CONTOUR
A Program for Contouring Smooth Functions Defined on a Regular Array

Robert L. Parker
David W. Caress
Peter Salameh

INTRODUCTION

When a contour map of a smooth function is required and the values are known on a rectangular array, *contour* is able to provide a pleasing plot with the minimum amount of effort. Those already familiar with *plotxy* will know the principles of operation. The user types a series of commands in a simple language that tells *contour* what to do. Most of the parameters have default values so that getting a quick look at an array is easy. Suppose the file MAT contains data arranged in 30 rows each of 10 numbers; then to obtain a contour plot one only need enter

```
file MAT
orient N
read 30 10
plot
stop
```

Here the ASCII input file is specified by **file**; the second command tells the program to plot the array oriented as it would be in a numerical listing, with the rows running horizontally; the program reads the data with **read**. The third number in the **read** statement controls the orientation of the map, in this case the map appears in the orientation of a numerical listing of the array. Then, because no explicit contour 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 plotfile named *mypost* has been generated which can be displayed at the terminal or on the laser printer with standard plot filters (see LOCAL IMPLEMENTATION).

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, MAT in the example above. It is easy to plot several maps on the same page, arrange different size and contour intervals, thicken the level lines of some contours, delete labeling at some levels as well as re-orient the picture. Titles, notes and captions may be supplied. The catalog below describes the various features. To find something look in the index.

COMMAND CATALOG

All commands must begin in the first position of the line; lines beginning with a space are ignored. Each command may be abbreviated to its first 4 letters although the full word is given in the catalog. The command line may be up to 120 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.

affine a b

Immediately after input, the data values to be contoured are subjected to a transformation: znew = a * z + b. This is an affine mapping. All subsequent commands (like **level** or **null**) act on the mapped values, not the originals. Omitting a and b returns to the default state of no mapping.

axes x1 x2 y1 y2 kindax

With this command numerals and tick marks are given to the x and y axes. The annotation for the x axis labels it as if it extended between x1 and x2, irrespective of the array dimensions or its actual length in inches. If you set x1 equal to x2 the x axis is not annotated. Similarly for y1 and y2. The optional integer kindax is used if logarithmic axes are wanted: kindax=0 means both axes are to be linear, 1 means log x axis, 2 means log y axis, and 3 means both x and y are to be logarithmic. Linear axes are substituted for log ones if negative values or reversed axes are specified.

border

Normally, a rectangular box is drawn around the contour map as a border. To suppress the box enter this command. To restore the border at a later stage enter **border** again; the command toggles the state back and forth.

caption text

The contour map is normally surrounded by a rectangular box. If you wish, a caption can be written under the box, left justified, containing the material 'text'. Also, unless deliberately changed, the font of all the lettering is the one specified at the beginning of the caption. (For a full description of the available fonts and math formatting capabilities see *plotxy* documentation.) The axis numerals are always in the font specified by the caption. If a long caption is desired, the user can enter up to nine additional lines of text, indicating **caption** material continues on the next line by placing an ampersand (&) at the end of the current line. The text is justified to the width of the box. Long captions do not carry forward to the next contour map.

code k1 k2 k3 . . .

The contour lines may be plotted with or without labels; they may be heavy, with regular weight, dashed or hatched. Each contour level is associated with an integer specifying the options as follows. The magnitude of the integer describes the kind of line: 0 or 1 means a line of ordinary weight; 2 means a heavy line; 3 to 9 means a dashed line, 10 a line hatched on the down-hill side, 11 hatched line with prongs uphill. The dashes become lighter as code is increased – the practical limit is reached at code 5 since higher values yield virtually indistinguishable lines. The sign is used as a signal for the label: a negative integer means no label is required. The label is also omitted if there is no room for it on the contour. The code numbers are associated with the contour levels in the order that they are defined in **level**, which need not be a monotone sequence. If the list of codes requires more than a single line, just continue onto subsequent lines as they are needed. To force one contour style for all levels use code ... has described next.

