Subject: CONTTW.PRO and EXPAND.PRO - some bug fixes, enhancements
Posted by ryba on Thu, 08 Oct 1992 17:37:17 GMT
View Forum Message <> Reply to Message

Here are fixed up versions of CONTTW.PRO and EXPAND.PRO - I'm still working on revisions, so if anyone has any suggestions, please email me. The major changes I made include converting EXPAND from a procedure to a function (seemed more natural to me), and some bug fixes with autoscaling and window selection, color table generalizations, etc. For example, if you already have a connection with the server that causes a shared colormap (!d.n_colors < 256), this should work properly. For displays I default to the vector fonts for improved portability and legibility, etc. I also have it now auto-detect what device (using !d.name), and converted a bunch of misuses of keyword_set() to n_elements(). I also added missing code for SUBTITLE and XTICKV, and re-implemented /HELP.

```
 ;+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++++++++
; NAME:
; CONTTW
; PURPOSE:
; Produce a color filled contour
; CATEGORY:
; Z4 - IMAGE PROCESSING, data display
; CALLING SEQUENCE:
; CONTTW,Z,X,Y {,keys}
;  Calling sequence and keys are nearly identical to CONTOUR.
; INPUTS:
; Z Array to be contoured
; X Vector of X coordinates
; Y Vector of Y coordinates
; KEYWORD PARAMETERS:
;  In this section (XYB) stands for X-axis, Y-axis, and Color Bar.
;  The Keywords: (XYB)RANGE, (XYB)STYLE, (XYB)TICKS, (XYB)TICKV, (XYB)MINOR,
;  (XYB)TITLE, (XYB)TYPE, (XYB)TICKNAME, TITLE, SUBTITLE, and CHARSIZE
;  are similar to their implementations in CONTOUR.  (defaults to associated
;  !x, !y, or !p values)
;
; /RIGHT, /LEFT, /TOP, /BOTTOM, and /NOBAR determines the location of
;      the optional color bar.
; TICKLEN, BLEN The default for this parameter is to draw the ticks
;      inward.  If outward tick marks are needed then this
;      parameter should be set to a negative value.  BLEN
;      controls the color bar; defaults to ticklen*10.
; CHARSIZE Size of axis labels (default = 1).
; THICK  Sets the thickness of all lines and characters
;      (labels of contour lines cannot be thickened).
```

```
; LEV  Vector of levels to contour.
; WINDOW=n Selects the window number to be used.
; /NEW  If this keyword is present, highest unused window is
;      used.
; WSIZE=[x,y] Sets the size of the window (default = [640,512]).
; /SHOW  If this keyword is present then the image will appear
;      as a viewable window.
; /OVERLAY Controls whether line contours should be overplotted.
; /INVERSE White on black background.
; /DATE  Dates the image along the bottom.
; TABLE  Uses requested color table (default = 5).
; RES  Determines the coarseness of the plot (default = 5).
; /RESET  Resets !X, !Y, !P, !QUIET at end of program.
;
; Color mapping is controlled by BRANGE, if present. Otherwise it is
; controlled by the first and last elements of LEV.  If both are missing,
; then the program makes an attempt at auto-scaling (no promises here).
;
; OUTPUTS: Either creates a color graphic or gray scale on screen or
;          in a postscript file or creates a TEK-4693D compatable file
;          if the appropriate software is available.
; COMMON BLOCKS: CONTTW_1
; SIDE EFFECTS: Loads a color table.
; RESTRICTIONS: Assumes A is on an evenly spaced grid in both x and y.  If
;   A is not on a regular grid, the user should use
;   POLY_2D or splines to place it on one.
;  Change the default color scale (defcol) value when installed
; PROCEDURE: Nothing too fancy to require documentation.
; MODIFICATION HISTORY:
; Aug 14, 1989 J. M. Zawodny, NASA/LaRC, MS 475, Hampton VA, 23665.
;       Jan 04, 1990 R. E. Boughner, modified for use with subarrays.
; Jul 27, 1990 J. M. Zawodny, Major upgrade to IDL version 2.
