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 cshell script that contains all the commands used to arrive at the final illustration, and (3) we explain the rationale behind the commands. 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 (both cshell and bash scripts; only the cshell scripts are discussed here) 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!). The examples are all written to be "quiet", that is no information is echoed to the screen. Thus, these scripts are well suited for background execution. Note that we also end each script by cleaning up after ourselves. Because *awk* is broken as designed on some systems, and *nawk* is not avaialble on others we refer to $AWK in the scripts below; the do_examples scripts will set this when running all examples.

· 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 will see how to make gridded files later). 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 cshell script.

```
#!/bin/csh
#       GMT EXAMPLE 01
#
#       @(#)job01.csh     1.1  09/21/98
#
# Purpose:      Make two contour maps based on the data in the file osu91a1f_16.grd
# GMT progs:    gmtset grdcontour psbasemap pscoast
# Unix progs:   rm
#
gmtset GRID_CROSS_SIZE 0 ANOT_FONT_SIZE 10
psbasemap -R0/6.5/0/9 -Jx1 -B0 -P -K -U"Example 1 in Cookbook" >! example_01.ps
pscoast -R-180/180/-90/90 -JH0/6 -X0.25 -Y0.5 -O -K -Bg30 -Dc -G200 >> example_01.ps
grdcontour -R osu91a1f_16.grd -JH -C10 -A50f7 -L-1000/-1 -Wc1ta -Wa3t8_8:0 -O -K -T0.1/0.02 >>
example_01.ps
grdcontour -R osu91a1f_16.grd -JH -C10 -A50f7 -L-1/1000 -O -K -T0.1/0.02 >> example_01.ps
pscoast -R0/360/-90/90 -JH180/6 -Y4 -O -K -Bg30:."Low Order Geoid": -Dc -G200 >> example_01.ps
grdcontour osu91a1f_16.grd -JH -C10 -A50f7 -L-1000/-1 -Wc1ta -Wa3t8_8:0 -O -K -T0.1/0.02:-+ >>
example_01.ps
grdcontour osu91a1f_16.grd -JH -C10 -A50f7 -L-1/1000 -O -T0.1/0.02:-+ >> example_01.ps
\rm -f .gmtcommands
```

