12 cells per degree, so the Cape Cod map consists of 36 by 36 cells.

```
[map,maplegend] = tbase(1,[41 44],[-72 -69]);
whos
   Name            Size      Bytes    Class

   map            36x36      10368    double array
   maplegend        1x3         24    double array

axesm('Mapprojection','mercator',...
  'MapLatLimit',[41 44],'MapLonLimit',[-72 -69],...
  'MLabelLocation',1,'MLineLocation',1,...
  'PLabelLocation',1,'PLineLocation',1,...
  'GLineStyle','-','GAltitude',inf) ;
gridm; mlabel; plabel
meshm(map,maplegend); demcmap(map)
```



Global Bathymetry Predicted from Satellite Altimetry

High resolution bathymetric data has been lacking for much of the world because few ships made sounding measurements away from heavily used sea lanes. Recently declassified satellite altimetry revealed that the topography of the ocean floor is mirrored in undulations of the ocean surface. Sandwell and Smith have developed a bathymetric model that combines ship soundings and satellite altimetry to infer the topography of the ocean floor. Land data is taken from the GTOPO30 elevation data, described below. The global topography is stored in a Mercator projection at a nominal resolution of 2 arc-seconds (or about 4 kilometers), covering the globe to 72 degrees north and south.

The data is available over the Internet via anonymous FTP from

<ftp://topex.ucsd.edu/pub/global_topo_2min/topo_6.2.img>.

This function reads the binary files as they are, i.e., you should not use byte swapping software on these files. Some documentation is also available over the World-Wide-Web at

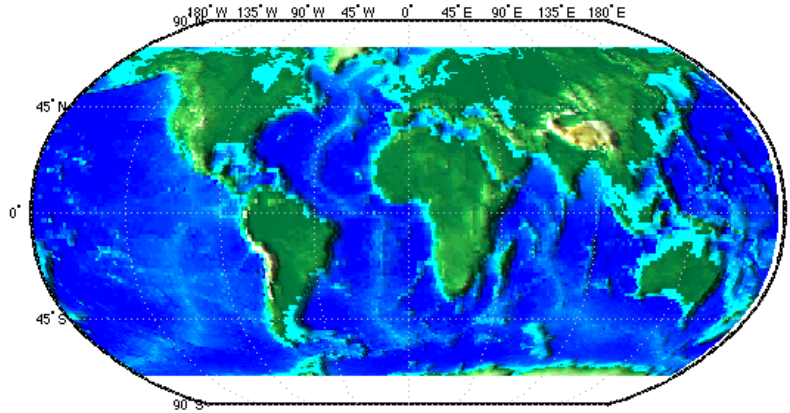
<http://topex.ucsd.edu/marine_topo/mar_topo.html>

and

<http://www.ngdc.noaa.gov/mgg/announcements/announce_predict.html>.

The satbath function reads the data into a general matrix map that can be displayed in any projection. Here is the entire file reduced by a factor of 50. You can see the latitude limits in the data.

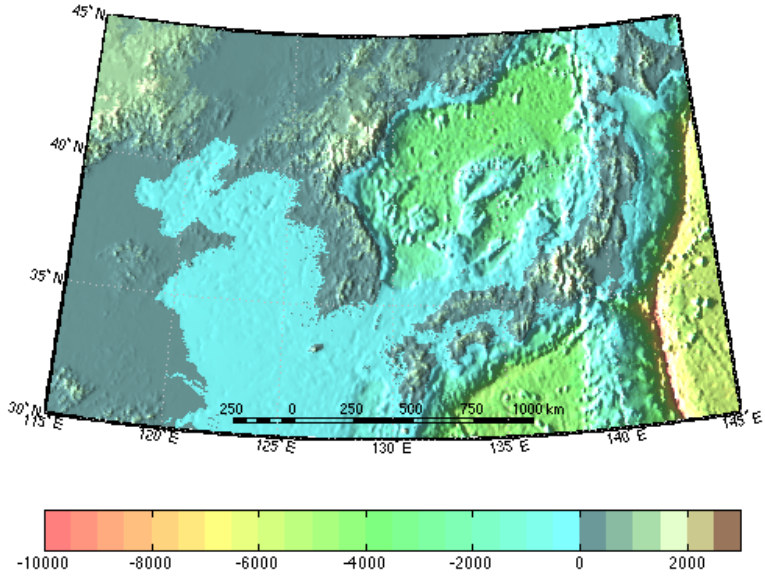
```
[latgrat,longrat,map] = satbath;

worldmap('world','none'); setm(gca,'MLabelParallel','n')
surfm(latgrat,longrat,map,map)
demcmap(map); daspectm('m',30)
camlight(-80,0); lighting phong; material([.3 5 0])
```



The richness of the data is evident in a map at close to the full resolution. You can bring out the small features of the data with a lightened hsv colormap for ocean depths, and a lighter version of the standard land colormap. This colormap is similar to one used in posters of this data by Sandwell and Smith.

```
latlim = [30 45]; lonlim = [115 145];
[latgrat,longrat,map] = satbath(2,latlim,lonlim);

worldmap(latlim,lonlim,'none')
surfm(latgrat,longrat,map,map)

landcmap = hsv2rgb([0.5 0.3 0.6; 0.25 0.2 1; 0.07 0.4 0.6]);
seacmap = [  1   0.5   0.5
             1   1     0.5
           0.5   1     0.5
           0.5   1     1  ];

demcmap('inc',map,500,seacmap,landcmap); daspectm('m',5)
material([0.4 3 0]); camlight(-80,0); lighting phong
scaleruler('MajorTick',0:250:1000,'MinorTick',0:50:250,...
           'RulerStyle','patches')
colorbar('horiz')
```

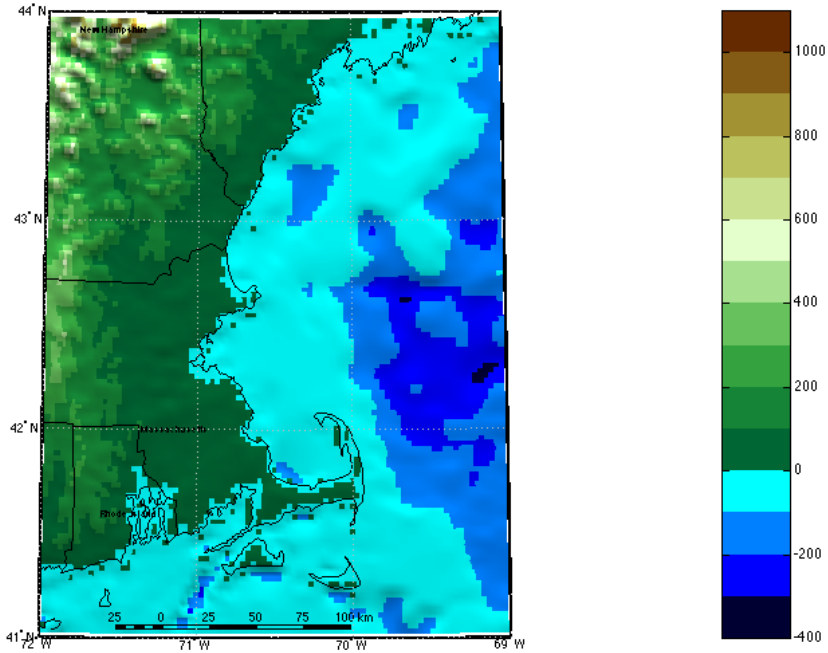
Here is the result:



The data at the limit of resolution can be seen in this map of Cape Cod. The position of the land-sea interface is off in some places. This may be because the dataset uses odd or even numbers to indicate the source of the data.

```
latlim = [41 44]; lonlim = [-72 -69];
[latgrat,longrat,map] = satbath(1,latlim,lonlim);

usamap(latlim,lonlim,'line')
zdatam('alltext',1100); zdatam('allline',1100)
surfm(latgrat,longrat,map,map)
demcmap('inc',map,100); daspectm('m',5)
material([0.4 3 0]); camlight(0,80); lighting phong
scaleruler('MajorTick',0:25:100,'MinorTick',0:5:25,...
           'RulerStyle','patches')
colorbar
```

Here is the displayed map:



GTOPO30

GTOPO30 is another source of gridded land elevation data at a relatively high resolution. The dataset provides global coverage of land elevations at a horizontal resolution of 30 arc-second (about 1 kilometer or better). The DCW

DEM, the precursor to GTOPO30, was based on aeronautical charts; hence, no bathymetric data was available. It was derived primarily from DTED0 and the Digital Chart of the World (DCW). GTOPO30 was developed by the Earth Resources Observation Systems (EROS) Data Center. Where no existing grid was available, elevations were computed using the Australian National University's ANUDEM computer program. This program combines elevation and drainage lines and points to interpolate elevations at the grid locations.

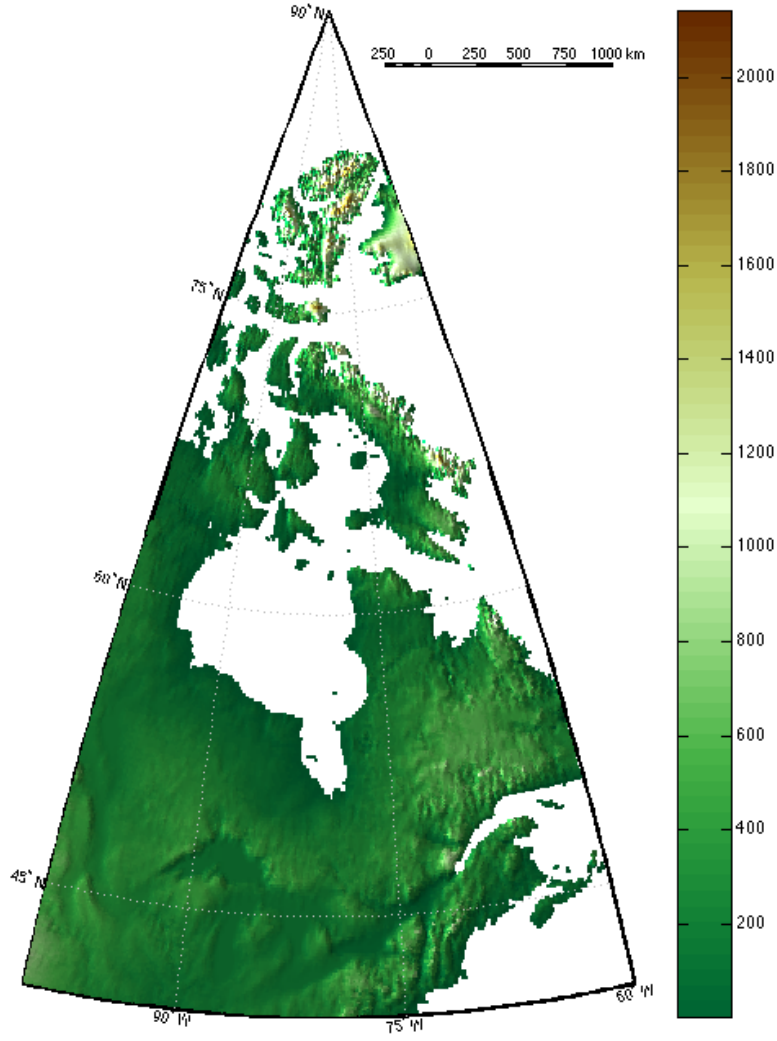
GTOPO30 is accessible over the Internet and is available on CD-ROM. Ordering information, as well as the actual data files can be found on-line. The data is available over the Internet via anonymous FTP from <ftp://edcftp.cr.usgs.gov/pub/data/gtopo30/global>. The data and some documentation are also available over the World-Wide-Web from <http://edcwww.cr.usgs.gov/landdaac/gtopo30/gtopo30.html> and <http://edcwww.cr.usgs.gov/landdaac/gtopo30/README.html>. The dataset consists of a total of 21,600 rows and 43,200 columns broken into 33 tiles, each generally covering 40-by-50 degrees. Each tile requires about 15 megabytes of storage when compressed, 80 megabytes uncompressed, and 160 megabytes to extract.

The Mapping Toolbox interface to GTOPO30 files is the `gtopo30` function. It reads the elevations into regular matrix maps, optionally reducing and trimming data in the process. The following examples show the '`W100N90`' tile, which covers the northeastern United States and eastern Canada. The files are available from the <http://edcwww.cr.usgs.gov/landdaac/gtopo30/gtopo30.html> Web page.

Read the entire elevation file, taking every twentieth point. Display the data as a lighted surface in a map axes prepared by `worldmap`. `worldmap` selects a polyconic projection for these map limits. Add a graphic scale and colorbar.

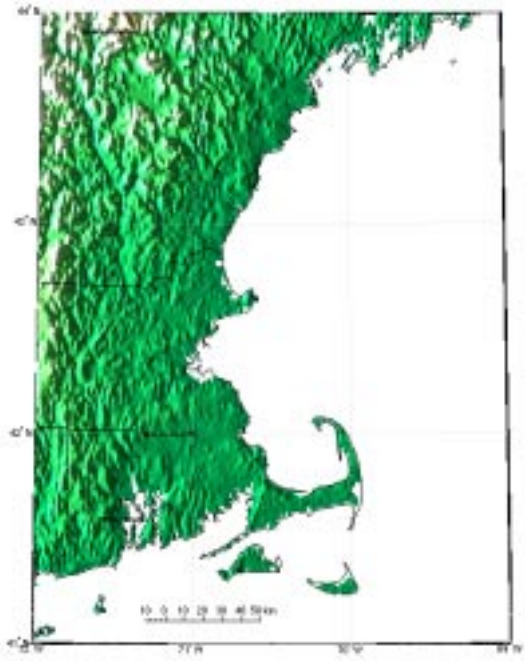
```
[map,maplegend] = gtopo30('W100N90',20);
[latlim,lonlim] = limitm(map,maplegend);

 worldmap(latlim,lonlim,'none')
meshm(map,maplegend,size(map),map);demcmap(map)
demcmap(map); daspectm('m',15)
camlight(-80,0); lighting phong; material([.5 5 0])
scaleruler('MajorTick',0:250:1000,'MinorTick',0:50:250,...
           'RulerStyle','patches')
colorbar
```

Here is the result:

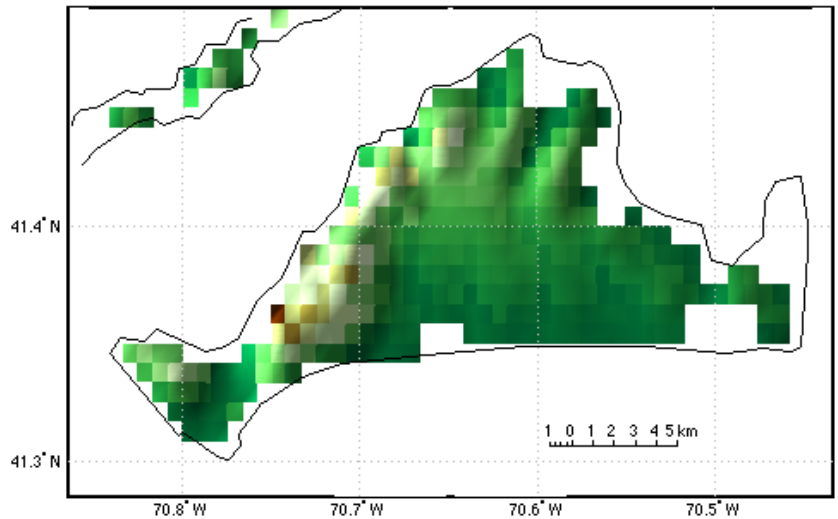


You can read a subset of the data by providing latitude and longitude limits. Here is the part of the dataset covering Cape Cod at the full resolution. The 30 arc-second GTOPO30 covers this region with a 360-by-360 matrix, which is 10 times finer than the 5 arc-minute (300 arc-second) ETOPO5 and TerrainBase data. This is also the same resolution as the data in the MATLAB cape workspace.

```
latlim = [41 44]; lonlim = [-72 -69];
[map,maplegend] = gtopo30('W100N90',1,latlim,lonlim);

figure
usamap(latlim,lonlim,'line')
zdatam('alltext',max(map(:)))
zdatam('allline',max(map(:)))
meshm(map,maplegend,size(map),map);demcmap(map)

demcmap(map); daspectm('m',15)
camlight(-80,0); lighting phong; material([.5 5 0])
scaleruler; colorbar
```



To see the ultimate resolution of the GTOPO30 data, extract only the area around Martha's Vineyard. Now you can see the individual pixels, and how the ocean areas are transparent. The ocean is coded as *no data*, and filled with `NaN`s.



Digital Terrain Elevation Data Model

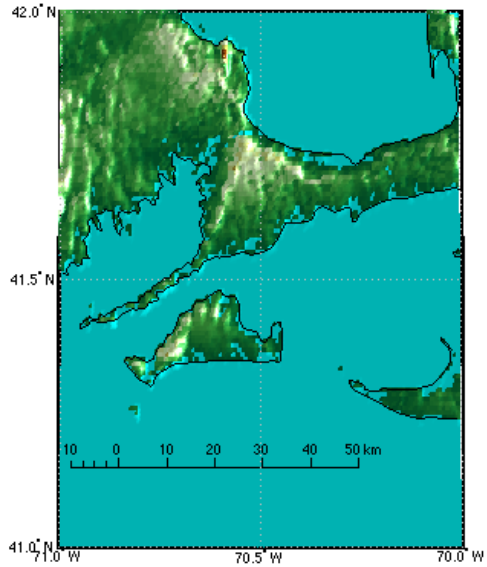
The Digital Terrain Elevation Data (DTED) Model is a series of gridded elevation models with global coverage at resolutions of 1 kilometer or finer. It is a product of the U. S. National Imagery and Mapping Agency (NIMA), formerly the Defense Mapping Agency (DMA). The data is provided as 1-by-1 degree tiles or elevations on geographic grids with product-dependent grid spacing.

The lowest resolution data is the DTED Level 0, with a grid spacing of 30 arc-seconds, or about 1 kilometer. This version of the data has been made available over the Internet at <http://164.214.2.59/geospatial/products/DTED/dted.html>. The intended coverage is global, but there are currently large gaps in coverage for South America and Africa. DTED Level 1 has a resolution of 3 arc-seconds, or about 100 meters, and was the primary source for the USGS 1:250,000 (1 degree) DEMs. DTED Level 2 and above are at even higher resolutions. DTED Level 1 and above are available to the U. S. Department of Defense and its contractors from NIMA.

The DTED files are binary. No line ending conversion or byte swapping is required. The files have filenames with the extension 'dtN', where N is the level of the DTED product.

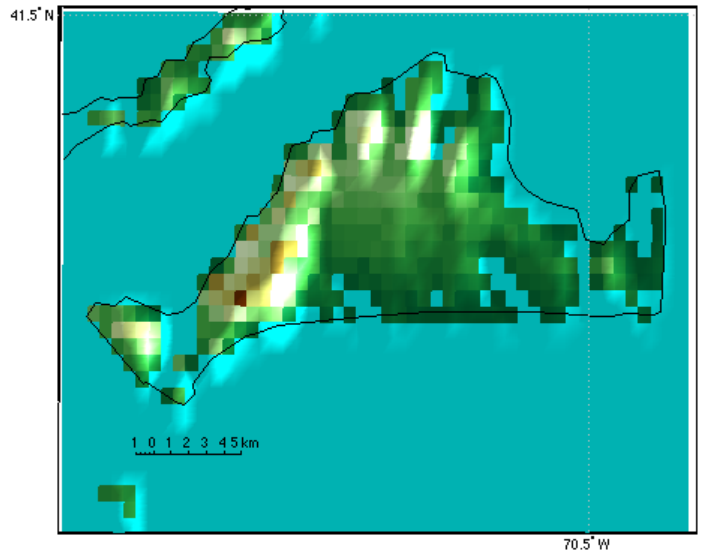
Retrieve the DTED0 data for Cape Cod from the Web page, and read the 1-degree tile at full resolution. The data is provided in a tar file named 'dn41w071.tar'. The DTED0 elevation data file is in the 'w071' directory under the name 'n41.dt0'. Read and display it as a lighted surface to show both large and small scale variations in the data. Because no bathymetric depths are provided, move elevations of zero down a bit to color them blue.

```
[map,maplegend] = dted('n41.dt0'); map(map==0)=-1;
[latlim,lonlim] = limitm(map,maplegend);

usamap(latlim,lonlim,'line')
meshm(map,maplegend,size(map),map); demcmap(map)
daspectm('m',40)
zdatam('allline',max(map(:)))
light; material([0.7 1 0.6]); lighting phong
scaleruler
```



Read and display the data covering just Martha's Vineyard at the full resolution.

```
clmo surface
latlim = [41.2344 41.5053]; lonlim = [-70.8613 -70.4249];
setm(gca,'MapLatLimit',latlim,'MapLonLimit',lonlim); tightmap

[map,maplegend] = dted('n41.dt0',1,latlim,lonlim);
map(map==0)=-1;

meshm(map,maplegend,size(map),map); demcmap(map)
light; material([0.7 1 0.6]); lighting phong; daspectm('m',30)
scaleruler off; scaleruler
```



Advanced Very High Resolution Radiometer (AVHRR) Global Change Data

The National Oceanic and Atmospheric Administration (NOAA) and others have compiled a global high resolution set of land vegetation measurements to monitor global climate change. The data is derived from the Advanced Very High Resolution Radiometer (AVHRR) sensor carried on NOAA meteorological satellites (NOAA-7, -9, 11). The sensor measures reflected radiation in five different wave-bands with a 1 km spatial resolution, which is then processed into a Non-dimensional Vegetation Index (NDVI). The NDVI has been used as the basis for a global landcover classification dataset. Other applications of the AVHRR data include oceanographic and atmospheric science. Many of those datasets are stored in HDF formats, which can be read using the MATLAB `imread` or `hdfread` functions. This section focuses on the land datasets, which are stored in several different map projections and require special interface functions.

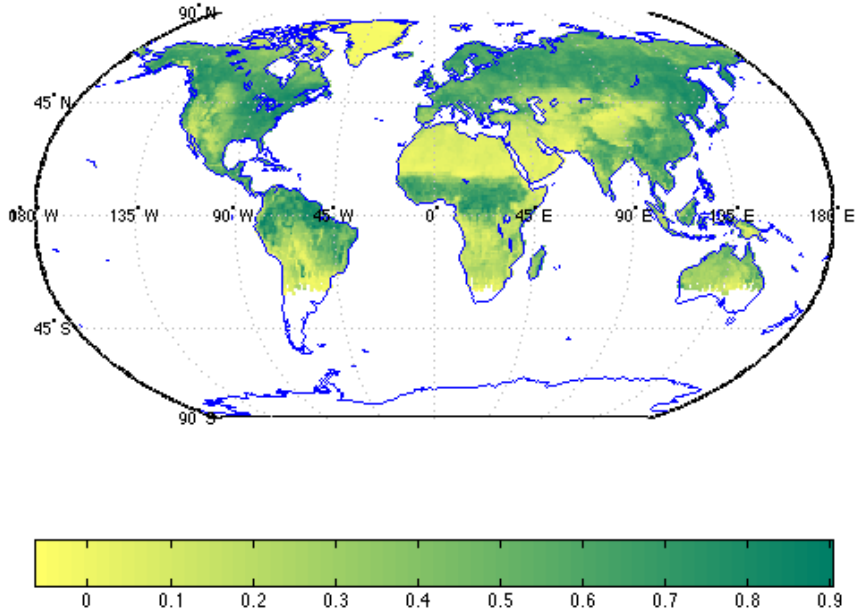
Low resolution monthly NDVI composites are available from

<ftp://daac.gsfc.nasa.gov/data/inter_disc/biosphere/avhrr_ndvi/>.

These are binary files stored as a 1 degree geographic grid. You can read these files with `readmtx`, a Mapping Toolbox function that reads matrices stored in files based on a description of the file format. Here is an example of the non-dimensional vegetation index data for September 1994. This data was retrieved from <ftp://daac.gsfc.nasa.gov/data/inter_disc/biosphere/avhrr_ndvi/1994/avhrr_pf.ndvi.1nmegl.9409.bin>.

Try the following:

```
map = readmtx('avhrr_pf.ndvi.1nmegl.9409.bin',180,360,'real*4');
map = flipud(map);
map(map<-9)=NaN;
maplegend = [1 90 -180];

worldmap('world','none'); plotm(coast);
meshm(map,maplegend,size(map))
colormap(flipud(summer))
colorbar('horiz')
```



What are the values of the non-dimensional vegetation index at particular locations?

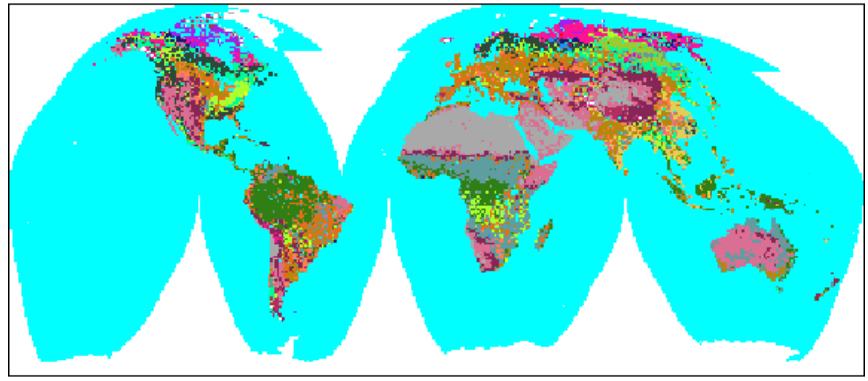
```
[lat,lon] = extractm(worldlo('gazette'),...
     {'Vancouver','Oslo','Cairo','Lagos','Tripoli'});
lat = lat(~isnan(lat)); lon = lon(~isnan(lon));
ltln2val(map,maplegend,lat,lon)
ans =
        0.712
        0.648
        0.512
        0.4
        0.128
```

Other AVHRR products are stored as matrices in projected coordinates. The standard projection for the 1 kilometer resolution data is the interrupted Goode, which can be read using the `avhrrgoode` interface function. Some of the data is also available in the Lambert Equal-Area Azimuthal projection. Extracting data from the Lambert files with `avhrrlambert` is much faster. If you have a choice, you should probably read data in the Lambert format.

The global 1 kilometers data in the Goode projection is stored as a binary matrix with 17347 rows and 40031 columns. The uncompressed file size is about 695 megabytes, slightly more than the capacity of one CD-ROM disk. Here is an image of the data as it is stored. This example uses the Global Land Cover Classification data in the USGS classification scheme, available from <ftp://edcftp.cr.usgs.gov/pub/data/glcc/globe/gusgs1_2.img.gz>. The colors correspond to different types of vegetation and land use.

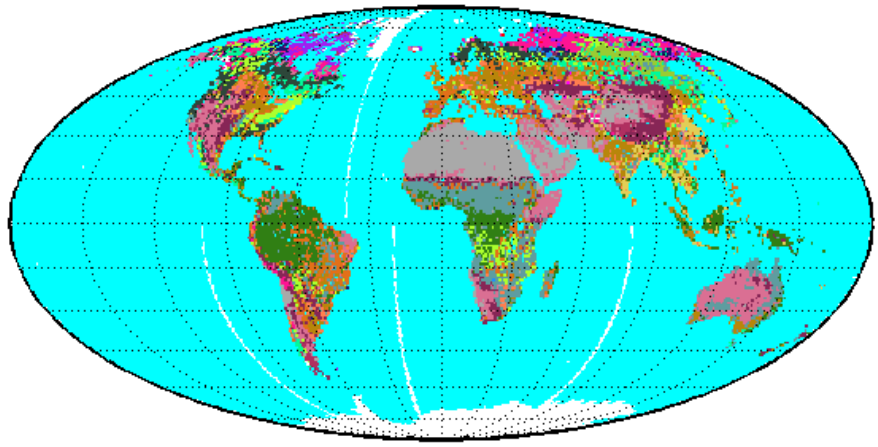
```
nrows = 17347;   % 16347 for some data on cd-rom
ncols = 40031;   % 40031
mtx = readmtx('gusgs1_2.img',nrows,ncols,'int8',...
              1:100:nrows,1:100:ncols);

h = image(mtx);
axis image

load usgslulegend
colormap(cmap); set(h,'CdataMapping','Direct')
showaxes off
```

Here is the displayed image:



To display this data on a map projection, extract the data with the avhrrgoode function. The next example reads the data at the same resolution as above, returning a general matrix map consisting of the latitude, longitude, and data matrices.

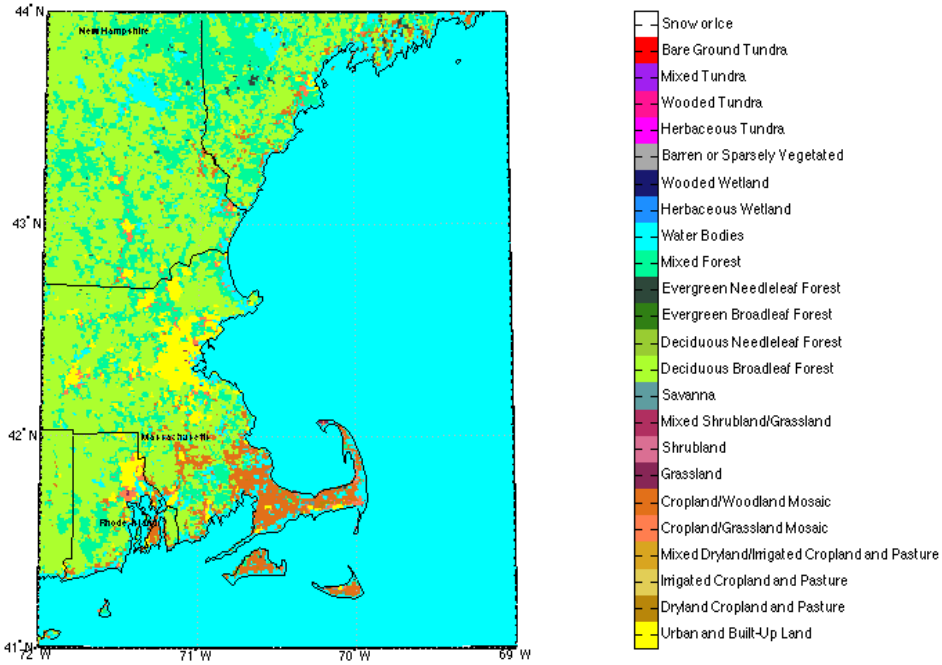
```
[latgrat,longrat,map] = avhrrgoode('global','gusgs1_2.img',100);

figure; axesm mollweid; framem; gridm;
surfm(latgrat,longrat,map); colormap(cmap)
set(handlem('surface'),'CdataMapping','Direct')
```



You can read the data for just a part of the world and control the resolution of the extracted data and the size of the graticule matrices. Data is sometimes provided in regional subsets of the very large global Goode files. You can read these by supplying the name of the region. For example, read the USGS land cover data for Cape Cod at the full resolution from the North America file,

available from <ftp://edcftp.cr.usgs.gov/pub/data/glcc/na/goode/nausgs1_2g.img.gz>.

```
latlim = [41 44]; lonlim = [-72 -69];
[latgrat,longrat,map] = avhrrgoode('north america',...
     'nausgs1_2g.img',1,latlim,lonlim);

usamap(latlim,lonlim,'line')
surfm(latgrat,longrat,map); colormap(cmap)
set(handlem('surface'),'CDataMapping','Direct')

set(gca,'Position',[.1 .1 .5 .8])
caxis([.5 24.5]); hcb = colorbar;
set(hcb,'YTick',1:24,'YTickLabel',USGSLandUse,...
    'Position',[.6 .1 .02 .8])
```

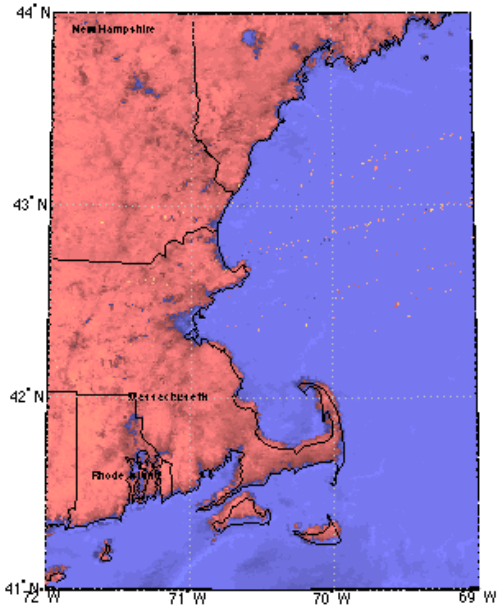


The full matrix of AVHRR data is too large to fit on one CD-ROM disk. When the data is provided on CD-ROM, the size of the data is reduced by dropping the last rows. To read the truncated files, specify the number of rows and columns in the non-standard file. This information can usually be found in the

'Readme' file. Here is the non-dimensional vegetation index data for Cape Cod from the "Global Land 1-km AVHRR Data Set – Vegetation Index 6/21-30, 1992" CD-ROM (distributed by the Land Processes Distributed Active Archive Center, EROS Data Center, Sioux Falls, South Dakota, 57198, USA). The empty matrix argument specifies that the graticule size will be the same size as the map. The geographic registration for this older CD seems to be less accurate.

```
[latgrat,longrat,map] = avhrrgoode('global','NDVI.IMG',...
     1,latlim,lonlim,[],16347,40031);

figure
usamap(latlim,lonlim,'line')
surfm(latgrat,longrat,map)

% Lighten the colormap
cmap = flipud(jet(64));
hsvmap = rgb2hsv(cmap); hsvmap(:,2) = .5; cmap = hsv2rgb(hsvmap);
colormap(cmap)
```

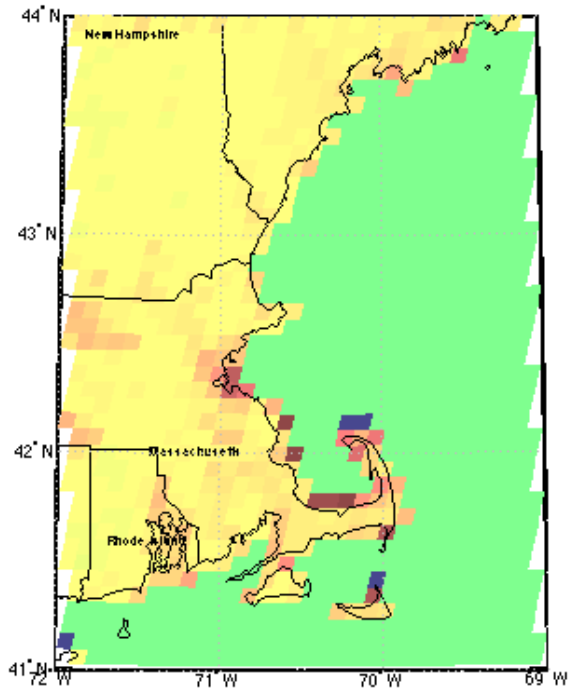
Here is the result:



AVHRR data is also available in the Goode projection at a resolution of 8 kilometers. You can read the global 8 km files with `avhrrgoode`, but not the 8 km regional files.

Read the global 8 km resolution non-dimensional vegetation index for Cape Cod. This data file is available from <ftp://daac.gsfc.nasa.gov/data/avhrr/global_8km/.1994_1997/1994/jun/avhrrpf.ndvi.1ntfgl.940621.gz>. Specify the non-standard number of rows, columns, and resolution in the file. The jagged edges in the surface are a result of trimming a slanted graticule to the map frame.

```
[latgrat,longrat,map] = avhrrgoode('global',...
     'avhrrpf.ndvi.1ntfgl.940621',1,...
     latlim,lonlim,[],2168,5004,8000);

clmo surface
surfm(latgrat,longrat,map)
```

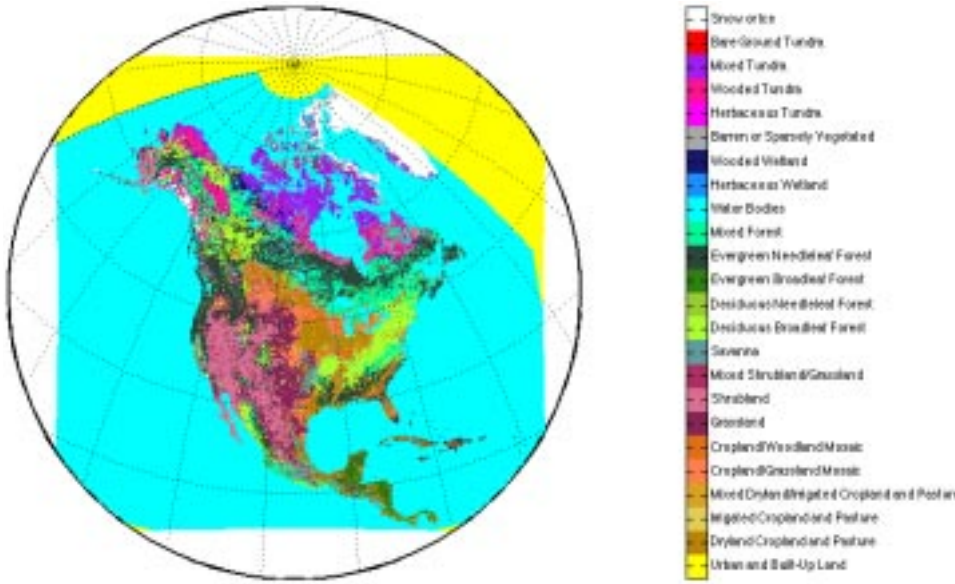


Regional versions of the 1 km Global Land Cover Classification data are also available saved in Lambert azimuthal equal area projections. These can be read using the `avhrrlambert` function. Reading data from the Lambert projection will be faster than from the interrupted Goode, because the graticule does not need to be trimmed to the boundaries of the individual projection zones.

