The Generic Mapping Tools

GMT

Version 4

A Map-Making Tutorial

by

Pål (Paul) Wessel

School of Ocean and Earth Science and Technology
University of Hawai'i at Mānoa

and

Walter H. F. Smith

Laboratory for Satellite Altimetry
NOAA/NESDIS/NODC

October 2004

Generic Mapping Tools Graphics

Contents

INTRODUCTION

The purpose of this tutorial is to introduce new users to GMT, outline the GMT environment, and enable you to make several forms of graphics without having to know too much about *UNIX* and *UNIX* tools. We will not be able to cover all aspects of GMT nor will we necessarily cover the selected topics in sufficient detail. Nevertheless, it is hoped that the exposure will prompt the users to improve their GMT and *UNIX* skills after completion of this short tutorial.

GMT overview: History, philosophy, and usage

Historical highlights

The GMT system was initiated in late 1987 at Lamont-Doherty Earth Observatory, Columbia University by graduate students Paul Wessel and Walter H. F. Smith. Version 1 was officially introduced to Lamont scientists in July 1988. GMT 1 migrated by word of mouth (and tape) to other institutions in the United States, UK, Japan, and France and attracted a small following. Paul took a Post-doctoral position at SOEST in December 1989 and continued the GMT development. Version 2.0 was released with an article in EOS, October 1991, and quickly spread worldwide. We obtained NSF-funding for GMT version 3.0 in 1993 which was released with another article in EOS on August 15, 1995. Significantly improved versions (3.1-3.3, 3.3.1–6), 3.4 and 3.4.1–4 were released between November 1998 and January 2004, culminating in the October 2004 introduction of 4.0 (and service release 3.4.5). GMT now is used by ∼6,000 users worldwide in a broad range of disciplines.

Philosophy

GMT follows the *UNIX* philosophy in which complex tasks are broken down into smaller and more manageable components. Individual GMT modules are small, easy to maintain, and can be used as any other *UNIX* tool. GMT is written in the ANSI C programming language (very portable), is POSIX compliant, and is independent of hardware constraints (e.g., memory). GMT was deliberately written for command-line usage, not a windows environment, in order to maximize flexibility. We standardized early on to use *PostScript* output instead of other graphics formats. Apart from the built-in support for coastlines, GMT completely decouples data retrieval from the main GMT programs. GMT uses architecture-independent file formats.

Why is GMT so popular?

The price is right! Also, GMT offers unlimited flexibility since it can be called from the command line, inside scripts, and from user programs. GMT has attracted many users because of its high quality *PostScript* output. GMT easily installs on almost any computer.

GMT installation considerations

GMT has been installed on machines ranging from super-computers to lap-top PCs. GMT only contains some 55,000 lines of code and has modest space/memory requirements. Minimum requirements are

- The netCDF library 3.4 or higher (free from www.unidata.edu).
- A C Compiler (free from www.gnu.org).
- About 100 Mb disk space (70 Mb additional for full- and high-resolution coast-lines).
- About 32 Mb memory.

In addition, we recommend access to a *PostScript* printer or equivalent (e.g., ***ghostscript***), *PostScript* previewer (e.g., ***ghostview***), any flavor of the *UNIX* operating system, and more disk space and memory.

1. SESSION ONE

1.1 Tutorial setup

1. We assume that GMT has been properly and fully installed and that you have the statement `setenv GMTHOME <path to GMT directory>` in your .login as described in the GMT README file.

2. All GMT man pages, documentation, and example scripts are available from the GMT documentation web page. It is assumed these pages have been installed locally at your site; if not they are always available from the main GMT home page[1].

3. We recommend you create a sub-directory called tutorial, cd into that directory, and copy all the tutorial files directly there with "`cp -r $GMTHOME/tutorial/* .` ".

4. As we discuss GMT principles it may be a good idea to consult the GMT Technical Reference and Cookbook for more detailed explanations.

5. The tutorial uses the supplemental GMT program **grdraster** to extract subsets of global gridded data sets. For your convenience we also supply the subsets in the event you do not wish to install **grdraster** and the public data sets it can read. Thus, run the **grdraster** commands if you have made the installation or ignore them if you have not.

6. For all but the simplest GMT jobs it is recommended that you place all the GMT (and *UNIX*) commands in a ***cshell*** script file and make it executable. To ensure that *UNIX* recognizes your script as a ***cshell*** script it is a good habit always to start the script with the line #!/bin/csh. All the examples in this tutorial assumes you are running the ***cshell***; if you are using something different then you are on your own.

7. Making a script executable is accomplished using the `chmod` command, e.g., the script figure_1.csh is made executable with "`chmod +x figure_1.csh`".

8. To view a *PostScript* file (e.g., map.ps) on a UNIX workstation we use ***ghostview*** map.ps. On some systems there will be similar commands, like imagetool and pageview on Sun workstations. In this text we will refer to ***ghostview***; please substitute the relevant *PostScript* previewer on your system.

9. Please cd into the directory tutorial. We are now ready to start.

1.2 The GMT environment: What happens when you run GMT?

To get a good grasp on GMT one must understand what is going on "under the hood". Figure 1.1 illustrates the relationships you need to be aware of at run-time.

1.2.1 Input data

A GMT program may or may not take input files. Three different types of input are recognized (more details can be found in Appendix B in the Technical Reference):

1. Data tables. These are "spreadsheet" tables with a fixed number of columns and unlimited number of rows. We distinguish between two groups:

   - ASCII (Preferred unless files are huge)
     - Single segment [Default]

[1] http://gmt.soest.hawaii.edu

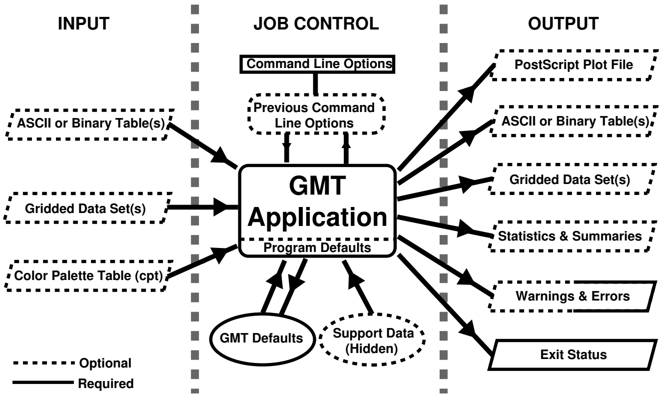

Figure 1.1: The **GMT** run-time environment.

