6. Cook-book

In this section we will be giving several examples of typical usage of GMT programs. In general, we will start with a raw data set, manipulate the numbers in various ways, then display the results in diagram or map view. The resulting plots will have in common that they are all made up of simpler plots that have been overlaid to create a complex illustration. We will mostly follow the following format: (1) First we explain what we want to achieve in plain language, (2) then we present a shell script that contains all the commands used to arrive at the final illustration, and (3) we explain the rationale behind the commands. For some examples we must inspect the results from one command before proceeding to the next; in those instances we wrote the script to prompt for the necessary input. A detailed discussion of each command is not given; we refer you to the manual pages for command line syntax, etc. For all illustrations the original *PostScript* files have been enclosed. See Appendix D if you would like an electronic version of all the shell-scripts and support data used below. Note that all examples assume the inch is the default measurement units. If you have chosen cm as the unit and run these scripts you'll come up short (literally!).

- Example 1: The making of contour maps.

We want to create two contour maps of the low order geoid using the Hammer equal area projection. Our gridded data file is called `osu91a1f_16.grd` and contains a global 1° by 1° gridded geoid. We would like to show one map centered on Greenwich and one centered on the dateline. Positive contours should be drawn with a solid pen and negative contours with a dashed pen. Annotations should occur for every 50 m contour level, and both contour maps should show the continents in light gray in the background. Finally, we want a rectangular frame surrounding the two maps. We accomplish all this with the following shell script.

```
#!/bin/csh
# This script will make two contour maps based on the data in the file osu91a1f_16.grd
#
psbasemap -R0/6.5/0/9 -Jx1 -B0 -P -K -U"Example 1 in Cookbook" >! example_1.ps
pscoast -R0/360/-90/90 -JH0/6 -X0.25 -Y0.5 -O -K -Bg30 -Dc -G200 >> example_1.ps
grdcontour -R osu91a1f_16.grd -JH -C10 -A50f7 -L-1000/-1 -Wc1ta -Wa3t8_8:0 -O -K -T0.1/0.02 >>
example_1.ps
grdcontour -R osu91a1f_16.grd -JH -C10 -A50f7 -L-1/1000 -O -K -T0.1/0.02 >> example_1.ps
pscoast -R0/360/-90/90 -JH180/6 -Y4 -O -K -Bg30:."Low Order Geoid": -Dc -G200 >> example_1.ps
grdcontour osu91a1f_16.grd -JH -C10 -A50f7 -L-1000/-1 -Wc1ta -Wa3t8_8:0 -O -K -T0.1/0.02:-+ >>
example_1.ps
grdcontour osu91a1f_16.grd -JH -C10 -A50f7 -L-1/1000 -O -T0.1/0.02 >> example_1.ps
```

The first command draws a box surrounding the maps. This is followed by two sequences of *pscoast–grdcontour–grdcontour*. They differ in that the first is centered on Greenwich; the second on the dateline. We use the limit option (**–L**) in *grdcontour* to select negative contours only and plot those with a dashed pen, then positive contours only and draw with a solid pen [Default]. The **–T** option causes tickmarks pointing in the downhill direction to be drawn on the innermost, closed contours. For the upper panel we also added - and + to the local lows and highs. You can find this illustration as Figure 1 at the end of this document.

- Example 2: Image presentations.

As our second example we will demonstrate how to make color images from gridded data sets. We will use the supplemental program *grdraster* to extract 2-D grdfiles of

bathymetry and Geosat geoid heights and put the two images on the same page. The region of interest is the Hawaiian islands, and due to the oblique trend of the island chain we prefer to rotate our geographical data sets using an oblique Mercator projection defined by the hotspot pole at (68˚W, 69˚N). We choose the point (190˚, 25.5˚) to be the center of our projection (e.g., the local origin), and we want to image a rectangular region defined by the longitudes and latitudes of the lower left and upper right corner of region. In our case we choose (160˚, 20˚) and (220˚, 30˚) as the corners. We use *grdimage* to make the illustration:

```
#!/bin/csh
#
# This script will make two color images of topography and geoid
#
# Extract data
grdraster 1 -R160/20/220/30r -JOc190/25.5/292/69/4.5 -Getopo5.grd=0/0.001/0
grdraster 4 -R -JO -Ggeoid5.grd
# Make illustration
grdimage geoid5.grd -R -JO -K -P -U/-1.25/-1/"Example 2 in Cookbook" -V -B10 -Cgeoid.cpt -X1.5 -Y1.25 >!
example_2.ps
psscale -Cgeoid.cpt -D5.1/1.35/2.88/0.4 -O -K -L -B:m::.GEOID: >> example_2.ps
grdimage etopo5.grd -R -JO -B10:."H@#awaiian@# T@#opo and @#G@#eoid:" -O -K -V -Ctopo.cpt -Y4.5 >>
example_2.ps
psscale -Ctopo.cpt -D5.1/1.35/2.88/0.4 -O -K -B:m::.TOPO: >> example_2.ps
echo "-0.4 7.5 36 0.0 1 2 a)" | pstext -R0/8.5/0/11 -Jx1 -O -K -Y-4.5 >> example_2.ps
echo "-0.4 3.0 36 0.0 1 2 b)" | pstext -R -Jx -O >> example_2.ps
```

