
Subject: Fill Open Contour Routine

Posted by [sterne](#) on Wed, 22 Jun 1994 20:06:02 GMT

[View Forum Message](#) <> [Reply to Message](#)

I've received numerous requests for the routine, so, here is my version of filled contours which works with my (really old) version of PV-Wave (3.1), and has been known to work with IDL as well.

If you have any comments on this, if you like it, if it breaks, whatever, please let me know, but please read the comments below if you have problems.

I've supplied two routines below. The first, CONTOUR1 is a wrapper around CONTOUR and POLYCONTR1. You can make filled contours by using a keyword /FILL in CONTOUR1 (see example below). Note that Wave had (may still have) a limit on the number of keywords a user-written routine can have, so if you have problems, you may have to comment some out (about 10). I've supplied it in the PV-Wave mode, so if your IDL/WAVE supports more keywords than mine, just uncomment in the appropriate places.

The second routine is POLYCNTR1, which does all the hard work of closing and filling the contours. Unfortunately, IDL and PV-Wave have a different convention for writing the contour file (since IDL version 3.1.1, maybe a little before that even), so look for the line

```
; ***** IMPORTANT *****
```

in POLYCNTR1, and follow the instructions there if you are running IDL. The posted version is configured for the PV-Wave conventions.

Finally, load them and try the following:

```
WAVE> a = randomu(seed,10,10)
WAVE> contour1,a,/xst,/yst,/spline,/fill
```

Routines follow below.

Phil

```
-----
;+
; NAME: CONTOUR1
; PURPOSE: Generate contour plot with or without filling the contours.
;         Provides a wrapper which calls CONTOUR and optionally POLYCONTR1.
; CATEGORY: Display
; CALLING SEQUENCE: Contour1, Z [, X, Y ,fill=fill and lots of other keywords]
; INPUTS:
;
; Z: Two dimensional data array used to generate the contours.
```

```

;
; OPTIONAL INPUT PARAMETERS:
;
; X = vector or 2 dimensional array specifying the X coordinates
; for the contour surface. If a vector, each element specifies the
; X coordinate for a column of Z (Example: X(0) specifies the X
; coordinate for Z(0,*)). If X is a two-dimensional array, each
; element of X specifies the X coordinate of the corresponding
; point in Z ( X(i,j) specifies the X coordinate for Z(i,j)).
;
; Y = as for X above, but for the Y coordinates.
;
; KEYWORD PARAMETERS: All keywords of contour, plus:
;
; 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.
;   FILL: If non-zero, contours are filled using POLYCNTR1
;   PATTERN = optional array of patterns, dimensioned (NX, NY, NPATTERN).
; OUTPUTS: None
; OPTIONAL OUTPUT PARAMETERS: None
; COMMON BLOCKS: None
; SIDE EFFECTS: Changes display.
; RESTRICTIONS:
; PROCEDURE:
; MODIFICATION HISTORY: Written PA Sterne (sterne1@llnl.gov) February 1993
;-
PRO contour1, Z, X, Y, $
;
; "Essential" keywords
;
fill=fill,color_index=color_index,pattern=pattern,$
c_colors=c_colors,follow=follow,noerase=noerase,path_filename=path_filename,$
;
; Eliminateable keywords. Any of these may be commented out along with
; the corresponding keyword_set line below. Comment out at least 9
; keywords of your choice to make it fit in PV-Wave's parameter limit.
; No such limit exists in IDL.
;
background=background,charsize=charsize,clip=clip,color=color,$
c_annotation=c_annotation,c_charsize=c_charsize,c_labels=c_labels,$
c_linestyle=c_linestyle,c_thick=c_thick,data=data,device=device,fancy=fancy,$
font=font,levels=levels,max_value=max_value,nlevels=nlevels,$
noclip=noclip,nodata=nodata,normal=normal,overplot=overplot,$
position=position,psym=psym,spline=spline,subtitle=subtitle,$
symsize=symsize,t3d=t3d,thick=thick,ticklen=ticklen,title=title,$
xcharsize=xcharsize,xmargin=xmargin,xminor=xminor,xrange=xrange,$

```

```

xstyle=xstyle,xtickname=xtickname,xticks=xticks,xtickv=xtick v,xtitle=xtitle,$
ycharsize=ycharsize,ymargin=ymargin,yminor=yminor,yrange=yrange,$
ystyle=ystyle,ytickname=ytickname,yticks=yticks,ytickv=ytick v,ytitle=ytitle,$
;zcharsize=zcharsize,zmargin=zmargin,zminor=zminor,zrange=zrange,$
;zstyle=zstyle,ztickname=ztickname,zticks=zticks,ztickv=ztick v,ztitle=zttitle,$
zaxis=zaxis,zvalue=zvalue

```

```

strkw = "
;
; "Essential" keyword_set lines
;
;
if keyword_set(noerase) then strkw = strkw+',noerase=noerase'
if keyword_set(follow) then strkw = strkw+',follow=follow'
;
; Elimlatable keyword_set lines. Comment out in correspondence with
; keywords above.
;
if keyword_set(background) then strkw = strkw+',background=background'
if keyword_set(charsize) then strkw = strkw+',charsize=charsize'
if keyword_set(clip) then strkw = strkw+',clip=clip'
if keyword_set(color) then strkw = strkw+',color=color'
if keyword_set(c_annotation) then strkw = strkw+',c_annotation=c_annotation'
if keyword_set(c_charsize) then strkw = strkw+',c_charsize=c_charsize'
if keyword_set(c_labels) then strkw = strkw+',c_labels=c_labels'
if keyword_set(c_linestyle) then strkw = strkw+',c_linestyle=c_linestyle'
if keyword_set(c_thick) then strkw = strkw+',c_thick=c_thick'
if keyword_set(data) then strkw = strkw+',data=data'
if keyword_set(device) then strkw = strkw+',device=device'
if keyword_set(fancy) then strkw = strkw+',fancy=fancy'
if keyword_set(font) then strkw = strkw+',font=font'
if keyword_set(levels) then strkw = strkw+',levels=levels'
if keyword_set(max_value) then strkw = strkw+',max_value=max_value'
if keyword_set(nlevels) then strkw = strkw+',nlevels=nlevels'
if keyword_set(noclip) then strkw = strkw+',noclip=noclip'
if keyword_set(nodata) then strkw = strkw+',nodata=nodata'
if keyword_set(normal) then strkw = strkw+',normal=normal'
if keyword_set(overplot) then strkw = strkw+',overplot=overplot'
if keyword_set(position) then strkw = strkw+',position=position'
if keyword_set(psym) then strkw = strkw+',psym=psym'
if keyword_set(spline) then strkw = strkw+',spline=spline'
if keyword_set(subtitle) then strkw = strkw+',subtitle=subtitle'
if keyword_set(symsize) then strkw = strkw+',symsize=symsize'
if keyword_set(t3d) then strkw = strkw+',t3d=t3d'
if keyword_set(thick) then strkw = strkw+',thick=thick'
if keyword_set(ticklen) then strkw = strkw+',ticklen=ticklen'
if keyword_set(title) then strkw = strkw+',title=title'
if keyword_set(xcharsize) then strkw = strkw+',xcharsize=xcharsize'
if keyword_set(xmargin) then strkw = strkw+',xmargin=xmargin'