   - Multi-segment with internal header records (**–M**)
 - Binary (to speed up input/output)
   - Single segment [Default]
   - Multi-segment (segment headers are all NaN fields) (**–M**)

2. Gridded dated sets. These are data matrices (evenly spaced in two coordinates) that come in two flavors:

   - Grid-line registration
   - Pixel registration

   You may choose among several file formats (even define your own format), but the GMT default is the architecture-indenpendent netCDF format.

3. Color palette table (For imaging, color plots, and contour maps). We will discuss these later.

1.2.2 Job Control

GMT programs may get operational parameters from several places:

1. Supplied command line options/switches or program defaults.
2. Short-hand notation to select previously used option arguments (stored in .gmtcommands4).
3. Implicitly using GMT defaults for a variety of parameters (stored in .gmtdefaults4).
4. May use hidden support data like coastlines or *PostScript* patterns.

1.2.3 Output data

There are 6 general categories of output produced by GMT:

1. *PostScript* plot commands.
2. Data Table(s).
3. Gridded data set(s).
4. Statistics & Summaries.

5. Warnings and Errors, written to *stderr*.
6. Exit status (0 means success, otherwise failure).

Note: GMT automatically creates and updates a history of past GMT command options for the common switches. This history file is called .gmtcommands and one will be created in every directory from which GMT programs are executed. Many initial problems with GMT usage result from not fully appreciating the relationships shown in Figure 1.1.

1.3 The UNIX Environment: Entry Level Knowledge

1.3.1 Redirection

Most GMT programs read their input from the terminal (called *stdin*) or from files, and write their output to the terminal (called *stdout*). To use files instead one can use *UNIX* redirection:

```
GMTprogram input-file >! output-file
GMTprogram < input-file >! output-file
GMTprogram input-file >> output-file       # Append to existing file
```

The exclamation sign (!) allows us to overwrite existing files.

1.3.2 Piping (|)

Sometimes we want to use the output from one program as input to another program. This is achieved with *UNIX* pipes:

```
Someprogram | GMTprogram1 | GMTprogram2 >!  Output-file (or | lp)
```

1.3.3 Standard error (*stderr*)

Most *UNIX* and GMT programs will on occasion write error messages. These are typically written to a separate data stream called *stderr* and can be redirected separately from the standard output (which goes to *stdout*). To redirect error messages we use

```
UNIXprogram >& errors.log
```

When we want to save both program output and error messages to separate files we use the following syntax:

```
(GMTprogram > output.d) >& errors.log
```

1.3.4 File name expansion or "wild cards"

UNIX provides several ways to select groups of files based on name patterns (Table 1.1):

Code	*Meaning*
*	Matches anything
?	Matches any single character
[*list*]	Matches characters in the list
[*range*]	Matches characters in the given range

Table 1.1: *UNIX* wildcards.

You can save much time by getting into the habit of selecting "good" filenames that make it easy to select subsets of all files using the *UNIX* wild card notation.

Examples

- GMTprogram data_*.d operates on all files starting with "data_" and ending in ".d".
- GMTprogram line_?.d works on all files starting with "line_" followed by any single character and ending in ".d".
- GMTprogram section_1[0-9]0.part_[12] only processes data from sections 100 through 190, only using every 10th profile, and gets both part 1 and 2.

1.4 Laboratory Exercises

We will begin our adventure by making some simple plot axes and coastline basemaps. We will do this in order to introduce the all-important **–B**, **–J**, and **–R** switches and to familiarize ourselves with a few selected GMT projections. The GMT programs we will utilize are **psbasemap** and **pscoast**. Please consult their manual pages on the GMT web site for reference.

1.4.1 Linear projection

We start by making the basemap frame for a linear *x*-*y* plot. We want it to go from 10 to 70 in *x*, annotating every 10, and from -3 to 8 in *y*, annotating every 1. The final plot should be 4 by 3 inches in size. Here's how we do it:

```
psbasemap -R10/70/-3/8 -JX4i/3i -B10/1:."My first plot": -P >! plot.ps
```

You can view the result with ***ghostview*** plot.ps.

Exercises

1. Try change the **–JX** values.
2. Try change the **–B** values.
3. Omit the **–P**.

1.4.2 Logarithmic projection

We next will show how to do a basemap for a log–log plot. We will assume that the raw *x* data range from 3 to 9613 and *y* ranges from $3.2 \cdot 10^{20}$ to $6.8 \cdot 10^{24}$. One possibility is

```
psbasemap -R1/10000/1e20/1e25 -JX9il/6il \
   -B2:"Wavelength (m)":/a1pf3:"Power (W)":WS >! plot.ps
```

(The backslash \ makes *UNIX* ignore the carriage return that follows and treat the two lines as one long command).

Exercises

1. Do not append **l** to the axes lengths.
2. Leave the **p** modifier out of the **–B** string.
3. Add **g**3 to each side of the slash in **–B**.

1.4.3 Mercator projection

Despite the problems of extreme horizontal exaggeration at high latitudes, the conformal Mercator projection (**–JM**) remains the stalwart of location maps used by scientists. It is one of several cylindrical projections offered by GMT; here we will only have time to focus on one such projection. The complete syntax is simply

–JM*width*

To make coastline maps we use **pscoast** which automatically will access the GMT coastline data base derived from the GSHHS database[2]. In addition to the common switches we may need to use some of several **pscoast** -specific options (see Table 1.2).

Option	*Purpose*
–A	Exclude small features or those of high hierarchical levels (see Appendix K)
–D	Select data resolution (**f**ull, **h**igh, **i**ntermediate, **l**ow, or **c**rude)
–G	Set color of dry areas (default does not paint)
–I	Draw rivers (chose features from one or more hierarchical categories)
–L	Plot map scale (length scale can be km, miles, or nautical miles)
–N	Draw political borders (including US state borders)
–S	Set color for wet areas (default does not paint)
–W	Draw coastlines and set pen thickness

Table 1.2: Main options when making coastline plots or overlays.

One of **–W**, **–G**, **–S** must be selected. Our first coastline example is from Latin America:

```
pscoast -R-90/-70/0/20 -JM6i -P -B5g5 -G180/120/60 >! map.ps
```

Exercises

1. Add the **–V** option.
2. Try **–R**270/290/0/20 instead. What happens to the annotations?
3. Edit your .gmtdefaults4 file, change **PLOT_DEGREE_FORMAT** to another setting (see the **gmtdefaults** man page), and plot again.
4. Pick another region and change land color.
5. Pick a region that includes the north or south poles.
6. Try **–W**0.25**p** instead of (or in addition to) **–G**.