;  Nov 27-Dec 13, 1991  A. C. Edwards, changed color bars, changed
;                       pixels, added xybtickv, xybtickname. Changed
;                       color bar procedure so tickmarks show. Added ability
;                       to make postscipt files. Added ability to do
;                       gray scale plots after checking !d.n_colors
;                       AND reset plotting variables only on command.
;  Jan-Feb, 1992        organizing the mess
;  Feb 20, 1992  Deleted /LOGO option so program would
;   compile in default program area (.SIZE not needed).
;  Sep 14, 1192  M. F. Ryba, slice and dice for Lincoln use
 ;------------------------------------------------------- ----------------
pro CONTTW,a,xold,yold $
, show=show, new=new, window=window, levels=lev, wsize=wsize $
, overlay=overlay, help=help, reset=reset, right=right, left=left, top=top $
, bottom=bottom, nobar=nobar, brange=brange, btitle=btitle, bticks=bticks $
, bminor=bminor, bstyle=bstyle, btype=btype, blen=blen, btickv=btickv $
```

```
, btickname=btickname, xrange=xrange, xtitle=xtitle, xticks=xticks, res=res $
, xminor=xminor, xstyle=xstyle, xtype=xtype, xtickv=xtickv, table=table $
, xtickname=xtickname, yrange=yrange, ytitle=ytitle, yticks=yticks $
, yminor=yminor, ystyle=ystyle, ytype=ytype, ytickv=ytickv, date=date $
, ytickname=ytickname, thick=thick, title=title, subtitle=subtitle $
, ticklen=ticklen $
, charsize=charsize, inverse=inverse, maxval=maxval, fillval=fillval

; Default color scale, display size (for non-Postscript)
defcol = 5
wdx = 640 & wdy = 512
; ===> HELP <===
if keyword_set(help) then begin
  doc_library,'conttw' & return
endif

; Save some Plotting Things here
save_p = !p & save_x = !x & save_y = !y & save_noerase = !p.noerase

; Test for sufficient information
if n_params() lt 3 then begin
 print,'CONTTW called with too few parameters'
 return
endif

; Find the min and max of array a
mia = min(a) & maa = max(a)
; Do we try to auto-scale?  (Don't try too hard!)
n_lev = n_elements(lev)
if ((n_elements(brange) eq 0) and (n_lev eq 0)) then begin
 if !quiet eq 0 then print,'  Auto-scale requested '
 dma    = (maa-mia)/10.
 brange = [mia,maa]
 lev = findgen(11)*dma+mia
 n_lev  = n_elements(lev)
endif

; If axis parameters have not been set then make defaults
; Default {x,y}style is exact axis range
if n_elements(xstyle) eq 0 then xstyle = !x.style or 1
if n_elements(ystyle) eq 0 then ystyle = !y.style or 1
if n_elements(bstyle) eq 0 then bstyle = 1
if n_elements(xtype) eq 0 then xtype = !x.type
if n_elements(ytype) eq 0 then ytype = !y.type
if n_elements(btype) eq 0 then btype = 0
if n_elements(xticks) eq 0 then xticks = !x.ticks
if n_elements(yticks) eq 0 then yticks = !y.ticks
if n_elements(bticks) eq 0 then bticks = n_lev-1
```

```
if n_elements(xtickv) eq 0 then xtickv = !x.tickv
if n_elements(ytickv) eq 0 then ytickv = !y.tickv
if n_elements(btickv) eq 0 then btickv = [0,0]
if n_elements(xtickname) eq 0 then xtickname = ''
if n_elements(ytickname) eq 0 then ytickname = ''
if n_elements(btickname) eq 0 then btickname = ''
if n_elements(xminor) eq 0 then xminor = !x.minor
if n_elements(yminor) eq 0 then yminor = !y.minor
if n_elements(bminor) eq 0 then bminor = 0
if n_elements(xtitle) eq 0 then xtitle = !x.title
if n_elements(ytitle) eq 0 then ytitle = !y.title
if n_elements(btitle) eq 0 then btitle = ''
if n_elements(brange) eq 0 then brange = [lev(0),lev(n_lev-1)]
if n_elements(xrange) eq 0 then xrange = !x.range
if n_elements(yrange) eq 0 then yrange = !y.range

; If plot parameters have not been set then make defaults
if n_elements(maxval) eq 0 then maxval = 1.e33
if n_elements(fillval) eq 0 then fillval = 1.e34
if n_elements(title) eq 0 then title = !p.title