Read the Global Land Cover Characteristics (GLCC) file covering North America in the Lambert projection with the USGS classification scheme. This file is available from <ftp://edcftp.cr.usgs.gov/pub/data/glcc/na/lambert/nausgs1_2l.img.gz>.

```
[latgrat,longrat,map] = avhrrlambert('north america',...
                                     'nausgs1_2l.img',25);

axesm('eqdazim','Origin',[50 -100],'FLatLimit',[-Inf 50],...
      'mlinelocation',15); framem; gridm
surfm(latgrat,longrat,map)
load usgslulegend; colormap(cmap)
set(handlem('surface'),'CDataMapping','Direct'); tightmap
caxis([.5 24.5]); hcb = colorbar;
set(hcb,'YTick',1:24,'YTickLabel',USGSLandUse,...
    'Position',[.7 .1 .02 .8])
set(gca,'Position',[-.05 .1 .8 .8])
```

Here is the displayed map:

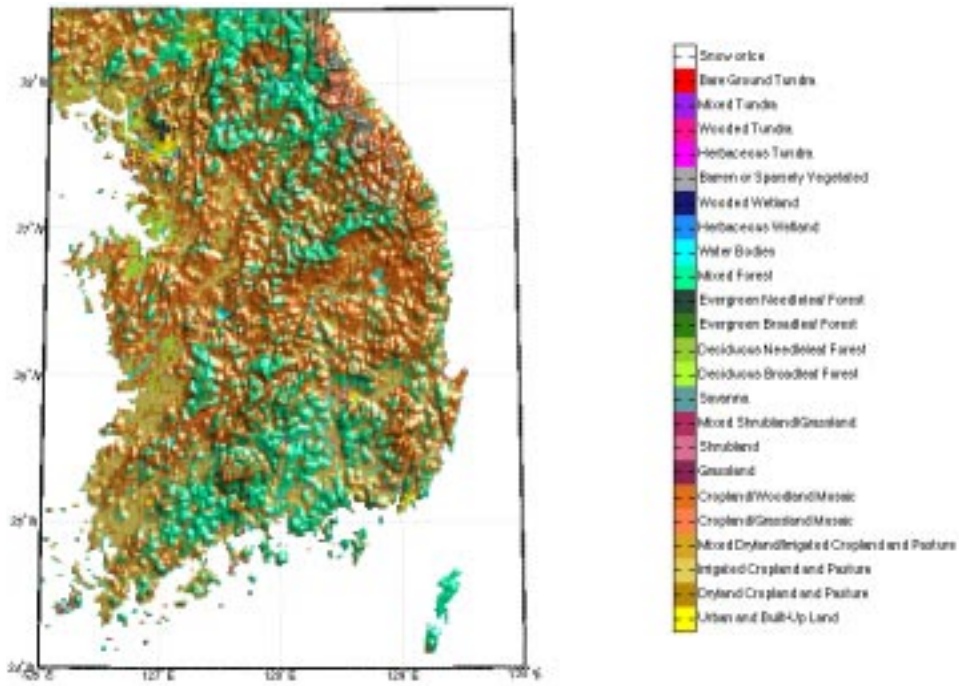


It is sometimes useful to drape land cover data onto elevation maps. The elevation data is displayed as a three-dimensional surface, onto which the new land cover map is texture mapped. The lighting effects allow you to see the land use change from intensive agriculture in low-lying flatlands to mixed cropland and woodland, and finally to pure woodland in more mountainous areas. Here are two examples combining the Global Land Cover Classification data with the GTOPO30 elevations. The first converts the general matrix map of land cover codes to a regular matrix map and combines it with a regular matrix map of elevations.

```
latlim = [34 38.5]; lonlim = [126 130];

% <ftp://edcftp.cr.usgs.gov/pub/data/glcc/ea/lamberta/
%        eausgs1_2la.img.gz>
[latgrat,longrat,mat] = avhrrlambert('asia',...
     'eausgs1_2la.img',1,latlim,lonlim);

[lcmap,lcmaplegend] = nanm(latlim,lonlim,.7/km2deg(1));
lcmap = imbedm(latgrat,longrat,mat,lcmap,lcmaplegend);
```

```
% <ftp://edcftp.cr.usgs.gov/pub/data/gtopo30/global/
%        e100n40.tar.gz>
[map,maplegend] = gtopo30('e100n40',2,latlim,lonlim);

worldmap(latlim,lonlim,'none')

h = meshm(map,maplegend,size(map),map)
set(h,'FaceColor','texturemap','CData',lcmap)
colormap(cmap)  % predefined colormap
set(h,'CDataMapping','direct')  % integer land cover codes

set(gca,'Position',[.1 .1 .5 .8])
daspectm('m',10); camlight; lighting phong;

caxis([.5 24.5]); hcb = colorbar;
set(hcb,'YTick',1:24,'YTickLabel',USGSLandUse,...
     'Position',[.6 .1 .02 .8])
```

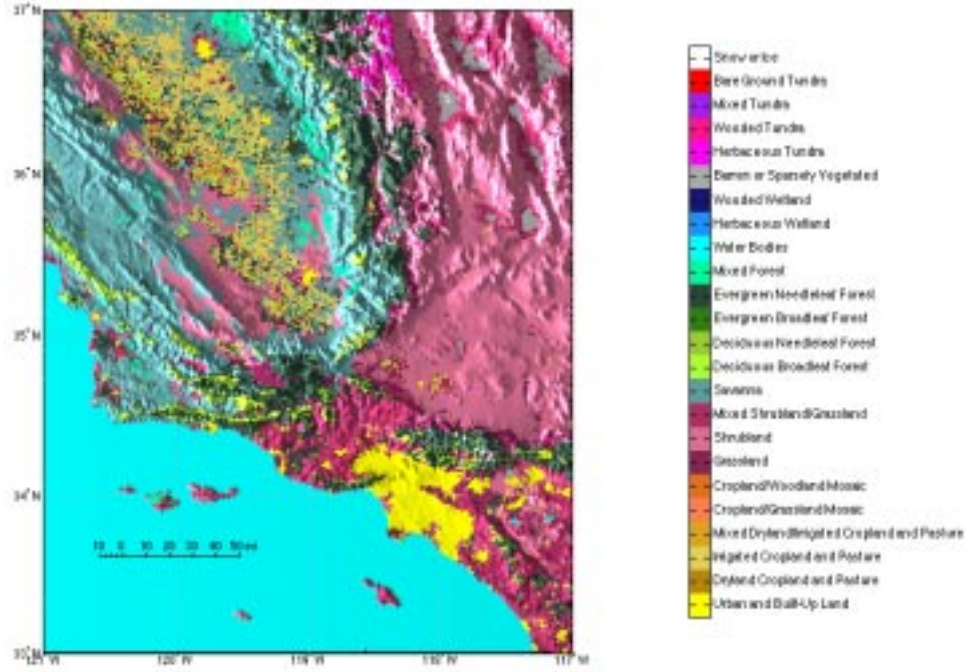
Here is the result:



The other approach is to interpolate a regular matrix map of elevations into a the general matrix. Here is an example using the land cover data and elevations for Southern California. You can see how forests are predominantly at the higher elevations around Los Angeles and the San Joaquin Valley. Notice also the string of urban areas along the treeline of the Sierra Nevada Mountains. These do not appear in other map sources, and may be spurious.

```
latlim = [33 37]; lonlim = [-121 -117]; % socal

[latgrat,longrat,lcmap] = avhrrlambert('north america',...
     'nausgs1_2l.img',1,latlim,lonlim);

latgratlim = [min(latgrat(:)) max(latgrat(:))];
longratlim = [min(longrat(:)) max(longrat(:))];

[elmap,elmaplegend] = gtopo30('W140N40',1,...
     latgratlim,longratlim);
elmap(isnan(elmap)) = -1;

z = ltln2val(elmap,elmaplegend,latgrat,longrat);

figure; worldmap(latlim,lonlim,'none');
setm(gca,'MapProjection','mercator')
hs = surfm(latgrat,longrat,lcmap,z);
set(hs,'CDataMapping','direct')
load usgslulegend; colormap(cmap)

daspectm('m',5); material([0.6 2 0])
camlight(-80,0); lighting phong
caxis([.5 24.5]); hcb = colorbar;
set(hcb,'YTick',1:24,'YTickLabel',USGSLandUse,...
     'Position',[.7 .1 .02 .8])
set(gca,'Position',[-0.04.05 .8 .9])
scaleruler('Units','mi'); tightmap
```

Here is the displayed map:



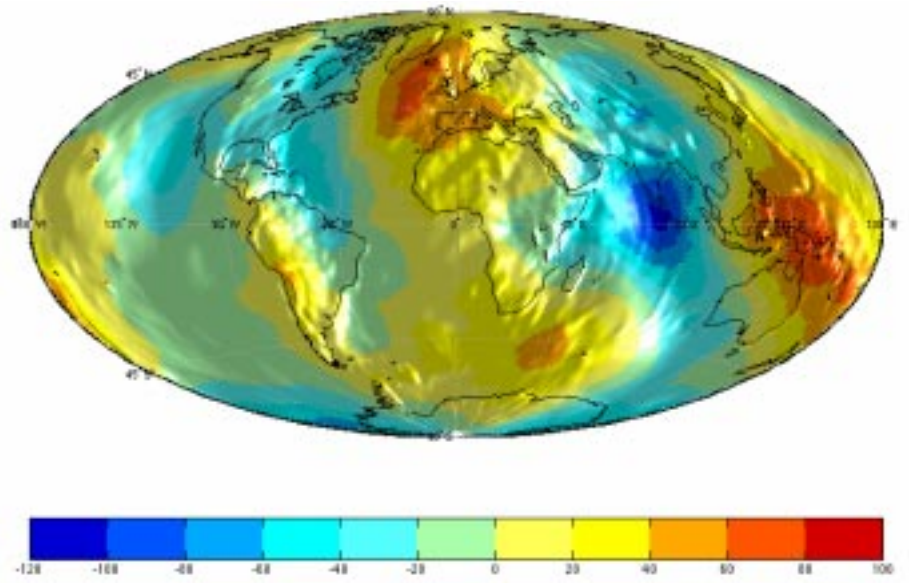
EGM96 Geoid Model

Much of the available matrix map data is elevations of the Earth's crust. For some applications it is important to know precisely the elevation and slope of sea level, or in the case of areas covered with land, where sea level would be. This surface of constant gravitational attraction is called the *geoid*, and improving man's understanding of it has been one of the primary concerns of the field of geodesy. Perhaps the best available global geoid model is EGM96, developed by NASA Goddard and NIMA. EGM 96 is a spherical harmonic model of the geoid complete to degree and order 360. The lower resolution atlas data in `geoid.mat` is derived from the EGM 96 grid.

The egm96geoid function reads the 'WW15MGH.GRD' file of geoid heights derived from the EGM 96 harmonic coefficients on a 15 minute grid. There are 721 rows and 1441 columns of values in the grid at full resolution. PC users should download the DOS self-extracting executable file in <ftp://164.214.2.59/pub/gg/gravity3/ww15mgh.exe>. Other users should download and extract the UNIX compressed file at <ftp://164.214.2.59/pub/gg/gravity3/WW15MGH.GRD.Z>.

Read the EGM 96 geoid grid for the world, taking every fifth point. Display it as a lighted surface with colormap contours in a Hammer projection.

```
[map,maplegend] = egm96geoid(5);

worldmap('world','none');
setm(gca,'MapProjection','hammer'); tightmap
meshm(map,maplegend,size(map),map); plotm(coast,'k')

light; lighting phong; material(0.6*[ 1 1 1])
set(gca,'DataAspectRatio',[1 1 200]); gridm reset
zdatam(handlem('allline'),max(map(:)))
zdatam(handlem('alltext'),max(map(:)))

caxis([-120 100]); colormap(jet(11)); colorbar('horiz')
```



The high resolution grid is most useful for computing the geoid height at particular locations. If you plan to use a higher order interpolation method, be sure to get a matrix large enough to have several points about your locations.

Read a subset of the geoid grid at full resolution and interpolate to find the geoid height at a point between grid points.

```
[map,maplegend] = egm96geoid(1,[-10 -12],[129 132]);
z = ltln2val(map,maplegend,-11.1,130.22,'bicubic')
z =
       52.444
```

U. S. Gridded Elevation Data

U. S. Geological Survey 1:250,000 (100 meter) DEMs

The U. S. Geological Survey has released a series of Digital Elevation Maps in 1-degree quadrangles covering the contiguous United States, Hawaii, and limited portions of Alaska. The 1-degree DEMs are also referred to as *3-arc second* or *1:250,000 scale* DEM data. The horizontal resolution of this data is about 100 meters.

The data is derived from the U.S. Defense Mapping Agency's DTED-1 digital elevation model, which itself was derived from cartographic and photographic sources. The cartographic sources are maps from the 7.5 minute through 1-degree series (1:24,000 scale through 1:250,000 scale). There is no bathymetric data in this DEM.

Most of the 1-degree digital elevation models have grid spacings of 3 arc-seconds in both the latitude and longitude directions. Alaska has grid spacings of 6 arc-seconds for latitudes between 50 and 70 degrees North and 9 arc-seconds for latitudes greater than 70 degrees North.

The grid for the digital elevation maps is based on the World Geodetic System 1984 (WGS 84). Older DEMs were based on WGS 72. Elevations are in meters relative to the National Geodetic Vertical Datum of 1929 (NGVD 29) in the continental U.S. and local mean sea level in Hawaii.

The specified absolute horizontal accuracy of the DEM is 130 m, while the specified absolute vertical accuracy is ±30m. The relative horizontal and vertical accuracies are not specified but should be much better than the absolute accuracies.

The DEM data files can be retrieved over the Internet from the EROS Data Center's Web site located at <http://edcwww.cr.usgs.gov/> or their FTP server at <ftp://edcftp.cr.usgs.gov/>. The files are available sorted by state and file name or can be selected from an index map. Further information on the dataset is also available at the sites listed.

The data files are named according to towns or features within the quadrangle region. If you know the geographic limits of the area you want to map, you can look up the names on the USGS Web site or use the Mapping Toolbox usgsdems function, which returns a list of map quadrangle filenames covered by your region. Here are the names of the files covered by the Cape Cod maps used throughout this document:

```
usgsdems([41 44], [-72 -69])
ans =
  'providence-w'
  'providence-e'
  'chatham-w'
  'boston-w'
  'boston-e'
  'portland-w'
  'portland-e'
  'bath-w'
```

The Cape Cod region extends over several of the 1-degree quadrangles. Displaying the whole region at the full, 3 arc-second resolution would require enormous amounts of memory and time.

Display the region covered by the 'providence-e' quadrangle using the usgsdem interface function and a downsampling factor of 5. The resulting map is a 241-by-241 element matrix. At full resolution, the map would be 1201-by-1201.

```
[map,maplegend] = usgsdem('providence-e',5);
map(map==0) = -1;

axesm mercator
meshm(map,maplegend,size(map),map); demcmap(map)
lightm(41.5,-70.5); material([.7 .7 1.5]); lighting gouraud
```

This is a zoomed-in view of the immediate region around Cape Cod Bay and Martha's Vineyard:

Zoom in further on Martha's Vineyards by providing appropriate latitude and longitude limits for the `usgsdem` function at full resolution. You can see some terracing effects in the data, an artifact of its derivation from topographic contour maps.

```
[map,maplegend] = usgsdem('providence-e',1,[41.2946 41.4829],...
                          [-70.8429 -70.4379]);
axesm mercator
meshm(map,maplegend,size(map),map); demcmap(map)
lightm(41.4,-70.65); material([.7 .7 1.5]); lighting gouraud
```

U.S. Geological Survey 1:24,000 (30m) DEMs

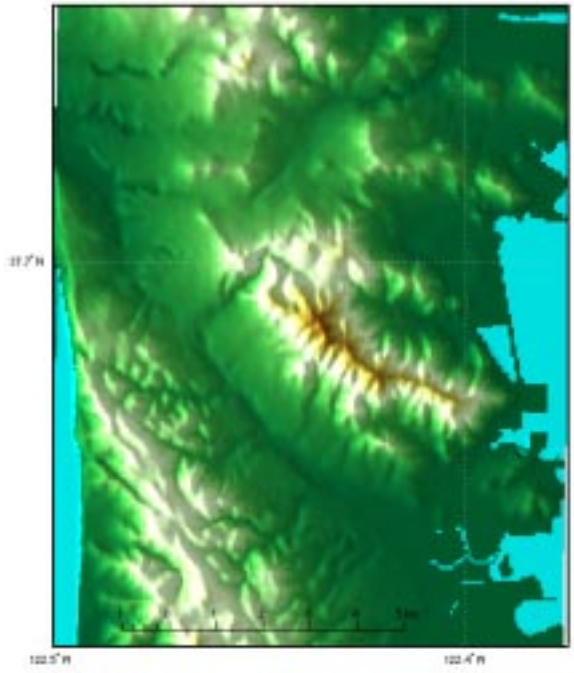
The highest resolution elevation data generally available is the USGS 1:24,000 scale DEMs. These maps have been generated from the contour lines on the paper 7.5 minute quadrangle map series. Data is available for much of the contiguous United States, Hawaii, and Puerto Rico. Because of the density of this data, little of it is available on-line. Some state agencies have collected and published files for smaller areas, such as the California Bay Area Regional Database at <ftp://bard.wr.usgs.gov/bard/dem/>. DEM files for other areas can be ordered from the USGS. These files have been released in several file formats. The USGS in the process of converting much of this data into the Spatial Data Transfer Standard. The `usgs24kdem` function reads data in the older standard format.

The data is stored as a series of profiles spaced 30 meters apart in a Universal Transverse Mercator grid. Unlike the 1:250,000 DEMs, adjacent quadrangles cannot be simply butted together to form a larger, gap-free map. There is no bathymetric data. The DEM files are ASCII files and can be transferred as text. Line ending conversion is not necessarily required.

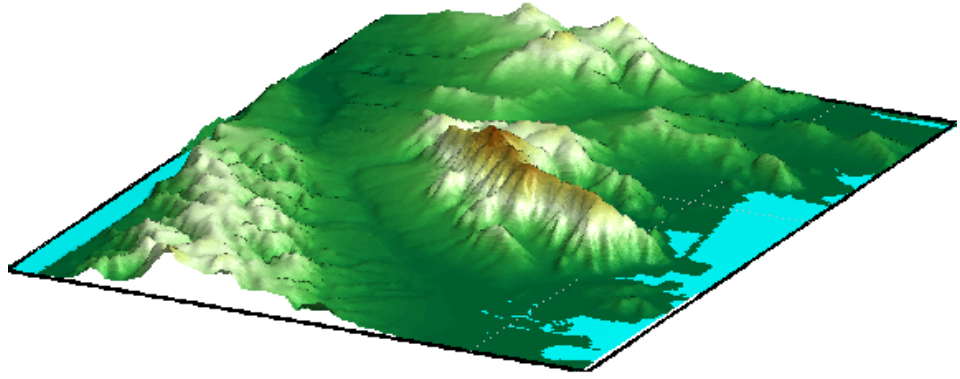
Read the 1:24,000 DEM for south San Francisco, available from the Bay Area Regional Database at <ftp://bard.wr.usgs.gov/bard/dem/dems24k/sanfrancisco/sanfranciscos.dem>. Read the entire file, taking every second point. The result is a 232-by-186 general matrix map. Because there are no negative elevations, move points at sea level down to color them blue.

```
[latgrat,longrat,map] = usgs24kdem('sanfranciscos.dem',2);
map(map==0) = -1;

latlim = [min(latgrat(:)) max(latgrat(:))];
lonlim = [min(longrat(:)) max(longrat(:))];

usamap(latlim,lonlim,'none')
surfm(latgrat,longrat,map,map-max(map(:)))
demcmap(map); daspectm('m',1); scaleruler
camlight(-80,0); lighting phong; material([.7 1 0])
```

Here is a top view of the area:



Here is a 3-D oblique view of the same region with different light and material properties:



Astronomical Data

The Mapping Toolbox provides an interface to a set of astronomical data called FK5, the Fifth Fundamental Catalog of Stars. This catalog consists of positions, proper motions, and other characteristics for over 4500 stars. The data is provided by the Astronomical Data Center (ADC), located at the NASA Goddard Space Flight Center, and can be retrieved over the Internet at the ADC's home page at <http://adc.gsfc.nasa.gov/> or by anonymous FTP at <ftp://adc.gsfc.nasa.gov/>. Documentation and other information can be found at these sites as well.

The data comes in two files, 'FK5.dat' and 'fk5_ext.dat.' You can read both files using the `readfk5` interface function. There are more than four thousand stars in the catalog, so reading and processing the data takes a while.

```
fk5  = readfk5('FK5.dat');
fk5e = readfk5('fk5_ext.dat');
whos
   Name            Size     Bytes    Class

   fk5           1x1535   5042752    struct array
   fk5e          1x3117  10226424    struct array
```

The star catalog is processed into MATLAB structures by the external interface function. The fields of the structures contain the different types of information for each star, while the structure elements represent the individual stars.

To view the fields of the structure, type the following:

```
fk5(1)
ans =
         FK5: 1
         RAh: 0
         RAm: 8
         RAs: 23.2650
        pmRA: 1.0390
         DEd: 29
         DEm: 5
         DEs: 25.5800
        pmDE: -16.3300
     RAh1950: 0
     RAm1950: 5
     RAs1950: 47.8770
    pmRA1950: 1.0360
     DEd1950: 28
     DEm1950: 48
     DEs1950: 51.9600
    pmDE1950: -16.3300
    EpRA1900: 43.3100
       e_RAs: 0.7000
      e_pmRA: 2
    EpDE1900: 33
       e_DEs: 1.3000
      e_pmDE: 3.1000
        Vmag: 2.0600
      n_Vmag: ''
      SpType: 'A0p'
         plx: 0.0240
          RV: -11.7000
       AGK3R: ''
         SRS: ''
          HD: '358'
          DM: 'BD+28    4'
          GC: '127'
```

If you just want to display the locations of the stars, you can find position and visual magnitude data in the `stars` workspace as part of the Mapping Toolbox atlas data. The remainder of this section illustrates the use of Mapping Toolbox functions to manipulate and transform star data for display.

The star positions are given in terms of right ascension and declination. You can display a star map by converting the positions to latitude and longitude coordinates. Right ascension is given in hours, minutes, and seconds, while declination is in degrees, minutes, and seconds.

```
hRA = [fk5.RAh fk5e.RAh]'; dDE = [fk5.DEd fk5e.DEd]';
mRA = [fk5.RAm fk5e.RAm]'; mDE = [fk5.DEm fk5e.DEm]';
sRA = [fk5.RAs fk5e.RAs]'; sDE = [fk5.DEs fk5e.DEs]';
lat = dms2deg(dDE,mDE,sDE);
lon = hms2hr(hRA,mRA,sRA)*360/24;
```

You probably want the map to look like you are inside the celestial sphere looking out, so switch the sign of the longitudes:

```
lon = zero22pi(-lon);
```

It is customary to display the stars with sizes proportional to the visual magnitudes. Extract the magnitudes from the data structures:

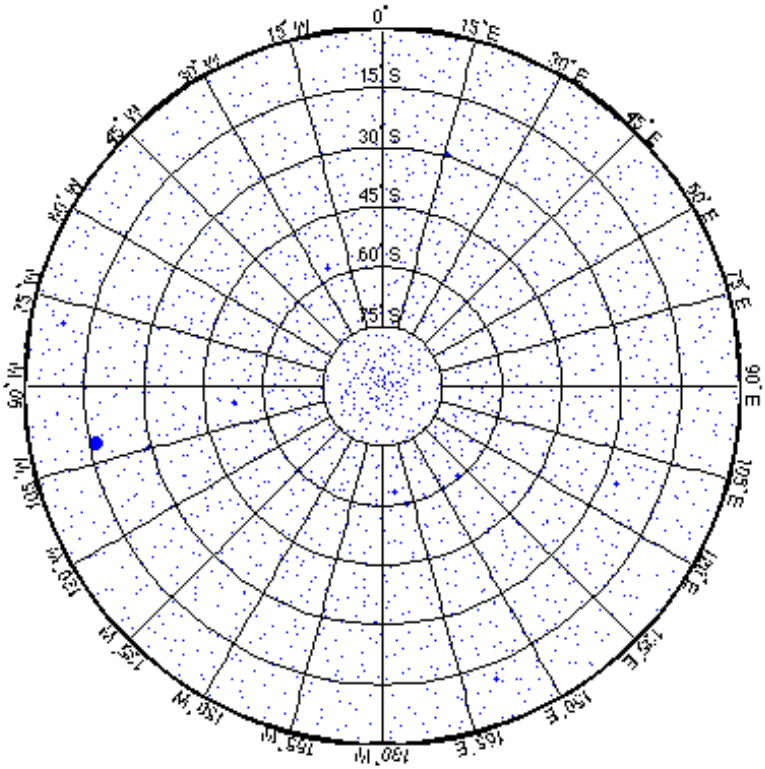
```
magn = [fk5.Vmag fk5e.Vmag]';
```

Faint stars have large visual magnitude numbers, while bright ones have smaller values. The magnitudes of particular bright stars were used to define the visual magnitude scale. It was later determined that certain stars were even brighter than the reference stars, resulting in negative visual magnitudes. You will need to convert the values to relative intensities for plotting.

```
relintensity = (magn+2).^(-100^(1/5));
clear fk5 fk5e
```

Stars in equatorial coordinates are typically displayed in the Equidistant Azimuthal projection. Here is the southern sky. You can see the southern cross at about 60° S, 175° E.

```
axesm eqdazim
framem; gridm; mlabel; plabel
setm(gca,'Origin',[-90 180 0],'FLatLimit',[-inf 90],...
  'MLineLocation',15,'MLabelLocation',-180:15:165,...
  'MLabelParallel','equator','MLineLimit',[-75 75],...
  'PLabelLocation',-15:-15:-75,'PlabelMeridian',...
  'LabelRotation','on','GLineWidth',.01,'GLineStyle','-')
set(handlem('alltext'),'HorizontalAlignment','center',...
  'VerticalAlignment','bottom')
scatterm(lat,long,sqrt(relintensity)*20,'filled')
```



The sky is often depicted in galactic coordinates, with the center of our galaxy as the origin of a polar coordinate system. You can use the Mapping Toolbox to transform the star positions from equatorial coordinates to galactic coordinates. First define the location of the galactic pole and the galactic center.

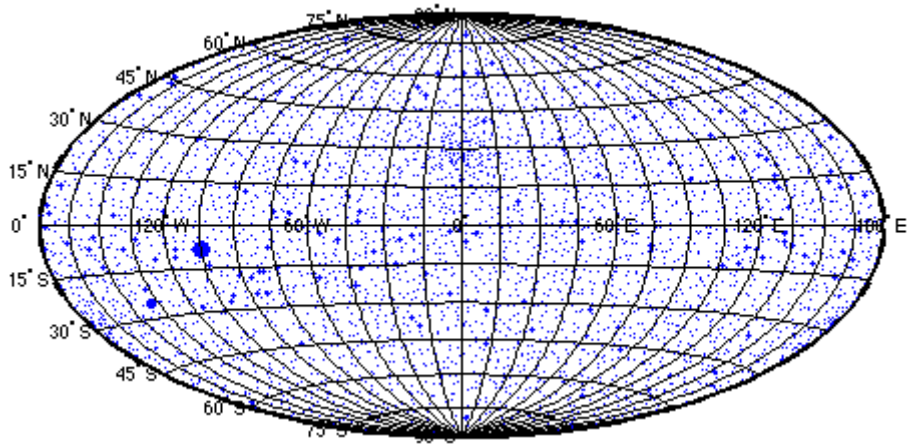
```
% galactic pole
poleRAhms = mat2hms([12 49 0]);
poleDEhms = mat2hms([27 24 0]);
polelat = dms2deg(poleDEhms);
polelon = zero22pi(-hms2hr(poleRAhms)*360/24);

% galactic center
cntrRAhms = mat2hms([17 42 0.4]);
cntrDEhms = mat2hms([-28 55 0]);
cntrlat = dms2deg(cntrDEhms);
cntrlon = zero22pi(-hms2hr(cntrRAhms)*360/24);
```

Next create a new origin vector based on the location of the new pole in the old equatorial coordinate system. Use it to rotate the locations of the pole and the center of the galactic coordinates. Note that the origin provided by the `newpole` function results in a system centered on the location of the new pole in the old coordinate system. Rotate the star positions and bring the galactic center to zero longitude.

```
% new origin vector
origin = newpole(polelat,polelon);

% rotate our known points
[newpolelat,newpolelon] = rotatem(polelat,polelon,...
                                  origin,'forward','degrees');
[newcntrlat,newcntrlon] = rotatem(cntrlat,cntrlon,...
                                  origin,'forward','degrees');

% rotate the stars
[glat,glon] = rotatem(lat,lon,origin,'forward','degrees');
glon = glon - newcntrlon;
```

The stars in galactic coordinates are commonly displayed in a pseudo-cylindrical projection like the Hammer.

```
figure
axesm hammer
framem; gridm; mlabel; plabel
setm(gca,'MLabelLocation',-120:60:180,...
  'MLineLocation',15,'MLabelParallel','equator',...
  'GLineWidth',.01,'GLineStyle','-')
scatterm(glat,glon,sqrt(relintensity)*30,'filled')
```



Importing Other Data

With the huge quantities of geographic information available, it is impossible to provide interface functions for every file format. To help you read data in other formats, the Mapping Toolbox provides a set of high–level file import functions. These allow you to read data from binary or text files by simply describing the contents of the file. The files may contain matrices or fixed length records of geographic data. MATLAB also provides some file import functions. Type `help iofun` for more information on the MATLAB functions.

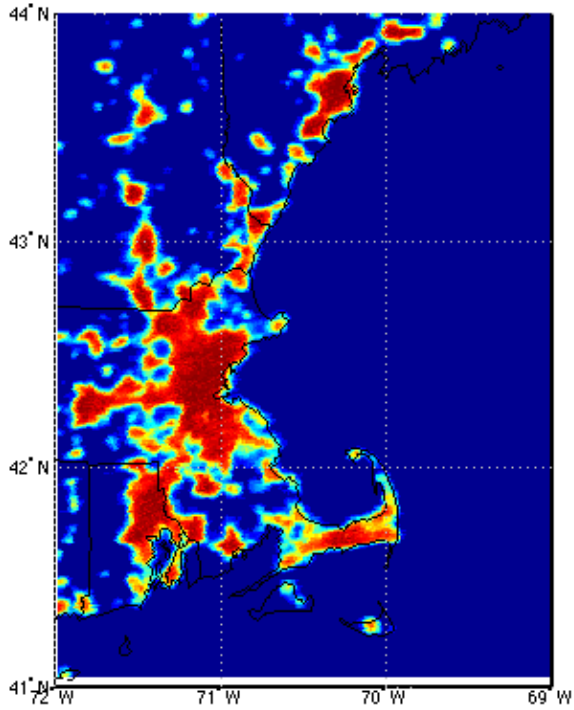
Much geographic data is provided as very large files containing matrices of values. You can read the part of the matrix of interest to you using the `readmtx` function. To illustrate the use of `readmtx`, read in data giving the intensity of stable light sources collected from the Defense Meteorological Satellite program. The data files are available from <ftp://ftp.ngdc.noaa.gov/DMSP/lights/>. We will read the 24 megabyte '`usa_pct.latlon.dat`' file for the Cape Cod region. The README file describes the format and contents of the files. From this you need the following minimum information: the number of rows and columns in the matrix, the location of the upper left corner, the grid spacing, and the storage format of the numbers.

Most matrix files can be read using the same procedure. First you map your geographic limits into matrix row and column limits and construct vectors of row and column indices to be read. Then you read the submatrix using `readmtx`, and construct the `maplegend`.

Here is an example:

```
% we want to read
latlim = [41 44]; lonlim = [-72 -69]; scalefactor = 1;

% file format and contents
nrows = 3240; ncols = 7800;
lat1 = 51; lon1 = -129;
yperrow = -dms2deg(mat2dms(0,0,30));
xpercol =  dms2deg(mat2dms(0,0,30));

% map latlim and lonlim to row and column limits
[rlim,clim] = yx2rc(latlim,lonlim,lat1,lon1,yperrow,xpercol)
rlim = sort(rlim);

readrows = rlim(1):scalefactor:rlim(2);
readcols = clim(1):scalefactor:clim(2);

% extract the map matrix
map = flipud(readmtx('usa_lights.latlon.dat',...
             nrows,ncols,'int8',readrows,readcols));

% construct maplegend
cellsize = scalefactor*xpercol;
maplegend = [1/cellsize, latlim(2)-cellsize/2, ...
             lonlim(1)-cellsize/2];

% display result
usamap(latlim,lonlim,'lineonly')
setm(gca,'mapproj','mercator'); tightmap
meshm(map,maplegend)
```

Here is the displayed map:



Boston appears to have grown since the data in the VMAP0 was compiled. The VMAP0 also missed the heavily populated part of Cape Cod.

Some data is stored on regular grids in projected coordinates. In that case, you need to define a map projection that matches the one used for the data. Because the grid is in projected coordinates rather than latitude and longitude, the map is a general matrix map and needs a graticule. To construct one, build a matrix of row and column indices, convert them to projected coordinates, and carry out the inverse projection. The resulting latitude, longitude, and map matrices can be displayed in any projection. Edit the `avhrrlambert` interface function to see an example.

Another common type of geographic data file format is fixed length records with fields of information. Examples of such data include geographic names, hydrographic soundings, ship track lines, and astronomical catalogs of stars. Use the `readfields` function to read selected fields or records from such files.

As an example, retrieve the U. S. Geological Survey Geographic Names Information System (GNIS) file containing a concise list of names and locations within the United States. The file can be found at <http://mapping.usgs.gov/pub/gnis/US_concise.gz>.

The README file describes the file format and contents. To read this data, create a structure with this information.

```
filestruct(1).length = 51;
filestruct(1).name = 'Feature Name';
filestruct(1).type = 'char';

filestruct(2).length = 10;
filestruct(2).name = 'Feature Type';
filestruct(2).type = 'char';

filestruct(3).length = 32;
filestruct(3).name = 'County';
filestruct(3).type = 'char';

filestruct(4).length = 17;
filestruct(4).name = 'State';
filestruct(4).type = 'char';

filestruct(5).length = 16;
filestruct(5).name = 'Geographic Coordinates';
filestruct(5).type = 'char';

filestruct(6).length = 6;
filestruct(6).name = 'Feature Elevation';
filestruct(6).type = 'char';

filestruct(7).length = 1;
filestruct(7).name = ''; %line ending
filestruct(7).type = 'char';
```

You can identify records that match a search string using the `grepfields` function. With no output arguments, records matching the string are displayed onscreen. When an output argument is provided, the record numbers of matching records are returned.

Search for the locations of The MathWorks and Systems Planning and Analysis.

```
grepfields('us_concise','natick')
Natick                          ppl    Middlesex
Massachusetts    421700N0712100W 180

indx = grepfields('us_concise','alexandria');
```

Use the `readfields` function to read the selected records into a structure.

```
s = readfields('us_concise',filestruct,indx)
s =
1x12 struct array with fields:
    FeatureName
    FeatureType
    County
    State
    GeographicCoordinates
    FeatureElevation

s(7)
ans =
              FeatureName: 'Alexandria'
              FeatureType: 'ppl'
                   County: 'Independent City'
                    State: 'Virginia'
    GeographicCoordinates: '384817N0770250W'
         FeatureElevation: '32'
```

If you are having trouble getting the file format right, you can check the record length of a file with line ending characters. Remember that DOS-based computers use two line ending characters.

```
fid = fopen('us_concise','r');
str = fgets(fid)
str =
Aasu                              ppl     Western
American Samoa   141751S1704530W

recordlength = length(str)
recordlength =
        133.00

fclose(fid);
```

If the file does not have line endings, you can use `fread` to read a few characters at a time.

A

Geographic Terms

Glossary Notes

This glossary of geographical terms is drawn extensively from *An Album of Map Projection, U.S. Geological Survey Professional Paper 1453*, by John P. Snyder and Philip M. Voxland.

The purpose of this glossary is to assist the user in understanding the Mapping Toolbox. For this reason, some terms in this glossary are specifically defined as they are used in this guide rather than as they might be more generally used in the geographic community.

Glossary

Antipodes - Two points on opposite sides of a planet.

Arc-second - 1/3600th of a degree (1 second) of latitude or longitude.

ARC/Info - A largely UNIX-oriented GIS developed and distributed by the Environmental Systems Research Institute, Inc.

Aspect - The conceptual placement of a projection system in relation to the Earth's axis (direct, normal, polar, equatorial, oblique, and so on).

Authalic projection - *See* Equal-area projection.

Axes - *See* Map axes.

Azimuth - The angle a line makes with a meridian, taken clockwise from north.

Azimuthal projection - A projection on which the azimuth or direction from a given central point to any other point is shown correctly. When a pole is the central point, all meridians are spaced at their true angles and are straight radii of concentric circles that represent the parallels. Also called a zenithal projection.

Bathymetry - The measurement of water depths of oceans, seas, lakes, and other bodies of water.

Bowditch, Nathaniel - A late 18th/early 19th century mathematician, astronomer, and sailor who "wrote the book" on navigation. John Hamilton Moore's *The Practical Navigator* was the leading navigational text when Bowditch first went out to sea, and had been for many years. Early in his first voyage, however, Bowditch began noticing errors in Moore's book, which he recorded and later used in preparing an American edition of Moore's work. The revisions were to such an extent that Bowditch was named the principal author, and the title was changed to *The New American Practical Navigator*, published in 1802. In 1868 the U.S. Navy bought the copyright to the book, which is still commonly referred to as "Bowditch" and considered the "bible" of navigation.

Cartography - The art or practice of making charts or maps.

Central meridian - The meridian passing through the center off a projection, often a straight line about which the projection is symmetrical.

Central projection - A projection in which the Earth is projected geometrically from the center of the Earth onto a plane or other surface. The Gnomonic and Central Cylindrical projections are examples.

Choropleth - A map consisting of areas of equal value separated by abrupt boundaries and colored or shaded according to those values.

Complex curves - Curves that are not elementary forms such as circles, ellipses, hyperbolas, parabolas, and sine curves.

Composite projection - A projection formed by connecting two or more projections along common lines such as parallels of latitude, necessary adjustments being made to achieve fit. The Goode Homolosine projection is an example.

Conformal projection - A projection on which all angles at each point are preserved. Also called an orthomorphic projection.

Conceptually projected - The convenient way to visualize a projection system, although it may not correspond to the actual mathematical projection method.

Conic projection - A projection resulting from the conceptual projection of the Earth onto a tangent or secant cone, which is then cut lengthwise and laid flat. When the axis of the cone coincides with the polar axis of the Earth, all meridians are straight equidistant radii of concentric circular arcs representing the parallels, but the meridians are spaced at less than their true angles. Mathematically, the projection is often only partially geometric.

Constant scale - A linear scale that remains the same along a particular line on a map, although that scale may not be the same as the stated or nominal scale of the map.

Contour - All points that are at the same elevation above or below a specified datum.

Conventional aspect - *See* Normal aspect.

Correct scale - A linear scale having exactly the same value as the stated or nominal scale of the map, or a scale factor of 1.0. Also called true scale.

Cylindrical projection - A projection resulting from the conceptual projection of the Earth onto a tangent or secant cylinder, which is then cut lengthwise and laid flat. When the axis of the cylinder coincides with the axis of the Earth, the meridians are straight, parallel, and equidistant, while the parallels of latitude are straight, parallel, and perpendicular to the meridians. Mathematically, the projection is often only partially geometric.

Dead reckoning - From “deduced reckoning,” the estimation of geographic position based on course, speed and time.

DEM (Digital Elevation Map/Model) - Elevation data in the form of a matrix map, generally on a regular grid. DEM also refers to the five primary types of digital elevation models produced by the U.S. Geological Survey, one of which is the 1-degree (3-arc-second resolution) model that is interfaced through the Mapping Toolbox.