1.4.4 Albers projection

The Albers projection (**–JB**) is an equal-area conical projection; its conformal cousin is the Lambert conic projection (**–JL**). Their usages are almost identical so we will only use the Albers here. The general syntax is

–JB$lon_0/lat_0/lat_1/lat_2/width$

where (lon_0, lat_0) is the map (projection) center and lat_1, lat_2 are the two standard parallels where the cone intersects the Earth's surface. We try the following command:

```
pscoast -R-130/-70/24/52 -JB-100/35/33/45/6i -B10g5:."Conic Projection": \
   -N1/2p -N2/0.25p -A500 -G200 -W0.25p -P >! map.ps
```

[2] See *Wessel and Smith* [1996].

Exercises

1. Change the parameter **GRID_CROSS_SIZE_PRIMARY** to make grid crosses instead of gridlines.
2. Change **–R** to a rectangular box specification instead of minimum and maximum values.

1.4.5 Orthographic projection

The azimuthal orthographic projection (**–JG**) is one of several projections with similar syntax and behavior; the one we have chosen mimics viewing the Earth from space at an infinite distance; it is neither conformal nor equal-area. The syntax for this projection is

–JG$lon_0/lat_0/width$

where (lon_0, lat_0) is the center of the map (projection). As an example we will try

```
pscoast -R0/360/-90/90 -JG280/30/6i -Bg30/g15 -Dc -A5000 -G255/255/255 \
    -S150/50/150 -P >! map.ps
```

Exercises

1. Use the rectangular option in **–R** to make a rectangular map showing the US only.

1.4.6 Eckert IV and VI projection

We conclude the survey of map projections with the Eckert IV and VI projections (**–JK**), two of several projections used for global thematic maps; They are both equal-area projections whose syntax is

–JK[**f**|**s**]$lon_0/width$

where **f** gives Eckert IV (4) and **s** (Default) gives Eckert VI (6). The lon_0 is the central meridian (which takes precedence over the mid-value implied by the **–R** setting). A simple Eckert VI world map is thus generated by

```
pscoast -R0/360/-90/90 -JKs180/9i -B60g30/30g15 -Dc -A5000 -G180/120/60 \
    -S100/180/255 -W0.25p >! map.ps
```

Exercises

1. Center the map on Greenwich.
2. Add a map scale with **–L**.

2. SESSION TWO

2.1 General Information

There are 18 GMT programs that directly create (or add overlays to) plots (Table 2.1); the remaining 45 are mostly concerned with data processing. This session will focus on the task of plotting lines, symbols, and text on maps. We will build on the skills we acquired while familiarizing ourselves with the various GMT map projections as well as how to select a data domain and boundary annotations.

Program	*Purpose*
	BASEMAPS
psbasemap	Create an empty basemap frame with optional scale
pscoast	Plot coastlines, filled continents, rivers, and political borders
pslegend	Create legend overlay
	POINTS AND LINES
pswiggle	Draw spatial time-series along their (x,y)-tracks
psxy	Plot symbols, polygons, and lines in 2-D
psxyz	Plot symbols, polygons, and lines in 3-D
	HISTOGRAMS
pshistogram	Plot a rectangular histogram
psrose	Plot a polar histogram(sector/rose diagram)
	CONTOURS
grdcontour	Contouring of 2-D gridded data sets
pscontour	Direct contouring or imaging of *xyz* data by optimal triangulation
	SURFACES
grdimage	Produce color images from 2-D gridded data
grdvector	Plot vector fields from 2-D gridded data
grdview	3-D perspective imaging of 2-D gridded data
	UTILITIES
psclip	Use polygon files to initiate custom clipping paths
psimage	Plot Sun rasterfiles
psmask	Create clipping paths or generate overlay to mask
psscale	Plot grayscale or colorscale bar
pstext	Plot textstrings on maps

Table 2.1: List of all 1-D and 2-D plotting programs in **GMT**.

Plotting lines and symbols, **psxy** is one of the most frequently used programs in GMT. In addition to the common command line switches it has numerous specific options, and expects different file formats depending on what action has been selected. These circumstances make **psxy** harder to master than most GMT tools. Table 2.2 shows a complete list of the options.

Option	*Purpose*
–A	Suppress line interpolation along great circles
–C*cpt*	Let symbol color be determined from *z*-values and the *cpt* file
–E[**x**|**X**][**y**|**Y**][*cap*][/*pen*]	Draw selected error bars with specified attributes
–G*fill*	Set color for symbol or fill for polygons
–L	Explicitly close polygons
–M[*flag*]	Multiple segment input data; headers start with *flag*
–N	Do Not clip symbols at map borders
–S[symbol][*size*]	Select one of several symbols (See Table 2.3)
–W*pen*	Set *pen* for line or symbol outline

Table 2.2: Optional switches in the **psxy** program.

The symbols can either be transparent (using **–W** only, not **–G**) or solid (**–G**, with optional outline using **–W**). The **–S** option takes the code for the desired symbol and optional size information. If no symbol is given it is expected to be given in the last column of each record in the input file. The *size* is optional since individual sizes for symbols may also be provided by the input data. The main symbols available to us are shown in Table 2.3.

Option	*Symbol*
–S-*size*	horizontal dash; *size* is length of dash
–Sa*size*	st**a**r; *size* is radius of circumscribing circle
–Sb*size*[/*base*][**u**]	**b**ar; *size* is bar width, append **u** if *size* is in *x*-units Bar extends from *base* [0] to the *y*-value
–Sc*size*	**c**ircle; *size* is the diameter
–Sd*size*	**d**iamond; *size* is its side
–Se	**e**llipse; *direction* (CCW from horizontal), *major*, and *minor* axes in inches are read from the input file
–SE	**e**llipse; *azimuth* (CW from vertical), *major*, and *minor* axes in kilometers are read from the input file
–Sg*size*	octa**g**on; *size* is its side
–Sh*size*	**h**exagon; *size* is its side
–Si*size*	**i**nverted triangle; *size* is its side
–Sk*symbol*/*size*	**k**ustom symbol; *size* is its side
–Sl*size*/*string*[%*font*]	**l**etter; *size* is fontsize. Append a letter or text string, and optionally a font
–Sn*size*	pe**n**tagon; *size* is its side
–Sp	**p**oint; no size needed (1 pixel at current resolution is used)
–Sr*size*	**r**ect, *width* and *height* are read from input file
–Ss*size*	**s**quare, *size* is its side
–St*size*	**t**riangle; *size* is its side
–Sv[*thick*/*length*/*width*][**n***norm*]	**v**ector; *direction* (CCW from horizontal) and *length* are read from input data Optionally, append the thickness of the vector and the width and length of the arrow-head. If the **n***norm* is appended, all vectors whose lengths are less than *norm* will have their attributes scaled by length/*norm*
–SV[*thick*/*length*/*width*][**n***norm*]	**v**ector, except *azimuth* (degrees east of north) is expected instead of *direction* The angle on the map is calculated based on the chosen map projection
–Sw[*size*	pie **w**edge; *start* and *stop* directions (CCW from horizontal) are read from input data
–Sx*size*	cross; *size* is length of crossing lines
–Sy*size*	vertical dash; *size* is length of dash