if n_elements(subtitle) eq 0 then subtitle = !p.subtitle
if n_elements(ticklen) eq 0 then ticklen = !p.ticklen
if n_elements(blen) eq 0 then blen = ticklen*10
if n_elements(charsize) eq 0 then charsize  = !p.charsize
if(n_elements(show) eq 0) then show  = 1
if keyword_set(overlay)  then overlay = 0  else overlay = 1
!p.title=''

; Set axis and plot parameters to thick
if n_elements(thick) gt 0 then begin
 !x.thick = thick & !y.thick = thick & !p.thick = thick
 !p.charthick = thick
endif else thick = 0

;  Determine whether to use gray scale or full color values
if (!d.n_colors eq 16) then begin
; **** These should be changed to suit your needs ****
 white       = 15  & black   =  0
 topcolor    = 13  & offset  = 1B
 totalcolors = 16  & ncolors = 14
endif else begin
nc = !d.n_colors
; **** These should be changed to suit your needs ****
 white       = nc-1 & black   = nc-2
 topcolor    = nc-3 & offset  =  0B
 totalcolors = nc   & ncolors = nc-2
endelse
```

```
; Skip this if doing a postscipt plot
if !d.name ne 'PS' then begin
   ; Set defaults
 if keyword_set(res)  then pixf      = res else pixf      = 5
 if keyword_set(date) then time_flag = 1   else time_flag = 0
 if keyword_set(logo) then logo_flag = 1   else logo_flag = 0
endif

if keyword_set(inverse) then begin
   ; Reverse screen colors
 text = white & bkgnd = black
endif else begin
 text = black & bkgnd = white
endelse

; Find out where color bar goes
barlocs = ['/NOBAR','/RIGHT','/LEFT','TOP','/BOTTOM']
barloc  = [keyword_set(nobar),keyword_set(right),keyword_set(left), $
        keyword_set(top),keyword_set(bottom)]
barloc  = where(barloc,len)
if (len gt 1) then begin
 print,'CONTTW called with incompatable keywords'
 print,barlocs(barloc)
 return
endif

; Default to /NOBAR
if (len eq 0) then barloc=0 else barloc=barloc(0)

; Well we are now safe!
quiet = !QUIET & !QUIET = 1

; SKIP THIS WINDOW STUFF IF POSTSCRIPT IS CALLED
if !d.name ne 'PS' then begin

   ; Set window, plot, and bar sizes
 if n_elements(wsize) eq 2 then begin
  wdx = wsize(0) & wdy = wsize(1)
 endif

   ; If a new window is requested or no window has ever been used.
   ; WINDOW keyword overrides /NEW.
 if(keyword_set(new) or !d.window eq -1) then wnum = !d.window + 1 $
 else $
 ; Using an old window or a specific window
  if n_elements(window) eq 1 then wnum=window else wnum=!d.window
 if keyword_set(show) then begin
  ; If we need to show the window
```

```
  window,wnum,xsize=wdx,ysize=wdy, $
    xpos=2,ypos=2,retain=2,color=totalcolors
  ; otherwise make it a pixmap
 endif else window,wnum,xsize=wdx,ysize=wdy,/pixmap,$
  color=totalcolors

; **** These should be changed to suit your needs ****
 ;"Zero" workspace to background color
 tv,replicate(bkgnd,wdx,wdy)

 ; We need to Overlay Multiple Objects
 !P.NOERASE = 1
endif

; **** These should be changed to suit your needs ****
; Select a color table
if n_elements(table) gt 0 then coltab=table else coltab = defcol
if totalcolors eq 16 then loadct,0 else begin
  loadct,coltab
; Make sure we have black and white at top
  r=[0b,255b]
  g=[0b,255b]
  b=[0b,255b]
  tvlct,r,g,b,ncolors
endelse

; Reset !COLOR for line drawing
if !d.name eq 'PS' then !p.color = black else !p.color = text

; Make extra room for large values along bar
if((abs(brange(0)) gt 100) or (abs(brange(1)) gt 100))then xb=1 else xb=0

; Make extra room for negative signs
if((brange(0) lt 0) or (brange(1) lt 0))then xn=1 else xn=0

; Take back extra room if there is not a bar title
if(btitle eq '') then xt=2 else xt=0

; Set position of plot and bar regions and plot color bar
bar  = byte(indgen(ncolors,2) mod ncolors)+offset
case barloc of
1: begin ; Bar on Right
  preg      = [.0,.0,.85,1.]