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 (again, we will deferr the actual making of gridded files to later examples). 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
#       GMT EXAMPLE 02
#
#       @(#)job02.csh     1.1  09/21/98
#
# Purpose:      Make two color images based gridded data
# GMT progs:   gmtset grd2cpt grdgradient grdimage makecpt psscale pstext
# Unix progs:   cat rm
#
gmtset HEADER_FONT_SIZE 30 OBLIQUE_ANOTATION 0 DEGREE_FORMAT 0
#get gridded data using GMT supplemental program grdraster
#grdraster 1 -R160/20/220/30r -JOc190/25.5/292/69/4.5 -Getopo5.grd=0/0.001/0
#grdraster 4 -R -JO -Ggeosat5.grd
makecpt -C1 -S16c -M6 >! g.cpt
grdimage geosat5.grd -R160/20/220/30r -JOc190/25.5/292/69/4.5 -E50 -K -P -U/-1.25/-1/"Example 2 in
Cookbook" -B10 -Cg.cpt -X1.5 -Y1.25 >! example_02.ps
psscale -Cg.cpt -D5.1/1.35/2.88/0.4 -O -K -L -B2:m::.GEOID: -E >> example_02.ps
grd2cpt etopo5.grd >! t.cpt
grdgradient etopo5.grd -A0 -Nt -Getopo5_int.grd
grdimage etopo5.grd -Ietopo5_int.grd -R -JO -E50 -B10:."H@#awaiian@# T@#opo and @#G@#eoid:" -O -K -
Ct.cpt -Y4.5 >> example_02.ps
psscale -Ct.cpt -D5.1/1.35/2.88/0.4 -O -K -I0.3 -B2:m::.TOPO: >> example_02.ps
cat << EOF | pstext -R0/8.5/0/11 -Jx1 -O -N -Y-4.5 >> example_02.ps
-0.4 7.5 30 0.0 1 2 a)
-0.4 3.0 30 0.0 1 2 b)
EOF
\rm -f .gmtcommands etopo5_int.grd ?.cpt
```

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. 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*. For your convenience we have commented out those lines and provided the two extracted files so you do not need *grdraster* to try this example. 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. We us *makecpt* to generate a linear color palette file geoid.cpt for the geoid and use *grd2cpt* to get a histogram-equalized cpt file topo.cpt for the topography data. To emphasize the structures in the data we calculate the slopes in the north-south direction using *grdgradient*;

these will be used to modulate the color image. 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*. We also annotate the color scales 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. There are several intermediate steps that produce helpful plots showing the effect of the various processing steps (example_3a-f), while the final plot example_03.ps shows 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
#       GMT EXAMPLE 03
#
#       @(#)job03.csh     1.2  09/24/98
#
# Purpose:      Resample track data, do spectral analysis, and plot
# GMT progs:    filter1d, fitcircle, gmtset, project, sample1d, spectrum1d,
#       trend1d, pshistogram, psxy, pstext
# Unix progs:   $AWK, cat, echo, head, paste, rm, tail
#
# This example begins with data files "ship.xyg" and "sat.xyg" which
# are measurements of a quantity "g" (a "gravity anomaly" which is an
# anomalous increase or decrease in the magnitude of the acceleration
# of gravity at sea level).  g is measured at a sequence of points "x,y"
# which in this case are "longitude,latitude".  The "sat.xyg" data were
# obtained by a satellite and the sequence of points lies almost along
# a great circle.  The "ship.xyg" data were obtained by a ship which
# tried to follow the satellite's path but deviated from it in places.
# Thus the two data sets are not measured at the same set of points,
# and we use various GMT tools to facilitate their comparison.
# The main illustration (example_03.ps) are accompanied with 5 support
# plots (03a-f) showing data distributions and various intermediate steps.
#
# First, we use "fitcircle" to find the parameters of a great circle
# most closely fitting the x,y points in "sat.xyg":
#
fitcircle sat.xyg -L2 >! report
set cpos = `grep "L2 Average Position" report`
set ppos = `grep "L2 N Hemisphere" report`
```

```
#
# Now we use "project" to project the data in both sat.xyg and ship.xyg
# into data.pg, where g is the same and p is the oblique longitude around
# the great circle.  We use -Q to get the p distance in kilometers, and -S
# to sort the output into increasing p values.
#
project  sat.xyg -C$cpos[1]/$cpos[2] -T$ppos[1]/$ppos[2] -S -Fpz -Q >! sat.pg
project ship.xyg -C$cpos[1]/$cpos[2] -T$ppos[1]/$ppos[2] -S -Fpz -Q >! ship.pg
#
# The minmax utility will report the minimum and maximum values for all columns.
# We use this information first with a large -I value to find the appropriate -R
# to use to plot the .pg data.
#
set plotr = `minmax -I100/25 -C sat.pg ship.pg`
gmtset MEASURE_UNIT INCH
psxy -R$plotr[1]/$plotr[2]/$plotr[3]/$plotr[4] -JX8/5 -Ba500f100:"Distance along great
circle":/a100f25:"Gravity anomaly (mGal)":WeSn -U/-1.75/-1.25/"Example 3a in Cookbook" -X2 -Y1.5 -K -
W1p sat.pg >! example_03a.ps
psxy -R -JX -O -Sp0.03 ship.pg >> example_03a.ps
#
# From this plot we see that the ship data have some "spikes" and also greatly
# differ from the satellite data at a point about p ~= +250 km, where both of
# them show a very large anomaly.
#
# To facilitate comparison of the two with a cross-spectral analysis using "spectrum1d",
# we resample both data sets at intervals of 1 km.  First we find out how the data are
# typically spaced using $AWK to get the delta-p between points and view it with
# "pshistogram".
#
$AWK '{ if (NR > 1) print $1 - last1; last1 = $1; }' ship.pg | pshistogram  -W0.1 -G0 -JX3 -K -X2 -Y1.5 -
B:."Ship": -U/-1.75/-1.25/"Example 3b in Cookbook" >! example_03b.ps
$AWK '{ if (NR > 1) print $1 - last1; last1 = $1; }' sat.pg  | pshistogram  -W0.1 -G0 -JX3 -O -X5 -B:."Sat": >>
example_03b.ps
#
# This experience shows that the satellite values are spaced fairly evenly, with
# delta-p between 3.222 and 3.418.  The ship values are spaced quite unevelnly, with
# delta-p between 0.095 and 9.017.  This means that when we want 1 km even sampling,
# we can use "sample1d" to interpolate the sat data, but the same procedure applied
# to the ship data could alias information at shorter wavelengths.  So we have to use
# "filter1d" to resample the ship data.  Also, since we observed spikes in the ship
# data, we use a median filter to clean up the ship values.  We will want to use "paste"
# to put the two sampled data sets together, so they must start and end at the same
# point, without NaNs.  So we want to get a starting and ending point which works for
# both of them.  Thus we need to start at max( min(ship.p), min(sat.p) ) and end
# conversely.  "minmax" can't do this easily since it will return min( min(), min() ),
# so we do a little head, paste $AWK to get what we want.
#
head -1 ship.pg >! ship.pg.extr
head -1 sat.pg >! sat.pg.extr
paste ship.pg.extr sat.pg.extr | $AWK '{ if ($1 > $3) print int($1); else print int($3); }' >! sampr1
tail -1 ship.pg >! ship.pg.extr
tail -1 sat.pg >! sat.pg.extr
paste ship.pg.extr sat.pg.extr | $AWK '{ if ($1 < $3) print int($1); else print int($3); }' >! sampr2
set sampr = `paste sampr1 sampr2`
#
# Now we can use sampr in $AWK to make a sampling points file for sample1d:
$AWK 'BEGIN { for (i = '$sampr[1]'; i <= '$sampr[2]'; i++) print i }' /dev/null >! samp.x
#
# Now we can resample the projected satellite data:
#
sample1d sat.pg -Nsamp.x >! samp_sat.pg
#
# For reasons above, we use filter1d to pre-treat the ship data.  We also need to sample it
# because of the gaps > 1 km we found.  So we use filter1d | sample1d.  We also use the -E
```

```
# on filter1d to use the data all the way out to sampr[1]/sampr[2] :
#
filter1d ship.pg -Fm1 -T$sampr[1]/$sampr[2]/1 -E | sample1d -Nsamp.x >! samp_ship.pg
#
# Now we plot them again to see if we have done the right thing:
#
psxy -R$plotr[1]/$plotr[2]/$plotr[3]/$plotr[4] -JX8/5 -Ba500f100:"Distance along great
circle":/a100f25:"Gravity anomaly (mGal)":WeSn -X2 -Y1.5 -K -W1p samp_sat.pg -U/-1.75/-1.25/"Example
3c in Cookbook" >! example_03c.ps
psxy -R -JX -O -Sp0.03 samp_ship.pg >> example_03c.ps
#
# Now to do the cross-spectra, assuming that the ship is the input and the sat is the output
# data, we do this:
#
paste samp_ship.pg samp_sat.pg | cut -f2,4 | spectrum1d -S256 -D1 -W -C >& /dev/null
#
# Now we want to plot the spectra.  The following 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 -Jx1 and specify positions in inches directly:
#
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 3d in Cookbook" -P -K -X2.5 -Sc0.07 -G0 -Ey/2 -Y1.5 >!
example_03.ps
echo "3.85 3.6 18 0.0 1 11 Coherency@+2@+" | pstext -R0/4/0/3.75 -Jx1 -O -K >> example_03.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_03.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_03.ps
psxy spectrum.ypower -R -JX -O -K -G0 -Sc0.07 -Ey/2 >> example_03.ps
echo "3.9 3.6 18 0.0 1 11 Input Power" | pstext -R0/4/0/3.75 -Jx1 -O -K >> example_03.ps
psxy -R -Jx -O -K -W5 box.d >> example_03.ps
psxy -R -Jx -O -K -G240 -L -W5 << END >> example_03.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_03.ps
echo "0.5 0.7 14 0.0 1 5 Ship" | pstext -R -Jx -O -K >> example_03.ps
echo "0.4 0.4" | psxy -R -Jx -O -K -Sc0.07 -G0 >> example_03.ps
echo "0.5 0.4 14 0.0 1 5 Satellite" | pstext -R -Jx -O >> example_03.ps
#
# Now we wonder if removing that large feature at 250 km would make any difference.
# We could throw away a section of data with $AWK or sed or head and tail, but we
# demonstrate the use of "trend1d" to identify outliers instead.  We will fit a
# straight line to the samp_ship.pg data by an iteratively-reweighted method and
# save the weights on output.  Then we will plot the weights and see how things
# look:
#
trend1d -Fxw -N2r samp_ship.pg >! samp_ship.xw
psxy -R$plotr[1]/$plotr[2]/$plotr[3]/$plotr[4] -JX8/4 -Ba500f100:"Distance along great
circle":/a100f25:"Gravity anomaly (mGal)":WeSn -U/-1.75/-1.25/"Example 3e in Cookbook" -X2 -Y1.5 -K -
Sp0.03 samp_ship.pg >! example_03d.ps
psxy -R$plotr[1]/$plotr[2]/0/1.1 -JX8/1.1 -O -Y4.25 -Bf100/a0.5f0.1:"Weight":Wesn -Sp0.03 samp_ship.xw
>> example_03d.ps
#
# From this we see that we might want to throw away values where w < 0.6.  So we try that,
# and this time we also use trend1d to return the residual from the model fit (the
# de-trended data):
```

```
trend1d -Fxrw -N2r samp_ship.pg | $AWK '{ if ($3 > 0.6) print $1, $2 }' | sample1d -Nsamp.x >!
samp2_ship.pg
trend1d -Fxrw -N2r samp_sat.pg  | $AWK '{ if ($3 > 0.6) print $1, $2 }' | sample1d -Nsamp.x >! samp2_sat.pg
#
# We plot these to see how they look:
#
set plotr = `minmax -I100/25 -C samp2_sat.pg samp2_ship.pg`
psxy -R$plotr[1]/$plotr[2]/$plotr[3]/$plotr[4] -JX8/5 -Ba500f100:"Distance along great
circle":/a50f25:"Gravity anomaly (mGal)":WeSn -U/-1.75/-1.25/"Example 3f in Cookbook" -X2 -Y1.5 -K -
W1p samp2_sat.pg >! example_03e.ps
psxy -R -JX -O -Sp0.03 samp2_ship.pg >> example_03e.ps
#
# Now we do the cross-spectral analysis again.  Comparing this plot (example_03f.ps) with
# the previous one (example_03.ps) we see that throwing out the large feature has reduced
# the power in both data sets and reduced the coherency at wavelengths between 20--60 km.
#
paste samp2_ship.pg samp2_sat.pg | cut -f2,4 | spectrum1d -S256 -D1 -W -C >& /dev/null
#
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 3g in Cookbook" -P -K -X2.5 -Sc0.07 -G0 -Ey/2 -Y1.5 >!
example_03f.ps
echo "3.85 3.6 18 0.0 1 11 Coherency@+2@+" | pstext -R0/4/0/3.75 -Jx1 -O -K >> example_03f.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_03f.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_03f.ps
psxy spectrum.ypower -R -JX -O -K -G0 -Sc0.07 -Ey/2 >> example_03f.ps
echo "3.9 3.6 18 0.0 1 11 Input Power" | pstext -R0/4/0/3.75 -Jx1 -O -K >> example_03f.ps
psxy -R -Jx -O -K -W5 box.d >> example_03f.ps
psxy -R -Jx -O -K -G240 -L -W5 << END >> example_03f.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_03f.ps
echo "0.5 0.7 14 0.0 1 5 Ship" | pstext -R -Jx -O -K >> example_03f.ps
echo "0.4 0.4" | psxy -R -Jx -O -K -Sc0.07 -G0 >> example_03f.ps
echo "0.5 0.4 14 0.0 1 5 Satellite" | pstext -R -Jx -O >> example_03f.ps
#
\rm -f box.d report samp* *.pg *.extr spectrum.* .gmtcommands
```