```

```

if keyword_set(xminor) then strkw = strkw+',xminor=xminor'
if keyword_set(xrange) then strkw = strkw+',xrange=xrange'
if keyword_set(xstyle) then strkw = strkw+',xstyle=xstyle'
if keyword_set(xtickname) then strkw = strkw+',xtickname=xtickname'
if keyword_set(xticks) then strkw = strkw+',xticks=xticks'
if keyword_set(xtickv) then strkw = strkw+',xtickv=xtickv'
if keyword_set(xtitle) then strkw = strkw+',xtitle=xtitle'
if keyword_set(ycharsize) then strkw = strkw+',ycharsize=ycharsize'
if keyword_set(ymargin) then strkw = strkw+',ymargin=ymargin'
if keyword_set(yminor) then strkw = strkw+',yminor=yminor'
if keyword_set(yrange) then strkw = strkw+',yrange=yrange'
if keyword_set(ystyle) then strkw = strkw+',ystyle=ystyle'
if keyword_set(ytickname) then strkw = strkw+',ytickname=ytickname'
if keyword_set(yticks) then strkw = strkw+',yticks=yticks'
if keyword_set(ytickv) then strkw = strkw+',ytickv=ytickv'
if keyword_set(ytitle) then strkw = strkw+',ytitle=ytitle'
;if keyword_set(zcharsize) then strkw = strkw+',zcharsize=zcharsize'
;if keyword_set(zmargin) then strkw = strkw+',zmargin=zmargin'
;if keyword_set(zminor) then strkw = strkw+',zminor=zminor'
;if keyword_set(zrange) then strkw = strkw+',zrange=zrange'
;if keyword_set(zstyle) then strkw = strkw+',zstyle=zstyle'
;if keyword_set(ztickname) then strkw = strkw+',ztickname=ztickname'
;if keyword_set(zticks) then strkw = strkw+',zticks=zticks'
;if keyword_set(ztickv) then strkw = strkw+',ztickv=ztickv'
;if keyword_set(ztitle) then strkw = strkw+',ztitle=ztitle'
if keyword_set(zaxis) then strkw = strkw+',zaxis=zaxis'
if keyword_set(zvalue) then strkw = strkw+',zvalue=zvalue'

```

```

xy = "
if n_params() eq 2 then xy = ',X'
if n_params() eq 3 then xy = ',X ,Y'

```

```

IF KEYWORD_SET(fill) THEN BEGIN ; fill contours

```

```

    filename = 'cntrwave.dat'

```

```

    if keyword_set(path_filename) then filename = path_filename
    str1 = 'contour, Z'+xy+', path_filename=filename'+strkw

```

```

    tr0 = execute(str1) ; call contour to generate path file

```

```

    strpl = 'polycntr1,filename,Z'+xy
    if keyword_set(color_index) then c_index = color_index $
        else c_index = replicate(-1,30)
    strpl=strpl+',color_index=c_index'
    if keyword_set(pattern) then strpl = strpl+',pattern=pattern'

```

```

    tr = execute(strpl) ; call polycntr1 to shade contours

```

```

if n_elements(c_colors) eq 0 then begin ;set contour line colors
  c_colors = c_index(1:n_elements(c_index)-1)
  tmp1 = where(c_colors gt max(c_colors)*0.6,ncnt1)
  tmp2 = where(c_colors le max(c_colors)*0.6,ncnt2)
  IF ncnt2 GT 0 THEN c_colors(tmp2) = 255b
  IF ncnt1 GT 0 THEN c_colors(tmp1) = 0b
endif
strkw = strkw+',c_colors=c_colors'

; set keywords for second contour call to overdraw and to ensure
; consistency with pathfile generating contour call.

if keyword_set(noerase) eq 0 then strkw = strkw+',/noerase'
if keyword_set(follow) eq 0 then strkw = strkw+',/follow'

endif

str2 = 'contour, Z'+xy+strkw+'

tr1 = execute(str2)
return
end

PRO polycntr1, filename, Z, X, Y, color_index = color_index, pattern = pat $
    , delete_file = delfile
;+
; NAME: POLYCNTR1
; PURPOSE:
; Fill the contours defined by a path file created by CONTOUR.
; CATEGORY:
; Graphics.
; CALLING SEQUENCE:
; POLYCNTR1, Filename, Z [, X, Y, COLOR_INDEX = color index]
; INPUTS:
; Filename = name of file containing contour paths. This
; file must have been made using CONTOUR, PATH=Filename, ...
;
; Z = 2 dimensional data array used to generate the contours.
;
; X = vector or 2 dimensional array specifying the X coordinates
; for the contour surface. If a vector, each element specifies the
; X coordinate for a column of Z (Example: X(0) specifies the X
; coordinate for Z(0,*)). If X is a two-dimensional array, each
; element of X specifies the X coordinate of the corresponding
; point in Z ( X(i,j) specifies the X coordinate for Z(i,j)).