  pmarx     = [8,1]
  pmary     = [4,2]
  !p.region = [.85,.0,1.,1.]
  !x.margin = [0,9.5+xb+xn-xt]
  !y.margin = [8,5]
```

```
      ; Locate bar area
   plot,[0,1],[0,1],/nodata,xstyle=4,ystyle=4, $
    ytit=btitle,chars=charsize-.5,back=white
      ; Make a Color Bar
   bar = transpose(bar)
   if !d.name eq 'PS' then begin
   ; Postscript color bar
    bardevx=[!p.clip(0),!p.clip(2)]
    bardevy=[!p.clip(1),!p.clip(3)]
    barnorm=convert_coord(bardevx,bardevy,$
     /to_normal,/device)
    barnx=barnorm(0,1)-barnorm(0,0)
    barny=barnorm(1,1)-barnorm(1,0)
    tv,bar,barnorm(0,0),barnorm(1,0),$
     /normal,xsize=barnx,ysize=barny
   endif else begin
   ; Screen color bar
    barw = 28 ; .25:
    barl = !p.clip(3)-!p.clip(1)+1
    rbar = expand(bar,barw,barl)
    tv,rbar,(!p.clip(0)+28),!p.clip(1)
   endelse
   rbar = 0   ; save memory
  ; Draw color bar axis
   axis,yaxis=1,ystyle=bstyle,yran=brange,ytitle=btitle, $
    ytype=btype,yticks=bticks,yminor=bminor,ticklen=blen, $
    charsize=charsize-.5,thick=thick,/nocli, $
    ytickv=btickv,yticknam=btickname
  end
 2: begin ; Bar on Left
   preg      = [.15,.0,1.,1.]
   pmarx     = [6,1.5]
   pmary     = [4,2]
   !p.region = [.0,.0,.15,1.0]
   !x.margin = [9+xb+xn-xt,0]
   !y.margin = [8,5]
      ; Locate bar area
   plot,[0,1],[0,1],/nodata,xstyle=4,ystyle=4,ytit=btitle, $
    chars=charsize-.5,back=white
  ; Make a Color Bar
   bar = transpose(bar)
   if !d.name eq 'PS' then begin
   ; Postscript color bar
    bardevx=[!p.clip(0),!p.clip(2)]
    bardevy=[!p.clip(1),!p.clip(3)]
    barnorm=convert_coord(bardevx,bardevy,$
     /to_normal,/device)
    barnx=barnorm(0,1)-barnorm(0,0)
```

```
   barny=barnorm(1,1)-barnorm(1,0)
   tv,bar,barnorm(0,0),barnorm(1,0),$
    /normal,xsize=barnx,ysize=barny
  endif else begin
  ; Screen color bar
   barw = 28 ; .25"
   barl = !p.clip(3)-!p.clip(1)+1
   rbar = expand(bar,barw,barl)
   tv,rbar,!p.clip(0),!p.clip(1)
  endelse
  rbar = 0   ; save memory
 ; Draw color bar axis
  axis,yaxis=0,ystyle=bstyle,yran=brange,ytitle=btitle, $
   ytype=btype,yticks=bticks,yminor=bminor,ticklen=blen, $
   charsize=charsize-.5,thick=thick,/noclip, $
   ytickv=btickv,ytickname=btickname
 end
3: begin ; Bar on Top
  preg      = [.0,.0,1.,.85]
  pmarx     = [8,1.5]
  pmary     = [3.5,2]
  !p.region = [.0,.85,1.,1.]