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 to do it:

```
#!/bin/csh
#       GMT EXAMPLE 04
#
#       @(#)job04.csh     1.2  09/21/98
#
# Purpose:      3-D mesh plot of Hawaiian topography and geoid
# GMT progs:   grdcontour, grdview, pscoast, pstext, psxyz
# Unix progs:   echo, rm
#
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 -K -P -X1.5 -Y1.5 -U/-1.25/-1.25/"Example 4
in Cookbook" >! example_04.ps
pscoast -Jm -E60/30 -R -B2/2NEsw -G0 -O -K >> example_04.ps
echo '205 26 0 0 1.1' | psxyz -Jm -E60/30 -R -SV0.2/0.5/0.4 -W4 -O -K -N >> example_04.ps
echo '205 29.2 36 -90 1 5 N' | pstext -Jm -E60/30 -R -O -K -N >> example_04.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 -
B2/2/2:"Topo (km)":neswZ -Y2.2 >> example_04.ps
echo '3.25 5.75 60 0.0 33 2 H@#awaiian@# R@#idge' | pstext -R0/10/0/10 -Jx1 -O >> example_04.ps
\rm -f zero.cpt
csh job4c.csh
```

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. The cpt files were taken from Example 2.

```
#!/bin/csh
#       GMT EXAMPLE 04
#
#       @(#)job4c.csh     1.1  09/21/98
#
# 3-D perspective color plot of Hawaiian topography and geoid
# GMT progs:   grdcontour, grdview, pscoast, pstext, psxyz
# Unix progs:   echo, rm
#
grdgradient geoid.grd -A0 -Gg_intens.grd -Nt0.75 -M
grdgradient topo.grd -A0 -Gt_intens.grd -Nt0.75 -M
#
grdview geoid.grd -Ig_intens.grd -JM6.75 -E60/30 -R195/210/18/25 -Cgeoid.cpt -Qi100 -K -X1.5 -Y1.25 -P -
U/-1.25/-1/"Example 4c in Cookbook" >! example_4c.ps
```

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

```
pscoast -JM -E60/30 -R -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 -G255/0/0 -O -K -N >> example_4c.ps
echo '205 29.2 36 -90 1 5 N' | pstext -JM -E60/30 -R -O -K -N >> example_4c.ps
grdview topo.grd -It_intens.grd -JM -JZ3.4 -Ctopo.cpt -E60/30 -R195/210/18/25/-6/4 -N-6/200/200/200 -
Qi100 -O -K -Y2.2 >> example_4c.ps
psbasemap -JM -JZ3.4 -E60/30 -R -Z-6 -O -K -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
\rm -f *_intens.grd .gmtcommands
```

· 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 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
#       GMT EXAMPLE 05
#
#       @(#)job05.csh     1.1  09/21/98
#
# Purpose:      Generate grid and show monochrome 3-D perspective
# GMT progs:    grdgradient, grdmath, grdview, pstext
# Unix progs:   echo, rm
#
grdmath -R-15/15/-15/15 -I0.3 X Y HYPOT DUP 2 MUL PI MUL 8 DIV COS EXCH NEG 10 DIV EXP MUL =
sombrero.grd
echo '-5        128     128     128     5       128     128     128' >! gray.cpt
grdgradient sombrero.grd -A225 -Gintensity.grd -Nt0.75
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 -E120/30 -U/-1.25/-0.75/"Example 5 in Cookbook" >! example_05.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_05.ps
\rm -f gray.cpt sombrero.grd intensity.grd .gmtcommands
```

The variations in intensity could be made more dramatic by using *grdmath* to scale the intensity file before running *grdview*. For very rough data sets one may improve the smoothness of the intensities by passing the output of *grdgradient* to *grdhisteq*. The shell-script above 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
#       GMT EXAMPLE 06
#
#       @(#)job06.csh     1.1  09/21/98
#
# Purpose:      Make standard and polar histograms
# GMT progs:   pshistogram, psrose
# Unix progs:   rm
#
psrose fractures.d -A10r -S1.8n -U/-2.25/-0.75/"Example 6 in Cookbook" -P -G0 -R0/1/0/360 -X2.5 -K -
B0.2g0.2/30g30 >! example_06.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_06.ps
\rm -f .gmtcommands
```