```

```

;
; Y = as for X above, but for the Y coordinates.
;
;
; 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).
; DELETE_FILE If present and non-zero, delete FILENAME once contours
; are read.
; OUTPUTS:
; The contours are filled on the display using normalized
; coordinates using the POLYFILL procedure.
; COMMON BLOCKS:
; None.
; SIDE EFFECTS:
; Contour temp file filename.tmp created in current directory.
; Display is updated.
; PROCEDURE:
; Scans file. Eliminates contours of 1 or 2 points. 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 contour temp file
; are both noted. The 2 dimensional array Z is used to determine
; which of these 2 contours encloses the higher ground. Closed
; contours are written directly to the contour temp file. Areas
; are sorted and contours are reread from the contour temp file
; and drawn in order of decreasing area.
; EXAMPLE:
; Create a 10 by 10 array of random numbers, contour to get
; path file, polycontour it, then overdraw the contour lines:
; b = RANDOMU(seed, 10,10) ;Insert random numbers
; CONTOUR, b, /SPLINE, PATH = 'path.dat' ;Make path file
; POLYCINTR1, 'path.dat',b ;Fill contours
; CONTOUR, b, /SPLINE, /NOERASE ;Overplot lines & labels
;
; Note: result looks best if the plot region is filled with data.
; Setting xstyle=1, ystyle=1, on CONTOUR accomplishes this.
; Contours are only be filled for the region spanned by the data.
; MODIFICATION HISTORY:
; DMS, AH, January, 1989.
; PAS = P. A. Sterne, Sterne1@llnl.gov
; PAS, LLNL, October 1991. Shades in order of contour area.
; PAS, LLNL, November 1992. Shades open contours too.
; PAS, LLNL, March 1993. Added in X,Y, and bugfixes.
; PAS,LLNL, Sept 1993. Fill background when all contours are closed.

```

```

;                               Fix for IDL 3.1.1 pathfile format.
;-

header = {contour_header,$
type : 0B, $
high_low : 0B, $
level : 0, $
num : 0L, $
value : 0.0 }
;
; ***** IMPORTANT *****
;
; set the variable pvoridl to 0 if you are using PVWave or early
; versions of IDL ( before 3.1.1, I think). Set it to 1 if you are
; using later versions of IDL. IDL changed the way in which contours
; are written, necessitating this change.
;
;
pvoridl = 0          ; PVWave or early IDL versions
;pvoridl = 1         ; IDL version 3.1.1 and later
;

tol = 1.e-5
asize = 100          ;# of elements in contour arrays
n = 0
cval = 0             ;Contour index
carea = fltarr(asize) ;Contour area
cstart = lonarr(asize) ;Starting byte of record
dtog0 = [!X.s(0), !Y.s(0)]
dtog1 = [!X.s(1), !Y.s(1)]
pos = fltarr(4)

sz = size(Z)
IF sz(0) NE 2 THEN BEGIN
  print, 'POLYCNTR1: Data array is not 2 dimensional. Returning'
  RETURN
ENDIF

regularxy = 0        ;zero for x,y vectors or not passed

CASE n_params() OF

  2: BEGIN           ; X and Y not passed to POLYCNTR1
    x1 = findgen(sz(1)) ; make X a vector
    y1 = findgen(sz(2)) ; make Y a vector
  END

  3: BEGIN           ; X passed as argument, but not Y
    szx = size(x)

```

```

x1 = x
IF szx(0) EQ 1 THEN y1 = findgen(sz(2)) $ ; make Y a vector
ELSE IF szx(0) EQ 2 THEN BEGIN
    y1 = replicate(1.0, sz(1))#findgen(sz(2)) ; make Y an array
    regularxy = 1
ENDIF
END

```

```

4: BEGIN                ;X and Y both passed as arguments
    x1 = x
    y1 = y
    szx = size(x)
    szy = size(y)
    IF max([szx(0), szy(0)]) EQ 2 THEN BEGIN
        regularxy = 1
        IF szx(0) EQ 1 THEN x1 = x#replicate(1.0, sz(2)) ; make X an array
        IF szy(0) EQ 1 THEN y1 = replicate(1.0, sz(1))#y ; make Y an array
    ENDIF
END

```

ENDCASE

```

IF regularxy EQ 0 THEN BEGIN
    pos = [dtog0+[x1(0), y1(0)]*dtog1, dtog0+[x1(sz(1)-1), y1(sz(2)-1)]*dtog1]
    scalexy = [min(abs(x1-shift(x1, 1))),min(abs(y1-shift(y1,1)))]*0.01
ENDIF ELSE BEGIN
    isx = indgen(sz(1))
    isy = sz(1)*(indgen(sz(2)-2)+1)
    bpts = [isx,isy+sz(1)-1,rotate(isx,2)+sz(1)*(sz(2)-1),rotate(isy,2) ]
    nbpts = 2*(sz(1)+sz(2)-2)
    xy2 = [[x1(bpts)], [y1(bpts)]]
    z2 = z(bpts)
    xyn = fltarr(2, nbpts)
    FOR i = 0, nbpts-1 DO xyn(*, i) = xy2(i, *)*dtog1 + dtog0
    scale = 0.01*min(sqrt(sum((shift(xy2, 1, 0)-xy2)^2, 1)))
    corners = [ 0, sz(1)-1,sz(1)+sz(2)-2,2*sz(1)+sz(2)-3]
    xpoly = fltarr(4)
    zpoly = fltarr(4)

```

ENDELSE

```

xyvert = pos([[0, 1], [2, 1], [2, 3], [0, 3]])
vlist = shift(pos,-1)
openr,unit,filename,/get_lun,delete=KEYWORD_SET(delfile)
openw,unit2,filename+'.tmp',/get_lun,/delete

