Subject: Spectrogram.pro for 2D spectrograms
Posted by Robert.M.Candey on Tue, 12 Nov 1996 08:00:00 GMT
View Forum Message <> Reply to Message

Below is an IDL routine (spectrogram.pro) for plotting a color spectrogram
of Z in contiguous or non-contiguous blocks with color Z(i,j). Data Z(i,j)
is shown as a box of size Xmin(i):Xmax(i) and Ymin(j):Ymax(j) with color
Z(i,j). This handles regularly and irregularly gridded data. The
align_center option is not completely correct and I welcome suggestions and
improvements Please send me notes of bugs and desired improvements. I can
also start a mailing list for announcements of updates if there is
interest.
--
Robert.M.Candey@gsfc.nasa.gov
NASA Goddard Space Flight Center, Code 632
Greenbelt, MD 20771 USA 1-301-286-6707

_____

```
function findGaps, times, gapFactor, avgDeltaT=avgDeltaT
; return array of indices into the array times for beginning of gaps
;   -1 if none; assumes no fill data
; 1995; Rick Burley
; 1996 March 17; Robert.M.Candey.1@gsfc.nasa.gov
; 1996 March 25 BC; added check for too few points to compute on
if (n_elements(gapFactor) le 0) then gapFactor = 1.5
if (n_elements(times) lt 4) then return, -1L ; or message, 'Too few points'
deltaT = times(1:*) - times
sd = stdev(deltaT, avgDeltaT)
; improve calculation of avgDeltaT
nogaps = where(abs(deltaT) le abs(avgDeltaT * gapFactor), wc)
if (wc gt 0) then begin
  sd = stdev(deltaT(nogaps), avgDeltaT)
  if (abs(sd/avgDeltaT) gt 0.5) then message, /info, $
   'DeltaTime inaccurate; gaps too big; ' + string(avgDeltaT, sd)
  gaps = where(abs(deltaT) gt abs(avgDeltaT * gapFactor))
endif
;#### could repeat until no change in sd or avgDeltatT
return, gaps
end ; findGaps

Pro Colorbar, scale, title, logZ=logZ, position=position, cCharSize=cCharSize
; Terri Martin and Robert Candey
; 21 June 1993
; added ccharsize ; 1995 July 26
; added save on !x, !y, !p; 1995 Aug 2; BC
; reduced ncolors when not enough pixels; 1995 Sep 14 BC
; added !p.multi=0; 1996 Aug 28 BC
```

```
; This procedure creates a colorbar for the right side of a spectrogram

common deviceTypeC, deviceType,file; required for inverting grayscale Postscript
xsave = !x & ysave = !y & zsave = !z & psave = !p
!p.multi = 0

if (n_elements(position) ne 0) then positiont = position else $
  positiont = [!x.window(1)+0.01, !y.window(0), !x.window(1)+0.04, !y.window(1)]
;  positiont = [0.9, 0.1, 0.93, 0.9]
if not keyword_set(logZ) then logZ=0
if (n_elements(title) le 0) then title = ''
nColors = !d.n_colors-2. ; reserve black and white at ends of colorscale
colors = bindgen(nColors) + 1B
if (n_elements(deviceType) ne 0) then if (deviceType eq 2) then $
   colors = (!d.n_colors-1B) - colors ; invert grayscale for Postscript

fontsize = 1.0
if n_elements(cCharSize) le 0 then cCharSize = 0.
if cCharSize gt 0 then begin
  fontsize = cCharSize
endif else begin
; alternative if cCharSize is undefined or le 0
  if !p.charsize gt 0. then fontsize = !p.charsize
  if !y.charsize gt 0. then fontsize = !y.charsize * fontsize
  if (!p.multi(1)>!p.multi(2)) gt 2 then fontsize = fontsize/2.