Table 2.3: The symbol option in **psxy**. Lower case symbols (**a, c, d, g, h, i, n, s, t, x**) will fit inside a circle of given diameter. Upper case symbols (**A, C, D, G, H, I, N, S, T, X**) will have area equal to that of a circle of given diameter.

Because some symbols require more input data than others, and because the size of symbols as well as their color can be determined from the input data, the format of data can be confusing. The general format for the input data is (optional items are in brackets []):

x y [z] [*size*] [σ_x] [σ_y] [*symbol*]

Thus, the only required input columns are the first two which must contain the longitude and latitude (or x and y). The remaining items apply when one (or more) of the following conditions are met:

1. If you want the color of each symbol to be determined individually, supply a cptfile with the **–C** option and let the 3rd data column contain the z-values to be used with the cptfile.

2. If you want the size of each symbol to be determined individually, append the size in a separate column.

3. To draw error bars, use the **–E** option and give one or two additional data columns with the $\pm dx$ and

$\pm dy$ values; the form of **–E** determines if one (**–Ex** or **–Ey**) or two (**–Exy**) columns are needed. If upper case flags **X** or **Y** are given then we will instead draw a "box-and-whisker" symbol and the σ_x (or σ_y) must represent 4 columns containing the minimum, the 25 and 75% quartiles, and the maximum value. The given x (or y) coordinate is taken as the 50% quartile (median).

4. If you draw vectors with **–Sv** (or **–SV**) then *size* is actually two columns containing the *direction* (or *azimuth*) and *length* of each vector.

5. If you draw ellipses (**–Se**) then *size* is actually three columns containing the *direction* and the *major* and *minor* axes in plot units (with **–SE** we expect *azimuth* instead and axes lengths in km).

Before we try some examples we need to review two key switches; they specify pen attributes and symbol or polygon fill. Please consult Chapter 4 in the GMT Technical Reference and Cookbook before experimenting with the examples below.

2.1.1 Examples

We will start off using the file data in your directory. Using the GMT utility **minmax** we find the extent of the data region:

```
minmax data
```

which returns

```
data: N = 7   <1/5>   <1/5>
```

telling us that the file data has 7 records and gives the minimum and maximum values for the first two columns. Given our knowledge of how to set up linear projections with **–R** and **–JX**, try the following:

1. Plot the data as transparent circles of size 0.3 inches.
2. Plot the data as solid white circles instead.
3. Plot the data using 0.5" stars, making them red with a thick (width = 1.5p), dashed pen.

To simply plot the data as a line we choose no symbol and specify a pen thickness instead:

```
psxy data -R -JX -P -B -W0.5p >! plot.ps
```

2.1.2 Exercises

1. Plot the data as a green-blue polygon instead.
2. Try using a predefined pattern.

A common question is : "How can I plot symbols connected by a line with psxy?". The surprising answer is that we must call **psxy** twice. While this sounds cumbersome there is a reason for this: Basically, polygons need to be kept in memory since they may need to be clipped, hence computer memory places a limit on how large polygons we may plot. Symbols, on the other hand, can be plotted one at the time so there is no limit to how many symbols one may plot. Therefore, to connect symbols with a line we must use the overlay approach:

```
psxy data -R -JX -B -P -K -W0.5p >! plot.ps
psxy data -R -JX -O -W -Si0.2i >> plot.ps
```

Our final **psxy** example involves a more complicated scenario in which we want to plot the epicenters of several earthquakes over the background of a coastline basemap. We want the symbols to have a size that reflects the magnitude of the earthquakes, and that their color should reflect the depth of the hypocenter. You will find the two files quakes.ngdc and quakes.cpt in your directory. The first few lines in the quakes.ngdc looks like this:

```
Historical Tsunami Earthquakes from the NGDC Database
Year  Mo  Da  Lat+N  Long+E  Dep  Mag
1987  01  04  49.77  149.29  489  4.1
1987  01  09  39.90  141.68  067  6.8
```

Thus the file has three header records (including the blank line), but we are only interested in columns 5, 4, 6, and 7. In addition to extract those columns we must also scale the magnitudes into symbols sizes in inches. Given their range it looks like multiplying the magnitude by 0.02 will work well. Reformatting this file to comply with the **psxy** input format can be done in a number of ways, including manual editing, using MATLAB, a spreadsheet program, or *UNIX* tools. Here, without further elaboration, we simply use the *UNIX* tool ***awk*** to do the job ($5 refers to the 5'th column etc., and NR is the current record number):

```
awk '{if (NR > 3) print $5, $4, $6, 0.02*$7}' quakes.ngdc >! quakes.d
```

The ***awk*** statement is automatically applied to each record, hence the output file quakes.d should now look like this (try it!):

```
149.29  49.77  489  0.082
141.68  39.90  067  0.136
...etc etc
```

We will follow conventional color schemes for seismicity and assign red to shallow quakes (depth 0–100 km), green to intermediate quakes (100–300 km), and blue to deep earthquakes (depth > 300 km). The quakes.cpt file establishes the relationship between depth and color:

```
# color palette for seismicity
#z0   red  green blue    z1   red green blue
0     255      0    0   100   255     0    0
100     0    255    0   300     0   255    0
300     0      0  255  1000     0     0  255
```

Apart from comment lines (starting with #), each record in the cpt file governs the color of a symbol whose z value falls in the range between z_0 and z_1. If the lower and upper red/green/blue triplets differ then an intermediate color will be linearly interpolated given the z value. Here, we have chosen constant color intervals.