```

while (not eof(unit)) do begin ;First pass reading contours

```
IF n EQ asize THEN BEGIN ;Expand our arrays?
```

```
  cstart = [cstart, cstart]
```

```
  carea = [carea, carea]
```

```
  asize = 2 * asize
```

```
ENDIF
```

```
a = fstat(unit2) ;File position
```

```
readu, unit, header ;Read header
```

```
IF (header.type EQ 0 AND header.num GT 1) THEN BEGIN ; Open contours
```

```
  cval = cval > fix(header.level) ; contour level
```

```
  xyarr = fltarr(2, header.num) ; Contour polygon
```

```
  hn = header.num - 1
```

```
  header1 = header
```

```
  header2 = header
```

```
  header1.type = 1 ;Reset header type to closed contour.
```

```
  header2.type = 1 ;Reset header type to closed contour.
```

```
  IF (pvoridl EQ 0) THEN readu, unit, xyarr $
```

```
  ELSE readu, unit, xyarr(0, *), xyarr(1, *)
```

```
  IF regularxy EQ 0 THEN BEGIN
```

```
; determine which side the open contour enters (side1) and exits (side2).
```

```
  xlst = [[1, 0, 1, 0], [2*hn+1, 2*hn, 2*hn+1, 2*hn]]
```

```
  sid1 = where(abs(xyarr(xlst(*, 0))-vlist) LT tol, count1)
```

```
  sid2 = where(abs(xyarr(xlst(*, 1))-vlist) LT tol, count2)
```

```
  IF count1 LE 0 AND header.num GT 2 THEN print, $
```

```
    'error open contour side1, n= ',n,' vertex=', xyarr(*, 0)
```

```
  IF count2 LE 0 AND header.num GT 2 THEN print, $
```

```
    'error open contour side2, n= ',n,' vertex=', xyarr(*, hn)
```

```
  IF (count1 LE 0) OR (count2 LE 0) THEN GOTO, RDREC
```

```
; close open contour
```

```
  side1 = max(sid1) ; side contour enters
```

```
  side2 = max(sid2) ; side contour exits
```

```
  vadd = shift(xyvert, 0, -side2-1)
```

```
  tmp = reverse(vadd, 2)
```

```
; close contour counter-clockwise (xyarr1) and clockwise (xyarr2)
```

```
  ivt = (-side2+side1+4) MOD 4
```

```
  IF ivt EQ 0 THEN BEGIN ; same side
```

```
    dotdir = total((xyarr(*, hn)-xyarr(*, 0))*(vadd(*, 3)-vadd(*, 0)))
```

```
    IF dotdir LT 0 THEN ivt = 4 - ivt
```

```
  ENDIF
```

```
  xyarr1 = fltarr(2, hn+2+ivt)
```

```
  xyarr2 = fltarr(2, hn+6-ivt)
```

```
header1.num = hn + 2 + ivt
header2.num = hn + 6 - ivt
```

```
IF ivt NE 0 THEN xyarr1(*, hn+1:hn+ivt) = vadd(*, 0:ivt-1)
IF ivt NE 4 THEN xyarr2(*, hn+1:hn+4-ivt) = tmp(*, 0:3-ivt)
```

```
xyarr1(*, 0:hn) = xyarr
xyarr2(*, 0:hn) = xyarr
xyarr1(*, header1.num-1) = xyarr(*, 0)
xyarr2(*, header2.num-1) = xyarr(*, 0)
```

```
; Check which of xyarr1/2 is higher.
```

```
sdxy = side2 MOD 2 ; 0 for side2=0,2; 1 for side2=1,3
sdsign = -2*(side2/2) + 1 ; +1 for side2=0,1 and -1 side2=2,3
dta = (xyarr(*, hn)-dtog0)/dtog1
IF sdxy EQ 0 THEN BEGIN
    vmin = min(abs(x1-dta(0)), it1)
    it1 = it1 > 1 < (sz(1)-2)
    it2 = ([-1, 0, 1]+it1)
    xpoly = [x1(it2), dta(0)] - dta(0)
    zpoly = [z(it2, (side2/2)*(sz(2)-1)), header.value]
ENDIF ELSE BEGIN
    vmin = min(abs(y1-dta(1)), it1)
    it1 = it1 > 1 < (sz(2)-2)
    it2 = ([-1, 0, 1]+it1)
    xpoly = [y1(it2), dta(1)] - dta(1)
    zpoly = [reform(z(((sdsign+1)/2)*(sz(1)-1), it2)), header.value]
ENDELSE
xsplin = xpoly(sort(xpoly))
zsplin = zpoly(sort(xpoly)) - header.value
dtsplin = [-1., 1.]*scalexy(sdxy)
val = rotate(spline(xsplin, zsplin, dtsplin), 2-2*(side2/2))
ind = [0B, 0B]
FOR i = 0, 1 DO IF val(i) GT 0. THEN ind(i) = 1b
IF ind(0) EQ ind(1) THEN print, ' both sides min or max, n=', n
```

```
; Check which of xyarr1/2 is higher. Add & subtract 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.
```

```
ENDIF ELSE BEGIN ; 2 dimensional X and Y arrays
```

```
; determine the boundary points on either side of the open contour
; where it enters (ptba1) and exits (ptba2).
```

```
dta1 = (xyarr(*, 0)-dtog0)/dtog1
```

```

dta2 = (xyarr(*, hn)-dtog0)/dtog1
vmin1 = min(((xy2(*,0)-dta1(0))^2+(xy2(*,1)-dta1(1))^2), it1)
vmin2 = min(((xy2(*,0)-dta2(0))^2+(xy2(*,1)-dta2(1))^2), it2)
itp1 = (it1 + 1) MOD nbpts
itm1 = (it1 - 1 + nbpts) MOD nbpts
itp2 = (it2 + 1) MOD nbpts
itm2 = (it2 - 1 + nbpts) MOD nbpts
grdp1 = xy2(itp1, *) - xy2(it1, *)
grdm1 = xy2(itm1, *) - xy2(it1, *)
grdp2 = xy2(itp2, *) - xy2(it2, *)
grdm2 = xy2(itm2, *) - xy2(it2, *)
diff1 = dta1 - xy2(it1, *)
diff2 = dta2 - xy2(it2, *)
IF total(diff1*diff1) GT tol THEN BEGIN
    dir1 = [total(grdp1*diff1), total(grdm1*diff1)]
    a1 = min(dir1/[total(grdp1*grdp1), total(grdm1*grdm1)], indir1)
    ptba1 = [it1-indir1, it1-indir1+1] ; bndry pt [below,above]
ENDIF ELSE ptba1 = [it1, it1]
indir2 = 0
IF total(diff2*diff2) GT tol THEN BEGIN
    dir2 = [total(grdp2*diff2), total(grdm2*diff2)]
    a1 = max(dir2/[total(grdp2*grdp2), total(grdm2*grdm2)], indir2)
    ptba2 = [it2-indir2, it2-indir2+1] ; contour boundary pt.
ENDIF ELSE ptba2 = [it2, it2]