endelse

plot, [0., 1.], [0., 1.], position = positiont, $
   /nodata, /noerase, xstyle = 4, ystyle = 1+4
;axis, yaxis = 1, ystyle = 1, yrange = scale, ytype=logZ, ycharsize=fontsize, $
;  ytitle = title, ticklen = -0.02*0.78/0.04 ; adjust for narrow window

if (abs(!x.window(1)-!x.window(0))*!d.x_size le 2) then begin
  message, 'Colorbar too narrow', /info
  !x = xsave & !y = ysave & !z = zsave & !p = psave ; restore original settings
  return
endif
colorStep =  ceil(float(ncolors)/(abs(!y.window(1)-!y.window(0))*!d.y_siz e)) > 1L
; could require 2 pixels per color; colorStep = colorStep * 2L
nSteps = fix(nColors / colorStep) < nColors
;if (!d.name eq 'PS')
if (!d.flags and 1L) then begin ; has scalable pixels (Postscript)
  colorStep = 1L
  nSteps = nColors
endif
;print,'colorStep = ', colorStep, ' nSteps = ', nSteps
```

```
for i = 0L, nSteps-1 do begin
  polyfill, [0.,1.,1.,0.], (i+[0.,0.,1.,1.])/nSteps, $
   color=colors(i*colorStep), noclip=0
endfor ; i
;for i = 0L, nColors-1, colorStep do begin
;  polyfill, [0.,1.,1.,0.], (i+[0.,0.,1.,1.])/nSteps, color=colors(i), noclip=0
;endfor ; i

; replot so the box gets put back over the filled area
plot, [0., 1.], [0., 1.], position = positiont, $
   /nodata, /noerase, xstyle = 4, ystyle = 1+4
axis, yaxis = 1, ystyle = 1, yrange = scale, ytype=logZ, ycharsize=fontsize, $
   ytitle = title, ticklen = -0.02*0.78/0.04 ; adjust for narrow window

!x = xsave & !y = ysave & !z = zsave & !p = psave ; restore original settings
return
end ; colorbar

pro align_center, X, Xmin, Xmax
  ; Scheme for aligning center of box on (X, Y) position
  nX = n_elements(X)
  dx1 = abs(X(1)-X(0)) & dx2 = abs(X(2)-X(1))
  if (abs(dx1-dx2) lt 1.e-6 * min([dx1, dx2])) then begin ; evenly spaced X
    w = lindgen(nX)+1
    Xs = spline(dindgen(nX),X,dindgen(nX+2)-1.) ; add outside points
  ;  dx = (Xs(w)+Xs(w+1))/2. - (Xs(w-1)+Xs(w))/2. == (Xs(w+1) - Xs(w-1))/2.
    Xmin = (Xs(w-1)+Xs(w))/2.
    Xmax = (Xs(w)+Xs(w+1))/2.
  endif else begin ; assume log spacing
    wh = where(x le 0, wc)
    if (wc eq 0) then begin
      alogX = alog10(x)
    endif else begin
      alogX = X*0 ; all 0's
      wh = where(x gt 0, wc)
      if (wc gt 0) then alogX(wh) = alog10(X(wh))
    endelse
    alogXs = spline(dindgen(nX),alogX,dindgen(nX+2)-1.) ; add outside points
    w = lindgen(nX)+1
  ;  dx = exp((alogXs(w)+alogXs(w+1))/2.) - exp((alogXs(w-1)+alogXs(w))/2.)
    Xmin = exp((alogXs(w-1)+alogXs(w))/2.)
    Xmax = exp((alogXs(w)+alogXs(w+1))/2.)
  endelse ; log spacing
return
end ; align_center

pro spectrogram, Z, X0, Y0, X1, Y1, X2, Y2, logY=logY, center=center, $
   colorbar=colorbar, ctitle=ctitle, cscale=cscale, cCharSize=cCharSize, $
```

```
   maxValue=maxValue, minValue=minValue, fillValue=fillValue, logZ=logZ, $
   noSkipGaps=noSkipGaps, quick=quick, reduce=reduce, noclip=noclip, $
   status=status, _Extra=extra
;  , $ position=position
;+
; NAME:
;  Spectrogram
;
; PURPOSE:
;  This function plots a color spectrogram of Z in contiguous or
;  non-contiguous blocks with color Z(i,j) (data Z(i,j) is shown as a box
;  of size Xmin(i):Xmax(i) and Ymin(j):Ymax(j) with color Z(i,j))
;
; CATEGORY:
;  Graphics
;
; CALLING SEQUENCE:
;  SPECTROGRAM, Z, [X, Y]
;  SPECTROGRAM, Z, Xmin, Ymin, Xmax, Ymax
;  SPECTROGRAM, Z, Xcenter, Ycenter, Xminus, Yminus, Xplus, Yplus
;
; INPUTS:
;  Z: 2-dimensional data array
;
; OPTIONAL INPUTS:
;  X: 1-dimensional data array of size Z(*,0) or 2-dim of size Z(*,*)
;  Y: 1-dimensional data array of size Z(0,*) or 2-dim of size Z(*,*)
;