We may now complete our example using the Mercator projection; we throw in a map scale out of pure generosity:

```
pscoast -R130/150/35/50 -JM6i -B5 -P -G200 -Lf134/49/42.5/500 -K >! map.ps
psxy -R -JM -O -Cquakes.cpt quakes.d -Sci -W0.25p >> map.ps
```

where the **i** appended to the **–Sc** option ensures that symbols sizes are interpreted to be in inches.

2.1.3 More exercises

1. Select another symbol.
2. Let the deep earthquakes be cyan instead of blue.

2.2 Plotting text strings

In many situations we need to annotate plots or maps with text strings; in GMT this is done using **pstext**. Apart from the common switches, there are 7 options that are particularly useful (Table 2.4).

Option	*Purpose*
–C*dx*/*dy*	Spacing between text and the text box (see **–W**)
–D*dx*/*dy*	Offsets the projected location of the strings
–G*fill*	Sets the color of the text
–L	Lists the font ids and exits
–N	Deactivates clipping at the borders
–S*pen*	Selects outline font and sets pen attributes
–W[*fill*][**o**[*pen*]]	Paint the text box; draw the outline if **o** is appended (also see **–C**)

Table 2.4: Some of the most frequently used options in **pstext**.

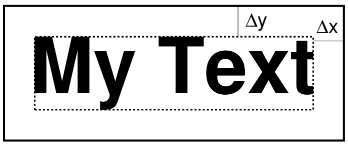

Figure 2.1: Relationship between the text box and the extra clearance.

The input data to **pstext** is expected to contain the following information:

x y size angle fontno justify text

The *size* argument is the font size in points, the *angle* is the angle (measured counterclockwise) between the text's baseline and the horizontal, *justify* indicates which point on the text-string should correspond to the given *x, y* location, and *text* is the text string or sentence to plot. Figure 2.2 illustrates these concepts and shows the relevant two-character codes used for justification.

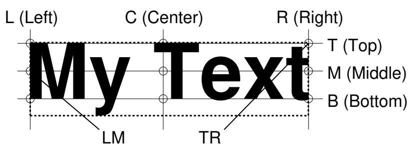

Figure 2.2: Justification (and corresponding character codes) for text strings.

The text string can be one or several words and may include octal codes for special characters and escape-sequences used to select subscripts or symbol fonts. The escape sequences that are recognized by GMT are given in Table 2.5.

Note that these escape sequences (as well as octal codes) can be used anywhere in GMT including in arguments to the **–B** option. A chart of octal codes can be found in Appendix F in the GMT technical reference book. For accented European characters you must set **CHAR_ENCODING** to ISOLatin1 in your .gmtdefaults4 file.

We will demonstrate **pstext** with the following script:

```
cat << EOF | pstext -R0/7/0/5 -Jx1i -P -B1g1 -G255/128/0 | ghostview -
1 1 30 0 4 BL It’s P@al, not Pal!
1 2 30 0 4 BL Try @%33%ZapfChancery@%% today
1 3 30 0 4 BL @~D@~g@-b@- = 2@~pr@~G@~D@~h.
1 4 30 0 4 BL University of Hawaii at M@!a\225noa
EOF
```

Code	*Effect*
@~	Turns symbol font on or off
@%*fontno*%	Switches to another font; @%% resets to previous font
@+	Turns superscript on or off
@-	Turns subscript on or off
@#	Turns small caps on or off
@!	Creates one composite character of the next two characters
@@	Prints the @ sign itself
@E @e	Æ æ
@O @o	Ø ø
@A @a	Å å

Table 2.5: **GMT** text escape sequences.

Here we have used the "here document" notation in *UNIX*: The $<<$ EOF will treat the following lines as the input file until it detects the word EOF. We pipe the *PostScript* directly through ***ghostview*** (the – tells ***ghostview*** that piping is happening).

2.3 Exercises

1. At $y = 5$, add the sentence "$z^2 = x^2 + y^2$".
2. At $y = 6$, add the sentence "It is 80° today".

3. SESSION THREE

3.1 Contouring gridded data sets

GMT comes with several utilities that can create gridded data sets; we will discuss two such programs later this session. First, we will assume that we already have gridded data sets. In the supplemental GMT archive there is a program that serves as a data extractor from several public domain global gridded data sets. Among these data are ETOPO5, crustal ages, gravity and geoid, and DEM for the continental US. Here, we will use **grdraster** to extract a GMT-ready grid that we will next use for contouring:

```
grdraster 1 -R-66/-60/30/35 -Gbermuda.grd -V
```

We first use the GMT program **grdinfo** to see what's in this file:

```
grdinfo bermuda.grd
```

The file contains bathymetry for the Bermuda region and has depth values from -5475 to -89 meters. We want to make a contour map of this data; this is a job for **grdcontour**. As with previous plot commands we need to set up the map projection with **–J**. Here, however, we do not have to specify the region since that is by default assumed to be the extent of the grid file. To generate any plot we will in addition need to supply information about which contours to draw. Unfortunately, **grdcontour** is a complicated program with too many options. We put a positive spin on this situation by touting its flexibility. Here are the most useful options:

Option	*Purpose*
–A*annot_int*	Annotation interval
–C*cont_int*	Contour interval
–G*gap*	Sets distance between contour annotations
–L*low*/*high*	Only draw contours within the *low* to *high* range
–N*unit*	Append *unit* to contour annotations
–Q*cut*	Do not draw contours with fewer than *cut* points
–S*smooth*	Resample contours every *x_inc*/*smooth* increment
–T[**+**|**-**][*gap*/*length*][:*LH*]	Draw tick-marks in downhill direction for innermost closed contours
	Add tick spacing and length, and characters to plot at the center of closed contours.
–W[**a**|**c**]*pen*	Set contour and annotation pens
–Z*factor*[/*offset*]	[Subtract *offset*] and multiply data by *factor* prior to processing

Table 3.1: The most useful options in **grdcontour**.

We will first make a plain contour map using 1 km as annotation interval and 250 m as contour interval. We choose a 7-inch-wide Mercator plot and annotate the borders every 2°:

```
grdcontour bermuda.grd -JM7i -C250 -A1000 -P -B2 | ghostview -
```

3.1.1 Exercises

1. Add smoothing with **–S**4.
2. Try tick all highs and lows with **–T**.
3. Skip small features with **–Q**10.
4. Override region using **–R**-70/-60/25/35.
5. Try another region that clips our data domain.
6. Scale data to km and use the km unit in the annotations.