```

; close contour counter-clockwise (xyarr1) and clockwise (xyarr2)

```

xy3 = shift(xyn, 0,-ptba2(1))
ptba1m = (ptba1 + nbpts - ptba2(1)) MOD nbpts
xyarr1 = [[xyarr], [xy3(*, 0:ptba1m(0))], [xyarr(*, 0)]]
xyarr2 = [[xyarr], [reverse(xy3(*, ptba1m(1):nbpts-1), 2)], $
    [xyarr(*, 0)]]
header1.num = (size(xyarr1))(2)
header2.num = (size(xyarr2))(2)

```

; Check which of xyarr1/2 is higher.

```

pt2 = ([it2-1, it2, it2+1]+nbpts) MOD nbpts
FOR i = 0 , 3 DO IF (it2 EQ corners(i)) THEN BEGIN
    IF indir2 EQ 0 THEN pt2 = (pt2+1) MOD nbpts
    IF indir2 EQ 1 THEN pt2 = (pt2-1+nbpts) MOD nbpts
ENDIF
FOR i = 0, 2 DO xpoly(i) = sqrt(total((xy2(pt2(i), *)-dta2)^2))
xpoly(0) = -xpoly(0)
xpoly(1) = (2*indir2-1)*xpoly(1)
xpoly(3) = 0.0
zpoly(0:2) = z2(pt2)
zpoly(3) = header.value

```

```

xsplin = xpoly(sort(xpoly))
zsplin = zpoly(sort(xpoly))
dtsplin = [-1., 1.]*scale
val = rotate(spline(xsplin, zsplin, dtsplin), 2)
ind = [0B, 0B]
FOR i = 0, 1 DO IF val(i) GT header.value THEN ind(i) = 1b
IF ind(0) EQ ind(1) THEN print, $
    ' 2-dim: both sides min or max, n=', n

```

ENDELSE

```

header1.high_low = ind(0)
cstart(n) = a.cur_ptr ;Position
carea(n) = poly_area(xyarr1(0, *), xyarr1(1, *))
writeu, unit2, header1
writeu, unit2, xyarr1
n = n + 1
a = fstat(unit2) ;File position
header2.high_low = ind(1)
cstart(n) = a.cur_ptr ;Position
carea(n) = poly_area(xyarr2(0, *), xyarr2(1, *))
writeu, unit2, header2
writeu, unit2, xyarr2
n = n + 1
ENDIF ELSE IF (header.num LE 2) THEN BEGIN ;Skip short contours
    xyarr = fltarr(2, header.num)
    IF (pvoridl EQ 0) THEN readu, unit, xyarr $
    ELSE readu, unit, xyarr(0, *), xyarr(1, *)
ENDIF ELSE BEGIN ;Closed contour

    cval = cval > fix(header.level) ;Color to draw
    cstart(n) = a.cur_ptr ;Position
    xyarr = fltarr(2, header.num) ;Define point to skip
    IF (pvoridl EQ 0) THEN readu, unit, xyarr $
    ELSE readu, unit, xyarr(0, *), xyarr(1, *)
    carea(n) = poly_area(xyarr(0, *), xyarr(1, *))
    writeu, unit2, header ;Write header
    writeu, unit2, xyarr
    n = n + 1
ENDELSE
RDREC:
ENDWHILE ; End of contour file
free_lun, unit

```

```

;print, 'cval,n=', cval,n ; max contour index ( = #contours-1) and #
cstart = cstart(0:n-1) ; Truncate
carea = carea(0:n-1)

```

```

order = rotate(sort(carea), 2) ;Subscripts of order
c_index = bindgen(cval+2)*byte(float(!D.n_colors)/(cval+1))
if keyword_set(color_index) eq 0 then color_index = c_index
if max(color_index) lt 0 then color_index = c_index
;print,color_index

IF regularxy EQ 0 THEN  xyv1 = [[xyvert], [xyvert(*, 0)]] $
      ELSE  xyv1 = [[xyn], [xyn(*, 0)]]
col = color_index(0)
polyfill,/normal, color=col, xyv1  ;shade in background color

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
;print,'n,level,high_low,value',j,header.level,header.high_low,header.value
  xyarr = fltarr(2, header.num) ;Define polygon array
  readu,unit2,xyarr  ;Read polygon
  inds = fix(header.level) + fix(header.high_low)
  col = color_index(inds) ;Drawing color
;print,'color',col
  IF N_ELEMENTS(pat) NE 0 THEN BEGIN
    s = size(pat)
    if s(0) ne 3 then begin
      print,'POLYCONTR1 - 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
end
--

```

```

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

```

Subject: Re: Fill Open Contour Routine
Posted by [sterne](#) on Mon, 27 Jun 1994 00:40:18 GMT
[View Forum Message](#) <> [Reply to Message](#)

There was a bug in the IDL version of the filled contours routine I posted last week. If you are running IDL and got an error message along the lines of:

READU: Expression must be named variable in this context: <FLOAT Array(1, 4)>

then please try this version. Many thanks to Jean Newman for the bug report. If anybody else has any other problems with it, please let me know.

The routine below is a drop-in replacement for the standard PV-Wave and former IDL routine POLYCONTOUR. Remember to change the value of the variable pvoridl to the appropriate value depending on whether you are running PV-Wave or IDL. This routine can be used with the wrapper routine CONTOUR1 in the previous post to give an easy way of filling contours, regardless of whether they are open or closed.