Departure - The arc length distance along a parallel of a point from a given meridian.

Developable surface - A simple geometric form capable of being flattened without stretching. Many map projections can be grouped by a particular developable surface: cylinder, cone, or plane.

Direct aspect - *See* Normal aspect.

Distortion - A variation of the area or linear scale on a map from that indicated by the stated map scale, or the variation of a shape or angle on a map from the corresponding shape or angle on the Earth.

DMS - Degrees-minutes-seconds angle notation of the form ddd° mm' ss". There are 60 seconds in a minute, and 60 minutes in a degree. In the Mapping Toolbox, when “dms” angles are represented by a single number, the format is dddmm.ss.

Ellipsoid - When used to represent the Earth, a solid geometric figure formed by rotating an ellipse about its minor (shorter) axis. Also called spheroid.

Equal-area projection - A projection on which the areas of all regions are shown in the same proportion to their true areas. Shapes may be greatly distorted. Also called an equivalent or authalic projection.

Equatorial aspect - An aspect of an azimuthal projection on which the center of projection or origin is some point along the Equator. For cylindrical and pseudocylindrical projections, this aspect is usually called conventional, direct, normal, or regular rather than equatorial.

Equidistant projection - A projection that maintains constant scale along all great circles from one or two points. When the projection is centered on a pole, the parallels are spaced in proportion to their true distances along each meridian.

Equireal projection - *See* Equal-area projection.

Equivalent projection - *See* Equal-area projection.

Flat-polar projection - A projection on which, in normal aspect, the pole is shown as a line rather than as a point.

Frame - *See* Map frame.

Free of distortion - Having no distortion of shape, area, or linear scale. On a flat map, this condition can exist only at certain points or along certain lines.

General matrix map - In the Mapping Toolbox, a matrix map defined with latitude-longitude coordinate matrices, allowing irregular, non-rectangular orientations.

Geoid - The true shape of the Earth, an irregular and complex shape which is usually modeled with a sphere or an ellipsoid. In the Mapping Toolbox, this term refers to the spherical or ellipsoidal model of the Earth *or other planet* in use rather than the true shape.

Geoid vector - In the Mapping Toolbox, a vector describing the geoid, or ellipsoid, model. The geoid vector has the form:

```
geoidvec = [semimajor-axis eccentricity]
```

Geometric projection - *See* Perspective projection.

Geographic data structure - In the Mapping Toolbox, A MATLAB data structure containing the data and other information for the proper display of map objects. Valid fields in the structure include `type`, `tag`, and `altitude`.

GIS (Geographic Information System) - A system, usually computer based, for the input, storage, retrieval, analysis, and display of interpreted geographic data.

Globular projection - Generally, a nonazimuthal projection developed before 1700 on which a hemisphere is enclosed in a circle and meridians and parallels are simple curves or straight lines.

Graticule - A network of lines representing a selection of the Earth's parallels and meridians for the purpose of projection. The vertices of the graticule grid are precisely projected, and the map data contained in any grid cell is warped to fit the resulting quadralateral. A finer graticule grid results in a higher projection fidelity at the expense of greater computational requirements.

Great circle - Any circle on the surface of a sphere, especially when the sphere represents the Earth, formed by the intersection of the surface with a plane passing through the center of the sphere. It is the shortest path between any two points along the circle and therefore important for navigation. All meridians and the Equator are great circles on the Earth taken as a sphere.

Grid - *See* Map grid.

HMS - Hours-minutes-seconds time notation of the form hh° mm' ss". In the Mapping Toolbox, when “hms” times are represented by a single number, the format is hhmm.ss.

Homalographic/homolographic projection - *See* Equal-area projection.

Hydrography - The science of measurement, description, and mapping of the surface waters of the Earth, especially with reference to their use in navigation. The term also refers to those parts of a map collectively that represent surface waters.

Hydrology - The scientific study of the waters of the Earth, especially with relation to the effects of precipitation and evaporation upon the occurrence and character of ground water.

Hypsography - The scientific study of the Earth's topologic configuration above sea level, especially the measurement and mapping of land elevation.

Indexed map - A matrix map in which entries are an index value into another data source. The `worldmtx` workspace contains an example of an indexed map. Each entry in the matrix map is an index into a data structure containing the names of the world countries.

Indicatrix - A circle or ellipse having the same shape as that of an infinitesimally small circle (having differential dimensions) on the Earth when it is plotted with finite dimensions on a map projection. Its axes lie in the directions of and are proportional to the maximum and minimum scales at that point. This is useful in illustrating the distortions of a given map projection. Often called a Tissot indicatrix after the originator of the concept. In the Mapping Toolbox, Tissot indicatrices may be displayed using the `tissot` command, and indicatrices for all supported projections are provided in the "Projections Reference" section of the *Mapping Toolbox Reference Guide*.

Interrupted projection - A projection designed to reduce peripheral distortion by making use of separate sections joined at certain points or along certain lines, usually the Equator in the normal aspect, and split along lines that are usually meridians. There is normally a central meridian for each section. The Mapping Toolbox does not include interrupted projections, but the user can separate data into sections and project these independently to achieve this effect.

Large-scale mapping - Mapping at a scale larger than about 1:75,000, although this limit is somewhat flexible.

Latitude (geographic) - The angle made by a perpendicular to a given point on the surface of a sphere or ellipsoid representing the Earth and the plane of the Equator (positive if the point is north of Equator, negative if it is south). One of the two common geographic coordinates of a point on the Earth.

Latitude of opposite sign - *See* Parallel of opposite sign.

Legs - Line segments connecting waypoints.

Legend - *See* Map legend.

Limiting forms - The form taken by a system of projection when the parameters of the formulas defining that projection are allowed to reach limits that cause it to be identical with another separately defined projection.

Logical map - A binary matrix map consisting entirely of 1s and 0s. An example of a logical matrix map can be created with the `topo` map by performing a logical test for positive elevations (`topo>0`). Each entry in the matrix map contains a 1 if it is above sea level, or a 0 if it is at or below sea level.

Longitude - The angle made by the plane of a meridian passing through a given point on the Earth's surface and the plane of the (prime) meridian passing through Greenwich, England, east or west to 180 (positive if the point is east, negative if it is west). One of the two common geographic coordinates of a point on the Earth.

Loxodrome - *See* Rhumb line.

Map A representation of geographic data. In the Mapping Toolbox, a map is any variable or set of variables (electronically) representing or assigning values to a geographic location or region, from a single point to an entire planet.

Map axes - In the Mapping Toolbox, a normal MATLAB axes altered for mapping display purposes. Several map axes properties are defined and stored in the `UserData` slot of the MATLAB axes. These properties control the appearance of the map display, much like the properties of the normal MATLAB axes control the appearance of the displayed plot. A map axes must first be defined in order to display maps using the Mapping Toolbox.

Map frame - In the Mapping Toolbox, a projected "box" or quadrangle enclosing the geographic display.

Map grid - A displayed network of lines representing parallels and meridians. The grid is used for visual reference and should not be confused with the graticule.

Map legend - In the Mapping Toolbox, a vector defining the geographic placement and unit cell size of a regular matrix map. A map legend has the form:

```
maplegend = [cells/angleunit north-latitude west-longitude]
```

MapInfo - A largely PC-oriented GIS developed and distributed by the MapInfo Corporation.

Matrix map - A map consisting of a matrix (or matrices) of values corresponding to specific geographic points. In the Mapping Toolbox, matrix maps can be defined as regular or general, depending on the structure and orientation of the geographic points. *See* Regular matrix map and General matrix map.

Meridian - A reference line on the Earth's surface formed by the intersection of the surface with a plane passing through both poles and some third point on the surface. This line is identified by its longitude. On the Earth as a sphere, this line is half a great circle; on the Earth as an ellipsoid, it is half an ellipse.

Minimum-error projection - A projection having the least possible total error of any projection in the designated classification, according to a given mathematical criterion. Usually, this criterion calls for the minimum sum of squares of deviations of linear scale from true scale throughout the map ("least squares").

NGVD 29 (National Geodetic Vertical Datum of 1929) - A reference surface established by the U.S. Coast Guard and Geodetic Survey of 1929, used as the datum for which relief features and elevation data are referenced in the conterminous United States; formerly called "mean sea level 1929."

Nominal scale - The stated scale at which a map projection is constructed.

Normal aspect - A form of a projection that provides the simplest graticule and calculations. It is the polar aspect for azimuthal projections, the aspect having a straight Equator for cylindrical and pseudocylindrical projections, and the aspect showing straight meridians for conic projections. Also called conventional, direct, or regular aspect.

Oblique aspect - An aspect of a projection on which the axis of the Earth is rotated so it is neither aligned with nor perpendicular to the conceptual axis of the map projection.

Orthoapsidal projection - A projection on which the surface of the Earth taken as a sphere is transformed onto a solid other than the sphere and then projected orthographically and obliquely onto a plane for the map.

Orthographic projection - A specific azimuthal projection or a type of projection in which the Earth is projected geometrically onto a surface by means of parallel projection lines.

Orthomorphic projection - *See* Conformal projection.

Parallel - A small circle on the surface of the Earth formed by the intersection of the surface of the reference sphere or ellipsoid with a plane parallel to the plane of the Equator. This line is identified by its latitude. The Equator (a great circle) is usually also treated as a parallel.

Parallel of opposite sign - A parallel that is equally distant from but on the opposite side of the Equator. For example, for lat 30(°N (or +30°), the parallel of opposite sign is lat 30° S (or -30°). Also called latitude of opposite sign.

Parameters - The values of constants as applied to a map projection for a specific map; examples are the values of the scale, the latitudes of the standard parallels, and the central meridian. The required parameters vary with the projection.

Perspective projection - A projection produced by projecting straight lines radiating from a selected point (or from infinity) through points on the surface of a sphere or ellipsoid and then onto a tangent or secant plane. Other perspective maps are projected onto a tangent or secant cylinder or cone by using straight lines passing through a single axis of the sphere or ellipsoid. Also called geometric projection.

Planar projection - A projection resulting from the conceptual projection of the Earth onto a tangent or secant plane. Usually, a planar projection is the same as an azimuthal projection. Mathematically, the projection is often only partially geometric.

Planimetric map - A map representing only the horizontal positions of features (without their elevations).

Polar aspect - An aspect of a projection, especially an azimuthal one, on which the Earth is viewed from the polar axis. For cylindrical or pseudocylindrical projections, this aspect is called transverse.

Pole - An extremity of a planet's axis of rotation. The North Pole is a singular point at 90°N for which longitude is ambiguous. The South Pole has the same characteristics and is located at 90°S.

Polyconic projection - A specific projection or member of a class of projections that are constructed like conic projections but with different cones for each parallel. In the normal aspect, all the parallels of latitude are nonconcentric circular arcs, except for a straight Equator, and the centers of these circles lie along a central axis.

Projection - A systematic representation of a curved 3-D surface such as the Earth onto a flat 2-D plane. Each map projection has specific properties that make it useful for specific purposes.

Pseudoconic projection - A projection that, in the normal aspect, has concentric circular arcs for parallels and on which the meridians are equally spaced along the parallels, like those on a conic projection, but on which meridians are curved.

Pseudocylindrical projection - A projection that, in the normal aspect, has straight parallel lines for parallels and on which the meridians are (usually) equally spaced along parallels, as they are on a cylindrical projection, but on which the meridians are curved.

Quadrangle - A region bounded by parallels north and south, and meridians east and west.

Raster map - *See* Matrix map.

Reckoning - The determination of geographic position by calculation.

Regional map - A small-scale map of an area covering at least 5 or 10 degrees of latitude and longitude but less than a hemisphere.

Regular aspect - *See* Normal aspect.

Regular matrix map - In the Mapping Toolbox, an equiangular (equal-angle) matrix map defined with a map legend vector, limited to a rectangular orientation.

Retroazimuthal projection - A projection on which the direction or azimuth from every point on the map to a given central point is shown correctly with respect to a vertical line parallel to the central meridian. The reverse of an azimuthal projection.

Rhumb line - A complex curve (a spherical helix) on a planet's surface that crosses every meridian at the same oblique angle; a navigator can proceed between any two points along a rhumb line by maintaining a constant heading. A rhumb line is a straight line on the Mercator projection. Also called a loxodrome.

Scale - The ratio of the distance on a map or globe to the corresponding distance on the Earth; usually stated in the form 1:5,000,000 for example.

Scale factor - The ratio of the scale at a particular location and direction on a map to the stated scale of the map. At a standard parallel, or other standard line, the scale factor is 1.0.

Secant cone, cylinder, or plane - A secant cone or cylinder intersects the sphere or ellipsoid along two separate lines; these lines are parallels of latitude if the axes of the geometric figures coincide. A secant plane intersects the sphere or ellipsoid along a line that is a parallel of latitude if the plane is at right angles to the axis.

Shaded Relief - Shading added to a map or image that makes it appear to have three-dimensional aspects. This type of enhancement is commonly done to satellite images and thematic maps utilizing digital topographic data to provide the appearance of terrain relief.

Similar (projection) - Subjective and qualitative term indicating a moderate or strong resemblance.

Singular points - Certain points on most but not all conformal projections at which conformality fails, such as the poles on the normal aspect of the Mercator projection.

Skew-oblique aspect - An aspect of a projection on which the axis of the Earth is rotated, so it is neither aligned with nor perpendicular to the conceptual axis of the map projection, and tilted, so the poles are at an angle to the conceptual axis of the map projection.

Small circle - A circle on the surface of a sphere formed by the intersection with a plane. Parallels of latitude are small circles on the Earth taken as a sphere. In the Mapping Toolbox, great circles, including the Equator and all meridians, are treated as special, limiting cases of small circles.

Small-scale mapping - Mapping at a scale smaller than about 1:1,000,000, although the limiting scale sometimes has been made as large as 1:250,000.

Spheroid - *See* Ellipsoid.

Standard parallel - In the normal aspect of a projection, a parallel of latitude along which the scale is as stated for that map. There are one or two standard parallels on most cylindrical and conic map projections and one on many polar stereographic projections.

Stereographic projection - A specific azimuthal projection or type of projection in which the Earth is projected geometrically onto a surface from a fixed (or moving) point on the opposite face of the Earth.

Tangent cone or cylinder - A cone or cylinder that just touches the sphere or ellipsoid along a single line. This line is a parallel of latitude if the axes of the geometric figures coincide.

Thematic map - A map designed to portray primarily a particular subject, such as population, railroads, or croplands.

Tissot indicatrix - *See* Indicatrix.

Topographic map - A map that usually represents the vertical positions or elevations of features as well as their horizontal positions. The `topo` workspace contains a simple example.

Transformed latitudes, longitudes, or poles - Graticule of meridians and parallels on a projection after the Earth has been turned with respect to the projection so that the Earth's axis no longer coincides with the conceptual axis of the projection. Used for oblique and transverse aspects of many projections.

Transverse aspect - An aspect of a map projection on which the axis of the Earth is rotated so that it is at right angles to the conceptual axis of the map projection. For azimuthal projections, this aspect is usually called equatorial rather than transverse.

True scale - *See* Correct scale.

Valued map - A matrix map in which entries represent some value or measurement. The `topo` workspace contains an example of a valued map. Each entry in the matrix map is an average elevation in meters for the geographic position represented by that cell.

Vector map - A map consisting of ordered latitude-longitude points, possibly connected. In the Mapping Toolbox, such map data is often represented by two vectors, representing latitude and longitude. Segments can be separated by the insertion of `NaN`'s in both vectors.

Waypoint - Points through which a track passes, usually corresponding to course or speed changes.

WGS 72 (World Geodetic System 1972 - An Earth-centered datum, used as a definition of DMA DEMs, presently stored in the USGS data base. The WGS 72 datum was the result of an extensive effort extending over approximately three years to collect selected satellite, surface gravity, and astrogeodetic data available throughout 1972. These data were combined using a unified WGS solution (a large-scale least squares adjustment).

WGS 84 (World Geodetic System 1984) - The WGS 84 was developed as a replacement for the WGS 72 by the military mapping community as a result of new and more accurate instrumentation and a more comprehensive control network of ground stations. The newly developed satellite radar altimeter was used to deduce geoid heights from oceanic regions between 70° north and south latitude. Geoid heights were also deduced from ground-based Doppler and ground-based laser satellite-tracking data, as well as surface gravity data. The ellipsoid associated with WGS 84 is GRS 80.

Zenithal projection - *See* Azimuthal projection.

B

Bibliography

1 Snyder, J. P., *Map Projections - A Working Manual*, U.S. Geological Survey Professional Paper 1395, Washington, D.C., 1987.

2 Maling, D. H., *Coordinate Systems and Map Projections*, 2nd Edition, Pergamon Press, New York, NY, 1992.

3 Snyder, J. P. and Voxland, P. M., *An Album of Map Projections*, U.S. Geological Survey Professional Paper 1453, Washington, D.C., 1994.

4 Snynder, J. P., *Flattening the Earth - 2000 Years of Map Projections*, University of Chicago Press, Chicago, IL, 1993.

Projections Reference

How to Use the Projections Reference Pages

The Projections Reference pages are organized in alphabetical order by the name of the map projection. The entries in this chapter contain the following:

Classification	Classifies the projection by the geometric or mathematical means of construction.
Syntax	Provides the name of the projection M-file used to specify a particular map projection.
Graticule	Describes the appearance of meridians, parallels, poles, and map symmetry.
Features	Describes the properties of the projection and identifies map distortion.
Parallels	Describes the standard parallels of projection.
Remarks	Describes the history of the projection and relationships to other projections.
Limitations	Describes any restrictions on using the projection.

Aitoff Projection

Classification Modified Azimuthal

Syntax `aitoff`

Graticule Meridians: Central meridian is a straight line half the length of the Equator. Other meridians are complex curves, equally spaced along the Equator, and concave towards the central meridian.

Parallels: Equator is straight. Other parallels are complex curves, equally spaced along the central meridian, and concave towards the nearest pole.

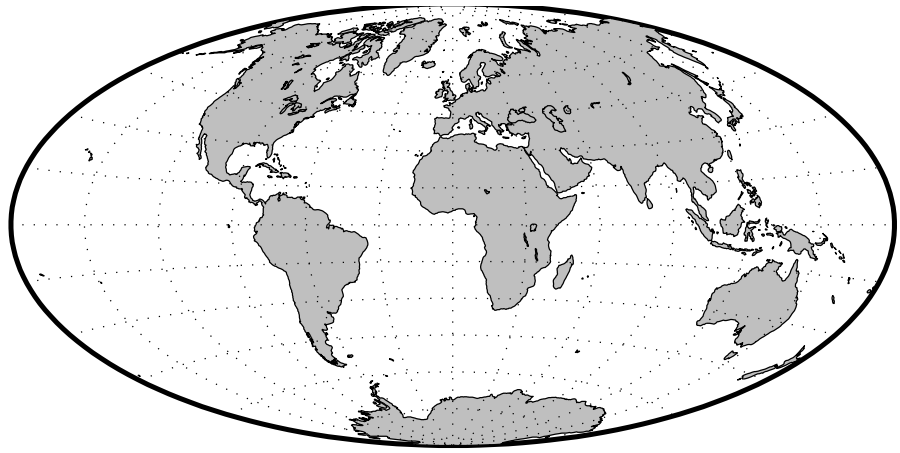
Poles: Points.

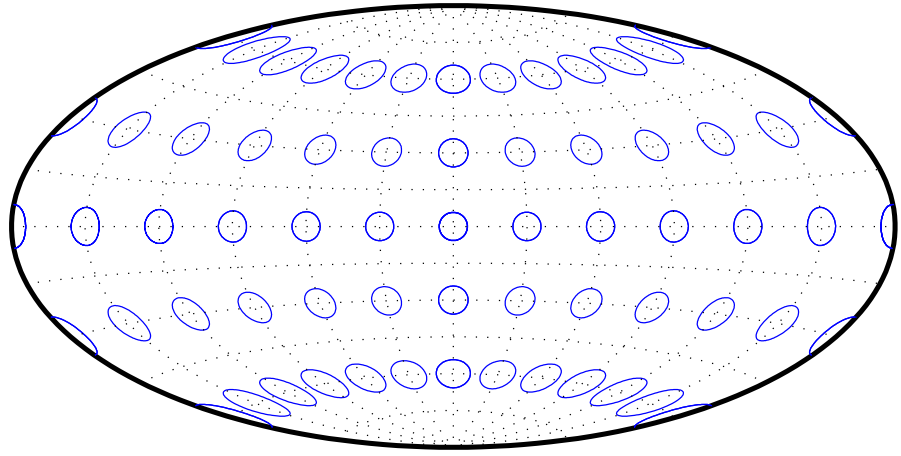
Symmetry: About the Equator and central meridian.

Features This projection is neither conformal nor equal area. The only point free of distortion is the center point. Distortion of shape and area are moderate throughout. This projection has less angular distortion on the outer meridians near the poles than pseudoazimuthal projections

Parallels There is no standard parallel for this projection.

Remarks This projection was created by David Aitoff in 1889. It is a modification of the Equidistant Azimuthal projection. The Aitoff projection inspired the similar Hammer projection, which is equal area.





Albers Equal-Area Conic Projection

Classification Conic

Syntax `eqaconic`

Graticule Meridians: Equally spaced straight lines converging to a common point, usually beyond the pole. The angles between the meridians are less than the true angles.

Parallels: Unequally spaced concentric circular arcs centered on the point of convergence. Spacing of parallels decreases away from the central latitudes.

Poles: Normally circular arcs, enclosing the same angle as the displayed parallels.

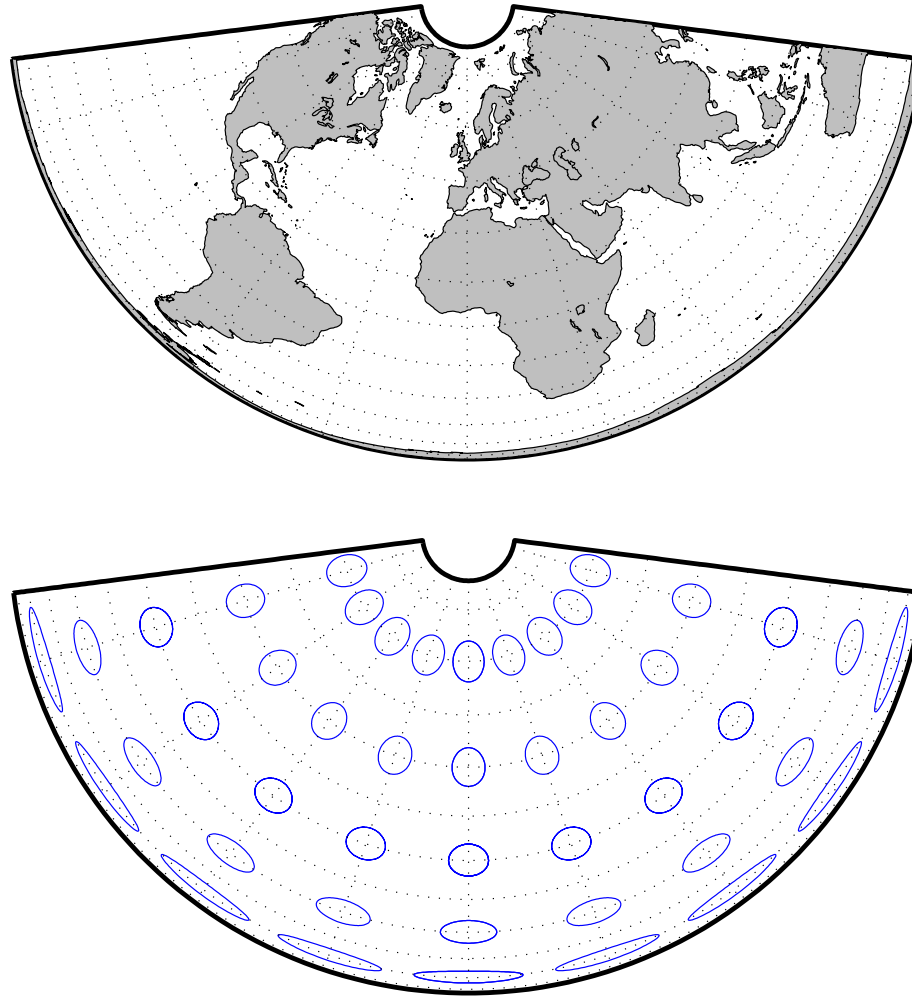
Symmetry: About any meridian.

Features This is an equal-area projection. Scale is true along the one or two selected standard parallels. Scale is constant along any parallel; the scale factor of a meridian at any given point is the reciprocal of that along the parallel to preserve equal-area. This projection is free of distortion along the standard parallels. Distortion is constant along any other parallel. This projection is neither conformal nor equidistant.

Parallels The cone of projection has interesting limiting forms. If a pole is selected as a single standard parallel, the cone is a plane and a Lambert Azimuthal Equal-Area projection results. If two parallels are chosen, not symmetric about the Equator, then a Lambert Equal-Area Conic projection results. If a pole is selected as one of the standard parallels, then the projected pole is a point, otherwise the projected pole is an arc. If the Equator is chosen as a single parallel, the cone becomes a cylinder and a Lambert Equal-Area Cylindrical projection is the result. Finally, if two parallels equidistant from the Equator are chosen as the standard parallels, a Behrmann or other equal-area cylindrical projection is the result. Suggested parallels for maps of the conterminous U.S. are [29.5 45.5]. The default parallels are [15 75].

Remarks This projection was presented by Heinrich Christian Albers in 1805.

Limitations Longitude data greater than 135° east or west of the central meridian is trimmed.



Apianus II Projection

Classification Pseudocylindrical

Syntax `apianus`

Graticule Meridians: Equally spaced elliptical curves converging at the poles.

Parallels: Equally spaced straight lines.

Poles: Points.

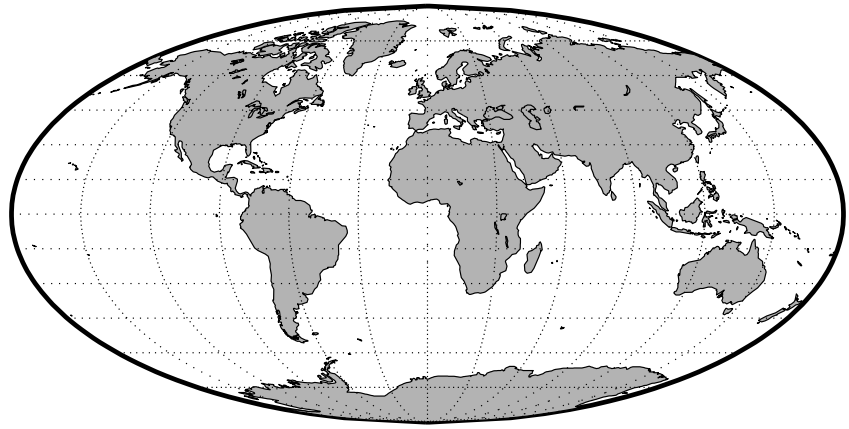
Symmetry: About the Equator and central meridian.

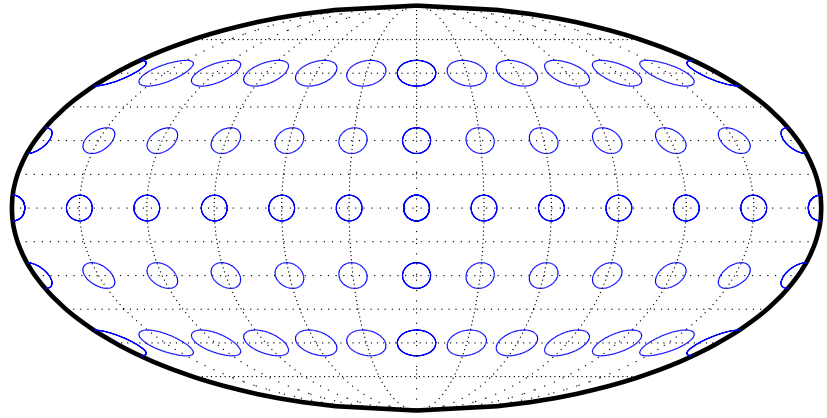
Features Scale is constant along any parallel or pair of parallels equidistant from the Equator, as well as along the central meridian. The Equator is free of angular distortion. This projection is not equal-area, equidistant, or conformal.

Parallels There is no standard parallel for this projection.

Remarks This projection was first described in 1524 by Peter Apian (or Bienewitz).

Limitations This projection is available for the spherical geoid only.





Balthasart Cylindrical Projection

Classification Cylindrical

Syntax `balthsrt`

Graticule Meridians: Equally spaced straight parallel lines.

Parallels: Unequally spaced straight parallel lines, perpendicular to the meridians. Spacing is closest near the poles.

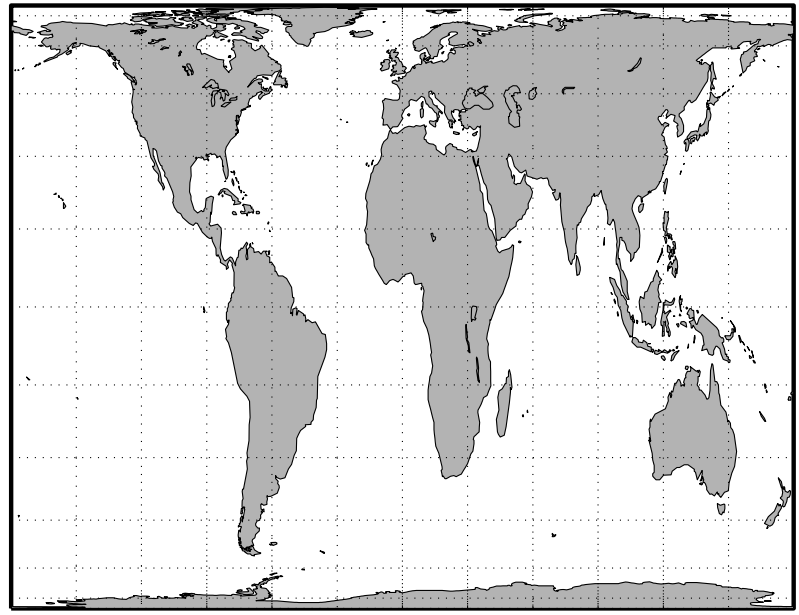
Poles: Straight lines equal in length to the Equator.

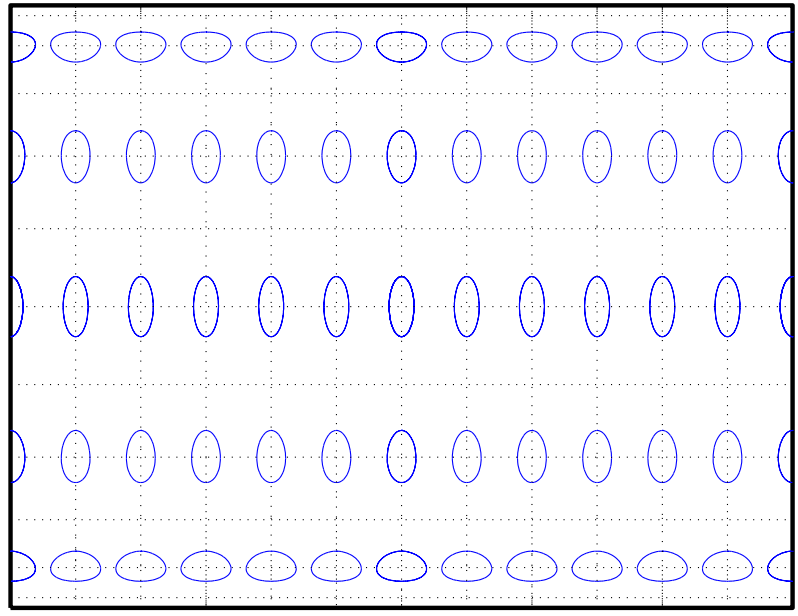
Symmetry: About any meridian or the Equator.

Features This is an orthographic projection onto a cylinder secant at the 50° parallels. It is equal-area, but distortion of shape increases with distance from the standard parallels. Scale is true along the standard parallels and constant between two parallels equidistant from the Equator. This projection is not equidistant.

Parallels For cylindrical projections, only one standard parallel is specified. The other standard parallel is the same latitude with the opposite sign. For this projection, the standard parallel is by definition fixed at 50°.

Remarks The Balthasart Cylindrical projection was presented in 1935 and is a special form of the Equal-Area Cylindrical projection secant at 50°N and S.





Behrmann Cylindrical Projection

Classification Cylindrical

Syntax `behrmann`

Graticule Meridians: Equally spaced straight parallel lines 0.42 as long as the Equator.

Parallels: Unequally spaced straight parallel lines, perpendicular to the meridians. Spacing is closest near the poles.

Poles: Straight lines equal in length to the Equator.

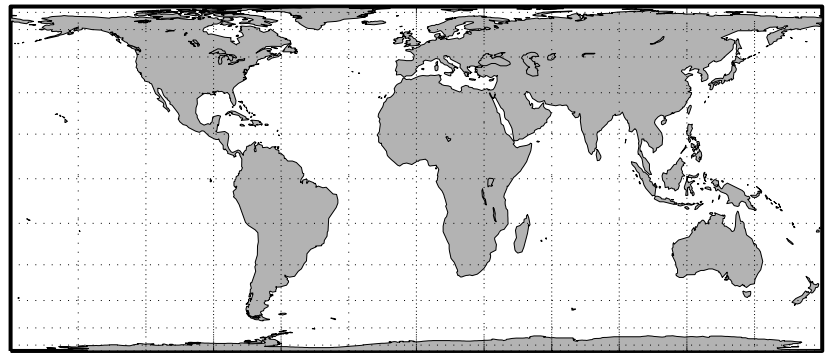
Symmetry: About any meridian or the Equator.

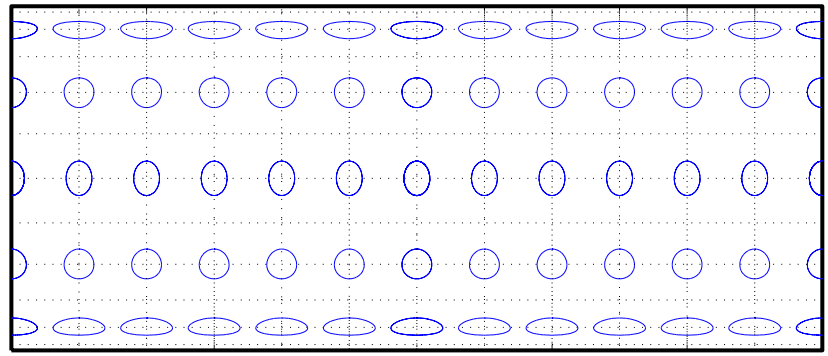
Features This is an orthographic projection onto a cylinder secant at the 30° parallels. It is equal-area, but distortion of shape increases with distance from the standard parallels. Scale is true along the standard parallels and constant between two parallels equidistant from the Equator. This projection is not equidistant.

Parallels For cylindrical projections, only one standard parallel is specified. The other standard parallel is the same latitude with the opposite sign. For this projection, the standard parallel is by definition fixed at 30°.

Remarks This projection is named for Walter Behrmann, who presented it in 1910 and is a special form of the Equal-Area Cylindrical projection secant at 30°N and S.

Behrmann Cylindrical Projection





Bolshoi Sovietskii Atlas Mira Projection

Classification Cylindrical

Syntax `bsam`

Graticule Meridians: Equally spaced straight parallel lines.

Parallels: Unequally spaced straight parallel lines, perpendicular to the meridians. Spacing increases toward the poles.

Poles: Straight lines equal in length to the Equator.

Symmetry: About any meridian or the Equator.

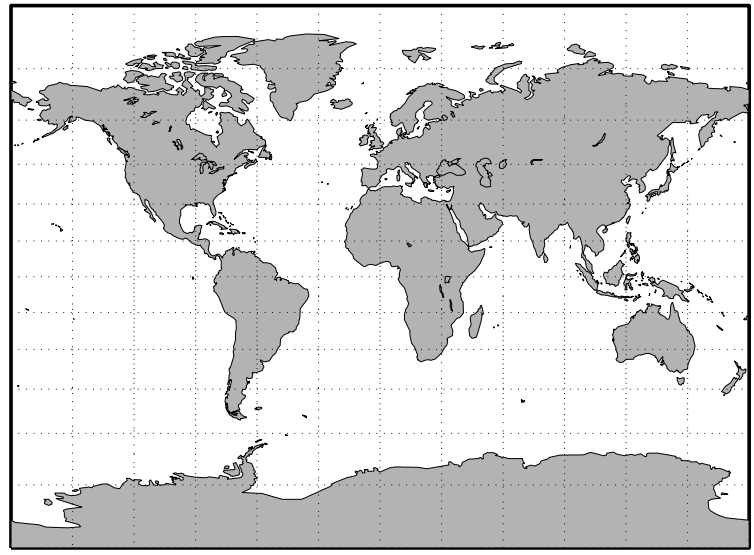
Features This is a perspective projection from a point on the Equator opposite a given meridian onto a cylinder secant at the 30° parallels. It is not equal-area, equidistant, or conformal. Scale is true along the standard parallels and constant between two parallels equidistant from the Equator. There is no distortion along the standard parallels, but it increases moderately away from these parallels, becoming severe at the poles.

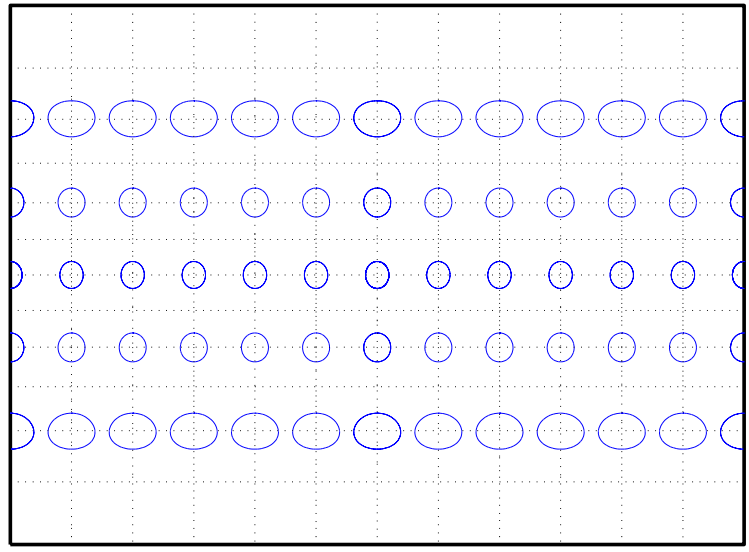
Parallels For cylindrical projections, only one standard parallel is specified. The other standard parallel is the same latitude with the opposite sign. For this projection, the standard parallel is by definition fixed at 30°.

Remarks This projection was first described in 1937, when it was used for maps in the *Bolshoi Sovietskii Atlas Mira* (Great Soviet World Atlas). It is commonly abbreviated as the BSAM projection. It is a special form of the Braun Perspective Cylindrical projection secant at 30°N and S.