The first step extracts the 2-D data sets from the local data base using *grdraster*, which is a supplemental utility program (see Appendix A) that may be adapted to reflect the nature of your data base format. It automatically figures out the required extent of the region given the two corners points and the projection. Because we specified verbose we are informed that the extreme meridians and parallels enclosing the oblique region is equivalent to **–R**159:50/220:10/3:10/47:35. This is the area extracted by *grdraster*. By using the embedded grdfile format mechanism we saved the topography using kilometers as the data unit. We now have two grdfiles with bathymetry and geoid heights, respectively. Prior to running the script we created two color palette files: topo.cpt for the topography and geoid.cpt for the geoid data:

topo.cpt:

-7.1	0	27	255	-4	0	255	120
-4	0	255	120	-2.5	120	255	0
-2.5	120	255	0	-0.5	255	220	0
-0.5	255	220	0	1.5	255	70	0
1.5	255	70	0	2.5	255	255	255

geoid.cpt:

-1	40	0	150	0	40	0	150	L
0	0	10	200	1	0	10	200	L
1	0	40	224	2	0	40	224	
2	13	129	248	3	13	129	248	
3	50	190	255	4	50	190	255	
4	97	225	240	5	97	225	240	
5	124	235	200	6	124	235	200	L

6	172	245	168	7	172	245	168	
7	223	245	141	8	223	245	141	
8	247	215	104	9	247	215	104	
9	255	160	69	10	255	160	69	
10	238	80	78	11	238	80	78	L
11	255	124	124	12	255	124	124	
12	245	179	174	13	245	179	174	
13	255	215	215	14	255	215	215	
14	255	255	255	15	255	255	255	U

Note that we chose a continuous color table for topography and a discontinuous one for the geoid. We select only a few contours to be annotated in *psscale* by appending an extra column in the geoid.cpt file. L means annotate lower color-change, U means annotate upper color-change (B would mean both). Next we run *grdimage* to create a color-code image of the Geosat geoid heights, and draw a color scale to the right of the image with *psscale*. Similarly, we run *grdimage* but specify **–Y**4.5 to plot above the previous image. Adding scale and label the two plots a) and b) completes the illustration (Figure 2).

- Example 3: Spectral estimation and xy-plots.

In this example we will show how to use the GMT programs *fitcircle*, *project*, *sample1d*, *spectrum1d*, *psxy*, and *pstext*. Suppose you have (lon,lat,gravity) along a satellite track in a file called sat.xyg, and (lon,lat,gravity) along a ship track in a file called ship.xyg. You want to make a cross-spectral analysis of these data. First, you will have to get the two data sets into equidistantly sampled time-series form. To do this, it will be convenient to project these along the great circle that best fits the sat track. We must use *fitcircle* to find this great circle and choose the L_2 estimates of best pole. We project the data using *project* to find out what their ranges are in the projected coordinate. The *minmax* utility will report the minimum and maximum values for multi-column ASCII tables. Use this information to select the range of the projected distance coordinate they have in common. The script prompts you for that information after reporting the values. We decide to make a file of equidistant sampling points spaced 1 km apart from -1167 to +1169, and use the UNIX utility *awk* to accomplish this step. We can then resample the projected data, and carry out the cross-spectral calculations, assuming that the ship is the input and the satellite is the output data. The remaining commands will plot the ship and sat power in one diagram and the coherency on another diagram, both on the same page. Note the extended use of *pstext* and *psxy* to put labels and legends directly on the plots. For that purpose we often use **-Jx**1 and specify positions in inches directly. Thus, the complete automated script reads:

```
#!/bin/csh
# First run fircircle and get L2 average position and N hemisphere pole
#
fitcircle sat.xyg -L2 >! report
set cpos = `grep "L2 Average Position" report | awk '{ print $(NF-1), $NF}'`
set ppos = `grep "L2 N Hemisphere" report | awk '{print $(NF-1), $NF}'`
#
# Project the data to find out what their ranges are in the projected coordinates:
#
project  sat.xyg -C$cpos[1]/$cpos[2] -T$ppos[1]/$ppos[2] -S -V -Fpz -M >! sat.pg
project ship.xyg -C$cpos[1]/$cpos[2] -T$ppos[1]/$ppos[2] -S -V -Fpz -M >! ship.pg
#
# Use minmax to find the range of the projected distance coordinate they have in common:
#
minmax sat.pg ship.pg
```

```
echo -n "Enter the min max distance the data sets have in common (here, -1167 1169: "
set line = $<
set limits = ($line)
#
# Make a file of equidistant sampling points spaced 1 km apart over the common interval.
#
awk 'BEGIN { for (i = '$limits[1]'; i <= '$limits[2]'; i++) print i }' /dev/null >! samp.x
#
# Now you can resample the projected data:
#
sample1d sat.pg -Nsamp.x >! samp_sat.pg
sample1d ship.pg -Nsamp.x >! samp_ship.pg
#
# Get the cross-spectra, assuming that the ship is the input and the sat is the output data:
#
paste samp_ship.pg samp_sat.pg | cut -f2,4 | spectrum1d -S256 -D1 -V -W -C
#
# Now we plot the spectra
#
psxy spectrum.coh -Ba1f3p:"Wavelength (km)":/a0.25f0.05:"Coherency@+2@+":WeSn -JX-4l/3.75 -
R1/1000/0/1 -U/-2.25/-1.25/"Example 3 in Cookbook" -P -K -X2.5 -Sc0.07 -G0 -Ey/2 -Y1.5 >! example_3.ps
echo "3.85 3.6 18 0.0 1 11 Coherency@+2@+" | pstext -R0/4/0/3.75 -Jx1 -O -K >> example_3.ps
cat << END >! box.d
2.375   3.75
2.375   3.25
4       3.25
END
psxy -R -Jx1 -O -K -W5 box.d >> example_3.ps
psxy -St0.07 -O -Ba1f3p/a1f3p:"Power (mGal@+2@+km)"::."Ship and Satellite Gravity":WeSn
spectrum.xpower -R1/1000/0.1/10000 -JX-4l/3.75l -Y4.2 -K -Ey/2 >> example_3.ps
psxy spectrum.ypower -R -JX -O -K -G0 -Sc0.07 -Ey/2 >> example_3.ps
echo "3.9 3.6 18 0.0 1 11 Input Power" | pstext -R0/4/0/3.75 -Jx1 -O -K >> example_3.ps
psxy -R -Jx -O -K -W5 box.d >> example_3.ps
psxy -R -Jx -O -K -G240 -L -W5 << END >> example_3.ps
0.25    0.25
1.4     0.25
1.4     0.9
0.25    0.9
END
echo "0.4 0.7" | psxy -R -Jx -O -K -St0.07 -G0 >> example_3.ps
echo "0.5 0.7 14 0.0 1 5 Ship" | pstext -R -Jx -O -K >> example_3.ps
echo "0.4 0.4" | psxy -R -Jx -O -K -Sc0.07 -G0 >> example_3.ps
echo "0.5 0.4 14 0.0 1 5 Satellite" | pstext -R -Jx -O >> example_3.ps
```