Enjoy,
Phil

```
-----

PRO polycntr1, filename, Z, X, Y, color_index = color_index, pattern = pat $
    , delete_file = delfile
;+
; NAME: POLYCNTR1
; PURPOSE:
; Fill the contours defined by a path file created by CONTOUR.
; CATEGORY:
; Graphics.
; CALLING SEQUENCE:
; POLYCNTR1, Filename, Z [, X, Y, COLOR_INDEX = color index]
; INPUTS:
; Filename = name of file containing contour paths. This
; file must have been made using CONTOUR, PATH=Filename, ...
;
; Z = 2 dimensional data array used to generate the contours.
;
; X = vector or 2 dimensional array specifying the X coordinates
; for the contour surface. If a vector, each element specifies the
; X coordinate for a column of Z (Example: X(0) specifies the X
; coordinate for Z(0,*)). If X is a two-dimensional array, each
; element of X specifies the X coordinate of the corresponding
; point in Z ( X(i,j) specifies the X coordinate for Z(i,j)).
;
; Y = as for X above, but for the Y coordinates.
;
; 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).
;   DELETE_FILE If present and non-zero, delete FILENAME once contours
;       are read.
; OUTPUTS:
; The contours are filled on the display using normalized
; coordinates using the POLYFILL procedure.
; COMMON BLOCKS:
; None.
; SIDE EFFECTS:
; Contour temp file filename.tmp created in current directory.
; Display is updated.
; PROCEDURE:
; Scans file. Eliminates contours of 1 or 2 points. 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 contour temp file
;   are both noted. The 2 dimensional array Z is used to determine
; which of these 2 contours encloses the higher ground. Closed
; contours are written directly to the contour temp file. Areas
; are sorted and contours are reread from the contour temp file
; and drawn in order of decreasing area.
; EXAMPLE:
; Create a 10 by 10 array of random numbers, contour to get
; path file, polycontour it, then overdraw the contour lines:
; b = RANDOMU(seed, 10,10) ;Insert random numbers
; CONTOUR, b, /SPLINE, PATH = 'path.dat' ;Make path file
; POLYCINTR1, 'path.dat',b ;Fill contours
; CONTOUR, b, /SPLINE, /NOERASE ;Overplot lines & labels
;
; Note: result looks best if the plot region is filled with data.
; Setting xstyle=1, ystyle=1, on CONTOUR accomplishes this.
; Contours are only be filled for the region spanned by the data.
; MODIFICATION HISTORY:
; DMS, AH, January, 1989.
;   PAS = P. A. Sterne, Sterne1@llnl.gov
; PAS, LLNL, October 1991. Shades in order of contour area.
; PAS, LLNL, November 1992. Shades open contours too.
;   PAS, LLNL, March 1993. Added in X,Y, and bugfixes.
;   PAS, LLNL, Sept 1993. Fill background when all contours are closed.
;       Fix for IDL 3.0.0 pathfile format.
;   PAS, LLNL, June 1994. Bugfix to IDL-specific component.
;-

```

```

header = {contour_header,$
type : 0B, $

```

```

high_low : 0B, $
level : 0, $
num : 0L, $
value : 0.0 }
;
; ***** IMPORTANT *****
;
;
; set the variable pvoridl to 0 if you are using PVWave or early
; versions of IDL ( before 3.0.0, I think). Set it to 1 if you are
; using later versions of IDL. IDL changed the way in which contours
; are written, necessitating this change.
;
;
pvoridl = 0          ; PVWave or early IDL versions
;pvoridl = 1         ; IDL version 3.0.0 and later
;

tol = 1.e-5
asize = 100          ;# of elements in contour arrays
n = 0
cval = 0             ;Contour index
carea = fltarr(asize) ;Contour area
cstart = lonarr(asize) ;Starting byte of record
dtog0 = [!X.s(0), !Y.s(0)]
dtog1 = [!X.s(1), !Y.s(1)]
pos = fltarr(4)

sz = size(Z)
IF sz(0) NE 2 THEN BEGIN
  print, 'POLYCNTR1: Data array is not 2 dimensional. Returning'
  RETURN
ENDIF

regularxy = 0        ;zero for x,y vectors or not passed

CASE n_params() OF

  2: BEGIN           ; X and Y not passed to POLYCNTR1
    x1 = findgen(sz(1)) ; make X a vector
    y1 = findgen(sz(2)) ; make Y a vector
  END

  3: BEGIN           ; X passed as argument, but not Y
    szx = size(x)
    x1 = x
    IF szx(0) EQ 1 THEN y1 = findgen(sz(2)) $ ; make Y a vector
    ELSE IF szx(0) EQ 2 THEN BEGIN
      y1 = replicate(1.0, sz(1))#findgen(sz(2)) ; make Y an array
      regularxy = 1
    END
  END

```

```
ENDIF
END
```

```
4: BEGIN ;X and Y both passed as arguments
  x1 = x
  y1 = y
  szx = size(x)
  szy = size(y)
  IF max([szx(0), szy(0)]) EQ 2 THEN BEGIN
    regularxy = 1
    IF szx(0) EQ 1 THEN x1 = x#replicate(1.0, sz(2)) ; make X an array
    IF szy(0) EQ 1 THEN y1 = replicate(1.0, sz(1))#y ; make Y an array
  ENDIF
END
```

```
ENDCASE
```

```
IF regularxy EQ 0 THEN BEGIN
  pos = [dtog0+[x1(0), y1(0)]*dtog1, dtog0+[x1(sz(1)-1), y1(sz(2)-1)]*dtog1]
  scalexy = [min(abs(x1-shift(x1, 1))),min(abs(y1-shift(y1,1)))]*0.01
ENDIF ELSE BEGIN
  isx = indgen(sz(1))
  isy = sz(1)*(indgen(sz(2)-2)+1)
  bpts = [isx,isy+sz(1)-1,rotate(isx,2)+sz(1)*(sz(2)-1),rotate(isy,2) ]
  nbpts = 2*(sz(1)+sz(2)-2)
  xy2 = [[x1(bpts)], [y1(bpts)]]
  z2 = z(bpts)
  xyn = fltarr(2, nbpts)
  FOR i = 0, nbpts-1 DO xyn(*, i) = xy2(i, *)*dtog1 + dtog0
  scale = 0.01*min(sqrt(sum((shift(xy2, 1, 0)-xy2)^2, 1)))
  corners = [ 0, sz(1)-1,sz(1)+sz(2)-2,2*sz(1)+sz(2)-3]
  xpoly = fltarr(4)
  zpoly = fltarr(4)
```

```
ENDELSE
```

```
xyvert = pos([[0, 1], [2, 1], [2, 3], [0, 3]])
vlist = shift(pos,-1)
openr,unit,filename,/get_lun,delete=KEYWORD_SET(delfile)
openw,unit2,filename+'.tmp',/get_lun,/delete
```

```
while (not eof(unit)) do begin ;First pass reading contours
```

```
  IF n EQ asize THEN BEGIN ;Expand our arrays?
    cstart = [cstart, cstart]
    carea = [carea, carea]
    asize = 2 * asize
  ENDIF
```

```

a = fstat(unit2)          ;File position
readu, unit, header       ;Read header
IF (header.type EQ 0 AND header.num GT 1) THEN BEGIN ; Open contours
  cval = cval > fix(header.level)          ; contour level
  xyarr = fltarr(2, header.num)            ; Contour polygon
  hn = header.num - 1
  header1 = header
  header2 = header
  header1.type = 1 ;Reset header type to closed contour.
  header2.type = 1 ;Reset header type to closed contour.
  IF (pvoridl EQ 0) THEN readu, unit, xyarr $
  ELSE BEGIN
xyarrp = transpose(xyarr)
readu, unit, xyarrp
xyarr= transpose(xyarrp)
  ENDELSE