· 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
#       GMT EXAMPLE 07
#
#       @(#)job07.csh     1.1  09/21/98
#
# Purpose:      Make a basemap with earthquakes and isochrons etc
# GMT progs:   pscoast, pstext, psxy
# Unix progs:   $AWK, echo, rm
#
pscoast -R-50/0/-10/20 -JM9 -K -GP0/26 -Dl -W1 -B10 -U"Example 7 in Cookbook" >! example_07.ps
psxy -R -JM -O -K -M fz.xy -W2ta >> example_07.ps
$AWK '{print $1-360.0, $2, $3*0.01}' quakes.xym | psxy -R -JM -O -K -H -Sc -G255 -L -W1 >> example_07.ps
psxy -R -JM -O -K -M isochron.xy -W3 >> example_07.ps
psxy -R -JM -O -K -M ridge.xy -W7 >> example_07.ps
psxy -R -JM -O -K -G255 -W4 -A << END >> example_07.ps
-14.5  15.2
-215.2
-217.8
-14.5  17.8
END
psxy -R -JM -O -K -G255 -W2 -A << END >> example_07.ps
-14.3515.35
-2.15  15.35
-2.15  17.65
-14.3517.65
END
echo "-13.5 16.5" | psxy -R -JM -O -K -Sc0.08 -G255 -W2 >> example_07.ps
echo "-12.5 16.5 18 0 6 5 ISC Earthquakes" | pstext -R -JM -O -K >> example_07.ps
pstext -R -JM -O -S3 -G255 << END >> example_07.ps
-43 -5 30 0 1 6 SOUTH
-43 -8 30 0 1 6 AMERICA
```

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

```
-7 11 30 0 1 6 AFRICA
END
\rm -f .gmtcommands
```

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
#       GMT EXAMPLE 08
#
#       @(#)job08.csh     1.1  09/21/98
#
# Purpose:      Make a 3-D bar plot
# GMT progs:   grd2xyz, pstext, psxyz
# Unix progs:   echo, rm
#
grd2xyz topo.grd | psxyz -B1/1/1000:"Topography (m)"::.ETOPO5:WSneZ+ -R-0.1/5.1/-0.1/5.1/-5000/0 -
JM5 -JZ6 -E200/30 -So0.0833333b-5000 -P -U"Example 8 in Cookbook" -W -G240 -K >! example_08.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_08.ps
\rm -f .gmtcommands
```

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 (which is the first part of the file name for each profile) 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 (i.e., the angle with the baseline must be 50˚). 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 with **–D** in order to have a small gap between the profile and the label:

```
#!/bin/csh
#       GMT EXAMPLE 09
#
#       @(#)job09.csh     1.1  09/21/98
#
# Purpose:      Make wiggle plot along track from geoid deflections
# GMT progs:   pswiggle, pstext, psxy
# Unix progs:   $AWK, ls, paste, tail, rm
#
pswiggle *.xys -R185/250/-68/-42 -U"Example 9 in Cookbook" -K -Jm0.13 -Ba10f5 -G0 -Z2000 -W1 -S240/-
67/500/@~m@~rad >! example_09.ps
psxy -R -Jm -O -K ridge.xy -W5 >> example_09.ps
psxy -R -Jm -O -K -M fz.xy -W2ta >> example_09.ps
if (-e tmp) then
   \rm -f tmp
endif
foreach file (*.xys)          # Make label file
   tail -1 $file >>! tmp
end
ls *.xys | $AWK -F. '{print $2}' >! tracks.lis
paste tmp tracks.lis | $AWK '{print $1, $2, 10, 50, 1, 7, $4}' | pstext -R -Jm -D-0.05/-0.05 -O >>
example_09.ps
\rm -f tmp tracks.lis .gmtcommands
```

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
#       GMT EXAMPLE 10
#
#       @(#)job10.csh     1.1 09/21/98
#
# Purpose:      Make 3-D bar graph on top of perspective map
# GMT progs:   pscoast, pstext, psxyz
# Unix progs:   $AWK, rm
#
pscoast -R-180/180/-90/90 -JX8/5d -Dc -G0 -E200/40 -K -U"Example 10 in Cookbook" >! example_10.ps
psxyz agu.d -R-180/180/-90/90/1/100000 -JX8/5d -JZ2.5l -So0.3b1 -G140 -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-180/180/-90/90 -JX -O -K -E200/40 -G255 -S2 >>
example_10.ps
echo "4.5 6 30 0 5 2 AGU 1991 Membership Distribution" | pstext -R0/11/0/8.5 -Jx1 -O >> example_10.ps
\rm -f .gmtcommands
```

· 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
#       GMT EXAMPLE 11
#
#       @(#)job11.csh     1.1  09/21/98
#
# Purpose:      Create a 3-D RGB Cube
# GMT progs:   gmtset, grdimage, grdmath, pstext, psxy
# Unix progs:   $AWK, rm
#
# 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.
#

grdmath -I1 -R0/255/0/255 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
$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

$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

$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

$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

$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
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

$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
```

```
\rm -f rgb_cube.cpt rgb_cube.grd .gmtcommands
```

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. Thus, in this example we do not actually make gridded files but still are able to contour and image the data. We use a color palette table topo.cpt (supplied with the script data separately). The script becomes:

```
#!/bin/csh
#       GMT EXAMPLE 12
#
#       @(#)job12.csh     1.1  09/21/98
#
# Purpose:      Illustrates Delaunay triangulation of points, and contouring
# GMT progs:    makecpt, minmax, pscontour, pstext, psxy, triangulate
# Unix progs:   $AWK, echo, rm
#
# First draw network and label the nodes
triangulate table_5.11 -M >! net.xy
psxy -R0/6.5/-0.2/6.5 -JX3.06/3.15 -B2f1WSNe -M net.xy -W2 -P -K -Y4.65 >! example_12.ps
psxy table_5.11 -R -JX -O -K -Sc0.12 -G255 -W >> 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 -X3.25 >> 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 %lg\n", $1, $2, $3}' table_5.11 | pstext -R -JX -O -K -W255o -C0.01/0.01 -
D0.08/0 -N >> example_12.ps
# Then contour the data and draw triangles using dashed pen; use "minmax" and "makecpt" to make a color
palette (.cpt) file
set zstuff = `$AWK '{ print $2, $3 }' table_5.11 | minmax -C -I25 | $AWK '{ print 0.5*($3 + $4), ($4-$3)/25 }'`
makecpt -C25 -S$zstuff[2]c -M$zstuff[1] >! topo.cpt
pscontour -R -JX table_5.11 -B2f1WSne -W3 -Ctopo.cpt -L1ta -G1 -X-3.25 -Y-3.65 -O -K -U"Example 12 in
Cookbook" >> example_12.ps
# Finally color the topography
pscontour -R -JX table_5.11 -B2f1eSnw -Ctopo.cpt -I -X3.25 -O -K >> example_12.ps
echo "3.16 8 30 0 1 2 Delaunay Triangulation" | pstext -R0/8/0/11 -Jx1 -O -X-3.25 >> example_12.ps
#
\rm net.xy topo.cpt .gmtcommands
```

