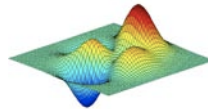


Applied Geoscience Data Analysis using Matlab

07 Nov 2013

(Lecture 20)

Referencing Matrix, Profiling & Terrain Analysis

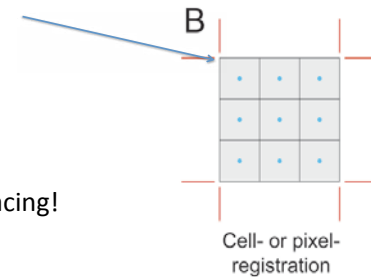


1

Referencing Vector (from last lecture)

$R = [\text{cells/angular unit, north-latitude, west-longitude}]$

Where north-latitude and west-longitude refer to the edges of the upper-left corner.



Requires equal x-y spacing!

Example from last lecture

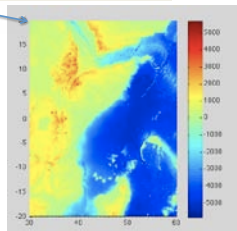
For ETOPO2 (2 minute or $2/60 = 1/30$ degree spacing)

(i.e., a dataset with $\frac{1}{2}$ the resolution of ETOPO1)

$R = [30, 20+1/60, 30-1/60];$

Limits of the node registered file, plus $\frac{1}{2}$ grid space to the north and west of north-west most node.

$R = 30.0 \quad 20.0167 \quad 29.9833$



Referencing Matrix (R)

- new/expanded in version 3.0 of the mapping toolbox
- Allows for a different spacing in the x and y dimensions of the grid

R is a 3-by-2 matrix that transforms pixel subscripts (row, column) to map coordinates (x,y) according to:

$$[x \ y] = [\text{row col } 1] * R$$

Or transforms pixel subscripts to/from geographic coordinates according to

$$[\text{lon lat}] = [\text{row col } 1] * R.$$

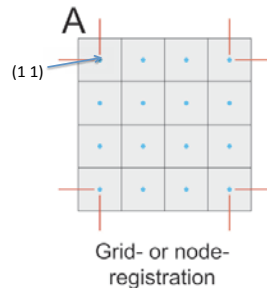
$R = \text{makerepmat}(X_{11}, Y_{11}, DX, DY)$

- DX and DY are scalars giving the spacing between pixels.

X_{11} and Y_{11} are scalars that specify the location of the center of the first (1,1) entry in the NODE registered sense.

DX is the difference in X (or longitude) between pixels in successive columns

DY is the difference in Y (or latitude) between pixels in successive rows.

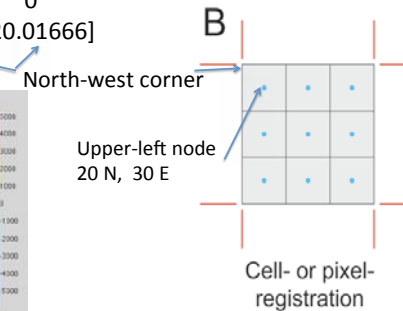
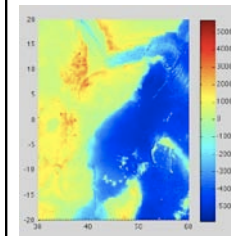


Referencing Matrix (example for ETOPO2 dataset)

$R = \text{makerepmat}(X_{11}, Y_{11}, DX, DY)$

$R = \text{makerepmat}(30, 20, 1/60, -1/60)$ % note the negative DY

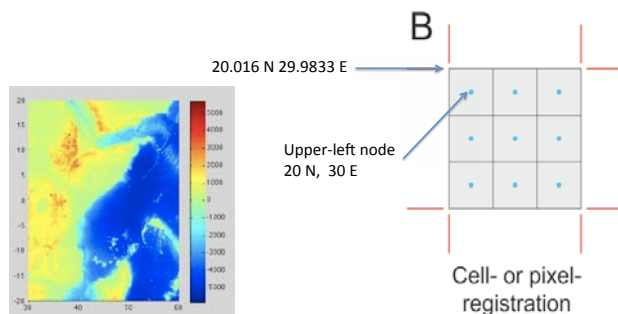
$R = \begin{bmatrix} 0 & -0.01667 \\ 0.01667 & 0 \\ 29.9833 & 20.01666 \end{bmatrix}$