```

```

IF regularxy EQ 0 THEN BEGIN

```

```

; determine which side the open contour enters (side1) and exits (side2).

```

```

xlst = [[1, 0, 1, 0], [2*hn+1, 2*hn, 2*hn+1, 2*hn]]
sid1 = where(abs(xyarr(xlst(*, 0))-vlist) LT tol, count1)
sid2 = where(abs(xyarr(xlst(*, 1))-vlist) LT tol, count2)
IF count1 LE 0 AND header.num GT 2 THEN print, $
  'error open contour side1, n= ',n,' vertex=', xyarr(*, 0)
IF count2 LE 0 AND header.num GT 2 THEN print, $
  'error open contour side2, n= ',n,' vertex=', xyarr(*, hn)
IF (count1 LE 0) OR (count2 LE 0) THEN GOTO, RDREC

```

```

; close open contour

```

```

side1 = max(sid1)    ; side contour enters
side2 = max(sid2)    ; side contour exits
vadd = shift(xyvert, 0, -side2-1)
tmp = reverse(vadd, 2)

```

```

; close contour counter-clockwise (xyarr1) and clockwise (xyarr2)

```

```

ivt = (-side2+side1+4) MOD 4
IF ivt EQ 0 THEN BEGIN ; same side
  dotdir = total((xyarr(*, hn)-xyarr(*, 0))*(vadd(*, 3)-vadd(*, 0)))
  IF dotdir LT 0 THEN ivt = 4 - ivt
ENDIF
xyarr1 = fltarr(2, hn+2+ivt)
xyarr2 = fltarr(2, hn+6-ivt)

```

```
header1.num = hn + 2 + ivt
header2.num = hn + 6 - ivt
```

```
IF ivt NE 0 THEN xyarr1(*, hn+1:hn+ivt) = vadd(*, 0:ivt-1)
IF ivt NE 4 THEN xyarr2(*, hn+1:hn+4-ivt) = tmp(*, 0:3-ivt)
```

```
xyarr1(*, 0:hn) = xyarr
xyarr2(*, 0:hn) = xyarr
xyarr1(*, header1.num-1) = xyarr(*, 0)
xyarr2(*, header2.num-1) = xyarr(*, 0)
```

```
; Check which of xyarr1/2 is higher.
```

```
sdxy = side2 MOD 2 ; 0 for side2=0,2; 1 for side2=1,3
sdsign = -2*(side2/2) + 1 ; +1 for side2=0,1 and -1 side2=2,3
dta = (xyarr(*, hn)-dtog0)/dtog1
IF sdxy EQ 0 THEN BEGIN
    vmin = min(abs(x1-dta(0)), it1)
    it1 = it1 > 1 < (sz(1)-2)
    it2 = ([-1, 0, 1]+it1)
    xpoly = [x1(it2), dta(0)] - dta(0)
    zpoly = [z(it2, (side2/2)*(sz(2)-1)), header.value]
ENDIF ELSE BEGIN
    vmin = min(abs(y1-dta(1)), it1)
    it1 = it1 > 1 < (sz(2)-2)
    it2 = ([-1, 0, 1]+it1)
    xpoly = [y1(it2), dta(1)] - dta(1)
    zpoly = [reform(z(((sdsign+1)/2)*(sz(1)-1), it2)), header.value]
ENDELSE
xsplin = xpoly(sort(xpoly))
zsplin = zpoly(sort(xpoly)) - header.value
dtsplin = [-1., 1.]*scalexy(sdxy)
val = rotate(spline(xsplin, zsplin, dtsplin), 2-2*(side2/2))
ind = [0B, 0B]
FOR i = 0, 1 DO IF val(i) GT 0. THEN ind(i) = 1b
IF ind(0) EQ ind(1) THEN print, ' both sides min or max, n=', n
```

```
; Check which of xyarr1/2 is higher. Add & subtract 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.
```

```
ENDIF ELSE BEGIN ; 2 dimensional X and Y arrays
```

```
; determine the boundary points on either side of the open contour
; where it enters (ptba1) and exits (ptba2).
```

```
dta1 = (xyarr(*, 0)-dtog0)/dtog1
```

```

dta2 = (xyarr(*, hn)-dtog0)/dtog1
vmin1 = min(((xy2(*,0)-dta1(0))^2+(xy2(*,1)-dta1(1))^2), it1)
vmin2 = min(((xy2(*,0)-dta2(0))^2+(xy2(*,1)-dta2(1))^2), it2)
itp1 = (it1 + 1) MOD nbpts
itm1 = (it1 - 1 + nbpts) MOD nbpts
itp2 = (it2 + 1) MOD nbpts
itm2 = (it2 - 1 + nbpts) MOD nbpts
grdp1 = xy2(itp1, *) - xy2(it1, *)
grdm1 = xy2(itm1, *) - xy2(it1, *)
grdp2 = xy2(itp2, *) - xy2(it2, *)
grdm2 = xy2(itm2, *) - xy2(it2, *)
diff1 = dta1 - xy2(it1, *)
diff2 = dta2 - xy2(it2, *)
IF total(diff1*diff1) GT tol THEN BEGIN
    dir1 = [total(grdp1*diff1), total(grdm1*diff1)]
    a1 = min(dir1/[total(grdp1*grdp1), total(grdm1*grdm1)], indir1)
    ptba1 = [it1-indir1, it1-indir1+1] ; bndry pt [below,above]
ENDIF ELSE ptba1 = [it1, it1]
indir2 = 0
IF total(diff2*diff2) GT tol THEN BEGIN
    dir2 = [total(grdp2*diff2), total(grdm2*diff2)]
    a1 = max(dir2/[total(grdp2*grdp2), total(grdm2*grdm2)], indir2)
    ptba2 = [it2-indir2, it2-indir2+1] ; contour boundary pt.
ENDIF ELSE ptba2 = [it2, it2]