The final illustration (Figure 3) shows that the ship gravity anomalies have more power than altimetry derived gravity for short wavelengths and that the coherency between the two signals improves dramatically for wavelengths > 20 km.

- Example 4: A 3-D perspective mesh plot.

This example will illustrate how to make a fairly complicated composite figure. We need a subset of the ETOPO5 bathymetry† and Geosat geoid data sets which we will extract from the local data bases using *grdraster*. We would like to show a 2-layer perspective plot where layer one shows a contour map of the marine geoid with the location of the Hawaiian islands superposed, and a second layer showing the 3-D mesh plot of the topography. We also add an arrow pointing north and some text. This is how one could do it:

† These data are available on CD-ROM from NGDC (contact pws@mail.ngdc.noaa.gov)

```
#!/bin/csh
#
# 3-D mesh plot of Hawaiian topography and geoid
#
grdraster 1 -R195/210/18/25 -Gtopo.grd
getraster 4 -R -Ggeoid.grd
echo '-10     255     0       255' >! zero.cpt
echo '0       100     10      100' >> zero.cpt
grdcontour geoid.grd -Jm0.45 -E60/30 -R195/210/18/25 -C1 -A5 -V -K -P -X1.5 -Y1.5 -U/-1.25/-1.25/"Example 4
in Cookbook" >! example_4.ps
pscoast -Jm -E60/30 -R -V -B2/2NEsw -G0 -O -K >> example_4.ps
echo '205 26 0 0 1.1' | psxyz -Jm -E60/30 -R -SV0.2/0.5/0.4 -W4 -O -K -N >> example_4.ps
echo '205 29.2 36 -90 1 5 N' | pstext -Jm -E60/30 -R -V -O -K -N >> example_4.ps
grdview topo.grd -Jm -Jz0.34 -Czero.cpt -E60/30 -R195/210/18/25/-6/4 -N-6/200/200/200 -Qsm -O -K -V -
B2/2/2:"Topo (km)":neswZ -Y2.2 >> example_4.ps
echo '3.25 5.75 60 0.0 33 2 H@#awaiian@# R@#idge' | pstext -R0/10/0/10 -Jx1 -O >> example_4.ps
```

The purpose of the color palette file zero.cpt is to have the positive topography mesh painted light gray (the remainder is white). Figure 4 shows the complete illustration.

A color version of this figure was used in our article in EOS Trans. AGU (Oct. 8th, 1991). It was created along similar lines, but instead of a mesh plot we chose a color-coded surface with artificial illumination from a light-source due north. We choose to use the **–Qi** option in *grdview* to achieve a high degree of smoothness. Here, we select dpi = 100 since that will be the resolution of our final raster (The EOS raster was 300 dpi). We used *grdgradient* to provide the intensity files. The following script creates the color *PostScript* file. Note that the size of the resulting output file is directly dependent on the square of the dpi chosen for the scanline conversion. A higher value for dpi in **–Qi** would have resulted in a much larger output file.

```
#!/bin/csh
# 3-D perspective color plot of Hawaiian topography and geoid
#
grdgradient geoid.grd -A0 -Gg_intens.grd -Nt0.5 -M -V
grdgradient topo.grd -A0 -Gt_intens.grd -Nt0.5 -M -V
grdview geoid.grd -Ig_intens.grd -Jm0.45 -E60/30 -R195/210/18/25 -Cgeoid.cpt -Qi100 -V -K -X1.5 -Y1.25 -P -
U/-1.25/-1/"Example 4c in Cookbook" >! example_4c.ps
pscoast -Jm -E60/30 -R -V -B2/2NEsw -G0 -O -K >> example_4c.ps
echo '205 26 0 0 1.1' | psxyz -Jm -E60/30 -R -SV0.2/0.5/0.4 -W4 -L -G255/0/0 -O -K -N >> example_4c.ps
echo '205 29.2 36 -90 1 5 N' | pstext -Jm -E60/30 -R -V -O -K -N >> example_4c.ps
grdview topo.grd -It_intens.grd -Jm -Jz0.34 -Ctopo.cpt -E60/30 -R195/210/18/25/-6/4 -N-6/200/200/200 -Qi100 - O
-K -V -Y2.2 >> example_4c.ps
psbasemap -Jm -Jz0.34 -E60/30 -R -Z-6 -O -K -V -B2/2/2:"Topo (km)":neZ >> example_4c.ps
echo '3.25 5.75 60 0.0 33 2 H@#awaiian@# R@#idge' | pstext -R0/10/0/10 -Jx1 -O >> example_4c.ps
```

The color palette files topo.cpt and geoid.cpt are the ones used in Example 2.