Especially when **interval** is used, a repeating pattern of codes can be useful, for example alternating labeled and unlabeled contours, and that would be tedious to specify, particulary if the number of levels is unknown. If the last code is the ellipsis ... this means repeat the pattern defined so far. Thus

```
code 1 -5 -5 ...
```

is equivalent to

```
code 1 -5 -5 1 -5 -5 1 -5 -5 1 -5 -5 1 -5 -5 1 -5 -5 1 -5 -5 etc
```

and means a labeled contour followed by two dashed unlabeled contours throughout the contour set. If ... does not terminate it, the the list continues on with regular contours, namely code=1.

color k1 k2 k3 . . .

If color plotting devices are available, then the color of the contour lines can be set using this command. Each contour level is associated with an integer specifying the color in the order specified in **level**, exactly as in the **code** command. For color values see **Color**.

Color ipen

With the upper-case C, the command sets the pen number or color id used for drawing the axes and labels and x-y line data. Use of the pen number varies with plotting device; the standard associated with IGPP PostScript implementation is: 0, 1 black; 2 red; 3 blue; 4 green; 5 brown; 6 orange; 7 yellow; 8 purple; 9 gray; 10 white (useful for superposition on other plots). The color of the axes, their labels and numerical annotation is set by the last invocation of **Color** before the **plot** call. Notes are colored according to the previous call to **Color**, so that interspersing **Color** and **note** calls leads to multicolored notes. Also line thickness can be set through **Color**: as in *plotxy*, with PostScript output one may add 1000 times n to ipen to make subsequent lines n/1000 inches thick; the default line thickness is 0.004 inches.
Default ipen = 1.

fancy crit

The contours are plotted in a fancy style: contours above crit as regular lines, those below are dashed; the contour exactly at crit drawn heavy. If the contour level crit is not in the list, one is added. The command **code** cancels the **fancy** option, and so does **fancy** when the parameter crit is omitted.

file filename

The rectangular array to be contoured (or other data to be plotted – see **read** and **xydata**) is held on 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 are present in one file, there is no need to issue this command more than once. If you do, or you issue the command without a filename, the file will be rewound to the beginning. The terminal is substituted for a diskfile when ∗ is used as a filename.

format type

The command specifies the format of the next **read**. The diskfile containing the array or x-y data 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 precise formats cannot be specified here (unlike in *plotxy*), and all formatted files are read with the FORTRAN statement READ (iunit, ∗) in the program.

Sometimes map data are provided in triples, that is, x-y-h values, where h might be local elevation. (Of course, *contour* cannot generate a map from such data; they must be gridded first by *zzgrid*, for example.) To make it easy to plot the site coordinates (x, y) on the contour map, the parameter type may be set to the numeral 3, meaning that when x-y data are read, they come in groups of three numbers; the third value in each triple is ignored by *contour*. To revert to x-y pairs, use a numeral 2.

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 dz

Contours will be plotted with a spacing dz at integer multiples of this unit. With three arguments the contour levels assume the form z1, z1+dz, z1+2dz, etc. up to z2. See also **levels**.

landscape

In the PostScript implementation only, inclusion of this command before the first **plot** call causes all output to be made in the landscape mode, that is with x along the long side of the paper.

letter h1 h2 h3 h4 h5

The height of the numerals of the contour labels is h1 in inches. The others specify the letter heights of: the title, the axis numerals, the axis labels, the caption. (The order after the contour labels can be remembered as progressing down the page.) The normal values are 0.08 and 0.1 inches for the others. Alternatively, the text itself can set its own height with an initial phrase like \0016\ which means make any text that follows 16/100 inches high.

level z1 z2 z3 . . .

The actual values to be contoured are specified by the numbers z1, z2, . . . The number of contour levels is simply the number of values supplied. The label and line integers (see **code**) are in one-to-one correspondence with this list which does not have to be in any special order. If there are too many values to fit onto a single line, continue onto as many additional lines as you need. If **level** is omitted, the program chooses its own contour values according to a crafty algorithm. Suppose you have issued **level** (perhaps for an earlier array) and would now like to revert to the choice of the program: just enter **level** * and your original list is canceled.