Limitations This projection is available for the spherical geoid only.

Bolshoi Sovietskii Atlas Mira Projection





Bonne Projection

Classification Pseudoconic

Syntax bonne

Graticule Central Meridian: A straight line.

Meridians: Complex curves connecting points equally spaced along each parallel and concave toward the central meridian.

Parallels: Concentric circular arcs spaced at true distances along the central meridian.

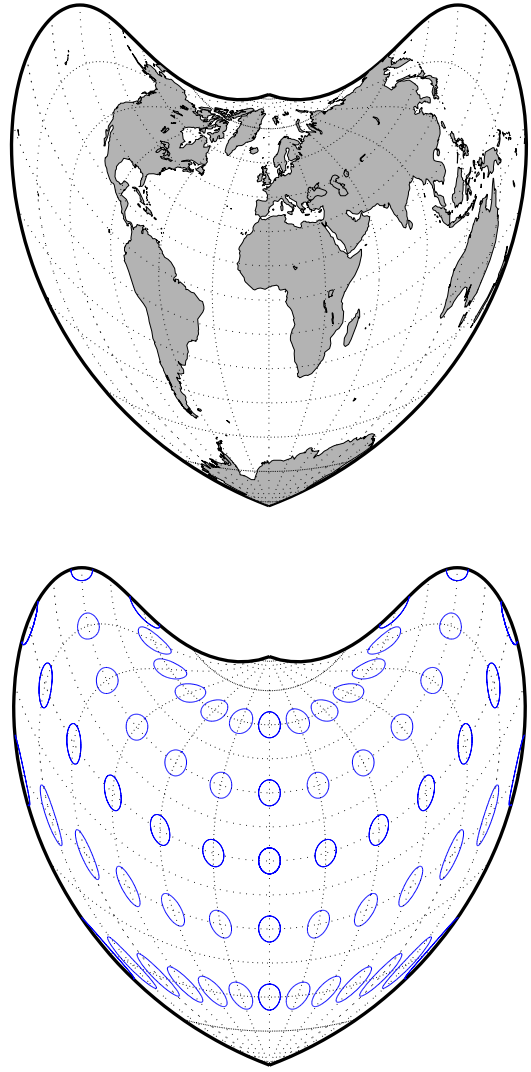
Poles: Points.

Symmetry: About the central meridian.

Features This is an equal-area projection. The curvature of the standard parallel is identical to that on a cone tangent at that latitude. The central meridian and the central parallel are free of distortion. This projection is not conformal.

Parallels This projection has one standard parallel, which is 30°N by default. It has two interesting limiting forms. If a pole is employed as the standard parallel, a Werner projection results; if the Equator is used, a Sinusoidal projection results.

Remarks This projection dates in a rudimentary form back to Claudius Ptolemy (about A.D. 100). It was further developed by Bernardus Sylvanus in 1511. It derives its name from its considerable use by Rigobert Bonne, especially in 1752.



Braun Perspective Cylindrical Projection

Classification Cylindrical

Syntax `braun`

Graticule Meridians: Equally spaced straight parallel lines.

Parallels: Unequally spaced straight parallel lines, perpendicular to the meridians. Spacing increases toward the poles.

Poles: Straight lines equal in length to the Equator.

Symmetry: About any meridian or the Equator.

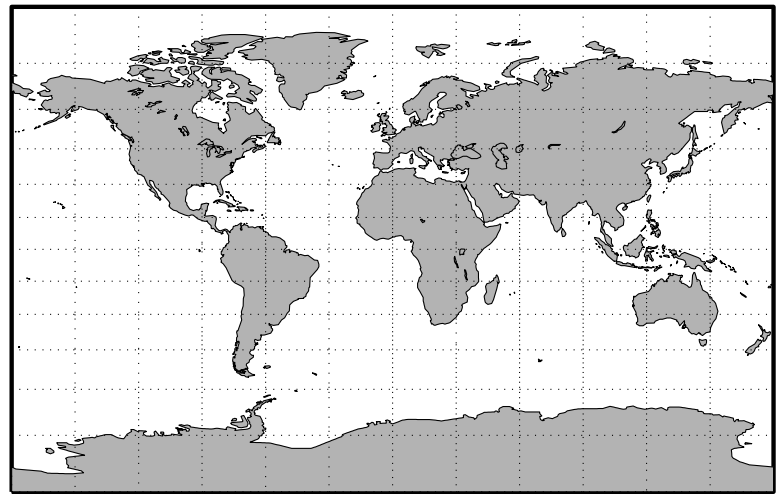
Features This is an perspective projection from a point on the Equator opposite a given meridian onto a cylinder secant at standard parallels. It is not equal-area, equidistant, or conformal. Scale is true along the standard parallels and constant between two parallels equidistant from the Equator. There is no distortion along the standard parallels, but it increases moderately away from these parallels, becoming severe at the poles.

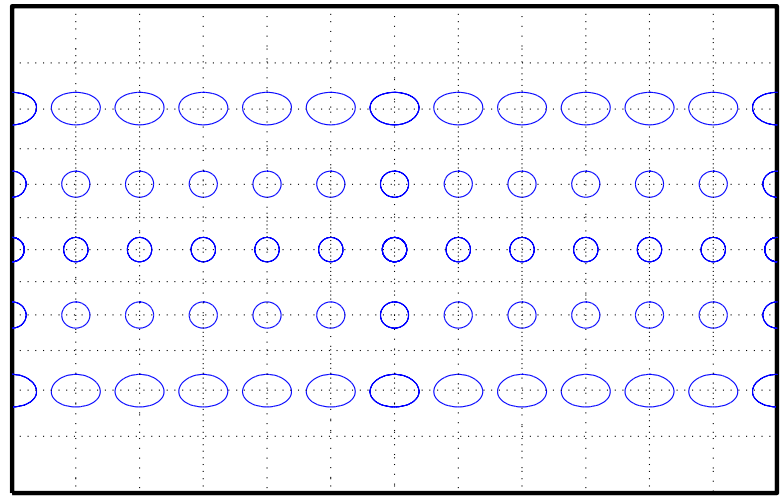
Parallels For cylindrical projections, only one standard parallel is specified. The other standard parallel is the same latitude with the opposite sign. For this projection, any latitude may be chosen; the default is arbitrarily set to 0°.

Remarks This projection was first described by Braun in 1867. It is less well known than the specific forms of it called the Gall Stereographic and the *Bolshoi Sovietskii Atlas Mira* projections.

Limitations This projection is available for the spherical geoid only.

Braun Perspective Cylindrical Projection





Breusing Harmonic Mean Projection

Classification Azimuthal

Syntax `breusing`

Graticule The graticule described is for the polar aspect.

Meridians: Equally spaced straight lines intersecting at the central pole.

Parallels: Unequally spaced circles centered on the central pole. The opposite hemisphere cannot be shown. Spacing increases (slightly) away from the central pole.

Poles: The central pole is a point, while the opposite pole cannot be shown.

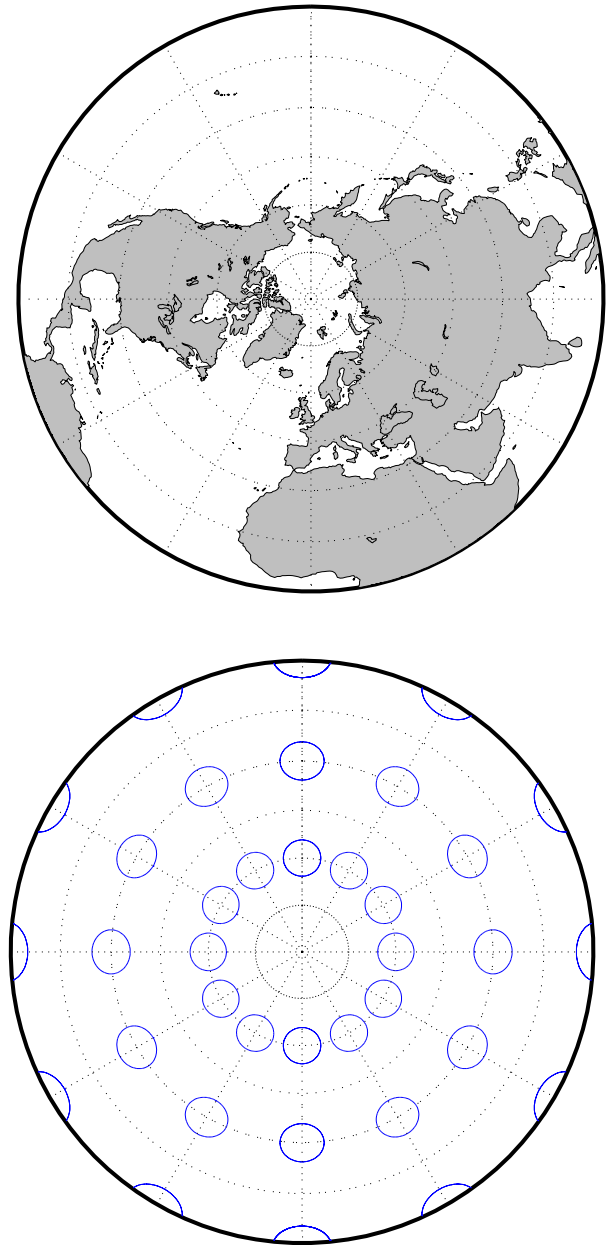
Symmetry: About any meridian.

Features This is a harmonic mean between a Stereographic and Lambert Equal-Area Azimuthal projection. It is not equal-area, equidistant, or conformal. There is no point at which scale is accurate in all directions. The primary feature of this projection is that it is minimum error – distortion is moderate throughout.

Parallels There are no standard parallels for azimuthal projections.

Remarks F. A. Arthur Breusing developed a geometric mean version of this projection in 1892. A. E. Young modified this to the harmonic mean version presented here in 1920. This projection is virtually indistinguishable from the Airy Minimum Error Azimuthal projection, presented by George Airy in 1861.

Limitations This projection is available for the spherical geoid only.



Briesemeister Projection

Classification Modified Azimuthal

Syntax `bries`

Graticule Meridians: Central meridian is straight. Other meridians are complex curves.

Parallels: Complex curves.

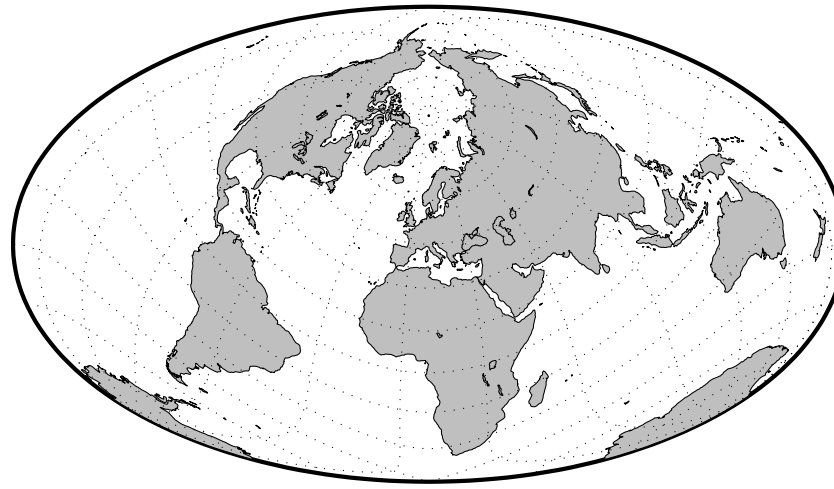
Poles: Points.

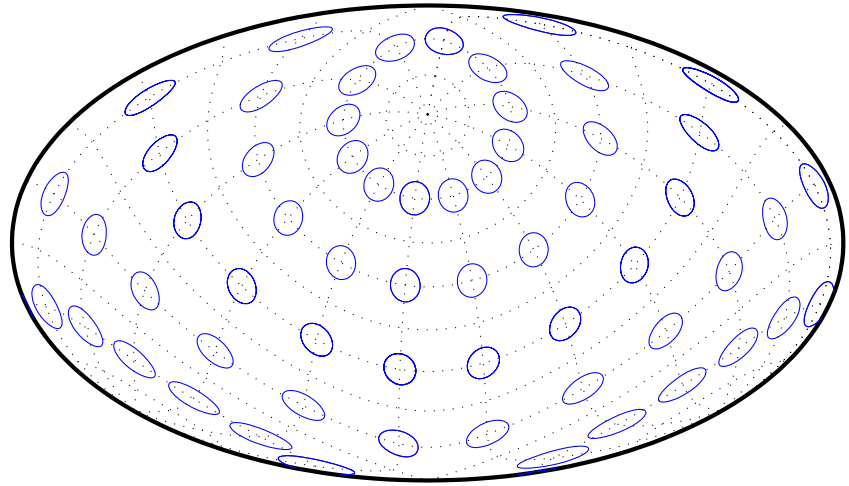
Symmetry: About the central meridian.

Features This equal-area projection groups the continents about the center of the projection. The only point free of distortion is the center point. Distortion of shape and area are moderate throughout.

Parallels There is no standard parallel for this projection.

Remarks This projection was presented by William Briesemeister in 1953. It is an oblique Hammer projection with an axis ratio of 1.75 to 1, instead of 2 to 1.





Cassini Cylindrical Projection

Classification Cylindrical

Syntax `cassini`

Graticule Central Meridian: Straight line (includes meridian opposite the central meridian in one continuous line).

Other Meridians: Straight lines if 90° from central meridian, complex curves concave toward the central meridian otherwise.

Parallels: Complex curves concave toward the nearest pole.

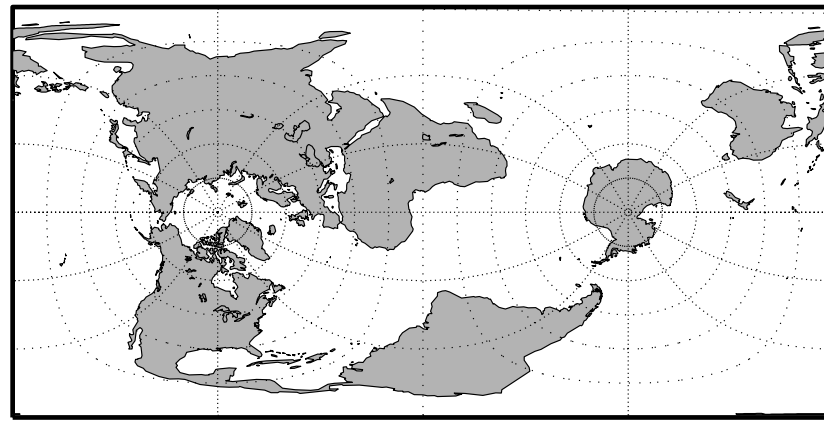
Poles: Points along the central meridian.

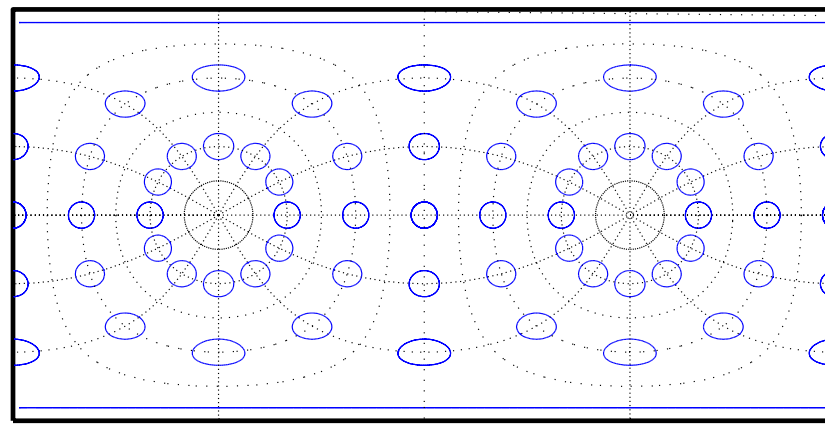
Symmetry: About any straight meridian or the Equator.

Features This is a projection onto a cylinder tangent at the central meridian. Distortion of both shape and area are functions of distance from the central meridian. Scale is true along the central meridian and along any straight line perpendicular to the central meridian (i.e., it is equidistant).

Parallels For cylindrical projections, only one standard parallel is specified. The other standard parallel is the same latitude with the opposite sign. For this projection, the standard parallel *of the base projection* is by definition fixed at 0°.

Remarks This projection is the transverse aspect of the Plate Carrée projection, developed by César François Cassini de Thury (1714-84). It is still used for the topographic mapping of a few countries.





Central Cylindrical Projection

Classification Cylindrical

Syntax `ccylin`

Graticule Meridians: Equally spaced straight parallel lines.

Parallels: Unequally spaced straight parallel lines, perpendicular to the meridians. Spacing increases toward the poles, more rapidly than that of the Mercator projection.

Poles: Cannot be shown.

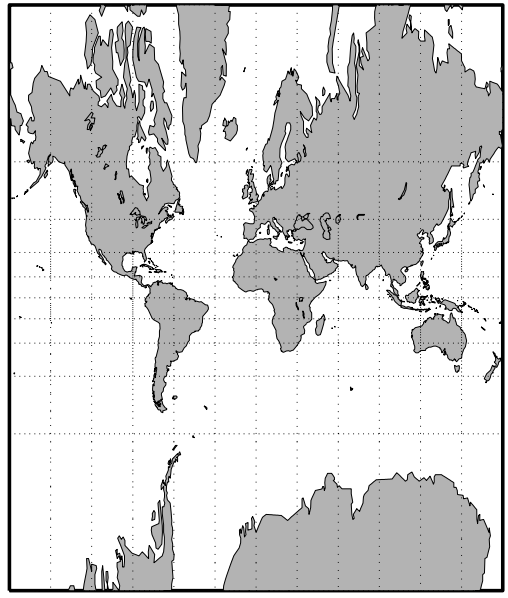
Symmetry: About any meridian or the Equator.

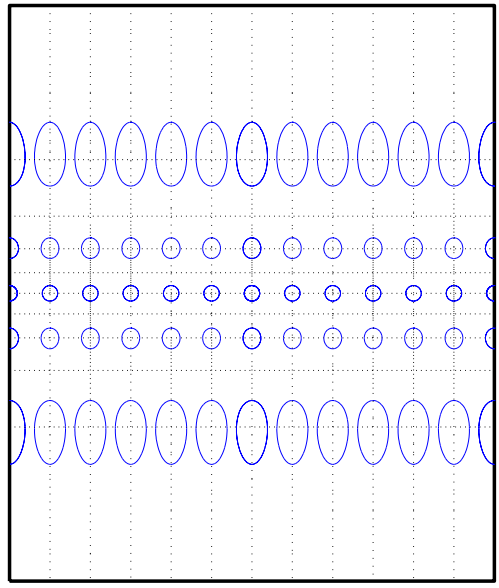
Features This is a perspective projection from the center of the Earth onto a cylinder tangent at the Equator. It is not equal-area, equidistant, or conformal. Scale is true along the Equator and constant between two parallels equidistant from the Equator. Scale becomes infinite at the poles. There is no distortion along the Equator, but it increases rapidly away from the Equator.

Parallels For cylindrical projections, only one standard parallel is specified. The other standard parallel is the same latitude with the opposite sign. For this projection, the standard parallel is by definition fixed at 0°.

Remarks The origin of this projection is unknown; it has little use beyond the educational aspects of its method of projection and as a comparison to the Mercator projection, which is not perspective. The transverse aspect of the Central Cylindrical is called the Wetch projection.

Limitations This projection is available for the spherical geoid only. Data at latitudes greater than 75° is trimmed to prevent large values from dominating the display.





Collignon Projection

Classification Pseudocylindrical

Syntax `collig`

Graticule Meridians: Equally spaced straight lines converging at the North Pole.

Parallels: Unequally spaced straight parallel lines, farthest apart near the North Pole, closest near the South Pole

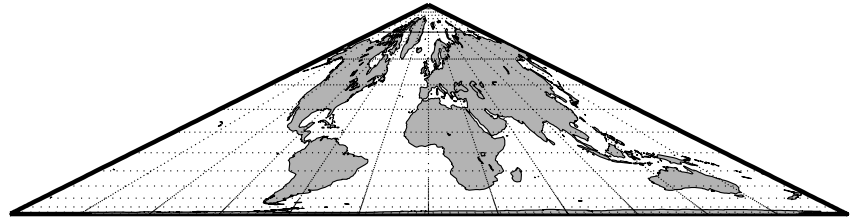
Poles: North Pole is a point, South Pole is a line 1.41 as long as the Equator.

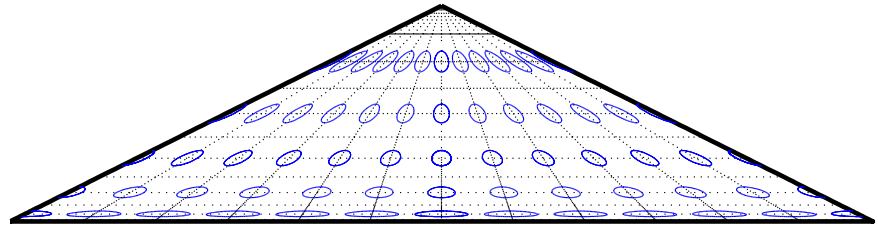
Symmetry: About the central meridian.

Features This is a novelty projection showing a straight-line, equal-area graticule. Scale is true along the 15°51'N parallel, constant along any parallel, and *different* for any pair of parallels. Distortion is severe in many regions, and is only absent at 15°51'N on the central meridian. This projection is not conformal or equidistant.

Parallels This projection has one standard parallel, which is by definition fixed at 15°51'.

Remarks This projection was presented by Édouard Collignon in 1865.





Craster Parabolic Projection

Classification Pseudocylindrical

Syntax `craster`

Graticule Central Meridian: Straight line half as long as the Equator.

Other Meridians: Equally spaced parabolas intersecting at the poles and concave toward the central meridian.

Parallels: Unequally spaced straight parallel lines, perpendicular to the central meridian. Spacing changes very gradually and is greatest near the Equator.

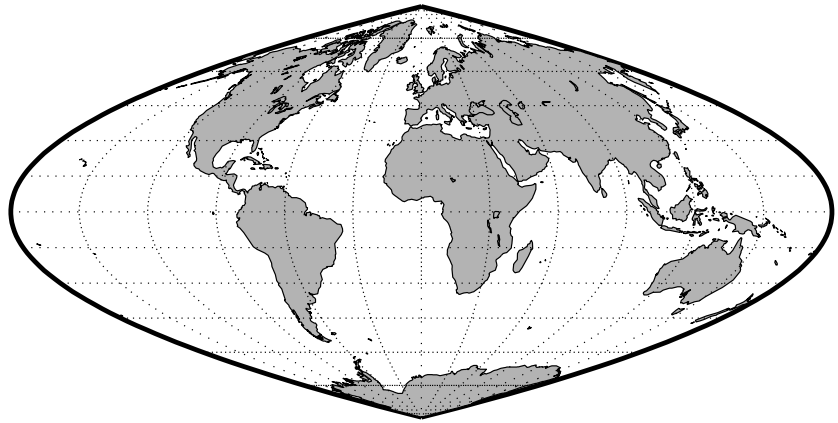
Poles: Points.

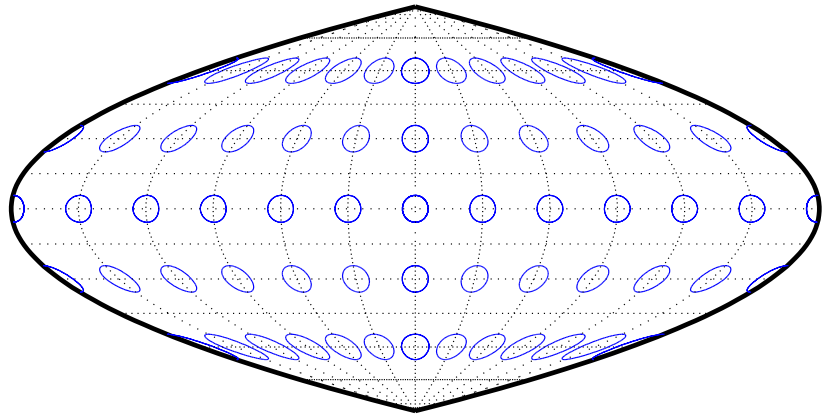
Symmetry: About the central meridian or the Equator.

Features This is an equal-area projection. Scale is true along the 36°46' parallels and is constant along any parallel and between any pair of parallels equidistant from the Equator. Distortion is severe near the outer meridians at high latitudes, but less so than the Sinusoidal projection. This projection is free of distortion only at the two points where the central meridian intersects the 36°46' parallels. This projection is not conformal or equidistant.

Parallels For this projection, only one standard parallel is specified. The other standard parallel is the same latitude with the opposite sign. The standard parallel is by definition fixed at 36°46'.

Remarks This projection was developed by John Evelyn Edmund Craster in 1929; it was further developed by Charles H. Deetz and O.S. Adams in 1934. It was presented independently in 1934 by Putnins as his P_4 projection.





Eckert I Projection

Classification Pseudocylindrical

Syntax `eckert1`

Graticule Central Meridian: Straight line half as long as the Equator.

Other Meridians: Equally spaced straight converging lines broken at the Equator.

Parallels: Equally spaced straight parallel lines, perpendicular to the central meridian.

Poles: Lines half as long as the Equator.

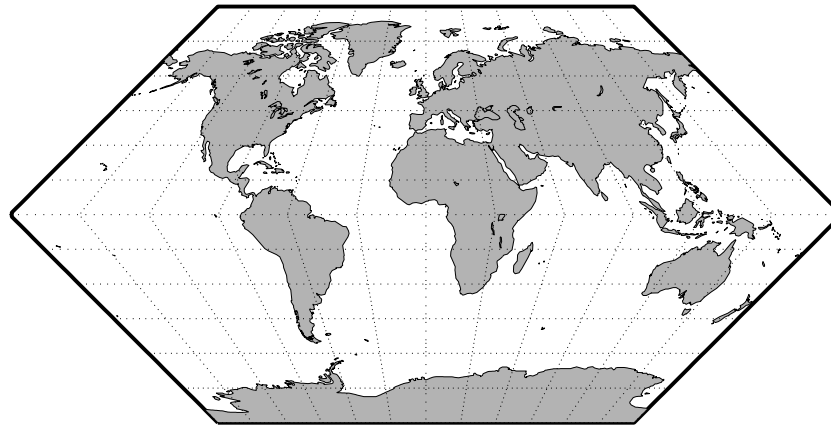
Symmetry: About the central meridian or the Equator.

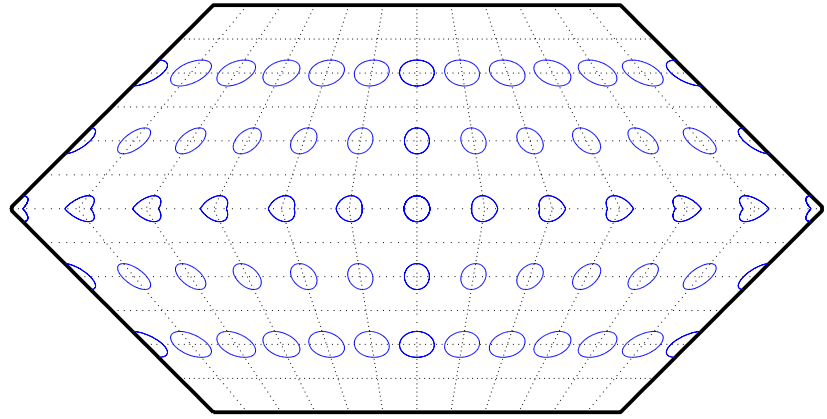
Features Scale is true along the 47°10' parallels and is constant along any parallel, between any pair of parallels equidistant from the Equator, and along any given meridian. It is not free of distortion at any point, and the break at the Equator introduces excessive distortion there; regardless of the appearance here, the Tissot indicatrices are of indeterminate shape along the Equator. This novelty projection is not equal-area or conformal.

Parallels For this projection, only one standard parallel is specified. The other standard parallel is the same latitude with the opposite sign. The standard parallel is by definition fixed at 47°10'.

Remarks This projection was presented by Max Eckert in 1906.

Limitations This projection is available for the spherical geoid only.





Eckert II Projection

Classification Pseudocylindrical

Syntax `eckert2`

Graticule Central Meridian: Straight line half as long as the Equator.

Other Meridians: Equally spaced straight converging lines broken at the Equator.

Parallels: Unequally spaced straight parallel lines, perpendicular to the central meridian. Spacing is widest near the Equator.

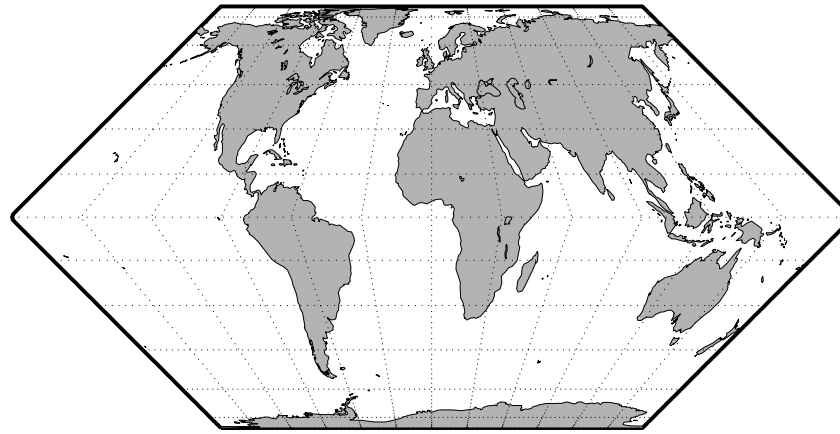
Poles: Lines half as long as the Equator.

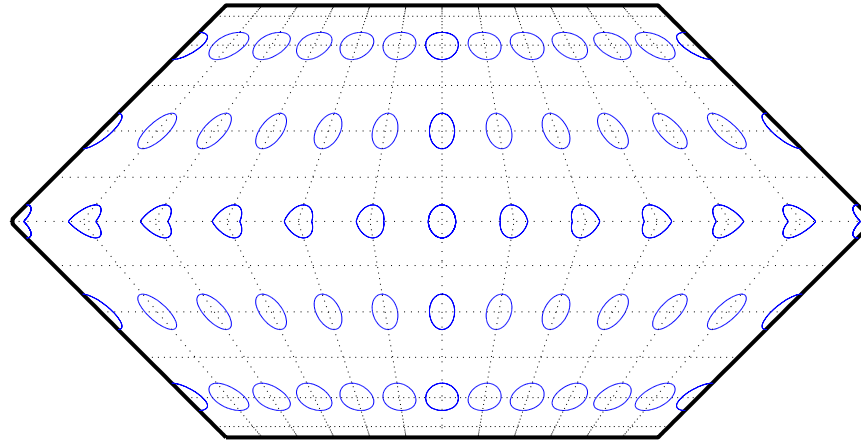
Symmetry: About the central meridian or the Equator.

Features This is an equal-area projection. Scale is true along the 55°10' parallels and is constant along any parallel and between any pair of parallels equidistant from the Equator. It is not free of distortion at any point except at 55°10'N and S along the central meridian; the break at the Equator introduces excessive distortion there. Regardless of the appearance here, the Tissot indicatrices are of indeterminate shape along the Equator. This novelty projection is not conformal or equidistant.

Parallels For this projection, only one standard parallel is specified. The other standard parallel is the same latitude with the opposite sign. The standard parallel is by definition fixed at 55°10'.

Remarks This projection was presented by Max Eckert in 1906.





Eckert III Projection

Classification Pseudocylindrical

Syntax `eckert3`

Graticule Central Meridian: Straight line half as long as the Equator.

Other Meridians: Equally spaced semiellipses concave toward the central meridian. The outer meridians, 180° east and west of the central meridian, are semicircles.

Parallels: Equally spaced straight parallel lines, perpendicular to the central meridian.

Poles: Lines half as long as the Equator.

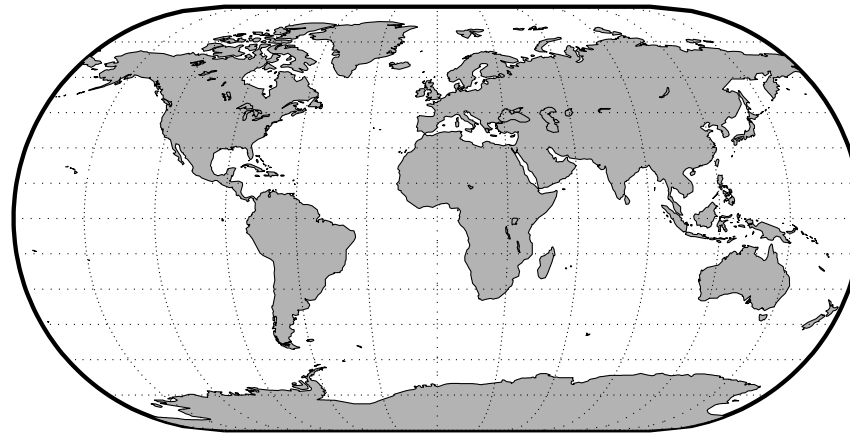
Symmetry: About the central meridian or the Equator.

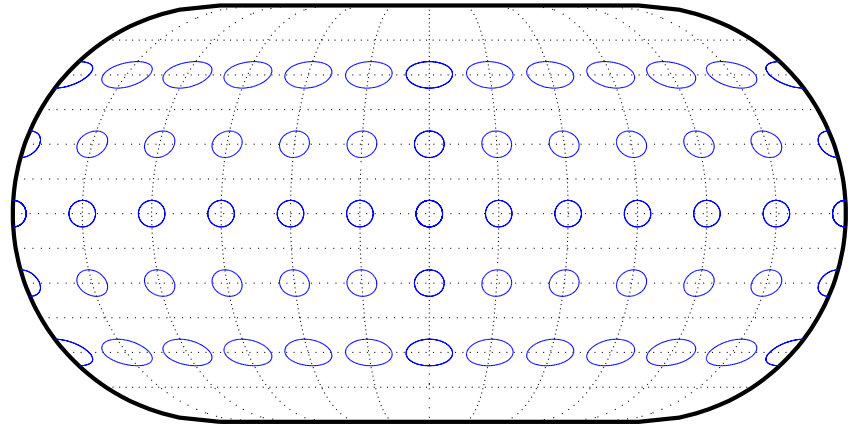
Features Scale is true along the 35°58' parallels and is constant along any parallel and between any pair of parallels equidistant from the Equator. No point is free of all scale distortion, but the Equator is free of angular distortion. This projection is not equal-area, conformal, or equidistant.

Parallels For this projection, only one standard parallel is specified. The other standard parallel is the same latitude with the opposite sign. The standard parallel is by definition fixed at 35°58'.

Remarks This projection was presented by Max Eckert in 1906.

Limitations This projection is available for the spherical geoid only.





Eckert IV Projection

Classification Pseudocylindrical

Syntax `eckert4`

Graticule Central Meridian: Straight line half as long as the Equator.

Other Meridians: Equally spaced semiellipses concave toward the central meridian. The outer meridians, 180° east and west of the central meridian, are semicircles.

Parallels: Unequally spaced straight parallel lines, perpendicular to the central meridian. Spacing is greatest toward the Equator.

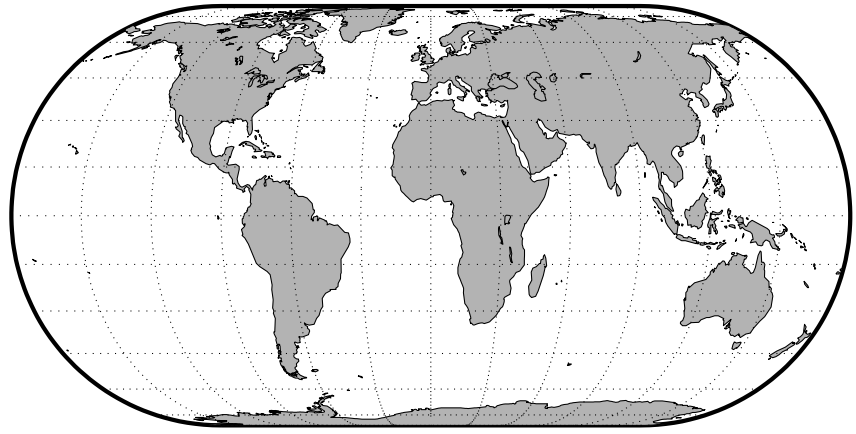
Poles: Lines half as long as the Equator.

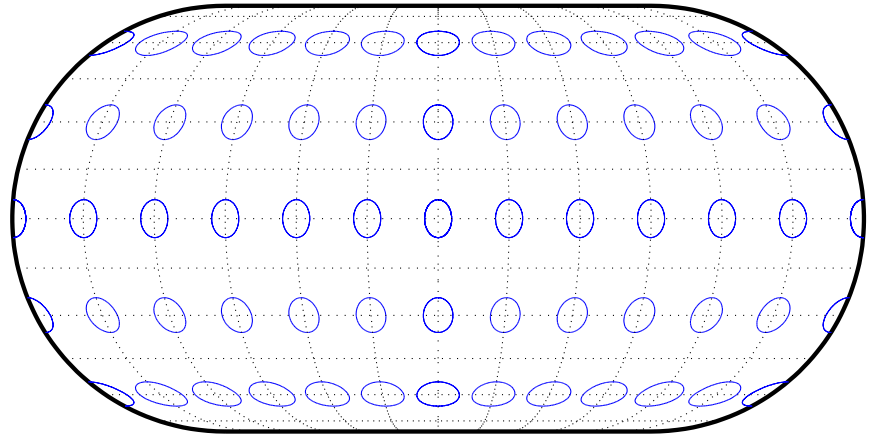
Symmetry: About the central meridian or the Equator.

Features This is an equal-area projection. Scale is true along the 40°30' parallels and is constant along any parallel and between any pair of parallels equidistant from the Equator. It is free of distortion only at the two points where the 40°30' parallels intersect the central meridian. This projection is not conformal or equidistant.

Parallels For this projection, only one standard parallel is specified. The other standard parallel is the same latitude with the opposite sign. The standard parallel is by definition fixed at 40°30'.

Remarks This projection was presented by Max Eckert in 1906.





Eckert V Projection

Classification Pseudocylindrical

Syntax `eckert5`

Graticule Central Meridian: Straight line half as long as the Equator.

Other Meridians: Equally spaced sinusoidal curves concave toward the central meridian.

Parallels: Equally spaced straight parallel lines, perpendicular to the central meridian.

Poles: Lines half as long as the Equator.