- Example 5: A 3-D illuminated surface in black and white.

Instead of a mesh plot we may choose to show 3-D surfaces using artificial illumination. For this example we will use *grdmath* to make a grdfile that contains the surface given by the function $z(x, y) = \cos(2\pi r/8)\cdot e^{-r/10}$, where $r^2 = (x^2 + y^2)$. The illumination is obtained by passing two grdfiles to *grdview*: One with the *z*-values (the surface) and another with intensity values (which should be in the ±1 range). We use *grdgradient* to compute the horizontal gradients in the direction of the artificial light source. The gradients are then given a normal distribution using *grdhisteq*. Note that we use native binary output from *grdgradient* so that we can pipe the results directly into *grdhisteq*. The gray.cpt file only has one line that states that all *z* values should have the gray level 128.

Thus, variations in shade are entirely due to variations in gradients, or illuminations. We choose to illuminate from the SW and view the surface from SE:

```
#!/bin/csh
grdmath -R-15/15/-15/15 -I0.3 -V X Y HYPOT DUP 2 MUL PI MUL 8 DIV COS EXCH NEG 10 DIV EXP MUL
= sombrero.grd
echo '-5 128 5 128' >! gray.cpt
grdgradient sombrero.grd -A225 -G=1 -N -V | grdhisteq =1 -Gintensity.grd -N -V
grdview sombrero.grd -JX6 -JZ2 -B5/5/0.5SEwnZ -N-1/255/255/255 -Qs -Iintensity.grd -X1.5 -Cgray.cpt -R-
15/15/-15/15/-1/1 -K -V -E120/30 -U/-1.25/-0.75/"Example 5 in Cookbook" >! example_5.ps
echo '4.1 5.5 50 0 33 2 z(r) = cos (2@~p@~r/8) * e@+-r/10@+' | pstext -R0/11/0/8.5 -Jx1 -O >> example_5.ps
```

The variations in intensity could be made more dramatic by using *grdmath* to scale the intensity file before running *grdview*. The shell-script will result in a plot like the one in Figure 5.

- Example 6: Plotting of histograms.

GMT provides two tools to render histograms: *pshistogram* and *psrose*. The former takes care of regular histograms whereas the latter deals with polar histograms (rose diagrams, sector diagrams, and windrose diagrams). We will show an example that involves both programs. The file `fractures.yx` contains a compilation of fracture lengths and directions as digitized from geological maps. The file `v3206.t` contains all the bathymetry measurements from *Vema* cruise 3206. Our complete figure (Figure 6) was made running this script:

```
#!/bin/csh
psrose fractures.d -A10r -S1.8n -U/-2.25/-0.75/"Example 6 in Cookbook" -V -P -G0 -R0/1/0/360 -X2.5 -K -
B0.2g0.2/30g30 >! example_6.ps
pshistogram -Ba2000f1000:"Topography (m)":/a10f5:"Frequency (%)"::."Two types of histograms":WSne
v3206.t -R-6000/0/0/30 -JX4.8/2.4 -G200 -O -Y5.5 -X-0.5 -L2 -Z1 -W250 >> example_6.ps
```

- Example 7: A simple location map.

Many scientific papers start out by showing a location map of the region of interest. This map will typically also contain certain features and labels. This example will present a location map for the equatorial Atlantic ocean, where fracture zones and mid-ocean ridge segments have been plotted. We also would like earthquake locations plotted and available isochrons. We have obtained one file, `quakes.xym`, which contains the position and magnitude of available earthquakes in the region. We choose to use magnitude/100 for the symbol-size in inches. The digital fracture zone traces (`fz.xy`) and isochrons (0 isochron as `ridge.xy`, the rest as `isochrons.xy`) were digitized from available maps[†]. We create the final location map (Figure 7) with the following script:

```
#!/bin/csh
pscoast -R-50/0/-10/20 -JM9 -K -GP0/26 -Dl -W1 -V -B10 -U"Example 7 in Cookbook" >! example_7.ps
psxy -R -JM -O -K -M fz.xy -W2ta -V >> example_7.ps
awk '{print $1-360.0, $2, $3*0.01}' quakes.xym | psxy -R -JM -O -K -H -Sc -G255 -L -W1 -V >> example_7.ps
psxy -R -JM -O -K -M isochron.xy -W3 -V >> example_7.ps
psxy -R -JM -O -K -M ridge.xy -W7 -V >> example_7.ps
psxy -R -JM -O -K -G255 -W4 -A -V << END >> example_7.ps
 -14.5      15.2
-2          15.2
-2          17.8
-14.5       17.8
```

[†] These data are available on CD-ROM from NGDC (contact pws@mail.ngdc.noaa.gov)

```
END
psxy -R -JM -O -K -G255 -W2 -A -V << END >> example_7.ps
-14.35     15.35
 -2.15     15.35
-2.15      17.65
-14.35     17.65
END
echo '-13.5 16.5' |psxy -R -JM -O -K -Sc0.08 -G255 -W2 -L -V >> example_7.ps
echo '-12.5 16.5 18 0 6 5 ISC Earthquakes' | pstext -R -JM -O -K -V >> example_7.ps
pstext -R -JM -O -S3 -G255 -V << END >> example_7.ps
-43 -5 30 0 1 6 SOUTH
-43 -8 30 0 1 6 AMERICA
-7 11 30 0 1 6 AFRICA
END
```

The same figure could equally well be made in color, which could be rasterized and made into a slide for a meeting presentation. The script is similar to the one outlined above, except we would choose a color for land and oceans, and select colored symbols and pens rather than black and white.