You can also use ellipsis ... to save typing a long list: *contour* will take a current positive increment and apply it until the final value is reached. Thus

```
levels -0.05 0 1 2 ... 10
```

creates a set of 12 levels, beginning with -0.05, 0, then running from 1 to 10 with unit increment. If no terminal value is provided, 10 addition levels are generated. Also

see **interval**.

note (x y [in] [r c]) text
note (x y ht [in] [r c]) text
note (p q x y [in] [r c]) text
note (p q x y ht theta color [in] [r c]) text

In its simplest form (first version; two numerical arguments) the characters of text (up to 80 characters) are plotted horizontally on the graph at the coordinates x, y. If the optional **in** appears the coordinates refer to the bottom left corner of the first text character, measured in inches from the intersection of the axes; otherwise x, y are in the units of the axes. When 3in is present the coordinate may be outside the frame surrounding the plot, but if that option is absent, the note will be ignored. If the optional r is included, the note is right justified, meaning the coordinates (x, y) refer to the base of the end of the line; similarly, if c appears, the text is centered above (x, y). Notice the parentheses surrounding the coordinates are mandatory. In the second version (three numerical arguments) the height of the plotted characters is the value ht inches; subsequent calls without the third argument inherit the height. With four arguments p, q give the location of an arrow head whose tail is placed tastefully near the text, which is positioned as before. The full-blown command allows the text to be rotated by the amount theta degrees counterclockwise and colored according to color. Up to 20 separate notes may be input. To clear the notes enter **note** and a blank command field – this must be done explicitly for new graphs with a fresh set of notes. If the arrow has zero length none is drawn; if there is no text but there is an arrow it is drawn between (x,y) and (p,q). See **Color** for instructions on how to obtain colored notes.

null zlow zhigh

This command allows the user to restrict contouring to regions with values between the given bounds. Any array value outside the interval [zlow, zhigh] is omitted from the contouring process – the boundaries of the **null**ed region are treated like the edge of the grid, so contours may begin and end there. Giving the command **null** by itself turns off the null. When **null** has been invoked, automatic level setting ignores array values outside the bounds.

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 in two different ways. First by giving the name of the desired orientation chosen from this list (only the first letter of the name is needed)

numerical: Oriented as numerical listing of the array.
vertical: The array is plotted reflected in a vertical line.
horizontal: The array is plotted reflected in a horizontal line.
upside-down: The array is plotted after 180 degree rotation.
clockwise: The array is rotated clockwise by 90 degrees before plotting.
anticlockwise: The array is rotated 90 degrees anticlockwise.
transpose: The matrix transpose is plotted.
kombo: Combination of c first, then h; no easier description.

The second method is to supply a set of operations in the option list, that reorient the original array. The letter N denotes the normal (or numerical) arrangement of the

array, oriented as it would be seen in a numerical listing of the input file, just like n. N is the default orientation, assumed if the **orient** is not given. T means the array is transposed before plotting. These two orientations can be followed by one or more 90-degree clockwise rotations denoted 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 represent operations performed on the array after it has been read. So you may type NRRVH which means the original array is is rotated clockwise twice, then reflected in the vertical axis, then in the horizontal. (It is an exercise to find the final state after this chain.) The actions on the array are performed in the order written, left-to-right, not the normal mathematical order of operators. Thus RV first rotates the array, then reflects it. Having a redundant set of multiple operations means that you can specify the orientation in the most comfortable way, but in fact it is never necessary to use more than two operations to achieve any orientation.

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, because of compatibility.

The **orient** command must come before the **read** that inputs the array. If you want several different orientations, you must read the array each time.

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.

plot
plot x y

This command instructs the program to perform the contouring of the array, draw the box, write the title and caption, etc. 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 page description instructions to the plotfile *mypost* which must be sent to a printer of the screen 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 is 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 one.

read m n
read m n r
read m

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. 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.