· 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 use *grdmath* to generate a surface $z(x, y) = x \cdot \exp(-x^2 - y^2)$ and use *grdgradient* to calculate 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
#       GMT EXAMPLE 13
#
#       @(#)job13.csh     1.1  09/21/98
#
# Purpose:      Illustrate vectors and contouring
# GMT progs:    grdmath, grdcontour, grdgradient, pstext
# Unix progs:   echo, rm
#
grdmath -R-2/2/-2/2 -I0.1 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 -X3.45 >> 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 -R0/6/0/4.5 -Jx1 -O -X-3.45 >>
example_13.ps
\rm -f z.grd dzdx.grd dzdy.grd .gmtcommands
```

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 logical blocks are drawn from a temporary file we make on the fly within the shell script. The processed data is then gridded with the *surface* program and contoured every 25 units. A most important point here is that *blockmean* or *blockmedian* should always be run prior to running *surface*, and both of these steps must use the same grid interval. 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 using *psxy*:

```
#!/bin/csh
#       GMT EXAMPLE 14
#
#       @(#)job14.csh     1.1  09/21/98
#
# Purpose:      Showing simple gridding, contouring, and resampling along tracks
# GMT progs:    blockmean, grdcontour, grdtrack, grdtrend, minmax, project, pstext
#       psxy, surface
# Unix progs:   $AWK, echo, rm
#
# First draw network and label the nodes
psxy table_5.11 -R0/7/0/7 -JX3.06/3.15 -B2f1WSNe -Sc0.05 -G0 -P -K -Y6.45 >! example_14.ps
$AWK '{printf "%lg %s 6 0 0 5 %lg\n", $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 -X3.25 >> 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 %lg\n", $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
grdcontour data.grd -JX -B2f1WSne -C25 -A50 -G3/10 -S4 -O -K -X-3.25 -Y-3.55 >> 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
grdcontour trend.grd -JX -B2f1wSne -C25 -A50 -G3/10 -S4 -O -K -X3.25 >> example_14.ps
project -C0/0 -E7/7 -G0.1 -Fxy >! 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` -JX6.3/1.4 data.d -W4 -O -K -X-3.25 -Y-1.9 -B1/50WSne >>
example_14.ps
psxy -R -JX trend.d -W2ta -O -U"Example 14 in Cookbook" >> example_14.ps
\rm mean.xyz track tmp *.grd *.d .gmtcommands
```

· Example 15: Gridding, contouring, and masking of unconstrained areas.

This example demonstrates some off the different ways one can use to grid data in GMT, and how to deal with unconstrained areas. We first convert a large ascii file to binary with *gmtconvert* since the binary file will read and process much faster. Our lower left plot illustrates the results of gridding using a nearest neighbor technique (*nearneighbor*) which is a local method: No output is given where there are no data. Next (lower right), we use a minimum curvature technique (*surface*) which is a global method. Hence, the contours cover the entire map allthough the data are only available for portions of the area (indicated by the gray areas plotted using *psmask*). The top left scenario illustrates how we can create a clip path (using *psmask*) based on the data coverage to eliminate contours outside the constrained area. Finally (top right) we simply employ *pscoast* to overlay gray landmasses to cover up the unwanted contours, and end by plotting a star at the deepest point on the map with *psxy*. This point was extracted from the gridded files using *grdinfo.*

```
#!/bin/csh
#       GMT EXAMPLE 15
#
#       @(#)job15.csh     1.1  09/21/98
#
# Purpose:      Gridding and clipping when data are missing
# GMT progs:    blockmedian, gmtconvert, grdclip, grdcontour, grdinfo, minmax,
#       nearneighbor, pscoast, psmask, pstext, surface
# Unix progs:   $AWK, echo, rm
#
gmtconvert ship.xyz -bo >! ship.b
set region = `minmax ship.b -I1 -bi3`
nearneighbor $region -I10m -S40k -Gship.grd ship.b -bi3
grdcontour ship.grd -JM3 -P -B2WSne -C250 -A1000 -K -U"Example 15 in Cookbook" >! example_15.ps
#
blockmedian $region -I10m ship.b -bi3 -bo >! ship_10m.b
surface $region -I10m ship_10m.b -Gship.grd -bi3
psmask $region -I10m ship.b -JM -O -K -T -G220 -bi3 -X3.6 >> example_15.ps
grdcontour ship.grd -JM -B2WSne -C250 -A1000 -O -K >> example_15.ps
#
psmask $region -I10m ship_10m.b -bi3 -JM -B2WSne -O -K -X-3.6 -Y3.75 >> example_15.ps
grdcontour ship.grd -JM -C250 -A1000 -O -K >> example_15.ps
psmask -C -O -K >> example_15.ps
#
grdclip ship.grd -Sa-1/NaN -Gship_clipped.grd
set info = `grdinfo -C -M ship_clipped.grd`
grdcontour ship_clipped.grd -JM -B2WSne -C250 -A1000 -O -K -X3.6 >> example_15.ps
pscoast $region -JM -O -K -G150 -W1 >> example_15.ps
echo $info[12] $info[13] | psxy -R -JM -O -K -Sa0.15 -W4 >> example_15.ps
echo "-0.3 3.6 24 0 1 CB Gridding with missing data" | pstext -R0/3/0/4 -Jx1 -O -N >> example_15.ps
\rm -f ship.b ship_10m.b ship.grd ship_clipped.grd
```

· Example 16: Gridding of data, continued.

pscontour (for contouring) and *triangulate* (for gridding) use the simplest method of interpolating data: a Delaunay triangulation (see Example 12) which forms $z(x, y)$ as a union of planar triangular facets. One advantage of this method is that it will not extrapolate $z(x, y)$ beyond the convex hull of the input (x, y) data. Another is that it will not estimate a z value above or below the local bounds on any triangle. A disadvantage is that the $z(x, y)$ surface is not differentiable, but has sharp kinks at triangle edges and thus also along contours. This may not look physically reasonable, but it can be filtered later (last panel below). *surface* can be used to generate a higher-order (smooth and differentiable) interpolation of $z(x, y)$ onto a grid, after which the grid may be illustrated (*grdcontour*, *grdimage*, *grdview*). *surface* will interpolate to all (x, y) points in a rectangular region, and thus will extrapolate beyond the convex hull of the data. However, this can be masked out in various ways (see Example 15).

A more serious objection is that *surface* may estimate z values outside the local range of the data (note area near $x = 0.8$, $y = 5.3$). This commonly happens when the default tension value of zero is used to create a "minimum curvature" (most smooth) interpolant. *surface* can be used with non-zero tension to partially overcome this problem. The limiting value tension = 1 should approximate the triangulation, while a value between 0 and 1 may yield a good compromise between the above two cases. A value of 0.5 is shown here. A side effect of the tension is that it tends to make the contours turn near the edges of the domain so that they approach the edge from a perpendicular direction. A solution is to use *surface* in a larger area and then use *grdcut* to cut out the desired smaller area. Another way to achieve a compromise is to interpolate the data to a grid and then filter the grid using *grdfft* or *grdfilter*. The latter can handle grids containing "NaN" values and it can do median and mode filters as well as convolutions. Shown here is *triangulate* followed by *grdfilter*. Note that the filter has done some extrapolation beyond the convex hull of the original x, y values. The "best" smooth approximation of $z(x, y)$ depends on the errors in the data and the physical laws obeyed by z. GMT cannot always do the "best" thing but it offers great flexibility through its combinations of tools.