3.2 Gridding of arbitrarily spaced data

Except in the situation above when a gridded file is available, we must convert our data to the right format readable by GMT before we can make contour plots and color-coded images. We distinguish between two scenarios:

1. The (x, y, z) data are available on a regular lattice grid.
2. The (x, y, z) data are distributed unevenly in the plane.

The former situation may require a simple reformatting (using **xyz2grd**), while the latter must be interpolated onto a regular lattice; this process is known as gridding. GMT supports three different approaches to gridding; here, we will briefly discuss the two most common techniques.

All GMT gridding programs have in common the requirement that the user must specify the grid domain and output filename:

–R*xmin*/*xmax*/*ymin*/*ymax*	The desired grid extent
–I*xinc*[**m**|**c**][/*yinc*[**m**|**c**]]	The grid spacing (append **m** or **c** for minutes or seconds of arc)
–G*gridfile*	The output grid filename

3.2.1 Nearest neighbor gridding

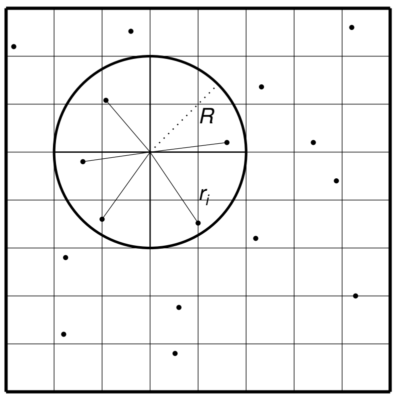

Figure 3.1: Search geometry for **nearneighbor**.

The GMT program **nearneighbor** implements a simple "nearest neighbor" averaging operation. It is the preferred way to grid data when the data density is high. **nearneighbor** is a local procedure which means it will only consider the control data that is close to the desired output grid node. Only data points inside a specified search radius will be used, and we may also impose the condition that each of the n sectors must have at least one data point in order to assign the nodal value. The nodal value is computed as a weighted average of the nearest data point per sector inside the search radius, with each point weighted according to its distance from the node as follows:

$$\bar{z} = \frac{\sum_{i=1}^{n} z_i w_i}{\sum_{i=1}^{n} w_i} \qquad w_i = \left(1 + \frac{9r_i^2}{R^2}\right)^{-1}$$

The most important switches are listed in Table 3.2.

We will grid the data in the file ship.xyz which contains ship observations of bathymetry off Baja California. We desire to make a 5' by 5' grid. Running **minmax** on the file yields

```
ship.xyz: N = 82970 <245/254.705><20/29.99131><-7708/-9>
```

so we choose the region accordingly:

Option	*Purpose*
–S*radius*[**k**]	Sets search radius. Append **k** to indicate radius in kilometers [Default is *x*-units]
–E*empty*	Assign this value to unconstrained nodes [Default is NaN]
–N*sectors*	Sector search, indicate number of sectors [Default is 4]
–W	Read relative weights from the 4th column of input data

Table 3.2: Switches used with the **nearneighbor** program.

```
nearneighbor -R245/255/20/30 -I5m -S40k -Gship.grd -V ship.xyz
```

We may get a view of the contour map using

```
grdcontour ship.grd -JM6i -P -B2 -C250 -A1000 | ghostview -
```

Exercises

1. Try using a 100 km search radius and a 10 minute grid spacing.

3.2.2 Gridding with Splines in Tension

As an alternative, we may use a global procedure to grid our data. This approach, implemented in the program **surface**, represents an improvement over standard minimum curvature algorithms by allowing users to introduce some tension into the surface. Physically, we are trying to force a thin elastic plate to go through all our data points; the values of this surface at the grid points become the gridded data. Mathematically, we want to find the function $z(x,y)$ that satisfies the following constraints:

$$\begin{array}{ll} z(x_k,y_k) = z_k, & \text{for all data } (x_k,y_k,z_k), k = 1,n \\ (1-t)\nabla^4 z - t\nabla^2 z = 0 & \text{elsewhere} \end{array}$$

where t is the “tension”, $0 \le t \le 1$. Basically, as $t \to 0$ we obtain the minimum curvature solution, while as $t \to \infty$ we go towards a harmonic solution (which is linear in cross-section). The theory behind all this is quite involved and we do not have the time to explain it all here, please see *Smith and Wessel* [1990] for details. Some of the most important switches for this program are indicated in Table 3.3[1].

Option	*Purpose*
–A*aspect*	Sets aspect ratio for anisotropic grids.
–C*limit*	Sets convergence limit. Default is 1/1000 of data range.
–T*tension*	Sets the tension [Default is 0]

Table 3.3: Some of the options in **surface**.

3.2.3 Preprocessing

The **surface** program assumes that the data have been preprocessed to eliminate aliasing, hence we must ensure that this step is completed prior to gridding. GMT comes with three preprocessors, called **blockmean**, **blockmedian**, and **blockmode**. The first averages values inside the grid-spacing boxes, the second returns median values, wile the latter returns modal values. As a rule of thumb, we use means for most smooth data (such as potential fields) and medians (or modes) for rough, non-Gaussian data (such as topography). In addition to the required **–R** and **–I** switches, these preprocessors all take the same options (listed in Table 3.4).

With respect to our ship data we preprocess it using the median method:

[1] The **–A** option is necessary for geographic grids since *x_inc* shrinks with latitude. Rule of thumb: set *aspect* = cosine of the average latitude.

Option	*Purpose*
–N	Choose pixel registration [Default is gridline]
–W[**i**|**o**]	Append **i** or **o** to read or write weights in the 4th column

Table 3.4: Some of the preprocessing options.

```
blockmedian -R245/255/20/30 -I5m -V ship.xyz >! ship_5m.xyz
```

The output data can now be used with surface:

```
surface ship_5m.xyz -R245/255/20/30 -I5m -Gship.grd -V
```

If you rerun **grdcontour** on the new grid file (try it!) you will notice a big difference compared to the grid made by **nearneighbor**: since **surface** is a global method it will evaluate the solution at all nodes, even if there are no data constraints. There are numerous options available to us at this point:

1. We can reset all nodes too far from a data constraint to the NaN value.
2. We can pour white paint over those regions where contours are unreliable.
3. We can plot the landmass which will cover most (but not all) of the unconstrained areas.
4. We can set up a clip path so that only the contours in the constrained region will show.