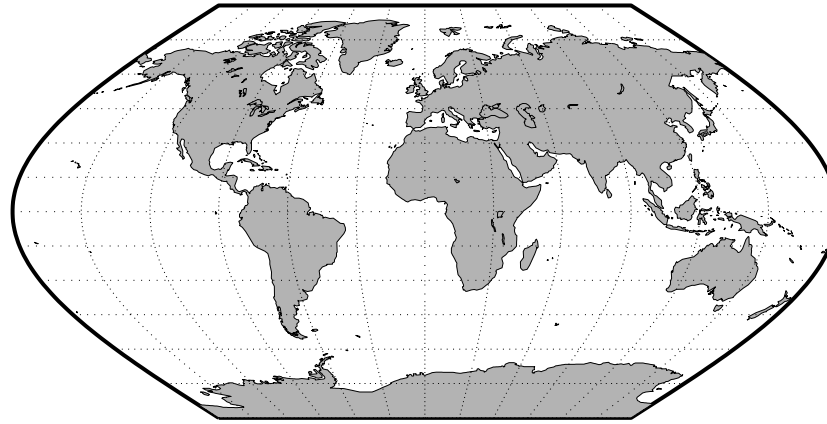
Symmetry: About the central meridian or the Equator.

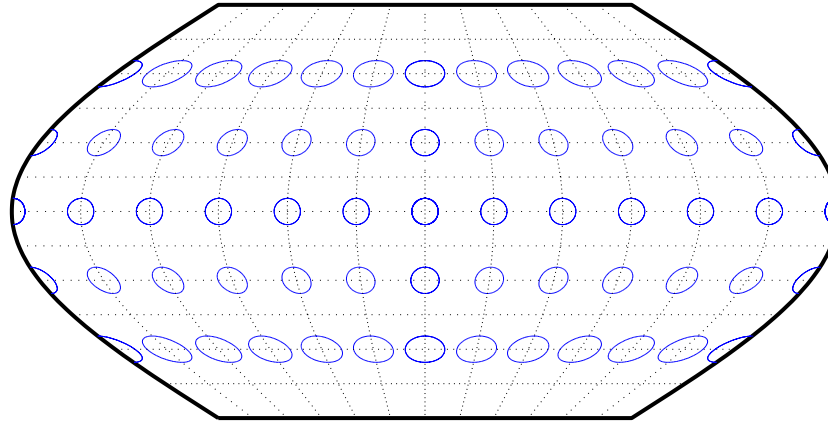
Features This projection is an arithmetic average of the *x* and *y* coordinates of the Sinusoidal and Plate Carrée projections. Scale is true along latitudes 37°55'N and S, and is constant along any parallel and between any pair of parallels equidistant from the Equator. There is no point free of all distortion, but the Equator is free of angular distortion. This projection is not equal-area, conformal, or equidistant.

Parallels This projection has one standard parallel, which is by definition fixed at 0°.

Remarks This projection was presented by Max Eckert in 1906.

Limitations This projection is available for the spherical geoid only.





Eckert VI Projection

Classification Pseudocylindrical

Syntax `eckert6`

Graticule Central Meridian: Straight line half as long as the Equator.

Other Meridians: Equally spaced sinusoidal curves concave toward the central meridian.

Parallels: Unequally spaced straight parallel lines, perpendicular to the central meridian. Spacing is greatest toward the Equator.

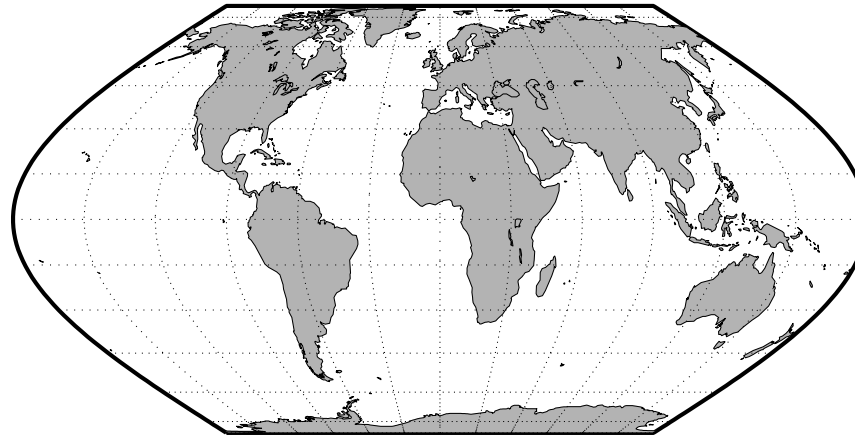
Poles: Lines half as long as the Equator.

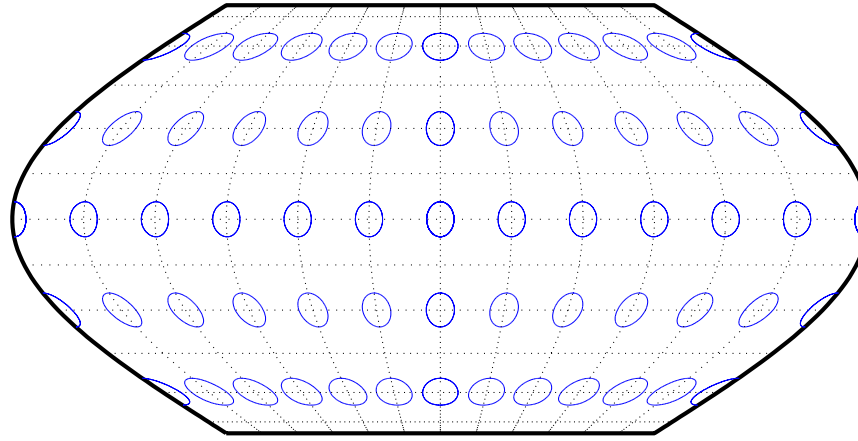
Symmetry: About the central meridian or the Equator.

Features This is an equal-area projection. Scale is true along the 49°16' parallels and is constant along any parallel and between any pair of parallels equidistant from the Equator. It is free of distortion only at the two points where the 49°16' parallels intersect the central meridian. This projection is not conformal or equidistant.

Parallels For this projection, only one standard parallel is specified. The other standard parallel is the same latitude with the opposite sign. The standard parallel is by definition fixed at 49°16'.

Remarks This projection was presented by Max Eckert in 1906.





Equal-Area Cylindrical Projection

Classification Cylindrical

Syntax `eqacylin`

Graticule Meridians: Equally spaced straight parallel lines.

Parallels: Unequally spaced straight parallel lines, perpendicular to the meridians. Spacing is closest near the poles.

Poles: Straight lines equal in length to the Equator.

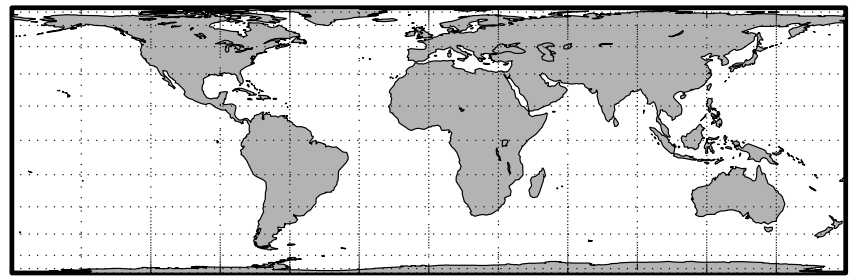
Symmetry: About any meridian or the Equator.

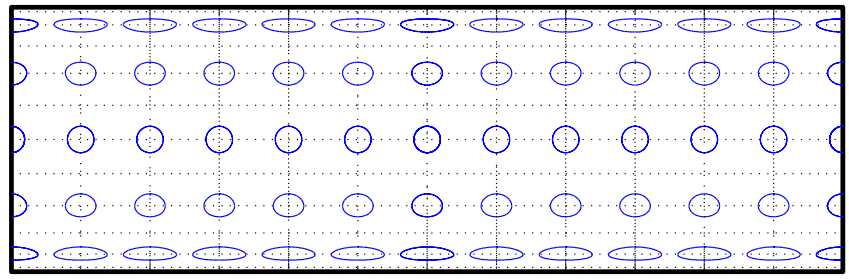
Features This is an orthographic projection onto a cylinder secant at the standard parallels. It is equal-area, but distortion of shape increases with distance from the standard parallels. Scale is true along the standard parallels and constant between two parallels equidistant from the Equator. This projection is not equidistant.

Parallels For cylindrical projections, only one standard parallel is specified. The other standard parallel is the same latitude with the opposite sign. For this projection, any latitude may be chosen; the default is arbitrarily set to 0° (the Lambert variation).

Remarks This projection was proposed by Johann Heinrich Lambert (1772), a prolific cartographer who proposed seven different important projections. The form of this projection tangent at the Equator is often called the Lambert Equal-Area Cylindrical projection. That and other special forms of this projection are included separately in this guide, including the Gall Orthographic, the Behrmann Cylindrical, the Balthasart Cylindrical, and the Trystan Edwards Cylindrical projections.

Equal-Area Cylindrical Projection





Equidistant Azimuthal Projection

Classification Azimuthal

Syntax `eqdazim`

Graticule The graticule described is for the polar aspect.

Meridians: Equally spaced straight lines intersecting at a central pole. The angles between them are the true angles.

Parallels: Equally spaced circles, centered on the central pole. The entire Earth may be shown.

Poles: Central pole is a point. The opposite pole is a bounding circle with a radius twice that of the Equator.

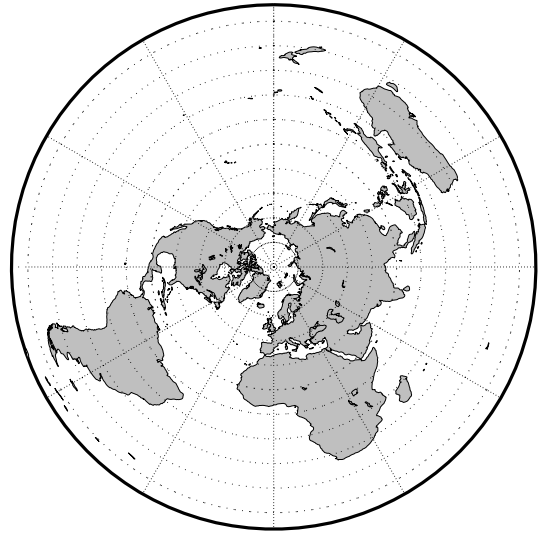
Symmetry: About any meridian.

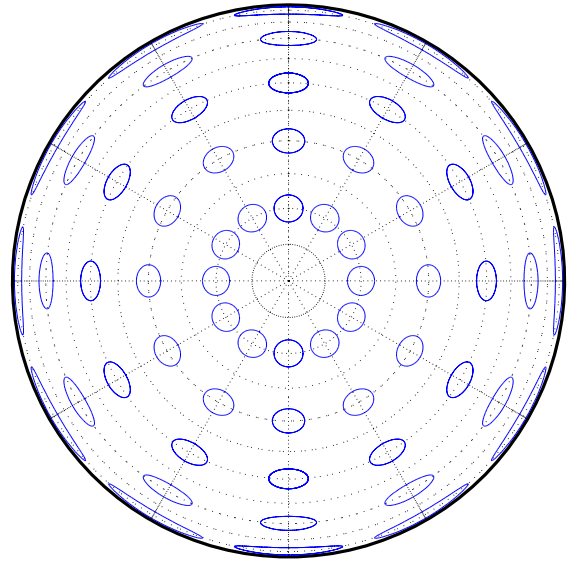
Features This is an equidistant projection. It is neither equal-area nor conformal. In the polar aspect, scale is true along any meridian. The projection is distortion free only at the center point. Distortion is moderate for the inner hemisphere, but it becomes extreme in the outer hemisphere.

Parallels There are no standard parallels for azimuthal projections.

Remarks This projection may have been first used by the ancient Egyptians for star charts. Several cartographers used it during the sixteenth century, including Guillaume Postel, who used it in 1581. Other names for this projection include Postel and Zenithal Equidistant.

Limitations This projection is available for the spherical geoid only.





Equidistant Conic Projection

Classification Conic

Syntax `eqdconic`

Graticule Meridians: Equally spaced straight lines converging to a common point, usually beyond the pole. The angles between the meridians are less than the true angles.

Parallels: Equally spaced concentric circular arcs centered on the point of meridanal convergence.

Poles: Normally circular arcs, enclosing the same angle as the displayed parallels.

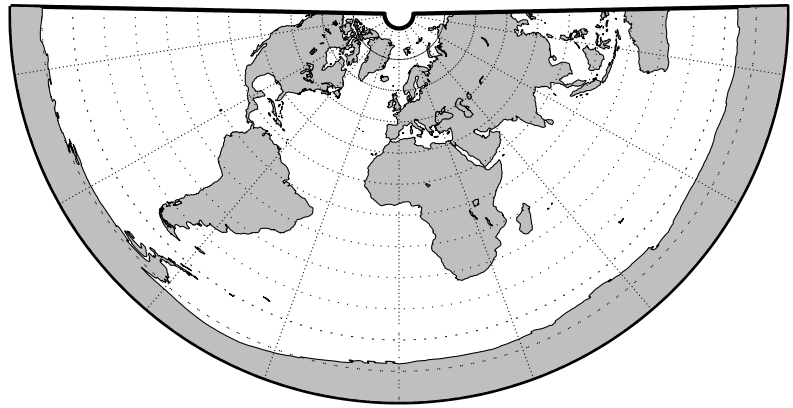
Symmetry: About any meridian.

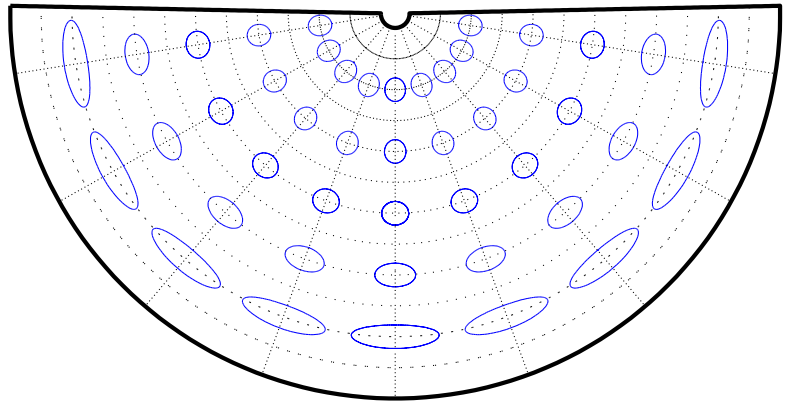
Features Scale is true along each meridian and the one or two selected standard parallels. Scale is constant along any parallel. This projection is free of distortion along the two standard parallels. Distortion is constant along any other parallel. This projection provides a compromise in distortion between conformal and equal-area conic projections, of which it is neither.

Parallels The cone of projection has interesting limiting forms. If a pole is selected as a single standard parallel, the cone is a plane, and an Equidistant Azimuthal projection results. If two parallels are chosen, not symmetric about the Equator, then an Equidistant Conic projection results. If a pole is selected as one of the standard parallels, then the projected pole is a point, otherwise the projected pole is an arc. If the Equator is so chosen, the cone becomes a cylinder and a Plate Carrée projection results. If two parallels equidistant from the Equator are chosen as the standard parallels, an Equidistant Cylindrical projection results. The default parallels are [15 75].

Remarks In a rudimentary form, this projection dates back to Claudius Ptolemy, about A.D. 100. Improvements were developed by Johannes Ruysch in 1508, Gerardus Mercator in the late 16th century, and Nicolas de l'Isle in 1745. It is also known as the Simple Conic or Conic projection.

Limitations Longitude data greater than 135° east or west of the central meridian is trimmed.





Equidistant Cylindrical Projection

Classification Cylindrical

Syntax `eqdcylin`

Graticule Meridians: Equally spaced straight parallel lines more than half as long as the Equator.

Parallels: Equally spaced straight parallel lines, perpendicular to and having wider spacing than the meridians.

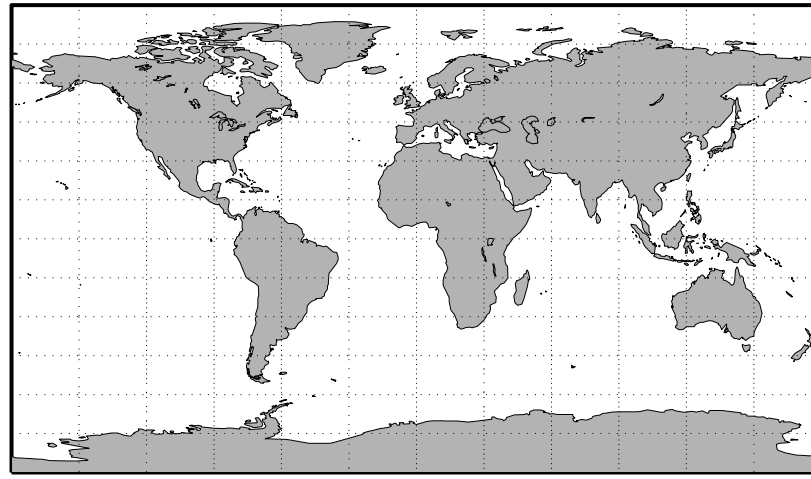
Poles: Straight lines equal in length to the Equator.

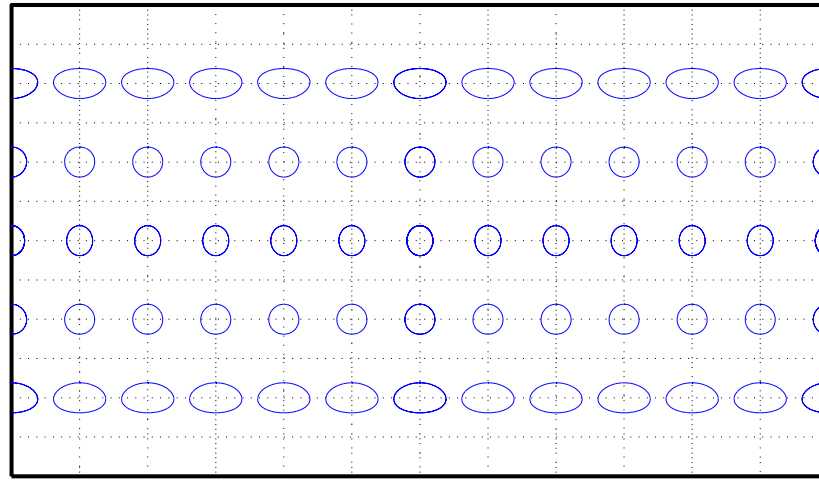
Symmetry: About any meridian or the Equator.

Features This is a projection onto a cylinder secant at the standard parallels. Distortion of both shape and area increase with distance from the standard parallels. Scale is true along all meridians (i.e., it is equidistant) and the standard parallels and is constant along any parallel and along the parallel of opposite sign.

Parallels For cylindrical projections, only one standard parallel is specified. The other standard parallel is the same latitude with the opposite sign. For this projection, any latitude can be chosen; the default is arbitrarily set to 30°.

Remarks This projection was first used by Marinus of Tyre about A.D. 100. Special forms of this projection are the Plate Carrée, with a standard parallel at 0°, and the Gall Isographic, with standard parallels at 45°N and S. Other names for this projection include Equirectangular, Rectangular, Projection of Marinus, *La Carte Parallélogrammatique*, and *Die Rechteckige Plattkarte*.





Fournier Projection

Classification Pseudocylindrical

Syntax `fournier`

Graticule Meridians: Equally spaced elliptical curves converging at the poles.

Parallels: Straight lines.

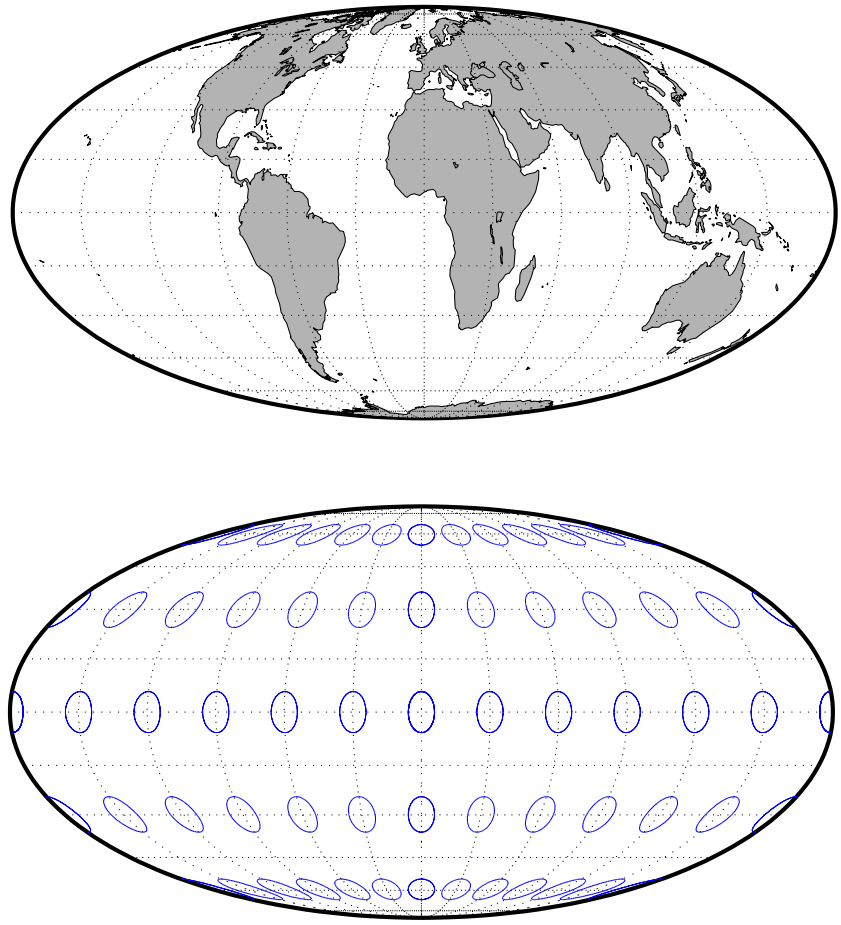
Poles: Points.

Symmetry: About the Equator and central meridian.

Features This projection is equal-area. Scale is constant along any parallel or pair of parallels equidistant from the Equator. This projection is neither equidistant nor conformal.

Parallels There is no standard parallel for this projection.

Remarks This projection was first described in 1643 by Georges Fournier. This is actually his second projection, the Fournier II.



Gall Isographic Projection

Purpose Cylindrical

Syntax `giso`

Graticule Meridians: Equally spaced straight parallel lines more than half as long as the Equator.

Parallels: Equally spaced straight parallel lines, perpendicular to and having wider spacing than the meridians.

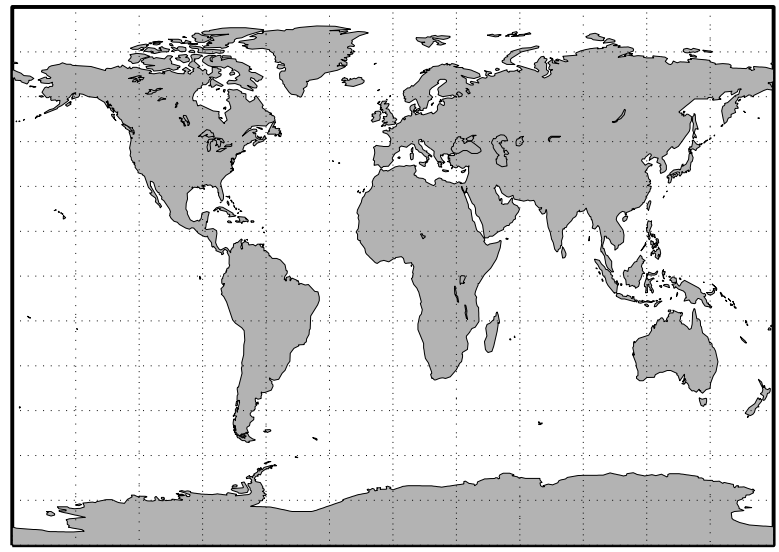
Poles: Straight lines equal in length to the Equator.

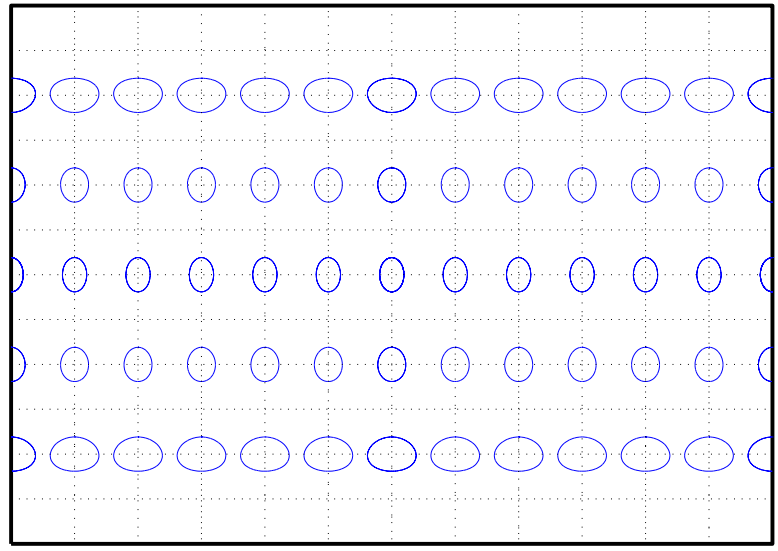
Symmetry: About any meridian or the Equator.

Features This is a projection onto a cylinder secant at the 45° parallels. Distortion of both shape and area increase with distance from the standard parallels. Scale is true along all meridians (i.e., it is equidistant) and the two standard parallels, and is constant along any parallel and along the parallel of opposite sign.

Parallels For cylindrical projections, only one standard parallel is specified. The other standard parallel is the same latitude with the opposite sign. For this projection, the standard parallel is by definition fixed at 45°.

Remarks This projection is a specific case of the Equidistant Cylindrical projection, with standard parallels at 45°N and S.





Gall Orthographic Projection

Classification Cylindrical

Syntax `gortho`

Graticule Meridians: Equally spaced straight parallel lines.

Parallels: Unequally spaced straight parallel lines, perpendicular to the meridians. Spacing is closest near the poles.

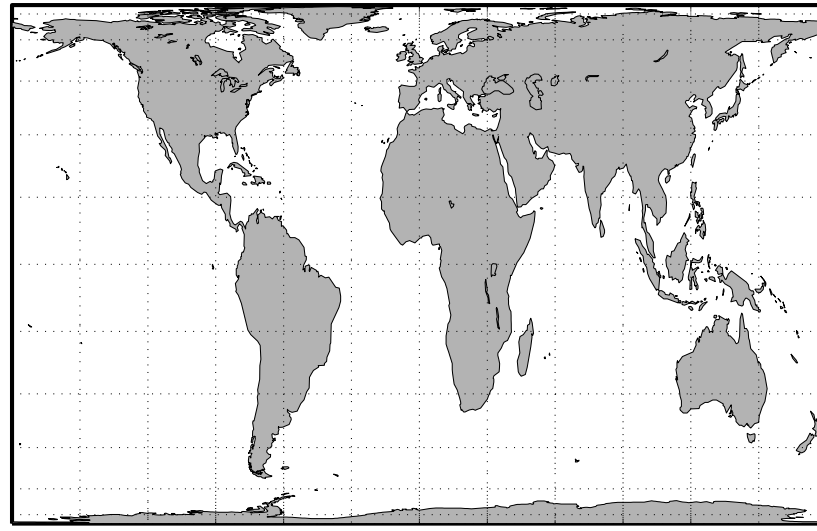
Poles: Straight lines equal in length to the Equator.

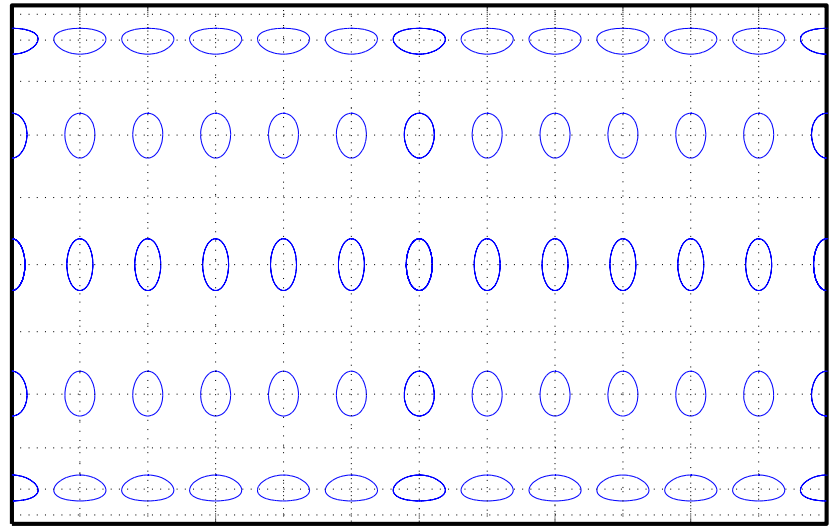
Symmetry: About any meridian or the Equator.

Features This is an orthographic projection onto a cylinder secant at the 45° parallels. It is equal-area, but distortion of shape increases with distance from the standard parallels. Scale is true along the standard parallels and constant between two parallels equidistant from the Equator. This projection is not equidistant.

Parallels For cylindrical projections, only one standard parallel is specified. The other standard parallel is the same latitude with the opposite sign. For this projection, the standard parallel is by definition fixed at 45°.

Remarks This projection is named for James Gall, who originated it in 1855 and is a special form of the Equal-Area Cylindrical projection secant at 45°N and S. This projection is also known as the Peters projection.





Gall Stereographic Projection

Classification Cylindrical

Syntax `gstereo`

Graticule Meridians: Equally spaced straight parallel lines 0.77 as long as the Equator.

Parallels: Unequally spaced straight parallel lines, perpendicular to the meridians. Spacing increases toward the poles.

Poles: Straight lines equal in length to the Equator.

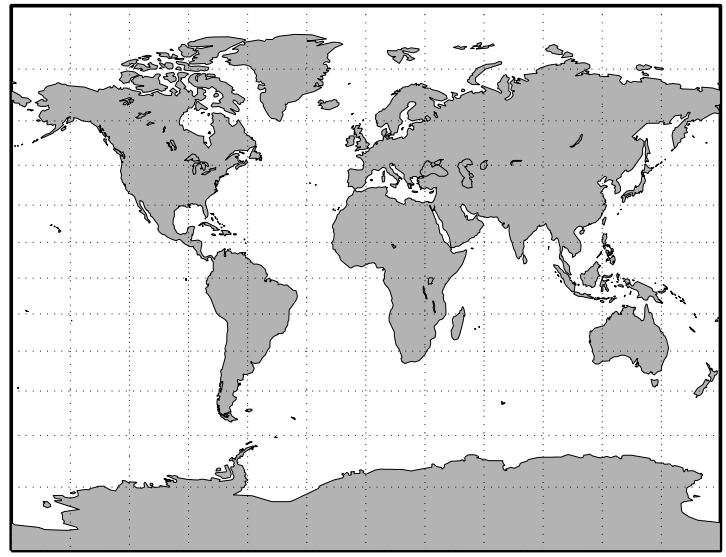
Symmetry: About any meridian or the Equator.

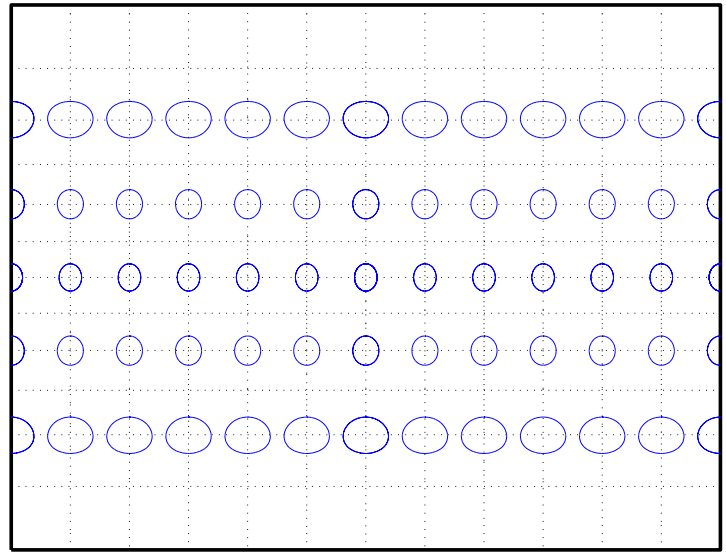
Features This is a perspective projection from a point on the Equator opposite a given meridian onto a cylinder secant at the 45° parallels. It is not equal-area, equidistant, or conformal. Scale is true along the standard parallels and constant between two parallels equidistant from the Equator. There is no distortion along the standard parallels, but it increases moderately away from these parallels, becoming severe at the poles.

Parallels For cylindrical projections, only one standard parallel is specified. The other standard parallel is the same latitude with the opposite sign. For this projection, the standard parallel is by definition fixed at 45°.

Remarks This projection was presented by James Gall in 1855. It is also known simply as the Gall projection. It is a special form of the Braun Perspective Cylindrical projection secant at 45°N and S.

Limitations This projection is available for the spherical geoid only.





Globe

Classification Spherical

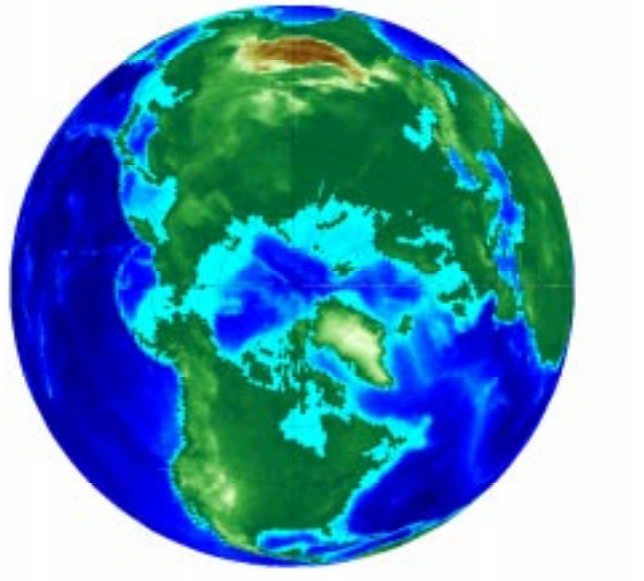
Syntax `globe`

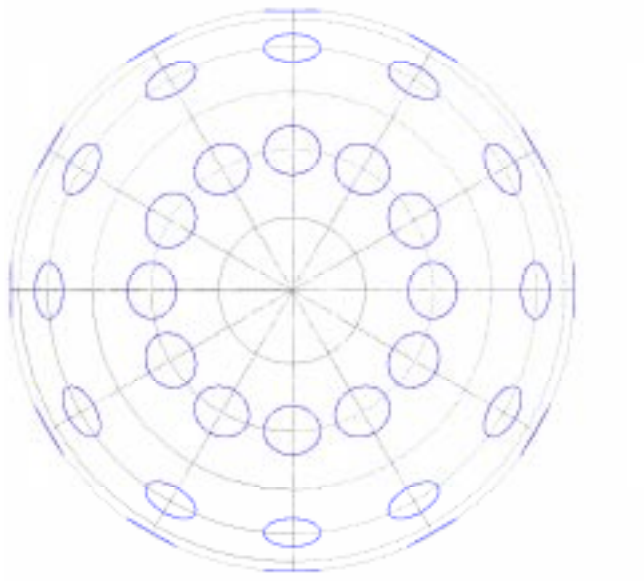
Graticule This *projection* is constructed by calculating a three-dimensional frame and displaying the map objects on the surface of this frame.

Features In the three-dimensional sense, this projection is true in scale, equal-area, conformal, minimum error, and equidistant everywhere. When displayed, however, this projection looks like an Orthographic azimuthal projection, provided that the MATLAB Axes `Projection` property is set to `'orthographic'`.

Parallels The globe requires no standard parallels.

Remarks This is the only three-dimensional representation provided for display. Unless some other display purpose requires three dimensions, the Orthographic projection's display is equivalent.





Gnomonic Projection

Classification Azimuthal

Syntax `gnomonic`

Graticule The graticule described is for a polar aspect.

Meridians: Equally spaced straight lines intersecting at the central pole. The angles displayed are the true angles between meridians.

Parallels: Unequally spaced circles centered on the central pole. Spacing increases rapidly away from this pole. The Equator and the opposite hemisphere cannot be shown

Pole: The central pole is a point; the other pole is not shown.

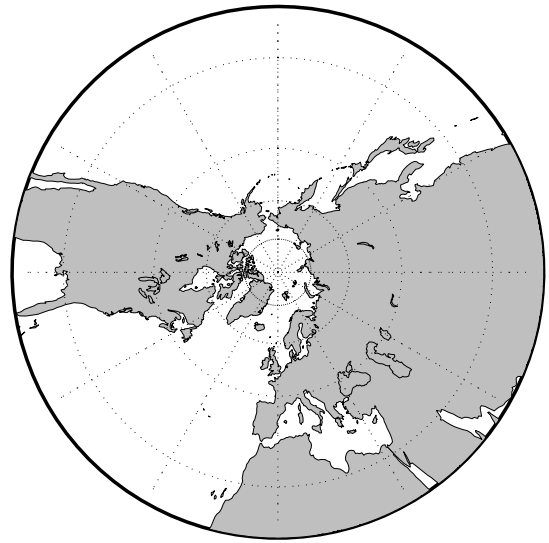
Symmetry: About any meridian.

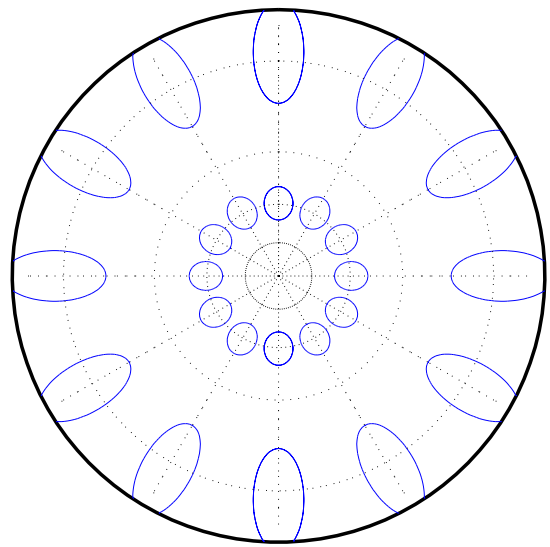
Features This is a perspective projection from the center of the globe on a plane tangent at the center point, which is a pole in the common polar aspect, but can be any point. Less than one hemisphere can be shown with this projection, regardless of its center point. The significant property of this projection is that all great circles are straight lines. This is useful in navigation, as a great circle is the shortest path between two points on the globe. Only the center point enjoys true scale and zero distortion. This projection is neither conformal nor equal-area.

Parallels There are no standard parallels for azimuthal projections.

Remarks This projection may have been first developed by Thales around 580 B.C. Its name is derived from the gnomon, the face of a sundial, since the meridians radiate like hour markings. This projection is also known as a Gnomic or Central projection.

