COLOR
A Program for Painting Color Contour Maps from a Regular Array

Robert L. Parker

INTRODUCTION

Complex monochrome contour maps consisting of level lines are often hard to interpret; the problem can be alleviated by resorting to colored regions or highlighting with gray shading. The program *color* provides contour maps in PostScript format. The colors are kept constant between specified level lines rather than making any attempt at continuous variation. The program operates in the same way as *plotxy* using the same command structure.

Numerical values are given in a rectangular array, saved as a diskfile. The user types a series of commands instructing the program to perform the tasks desired. Most of the parameters have default values so that to get a quick look at an array is very easy. Suppose the file MAT contains data arranged in 30 rows each of 10 numbers; then to obtain a color map one only need enter

```
file MAT
orient N
read 30 10
palette 1
show
plot
stop
```

Here the ASCII input file is specified by **file**; the program reads the data with **read**. The **orient** statement causes the data to be plotted in the orientation of a numerical listing of the array. Next the color are specified with a **palette** command. If this is omitted, the map will be painted in gray tones. Because no explicit levels were provided, the program generates its own and produces the contour map with **plot**. Naturally **stop** halts the program having flushed all the buffers. A PostScript plotfile named *mypost* has been generated which is automatically displayed at the terminal running *pageview*; it may be printed on a color PostScript printer; all printers in IGPP handle PostScript format.

All parameters and conditions continue to apply until explicitly changed by the user. Thus unless **file** is changed further **read** commands take data from the original data file. Or, suppose you wish to plot several variants of the same array; there is no need to read the array again. Notes and captions are provided in a primitive way. The catalog below describes the commands provided.

COMMAND CATALOG

All commands must begin in column 1. Each may be abbreviated to its first 4 letters although the full word is given in the catalog. The command line may be up to 80 characters in length. When a number or numbers follow a command, there must be at least one space after the command word. Numbers may be separated by a comma or spaces. Any line beginning with a blank is ignored.

above *z h s b*

The most flexible but also the most tedious way of specifying the colors you want in each region is with this command. The numbers z, h, s, b define a level, and a color description. Values on the map **above** the level z (and below the next higher level you give) receive the color defined by the Hue-Saturation-Brightness triple h, s, b. For more on what this means see **NOTES**. There must be a separate **above** command for each color in the contour map. Order does not matter. It is simpler to use one of the predefined palettes: see **palette**. Values below the lowest level mentioned in this command are assigned an extrapolated color close the color immediately above it. See also **levels** and **interval**.

affine *a b*

Immediately after input, the array values are transformed according to: znew= z * a + b, which is an affine mapping. All subsequent commands act on the transformed values, not the originals.

axes *x1 x2 y1 y2*
axes *x1 x2 y1 y2 hidden*

The bottom and left sides of the contour map can be annotated with linear axes covering the intervals specified. These intervals have nothing to do with the dimension of the array being contoured and can be set arbitrarily. Labeling of the axes is not provided explicitly but can be accomplished through the **note** command.

The command is also used to supply a coordinate system for additional lines that may be added on top of the contour map. (see **lines**). The limits are also written on the map at plot time, but this notation can be suppressed by including the word *hidden* as indicated. The axes and the lines do not participate in the transformations of the **mapping** command.

border *flag [mask]*

If a line surrounding the final map is desired, the user can invoke **border** either with a blank command field, or with flag=on. To cancel the border, when it has been set in an earlier plot, set flag=off. The default maps are borderless. If the code mask is included, the region between the normal bounding rectangle and an inscribed ellipse (or circle if width and height are equal) is set to be white, and is therefore masked out. An elliptical border is always supplied in this case. This option is useful for certain kinds of contour maps, like Lambert equal-area or Hammer-Aitoff.

dash
dash *z1 z2 z3 ...*

See **outline**: this command draws a dashed contour outline instead of a solid line, but is otherwise identical with **outline**. If a contour level appears in both commands, the dashed option takes precedence. Also see **heavy**, **weight**.

