Subject: Polycontour for unclosed contours
Posted by sterne on Wed, 13 Jan 1993 01:16:42 GMT
View Forum Message <> Reply to Message

patrick ryan writes:

> the problem i still have is that polycontour bites.
> if anyone has a color contouring routine which knows how
> to handle unclosed contours, please post it.

Here is my fix to the POLYCONTOUR routine. There are basically 2 things wrong with the routine that comes with PV-WAVE (and, I imagine, IDL).

1. In the standard routine, contours are filled from the lowest value up. That's fine if you have a positive gaussian or a simple mountain, but if you have a dip, the upper value will shade over the lower contours. This routine plots in descending AREA to eliminate this problem.

2. The standard routine cannot treat open coutours. This routine deals with them by closing the open contour in the clock-wise and the counterclockwise directions and evaluating the value of the function inside each of these to determine the appropriate shading index for each of these 2 contours. This requires an extra argument in the POLYCONTOUR subroutine call - the two dimensional array which defines the coutour plot.

This routine reads the file generated by CONTOUR,PATH=Filename, closes the open contours as described above, and writes out all contours to be shaded in Filename.temp. I've writen this for a Unix system, so the I/O will need to be modified for VMS.

This routine uses a bilinear interpolation routine, BLINEAR, which I have also included. I cannot say anything good about BILINEAR routine which came with WAVE, and the Johns Hopkins routines from which I took this bilinear interpolator have been a godsend!