- Example 8: A 3-D histogram.

The program *psxyz* allows us to plot three-dimensional symbols, including columnar plots. As a simple demonstration, we will convert a gridded netCDF of bathymetry into an ASCII xyz table and use the height information to draw a 2-D histogram in a 3-D perspective view. Our gridded bathymetry file is called topo.grd and covers the region from 0 to 5 ˚E and 0 to 5 ˚N. Depth ranges from -5000 meter to sea-level. We produce the illustration by running this command:

```
#!/bin/csh
grd2xyz topo.grd | psxyz -B1/1/1000:"Topography (m)"::.ETOPO5:WSneZ+ -R-0.1/5.1/-0.1/5.1/-5000/0 -JM5 -
JZ5 -E200/30 -So0.0833333b-5000 -P -U"Example 8 in Cookbook" -L -G240 -K >! example_8.ps
echo '0.1 4.9 24 0 1 9 This is the surface of cube' | pstext -R -JM -JZ -Z0 -E200/30 -O >> example_8.ps
```

The output can be viewed in Figure 8.

- Example 9: Plotting time-series along tracks.

A common application in many scientific disciplines involves plotting one or several time-series as “wiggles” along tracks. Marine geophysicists often display magnetic anomalies in this manner, and seismologists use the technique when plotting individual seismic traces. In our example we will show how a set of Geosat sea surface slope profiles from the south Pacific can be plotted as “wiggles” using the *pswiggle* program. We will embellish the plot with track numbers, the location of the Pacific-Antarctic Ridge, recognized fracture zones in the area, and a “wiggle” scale. The Geosat tracks are stored in the files *.xys, the ridge in ridge.xy, and all the fracture zones are stored in the multiple segment file fz.xy. We extract the profile id and the last point in each of the track files to construct an input file for *pstext* that will label each profile with the track number. We know the profiles trend approximately N40˚E so we want the labels to have that same orientation. We do this by extracting the last record from each track, paste this file with the tracks.lis file, and use *awk* to create the format needed for *pstext*. Note we offset the positions by 0.05 inch in order to have a small gap between the profile and the label:

```
#!/bin/csh
pswiggle *.xys -R185/250/-68/-42 -U"Example 9 in Cookbook" -K -Jm0.13 -Ba10f5 -G0 -Z2000 -W1 -S240/-
67/500/@~m@~rad >! example_9.ps
```

```
psxy -R -Jm -O -K ridge.xy -W5 >> example_9.ps
psxy -R -Jm -O -K -M fz.xy -W2ta >> example_9.ps
\rm -f tmp
foreach file (*.xys)                    # Make label file
   tail -1 $file | mapproject -R -Jm >>! tmp
end
ls *.xys | awk -F. '{print $2}' >! tracks.lis
paste tmp tracks.lis | awk '{print $1-0.05, $2-0.05, 10, 50, 1, 7, $4}' | pstext -R0/10/0/10 -Jx1 -O >>
example_9.ps
```

The output shows the sea-surface slopes along 42 descending Geosat tracks in the Eltanin and Udintsev fracture zone region in a Mercator projection (Figure 9).

- Example 10: A geographical bar graph plot.

Our next and perhaps silliest example presents a three-dimensional bargraph plot showing the geographic distribution of the membership in the American Geophysical Union (AGU). The input data was taken from the 1991 AGU member directory and added up to give total members per continent. We decide to plot a 3-D column centered on each continent with a height that is proportional to the logarithm of the membership. A $\log_{10}$ -scale is used since the memberships vary by almost 3 orders of magnitude. We choose a plain linear projection for the basemap and add the columns and text on top. Our script reads:

```
#!/bin/csh
pscoast -R-180/180/-90/90 -JX8/5d -G0 -E200/40 -V -K -U"Example 10 in Cookbook" >! example_10.ps
psxyz agu.d -R-180/180/-90/90/1/100000 -JX8/5 -JZ2.5l -So0.3b1 -G140 -L -W2 -
B60g60/30g30/a1p:Memberships:WSneZ -O -K -E200/40 >> example_10.ps
awk '{print $1-10, $2, 20, 0, 0, 7, $3}' agu.d | pstext -R -JX -O -K -E200/40 -G255 -S2 >> example_10.ps
echo 3244.5 6 30 0 5 2 AGU Membership Distribution325 | pstext -R0/11/0/8.5 -Jx1 -O >> example_10.ps
```

- Example 11: Making a 3-D RGB color cube.

In this example we generate a series of 6 color images, arranged in the shape of a cross, that can be cut out and assembled into a 3-D color cube. The six faces of the cube represent the outside of the R-G-B color space. On each face one of the color components is fixed at either 0 or 255 and the other two components vary smoothly across the face from 0 to 255. The cube is configured as a right-handed coordinate system with *x*-*y*-*z* mapping R-G-B. Hence, the 8 corners of the cube represent the primaries red, green, and blue, plus the secondaries cyan, magenta and yellow, plus black and white.