;  Xmin: 1- or 2- dim array of X values for left side of each data box
;  Ymin: 1- or 2- dim array of Y values for bottom side of each data box
;  Xmax: 1- or 2- dim array of X values for right side of each data box
;  Ymax: 1- or 2- dim array of Y values for top side of each data box
;
;  Xcenter:1- or 2- dim array of X values for center side of each data box
;  Ycenter:1- or 2- dim array of Y values for center side of each data box
;  Xminus:  Xcenter-Xminus defines left side of each data box
;  Yminus:  Ycenter-Yminus defines bottom side of each data box
;  Xplus:   Xcenter+Xplus defines right side of each data box
;  Yplus:   Ycenter+Yplus defines top side of each data box
;  (this allows gaps or overlaps in boxes; if not desired, use Z,X,Y case)
;
; KEYWORD PARAMETERS:
;  LOGZ=logZ:  Scale Z data and color scale logarithmically
;  LOGY=logY:  Scale Y axis logarithmically
;  MAXVALUE=maxValue: Max value of data to plot; values above are ignored
;  MINVALUE=minValue: Min value of data to plot; values below are ignored
;  FILLVALUE=fillValue: Data with this value are ignored
;  COLORBAR=colorbar:  Switch to create color bar on right
```

```
;  CTITLE=ctitle: String title for colorbar
;  CSCALE=cscale: scale for colorbar range [min, max];
;  CCHARSIZE=cCharSize: Character size for axis on color bar
;  CENTER=center: Center boxes on (X,Y) location (only for 3 parameter
;           case: spectrogram, Z, X, Y, /center)
;  NOSKIPGAPS=noSkipGaps: Turns off treating large delta X as missing data
;     and skip; not done anyway if Center option selected
;  QUICK=quick:   Allow quick and dirty plotting ignoring X and Y sizes
;           (for X and Z plot devices only)
;  REDUCE=reduce: Reduce the number of X values to polyfill to not more
;          than twice the number of pixels across the plot, by
;          sampling every so many values; for non-Postscript
;          devices only; done for speed, alternative to /quick
;  NOCLIP=noclip: Polyfill bug in Z device; defaults to noclip=0
;  STATUS=status:  Return 0 if plot okay, else -1 for an error,
;          status variable must be predefined before call
;  _EXTRA=extra:  Any extra parameters to pass on to plot outline
;          Add your own title, xtitle, ytitle
;       May be able to over-ride plot location/size with position
;
; OUTPUTS:
;  No outputs.
;
; COMMON BLOCKS:
;  DEVICETYPEC: deviceType
;  Shared with DeviceOpen.pro to allow inverting grayscale Postscript
;
; SIDE EFFECTS:
;  Creates plot to screen or file.
;
; RESTRICTIONS:
;  Sets a specific X margin to allow for the colorbar.
;
; PROCEDURE:
;  Uses plot,/nodata to setup the plot area and then uses polyfill to
;  on each data value to color each small square of the spectrogram
;  A colorbar is plotted on the right if cscale is set.
;
; EXAMPLE:
;  Create a spectrogram plot of 2 dimensional data Z = dist(50)
;  spectrogram, dist(50), /colorbar
;   spectrogram,dist(50),findgen(50),findgen(50)+1,findgen(50),f indgen(50)+1
;
; MODIFICATION HISTORY:
;  Written by: Bobby Candey, NASA GSFC Code 632, 1993 August 27
;     Robert.M.Candey.1@gsfc.nasa.gov
;  1993 Nov 9  BC, removed timeaxis call and made more generic
;  1994 Sept 19   BC, update with documentation and higher level routine
```

```
;  1994 Nov 28 BC, added handling of 2-dim X and Y
;  1994 Dec 6  BC, merged routines and added center-minus-plus option
;  1995 April 10  BC, completed initial coding of version 5
;  1995 Jun 22 BC, added smaller font size for color bar axis
;  1995 July 26   BC, added cCharSize and reforming