Enjoy,
Phil

```
;
; $Header: /usr5/wave/release_2.2/lib/RCS/polycontour.pro,v 1.1 90/01/23 16:44:33 wave Exp $
;
pro polycntr1, filename, image, color_index=color_index, pattern = pat,$
          pos = pos
;+
```

```
; NAME:  POLYCNTR1
; PURPOSE:
; Fill the contours defined by a path file created by CONTOUR.
; CATEGORY:
; Graphics.
; CALLING SEQUENCE:
; POLYCNTR1, Filename, Image [, COLOR_INDEX = color index]
; INPUTS:
; Filename = name of file containing contour paths.  This
;  file must have been made using CONTOUR, PATH=Filename, ...
;       Image = image file used to generate the original contours.
; KEYWORD PARAMETERS:
; COLOR_INDEX = color index array.  Element i contains the color
;  of contour level number i-1.  Element 0 contains the
;  background color.  There must be one more color index than
;  levels.
; PATTERN = optional array of patterns, dimensioned (NX, NY, NPATTERN).
; POS = 4 element array, [xmin,ymin,xmax,ymax] in normal coordinates.
;            Defaults to !p.position values if not specified.
; OUTPUTS:
; The contours are filled on the display using normalized
; coordinates using the POLYFILL procedure.
; COMMON BLOCKS:
; None.
; SIDE EFFECTS:
; Contour temp file filename.temp created in current directory.
; Display is updated.
; PROCEDURE:
; Scans file.  Eliminates contours of 2 points with a check
;       that both points are the same.  Open contours are closed clockwise
; and counterclockwise around the plot border and both contours
;       are written to the contour temp file.  Area of the two plots
;       and starting bytes in the temp contour file are both noted.
; Image is used to determine which of these 2 contours encloses
; the higher ground.  Closed contours are written directly to
; the temp contour file. Areas are sorted and contours are
; reread from the temp contour file and drawn in order of
; decreasing area.
; EXAMPLE:
; Create a 8 by 8 array of random numbers,
; polycontour it, then overdraw the contour lines:
; b = FLTARR(10,10)  ;Big array of 0's
; b(1,1) = RANDOMU(seed, 8,8) ;Insert random numbers
; CONTOUR, b, /SPLINE, PATH = 'path.dat' ;Make path file
; POLYCONTOUR, 'path.dat'  ;Fill contours
; CONTOUR, b, /SPLINE, /NOERASE ;Overplot lines & labels
;
; Suggestion:  Use TEK_COLOR to load a color table suitable
```

```
;  for viewing this display.
; MODIFICATION HISTORY:
; DMS, AH, January, 1989.
; PAS, LLNL, October 1991.  Converted to order on area of contour
; PAS, LLNL, November 1992. Amended to treat open contours.
;-

header = {contour_header,$
 type : 0B, $
 high_low : 0B, $
 level : 0, $
 num : 0L, $
 value : 0.0 }

max = 0
if n_elements(color_index) eq 0 then color_index = indgen(25)+1
asize = 100  ;# of elements in our arrays
n = 0
cval = intarr(asize) ;Contour index
carea= fltarr(asize)
cstart = lonarr(asize) ;Starting byte of record
if n_elements(pos) eq 0 then pos = !p.position
xvert = [pos(0),pos(2),pos(2),pos(0)]
yvert = [pos(1),pos(1),pos(3),pos(3)]
openr,unit,filename,/get_lun
openw,unit2,filename+'.temp',/get_lun
while (not eof(unit)) do begin  ;First pass
   if n eq asize then begin  ;Expand our arrays?
      cval = [cval,cval]  ;Yes, double them
      cstart = [cstart,cstart]
      carea = [carea,carea]
      asize = 2 * asize
   endif
;
   a = fstat(unit2)   ;File position
   readu,unit,header   ;Read header
   if (header.num eq 2) then begin
      xyarr = fltarr(2,2)
      readu,unit,xyarr
      tempval = total(abs(xyarr(*,0)-xyarr(*,1)))
      if tempval gt 1.e-05 then print, 'two pt contour, diff pts'
   endif else if (header.type eq 0) then begin  ; Open contours
      xyarr = fltarr(2,header.num)  ; Define point to read
      header1 = header
      header2 = header
      header1.type = 1
      header2.type = 1