Limitations This projection is available for the spherical geoid only. Data greater than 65° distant from the center point is trimmed.





Goode Homolosine Projection

Classification Pseudocylindrical

Syntax goode

Graticule Central Meridian: Straight line 0.44 as long as the Equator.

Other Meridians: Equally spaced sinusoidal curves between the 40°44'11.8" parallels and elliptical arcs elsewhere, all concave toward the central meridian. The result is a slight, visible bend in the meridians at 40°44'11.8" N and S.

Parallels: Straight parallel lines, perpendicular to the central meridian. Equally spaced between the 40°44'11.8" parallels, with gradually decreasing spacing outside these parallels.

Poles: Points.

Symmetry: About the central meridian or the Equator.

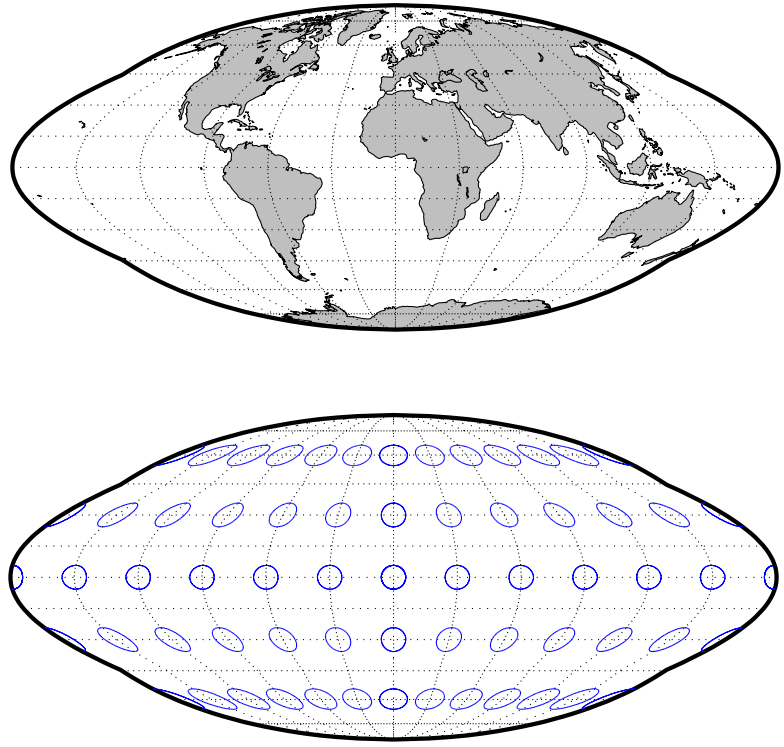
Features This is an equal-area projection. Scale is true along all parallels and the central meridian between 40°44'11.8" N and S, and is constant along any parallel and between any pair of parallels equidistant from the Equator for all latitudes. Its distortion is identical to that of the Sinusoidal projection between 40°44'11.8" N and S, and to that of the Mollweide projection elsewhere. This projection is not conformal or equidistant.

Parallels This projection has one standard parallel, which is by definition fixed at 0°.

Remarks This projection was developed by J. Paul Goode in 1916. It is sometimes called simply the Homolosine projection, and it is usually used in an interrupted form. It is a merging of the Sinusoidal and Mollweide projections.

Limitations This projection is available in an uninterrupted form only.

Goode Homolosine Projection



Hammer Projection

Classification Modified Azimuthal

Syntax `hammer`

Graticule Meridians: Central meridian is a straight line half the length of the Equator. Other meridians are complex curves, equally spaced along the Equator, and concave towards the central meridian.

Parallels: Equator is straight. Other parallels are complex curves, equally spaced along the central meridian, and concave towards the nearest pole.

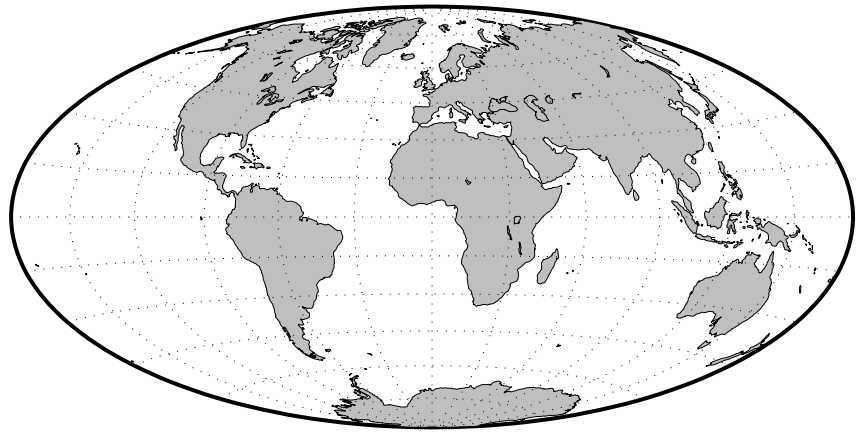
Poles: Points.

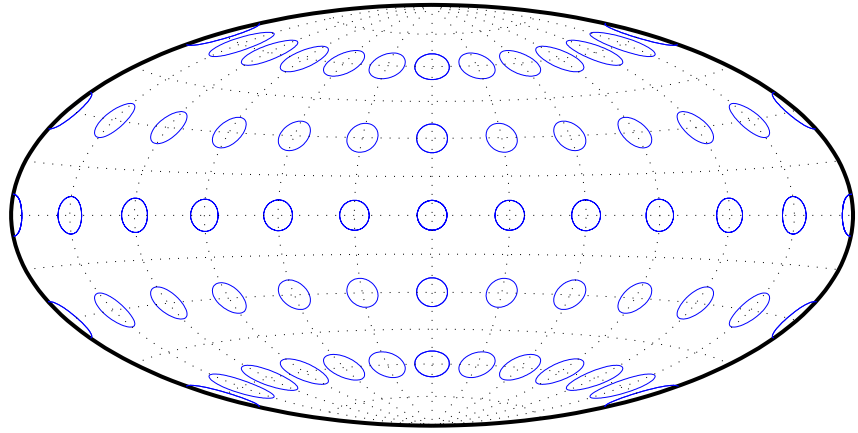
Symmetry: About the Equator and central meridian.

Features This projection is equal-area. The only point free of distortion is the center point. Distortion of shape is moderate throughout. This projection has less angular distortion on the outer meridians near the poles than pseudoazimuthal projections

Parallels There is no standard parallel for this projection.

Remarks This projection was presented by H. H. Ernst von Hammer in 1892. It is a modification of the Lambert Azimuthal Equal Area projection. Inspired by Aitoff projection, it is also known as the Hammer-Aitoff. It in turn inspired the Briesemeister, a modified oblique Hammer projection. John Bartholomew's Nordic projection is an oblique Hammer centered on 45 degrees north and the Greenwich meridian. The Hammer projection is used in whole-world maps and astronomical maps in galactic coordinates.





Hatano Asymmetrical Equal-Area Projection

Classification Pseudocylindrical

Syntax `hatano`

Graticule Central Meridian: Straight line 0.48 as long as the Equator.

Other Meridians: Equally spaced elliptical arcs concave toward the central meridian. The eccentricity of each ellipse changes at the Equator.

Parallels: Unequally spaced straight parallel lines, perpendicular to the central meridian. Spacing is not symmetrical about the Equator.

Poles: The North Pole is a line two-thirds the length of the Equator; the South Pole is a line three-fourths the length of the Equator.

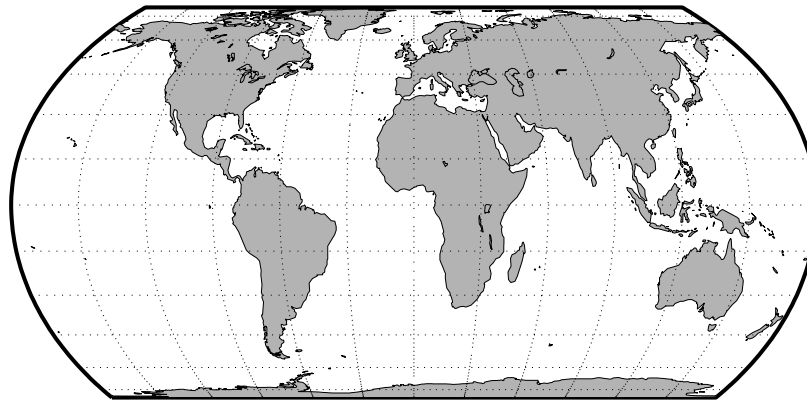
Symmetry: About the central meridian but *not* the Equator.

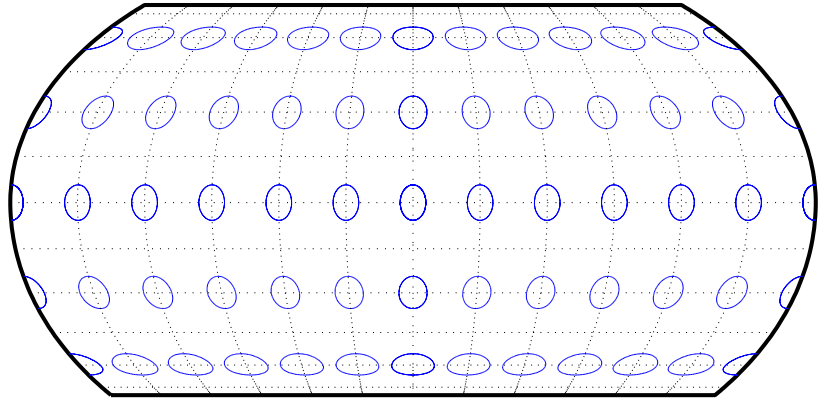
Features This is an equal-area projection. Scale is true along 40°42'N and 38°27'S, and is constant along any parallel but generally *not* between pairs of parallels equidistant from the Equator. It is free of distortion only along the central meridian at 40°42'N and 38°27'S. This projection is not conformal or equidistant.

Parallels Because of the asymmetrical nature of this projection, two standard parallels must be specified. The standard parallels are by definition fixed at 40°42'N and 38°27'S.

Remarks This projection was presented by Masataka Hatano in 1972.

Hatano Asymmetrical Equal-Area Projection





Kavraisky V Projection

Classification Pseudocylindrical

Syntax `kavrsky5`

Graticule Meridians: Complex curves converging at the poles. A sine function is used for *y*, but the meridians are not sine curves.

Parallels: Unequally spaced straight lines.

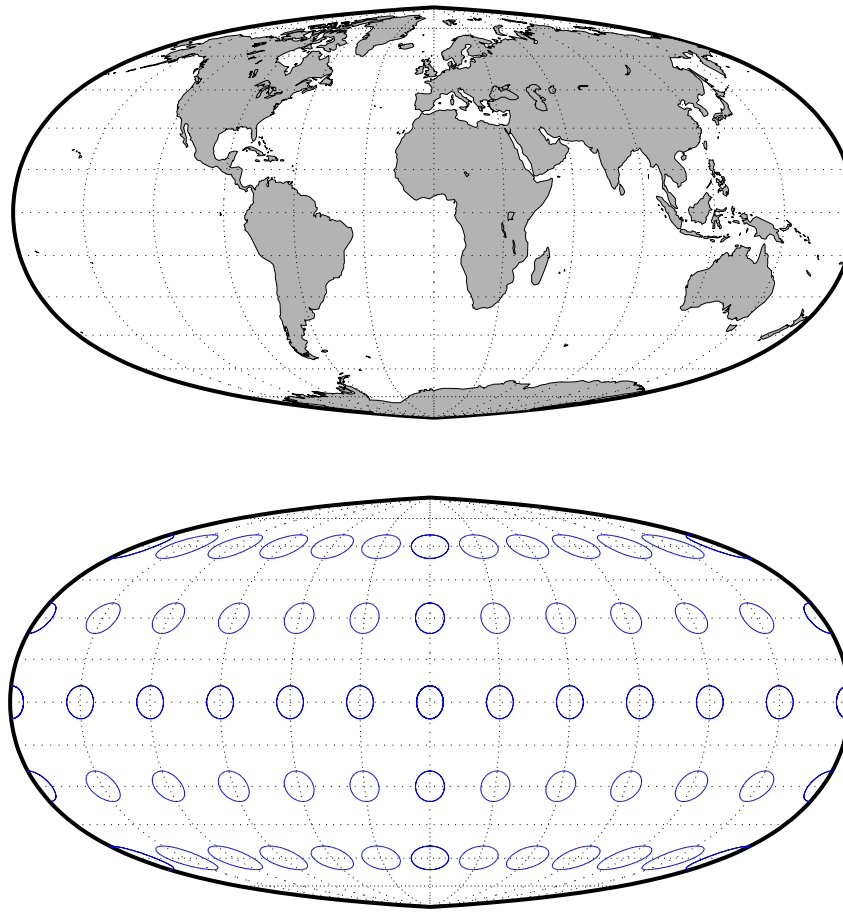
Poles: Points.

Symmetry: About the Equator and the central meridian.

Features This is an equal-area projection. Scale is true along the fixed standard parallels at 35°, and 0.9 true along the Equator. This projection is neither conformal nor equidistant.

Parallels The fixed standard parallels are at 35°.

Remarks This projection was described by V. V. Kavraisky in 1933.



Kavraisky VI Projection

Classification Pseudocylindrical

Syntax `kavrsky6`

Graticule Central Meridian: Straight line half the length of the Equator.

Meridians: Sine curves (60° segments).

Parallels: Unequally spaced straight lines.

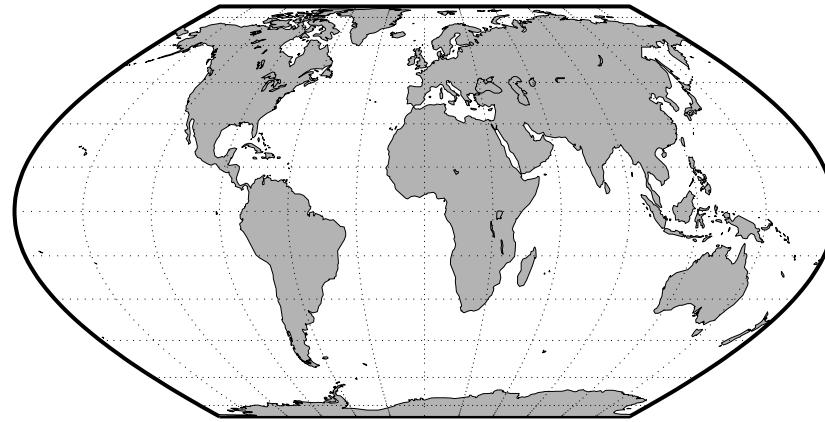
Poles: Straight lines half the length of the Equator.

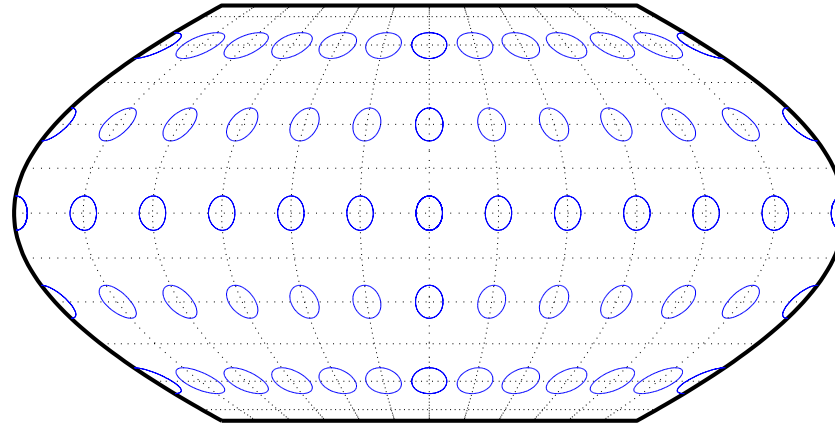
Symmetry: About the Equator and the central meridian.

Features This is an equal-area projection. Scale is constant along any parallel or pair of equidistant parallels. This projection is neither conformal nor equidistant.

Parallels There are no standard parallels for this projection.

Remarks This projection was described by V. V. Kavraisky in 1936. It is also called the Wagner I, for Karlheinz Wagner, who described it in 1932.





Lambert Azimuthal Equal-Area Projection

Classification Azimuthal

Syntax `eqaazim`

Graticule The graticule described is for a polar aspect.

Meridians: Equally spaced straight lines intersecting at the central pole. The angles displayed are the true angles between meridians.

Parallels: Unequally spaced circles centered on the central pole. The entire Earth can be shown. Spacing decreases away from the central pole.

Pole: The central pole is a point; the other pole is a bounding circle with 1.41 the radius of the Equator.

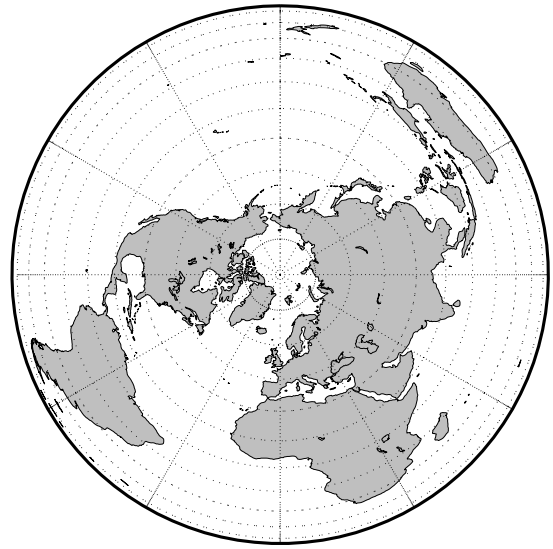
Symmetry: About any meridian.

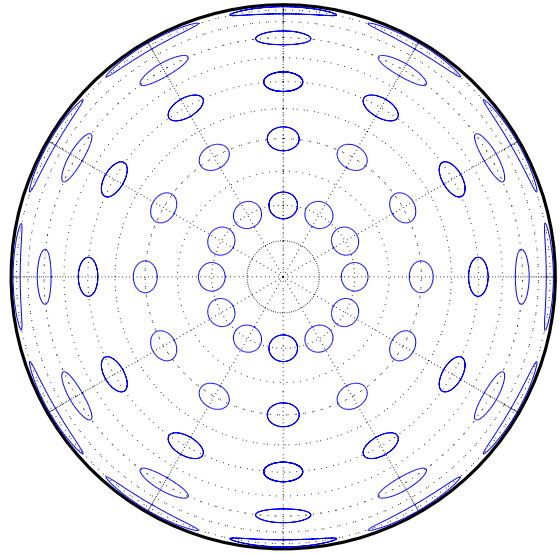
Features This nonperspective projection is equal-area. Only the center point is free of distortion, but distortion is moderate within 90° of this point. Scale is true only at the center point, increasing tangentially and decreasing radially with distance from the center point. This projection is neither conformal nor equidistant.

Parallels There are no standard parallels for azimuthal projections.

Remarks This projection was presented by Johann Heinrich Lambert in 1772. It is also known as the Zenithal Equal-Area and the Zenithal Equivalent projection, and the Lorgna projection in its polar aspect.

Limitations Data greater than 160° distant from the center point is trimmed.





Lambert Conformal Conic Projection

Classification Conic

Syntax `lambert`

Graticule Meridians: Equally spaced straight lines converging at one of the poles. The angles between the meridians are less than the true angles.

Parallels: Unequally spaced concentric circular arcs centered on the pole of convergence. Spacing of parallels increases away from the central latitudes.

Poles: The pole nearest a standard parallel is a point, the other cannot be shown.

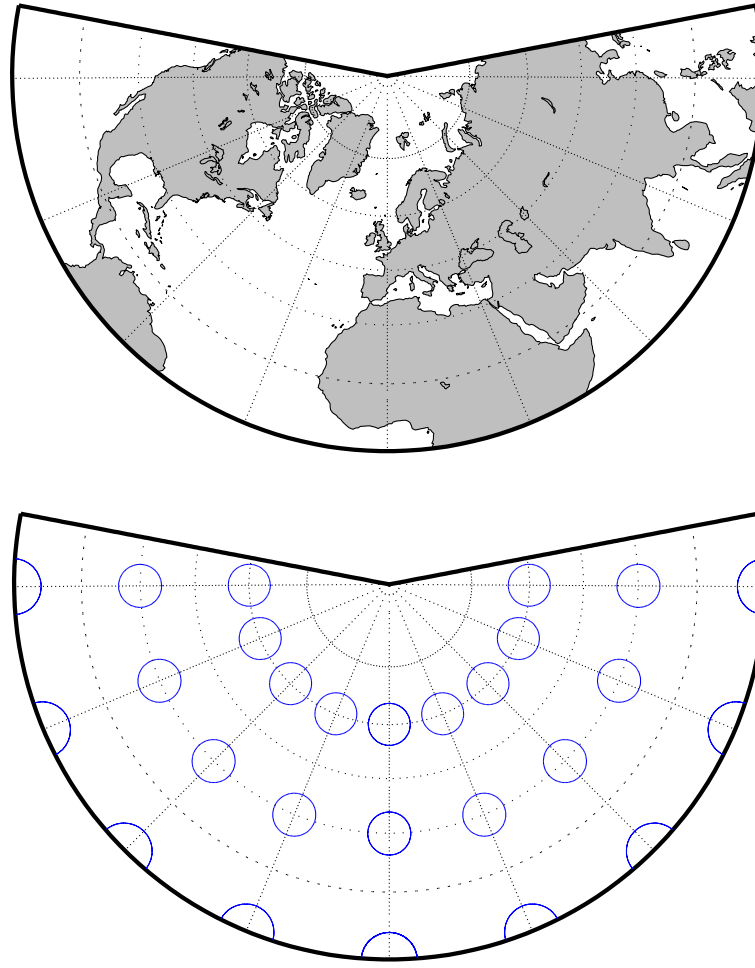
Symmetry: About any meridian.

Features Scale is true along the one or two selected standard parallels. Scale is constant along any parallel and is the same in every direction at any point. This projection is free of distortion along the standard parallels. Distortion is constant along any other parallel. This projection is conformal everywhere but the poles; it is neither equal-area nor equidistant.

Parallels The cone of projection has interesting limiting forms. If a pole is selected as a single standard parallel, the cone is a plane, and a Stereographic Azimuthal projection results. If two parallels are chosen, not symmetric about the Equator, then a Lambert Conformal Conic projection results. If a pole is selected as one of the standard parallels, then the projected pole is a point, otherwise the projected pole is an arc. If the Equator or two parallels equidistant from the Equator are chosen as the standard parallels, the cone becomes a cylinder, and a Mercator projection results. The default parallels are [15 75].

Remarks This projection was presented by Johann Heinrich Lambert in 1772 and is also known as a Conical Orthomorphic projection.

Limitations Longitude data greater than 135° east or west of the central meridian is trimmed. The default map limits are [0 90] to avoid extreme area distortion.



Lambert Equal-Area Cylindrical Projection

Classification Cylindrical

Syntax `lambcyln`

Graticule Meridians: Equally spaced straight parallel lines 0.32 as long as the Equator.

Parallels: Unequally spaced straight parallel lines, perpendicular to the meridians. Spacing is closest near the poles.

Poles: Straight lines equal in length to the Equator.

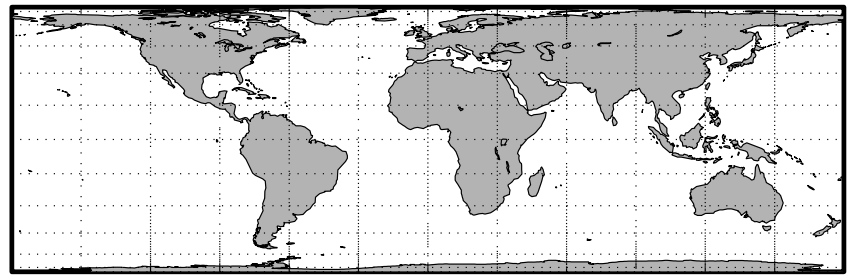
Symmetry: About any meridian or the Equator.

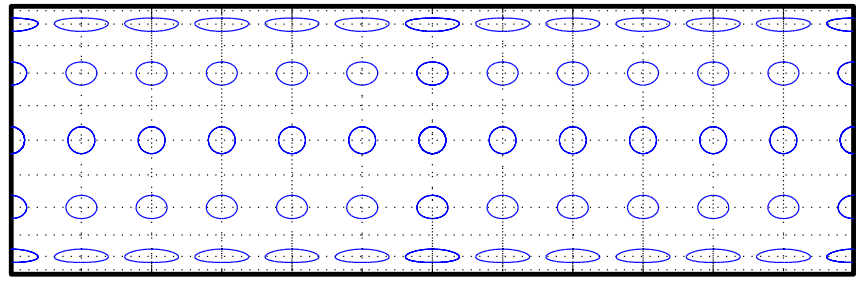
Features This is an orthographic projection onto a cylinder tangent at the Equator. It is equal-area, but distortion of shape increases with distance from the Equator. Scale is true along the Equator and constant between two parallels equidistant from the Equator. This projection is not equidistant.

Parallels For cylindrical projections, only one standard parallel is specified. The other standard parallel is the same latitude with the opposite sign. For this projection, the standard parallel is by definition fixed at 0°.

Remarks This projection is named for Johann Heinrich Lambert and is a special form of the Equal-Area Cylindrical projection tangent at the Equator.

Lambert Equal-Area Cylindrical Projection





Loximuthal Projection

Classification Pseudocylindrical

Syntax `loximuth`

Graticule Central Meridian: Straight line at least half as long as the Equator. Actual length depends on the choice of central latitude. Length is 0.5 when the central latitude is the Equator, for example, and 0.65 for central latitudes of 40°.

Other Meridians: Complex curves intersecting at the poles and concave toward the central meridian.

Parallels: Equally spaced straight parallel lines, perpendicular to the central meridian.

Poles: Points.

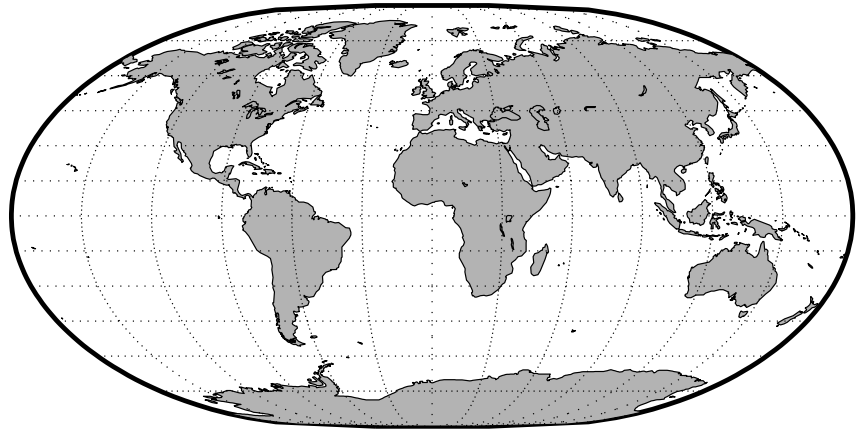
Symmetry: About the central meridian. Symmetry about the Equator only when it is the central latitude.

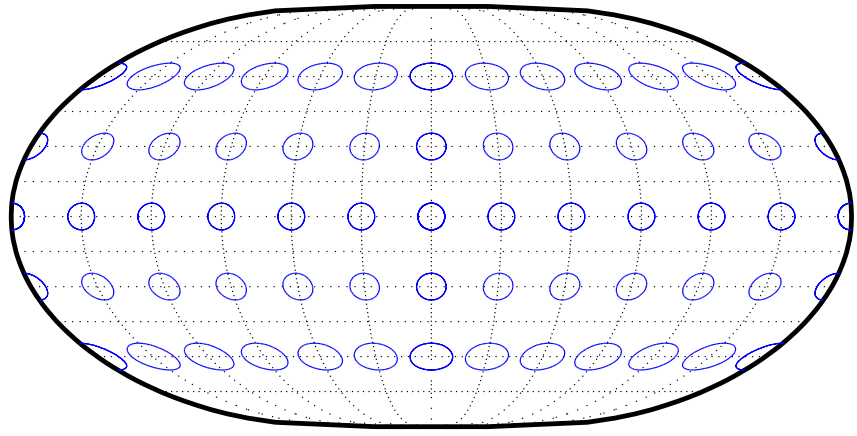
Features This projection has the special property that from the central point (the intersection of the central latitude with the central meridian), rhumb lines (loxodromes) are shown as straight, true to scale, and correct in azimuth from the center. This differs from the Mercator projection, in that rhumb lines are here shown in true scale and that unlike the Mercator, this projection does not maintain true azimuth for all points along the rhumb lines. Scale is true along the central meridian and is constant along any parallel, but not, generally, between parallels. It is free of distortion only at the central point and can be severely distorted in places. However, this projection is designed for its specific special property, in which distortion is not a concern.

Parallels For this projection, only one standard parallel is specified: the central latitude described above. Specification of this central latitude defines the center of the Loximuthal projection. The default value is 0°.

Remarks This projection was presented by Karl Siemon in 1935 and independently by Waldo R. Tobler in 1966. The Bordone Oval projection of 1520 was very similar to the Equator-centered Loximuthal.

Limitations This projection is available for the spherical geoid only.





McBryde-Thomas Flat-Polar Parabolic Projection

Classification Pseudocylindrical

Syntax `flatplrp`

Graticule Central Meridian: Straight line 0.48 as long as the Equator.

Other Meridians: Equally spaced parabolic curves concave toward the central meridian.

Parallels: Unequally spaced straight parallel lines, perpendicular to the central meridian. Spacing is greatest near the Equator.

Poles: Lines one-third as long as the Equator.

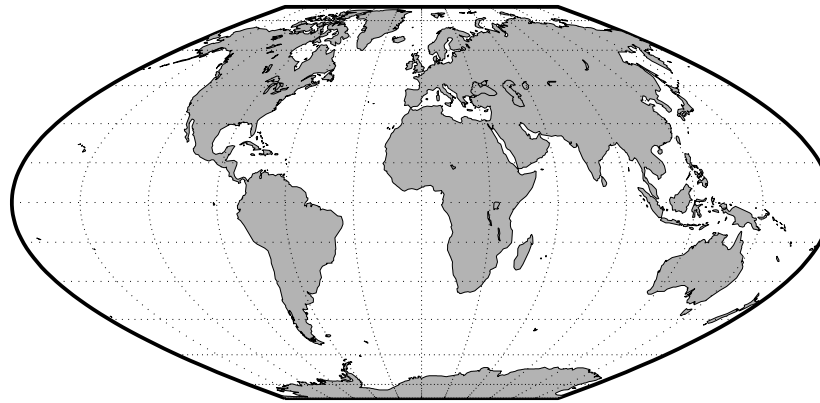
Symmetry: About the central meridian or the Equator.

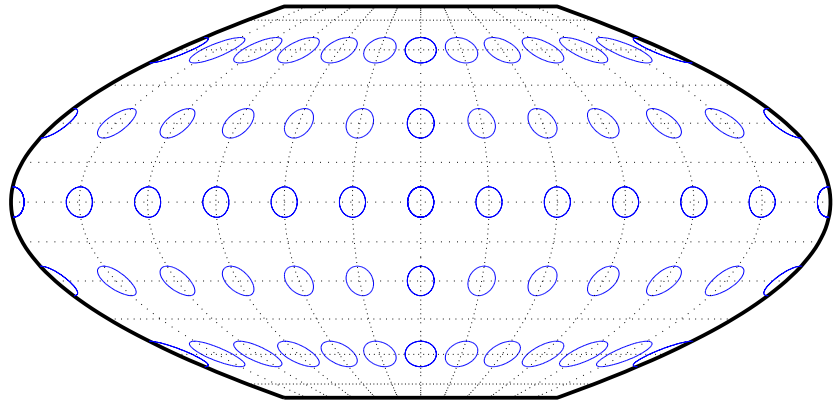
Features This is an equal-area projection. Scale is true along the 45°30' parallels and is constant along any parallel and between any pair of parallels equidistant from the Equator. Distortion is severe near the outer meridians at high latitudes, but less so than on the pointed-polar projections. It is free of distortion only at the two points where the central meridian intersects the 45°30' parallels. This projection is not conformal or equidistant.

Parallels For this projection, only one standard parallel is specified. The other standard parallel is the same latitude with the opposite sign. The standard parallel is by definition fixed at 45°30'.

Remarks This projection was presented by F. Webster McBryde and Paul D. Thomas in 1949.

McBryde-Thomas Flat-Polar Parabolic Projection





McBryde-Thomas Flat-Polar Quartic Projection

Classification Pseudocylindrical

Syntax `flatplrq`

Graticule Central Meridian: Straight line 0.45 as long as the Equator.

Other Meridians: Equally spaced quartic (fourth-order equation) curves concave toward the central meridian.

Parallels: Unequally spaced straight parallel lines, perpendicular to the central meridian. Spacing is greatest near the Equator.

Poles: Lines one-third as long as the Equator.

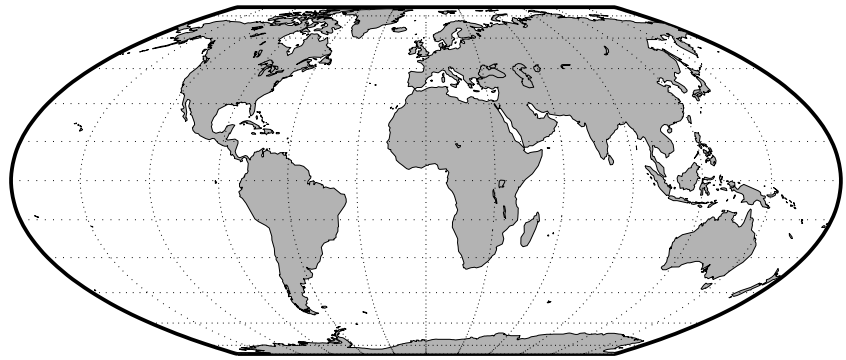
Symmetry: About the central meridian or the Equator.

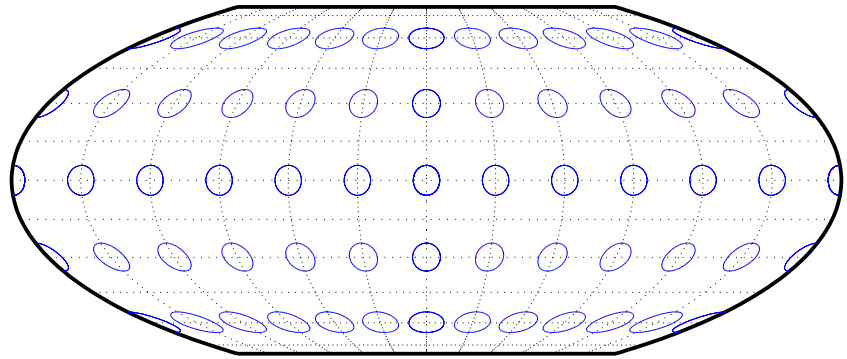
Features This is an equal-area projection. Scale is true along the 33°45' parallels and is constant along any parallel and between any pair of parallels equidistant from the Equator. Distortion is severe near the outer meridians at high latitudes, but less so than on the pointed-polar projections. It is free of distortion only at the two points where the central meridian intersects the 33°45' parallels. This projection is not conformal or equidistant.

Parallels For this projection, only one standard parallel is specified. The other standard parallel is the same latitude with the opposite sign. The standard parallel is by definition fixed at 33°45'.

Remarks This projection was presented by F. Webster McBryde and Paul D. Thomas in 1949, and is also known simply as the Flat-Polar Quartic projection.

McBryde-Thomas Flat-Polar Quartic Projection





McBryde-Thomas Flat-Polar Sinusoidal Projection

Classification Pseudocylindrical

Syntax `flatplrs`

Graticule Central Meridian: Straight line half as long as the Equator.

Other Meridians: Equally spaced sinusoidal curves intersecting at the poles and concave toward the central meridian.

Parallels: Unequally spaced straight parallel lines, perpendicular to the central meridian. Spacing is widest near the Equator.

Poles: Lines one-third as long as the Equator.

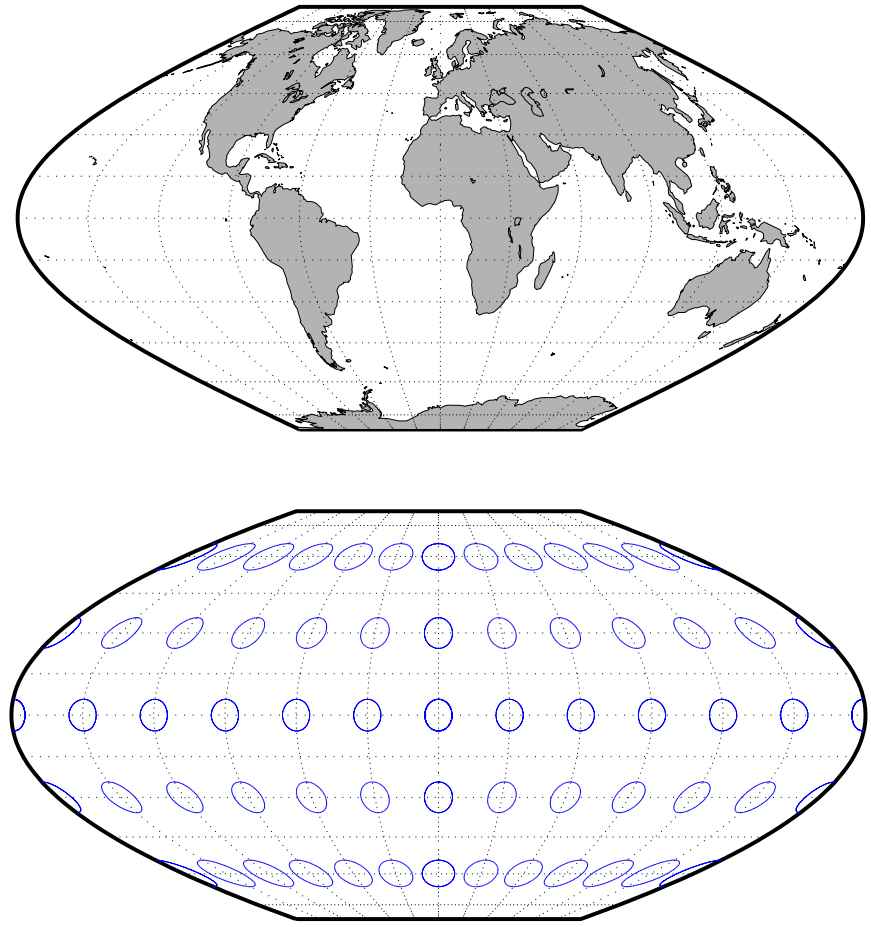
Symmetry: About the central meridian or the Equator.

Features This projection is equal-area. Scale is true along the 55°51' parallels and is constant along any parallel and between any pair of parallels equidistant from the Equator. It is free of distortion only at the two points where the central meridian intersects the 55°51' parallels. This projection is not conformal or equidistant.