heavy
heavy *z1 z2 z3 ...*

See **outline** and **dash**: provides for a heavy contour outlining the colored areas. The command allows the user to highlight certain levels. For example

interval 100
outline
heavy 0
sets equally space levels at 100 units, provides a solid outline for all of them, and gives the level zero a heavy weight. Also see **dash**, **outline**, and **weight**.

file *filename*

The rectangular data array to be contoured is held in a diskfile named 'filename'. This command identifies the file to the program and must be given before a **read** command can be issued. If several arrays exist on one file, there is no need to issue this command more than once. If you issue the command without a filename, the current file will be rewound to the beginning. The terminal can be substituted for a diskfile if * is used as a filename, but of course the file cannot then be rewound.

format *type*

The command specifies the format of the next **read**. The diskfile containing the array may be an ASCII formatted file or a binary (unformatted, sequential access) file. The default is formatted and then type is just blank. If type is *b* this specifies a binary file. See **read** for the structure of binary files. Note that precise formats cannot be specified here (unlike in *plotxy*), and that all formatted files are read with the FORTRAN statement READ (iunit, *) in the program.

height *h*

The height of the finished contour map in inches is specified with this command. If **height** is not issued or h is zero, then the size of the figure is determined from **width** and the natural proportions implied by the dimensions of the array. If neither **height** nor **width** is assigned, the program takes the larger side of the array to be 6 inches long and the other one in proportion according to the dimensions.

interval *dz*
interval *z1 z2*
interval *z1 z2 dz*

If you use **palette**, and omit the **level** command, the levels at which color changes occur will be left to the program to determine. Usually, the variation spans the range of numbers in the array, but if you desire, that range can be changed to the interval (z1, z2) using this command. This is useful if there are several different data sets are to be mapped with the same contour scheme. Alternatively, you can supply the exact values you want through **level** or **above**. If a positive value dz is provided, the color transitions will begin at z1 and march uniformly through the interval, advancing by dz at each step. If only dz is supplied the all contour levels of the form integer*dz in the data range will appear.

note *(x, y) text*
note *(x, y, h) text*
note *(x, y, h, angle) text*

Write the text with the bottom left corner of the first character at coordinates x, y, in inches referred to the bottom left corner of the contour map. If the parameter h is present, it gives the height of the text; otherwise, or if h=0, **size** gives the character height. The fourth, optional parameter is the angle in clockwise degrees that

the note makes with the horizontal. There can be as many **note** commands as needed. Notes are **not** carried forward, however, to later contour maps, so that if required, the notes must be redefined for each new map.

label *text*

An identifying label comprised of the given text will be printed under the map. More elaborate captions, axis labels and titles can be obtained through **note**.

size *h* [strict]

Text takes the character height h, although provision is made for variable heights of notes. If necessary, the numbers labeling the levels in the color key will be reduced in size to avoid overlapping. When large numbers are important (for clarity in overheads, etc) this shrinking can be overridden by including the word *strict* as shown. Then to relieve overcrowding, not all the levels will receive a numerical label.

landscape

To orient the plot so that x runs along the long axis of the paper include this command. Once turned on it cannot be revoked.

level *z1 z2 z3 . . .*

If you use **palette**, the values at which color transitions occur may be specified precisely by the numbers z1, z2, . . . The number of contour levels is simply the number of values supplied. If there are too many values to fit onto a single line, just repeat the command with the remainder of the list. See also **interval**.

lines *filename* [color=X]

Continuous lines may be superposed on the contour map by reading the x-y coordinates from the named file, which is always read to the end-of-file and must contain numerical x-y pairs in the desired order, one pair per line. The coordinates for the plotting are defined by the **axes** command. To break the line, insert a point lying outside the plot limits. The color of the line may be specified by setting X to one of the eight colors: black, white, red, blue, green, yellow, orange, or purple. The command may be repeated with different files to accumulate information from several sources. As usual the x-y line data will be carried forward onto subsequent plots, unless they are all cancelled by calling **lines** without a filename. If further data are input in later plots, any previous set of x-y data is overwritten. This command is not as flexible as the one available in *contour*. When **mapping** is used the coordinates are not transformed, but plotted directly on the map in the form read in. See also **symbol**.