$R = \begin{bmatrix} 0 & -0.01667 \\ 0.01667 & 0 \\ 29.9833 & 20.01666 \end{bmatrix}$

$[\text{lon lat}] = [\text{row col 1}] * R.$

$[1 \ 1 \ 1] * R$ Did this work?
 $= 30 \ 20$



Extracting values z from the geo-referenced gridded data

$VAL = \text{Itln2val}(Z, R, LAT, LON)$ interpolates a regular data grid Z with referencing vector R at the points specified by vectors of latitude and longitude, LAT and LON . R is either a 1-by-3 vector containing elements:

$[\text{cells/degree northern_latitude_limit western_longitude_limit}]$

or a 3-by-2 referencing matrix that transforms raster row and column indices to/from geographic coordinates according to:

$[\text{lon lat}] = [\text{row col 1}] * R.$

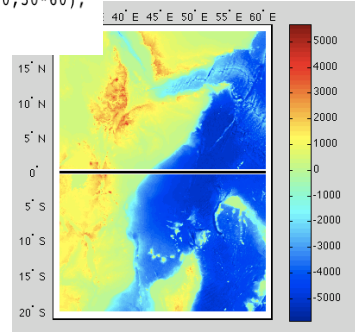
If R is a referencing matrix.

Profiling: 1st define points to sample

TRACK2 Geographic tracks from starting and ending points

`[lat,lon] = TRACK2(lat1,lon1,lat2,lon2)` computes great circle tracks on a sphere starting at the point lat1, lon1 and ending at lat2, lon2. The inputs can be scalar or column vectors.

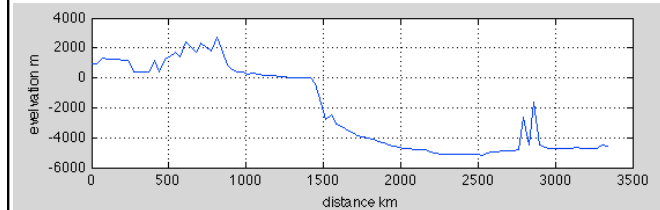
```
[tr_lat,tr_lon]=track2(0,30,0,60,30*60);
h=linem(tr_lat,tr_lon,'k');
set(h,'LineWidth',3)
```



Also See
track1
trackg

Then extract the values at these points

```
trz=ltln2val(ETOP01,R,tr_lat,tr_lon);
trd=distance(tr_lat(1),tr_lon(1),tr_lat,tr_lon);
figure; plot(deg2km(trd),trz); grid on
xlabel('distance km'); ylabel('elevation m');
```



An alternative to ltln2val.

`[zi,rng] = mapprofile(Z,R,lat,lon)` accepts as input a regular data grid and waypoint vectors. No displayed grid is required. Sets of waypoints may be separated by NaNs into line sequences. The output ranges are measured from the first waypoint within a sequence. R is either a 1-by-3 vector containing elements:

`[cells/degree northern_latitude_limit western_longitude_limit]`

or a 3-by-2 referencing matrix that transforms raster row and column indices to/from geographic coordinates according to:

$$[\text{lon lat}] = [\text{row col } 1] * R.$$

If R is a referencing matrix,

Many of the mapping toolbox function default to a spherical earth model; however, you can also specify the reference ellipsoid.

For example.

`[zi,rng] = mapprofile(Z,R,lat,lon,ellipsoid)`

Or

`[arclen,az] = distance(lat1,lon1,lat2,lon2,ellipsoid)`

- Where ellipsoid is a vector of the form:
`[semimajor_axis, eccentricity]`.
- The output, arclen, is expressed in the same length units as the semimajor axis of the ellipsoid.