      readu,unit,xyarr
```

```
      side1 = 4
      if abs(xyarr(0,0) - xvert(0)) lt 1.e-5 then side1 = 3
      if abs(xyarr(0,0) - xvert(2)) lt 1.e-5 then side1 = 1
      if abs(xyarr(1,0) - yvert(0)) lt 1.e-5 then side1 = 0
      if abs(xyarr(1,0) - yvert(2)) lt 1.e-5 then side1 = 2
      if side1 eq 4 then print,$
           'error open contour side1, n= ',n,'   vertex=',xyarr(*,0),$
           '   mins and maxes = ',pos
      side2 = 4
      hn = header.num - 1
      if abs(xyarr(0,hn) - xvert(0)) lt 1.e-5 then side2 = 3
      if abs(xyarr(0,hn) - xvert(2)) lt 1.e-5 then side2 = 1
      if abs(xyarr(1,hn) - yvert(0)) lt 1.e-5 then side2 = 0
      if abs(xyarr(1,hn) - yvert(2)) lt 1.e-5 then side2 = 2
      if side2 eq 4 then print,$
           'error open contour side2, n= ',n,'   vertex=',xyarr(*,hn),$
           '   mins and maxes = ',pos
; same side:
      if side1 eq side2 then begin
        vadd = fltarr(2,4)
  vadd(0,*) = shift(xvert,-side1-1)
  vadd(1,*) = shift(yvert,-side1-1)
  dotdir = total((xyarr(*,hn)-xyarr(*,0))*(vadd(*,0)-vadd(*,3)))
  if dotdir gt 0 then begin
    xyarr2 = fltarr(2,hn+2)
    xyarr2(*,0:hn) = xyarr
    xyarr2(*,hn+1) = xyarr(*,0)
    xyarr1 = fltarr(2,hn+6)
    xyarr1(*,0:hn) = xyarr
    xyarr1(*,hn+1:hn+4) = vadd
    xyarr1(*,hn+5) = xyarr(*,0)
    header1.num = hn+6
    header2.num = hn+2
  endif else begin
    xyarr1 = fltarr(2,hn+2)
    xyarr1(*,0:hn) = xyarr
    xyarr1(*,hn+1) = xyarr(*,0)
    xyarr2 = fltarr(2,hn+6)
    xyarr2(*,0:hn) = xyarr
          xyarr2(*,hn+1:hn+4) = reverse(vadd,2)
    xyarr2(*,hn+5) = xyarr(*,0)
    header1.num = hn + 2
    header2.num = hn + 6
  endelse
     endif else begin
; different sides
  vadd = fltarr(2,4)
  vadd(0,*) = shift(xvert,-side2-1)
```

```
  vadd(1,*) = shift(yvert,-side2-1)
  ivt = (-side2+side1+4) mod 4
  xyarr1 = fltarr(2,hn+2+ivt)
  xyarr2 = fltarr(2,hn+6-ivt)
  xyarr1(*,0:hn) = xyarr
  xyarr2(*,0:hn) = xyarr
  xyarr1(*,hn+1:hn+ivt) = vadd(*,0:ivt-1)
  xyarr1(*,hn+ivt+1) = xyarr(*,0)
  tmp = reverse(vadd,2)
  xyarr2(*,hn+1:hn+4-ivt) = tmp(*,0:3-ivt)
  xyarr2(*,hn+5-ivt) = xyarr(*,0)
  header1.num = hn + 2 + ivt
  header2.num = hn + 6 - ivt
     endelse

; Check which of xyarr1/2 is higher by adding and subtracting
; 0.005 of the +ve side vector to the last xyarr point.
; adding puts point in xyarr1, subtracting puts it in xyarr2.
; if xyarr1 pt > xyarr2 pt, then color = c for xyarr1 and c-1 for xyarr2
; reverse for xyarr1 pt < xyarr2 pt.

     refx1 = xyarr(*,hn) + 0.005*(vadd(*,0)-vadd(*,3))
     refx2 = xyarr(*,hn) - 0.005*(vadd(*,0)-vadd(*,3))
     dtax1 = [(refx1(0)-!x.s(0))/!x.s(1),(refx1(1)-!y.s(0))/!y.s(1)]
     dtax2 = [(refx2(0)-!x.s(0))/!x.s(1),(refx2(1)-!y.s(0))/!y.s(1)]
     val1 = blinear(image,[dtax1(0)],[dtax1(1)])
     val2 = blinear(image,[dtax2(0)],[dtax2(1)])
     if val1(0) gt val2(0) then begin
  c = fix(header.level)
  header1.high_low = 1
  max = max > c
  cval(n) = c     ;Contour index
  cstart(n) = a.cur_ptr    ;Position
  carea(n) = poly_area(xyarr1(0,*),xyarr1(1,*))
  writeu,unit2,header1    ;Write header
        writeu,unit2,xyarr1
  n = n + 1
  a = fstat(unit2)    ;File position
  header2.high_low = 0
  cval(n) = c - 1    ;Contour index
  cstart(n) = a.cur_ptr    ;Position
  carea(n) = poly_area(xyarr2(0,*),xyarr2(1,*))
  writeu,unit2,header2    ;Write header
        writeu,unit2,xyarr2
  n = n + 1
     endif else begin
  c = fix(header.level)
  header1.high_low = 0
```

```
  max = max > c
  cval(n) = c - 1    ;Contour index
  cstart(n) = a.cur_ptr    ;Position
  carea(n) = poly_area(xyarr1(0,*),xyarr1(1,*))
  writeu,unit2,header1    ;Write header
        writeu,unit2,xyarr1
  n = n + 1
  a = fstat(unit2)    ;File position
  header2.high_low = 1