The command **read** may be followed by a single integer m. Then instead of a matrix, the data comprise a series of lines or points to be superimposed on your contour diagram. The numbers are read from the current diskfile as m x-y pairs located in a coordinate frame defined by the current **axes** which must be set up before this kind of input can be done. The maximum number of such x-y series allowed can be found by invoking **status**. Normally, a solid straight line will join successive points but alternatives can be set with **xydata**. Also see **format**. When a point lies outside the current frame, it causes the line to break; the line will restart at the next point within the frame. If you plot more than one map in a run, the x-y data will be carried onto the next plot unless you purge them: **read** with m=0 removes the current x-y data series. Such superimposed material inherits its color from the previous f3Color command.

save [filename] [-off] Save the coordinates of the contours to the named disfile or, if the name is omitted, to a file called fort.12. If the **axes** command has been issued with both x and y definitions then the coordinates referred to those axes. Otherwise the coordinates are in inches with the origin at the bottom left. Three numbers are written on each line of the ASCII file: x and y, the coordinates, followed by z, the contour value. Breaks between contour lines are indicated by blank lines in the file.

Contours are accumulated in the named file from all subsequent graphs, until turned off with the -off option, or the name is switched to something else.

segment ns

Segment specifies the minimum number of segments a contour must have in order to be plotted. This command allows the user to suppress small contour loops. The default value is ns = 0.

skip ns

Skip specifies the number of initial lines to skip in an ASCII data file before reading begins; this enables the user to include heading information in the file. The default value is ns = 0.

smooth ks

Smooth saves the smoothing parameter ks. If ks > 1 then contour segments are drawn as smooth arcs with ks pieces per segment. If ks = 1 then the contours are

drawn as simple line segments. The default smoothing parameter is 15.

status

You may discover the present values of your contour levels, or those the program assigns, the upper and lower bounds of the array and all the other options by issuing this command. The command also lists some of the critical parameter limitations (largest array size, maximum number of allowed contour levels etc.) of the version of the program you are executing.

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 *contour* by repeated **plot** commands.

title text

The material 'text' is written centered above the box. The text may be up 115 characters in length.

width w

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

xlabel text

The material 'text' is written centered under the contour map but above the caption.

xydata k
xydata k h

When x-y data pairs are being input with **read**, the plotted points are interpolated with solid straight lines of regular weight when k is 0 or 1 (or this command is omitted). Heavy lines are invoked by setting k=2 and dashed lines with 3, 4 and higher, exactly according to the **code** convention. The specification holds for subsequent **read** commands that read x-y data, but does not alter the codes for contours. If more than one contour map is drawn in a single run, the xydata data carry forward from the earlier maps, unless they are purged with
read 0
Finally, discrete symbols can be plotted at the coordinates if h is present and is assigned the symbol height in inches; then the meaning of k is changed to become a symbol type following the *plotxy* convention:

0 square	8 upward arrow	16 small circle
1 triangle	9 hourglass	17 circle
2 octagon	10 campstool	18 large circle
3 diamond	11 hexagon	19 small filled disk
4 plus	12 Y	20 small filled square
5 asterisk	13 vertical bar	21 small filled triangle
6 cross	14 star of David	
7 barred square	15 dot	

ylabel text

The material 'text' is written centered vertically at the left side of the contour map.

NOTES

There are various intrinsic limits in the program that cause it to fail but, it is sincerely hoped, an informative message will be printed first. To see what limitations are in force for your version of the program enter the **status** command.

If a transposed array has been written (which happens if you use the FORTRAN statement WRITE (iunit,*) A for example) it can be read back in its regular form by the command

```
orient T
read  n m
```

where m is the original row dimension and n the original column dimension, that is, A was dimensioned A(m,n).

It may happen you would like x-y data on your map, but would prefer a picture without the annotated axes. After having **read** in the x-y data, cancel the reading coordinate frame with

```
axes 0 0 0 0
```

and the final contour map will be unadorned. It is equally possible to plot quite different numerical axes on the finished map from the ones used to read x-y data. Beware, though: **note** expects and uses the axes at plot time unless you have used the inches specification in every one.