symbol *filename kind h* [color=X]

It is often useful to mark points on the map with symbols. This command identifies a disk file, like the one used in the **lines** command, and causes a symbol to be plotted at each point. The file is always read to the end. Only one symbol is associated with a particular file, its type defined by *kind* and its height *h* in inches. You may read more than one file with further **symbol** commands to get a variety of symbols. The code *kind* conforms approximately to that used in *plotxy*; the order is unfortunately not very logical: 0 square; 1 triangle; 2 octagon; 3 diamond; 4 plus; 5

asterisk; 6 cross; 7 5-ray; 8 Y upside down; 9 pentagon; 10 triangle, base up; 11 hexagon; 12 Y; 13 bar; 14 6-ray; 15 dot; 16 heptagon; 17 circle; 18 large circle (double height); 19 small filled circle; 20 small filled square; 21 small filled triangle. Colors are set by choosing X from the list: black, white, red, blue, green, yellow, orange, purple.

All the symbols are carried forward to the next map if no new **symbol** commands are issued; but previously defined symbols will be cancelled if a new symbol command is included in the commands for a later map – then only the new set will be plotted. All previously defined symbols are deleted if **symbol** is entered without arguments. See also *lines*.

nobar
nobar *off*

Normally a key to the colors is drawn to the right of the map. To prevent this, enter **nobar**. To reinstate the bar use the second form with the word "off" after the command.

orient *options*

There are eight different orientations on a page of an array with sides parallel to the edges of the paper; the user can specify which one by supplying a set of operations in the option list. If the letter N follows the command this is the normal (or numerical) arrangement of the array, oriented as it would be seen in a numerical listing of the input file, with row 1 across the top and column number increasing to the right. T means the array is transposed before plotting. Either of these two can be followed by one or more 90-degree clockwise rotations given by R. In addition you can reflect the array about a horizontal line with H; or reflect about a vertical line with V. All of these letters are in fact operations. Hence you may type NTTRRRVR, which means the numerical array is transposed, transposed again, rotated clockwise three times, reflected in the vertical axis, then rotated again. (After all this, the array is at NH.) The actions on the array are performed in the order written, left-to-right. Thus RV first rotates the array, then reflects it. Having a redundant set of operations means that you can specify the orientation in the most comfortable way.

There is an alternative way of giving the orientation in the **read** command; this is not recommended but is retained for compatibility with earlier versions of *contour*. If both the **orient** command and the old style are used together, the new command takes precedence. If you omit the **orient** command, the default option list is RRR, to retain compatibility.

If you want to plot several different orientations of the same array, you must read the array each time.

outline
outline *z1 z2 z3 ...*
outline *off*

In the first, form run a dark contour line along each color boundary. In the second form, specific contour levels are given, and the outline runs only along those levels. If a level in the list is not in the contour set, it will not be created. To turn off the outlining, use the third form, with *off* as a parameter. Boundaries indicated in this command will also be marked in the color key at the side of the map. See **heavy**

and **dash** for variants on the basic theme. See also **weight**.

output *filename*

An alternative plotfile name to the default *mypost* may be specified with this command; it must be invoked before the first call to **plot** and after that it will be ignored.

palette *p*
palette *0 d q*

Because of the tedium of specifying the color of each region individually with **above**, a number of predefined schemes have been provided and can be invoked with this command. The integer p may be 0 (for a gray scale), 1, 2, etc up to the maximum provided, currently 7. Here is rough description of the current collection: p=1, Blue-Red ranges from deep blue at the low end to dark red at the top; values in the middle of the range are pale; p=2, Bright Spectrum, progresses roughly in spectral order from deep blue, through green, yellow, orange, brown, red and violet; p=3, Rainbow, is a more muted version of p=2; p=4, Harvard, runs from dark blue, light blue, yellow, orange then brown - no green; p=5, Sand, yellow through light brown, all pale colors; p=6, Cartographic, blue though green, yellow, brown all very pale; p=7, Oceanographic, dark deep blue through pale blue.