  cval(n) = c     ;Contour index
  cstart(n) = a.cur_ptr    ;Position
  carea(n) = poly_area(xyarr2(0,*),xyarr2(1,*))
  writeu,unit2,header2    ;Write header
        writeu,unit2,xyarr2
  n = n + 1
     endelse
   endif else begin

;Color to draw
      c = fix(header.level) + fix(header.high_low) - 1
      max = max > c
      cval(n) = c    ;Contour index
      cstart(n) = a.cur_ptr   ;Position
      xyarr = fltarr(2,header.num)  ;Define point to skip
      readu,unit,xyarr
      carea(n) = poly_area(xyarr(0,*),xyarr(1,*))
      writeu,unit2,header   ;Write header
      writeu,unit2,xyarr
      n = n + 1
   endelse
endwhile
free_lun, unit

cval = cval(0:n-1)   ;Truncate
cstart = cstart(0:n-1)
carea = carea(0:n-1)
order = rotate(sort(carea),2)  ;Subscripts of order
for i=0,n-1 do begin   ;Draw each contour
 j = order(i)   ;Index of record
 point_lun,unit2,cstart(j)
 readu,unit2,header  ;Reread header
 xyarr = fltarr(2,header.num) ;Define points
 readu,unit2,xyarr  ;Read points
 col = cval(j)   ;Drawing color
 col = color_index(col+1)

 if n_elements(pat) ne 0 then begin
  s = size(pat)
```

```
  if s(0) ne 3 then begin
   print,'POLYCONTOUR - pattern array not 3d'
   return
   endif

  polyfill,/NORMAL, pattern=pat(*,*, i mod s(3)), xyarr
 endif else $
   polyfill, /NORMAL, color= col,xyarr ;Fill contour
 endfor
free_lun, unit2   ;Done
command = 'rm '+filename+'.temp'
spawn,command
end


 ;---------------------------------------------------------- --
;+
; NAME:
;       BILINEAR
; PURPOSE:
;       Do bilinear interpolation into a 2-d array.
; CATEGORY:
; CALLING SEQUENCE:
;       zout = bilinear(zz, x, y)
; INPUTS:
;       zz = 2-d array to interpolate.            in
;       x,y = x,y coordinates of desired points.  in
; KEYWORD PARAMETERS:
; OUTPUTS:
;       zout = resulting interpolated values.     out
; COMMON BLOCKS:
; NOTES:
;       Notes: x and y may be any shape.
; MODIFICATION HISTORY:
;       R. Sterner, 16 Oct, 1990
; P.A. Sterne, 18 Sept. 1991 to accept vector inputs x, y
;                          and output array zz
;
; Copyright (C) 1990, Johns Hopkins University/Applied Physics Laboratory
; This software may be used, copied, or redistributed as long as it is not
; sold and this copyright notice is reproduced on each copy made.  This
; routine is provided as is without any express or implied warranties
; whatsoever.  Other limitations apply as described in the file disclaimer.txt.
;-
 ;---------------------------------------------------------- --

 function blinear, zz, xin, yin, help=hlp
```

```
 if (n_params(0) lt 3) or keyword_set(hlp) then begin
   print,' Do bilinear interpolation into a 2-d array.'
   print,' zout = bilinear(zz, x, y)'
   print,'   zz = 2-d array to interpolate.           in'
   print,'   x,y = x,y coordinates of desired points.  in'
   print,'   zout = resulting interpolated values.     out'
   print,' Notes: x and y may be any shape.'
   return, -1
 endif
;
 nx = size(xin)
 ny = size(yin)
;
 if nx(0) lt 2 then begin
    ones = replicate(1,ny(1))
    x = xin#ones   ; create nx*ny array of x coords
 endif else begin
     x = xin
 endelse
;
 if ny(0) lt 2 then begin
    ones = replicate(1,nx(1))
    y = ones#yin   ; create nx*ny array of y coords
 endif else begin
     y = yin
 endelse
;
 x1 = floor(x)  ; Get four pixels around desired pt.
 x2 = x1 + 1
 ya = floor(y)
 yb = ya + 1
 fx = (x-x1)/(x2-x1) ; Fractional position inside 4 pts.
 fy = (y-ya)/(yb-ya)
 za = fx*zz(x2,ya) + zz(x1,ya)*(1-fx) ; Interpolate in X.
 zb = fx*zz(x2,yb) + zz(x1,yb)*(1-fx)
 z = fy*zb + za*(1-fy)   ; Interpolate in Y.
 return, z
 end
```

--

Philip Sterne | sterne@dublin.llnl.gov
Lawrence Livermore National Laboratory | Phone (510) 422-2510
Livermore, CA 94550 | Fax (510) 422-7300