Sample Ellipses with Various Eccentricities

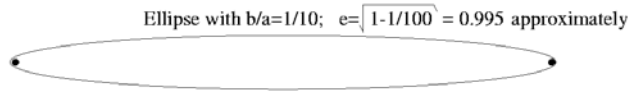
(focal points are marked; the diagrams are only approximately to scale)



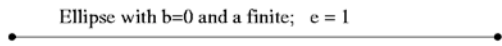
Circle with $b/a=1$; $e=0$



Ellipse with $b/a=3/5$; $e=\sqrt{1-9/25} = 4/5 = 0.8$



Ellipse with $b/a=1/10$; $e=\sqrt{1-1/100} = 0.995$ approximately



Ellipse with $b=0$ and a finite; $e=1$

DJ Jeffery
UNLV 2003

We can get specific ellipsoid models using almanac

almanac

R2012b

Parameters for Earth, planets, Sun, and Moon

Syntax

```
almanac
almanac(body)
data = almanac(body,parameter)
data = almanac(body,parameter,units)
data = almanac(parameter,units,referencebody)
```

Description

almanac is not recommended. Use [earthRadius](#), [referenceEllipsoid](#), [referenceSphere](#), or [wgs84Ellipsoid](#) instead.

Almanac (perhaps) being phased out in 2012 version,
but still functioning

'clarke80'	1880 Clarke ellipsoid
'international'	1924 International ellipsoid
'krasovsky'	1940 Krasovsky ellipsoid
'wgs60'	1960 World Geodetic System ellipsoid
'iau65'	1965 International Astronomical Union ellipsoid
'wgs66'	1966 World Geodetic System ellipsoid
'iau68'	1968 International Astronomical Union ellipsoid
'wgs72'	1972 World Geodetic System ellipsoid
'gra80'	1980 Geodetic Reference System ellipsoid
'wgs84'	1984 World Geodetic System ellipsoid

```
>> distance(35,-80,36,-81) % spherical earth
```

```
ans = 1.2895degrees
```

```
>> distance(35,-80,36,-81,almanac('earth','wgs84','degrees'))
```

```
ans = 1.2889 degrees
```

```
>> distance(35,-80,36,-81,almanac('earth','wgs84','meters'))
```

```
ans = 1.4332e+05 meters
```

```
>> distance(35,-80,36,-81,almanac('earth','wgs84','km'))
```

```
ans = 143.3216 km
```

A couple other ways to get topography & bathymetry: GeoMapApp



Explore our planet with
GeoMapApp

GeoMapApp®

GeoMapApp Links

- GeoMapApp Home
- Image Gallery
- Development History
- Help Pages
- Multimedia Tutorials
- Education
- GeoMapApp At Sea

Download Links

- Unix/Linux
- Macintosh
- Windows

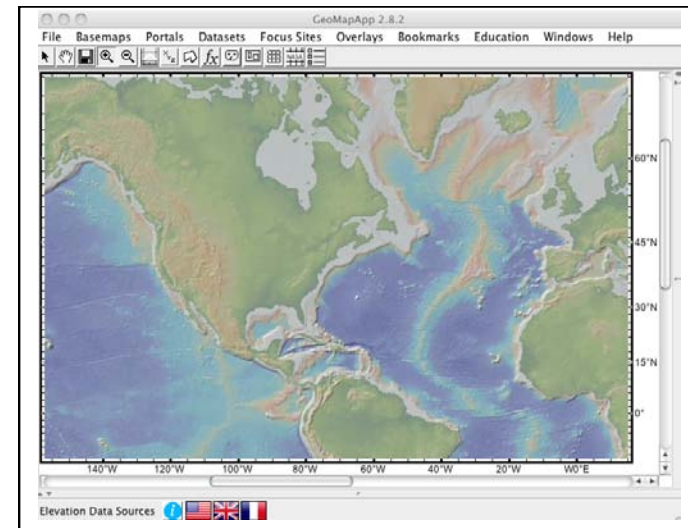
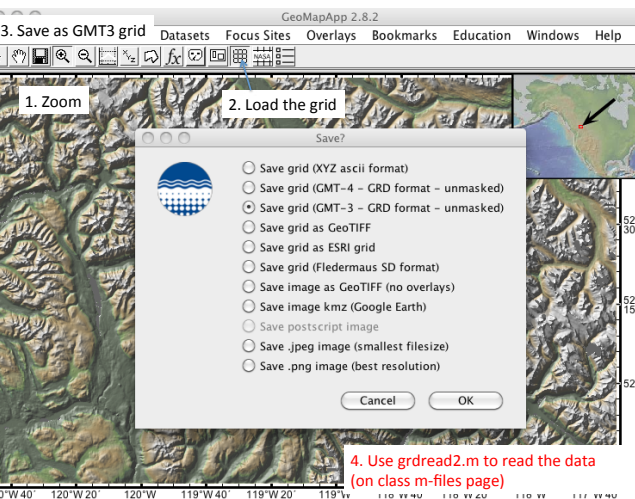
New! October 20, 2010: GeoMapApp version 2.8.2

Click [here](#) to launch version 2.8.2 of **GeoMapApp** using Java WebStart (J) or select from the **Download Links** menu to the left to install the application.