Control of the lettering font is achieved by the same mechanism as in *plotxy* – one of the phrases \ sim \ \ dup \ \ com \ \ ita \ switches the subsequent text to the corresponding font. A uniform font is obtained by setting it in the **caption** unless specific changes are made in the **title** etc; in any case the contour label numerals are in the same font as the caption. There need be no text in the caption besides the font phrase. The axis numerals are in the same font as the associated labels.

The contouring algorithm, written by Dr David P. Anderson of University of Wisconsin, is generally quite satisfactory, but very occasionally with a wild data set, inconsistent contours that cross each other, or even wander outside the box can be generated. To solve this problem you should set ks=1 in **smooth**.

AN EXAMPLE

The following commands produced the first contour plots at the end of this document. Two diagrams are produced, one with default parameters, the other somewhat cleaned up with added labels and axes.

```
width 4.5
title Default parameters
file *
orient N
read 13 11
  -31  -60 -100 -150 -199 -231 -236 -214 -176 -134  -97
   -7  -38  -92 -170 -259 -326 -342 -305 -240 -174 -119
```

```
   42   23  -32 -143 -301 -444 -493 -434 -326 -222 -143
  130  154  142   35 -216 -523 -673 -599 -427 -272 -166
  253  373  510  562  299 -311 -763 -746 -519 -315 -184
  386  638 1042 1527 1594  632 -498 -763 -559 -336 -193
  474  822 1440 2346 2909 1851   63 -607 -523 -329 -192
  477  812 1385 2181 2641 1770  249 -436 -444 -301 -184
  402  638  986 1367 1445  869   54 -344 -361 -262 -170
  293  421  570  669  584  261 -101 -281 -285 -219 -151
  192  246  287  280  185   16 -145 -223 -219 -177 -130
  115  131  130   98   30  -59 -135 -170 -166 -140 -108
   64   63   50   20  -23  -72 -111 -129 -126 -109  -88
plot 1 2

 Begin a second contour map
levels -750 -500 -250 0 250 500 750 1000 1250
          1500 1750 2000 2250 2500 2750
code  5   5  5 2 -1  1  -1 1  -1  1  -1 -1 -1 -1
caption \it\
title Magnetic anomaly of a dipole  \mu\
axes -5 5 -6 6
xlabel East km from source
ylabel North km from source

plot 6 0
stop
```

LOCAL IMPLEMENTATION

On the Mac OS X a file *mypost* is generated by the program, which can be previewed with the standard utility *gv* and if needed, sent to a laser printer with the Unix utility *lpr*. This arrangement allows the most flexible combination with other programs like *color*, a shaded color contouring program written along the same lines as *contour* and *plotxy*.

INDEX

additional lines **read**, **xydata**
annotation **axes**, **note**, **xlabel**, **ylabel**, **note**
array dimensions **read**
array orientation **orient**
automatic levels **level**
binary files **format**, **read, NOTES**
blank areas **null**
boundary box **border**
caption **caption**
colored contours **color**
colored axes, notes, titles **Color**
contour levels **level**
dark lines **code**

dashed contours **code fancy**
dashed lines **xydata**
data format **read**, **format**
datafile **file**
displaying results LOCAL IMPLEMENTATION
down-hill hatch **code**
ellipsis **code**, **interval**
embellishments **xydata**, **note**
equally-spaced contours **interval**
fonts **NOTES**
forbidden areas **null**
heavy lines **code fancy**
label font **caption**, **NOTES**
labels **code**, **xlabel ylabel**
letter height **letter**
map size **height**, **width**
masking unwanted region **null**
maximum array size **NOTES**
more than one map **plot**
notes **note**
numeral height **letter**
orientation **orient**
origin of map **plot**, **NOTES**
plotfile **plot**
reflection **orient**
rotation **orient**
scaling data values **affine**
several maps **plot**, **stop**
size **height**, **width**
title **title**, **caption**
transposed array **orient**
unlabeled contours **code**