The method for generating the 6 color faces utilizes *awk* in two steps. First, a *z*-grid is composed which is 256 by 256 with *z*-values increasing in a planar fashion from 0 to 65535. This *z*-grid is common to all six faces. The color variations are generated by creating a different color palette for each face using the supplied *awk* script rgb_cube.awk. This script generates a "cpt" file appropriate for each face using arguments for each of the three color components. The arguments specify if that component (r,g,b) is to be held fixed at 0 or 255, is to vary in *x*, or is to vary in *y*. If the color is to increase in *x* or *y*, a lower case *x* or *y* is specified; if the color is to decrease in *x* or *y*, an upper case *X* or *Y* is used. Here is the shell script and accompanying *awk* script to generate the RGB cube:

```
#!/bin/csh
#
# Example 11: Create a 3-D RGB Cube
#
# First create a Plane from (0,0,0) to (255,255,255).
# Only needs to be done once, and is used on each of the 6 faces of the cube.
```

```
#

echo "Creating z-grid..."

grdmath -I1 -R0/255/0/255 -V Y 256 MUL X ADD = rgb_cube.grd

#
# For each of the 6 faces, create a color palette with one color (r,g,b) fixed at either the min.
# of 0 or max. of 255, and the other two components varying smoothly across the face from
# 0 to 255.
#
# This uses awk script "rgb_cube.awk", with arguments specifying which color (r,g,b) is held
# constant at 0 or 255, which color varies in the x-direction of the face, and which color varies in
# the y-direction.  If the color is to increase in x (y), a lower case x (y) is indicated; if the color is
# to decrease in the x (y) direction, an upper case X (Y) is used.
#
# Use grdimage to paint the faces and psxy to add "cut-along-the-dotted" lines.
#

gmtset TICK_LENGTH 0

pstext -R0/8/0/11 -Jx1 < /dev/null -P -U"Example 11 in Cookbook" -K >! example_11.ps
echo -n "Top..."
awk -f rgb_cube.awk r=x g=y b=255 < /dev/null >! rgb_cube.cpt
grdimage rgb_cube.grd -Crgb_cube.cpt -JX2.56/2.56 -R0/255/0/255 -K -O -X1.97 -Y4.5 -B256wesn >>
example_11.ps

echo -n " Right..."
awk -f rgb_cube.awk r=255 g=y b=X < /dev/null >! rgb_cube.cpt
grdimage rgb_cube.grd -Crgb_cube.cpt -JX -K -O -X2.56 -B256wesn >> example_11.ps

echo -n " Back"
awk -f rgb_cube.awk r=x g=255 b=Y < /dev/null >! rgb_cube.cpt
grdimage rgb_cube.grd -Crgb_cube.cpt -JX -K -O -X-2.56 -Y2.56 -B256wesn >> example_11.ps

psxy -W1to -JX -R -K -O -X2.56 << END >> example_11.ps
0 0
20 20
20 235
0 255
END

psxy -W1to -JX -R -K -O -X-2.56 -Y2.56 << END >> example_11.ps
0 0
20 20
235 20
 255 0
END

psxy -W1to -JX -R -K -O -X-2.56 -Y-2.56 << END >> example_11.ps
255 0
235 20
235 235
255 255
END

echo -n " Left"
awk -f rgb_cube.awk r=0 g=y b=x < /dev/null >! rgb_cube.cpt
grdimage rgb_cube.grd -Crgb_cube.cpt -JX -K -O -Y-2.56 -B256wesn >> example_11.ps

echo -n " Front"
awk -f rgb_cube.awk r=x g=0 b=y < /dev/null >! rgb_cube.cpt
grdimage rgb_cube.grd -Crgb_cube.cpt -JX -K -O -X2.56 -Y-2.56 -B256wesn >> example_11.ps
```

```
pstext -JX -R -G255 -K -O << END >> example_11.ps
10 10 14 0 -Times-BoldItalic 1 GMT 3.0
END

psxy -W1to -JX -R -K -O -X2.56 << END >> example_11.ps
0 0
20 20
20 235
0 255
END

psxy -W1to -JX -R -K -O -X-5.12 << END >> example_11.ps
255 0
235 20
235 235
255 255
END

echo -n " Bottom"
awk -f rgb_cube.awk r=x g=Y b=0 < /dev/null >! rgb_cube.cpt
grdimage rgb_cube.grd -Crgb_cube.cpt -JX -K -O -X2.56 -Y-2.56 -B256wesn >> example_11.ps

psxy -W1to -JX -R -K -O -X2.56 << END >> example_11.ps
0 0
20 20
20 235
0 255
END

psxy -W1to -JX -R -O -X-5.12 << END >> example_11.ps
255 0
235 20
235 235
255 255
END

echo " Done!"
```