```
#!/bin/csh
#       GMT EXAMPLE 16
#
#       @(#)job16.csh     1.2  09/24/98
#
# Purpose:      Illustrates interpolation methods using same data as Example 12.
# GMT progs:   makecpt, surface, triangulate
# Unix progs:   $AWK, rm
#
# First make a cpt file as in example 12:
#
set zstuff = `$AWK '{ print $2, $3 }' table_5.11 | minmax -C -I25 | $AWK '{ print 0.5*($3 + $4), ($4-$3)/25 }'`
makecpt -C25 -S$zstuff[2]c -M$zstuff[1] >! topo.cpt
#
# Now illustrate various means of contouring, using triangulate and surface.
#
gmtset MEASURE_UNIT INCH ANOT_FONT_SIZE 9
#
pscontour -R0/6.5/-0.2/6.5 -Jx0.45 -P -K -Y4.75 -Ba2f1WSne table_5.11 -Ctopo.cpt -I >! example_16.ps
echo "3.25 7 18 0 4 CB pscontour (triangulate)" | pstext -R -Jx -O -K -N >> example_16.ps
#
surface table_5.11 -R -I0.1 -Graws0.grd
grdview raws0.grd -R -Jx -Ba2f1WSne -Ctopo.cpt -Qs -O -K -X3.5 >> example_16.ps
echo "3.25 7 18 0 4 CB surface (tension = 0)" | pstext -R -Jx -O -K -N >> example_16.ps
#
surface table_5.11 -R -I0.1 -Graws5.grd -T0.5
```

```
grdview raws5.grd -R -Jx -Ba2f1WSne -Ctopo.cpt -Qs -O -K -Y-3.75 -X-3.5 -U"Example 16 in Cookbook" >>
example_16.ps
echo "3.25 7 18 0 4 CB surface (tension = 0.5)" | pstext -R -Jx -O -K -N >> example_16.ps
#
triangulate table_5.11 -Grawt.grd -R -I0.1 > /dev/null
grdfilter rawt.grd -Gfiltered.grd -D0 -Fc1
grdview filtered.grd -R -Jx -Ba2f1WSne -Ctopo.cpt -Qs -O -K -X3.5 >> example_16.ps
echo "3.25 7 18 0 4 CB triangulate @%12%@\256@%% grdfilter" | pstext -R -Jx -O -K -N >> example_16.ps
echo "3.2125 7.5 32 0 4 CB Gridding of Data" | pstext -R0/10/0/10 -Jx1 -O -N -X-3.5 >> example_16.ps
#
\rm topo.cpt *.grd .gmtcommands
```

· Example 17: Images clipped by coastlines.

This example demonstrates how *pscoast* can be used to set up clippaths based on coastlines. This approach is well suited when different gridded data sets are to be merged on a plot using different color palette files. Merging the files themselves may not be doable since they may represent different data sets, as we show in this example. Here, we lay down a color map of the geoid field near India with *grdimage*, use *pscoast* to set up land clippaths, and then overlay topography from the ETOPO5 data set with another call to *grdimage*. We finally undo the clippath with a second call to *pscoast* with the option **–Q**.

```
#!/bin/csh
#       GMT EXAMPLE 17
#
#       @(#)job17.csh     1.3  09/25/98
#
# Purpose:      Illustrates clipping of images using coastlines
# GMT progs:   grd2cpt, grdgradient, grdimage, pscoast
# Unix progs:   rm
#
# Get Geoid and Topography for the region
#grdraster 1 -R60/90/-10/25 -Getopo5.grd
#grdraster 4 -R60/90/-10/25 -Ggeoid.grd

# First generate geoid image w/ shading

grd2cpt geoid.grd >! geoid.cpt
grdgradient geoid.grd -Nt1 -A45 -Ggeoid_i.grd
grdimage geoid.grd -Igeoid_i.grd -JM6.5 -Cgeoid.cpt -P -K -U"Example 17 in Cookbook" >! example_17.ps

# Then use pscoast to initiate clip path for land

pscoast -R60/90/-10/25 -JM -O -K -Dl -Gc >> example_17.ps

# Now generate topography image w/shading

echo "-10000 150 10000 150" >! gray.cpt
grdgradient etopo5.grd -Nt1 -A45 -Getopo5_i.grd
grdimage etopo5.grd -Ietopo5_i.grd -JM -Cgray.cpt -O -K >> example_17.ps

# Finally undo clipping and overlay basemap

pscoast -R -JM -O -Q -B10f5:.”Clipping of Images”: >> example_17.ps

# Clean up

\rm -f geoid.cpt gray.cpt *_i.grd .gmt*
```

· Example 18: Volumes and Spatial Selections.

To demonstrate potential usage of the new programs *grdvolume* and *gmtselect* we extract a subset of the Sandwell & Smith altimetric gravity field† for the northern Pacific and decide to isolate all seamounts that (1) exceed 50 mGal in amplitude and (2) are within 200 km of the Pratt seamount. We do this by dumping the 50 mGal contours to disk, then making a simple $AWK script *center.awk* that returns the mean location of the points making up each closed polygon, and then pass these locations to *gmtselect* which retains only the points within 200 km of Pratt. We then mask out all the data outside this radius and use *grdvolume* to determine the combined area and volumes of the chosen seamounts.