GeoMapApp is an earth science exploration and visualization application that is continually being expanded as part of the Marine Geoscience Data System (MGDS) at the Lamont-Doherty Earth Observatory of **Columbia University**. The application provides direct access to the Global Multi-Resolution Topography (GMRT) compilation that hosts high resolution (~100 m node spacing) bathymetry from multibeam data for ocean areas and Shuttle Radar Topography Mission elevations over land.

Requirements

The application runs in the Windows XP, Windows Vista, Mac OS X (10.4, 10.5), Linux and Solaris operating systems using the [Java Runtime Environment](#) (requires version 1.5.0_08 or more recent).

3. Save as GMT3 grid

1. Zoom

2. Load the grid

Save?

- ☐ Save grid (XYZ ascii format)
- ☐ Save grid (GMT-4 - GRD format - unmasked)
- ☐ Save grid (GMT-3 - GRD format - unmasked)
- ☐ Save grid as GeoTIFF
- ☐ Save grid as ESRI grid
- ☐ Save grid (Fiedermaus SD format)
- ☐ Save image as GeoTIFF (no overlays)
- ☐ Save image kmz (Google Earth)
- ☐ Save postscript image
- ☐ Save .jpeg image (smallest filesize)
- ☐ Save .png image (best resolution)

Cancel OK

4. Use `grdread2.m` to read the data (on class m-files page)

<http://www.marine-geo.org/portals/gmrt/>



MGDS GMRT Data Portal

<http://www.marine-geo.org/portals/gmrt/>

MARINE GEOSCIENCE DATA SYSTEM

Global Multi-Resolution Topography Data Portal

Portal Links

- Portal Home
- What's New
- Project Documents
- Related Links
- Contributors
- Media Bank
- Tutorials
- GeoMapApp
- Virtual Ocean

Access GMRT

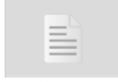
- Create Map/Grd
- Download for GoogleEarth
- Web Services

The Global Multi-Resolution Topography (GMRT) synthesis is a continuously-updated compilation of seafloor topography derived from multibeam bathymetry data. The synthesis began as the Ridge Multibeam Synthesis, was expanded to include multibeam bathymetry data from the Southern Ocean and now includes other bathymetry from throughout the global and coastal oceans. It is maintained as a multi-resolution gridded digital elevation model of seafloor bathymetry to ~100 m spatial resolution, and is merged with ~100 m resolution and topography data from the Shuttle Radar Topography Mission (SRTM30+). In the oceans, multibeam data are merged with lower-resolution compilations including satellite predicted bathymetry (Smith & Sandwell), the International Bathymetric Chart of the Arctic Ocean (IBCAO), and the SCAR Subglacial Topographic Model of the Antarctic (BEDMAP).

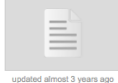
Additional information about the filing method and procedures used for creating and serving the Global Multi-Resolution Topography (GMRT) synthesis is available in [Ryan et al., 2009](#). The data DOI for GMRT is 10.1594/IEDA.3001000. View the list of contributors whose data have been incorporated into GMRT.

Download from the mathworks central page

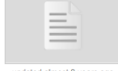
Sort By: Date Updated (Newest - Oldest) Watch Subscribe 1 - 3 of 3



grdwrite2 by Kelsey Jordahl
Uses built-in netCDF capability to write a grid file to be read by GMT (Generic Mapping Tools). (data export, netcdf, map)
fx `grdwrite2(x,y,z,file);`
1 Rating
1 Comment
9 Downloads (30 Days)
updated 1 year ago



grdread2 by Kelsey Jordahl
Uses built-in netCDF capability to load a grid file created by GMT (Generic Mapping Tools) v3 or v4. (data import, earth science, map)
fx `[x,y,z]=grdread2(file);`
4 Comments
23 Downloads (30 Days)
updated almost 3 years ago



grdinfo2 by Kelsey Jordahl
Uses built-in netCDF capability to display information about a grid file created by GMT v3 or v4. (earth science, map, netcdf)
fx `grdinfo2(file);`
0 Ratings
0 Comments
6 Downloads (30 Days)

grdread2 read in as NODE registered GMT data

```
>> [X,Y,Z]=grdread2('canrockies_gmt3.grd');
>> whos
```