Parallels For this projection, only one standard parallel is specified. The other standard parallel is the same latitude with the opposite sign. The standard parallel is by definition fixed at 55°51'.

Remarks This projection was presented by F. Webster McBryde and Paul D. Thomas in 1949.

McBryde-Thomas Flat-Polar Sinusoidal Projection



Mercator Projection

Classification Cylindrical

Syntax `mercator`

Graticule Meridians: Equally spaced straight parallel lines.

Parallels: Unequally spaced straight parallel lines, perpendicular to the meridians. Spacing increases toward the poles.

Poles: Cannot be shown.

Symmetry: About any meridian or the Equator.

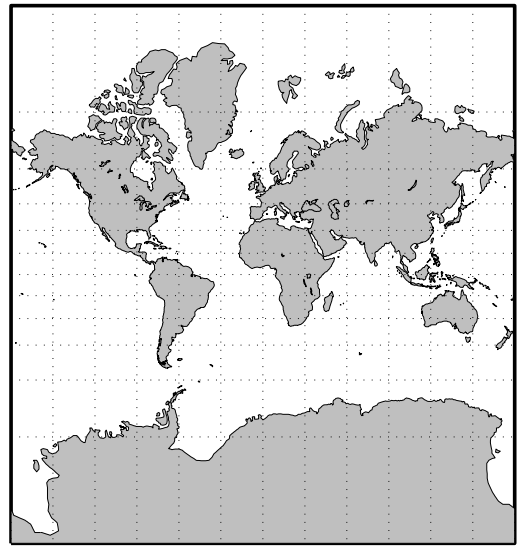
Features This is a projection with parallel spacing calculated to maintain conformality. It is not equal-area, equidistant, or perspective. Scale is true along the standard parallels and constant between two parallels equidistant from the Equator. It is also constant in all directions near any given point. Scale becomes infinite at the poles. The appearance of the Mercator projection is unaffected by the selection of standard parallels; they serve only to define the latitude of true scale.

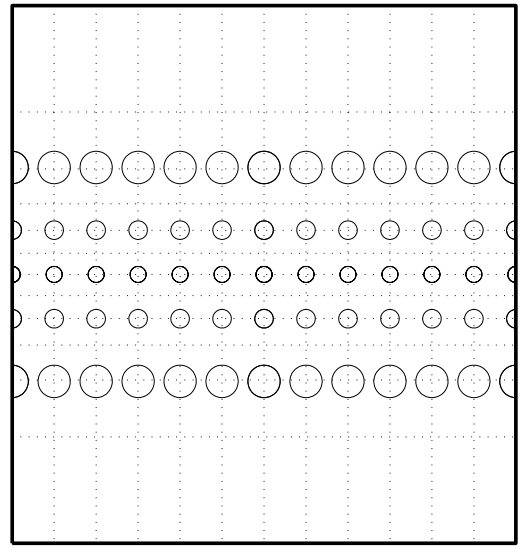
The Mercator, which may be the most famous of all projections, has the special feature that all rhumb lines, or loxodromes (lines that make equal angles with all meridians, i.e., lines of constant heading), are straight lines. This makes it an excellent projection for navigational purposes. However, the extreme area distortion makes it unsuitable for general maps of large areas.

Parallels For cylindrical projections, only one standard parallel is specified. The other standard parallel is the same latitude with the opposite sign. For this projection, any latitude less than 86° may be chosen; the default is arbitrarily set to 0°.

Remarks The Mercator projection is named for Gerardus Mercator, who presented it *for navigation* in 1569. It is now known to have been used for the Tunhuang star chart as early as 940 by Ch'ien Lo-Chih. It was first used in Europe by Erhard Etzlaub in 1511. It is also, but rarely, called the Wright projection, after Edward Wright, who developed the mathematics behind the projection in 1599.

Limitations Data at latitudes greater than 86° is trimmed to prevent large *y*-values from dominating the display.





Miller Cylindrical Projection

Classification Cylindrical

Syntax `miller`

Graticule Meridians: Equally spaced straight parallel lines 0.73 as long as the Equator.

Parallels: Unequally spaced straight parallel lines, perpendicular to the meridians. Spacing increases toward the poles, less rapidly than that of the Mercator projection.

Poles: Straight lines equal in length to the Equator.

Symmetry: About any meridian or the Equator.

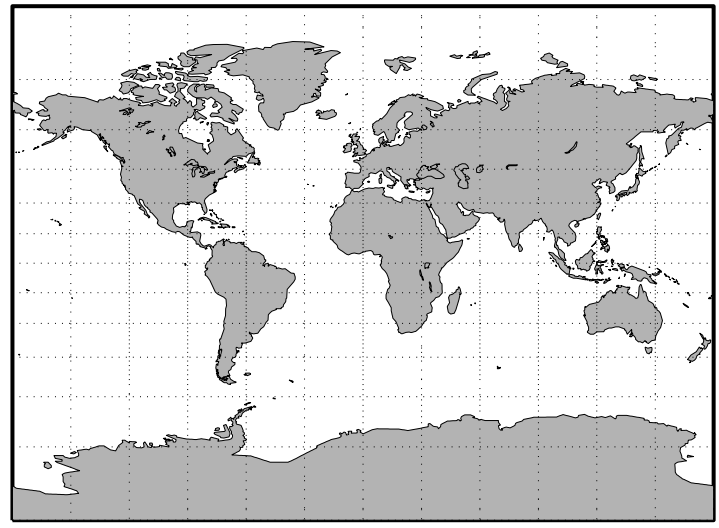
Features This is a projection with parallel spacing calculated to maintain a look similar to the Mercator projection while reducing the distortion near the poles and allowing the poles to be displayed. It is not equal-area, equidistant, conformal, or perspective. Scale is true along the Equator and constant between two parallels equidistant from the Equator. There is no distortion near the Equator, and it increases moderately away from the Equator, but it becomes severe at the poles.

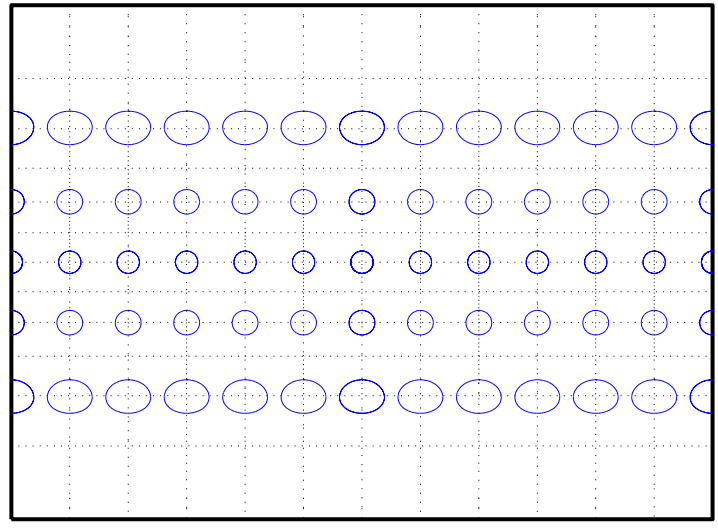
The Miller Cylindrical projection is derived from the Mercator projection; parallels are spaced from the Equator by calculating the distance on the Mercator for a parallel at 80% of the true latitude and dividing the result by 0.8. The result is that the two projections are almost identical near the Equator.

Parallels For cylindrical projections, only one standard parallel is specified. The other standard parallel is the same latitude with the opposite sign. For this projection, the standard parallel is by definition fixed at 0°.

Remarks This projection was presented by Osborn Maitland Miller of the American Geographical Society in 1942. It is often used in place of the Mercator projection for atlas maps of the world, for which it is somewhat more appropriate.

Limitations This projection is available for the spherical geoid only.





Mollweide Projection

Classification Pseudocylindrical

Syntax `mollweid`

Graticule Central Meridian: Straight line half as long as the Equator.

Other Meridians: Meridians 90° east and west of the central meridian form a circle. The others are equally spaced semiellipses intersecting at the poles and concave toward the central meridian.

Parallels: Unequally spaced straight parallel lines, perpendicular to the central meridian. Spacing is greatest toward the Equator, but the spacing changes gradually.

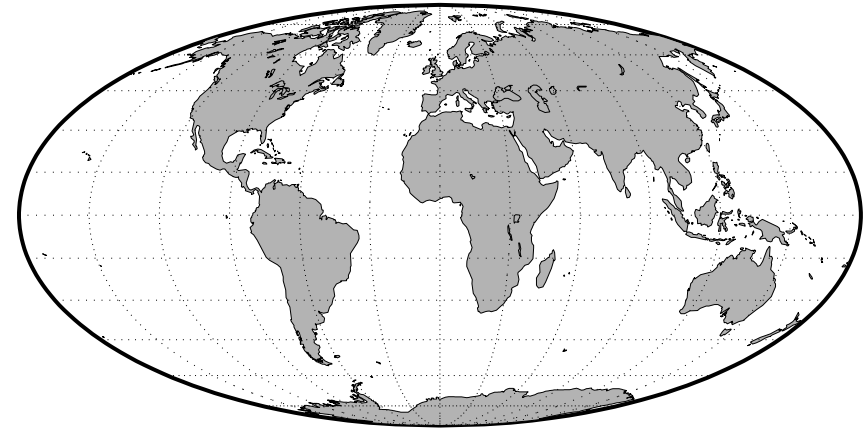
Poles: Points.

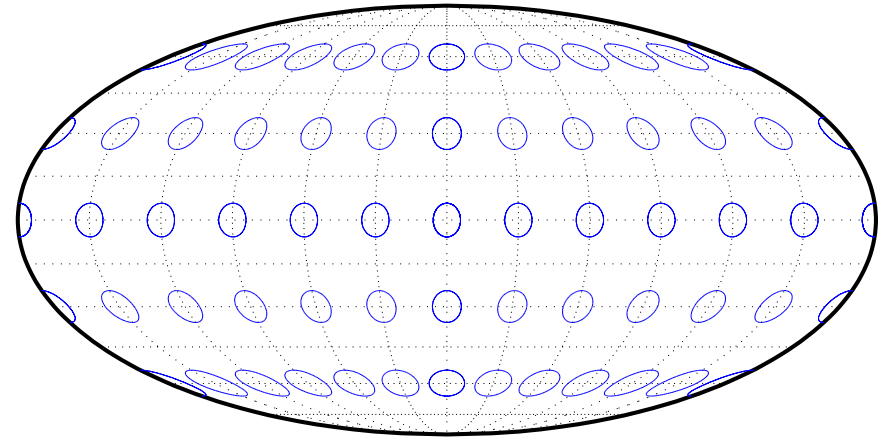
Symmetry: About the central meridian or the Equator.

Features This is an equal-area projection. Scale is true along the 40°44' parallels and is constant along any parallel and between any pair of parallels equidistant from the Equator. It is free of distortion only at the two points where the 40°44' parallels intersect the central meridian. This projection is not conformal or equidistant.

Parallels For this projection, only one standard parallel is specified. The other standard parallel is the same latitude with the opposite sign. The standard parallel is by definition fixed at 40°44'.

Remarks This projection was presented by Carl B. Mollweide in 1805. Its other names include the Homolographic, the Homalographic, the Babinet, and the Elliptical projections. It is occasionally used for thematic world maps, and it is combined with the Sinusoidal to produce the Goode Homolosine projection.





Murdoch I Conic Projection

Classification Conic

Syntax `murdoch1`

Graticule Meridians: Equally spaced straight lines converging at one of the poles.

Parallels: Equally spaced concentric circular arcs.

Poles: Arcs, one of which might become a point in the limit.

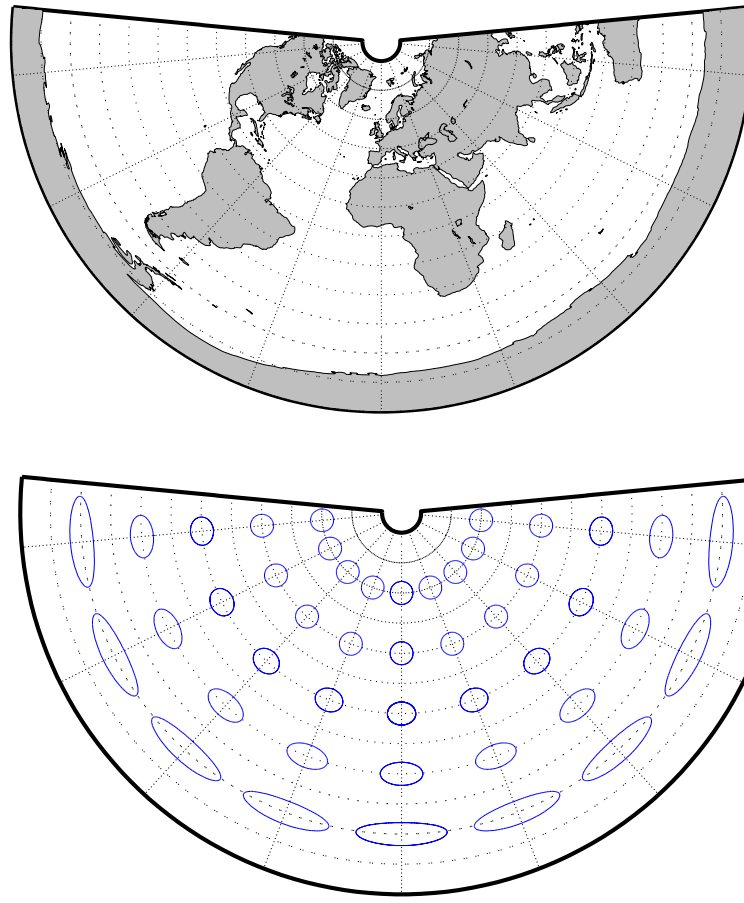
Symmetry: About any meridian.

Features This is an equidistant projection that is nearly minimum-error. Scale is true along any meridian and is constant along any parallel. Scale is also true along two standard parallels. These must be calculated, however (see below). The total area of the mapped area is correct, but it is not equal-area everywhere.

Parallels The parallels for this projection are not standard parallels, but rather limiting parallels. The special feature of this map, correct total area, holds between these parallels. The default parallels are [15 75].

Remarks Described by Patrick Murdoch in 1758.

Limitations This projection is available for the spherical geoid only. Longitude data greater than 135° east or west of the central meridian is trimmed.



Murdoch III Minimum Error Conic Projection

Classification Conic

Syntax `murdoch3`

Graticule Meridians: Equally spaced straight lines converging at one of the poles.

Parallels: Equally spaced concentric circular arcs.

Poles: Arcs, one of which might become a point in the limit.

Symmetry: About any meridian.

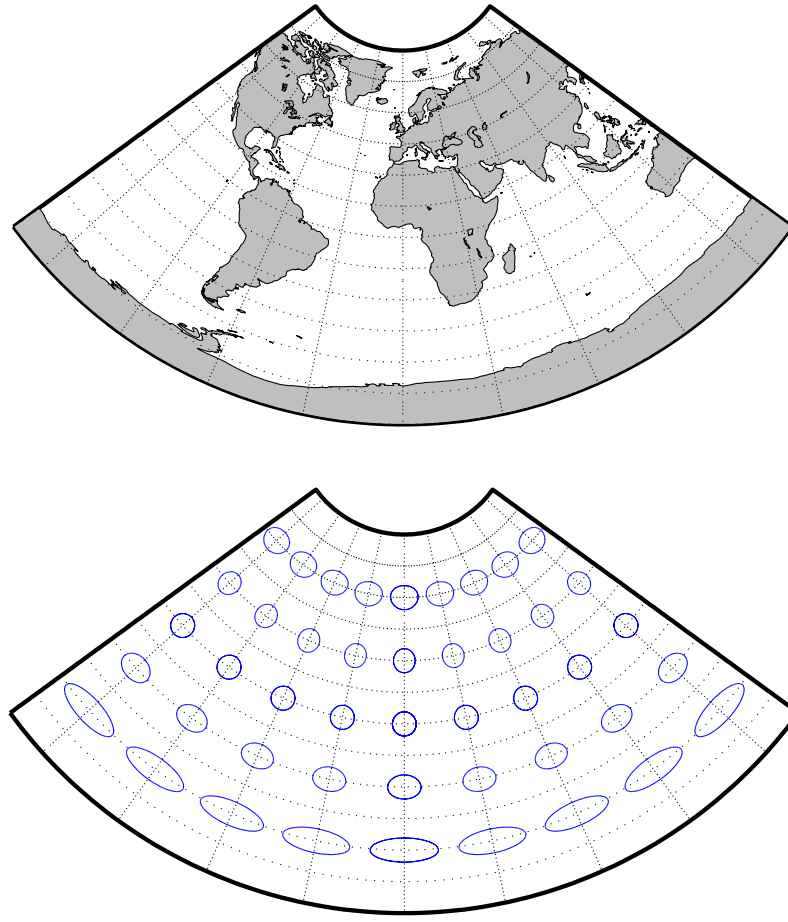
Features This is an equidistant projection that is minimum-error. Scale is true along any meridian and is constant along any parallel. Scale is also true along two standard parallels. These must be calculated, however (see below). The total area of the mapped area is correct, but it is not equal-area everywhere.

Parallels The parallels for this projection are not standard parallels, but rather limiting parallels. The special feature of this map, correct total area, holds between these parallels. The default parallels are [15 75].

Remarks Described by Patrick Murdoch in 1758, with errors corrected by Everett in 1904.

Limitations This projection is available for the spherical geoid only. Longitude data greater than 135° east or west of the central meridian is trimmed.

Murdoch III Minimum Error Conic Projection



Orthographic Projection

Classification Azimuthal

Syntax `ortho`

Graticule The graticule described is for a polar aspect.

Meridians: Equally spaced straight lines intersecting at the central pole. The angles displayed are the true angles between meridians.

Parallels: Unequally spaced circles centered on the central pole. Spacing decreases away from this pole. The opposite hemisphere cannot be shown.

Pole: The central pole is a point; the other pole is not shown.

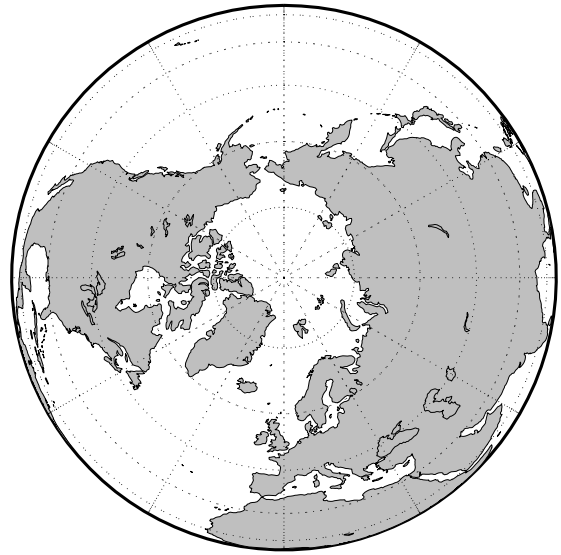
Symmetry: About any meridian.

Features This is a perspective projection on a plane tangent at the center point from an infinite distance (that is, orthogonally). The center point is a pole in the common polar aspect, but can be any point. This projection has two significant properties. It looks like a globe, providing views of the Earth resembling those seen from outer space. Additionally, all great and small circles are either straight lines or elliptical arcs on this projection. Scale is true only at the center point and is constant in the circumferential direction along any circle having the center point as its center. Distortion increases rapidly away from the center point, the only place that is distortion-free. This projection is neither conformal nor equal-area.

Parallels There are no standard parallels for azimuthal projections.

Remarks This projection appears to have been developed by the Egyptians and Greeks by the second century B.C.

Limitations This projection is available for the spherical geoid only. Data greater than 89° distant from the center point is trimmed.



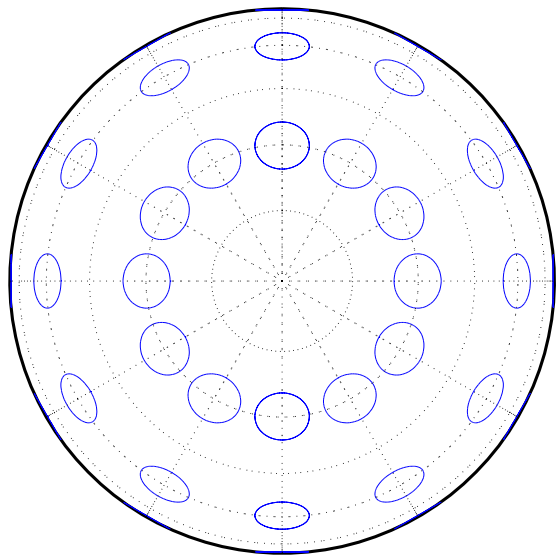


Plate Carrée Projection

Classification Cylindrical

Syntax `pcarree`

Graticule Meridians: Equally spaced straight parallel lines half as long as the Equator.

Parallels: Equally spaced straight parallel lines, perpendicular to and having the same spacing as the meridians.

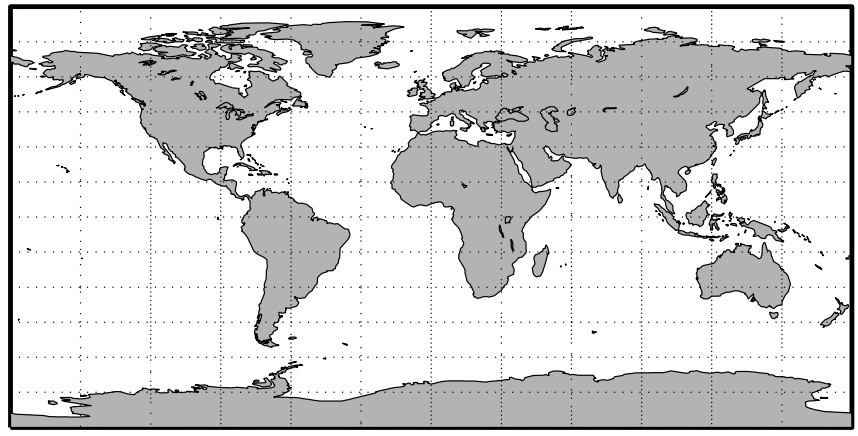
Poles: Straight lines equal in length to the Equator.

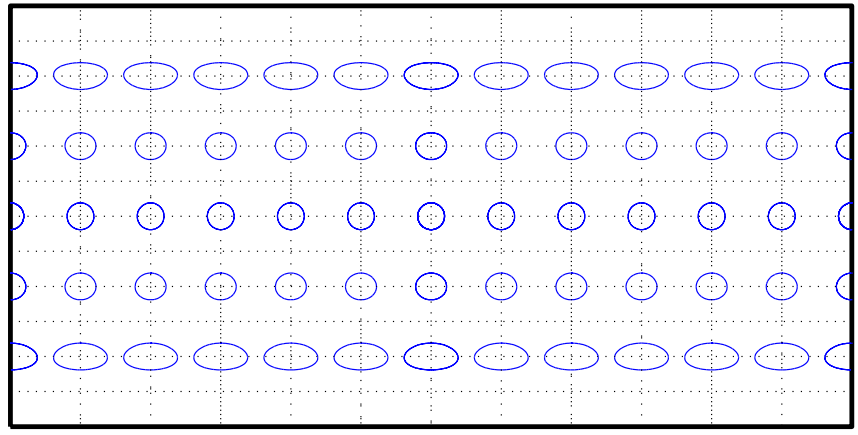
Symmetry: About any meridian or the Equator.

Features This is a projection onto a cylinder tangent at the Equator. Distortion of both shape and area increases with distance from the Equator. Scale is true along all meridians (i.e., it is equidistant) and the Equator and is constant along any parallel and along the parallel of opposite sign.

Parallels For cylindrical projections, only one standard parallel is specified. The other standard parallel is the same latitude with the opposite sign. For this projection, the standard parallel is by definition fixed at 0°.

Remarks This projection, like the more general Equidistant Cylindrical, is credited to Marinus of Tyre, thought to have invented it about A.D. 100. It may, in fact, have been originated by Erastosthenes, who lived approximately 275-195 B.C. The Plate Carrée has the most simply constructed graticule of any projection. It was used frequently in the 15th and 16th centuries and is quite common today in very simple computer mapping programs. It is the simplest and limiting form of the Equidistant Cylindrical projection. Another name for this projection is the Simple Cylindrical. Its transverse aspect is the Cassini projection.





Polyconic Projection

Classification Polyconic

Syntax `polycon`

Graticule Central Meridian: A straight line.

Meridians: Complex curves spaced equally along the Equator and each parallel, and concave toward the central meridian.

Parallels: The Equator is a straight line. All other parallels are nonconcentric circular arcs spaced at true distances along the central meridian.

Poles: Normally circular arcs, enclosing the same angle as the displayed parallels.

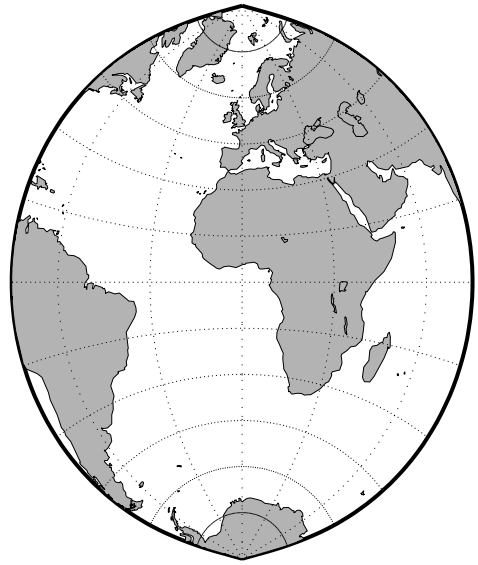
Symmetry: About the Equator or the central meridian.

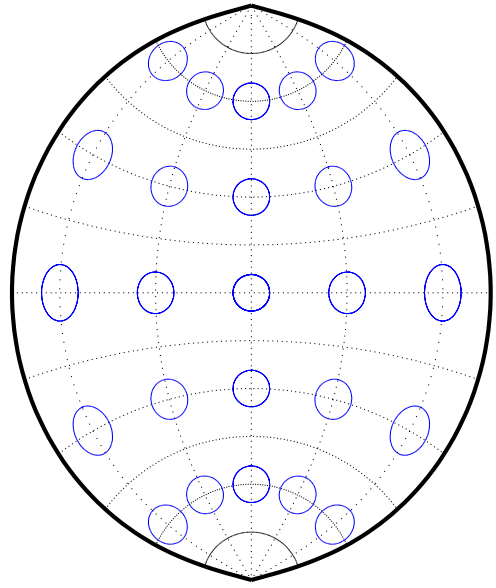
Features For this projection, each parallel has a curvature identical to its curvature on a cone tangent at that latitude. Since each parallel has its own cone, this is a "polyconic" projection. Scale is true along the central meridian and along each parallel. This projection is free of distortion only along the central meridian; distortion can be severe at extreme longitudes. This projection is neither conformal nor equal-area.

Parallels By definition, this projection has no standard parallels, since every parallel is a *standard parallel*.

Remarks This projection was apparently originated about 1820 by Ferdinand Rudolph Hassler. It is also known as the American Polyconic and the Ordinary Polyconic projection.

Limitations Longitude data greater than 75° east or west of the central meridian is trimmed.





Putnins P5 Projection

Classification Pseudocylindrical

Syntax `putnins5`

Graticule Central Meridian: Straight line half as long as the Equator.

Other Meridians: Equally spaced portions of hyperbolas intersecting at the poles and concave toward the central meridian.

Parallels: Equally spaced straight parallel lines, perpendicular to the central meridian.

Poles: Points.

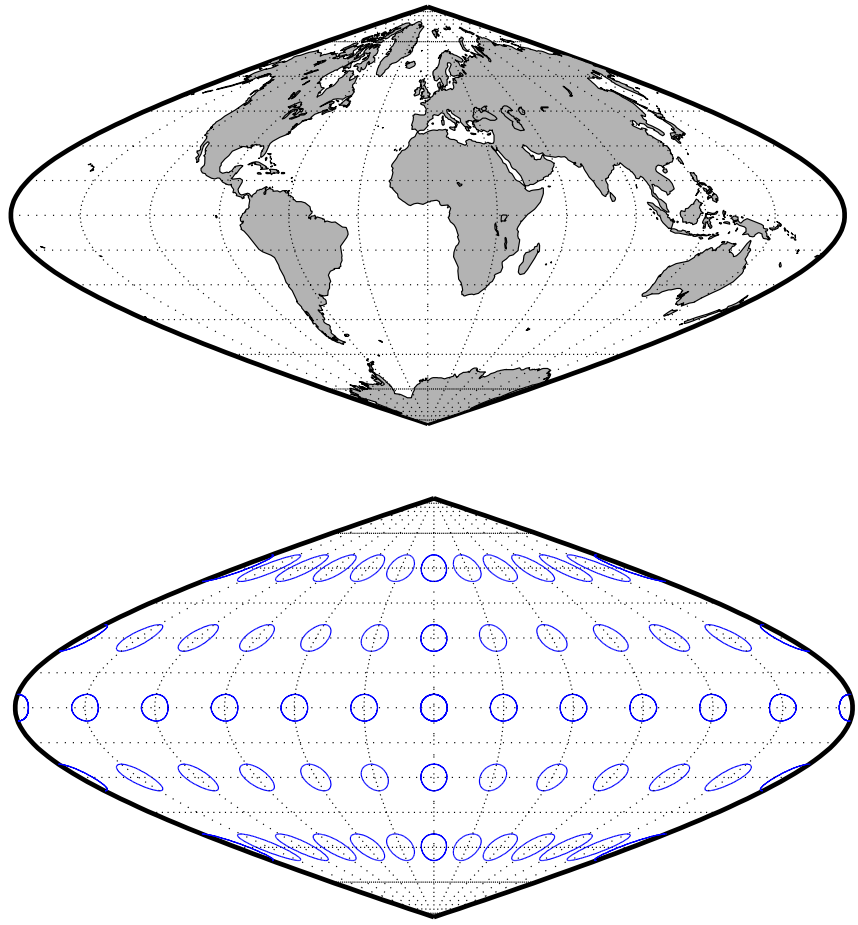
Symmetry: About the central meridian or the Equator.

Features Scale is true along the 21°14' parallels and is constant along any parallel, between any pair of parallels equidistant from the Equator, and along the central meridian. It is not free of distortion at any point. This projection is not equal-area, conformal, or equidistant.

Parallels For this projection, only one standard parallel is specified. The other standard parallel is the same latitude with the opposite sign. The standard parallel is by definition fixed at 21°14'.

Remarks This projection was presented by Reinholds V. Putnins in 1934.

Limitations This projection is available for the spherical geoid only.



Quartic Authalic Projection

Classification Pseudocylindrical

Syntax `quartic`

Graticule Central Meridian: Straight line 0.45 as long as the Equator.

Other Meridians: Equally spaced quartic (fourth-order equation) curves concave toward the central meridian.

Parallels: Unequally spaced straight parallel lines, perpendicular to the central meridian. Spacing changes gradually and is greatest near the Equator.

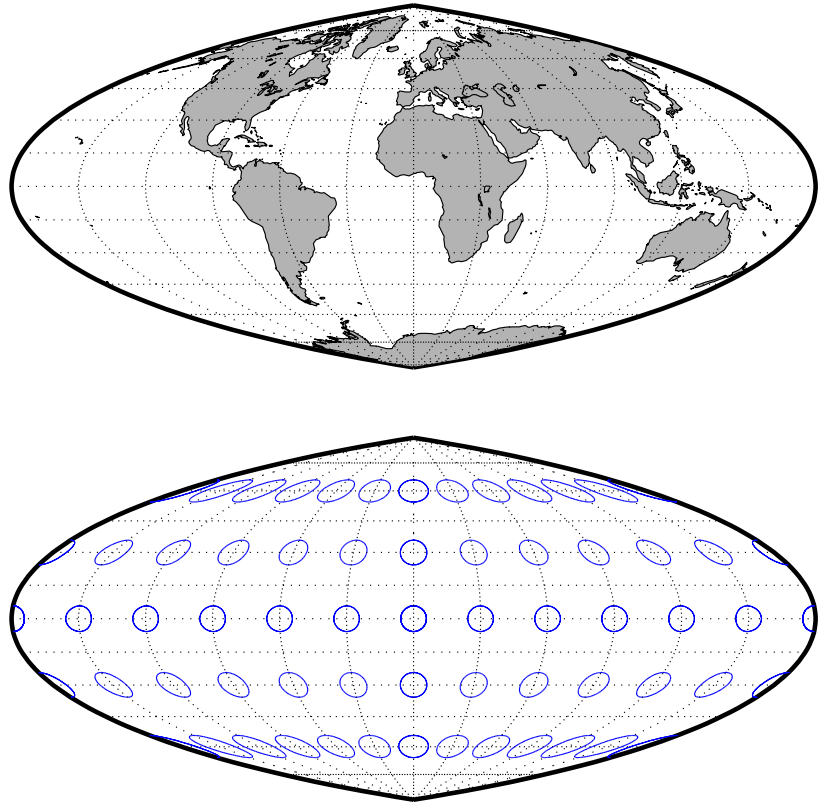
Poles: Points.

Symmetry: About the central meridian or the Equator.

Features This is an equal-area projection. Scale is true along the Equator and is constant along any parallel and between any pair of parallels equidistant from the Equator. Distortion is severe near the outer meridians at high latitudes, but less so than on the Sinusoidal projection. It is free of distortion along the Equator. This projection is not conformal or equidistant.

Parallels This projection has one standard parallel, which is by definition fixed at 0°.

Remarks This projection was presented by Karl Siemon in 1937 and independently by Oscar Sherman Adams in 1945.



Robinson Projection

Classification Pseudocylindrical

Syntax `robinson`

Graticule Central Meridian: Straight line 0.51 as long as the Equator.

Other Meridians: Equally spaced, resemble elliptical arcs and are concave toward the central meridian.

Parallels: Straight parallel lines, perpendicular to the central meridian. Spacing is equal between the 38° parallels, decreasing outside these limits.

Poles: Lines 0.53 as long as the Equator.

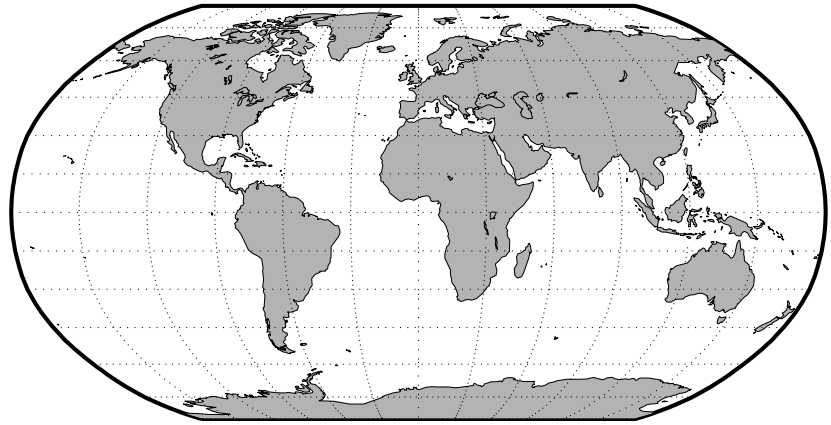
Symmetry: About the central meridian or the Equator.

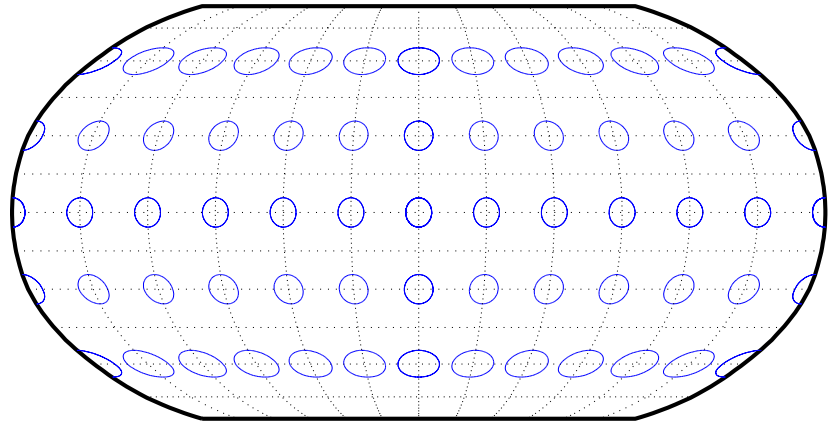
Features Scale is true along the 38° parallels and is constant along any parallel or between any pair of parallels equidistant from the Equator. It is not free of distortion at any point, but distortion is very low within about 45° of the center and along the Equator. This projection is not equal-area, conformal, or equidistant; however, it is considered to *look right* for world maps, and hence is widely used by Rand McNally, the National Geographic Society, and others. This feature is achieved through the use of tabular coordinates rather than mathematical formulae for the graticules.

Parallels For this projection, only one standard parallel is specified. The other standard parallel is the same latitude with the opposite sign. The standard parallel is by definition fixed at 38°.

Remarks This projection was presented by Arthur H. Robinson in 1963, and is also called the Orthophanic projection, which means *right appearing.*

Limitations This projection is available for the spherical geoid only.





Sinusoidal Projection

Classification Pseudocylindrical

Syntax `sinusoid`

Graticule Central Meridian: Straight line half as long as the Equator.

Other Meridians: Equally spaced sinusoidal curves intersecting at the poles and concave toward the central meridian.

Parallels: Equally spaced straight parallel lines, perpendicular to the central meridian.

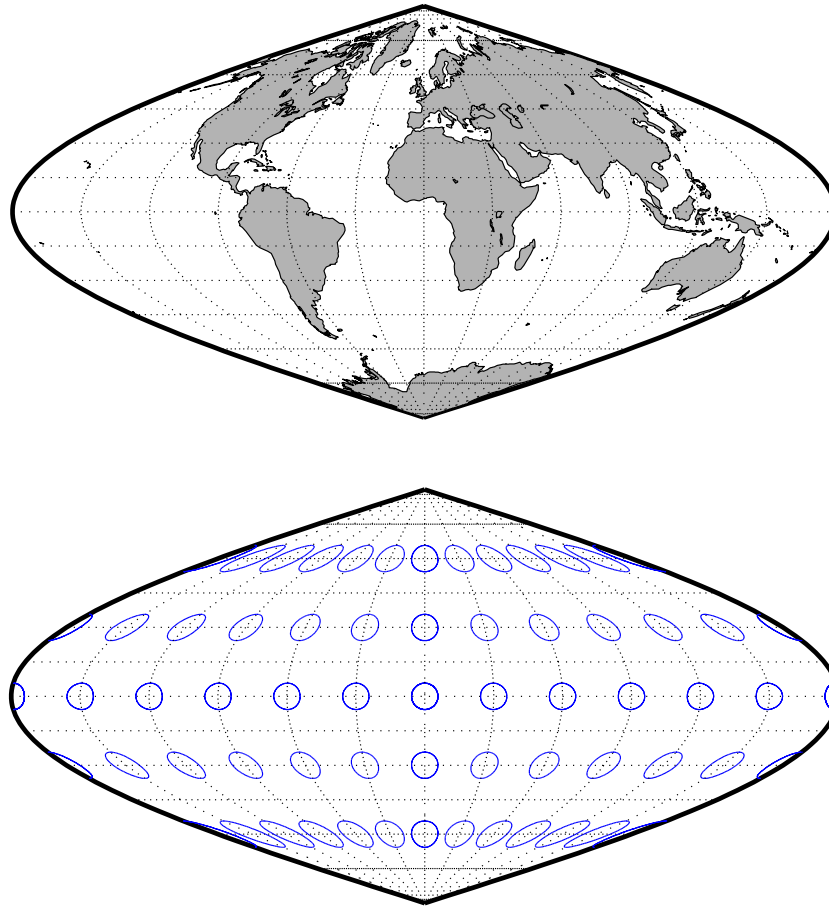
Poles: Points.