Here we have only time to explore the latter approach. The **psmask** program can read the same preprocessed data and set up a contour mask based on the data distribution. Once the clip path is activated we can contour the final grid; we finally deactivate the clipping with a second call to **psmask**. Here's the recipe:

```
psmask -R245/255/20/30 -I5m ship_5m.xyz -JM6i -B2 -P -K -V >! map.ps
grdcontour ship.grd -JM -O -K -C250 -A1000 >> map.ps
psmask -C -O >> map.ps
```

3.3 Exercises

1. Add the continents using any color you want.
2. Color the clip path light gray (use **–G** in the first **psmask** call).

4. SESSION FOUR

In our final session we will concentrate on color images and perspective views of gridded data sets. Before we start that discussion we need to cover two important aspects of plotting that must be understood. These are

1. Color tables and pseudo-colors in GMT.
2. Artificial illumination and how it affects colors.

4.1 Cpt files

The cpt file is discussed in detail in the GMT Technical Reference and COokbook, Chapter 4. Please review the format before experimenting further.

Cpt files can be created in any number of ways. GMT provides two mechanisms:

1. Create simple, linear color tables given a master color table (several are built-in) and the desired z-values at color boundaries (**makecpt**)
2. Create color tables based on a master cpt color table and the histogram-equalized distribution of z-values in a gridded data file (**grd2cpt**)

One can also make these files manually or with ***awk*** or other tools. Here we will limit our discussion to **makecpt**. Its main argument is the name of the master color table (a list is shown if you run the program with no arguments) and the equidistant z-values to go with it. The main options are given below.

Option	*Purpose*
–C	Set the name of the master cpt file to use
–I	Reverse the sense of the color progression
–V	Run in verbose mode
–Z	Make a continuous rather than discrete table

Table 4.1: Prime options available in **makecpt**.

To make discrete and continuous color cpt files for data that ranges from -20 to 60, with color changes at every 10, try these two variants:

```
makecpt -Crainbow -T-20/60/10 >! disc.cpt
makecpt -Crainbow -T-20/60/10 -Z >! cont.cpt
```

We can plot these color tables with **psscale**; the options worth mentioning here are listed in Table 4.2. In addition, the **–B** option can be used to set the title and unit label (and optionally to set the annotation-, tick-, and grid-line intervals for the colorbars.)

Option	*Purpose*
–C*cptfile*	The required cpt file
–D*xpos*/*ypos*/*length*/*width*[**h**]	Sets the position of the center/left and dimensions of scale bar.
	Append **h** to get horizontal bar and give center/top instead
–I*max_intensity*	Add illumination effects

Table 4.2: The main switches and options in **psscale**.

```
psbasemap -R0/8.5/0/11 -Jx1i -P -B0 -K >! bar.ps
psscale -D3i/3i/4i/0.5ih -Cdisc.cpt -B:discrete: -O -K >> bar.ps
psscale -D3i/5i/4i/0.5ih -Ccont.cpt -B:continuous: -O -K >> bar.ps
psscale -D3i/7i/4i/0.5ih -Cdisc.cpt -B:discrete: -I0.5 -O -K >> bar.ps
psscale -D3i/9i/4i/0.5ih -Ccont.cpt -B:continuous: -I0.5 -O >> bar.ps
```

4.1.1 Exercises

1. Redo the **makecpt** exercise using the master table *hot* and redo the bar plot.
2. Try specifying **–B**10g5.

4.2 Illumination and intensities

GMT allows for artificial illumination and shading. What this means is that we imagine an artificial sun placed at infinity in some azimuth and elevation position illuminating our surface. The parts of the surface that slope toward the sun should brighten while those sides facing away should become darker; no shadows are cast as a result of topographic undulations.

While it is clear that the actual slopes of the surface and the orientation of the sun enter into these calculations, there is clearly an arbitrary element when the surface is not topographic relief but some other quantity. For instance, what does the slope toward the sun mean if we are plotting a grid of heat flow anomalies? While there are many ways to accomplish what we want, GMT offers a relatively simple way: We may calculate the gradient of the surface in the direction of the sun and normalize these values to fall in the ± 1 range; +1 means maximum sun exposure and -1 means complete shade. Although we will not show it here, it should be added that GMT treats the intensities as a separate data set. Thus, while these values are often derived from the relief surface we want to image they could be separately observed quantities such as back-scatter information.

Colors in GMT are specified in the RGB system used for computer screens; it mixes red, green, and blue light to achieve other colors. The RGB system is a Cartesian coordinate system and produces a color cube. For reasons better explained in Appendix I in the Reference book it is difficult to darken and brighten a color based on its RGB values and an alternative coordinate system is used instead; here we use the HSV system. If you hold the color cube so that the black and white corners are along a vertical axis, then the other 6 corners project onto the horizontal plane to form a hexagon; the corners of this hexagon are the primary colors Red, Yellow, Green, Cyan, Blue, and Magenta. The CMY colors are the complimentary colors and are used when paints are mixed to produce a new color (this is how printers operate; they also add pure black (K) to avoid making gray from CMY). In this coordinate system the angle 0–360° is the hue (H); the Saturation and Value are harder to explain. Suffice it to say here that we intend to darken any pure color (on the cube facets) by keeping H fixed and adding black and brighten it by adding white; for interior points in the cube we will add or remove gray. This operation is efficiently done in the HSV coordinate system; hence all GMT shading operations involve translating from RGB to HSV, do the illumination effect, and transform back the modified RGB values.

4.3 Color images

Once a cpt file has been made it is relatively straightforward to generate a color image of a gridded data. Here, we will extract a subset of the global 30” DEM (data id 9) from USGS:

```
grdraster 9 -R-108/-103/35/40 -Gus.grd
```

Using **grdinfo** we find that the data ranges from $\sim$1000m to $\sim$4300m so we make a cpt file accordingly:

```
makecpt -Crainbow -T1000/5000/500 -Z >! topo.cpt
```

Color images are made with **grdimage** which takes the usual common command options (by default the **–R** is taken from the data set) and a cptfile; the main other options are

We want to make a plain color map with a color bar superimposed above the plot. We try

```
grdimage us.grd -JM6i -P -B2 -Ctopo.cpt -V -K >! topo.ps
psscale -D3i/8.5i/5i/0.25ih -Ctopo.cpt -I0.4 -B/:m: -O >> topo.ps
```

Option	*Purpose*
–E*dpi*	Sets the desired resolution of the image [Default is data resolution]
–I*intenfile*	Use artificial illumination using intensities from *intensfile*
–M	Force grayshade using the (television) YIQ conversion

Table 4.3: The main options in **grdimage**.