```

; close contour counter-clockwise (xyarr1) and clockwise (xyarr2)

```

xy3 = shift(xyn, 0,-ptba2(1))
ptba1m = (ptba1 + nbpts - ptba2(1)) MOD nbpts
xyarr1 = [[xyarr], [xy3(*, 0:ptba1m(0))], [xyarr(*, 0)]]
xyarr2 = [[xyarr], [reverse(xy3(*, ptba1m(1):nbpts-1), 2)], $
    [xyarr(*, 0)]]
header1.num = (size(xyarr1))(2)
header2.num = (size(xyarr2))(2)

```

; Check which of xyarr1/2 is higher.

```

pt2 = ([it2-1, it2, it2+1]+nbpts) MOD nbpts
FOR i = 0, 3 DO IF (it2 EQ corners(i)) THEN BEGIN
    IF indir2 EQ 0 THEN pt2 = (pt2+1) MOD nbpts
    IF indir2 EQ 1 THEN pt2 = (pt2-1+nbpts) MOD nbpts
ENDIF
FOR i = 0, 2 DO xpoly(i) = sqrt(total((xy2(pt2(i), *)-dta2)^2))
xpoly(0) = -xpoly(0)
xpoly(1) = (2*indir2-1)*xpoly(1)
xpoly(3) = 0.0
zpoly(0:2) = z2(pt2)
zpoly(3) = header.value

```

```

xsplin = xpoly(sort(xpoly))
zsplin = zpoly(sort(xpoly))
dtsplin = [-1., 1.]*scale
val = rotate(spline(xsplin, zsplin, dtsplin), 2)
ind = [0B, 0B]
FOR i = 0, 1 DO IF val(i) GT header.value THEN ind(i) = 1b
IF ind(0) EQ ind(1) THEN print, $
    ' 2-dim: both sides min or max, n=', n

```

ENDELSE

```

header1.high_low = ind(0)
cstart(n) = a.cur_ptr ;Position
carea(n) = poly_area(xyarr1(0, *), xyarr1(1, *))
writeu, unit2, header1
writeu, unit2, xyarr1
n = n + 1
a = fstat(unit2) ;File position
header2.high_low = ind(1)
cstart(n) = a.cur_ptr ;Position
carea(n) = poly_area(xyarr2(0, *), xyarr2(1, *))
writeu, unit2, header2
writeu, unit2, xyarr2
n = n + 1
ENDIF ELSE IF (header.num LE 2) THEN BEGIN ;Skip short contours
    xyarr = fltarr(2, header.num)
    IF (pvoridl EQ 0) THEN readu, unit, xyarr $
        ELSE BEGIN
            xyarrp = transpose(xyarr)
            readu, unit, xyarrp
            xyarr= transpose(xyarrp)
        ENDELSE

```

ENDIF ELSE BEGIN ;Closed contour

```

    cval = cval > fix(header.level) ;Color to draw
    cstart(n) = a.cur_ptr ;Position
    xyarr = fltarr(2, header.num) ;Define point to skip
    IF (pvoridl EQ 0) THEN readu, unit, xyarr $
        ELSE BEGIN
            xyarrp = transpose(xyarr)
            readu, unit, xyarrp
            xyarr= transpose(xyarrp)
        ENDELSE

```

```

    carea(n) = poly_area(xyarr(0, *), xyarr(1, *))
    writeu, unit2, header ;Write header
    writeu, unit2, xyarr

```

```

    n = n + 1
  ENDELSE
RDREC:
ENDWHILE          ; End of contour file
free_lun, unit

;print, 'cval,n=', cval,n      ; max contour index ( = #contours-1) and #
cstart = cstart(0:n-1)      ; Truncate
carea = carea(0:n-1)
order = rotate(sort(carea), 2) ;Subscripts of order
c_index = bindgen(cval+2)*byte(float(!D.n_colors)/(cval+1))
if keyword_set(color_index) eq 0 then color_index = c_index
if max(color_index) lt 0 then color_index = c_index
;print,color_index

IF regularxy EQ 0 THEN  xyv1 = [[xyvert], [xyvert(*, 0)]] $
      ELSE  xyv1 = [[xyn], [xyn(*, 0)]]
col = color_index(0)
polyfill,/normal, color=col, xyv1  ;shade in background color

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
;print,'n,level,high_low,value',j,header.level,header.high_low,header.value
  xyarr = fltarr(2, header.num) ;Define polygon array
  readu,unit2,xyarr  ;Read polygon
  inds = fix(header.level) + fix(header.high_low)
  col = color_index(inds) ;Drawing color
;print,'color',col
  IF N_ELEMENTS(pat) NE 0 THEN BEGIN
    s = size(pat)
    if s(0) ne 3 then begin
      print,'POLYCONTR1 - 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
end
--

```

```

-----
Philip Sterne |  Sterne@dublin.lnl.gov
Lawrence Livermore National Laboratory |  Phone (510) 422-2510

```