  !x.margin = [15,6]
  !y.margin = [0,4-xt]
     ; Locate bar area
  plot,[0,1],[0,1],/nodata,xstyle=4,ystyle=4,xtit=btitle, $
   chars=charsize-.5,back=white
 ; Make a Color Bar
  if !d.name eq 'PS' then begin
  ; Postscript color bar
   bardevx=[!p.clip(0),!p.clip(2)]
   bardevy=[!p.clip(1),!p.clip(3)]
   barnorm=convert_coord(bardevx,bardevy,$
    /to_normal,/device)
   barnx=barnorm(0,1)-barnorm(0,0)
   barny=barnorm(1,1)-barnorm(1,0)
   tv,bar,barnorm(0,0),barnorm(1,0),$
    /normal,xsize=barnx,ysize=barny
  endif else begin
  ; Screen color bar
   barw = !p.clip(2)-!p.clip(0)+1
   barl = 28  ;calculated to be .25"
   rbar = expand(bar,barw,barl)
   tv,rbar,!p.clip(0),(!p.clip(3)-28)
  endelse
  rbar = 0   ; save memory
     ; Draw color bar axis
  axis,xaxis=1,xstyle=bstyle,xran=brange,xtitle=btitle, $
```

```
   xtype=btype,xticks=bticks,xminor=bminor,ticklen=blen, $
   charsize=charsize-.5,thick=thick,/noclip, $
   xtickv=btickv,xtickname=btickname
 end
4: begin ; Bar on Bottom
  preg      = [.0,.15,1.,1.]
  pmarx     = [8,1.5]
  pmary     = [3.5,3]
  !p.region = [.0,.0,1.,.15]
  !x.margin = [15,6]
  !y.margin = [4.5-xt,-.5]
      ; Locate color bar
  plot,[0,1],[0,1],/nodata,xstyle=4,ystyle=4,xtit=btitle, $
   chars=charsize-.5,back=white
      ; Make a Color Bar
  if !d.name eq 'PS' then begin
  ; Postscript color bar
   bardevx=[!p.clip(0),!p.clip(2)]
   bardevy=[!p.clip(1),!p.clip(3)]
   barnorm=convert_coord(bardevx,bardevy,$
    /to_normal,/device)
   barnx=barnorm(0,1)-barnorm(0,0)
   barny=barnorm(1,1)-barnorm(1,0)
   tv,bar,barnorm(0,0),barnorm(1,0),$
    /normal,xsize=barnx,ysize=barny
  endif else begin
  ; Screen color bar
   barw = !p.clip(2)-!p.clip(0)+1
   barl = 28 ; .25"
   rbar = expand(bar,barw,barl)
   tv,rbar,!p.clip(0),!p.clip(1)
  endelse
  rbar = 0   ; save memory
 ; Draw color bar axis
  axis,xaxis=0,xstyle=bstyle,xran=brange,xtitle=btitle, $
   xtype=btype,xticks=bticks,xminor=bminor,ticklen=blen, $
   charsize=charsize-.5,thick=thick,/noclip, $
   xtickv=btickv,xtickname=btickname
 end
ELSE: begin ; No Bar
  preg  = [.0,.0,1.,1.]