Name	Size	Bytes	Class
X	1x723	5784	double
Y	1x475	3800	double
Z	475x723	1373700	single

```
dx=mode(diff(X)); % = 0.00439
```

```
dy=mode(diff(Y)); % = 0.0026
```

```
R=makereformat(X(1),Y(1),dx,dy)
```

```
R = [ 0      0.00269
      0.00439  0
      -120.744 51.680
```

Actual Header File Info:

```
grdinfo2('canrockies_gmt3.grd')
```

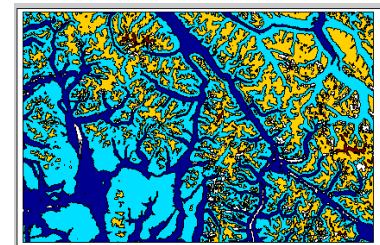
Gridline node registration used

```
x_min: -120.739746094    y_min: 51.6834549965
x_max: -117.566894531    y_max: 52.9565806756
x_inc: 0.00439453125     y_inc: 0.00268591915434
name: Longitude nx: 723   name: Latitude ny: 475
```

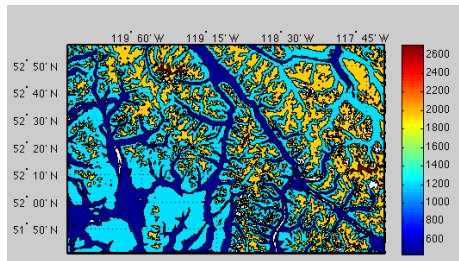
We can check that our R matrix return the correct result

```
[1 1 1]*R          [475 7231]*R
= -120.739746094    = -117.566894531
51.6834549965       52.9565806756
```

```
[X,Y,Z]=grdread2('canrockies_gmt3.grd');
dx=mode(diff(X));
dy=mode(diff(Y));
Z=double(Z); % make into a floating point value
R=makereformat(X(1),Y(1),dx,dy);
figure; axesm('MapProjection','mercator');
contourf(Z,R,3,'k')
```




```
setm(gca, 'MapLatLimit',[min(Y) max(Y)], 'MapLonLimit',[min(X) max(X)])
setm(gca, 'ParallelLabel','on', 'MeridianLabel','on', 'Grid','on', 'LabelUnits','dm')
setm(gca, 'MLabelLocation',45/60, 'PLabelLocation',10/60)
setm(gca, 'MlineLocation',45/60, 'PLineLocation',10/60)
tightmap; colorbar('vert');
```



Terrain Analysis:

Digital estimation of the slope, aspect, and curvatures of terrain data.

Leading to: terrain classification, flow path analysis, catchment delineation, solar radiation, channel lines, line of sight calculation.

Mapping toolbox function: gradientm

[aspect, slope, gradN, gradE] = gradientm(Z, R)

Computes the slope, aspect and north and east components of the gradient for a regular data grid Z with 1x 3 referencing vector or 3 x 2 reference matrix (if x and y grid spacing are different).

If the grid contains elevations in meters, the resulting aspect and slope are in units of **degrees clockwise from north and up from the horizontal**.

The north and east gradient components are the **change in the map variable per meter of distance in the north and east directions**.