If you ask for more levels than the palette provides, the HSB coordinates will be interpolated and that may or may not result in distinct colors in the final plot. For more on color choices see **NOTES**.

To enter a customized color palette use the **table** to read values from a file.

When p=0 a gray scale is provided. This can be tuned with the optional numbers d and q. The intensity of the graytone is controlled by the range of values in the array or the values set by **interval**. The number d controls the density of the blackest parts of the plot; d should be no greater than unity in magnitude. If d is positive the darkness runs from zero (pure white) for the smallest array value to a maximum of d (where 1 is completely black) corresponding to maximum values. If contours are to be superposed, d should be no more than 0.7 to prevent the graytone obscuring the contour lines. When d is negative, light tones are used for high values in the array and dark ones for the smaller values.

When a linear mapping is made between gray level and function magnitude the overall aspect of the picture is rather dark. To lighten the plot the second argument q allows stretching of the gray scale through a simple power law. Thus when q is 2 (the default) the darkness value is squared, which has the effect of reducing the number of very dark regions; q need not be an integer but must be positive. You should experiment with these arguments for the best effect.

Obviously, the commands **interval**, **levels**, **above** are alternative ways of specifying the transition levels. If more than one of them is present, the order of precedence is: **above**, **levels**, **interval**, meaning that the **above** command will cancel the effect of the other two if present, and so on. This holds irrespective of the order in which the commands are entered.

plot
plot *x y*

This command instructs the program to perform the contouring of the array. The numbers x and y are usually omitted; see below. All the specifications up to this point are used or defaults assumed if the user is silent on essential matters. What is done is merely to write plotting instructions to the plotfile *mypost* which must be sent to a PostScript graphics output device at a later stage for inspection.

Another contour map may be plotted without restarting the program by reading more data and calling **plot** again. If new array data are input, they supersede the earlier numbers, otherwise the old array will be used. The next map will be placed tastefully above the previous one unless you want something special. The numbers x and y in the argument can be used to specify the coordinates in inches of the new plot origin (bottom-left corner of the map) relative to the old origin. If x and y are given for the first map, the bottom-left corner of the map will positioned at these coordinates with respect to the bottom-left corner of the paper.

show

Under the Sun operating system Fortran can launch a shell process; **show** starts a PostScript display program that sends the contour map image to the terminal.

read *m n*
read *m n r*

The diskfile whose name has already been supplied with **file** is read. When there are 2 or 3 arguments the rectangular array of data is assumed to be arranged in a series of m rows, and each row is n numbers long. In an ASCII file a single row can continue onto as many lines are necessary; in fact one number per line is acceptable. In FORTRAN the file could be created by

```
WRITE (iunit,format) ((A(I,J), J=1,N), I=1,M)
```

or for a binary file invoked by **format** b

```
      DO 1100 I=1, M
1100  WRITE (iunit) (A(I,J), J=1,N)
```

where m = M and n = N. In the formatted case the loops may also be explicit rather than implied. The program reads the file with a single FORTRAN READ statement and an ∗ format for formatted files (see Notes for cautions).

The number r in the command is part of an option to allow orientation of the array, now obsolete; you should use the **orient** command instead. It is described here for completeness. First m may be negative: this means the plotted array has been reflected about a horizontal plane. Also n may be negative: then the plotted array is reflected in a vertical plane. Finally the array (after any reflections) may be rotated through r 90-degree rotations clockwise before it is plotted.

skip *n*

When reading from a diskfile in ASCII the program will skip the first n lines before reading in data. This allows the user to put header information in the file. Skipping occurs immediately after the file is opened, so that when there are several arrays in the same file, no skipping is performed before the second and subsequent data sets are input.