  pmarx = [8,1]
  pmary = [4,2]
  goto,nobar
 end
endcase
!p.noerase = 1
```

```
NOBAR:
; Set the position of the contour box
!p.region = preg
!x.margin = pmarx & !y.margin = pmary

; Make the axes invisible
xss = xstyle or 4 & yss = ystyle or 4

; Set up the contour box (This is done to set !p.clip)
 contour,a,xold,yold,/nodata,XRANGE=xrange,XSTYLE=xss,XTITLE= xtitle, $
 CHARSIZE=charsize,YRANGE=yrange,YSTYLE=yss,YTICKS=yticks, $
 YTITLE=ytitle,TITLE=title,SUBTITLE=subtitle,TICKLEN=ticklen

; Calculate the dimensions for the image
xcmi = min(!x.crange) & ycmi = min(!y.crange)
xcma = max(!x.crange) & ycma = max(!y.crange)
xmin = min(xold)>xcmi & ymin = min(yold)>ycmi
xmax = max(xold)<xcma & ymax = max(yold)<ycma

; Determine whether to flip a before plotting
sizex  = size(xold) & sizey  = size(yold)
lastx  = sizex(1)-1 & lasty  = sizey(1)-1
xorder = (xmax-xmin)*(xold(lastx)-xold(0))
yorder = (ymax-ymin)*(yold(lasty)-yold(0))

; if xorder, yorder are negative,then image should be rotated 180
if(xorder lt 0 and yorder lt 0) then begin
 anew = rotate(a,2)
 x    = reverse(xold)
 y    = reverse(yold)
endif
; if xorder is negative, then image should be flipped left to right
if(xorder lt 0 and yorder gt 0) then begin
 anew = rotate(a,5)
 x    = reverse(xold)
 y    = yold
endif
; if yorder is negative, then image should be flipped top to bottom
if(xorder gt 0 and yorder lt 0) then begin
 anew = rotate(a,7)
 x    = xold
 y    = reverse(yold)
endif
; if xorder, yorder are positive, then no change
if(xorder gt 0 and yorder gt 0) then begin
 anew = a
 x    = xold
 y    = yold
```

```
endif

; Determine the size of the plot box
pdx = !p.clip(2)-!p.clip(0)+1 & pdy = !p.clip(3)-!p.clip(1)+1

; Calculate the x-limits of the array
if(!x.crange(0) le !x.crange(1)) then begin
 loc  = where(x ge xmin)
 x0   = loc(0)
 xmin = x(x0)
 loc  =  where((x gt xmax),count)
 if (count eq 0) then begin
  x1  = n_elements(x)-1
 endif else begin
  x1   = loc(0)-1
  xmax = x(x1)
 endelse
 xout = xmin
endif else begin
 loc  = where(x le xmax)
 x0   = loc(0)
 xmax = x(x0)
 loc  =  where((x lt xmin),count)
 if (count eq 0) then begin
  x1   = n_elements(x)-1
 endif else begin
  x1   = loc(0)-1
  xmin = x(x1)
 endelse
 xout = xmax
endelse

; Calculate the y-limits of the array
if(!y.crange(0) le !y.crange(1)) then begin
 loc  = where(y ge ymin)
 y0   = loc(0)
 ymin = y(y0)
 loc  =  where((y gt ymax),count)
 if (count eq 0) then begin
  y1   = n_elements(y)-1
 endif else begin
  y1   = loc(0)-1
  ymax = y(y1)
 endelse
 yout = ymin
endif else begin
 loc  = where(y le ymax)
 y0   = loc(0)
```

```
 ymax = y(y0)
 loc  =  where((y lt ymin),count)
 if (count eq 0) then begin
  y1   = n_elements(y)-1
 endif else begin
  y1   = loc(0)-1
  ymin = y(y1)
 endelse
 yout = ymax
endelse

; If set for postscript, output to postscript file convert ranges from data
; to normal for postscipt otherwise output to screen.