The plain color map lacks detail and fails to reveal the topographic complexity of this Rocky Mountain region. What it needs is artificial illumination. We want to simulate shading by a sun source in the east, hence we derive the required intensities from the gradients of the topography in the N90°E direction using **grdgradient**. Other than the required input and output filenames, the available options are

Option	*Purpose*
–A*azimuth*	Azimuthal direction for gradients
–M	Indicates that this is a geographic grid
–N[**t**|**e**][*norm*[/*offset*]]	Normalize gradients by *norm/offset* [= 1/0 by default]. Insert **t** to normalize by the tan^{-1} transformation. Insert **e** to normalize by the cumulative Laplace distribution.

Table 4.4: The **grdgradient** options.

Figure 4.1 shows that raw slopes from bathymetry tend to be far from normally distributed (left). By using the inverse tangent transformation we can ensure a more uniform distribution (right). The inverse tangent transform simply takes the raw slope estimate (the x value at the arrow) and returns the corresponding inverse tangent value (normalized to fall in the ± 1 range; horizontal arrow pointing to the y-value).

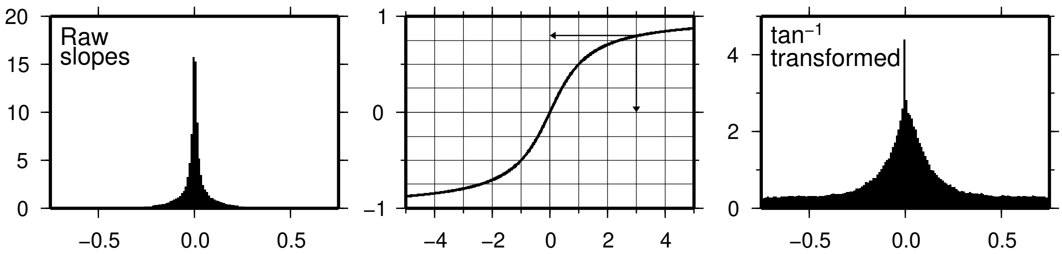

Figure 4.1: How the inverse tangent operation works.

Both **–Ne** and **–Nt** yield well behaved gradients. Personally, we prefer to use the **–Ne** option; the value of *norm* is subjective and you may experiment somewhat in the 0.5–5 range. For our case we choose

```
grdgradient us.grd -Ne0.8 -A100 -M -Gus_i.grd
```

Given the cpt file and the two gridded data sets we can create the shaded relief image:

```
grdimage us.grd -Ius_i.grd -JM6i -P -B2 -Ctopo.cpt -K >! topo.ps
psscale -D3i/8.5i/5i/0.25ih -Ctopo.cpt -I0.4 -B/:m: -O >> topo.ps
```

4.3.1 Exercises

1. Force a gray-shade image.
2. Rerun **grdgradient** with **–N**1.

4.4 Perspective views

Our final undertaking in this tutorial is to examine three-dimensional perspective views. GMT is currently limited to vantage points at infinity; thus we are unable to do fly-by's through canyons etc. The GMT module that produces perspective views of gridded data files is **grdview**. It can make two kinds of plots:

1. Mesh or wire-frame plot (with or without superimposed contours)
2. Color-coded surface (with optional shading, contours, or draping).

Regardless of plot type, some arguments must be specified; these are

1. *relief_file*; a gridded data set of the surface.
2. **–J** for the desired map projection.
3. **–JZ***height* for the vertical scaling.
4. **–E***azimuth*/*elevation* for vantage point.

In addition, some options may be required:

Option	*Purpose*
–C*cptfile*	The *cptfile* is required for color -coded surfaces and for contoured mesh plots
–G*drape_file*	Assign colors using *drape_file* instead of *relief_file*
–I*intens_file*	File with illumination intensities
–Qm	Selects mesh plot
–Qs[**m**]	Surface plot using polygons; append **m** to show mesh. This option allows for **–W**
–Qi*dpi*[**g**]	Image by scan-line conversion. Specify *dpi*; append **g** to force gray-shade image. **–B** is disabled.
–W*pen*	Draw contours on top of surface (except with **–Qi**)

Table 4.5: The most useful options in **grdview**.

4.4.1 Mesh-plot

Mesh plots work best on smaller data sets. We again use the small subset of the ETOPO5 data over Bermuda and make a quick-and-dirty cpt file:

```
grd2cpt bermuda.grd -Cocean >! bermuda.cpt
```

A simple mesh plot can therefore be obtained with

```
grdview bermuda.grd -JM5i -P -JZ2i -E135/30 -B2 -Cbermuda.cpt >! map.ps
```

Exercises

1. Select another vantage point and vertical height.

4.4.2 Color-coded view

We will make a perspective, color-coded view of the US Rockies from the southeast. This is done using

```
grdview us.grd -JM6i -E135/35 -Qi50 -Ius_i.grd -Ctopo.cpt -V -B2 \
    -JZ0.5i >! view.ps
```

This plot is pretty crude since we selected 50 dpi but it is fast to render and allows us to try alternate values for vantage point and scaling. When we settle on the final values we select the appropriate *dpi* for the final output device and let it rip.

Exercises

1. Choose another vantage point and scaling.
2. Redo **grdgradient** with another illumination direction and replot.
3. Select a higher *dpi*, e.g., 200.

5. References

1. Smith, W.H.F., and P. Wessel, Gridding with continous curvature splines in tension, *Geophysics*, *55*, 293–305, 1990.

2. Wessel, P., and W.H.F. Smith, Free software helps map and display data, *EOS Trans. AGU*, *72*, 441, 1991.

3. Wessel, P., and W.H.F. Smith, New version of the Generic Mapping Tools released, *EOS Trans. AGU*, *76*, 329, 1995.

4. Wessel, P., and W.H.F. Smith, A global, self-consistent, hierarchical, high-resolution shoreline database, *J. Geophys. Res.*, *101*, 8741–8743, 1996.

5. Wessel, P., and W.H.F. Smith, New, improved version of the Generic Mapping Tools released, *EOS Trans. AGU*, *79*, 579, 1998.

6. Wessel, P., and W.H.F. Smith, The Generic Mapping Tools Technical Reference and Cookbook, *Version 4.0*, pp. 132, 2004.

Index