Appendix H: Hints and known bugs concerning display of GMT *PostScript*

GMT creates valid (so far as we know) Adobe *PostScript* Level I. It does not use operators specific to Level II and should therefore produce output that will print on old as well as new *PostScript* printers Sometimes unexpected things happen when GMT output is sent to certain printers or displays. This section lists some things we have learned from experience, and some work-arounds.

• *PostScript* driver bugs.

When you try to display a *PostScript* file on a device, such as a printer or your screen, then a program called a *PostScript* device driver has to compute which device pixels should receive which colors (black or white in the case of a simple laser printer) in order to display the file. At this stage, certain device-dependent things may happen. These are not limitations of GMT or *PostScript*, but of the particular display device. The following bugs are known to us based on our experiences:

Early versions of the Sun SPARCprinter software caused linewidth-dependent path displacement. We reported this bug and it has been fixed in newer versions of the software. Try using *psxy* to draw $y = f(x)$ twice, once with a thin pen (**–W**1) and once with a fat pen (**–W**10); if they do not plot on top of each other, you have this kind of bug and need new software. The problem may also show up when you plot a mixture of solid and dashed (or dotted) lines of various pen thickness

The first version of the HP Laserjet 4M had bugs in the driver program. We reported it, and they have just released (we got ours Aug. 10 1993) a new one. The old one was *PostScript* SIMM, part number C2080-60001; the new one is called *PostScript* SIMM, part number C2080-60002. You need to get this one plugged into your printer if you have an HP LaserJet 4M.

Apple Laserwriters with the older versions of Apple's *PostScript* driver will give the error "limitcheck" and fail to plot when they encounter a path exceeding about 1000–1500 points. Try to get a newer driver from Apple, but if you can't do that, set the parameter MAX_L1_PATH to 1000–1500 or even smaller in the file src/pslib_inc.h and recompile GMT. The number of points in a *PostScript* path can be arbitrarily large, in principle; GMT will only create paths longer than MAX_L1_PATH if the path represents a filled polygon or clipping path. Line-drawings (no fill) will be split so that no segment exceeds MAX_L1_PATH. This means *psxy* **–G** will issue a warning when you plot a polygon with more than MAX_L1_PATH points in it. It is then your responsibility to split the large polygon into several smaller segments. If *pscoast* gives such warnings and the file fails to plot you may have to select one of the lower resolution databases The path limitation exemplified by these Apple printers is what makes the higher-resolution coastlines for *pscoast* non-trivial: such coastlines have to be organized so that fill operations do not generate excessively large paths. Some HP *PostScript* cartridges for the Laserjet III also have trouble with paths exceeding 1500 points; they may successfully print the file, but it can take all night!

8-bit color screen displays (and programs which use only 8-bits, even on 24-bit monitors, such as Sun's Pageview under OW3) may not dither cleverly, and so the color they show you may not resemble the color your *PostScript* file is asking for. Therefore, if you choose colors you like on the screen, you may be surprised to find that your plot looks different on the hardcopy printer or film writer. The only thing you can do is be aware of this, and make some test cases on your hardcopy devices and compare them with the screen, until you get used to this effect. (Each hardcopy device is also a little different, and so you will eventually find that you want to tune your color choices for each device.) The rgb color cube in example 11 may help.

Some versions of Sun's OW program Pageview have only a limited number of colors available; the number can be increased somewhat by starting openwin with the option "openwin -cubesize large". (Our SPARC-10 doesn't seem to need this anymore, but earlier machines did.)

Many color hardcopy devices use CMYK color systems. GMT *PostScript* uses RGB (even if your .cpt files are using HSV). The three coordinates of RGB space can be mapped into three coordinates in CMY space, and in theory K (black) is superfluous. But it is hard to get CMY inks to mix into a good black or gray, so these printers supply a black ink as well, hence CMYK. The *PostScript* driver for a CMYK printer should be smart enough to compute what portion of CMY can be drawn in K, and use K for this and remove it from CMY; however, some of them aren't.

In early releases of GMT we always used the *PostScript* command setrgbcolor(r,g,b) to specify colors, even if the color happened to be a shade of gray (r=g=b) or black (r=g=b=0). One of our users found that black came out muddy brown when he used FreedomOfPress to make a Versatec plot of a GMT map. He found that if he used the *PostScript* command setgray(g) (where g is a graylevel) then the problem went away. Apparently, his installation of FreedomOfPress uses only CMY with the command setrgbcolor, and so setrgbcolor(0,0,0) tries to make black out of CMY instead of K. To fix this, in release 2.1 of GMT we changed some routines in pslib.c to check if (r=g and r=b), in which case setgray(g), else setrgbcolor(r,g,b).