if !d.name eq 'PS' then begin
 idx      = 300
 idy      = 225
 normcoor = convert_coord([xmin,xmax],[ymin,ymax],/to_normal,/data)
 nx       = normcoor(0,1)-normcoor(0,0)
 ny       = normcoor(1,1)-normcoor(1,0)
 work =  expand(anew(x0:x1,y0:y1),idx,idy,maxval=maxval,fillval=fillv al)

   ; Check for nodata
 m = where(work eq fillval,num)

   ; Scale it to the color scale
 if(btype eq 0) then begin
  image = byte(topcolor/float(brange(1)-brange(0))* $
           (((work>brange(0))<brange(1))-brange(0)))+offset
 endif else begin
  logb  = alog10(brange)
  image = byte(topcolor/(logb(1)-logb(0))*(((alog10(work) $
   >logb(0))<logb(1))-logb(0)))+offset
 endelse

   ; Fill nodata with proper color
 if num ne 0 then image(m) = bkgnd

endif else begin   ; screen output

   ; Is the X-axis linear or logarithmic
 if((xtype and 1) eq 0) then begin ; Linear X
  idx  = fix((xmax-xmin)*(pdx-1)/(xcma-xcmi))+1
 endif else begin   ; Log X
  idx  = fix(alog(xmax/xmin)/alog(xcma/xcmi)*(pdx-1))+1
   ; /data does not work in LOG coordinates so fix xout
  xout = alog(xout/xcmi)/alog(xcma/xcmi)*(xcma-xcmi)+xcmi
 endelse
```

```
   ; Is the Y-axis linear or logarithmic
 if((ytype and 1) eq 0) then begin ; Linear Y
  idy= fix((ymax-ymin)*(pdy-1)/(ycma-ycmi))+1
 endif else begin   ; Log Y
  idy= fix(alog(ymax/ymin)/alog(ycma/ycmi)*(pdy-1))+1
   ; /data does not work in LOG coordinates so fix yout
  yout = alog(yout/ycmi)/alog(ycma/ycmi)*(ycma-ycmi)+ycmi
endelse

; Expand the input array to the proper image dimensions for screen
work =  expand(anew(x0:x1,y0:y1),idx/pixf+1,idy/pixf+1,maxv=maxval,f illv=fillval)

; Check for nodata
m = where(work eq fillval,num)

; Scale it to the color scale
if(btype eq 0) then begin
 work = byte(topcolor/float(brange(1)-brange(0))* $
  (((work>brange(0))<brange(1))-brange(0)))
endif else begin
 logb = alog10(brange)
 work = byte(topcolor/(logb(1)-logb(0))* $
  (((alog10(work)>logb(0))<logb(1))-logb(0)))
endelse

; Fill nodata with proper color
if(num ne 0.) then work(m) = bkgnd

   ; Replicate Pixel to Achieve Final Size
 bigp = replicate(0B,pixf,pixf)

   ; These loops are ugly but faster than alternatives
 image = bytarr(idx+pixf,idy+pixf)
 for ky=0,idy/pixf do begin
  iiy=ky*pixf
 ; Fill in the image
  for kx=0,idx/pixf do image(kx*pixf,iiy)=work(kx,ky)+bigp
 endfor

endelse

; output image to ps file or screen
if !d.name eq 'PS' then begin
 image=image(0:idx-1,0:idy-1)
  tv,image,normcoor(0,0),normcoor(1,0),/normal,xsize=nx,ysize= ny
endif else begin
   ; Trim to fit and output to window at proper position
 tv,image(0:idx-1,0:idy-1),xout,yout,/data
```

```
endelse

; Save memory
work  = 0 & image = 0

; Draw all the axes and overlay a contour if requested
 contour,anew,x,y,levels=lev,/follow,nodata=overlay,XRANGE=xr ange,XSTYLE= $
 xstyle,XTICKS=xticks,XTITLE=xtitle,xtype=xtype, YRANGE=yrange, $
  YSTYLE=ystyle,YTICKS=yticks,YTITLE=ytitle,ytype=ytype,TITLE= title, $
  SUBTITLE=subtitle,TICKLEN=ticklen,CHARSIZE=charsize,/spline, $
  XTICKNAME=xtickname,YTICKNAME=ytickname,XMINOR=xminor,YMINOR =yminor

; Skip this if postscript
If !d.name ne 'PS' then begin
   ; Place Date and Time on Plot if available
 if(time_flag) then begin
  tstr    = systime(0)
  timestr = strmid(tstr,4,3) + ' ' + strmid(tstr,8,2) + $
   ', ' + strmid(tstr,20,4)
  xyouts,.98,.02,timestr,size=1.0,/normal,align=1.
 endif

 if not keyword_set(show) then wshow,!d.window,1
endif

if keyword_set(reset) then begin
   ; Restore Saved Plotting Variables
 !x = save_x & !y = save_y & !p = save_p
 !QUIET = quiet
endif
!p.noerase = save_noerase

return
end

 ;+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++++++++
; NAME:
; EXPAND
; PURPOSE:
; Array magnification  (CONGRIDI like except that this really works!)