```
#!/bin/csh
#       GMT EXAMPLE 18
#
#       @(#)job18.csh     1.1  09/25/98
#
# Purpose:      Illustrates volumes of grids inside contours and spatial
#       selection of data
# GMT progs:   gmtset, gmtselect, grdclip, grdcontour, grdgradient, grdimage,
#       grdmath, grdvolume, makecpt, pscoast, psscale, pstext, psxy
# Unix progs:   $AWK, cat, rm
#
# Get Sandwell/Smith gravity for the region
#img2latlongrd /home/pahoehoe4/data/grids/world_grav.img.7.2 -R-149/-135/52.5/58 -Gss_grav.grd -T1

# Use spherical projection since SS data define on sphere
gmtset ELLIPSOID Sphere

# Define location of Pratt seamount
echo "-142.65   56.25" >! pratt.d

# First generate gravity image w/ shading, label Pratt, and draw a circle
# of radius = 200 km centered on Pratt.

makecpt -C10 -S12c -Z >! grav.cpt
grdgradient ss_grav.grd -Nt1 -A45 -Gss_grav_i.grd
grdimage ss_grav.grd -Iss_grav_i.grd -JM5.5 -Cgrav.cpt -B2f1 -P -K -X1.5 -Y5.85 >! example_18.ps
pscoast -R-149/-135/52.5/58 -JM -O -K -Di -G160 -W1 >> example_18.ps
psscale -D2.75/-0.4/4/0.15h -Cgrav.cpt -B20f10g10/:mGal: -O -K >> example_18.ps
$AWK '{print $1, $2, 12, 0, 1, "LB", "Pratt"}' pratt.d | pstext -R -JM -O -K -D0.1/0.1 >> example_18.ps
$AWK '{print $1, $2, 0, 200, 200}' pratt.d | psxy -R -JM -O -K -SE -W1 >> example_18.ps

# Then draw 10 mGal contours and overlay 50 mGal contour in green

grdcontour ss_grav.grd -JM -C20 -B2f1WSEn -O -K -Y-4.85 -U/-1.25/-0.75/"Example 18 in Cookbook" >>
example_18.ps
grdcontour ss_grav.grd -JM -C10 -L49/51 -O -K -Dsm -Wc3/0/255/0 >> example_18.ps
pscoast -R -JM -O -K -Di -G160 -W1 >> example_18.ps
$AWK '{print $1, $2, 0, 200, 200}' pratt.d | psxy -R -JM -O -K -SE -W1 >> example_18.ps
\rm -f sm_*[0-9].xyz        # Only consider closed contours

# Now determine centers of each enclosed seamount > 50 mGal but only plot
# the ones within 200 km of Pratt seamount.

# First make a simple $AWK script that returns the average position
# of a file with coordinates x, y (remember to escape the $ sign)

cat << EOF >! center.awk
BEGIN {
```

† htt://topex.ucsd.edu/marine_grav/mar_grav.html

```
   x = 0
   y = 0
   n = 0
}
{
   x += \$1
   y += \$2
   n++
}
END {
   print x/n, y/n
}
EOF

# Now determine mean location of each closed contour and
# add it to the file centers.d

\rm -f centers.d
foreach file (sm_*.xyz)
   $AWK -f center.awk $file >>! centers.d
end

# Only plot the ones within 200 km

gmtselect -R -JM -C200/pratt.d centers.d | psxy -R -JM -O -K -SC0.04 -G255/0/0 -W1 >> example_18.ps
psxy -R -JM -O -K -ST0.1 -G255/255/0 -W1 pratt.d >> example_18.ps

# Then report the volume and area of these seamounts only
# by masking out data outside the 200 km-radius circle
# and then evaluate area/volume for the 50 mGal contour

grdmath -R -I2m -F -142.65 56.25 GDIST = mask.grd
grdclip mask.grd -Sa200/NaN -Sb200/1 -Gmask.grd
grdmath ss_grav.grd mask.grd MUL = tmp.grd
set info = `grdvolume tmp.grd -C50 -Sk`

psxy -R -JM -A -O -K -L -W3 -G255 << EOF >> example_18.ps
-148.5 52.75
-140.552.75
-140.553.75
-148.553.75
EOF
pstext -R -JM -O << EOF >> example_18.ps
-148 53.08 14 0 1 LM Areas: $info[2] km@+2@+
-148 53.42 14 0 1 LM Volumes: $info[3] mGal\264km@+2@+
EOF

# Clean up

\rm -f grav.cpt sm_*.xyz *_i.grd tmp.grd mask.grd pratt.d center* .gmt*
```

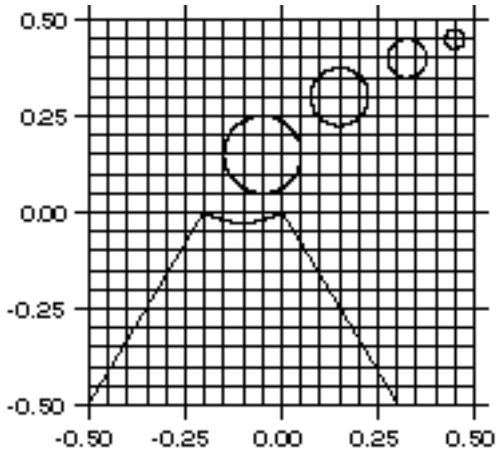
· Example 19: Color patterns on maps.

GMT 3.1 introduced color patterns and this examples give a few cases of how to use this new feature. We make a phony poster that advertises an international conference on GMT in Honolulu. We use *grdmath*, *makecpt*, and *grdimage* to draw pleasing color backgrounds on maps, and overlay *pscoast* clippaths to have the patterns change at the coastlines. The middle panel demonstrates a simple *pscoast* call where the built-in pattern # 86 is drawn at 100 dpi but with the black and white pixels replaced with color combinations. The final panel repeats the top panel except that the land and sea images have changed places.

```
#!/bin/csh
#       GMT EXAMPLE 19
#
#       @(#)job19.csh     1.1  09/25/98
#
# Purpose:      Illustrates various color pattern effects for maps
# GMT progs:   grdimage, grdmath, pscoast, pstext
# Unix progs:   rm
#
# First make a worldmap with graded blue oceans and rainbow continents

grdmath -R-180/180/-90/90 -I1 -F Y COSD 2 POW = lat.grd
grdmath -R-180/180/-90/90 -I1 -F X = lon.grd
echo "0 255 255 255 1 0 0 255" >! lat.cpt
makecpt -C60 -S6c -Z >! lon.cpt
grdimage lat.grd -JI0/6.5 -Clat.cpt -P -K -Y7.5 -B0 >! example_19.ps
pscoast -R -JI -O -K -Dc -A5000 -Gc >> example_19.ps
grdimage lon.grd -JI -Clon.cpt -O -K >> example_19.ps
pscoast -R -JI -O -K -Q >> example_19.ps
pscoast -R -JI -O -K -Dc -A5000 -W1 >> example_19.ps
echo "0 20 32 0 1 CM FIRST INTERNATIONAL" | pstext -R -JI -O -K -G255/0/0 -S2 >> example_19.ps
echo "0 -10 32 0 1 CM GMT CONFERENCE" | pstext -R -JI -O -K -G255/0/0 -S2 >> example_19.ps
echo "0 -30 18 0 1 CM Honolulu, Hawaii, April 1, 2000" | pstext -R -JI -O -K -G0/255/50 -S1 >>
example_19.ps