The *awk* script rgb_cube.awk is as follows:

```
END{
	z=-.5;

	if(r=="X" || g=="X" || b=="X"){
		xl=255; xr=0; xd=-255;
	}else{
		xl=0; xr=255; xd=255;
	}

	if(r=="Y" || g=="Y" || b=="Y"){
		yb=255; yt=-1; yd=-1;
	}else{
		yb=0; yt=256; yd=1;
	}
	for(y=yb; y!=yt; y+=yd){
		x=xl;

		if(r=="x" || r=="X"){
			if(g=="y" || g=="Y"){
				printf("%7.1f %3d %3d %3d " ,z,x,y,b);
				x+=xd; z+=256;
				printf("%7.1f %3d %3d %3dn",z,x,y,b);
			}else{
```

```
                printf("%7.1f %3d %3d %3d " ,z,x,g,y);
                x+=xd; z+=256;
                printf("%7.1f %3d %3d %3dn",z,x,g,y);
            }

        }else if(g=="x" || g=="X"){
            if(r=="y" || r=="Y"){
                printf("%7.1f %3d %3d %3d " ,z,y,x,b);
                x+=xd; z+=256;
                printf("%7.1f %3d %3d %3dn",z,y,x,b);
            }else{
                printf("%7.1f %3d %3d %3d " ,z,r,x,y);
                x+=xd; z+=256;
                printf(" %7.1f %3d %3d %3dn",z,r,x,y);
            }

        }else{
            if(r=="y" || r=="Y"){
                printf("%7.1f %3d %3d %3d " ,z,y,g,x);
                x+=xd; z+=256;
                printf("%7.1f %3d %3d %3dn",z,y,g,x);
            }else{
                printf("%7.1f %3d %3d %3d " ,z,r,y,x);
                x+=xd; z+=256;
                printf("%7.1f %3d %3d %3dn",z,r,y,x);
            }
        }
    }
    exit;
}
```

- Example 12: Optimal triangulation of data.

Our next example operates on a data set of topographic readings non-uniformly distributed in the plane (Table 5.11 in Davis: *Statistics and Data Analysis in Geology*, J. Wiley). We use *triangulate* to perform the optimal Delaunay triangulation, then use the output to draw the resulting network. We label the node numbers as well as the node values, and call *pscontour* to make a contour map and image directly from the raw data. We use a color palette table topo.cpt (supplied with the script data separately). The script becomes:

```
#!/bin/csh
# First draw network and label the nodes
triangulate table_5.11 -M >! net.xy
psxy -R0/6.5/-0.2/6.5 -JX3.4/3.5 -B2f1WSNe -M net.xy -W2 -P -K -Y5 >! example_12.ps
awk '{print $1, $2}' table_5.11 | psxy -R -JX -O -K -Sc0.12 -G255 -L >> example_12.ps
awk '{print $1, $2, 6, 0, 0, 6, NR-1}' table_5.11 | pstext -R -JX -O -K >> example_12.ps
# Then draw network and print the node values
psxy -R -JX -B2f1eSNw -M net.xy -W2 -O -K -X3.6 >> example_12.ps
psxy -R -JX -O -K table_5.11 -Sc0.03 -G0 >> example_12.ps
awk '{printf "%lg %s 6 0 0 5 %lgn", $1+0.08, $2, $3}' table_5.11 | pstext -R -JX -O -K -W255o -C0.01/0.01 -N
>> example_12.ps
# Then contour the data and draw triangles using dashed pen
pscontour -R -JX table_5.11 -B2f1WSne -W3 -Ctopo.cpt -X-3.6 -Y-4 -O -K -U"Example 12 in Cookbook" >>
example_12.ps
psxy -R -JX net.xy -W1ta -M -O -K >> example_12.ps
# Finally color the topography
pscontour -R -JX table_5.11 -B2f1eSnw -Ctopo.cpt -G -X3.6 -O -K >> example_12.ps
echo '3.5 8.4 30 0 1 2 Delaunay Triangulation' | pstext -R0/8/0/11 -Jx1 -O -X-3.6 >> example_12.ps
```

- Example 13: Plotting of vector fields.

In many areas, such as fluid dynamics and elasticity, it is desirable to plot vector fields of various kinds. GMT provides a way to illustrate 2-component vector fields using the *grdvector* utility. The two components of the field (Cartesian or polar components) are stored in separate grdfiles. In this example we generate a surface $z(x, y) = x \cdot \exp(-x^2 - y^2)$ and use *grdgradient* to calculate grad (z) by returning the x- and y- derivatives separately. We superpose the gradient vector field and the surface z and also plot the components of the gradient in separate windows:

```
#!/bin/csh
#
grdmath -R-2/2/-2/2 -I0.1 -V X Y R2 NEG EXP X MUL = z.grd

grdgradient z.grd -A270 -Gdzdx.grd
grdgradient z.grd -A180 -Gdzdy.grd
grdcontour dzdx.grd -JX3 -B1/1WSne -C0.1 -A0.5 -K -P -G2/10 -S4 -T0.1/0.03 -U"Example 13 in Cookbook" >!
example_13.ps
grdcontour dzdy.grd -JX3 -B1/1WSne -C0.05 -A0.2 -O -K -G2/10 -S4 -T0.1/0.03 -X3.45 >> example_13.ps
grdcontour z.grd -JX3 -B1/1WSne -C0.05 -A0.1 -O -K -G2/10 -S4 -T0.1/0.03 -X-3.45 -Y3.45 >> example_13.ps
grdcontour z.grd -JX3 -B1/1WSne -C0.05 -O -K -G2/10 -S4 -X3.5 >> example_13.ps
grdvector dzdx.grd dzdy.grd -I0.2 -JX3 -O -K -Q0.03/0.10/0.09n0.25 -G0 -S5 >> example_13.ps
echo '3.2 3.6 40 0 6 2 z(x,y) = x * exp(-x@+2@+-y@+2@+)' | pstext -R0/8.5/0/5 -Jx1 -O -X-3.45 >>
example_13.ps
```

The angles 270˚ and 180˚ are used instead of the expected 90˚ and 0˚ because of the sign convention used in *grdgradient* (see its man page for details). A *pstext* call to place a header finishes the plot.