; CATEGORY:
; Z4 - IMAGE PROCESSING
; CALLING SEQUENCE:
; RESULT = EXPAND(A,NX,NY [,MAXVAL=MAXVAL,FILLVAL=FILLVAL])
; INPUTS:
; A Array to be magnified
; NX Desired size of X Dimension
```

```
; NY Desired size of Y Dimension
; Keywords:
; MAXVAL Largest good value. Elements greater than this are ignored
; FILLVAL Value to use when elements larger than MAXVAL are encountered.
;  Defaults to -1.
; OUTPUTS:
; Magnified Floating point image of A array (NX by NY)
; COMMON BLOCKS:
; NONE
; SIDE EFFECTS:
; NONE
; RESTRICTIONS:
; A must be two Dimensional
; PROCEDURE:
; Bilinear interpolation.
; Not really fast if you have to swap memory (eg. NX*NY is a big number).
; OK Postscript users don't forget that postscript pixels are scaleable!
; MODIFICATION HISTORY:
; Aug 15, 1989 J. M. Zawodny, NASA/LaRC, MS 475, Hampton VA, 23665.
; Aug 26, 1992 JMZ, Added maxval and fillval keywords.
; Sep 10, 1992 JMZ, converted to use INTERPOLATE function (tnx Wayne!)
; Oct  8, 1992    M.F. Ryba, MIT Lincoln Lab, converted to a function
; Please send suggestions and bugreports to zawodny@arbd0.larc.nasa.gov
;-
function EXPAND,a,nx,ny,maxval=maxval,fillval=fillval

 s=size(a)
 if(s(0) ne 2) then begin
  print,'EXPAND: *** array must be 2-Dimensional ***'
  retall  ; This will completely terminate the MAIN program!!!
 endif

   ; Get dimensions of the input array
 ix = s(1)
 iy = s(2)

   ; Calculate the new grid in terms of the old grid
 ux = (ix-1.) * findgen(nx) / (nx-1.)
 uy = (iy-1.) * findgen(ny) / (ny-1.)

   ; Are we to look for and ignore bad data? (can't use KEYWORD_SET here)
 if n_elements(maxval) eq 0 then begin
 ; NO
  result = interpolate(a,ux,uy,/grid)
 endif else begin
 ; YES then calculate the indicies and u-arrays
  mx = long(ux)<(ix-2)
  my = long(uy)<(iy-2)
```

```
  uxa = ux # replicate(1,ny)
  uya = replicate(1,nx) # uy

 ;Index vectors to A and RESULT arrays
  mxy = (mx # replicate(1L,ny)) + (replicate(long(ix),nx) # my)
  ind = lindgen(nx,ny)

 ; Fill RESULT with fill value, defaulting to -1 if none specified
  if n_elements(fillval) le 0 then fillval = -1.
  result = replicate(fillval,nx,ny)

 ; Remove those elements which would be utilizing "bad" values from A
 ; Check lower left
  m      = where(a(mxy) le maxval,num)
  if(num eq 0) then goto,out
  mxy    = mxy(m)
  ind    = ind(m)
 ; Check lower right
  m      = where(a(mxy+1) le maxval,num)
  mxy    = mxy(m)
  ind    = ind(m)
 ; Check upper left
  m      = where(a(mxy+ix) le maxval,num)
  mxy    = mxy(m)
  ind    = ind(m)
 ; Check upper right
  m      = where(a(mxy+(ix+1)) le maxval,num)
  mxy    = mxy(m)
  ind    = ind(m)

  ; Interpolate only the points which will not be the fill value
   result(ind) = interpolate(a,uxa(ind),uya(ind))
  endelse

; Done
return,result

OUT: ; If we had a problem
print,'Entire input array is greater than MAXVAL, ('+strtrim(maxval,2)+')'
return,result

end
--
Dr. Marty Ryba                  | Generation X:
Group 101 - Air Defense Techniques |
MIT Lincoln Laboratory            |  Too young to be cynical,
ryba@ll.mit.edu                 |  too old to be optimistic.
```