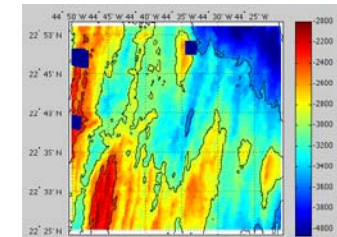
[...] = gradientm(lat, lon, Z)

lat and lon are the latitudes and longitudes of the geo-located points, and are in degrees.

```
[X,Y,Z]=grdread2('grid3_rs.grd');
Z=double(Z); dx=mode(diff(X)); dy=mode(diff(Y));
R=makerepmat(X(1),Y(1),dx,dy) % spacing not equal

figure; axesm('MapProjection','mercator');
meshm(Z,R); view(0,90); caxis([min(Z(:)),max(Z(:))]);
contour(Z,R,10,'k')
setm(gca, 'MapLatLimit',[min(Y) max(Y)], 'MapLonLimit',[min(X) max(X)])
setm(gca, 'ParallelLabel','on', 'MeridianLabel','on', 'Grid','on', 'LabelUnits','dm')
setm(gca, 'MLabelLocation',5/60, 'PLabelLocation',5/60)
setm(gca, 'MlineLocation',5/60, 'PLineLocation',5/60)
tightmap; colorbar('vert');
```

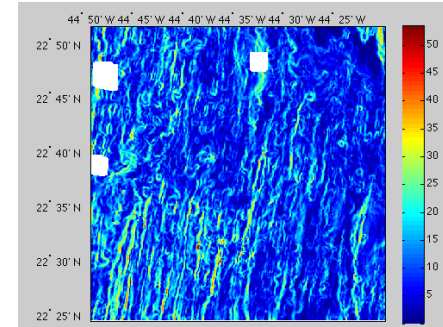
Terrain Analysis Example Flanks of Mid Atlantic Ridge



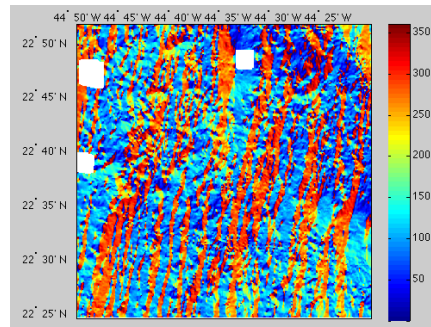
```
[aspect,slope,gradN,gradE] = gradientm(Z,R); % does all  
the calculations
```

```
[LAT, LON] = meshgrat(Z, R); % need for surfm
```

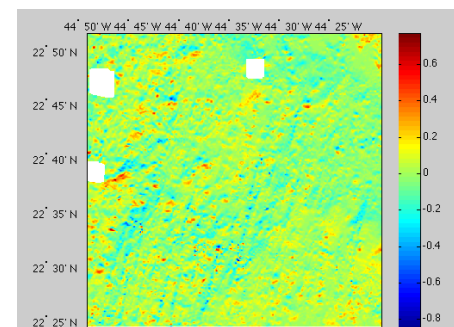
```
figure; axesm('MapProjection','mercator');  
surfm(LAT,LON,slope);  
setm(gca, 'MapLatLimit',[min(Y) max(Y)],'MapLonLimit',[min(X) max(X)])  
setm(gca,'ParallelLabel','on','MeridianLabel','on','LabelUnits','dm')  
setm(gca,'MLabelLocation',5/60,'PLabelLocation',5/60)  
setm(gca,'MlineLocation',5/60,'PLineLocation',5/60)  
tightmap; colorbar('vert');
```



```
figure; axesm('MapProjection','mercator');  
surfm(LAT,LON,aspect);  
setm(gca, 'MapLatLimit',[min(Y) max(Y)],'MapLonLimit',[min(X) max(X)])  
setm(gca,'ParallelLabel','on','MeridianLabel','on','LabelUnits','dm')  
setm(gca,'MLabelLocation',5/60,'PLabelLocation',5/60)  
setm(gca,'MlineLocation',5/60,'PLineLocation',5/60)  
tightmap; colorbar('vert');
```



```
figure; axesm('MapProjection','mercator');  
surfm(LAT,LON,gradN);  
setm(gca, 'MapLatLimit',[min(Y) max(Y)],'MapLonLimit',[min(X) max(X)])  
setm(gca,'ParallelLabel','on','MeridianLabel','on','LabelUnits','dm')  
setm(gca,'MLabelLocation',5/60,'PLabelLocation',5/60)  
setm(gca,'MlineLocation',5/60,'PLineLocation',5/60)  
tightmap; colorbar('vert');
```




```
figure;  
v=2:4:54;  
c=isnan(slope);  
hist(slope(~c),v); xlim([0,50])  
xlabel('slope'); ylabel('number of points')
```