Symmetry: About the central meridian or the Equator.

Features This projection is equal-area. Scale is true along every parallel and along the central meridian. There is no distortion along the Equator or along the central meridian, but it becomes severe near the outer meridians at high latitudes.

Parallels This projection has one standard parallel, which is by definition fixed at 0°.

Remarks This projection was developed in the 16th century. It was used by Jean Cossin in 1570 and by Jodocus Hondius in Mercator atlases of the early 17th century. It is the oldest pseudocylindrical projection currently in use, and is sometimes called the Sanson-Flamsteed or the Mercator Equal-Area projection.



Stereographic Projection

Classification Azimuthal

Syntax `stereo`

Graticule The graticule described is for a polar aspect.

Meridians: Equally spaced straight lines intersecting at the central pole. The angles displayed are the true angles between meridians.

Parallels: Unequally spaced circles centered on the central pole. Spacing increases gradually away from this pole. The opposite hemisphere cannot be shown

Pole: The central pole is a point; the other pole is not shown.

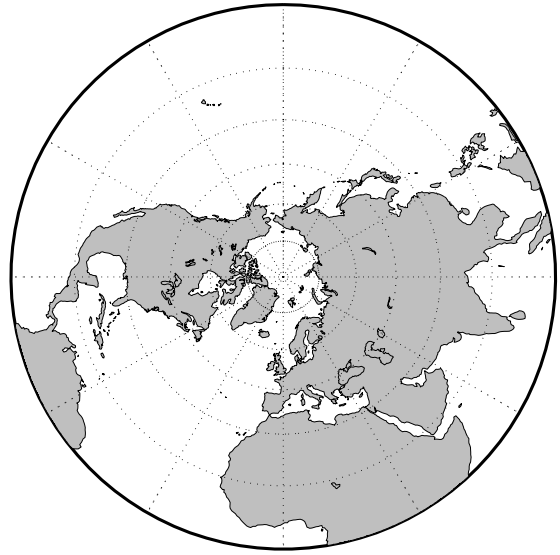
Symmetry: About any meridian.

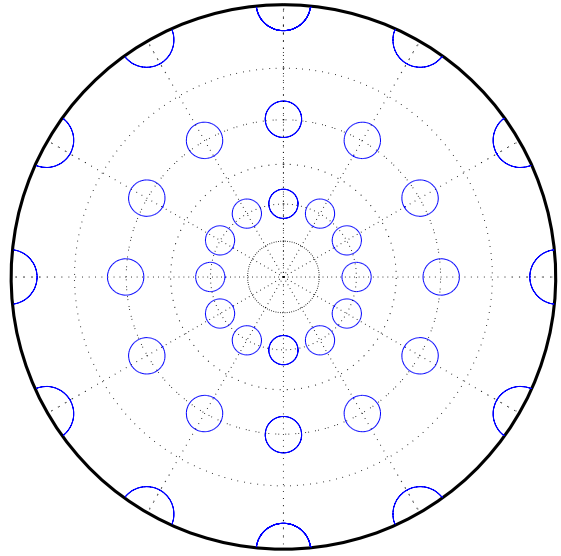
Features This is a perspective projection on a plane tangent at the center point from the point antipodal to the center point. The center point is a pole in the common polar aspect, but can be any point. This projection has two significant properties. It is conformal, being free from angular distortion. Additionally, all great and small circles are either straight lines or circular arcs on this projection. Scale is true only at the center point and is constant along any circle having the center point as its center. This projection is not equal-area.

Parallels There are no standard parallels for azimuthal projections.

Remarks The polar aspect of this projection appears to have been developed by the Egyptians and Greeks by the second century B.C.

Limitations Data greater than 90° distant from the center point is trimmed.





Tissot Modified Sinusoidal Projection

Classification Pseudocylindrical

Syntax `modsine`

Graticule Meridians: Sine curves converging at the Poles.

Parallels: Equally spaced straight lines.

Poles: Points.

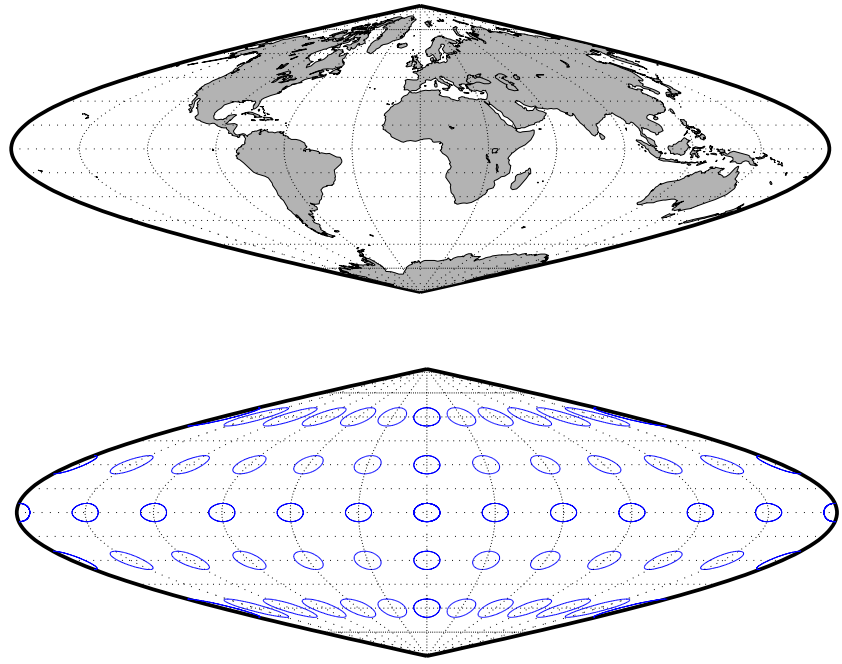
Symmetry: About the Equator and the central meridian

Features This is an equal-area projection. Scale is constant along any parallel or any pair of equidistant parallels, and along the central meridian. It is not equidistant or conformal.

Parallels There are no standard parallels for this projection

Remarks This projection was first described by N. A. Tissot in 1881

Limitations This projection is available for the spherical geoid only.



Trystan Edwards Cylindrical Projection

Classification Cylindrical

Syntax `trystan`

Graticule Meridians: Equally spaced straight parallel lines.

Parallels: Unequally spaced straight parallel lines, perpendicular to the meridians. Spacing is closest near the poles.

Poles: Straight lines equal in length to the Equator.

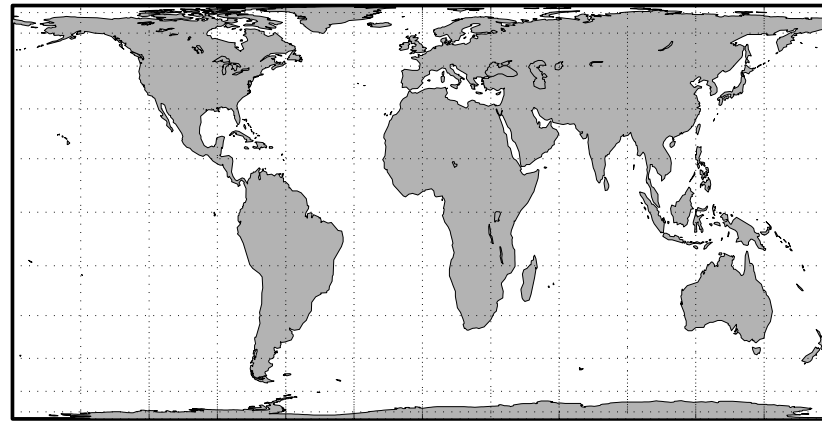
Symmetry: About any meridian or the Equator.

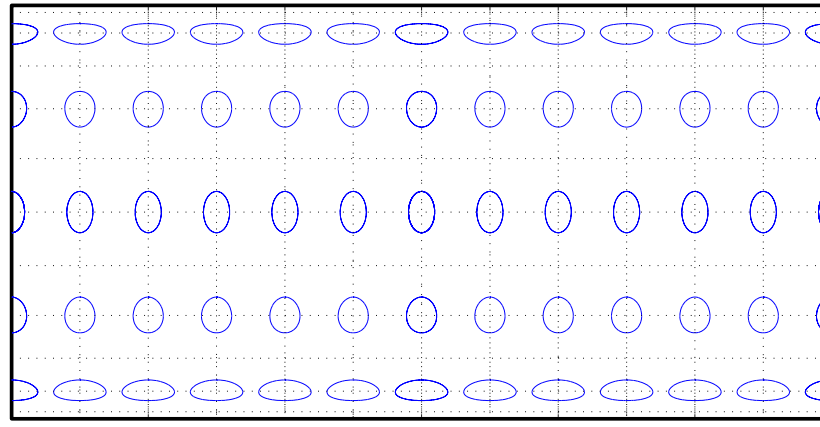
Features This is an orthographic projection onto a cylinder secant at the 37°24' parallels. It is equal-area, but distortion of shape increases with distance from the standard parallels. Scale is true along the standard parallels and constant between two parallels equidistant from the Equator. This projection is not equidistant.

Parallels For cylindrical projections, only one standard parallel is specified. The other standard parallel is the same latitude with the opposite sign. For this projection, the standard parallel is by definition fixed at 37°24'.

Remarks This projection is named for Trystan Edwards, who presented it in 1953. It is a special form of the Equal-Area Cylindrical projection secant at 37°24'N and S.

Trystan Edwards Cylindrical Projection





Universal Polar Stereographic Projection

Classification Azimuthal

Syntax ups

Graticule The graticule described is for the southern zone.

Meridians: Equally spaced straight lines centered on the South Pole. The angles displayed are the true angles between meridians.

Parallels: Unequally spaced circles centered on the South Pole. Spacing increases gradually away from the circle of true scale along latitude 87 degrees, 7 minutes N. The opposite pole cannot be shown.

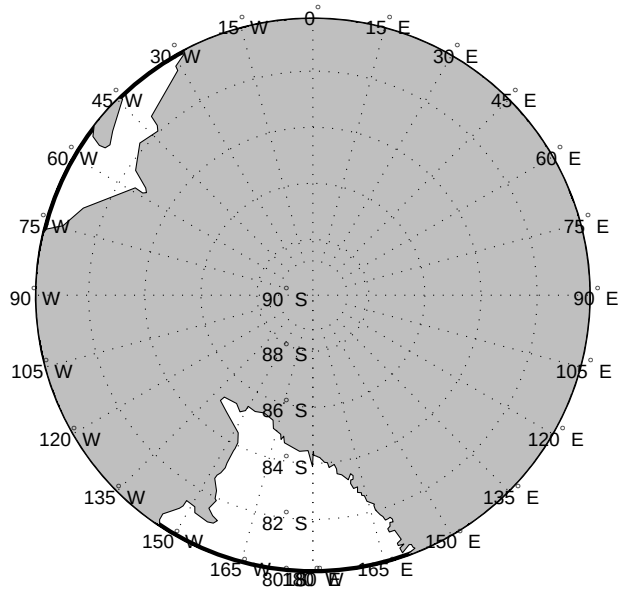
Poles: The South Pole is a point. The North Pole is not shown.

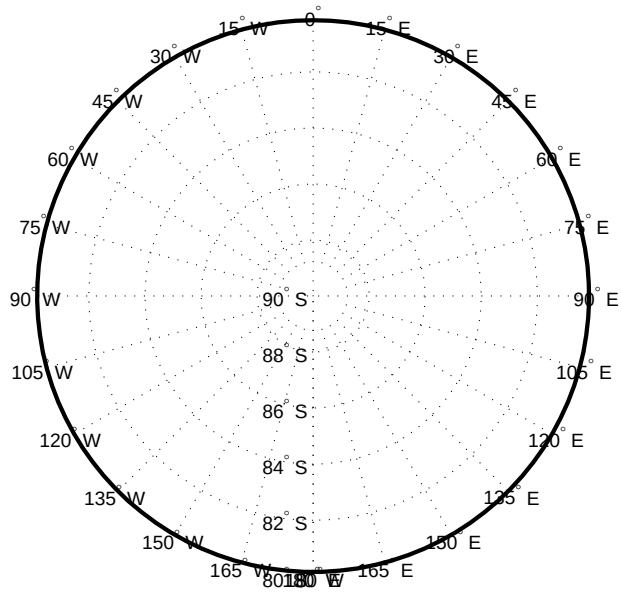
Symmetry: About any meridian.

Features This is a perspective projection on a plane tangent to either the North or South Pole. It is conformal, being free from angular distortion. Additionally, all great and small circles are either straight lines or circular arcs on this projection. Scale is true along latitudes 87 degrees, 7 minutes N or S, and is constant along any other parallel. This projection is not equal area.

Parallels The parallels 87 degrees, 7 minutes N and S are lines of true scale by virtue of the scale factor. There are no standard parallels for azimuthal projections.

Remarks This projection is a special case of the stereographic projection in the polar aspect. It is used as part of the Universal Transverse Mercator (UTM) system to extend coverage to the poles. This projection has two zones: 'North' for latitudes 84° N to 90° N, and 'South' for latitudes 80° S to 90° S. The defaults for this projection are: scale factor is 0.994, false easting and northing are 2,000,000 meters. The international ellipsoid in units of meters is used as the geoid model.


0°
15° W
15° E
30° W
30° E
45° W
45° E
60° W
60° E
75° W
75° E
90° W
90° E
105° W
105° E
120° W
120° E
135° W
135° E
150° W
150° E
165° W
165° E
90° S
88° S
86° S
84° S
82° S


0°
15° W
15° E
30° W
30° E
45° W
45° E
60° W
60° E
75° W
75° E
90° W
90° E
105° W
105° E
120° W
120° E
135° W
135° E
150° W
150° E
165° W
165° E
90° S
88° S
86° S
84° S
82° S

Universal Transverse Mercator Projection

Classification Cylindrical

Syntax `utm`

Graticule Meridians: Complex curves concave towards the central meridian.

Parallels: Complex curves concave towards the nearest pole.

Poles: Not shown.

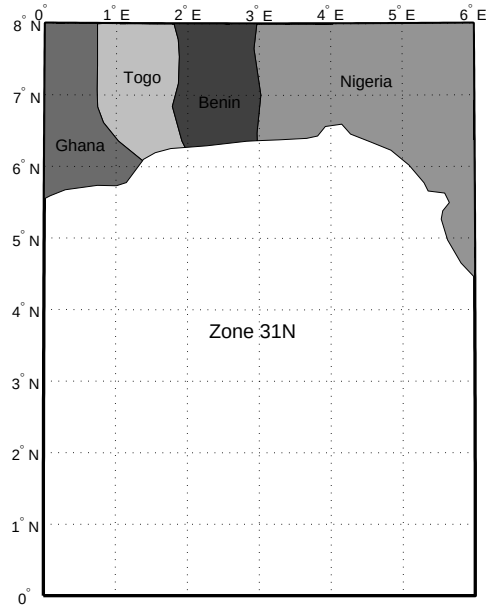
Symmetry: About the central meridian or the Equator.

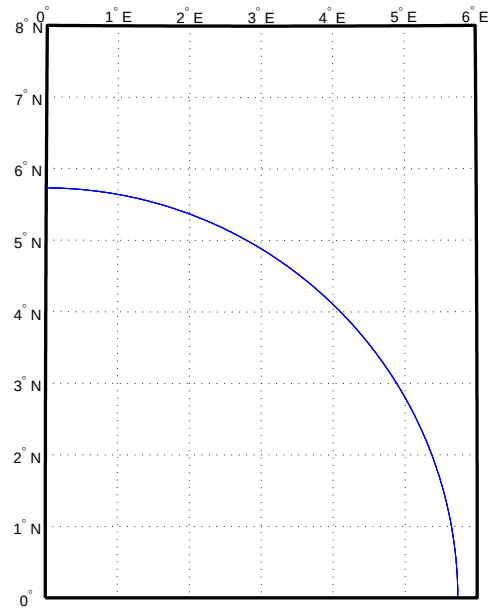
Features This is a conformal projection with parameters chosen to minimize distortion over a defined set of small areas. It is not equal area, equidistant, or perspective. Scale is true along two straight lines on the map approximately 180 kilometers east and west of the central meridian, and is constant along other straight lines equidistant from the central meridian. Scale is less than true between, and greater than true outside the lines of true scale.

Parallels There are no standard parallels for this projection. There are two lines of zero distortion by virtue of the scale factor.

Remarks The UTM system divides the world between 80° S and 84° degrees N into a set of quadrangles called zones. Zones generally cover 6 degrees of longitude and 8 degrees of latitude. Each zone has a set of defined projection parameters, including central meridian, false eastings and northings and the reference ellipsoid. The projection equations are the Gauss-Krüger versions of the Transverse Mercator. The projected coordinates form a grid system, in which a location is specified by the zone, easting and northing.

The UTM system was introduced in the 1940's by the U.S. Army. It is widely used in topographic and military mapping.


0°
1° E
2° E
3° E
4° E
5° E
6° E
8° N
7° N
6° N
5° N
4° N
3° N
2° N
1° N
0°
Togo
Benin
Nigeria
Ghana
Zone 31N


0°
1° E
2° E
3° E
4° E
5° E
6° E
8° N
7° N
6° N
5° N
4° N
3° N
2° N
1° N
0°

Van der Grinten I Projection

Classification Polyconic

Syntax `vgrint1`

Graticule Central Meridian: A straight line.

Meridians: Circular curves spaced equally along the equator and concave toward the central meridian.

Parallels: The Equator is a straight line. All other parallels are circular arcs concave toward the nearest pole.

Poles: Points.

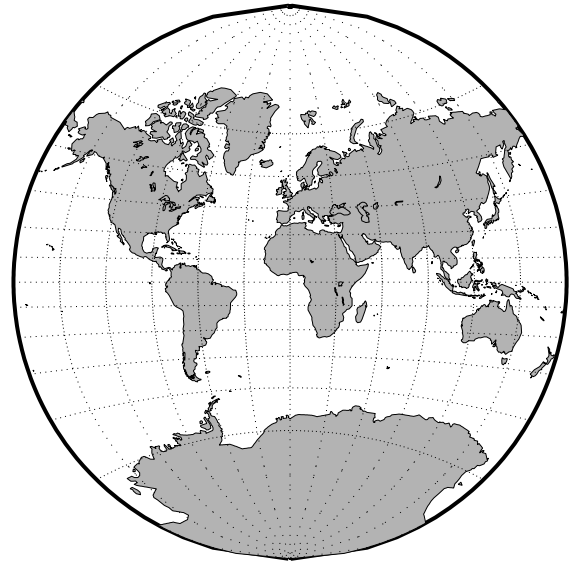
Symmetry: About the Equator or the central meridian.

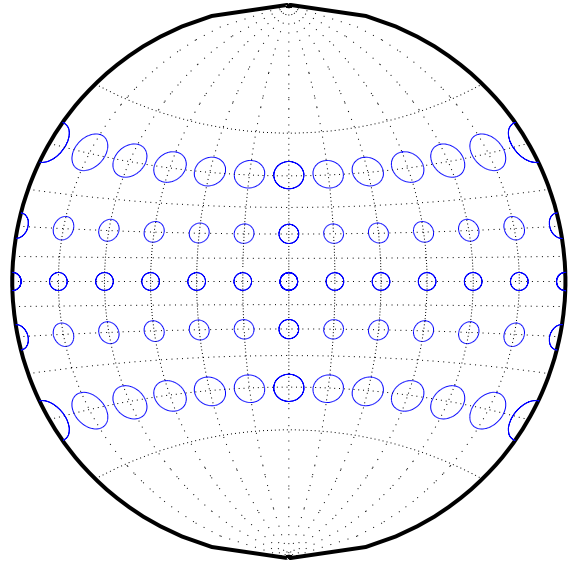
Features In this projection, the world is enclosed in a circle. Scale is true along the Equator and increases rapidly away from the Equator. Area distortion is extreme near the poles. This projection is neither conformal nor equal-area.

Parallels There are no standard parallels for this projection.

Remarks This projection was presented by Alphons J. Van der Grinten in 1898. He obtained a U.S. patent for it in 1904. It is also known simply as the Van der Grinten projection (without the “I”).

Limitations This projection is available for the spherical geoid only.





Vertical Perspective Azimuthal Projection

Classification Azimuthal

Syntax `vperspec`

Graticule The graticule described is for a polar aspect.

Meridians: Equally spaced straight lines intersecting at the central pole. The angles displayed are true angles between meridians.

Parallels: Unequally spaced circles centered on the central pole. Spacing decreases away from this pole. The opposite hemisphere cannot be shown, nor can distant parts of the closer hemisphere. The limit of visibility depends on the observation altitude.

Poles: The central pole is a point. The other pole is not shown.

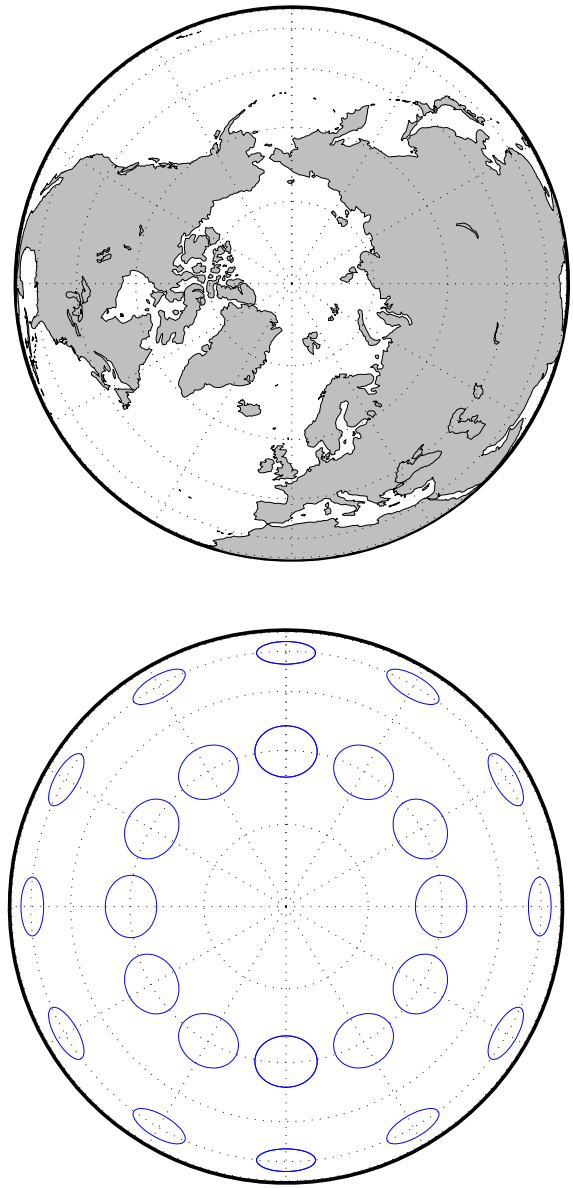
Symmetry: About any meridian.

Features This is a perspective projection on a plane tangent at the center point from a finite distance. Scale is true only at the center point, and is constant in the circumferential direction along any circle having the center point as its center. Distortion increases rapidly away from the center point, the only point which is distortion free. This projection is neither conformal nor equal area.

Parallels The standard parallel contains the observation altitude above the surface in the same units as the geoid semi-major axis.

Remarks This projection provides views of the globe resembling those seen from a spacecraft in orbit. The Orthographic projection is a limiting form with the observer at an infinite distance.

Limitations This projection is available for the spherical geoid only. Data more distant than the limit of visibility is trimmed.



Wagner IV Projection

Classification Pseudocylindrical

Syntax `wagner4`

Graticule Central Meridian: Straight line half as long as the Equator.

Other Meridians: Equally spaced portions of ellipses concave toward the central meridian. The meridians 103°55' east and west of the central meridian are circular arcs.

Parallels: Unequally spaced straight parallel lines, perpendicular to the central meridian. Spacing is greatest toward the Equator.

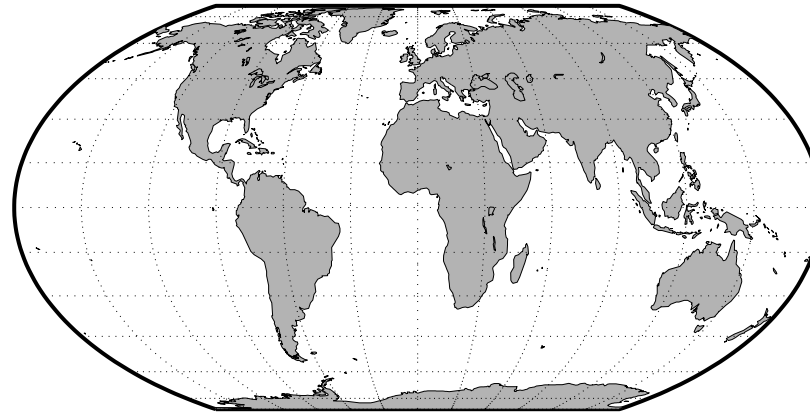
Poles: Lines half as long as the Equator.

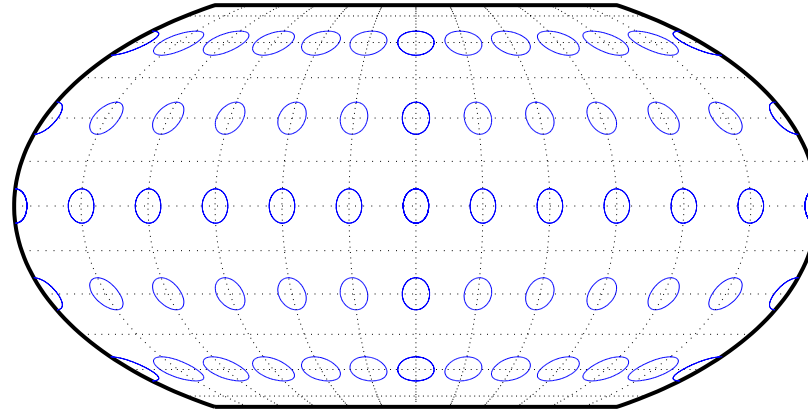
Symmetry: About the central meridian or the Equator.

Features This is an equal-area projection. Scale is true along the 42°59' parallels and is constant along any parallel and between any pair of parallels equidistant from the Equator. Distortion is not as extreme near the outer meridians at high latitudes as for pointed-polar pseudocylindrical projections, but there is considerable distortion throughout the polar regions. It is free of distortion only at the two points where the 42°59' parallels intersect the central meridian. This projection is not conformal or equidistant.

Parallels For this projection, only one standard parallel is specified. The other standard parallel is the same latitude with the opposite sign. The standard parallel is by definition fixed at 42°59'.

Remarks This projection was presented by Karlheinz Wagner in 1932.





Werner Projection

Classification Pseudoconic

Syntax `werner`

Graticule Central Meridian: A straight line.

Meridians: Complex curves connecting points equally spaced along each parallel and concave toward the central meridian.

Parallels: Concentric circular arcs spaced at true distances along the central meridian, centered on one of the poles.

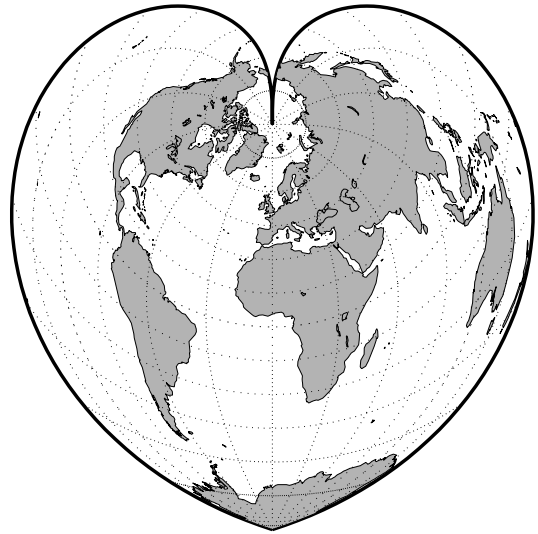
Poles: Points.

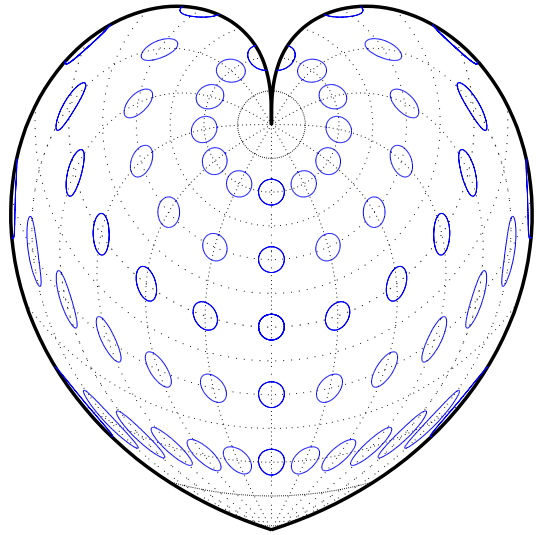
Symmetry: About the central meridian.

Features This is an equal-area projection. It is a Bonne projection with one of the poles as its standard parallel. The central meridian is free of distortion. This projection is not conformal. Its heart shape gives it the additional descriptor *cordiform*.

Parallels The standard parallel for this projection is set to 90° N.

Remarks This projection was developed by Johannes Stabius (Stab) about 1500 and was promoted by Johannes Werner in 1514. It is also called the Stab-Werner projection.





Wetch Cylindrical Projection

Classification Cylindrical

Syntax `wetch`

Graticule Central Meridian: Straight line (includes meridian opposite the central meridian in one continuous line).

Other Meridians: Straight lines if 90° from central meridian, complex curves concave toward the central meridian otherwise.

Parallels: Complex curves concave toward the nearest pole.

Poles: Points along the central meridian.

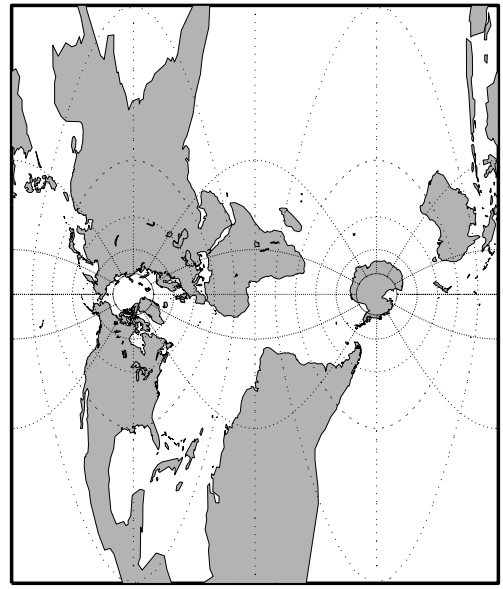
Symmetry: About any straight meridian or the Equator.

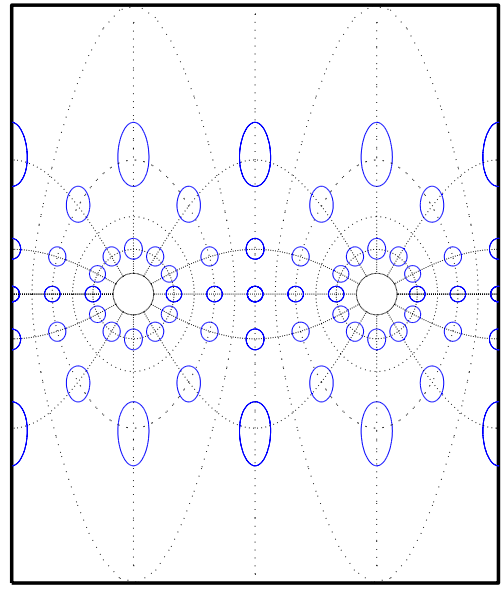
Features This is a perspective projection from the center of the Earth onto a cylinder tangent to the central meridian. It is not equal-area, equidistant, or conformal. Scale is true along the central meridian and constant between two points equidistant in *x* and *y* from the central meridian. There is no distortion along the central meridian, but it increases rapidly away from the central meridian in the *y*-direction.

Parallels For cylindrical projections, only one standard parallel is specified. The other standard parallel is the same latitude with the opposite sign. For this projection, which is the transverse aspect of the Central Cylindrical, the standard parallel *of the base projection* is by definition fixed at 0°.

Remarks This is the transverse aspect of the Central Cylindrical projection discussed by J. Wetch in the early 19th century.

Limitations This projection is available for the spherical geoid only. To prevent large *y*-values from dominating the display, data at *y*-values that would correspond to latitudes of greater than 75° in the normal aspect of the Central Cylindrical projection is trimmed.





Wiechel Projection

Classification Pseudoazimuthal

Syntax `wiechel`

Graticule The graticule described is for a polar aspect.

Meridians: Equally spaced semicircles from pole to pole, concave toward the west.

Parallels: Concentric circles.

Pole: The central pole is a point; the other pole is a bounding circle.

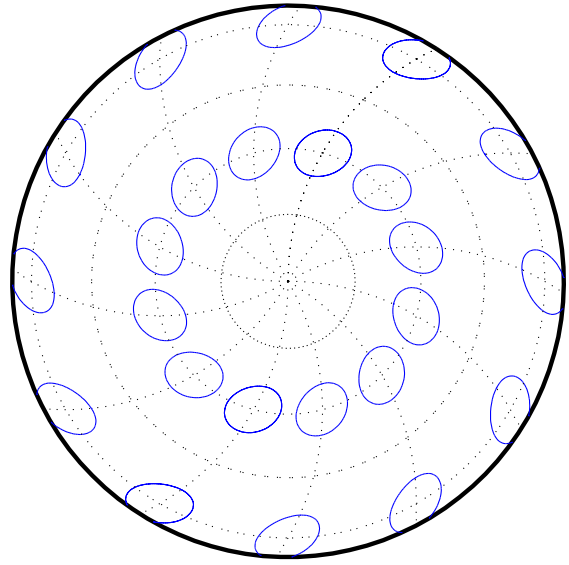
Symmetry: Radially about the center point.

Features This equal-area projection is a novelty map, usually centered at a pole, showing semicircular meridians in a pinwheel arrangement. Scale is correct along the meridians. This projection is not conformal.

Parallels There are no standard parallels for azimuthal projections.

Remarks This projection was presented by H. Wiechel in 1879.

Limitations Data greater than 65° distant from the center point is trimmed.



Winkel I Projection

Classification Pseudocylindrical

Syntax `winkel`

Graticule Central Meridian: Straight line at least half as long as the Equator.

Other Meridians: Equally spaced sinusoidal curves concave toward the central meridian.

Parallels: Equally spaced straight parallel lines, perpendicular to the central meridian.

Poles: Lines at least half as long as the Equator.

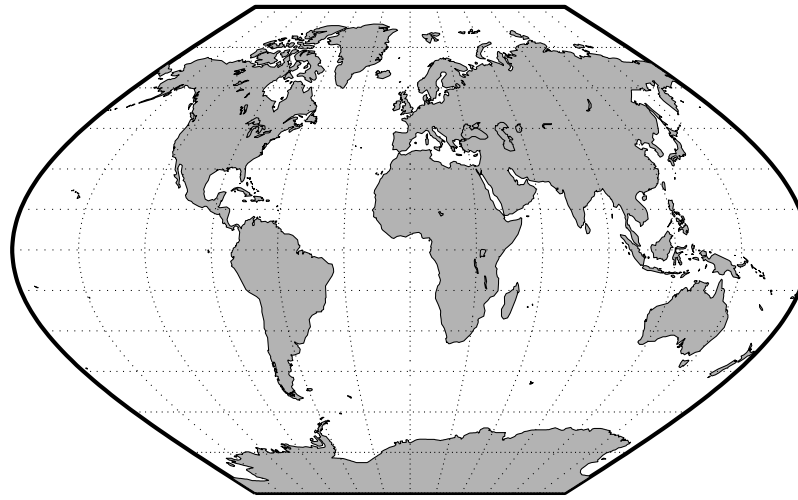
Symmetry: About the central meridian or the Equator.

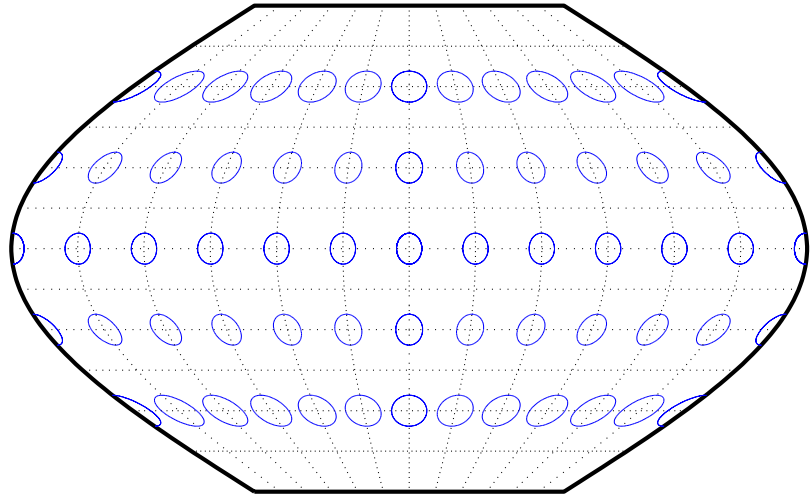
Features This projection is an arimethical average of the *x* and *y* coordinates of the Sinusoidal and Equidistant Cylindrical projections. Scale is true along the standard parallels and is constant along any parallel and between any pair of parallels equidistant from the Equator. There is no point free of distortion. This projection is not equal-area, conformal, or equidistant.

Parallels For this projection, only one standard parallel is specified. The other standard parallel is the same latitude with the opposite sign. Any latitude may be chosen; the default is set to 50°28'.

Remarks This projection was developed by Oswald Winkel in 1914. Its limiting form is the Eckert V when a standard parallel of 0° is chosen.

Limitations This projection is available for the spherical geoid only.





Winkel I Projection

Index

C

D

G

H

I

K

L

M

N

O

P

Q

R

S

T

U

V

W

Y

Z