Recent experience with some Tektronix Phaser printers and with commercial printing shops has shown that this substitution creates problems precisely opposite of the problems our Versatec user has. The Tektronix and commercial (we think it was a Scitex) machines do not use K when you say setgray(0) but they do when you say setrgbcolor(0,0,0). We believe that these problems are likely to disappear as the various software developers make their codes more robust. Note that this is not a fault with GMT: r = g = b = 0 means black and should plot that way. Thus, the GMT source code as shipped to you checks whether r=g and r=b, in which case it uses setgray, else setrgbcolor. If your gray tones are not being drawn with K, you have two work-around options: (1) edit the source for pslib.c or (2) edit your *PostScript* file and try using setrgbcolor in all cases. The simplest way to do so is to redefine the setgray operator to use setrgbcolor. Insert the line

```
/setgray  {dup dup setrgbcolor} def
```

immediately following the first line in the file (starts with %!PS.)

Some color film writers are very sensitive to the brand of film. If black doesn't look black on your color slides, try a different film.

• Resolution and dots per inch.

The parameter DOTS_PR_INCH can be set by the user through the .gmtdefaults file or *gmtset*. By default it is equal to the value in the src/gmt_defaults.h file, which is supplied with 300 when you get GMT from us. This seems a good size for most applications, but should ideally reflect the resolution of your hardcopy device (most laserwriters have 300 dpi, hence our default value). GMT computes what the plot should look like in double precision floating point coordinates, and then converts these to integer coordinates at DOTS_PR_INCH resolution. This helps us find out that certain points in a path lie on top of other points, and we can remove these, making smaller paths. Small paths are important for the laserwriter bugs above, and also to make fill operations compute faster. Some users have set their DOTS_PR_INCH to very large numbers. This only makes the *PostScript* output bigger without affecting the appearence of the plot. However, if you want to make a plot which fits on a page at first, and then later magnify this same *PostScript* file to a huge size, the higher DPI is important. Your data may not have the higher resolution but on certain devices the edges of fonts will not look crisp if they are not drawn with an effective resolution of 300 dpi or so. Beware of making an excessively large path. Note that if you change dpi the linewidths produced by your **–W** options will change, unless you have appended **p** for linewidth in points.

· European Characters

Note for users of "pageview" in Sun OpenWindows: **GMT** now offers some octal escape sequences to load European alphabet characters in text strings (see section 4.15). When this feature is enabled, the header on **GMT** *PostScript* output includes a section defining special fonts. The definition is added to the header whether or not your plot actually uses the fonts.

Users who view their **GMT** *PostScript* output using "pageview" in OpenWindows on Sun computers or user older laserwriters may have difficulties with the European font definition. If your installation of OpenWindows followed a space-saving suggestion of Sun, you may have excluded the European fonts, in which case pageview will fail to show you anything when you try to view a plot.

Ask your system administrator about this, or run this simple test: (1) View a **GMT** *PostScript* file with "pageview". If it comes up OK, you will be fine. If it comes up blank, open the "Edit PostScript" button and examine the lower window for error messages. (The European font problem generates lots of error messages in this window). (2) Verify that the *PostScript* file is OK, by sending it to a laser printer and making sure it comes out. (3) If the *PostScript* file is OK but it chokes "pageview", then edit the PostScript file, cutting out everything between the lines:

```
%%%%% START OF EUROPEAN FONT DEFINITION %%%%%
<bunch of definitions
%%%%% END OF EUROPEAN FONT DEFINITION %%%%%
```

Now try "pageview" on the edited version. If it now comes up, you have a limited subset of OpenWindows installed. If you discover that these fonts cause you trouble, then you can edit your .gmtdefaults file to set WANT_EURO_FONT = FALSE, which will suppress the printing of this definition in the **GMT** *PostScript* header. With this set to FALSE, you can make output which will be viewable in pageview without any editing. However, you would have to reset this to TRUE before attempting to use European fonts, and then the output will become un-pageview-able again. If you try to concatenate segments of **GMT** *PostScript* made with and without the European fonts enabled, then you may find that you have problems, either with the definition, or because you ask for something not defined.

• Hints.

When making images and perspective views of large amounts of data, the GMT programs can take some time to run, the resulting *PostScript* files can be very large, and the time to display the plot can be long. Fine tuning a plot script can take lots of trial and error. We recommend using *grdsample* to make a low resolution version of the data files you are plotting, and practice with that, so it is faster; when the script is perfect, use the full-resolution data files. We often begin building a script using only *psbasemap* and/or *pscoast* to get the various plots oriented correctly on the page; once this works we replace the *psbasemap* calls with the actually desired GMT programs.

If you want to make color shaded relief images and you haven't had much experience with it, here is a good first cut at the problem: Set your COLOR_MODEL to HSV using gmtset. Use *makecpt* or *grd2cpt* to make a continuous color palette spanning the range of your data. Use the **–Nt** option on *grdgradient*. Try the result, and then play with the tuning of the .gmtdefaults, the .cpt file, and the gradient file.