- Example 14: Gridding of data and trend surfaces.

This example shows how one goes from randomly spaced data points to an evenly sampled surface. First we plot the distribution and values of our raw data set (table 5.11 from example 12). We choose an equidistant grid and run *blockmean* which preprocesses the data to avoid aliasing. The dashed lines indicate the logical blocks used by *blockmean*; all points inside a given bin will be averaged. The processed data is then gridded with the *surface* program and contoured every 25 units. We use *grdtrend* to fit a bicubic trend surface to the gridded data, contour it as well, and sample both gridded files along a diagonal transect using *grdtrack*. The bottom panel compares the gridded (solid line) and bicubic trend (dashed line) along the transect:

```
#!/bin/csh
# First plot data and label the nodes
psxy table_5.11 -R0/7/0/7 -JX3.4/3.5 -B2f1WSNe -Sc0.05 -G0 -P -K -Y6.9 >! example_14.ps
awk '{printf "%lg %s 6 0 0 5 %lgn", $1+0.08, $2, $3}' table_5.11 | pstext -R -JX -O -K -N >> example_14.ps
blockmean table_5.11 -R0/7/0/7 -I1 >! mean.xyz
# Then draw blocmean cells
\rm -f tmp
foreach x (0.5 1.5 2.5 3.5 4.5 5.5 6.5)
   (echo '> new line'; echo $x 0; echo $x 7) >>! tmp
   (echo '> new line'; echo 0 $x; echo 7 $x) >> tmp
end
psxy -R -JX -B2f1eSNw -M tmp -W1ta -O -K -X3.6 >> example_14.ps
psxy -R -JX -B2f1eSNw mean.xyz -Ss0.05 -G0 -O -K >> example_14.ps
awk '{printf "%lg %s 6 0 0 5 %lgn", $1+0.1, $2, $3}' mean.xyz | pstext -R -JX -O -K -W255o -C0.01/0.01 -N >>
example_14.ps
# Then surface and contour the data
surface mean.xyz -R -I1 -Gdata.grd -V
```

```
grdcontour data.grd -JX -B2f1WSne -C25 -A50 -G3/10 -S4 -O -K -X-3.6 -Y-3.9 -V >> example_14.ps
psxy -R -JX mean.xyz -Ss0.05 -G0 -O -K >> example_14.ps
# Fit bicubic trend to data and compare to gridded surface
grdtrend data.grd -N10 -Ttrend.grd -V
grdcontour trend.grd -JX -B2f1wSne -C25 -A50 -G3/10 -S4 -O -K -X3.6 >> example_14.ps
project -C0/0 -E7/7 -G0.1 >! track
psxy -R -JX track -W4to -O -K >> example_14.ps
# Sample along diagonal
grdtrack track -Gdata.grd | cut -f3,4 >! data.d
grdtrack track -Gtrend.grd | cut -f3,4 >! trend.d
psxy `minmax data.d trend.d -I0.5/25` -JX7/1.5 data.d -W4 -O -K -X-3.6 -Y-2 -B1/50WSne >> example_14.ps
psxy -R -JX trend.d -W2ta -O -U"Example 14 in Cookbook" >> example_14.ps
```

7. Mailing lists, updates, and bug reports.

Most public-domain (and even commercial) software comes with bugs, and the speed with which such bugs are detected and removed depends to a large degree on the willingness of the user community to report these to us in a useful manner. When your car breaks down, simply telling the mechanic that it doesn't work will hardly speed up the repair or cut back costs! Therefore, we ask that if you detect a bug, first make sure that it in fact is a bug and not a user error. Then, send us email about the problem. Be sure to include all the information necessary for us to recreate the situation in which the bug occurred. This will include the full command line used and, if possible, the data file used by the program. Send the bug-reports to gmt@soest.hawaii.edu. We will try to fix bugs as soon as our schedules permit and inform users about the bug and availability of updated code (See Appendix D).

With the distribution you will find two license registration forms; one for non-profit organizations and one for for-profit companies. We ask that you fill out the appropriate version and send it to us. There is no need to fill out a new license if you already are a registered user of GMT v2.x.

Two electronic mailing lists are available for users to subscribe to. The first is called gmtgroup@soest.hawaii.edu and is primarily a way for us to notify the users when bugs have been fixed or when new updates have been installed in the ftp directory (See Appendix D). We also maintain another list (gmthelp@soest.hawaii.edu) which interested users may subscribe to. It basically provides a forum for GMT users to exchange ideas and ask questions about GMT usage, installation and portability, etc. Please use this utility rather than sending questions directly to gmt@soest. We hope you appreciate that we simply do not have time to be everybody's personal GMT tutor.

The electronic mailing lists are maintained automatically by a program. To subscribe to one or both of the lists, send a message to listserver@soest.hawaii.edu containing the command(s):

subscribe gmtgroup <your full name, not email address>
subscribe gmthelp <your full name, not email address>

(Do not type the angular brackets <>). For information on what commands you may send, send a message containing the word help. You must interact with the listserver to be added to the mailing lists! We strongly recommend that you at least subscribe to gmtgroup since this is how we can notify you of future updates and bug-fixes. Most new users will also benefit from having the other forum (gmthelp) as they struggle to realign their sense of logic with that of GMT. While anybody may post messages to gmthelp, access to gmtgroup is restricted to minimize net traffic. Any message sent to gmtgroup will be intercepted by the GMT manager who will determine if the message is important enough to cause thousands of mailtools to go BEEP. Communication with other GMT users should go via gmthelp.