stop

Flushes the plot buffers, closes the plotfile *mypost* and data diskfile then halts the program. You may create as many different contour maps as you desire in one execution of *color* by repeated **plot** commands.

table *filename*

Reads a substitute color table for palette 1 from the named file, which must be an ASCII file in form of a table with three columns comprising successive values of hue, saturation and brightness. Setting **palette** 1 after this command will get the new colors, the old ones having been overwritten.

verbose *n*

As execution proceeds *color* prints a minimum of information. Most printing can be suppressed by setting n to 0. With n set to 1 information is printed that can be useful at a later stage, particularly the actual color values used by the program, which you may want to modify slightly.

weight *w1 w2 w3* The line weights for regular outlines (w1), dashed outlines (w2) and heavy outlines (w3) may be set in units of 0.001 inches. The defaults are w1=w2=4, w3=12. see **outline**.

width *w*

As with **height** this command sets the width of the map in inches. If w is zero or left unset, the width is computed from the height and the natural proportions of the rectangular array.

NOTES

Unlike black-and-white plotting, color rendition depends very strongly upon hardware. PostScript provides a hardware-independent description of the colors, but what those colors actually look like varies from device to device. I have debugged and designed palettes on a 16-inch Sony color monitor. The transition to paper is somewhat unpredictable and some experimentation may be advisable for satisfactory results.

Using **above** you can describe a color precisely through its HSB (Hue-Saturation-Brightness) coordinates. Roughly speaking these have the following meanings. Hue is a number ranging from 0 to 1 that takes the color around the perimeter of a color disk: 0 is pure red, 0.333 is pure green; 0.667 is pure blue and 1.0 is back at red again. Yellow lies between red and green at about 0.13, purple between blue and red at about 0.8. These colors can be modified in their saturation, the second coordinate: roughly speaking, saturation 1 is the pure color, and decreasing saturation adds white or gray (depending on brightness). Thus 0 saturation is a gray tone independent of the hue. The third coordinate, brightness, is self-explanatory: red with a low brightness is brown; zero brightness is always black, independently of hue and saturation.

It is unwise to attempt the representation of too many levels through a large number of colors. I have found 20 to be a reasonable number. When contour levels are unevenly spaced, the color key is also drawn with uneven spacing to reflect this fact. But it can easily happen that closely spaced levels might cause neighboring numerical labels to overlap. *Color* avoids this by distorting the spacing in the color key to provide enough room for a label at every level. If the *strict* option is invoked in the **size** command, some levels will be unlabeled, which may lead to a serious loss of information, and therefore *strict* sizing is not recommended when contours levels are set unevenly.

GRID TRANSFORMATIONS

The following describes some special functions for drawing contour maps in a transformed plane. The easiest way to understand the functions is to consider the example of a map projection: suppose the array represents data on a regular grid of latitude and longitude, but the user wants the final picture in some map projection. This is accomplished by transforming the contour outlines just before they are output.

mapping *n*

Define the kind of mapping to be used: n=1 means uniform mapping from the grid, that is, no transformation is done; n=2 transforms the grid onto an Aitoff equal-area map projection; n=3 maps x and y separately through a mapping read from a diskfile (see **tensor**). In the case of the Aitoff mapping, it is assumed that x coordinate in the original runs through 360 degrees of longitude, and that the y coordinate spans –90 to 90 in latitude. The mapping remains in force until changed.

tensor *filename*

In **mapping** 3 the spacing in the rectangular array between the adjacent columns of numbers (the x spacing) need not be constant, but is defined by the user in the file named in the command; and similarly with rows (y spacing). If the input array has kx columns across (at plot time – recall it may have been rotated first), there must be kx-1 numbers in the file giving the relative spacing between adjacent columns, starting at the left; these must be followed by ky-1 numbers giving the relative spacing in y. The values give the ratio of the separations not the actual intervals since the size of the final plot is determined by the commands **width** and **height**. If the rotation number in **read** is odd kx=n and ky=m; otherwise they are interchanged. The reason for the name is that this is a tensor-product transformation.