# Then show example of color patterns

pscoast -R -JI -O -K -Dc -A5000 -Gp100/86:F255/0/0B255/255/0 -Sp100/7:F255/0/0B0/0/0 -B0 -Y-3.25 >>
example_19.ps
echo "0 15 32 0 1 CM SILLY USES OF" | pstext -R -JI -O -K -G50/255/50 -S2 >> example_19.ps
echo "0 -15 32 0 1 CM GMT COLOR PATTERNS" | pstext -R -JI -O -K -G255/0/255 -S2 >> example_19.ps

# Finally repeat 1st plot but exchange the patterns

grdimage lon.grd -JI0 -Clon.cpt -O -K -Y-3.25 -B0 -U"Example 19 in Cookbook" >> example_19.ps
pscoast -R -JI -O -K -Dc -A5000 -Gc >> example_19.ps
grdimage lat.grd -JI -Clat.cpt -O -K >> example_19.ps
pscoast -R -JI -O -K -Q >> example_19.ps
pscoast -R -JI -O -K -Dc -A5000 -W1 >> example_19.ps
echo "0 20 32 0 1 CM FIRST INTERNATIONAL" | pstext -R -JI -O -K -G255/0/0 -S2 >> example_19.ps
echo "0 -10 32 0 1 CM GMT CONFERENCE" | pstext -R -JI -O -K -G255/0/0 -S2 >> example_19.ps
echo "0 -30 18 0 1 CM Honolulu, Hawaii, April 1, 2000" | pstext -R -JI -O -G0/255/50 -S1 >> example_19.ps

\rm -f l*.grd l*.cpt .gmt*
```

· Example 20: Custom map symbols.

One is often required to make special maps that shows the distribution of certain features but one would prefer to use a custom symbol instead of the built-in circles, squares, triangles etc. in the GMT plotting programs *psxy* and *psxyz*. Here we demonstrate one approach that allows for a fair bit of flexibility in designing one's own symbols. The following recipe is used when designing a new symbol. (1) Use *psbasemap* (or engineering paper!) to set up an empty grid that goes from -0.5 to +0.5 in both *x* and *y*. Use ruler and compass to draw your new symbol using straight lines and arcs of circles. This is how your symbol will look when a size of 1 inch is chosen. The following illustrates a new symbol we will call volcano.



(2) After designing the symbol we will encode it using a simple set of rules. In our case we describe our volcano using no more than 4 possible constructs:

x_0 y_0 M [**–G***fill*] [**–W***pen*]	Start new element at x_0, y_0
x_1 y_1 D	Draw straight line from current point to x_1, y_1
x_0 y_0 r $\Delta\alpha$ C [**–G***fill*] [**–W***pen*]	Draw single circle of radius r around x_0, y_0 with step-size $\Delta\alpha$ °
x_0 y_0 r α_1 α_2 $\Delta\alpha$ A	Draw arc segment of radius r from angle α_1 to α_2 every $\Delta\alpha$ °

The optional **–G** and **–W** can be used to hardwire the color fill and pen for segments. By default the segments are painted based on the values of the command line settings.

Manually applying these rules to our symbol results in a definition file volcano.def:

```
# Segment info file for volcano symbol
# These instructions are intended for make_symbol which will generate an $AWK-script that
# creates multiple-segment output describing the desired symbol at the chosen size.
#
# Main cone with caldera
-0.5    -0.5        M
-0.2    0           D
-0.1    0.173205081     0.2     240     300     2       A
0 0     D
0.3     -0.5        D
# Smoke bubbles
-0.05   0.15        0.1     5       C
0.15    0.3         0.075   5       C
0.325   0.4         0.05    10      C
0.45    0.45        0.025   10      C
```

The values refer to positions and dimensions illustrated in the figure above. (3) Given a proper definition file we use the supplied $AWK-script *make_symbol* to create a custom $AWK-script that will read locations and sizes of the desired symbols and replace them with line-drawings like the one in the figure but translated and scaled accordingly. The output is a multi-segment file (see Appendix B) which may be plotted with *psxy* or *psxyz*.

We are now ready to give it a try. Based on the hotspot locations in the file hotspots.d (with a 3rd column giving the desired symbol sizes in inches) we lay down a world map and overlay red volcano symbols using our custom-built $AWK script volcano.awk and *psxy*. Note that you must first transform the coordinates using *mapproject* prior to running the $AWK script. Without further discussion we also make a definition for a multi-colored bulls-eye symbol:

```
#
# Segment info file for bullseye symbol
# These instructions are intended for make_symbol which will generate an $AWK-script that creates
# multiple-segment output describing the desired symbol at the chosen size. The symbol will be
# painted drawn given the -G -W options for each segment.
#
0 -0.7M         -W4/255/0/0
0 0.7 D
-0.7    0       M       -W4/255/0/0
0.7     0       D
0 0     0.45    5       C       -Gp0/12
0 0     0.45    5       C       -W1
0 0     0.35    5       C       -G255/255/0 -W1
0 0     0.25    5       C       -Gp0/9
0 0     0.25    5       C       -W1
0 0     0.15    5       C       -G255/255/0 -W1
0 0     0.05    5       C       -G255 -W1
```

Here is our final map script:

```
#!/bin/csh
#       GMT EXAMPLE 20
#
#       @(#)job20.csh     1.3  10/08/98
#
# Purpose:      Extend GMT to plot custom symbols
# GMT progs:    pscoast, psxy, mapproject
# Unix progs:   rm, $AWK
#
# Make volcano- and bullseye-symbol using make_symbol on volcano.def and bullseye.def

$AWK -f make_symbol < volcano.def >! volcano.awk
$AWK -f make_symbol < bullseye.def >! bullseye.awk

# Plot a world-map with volcano symbols of different sizes on top given locations and sizes in hotspots.d

pscoast -R0/360/-90/90 -JR180/9 -B60/30:."HotSpot Islands and Cities": -G0/150/0 -S200/200/255 -Dc -
A5000 -K -U"Example 20 in Cookbook" >! example_20.ps

mapproject -R -JR hotspots.d | $AWK -f volcano.awk | psxy -R0/9/0/9 -Jx1 -O -K -M -W1 -G255/0/0 -L >>
example_20.ps

# Overlay a few bullseyes at NY, Cairo, and Perth

mapproject -R0/360/-90/90 -JR180/9 cities.d | $AWK -f bullseye.awk | psxy -R0/9/0/9 -Jx1 -O -M >>
example_20.ps

\rm -f volcano.awk bullseye.awk
```

Given these guidelines you can easily make your own symbols. Symbols with more than one color can be obtained by making several symbol components. E.g., to have yellow smoke coming out of red volcanoes we would make two symbols: one with just the cone and caldera and the other with the bubbles. These would be plotted consecutively using the desired colors. Alternatively, like in bullseye.def, we may specify colors directly for the various segments. Note that the custom symbols, unlike the built-in symbols in GMT, can be used with the built-in patterns (Appendix E). Other approaches are also possible, of course.