;  1995 Oct 11 BC, added min/maxValue scaling to bytscl
;  1996 March 17  BC, added skip over time gaps and added fillValue
;  1996 March 18  BC, added /reduce for speed plotting
;  1996 March 25  BC, added noclip keyword for Z device bug with polyfill
;  1996 April 8   BC, added status keyword and plot a blank on all fill
;  1996 April 16  BC, moved Cscale to override min/maxValue
;  1996 April 17  BC, added colorBar switch to allow autoscale
;  1996 August 28 BC, added save for !p.multi so overplot works
;-

common deviceTypeC, deviceType, file;required for inverting grayscale Postscript

if (n_params(0) lt 1) then $
   message, 'spectrogram, Z, [X, Y] or [Xmin, Ymin, Xmax, Ymax]'
if (n_elements(status) gt 0) then doStatus = 1 else doStatus = 0
status = 0L
;if doStatus then begin
;  catch, error_status
;  if (error_status ne 0) then begin
;    message, 'General error: '+string(error_status)+': '+!err_string, /info
;    status = -1L & return
;  endif ; else return
;endif

Z = reform(Z) ; remove extraneous dimensions
Zsize = size(Z)
if (Zsize(0) ne 2) then begin
  msgText = 'Requires 2-dimensional Z array'
  if doStatus then begin
    message, msgText, /info & status = -1L & return
  endif else message, msgText
endif

case n_params(0) of
1: begin ; Z only
  Xmin = rebin(dindgen(Zsize(1)),Zsize(1),Zsize(2),/sample) & Xmax = Xmin + 1
  Ymin = rebin(dindgen(1,Zsize(2)),Zsize(1),Zsize(2),/sample) & Ymax = Ymin + 1
end ; Z only
3: begin ; Z, X, Y
  X0 = reform(X0) & Y0 = reform(Y0) ; remove extraneous dimensions
  Vsize = size(X0) & if not ((Vsize(Vsize(0)+2) eq Zsize(1)) or $
      ((Vsize(1) eq Zsize(1)) and (Vsize(2) eq Zsize(2)))) then begin
    msgText = 'X must be of same size as Z(*,0) or Z(*,*)'
```

```
    if doStatus then begin
      message, msgText, /info & status = -1L & return
    endif else message, msgText
  endif
  if not (Vsize(Vsize(0)+2) eq Zsize(1)) then Xmin = X0 else $
   Xmin = rebin(X0,Zsize(1),Zsize(2),/sample)
  Vsize = size(Y0) & if not ((Vsize(Vsize(0)+2) eq Zsize(2)) or $
      ((Vsize(1) eq Zsize(1)) and (Vsize(2) eq Zsize(2)))) then begin
    msgText = 'Y must be of same size as Z(0,*) or Z(*,*)'
    if doStatus then begin
      message, msgText, /info & status = -1L & return
    endif else message, msgText
  endif
  if not (Vsize(Vsize(0)+2) eq Zsize(2)) then Ymin = Y0 else $
   Ymin =  rebin(reform(Y0,1,Zsize(2),/overwrite),Zsize(1),Zsize(2),/sa mple)
  if not keyword_set(center) then begin
  ; Scheme for aligning lower left corner of box on (X, Y) position
;    Xmax = [X(1:*), X(nX-1)*2 - X(nX-2)] ; add deltaX to last item; 1 dim case
    Xmax = shift(Xmin,-1,0)   ; shift all elements in 1st dim to the left
    Xmax(Zsize(1)-1,*) = Xmin(Zsize(1)-1,*)*2 - Xmin(Zsize(1)-2,*)
    Ymax = shift(Ymin,0,-1)   ; shift all elements in 2nd dim to the bottom
    Ymax(*,Zsize(2)-1) = Ymin(*,Zsize(2)-1)*2 - Ymin(*,Zsize(2)-2)
    if not keyword_set(noSkipGaps) then begin
;     #### only uses first row of Xmin; assumes no fill data
      gaps = findGaps(Xmin(*,0), 1.5, avg=avgDeltaX)
      if (gaps(0) lt 0) then nGaps = 0 else nGaps = n_elements(gaps)
      if (nGaps gt 0) then for k = 0L, nGaps-1 do $
      Xmax(gaps(k),*) = Xmin(gaps(k),*) + avgDeltaX
    endif ; noSkipGaps
  endif else begin ; center box
;    align_center, X, Xmin, Xmax ; 1 dim case
    Xmax = Xmin
    for i = 0L, Zsize(2)-1 do begin
      align_center, Xmin(*,i), Xmint, Xmaxt
      Xmin(*,i) = Xmint & Xmax(*,i) = Xmaxt
    endfor ; i
    Ymax = Ymin
    for i = 0L, Zsize(1)-1 do begin
      align_center, Ymin(i,*), Ymint, Ymaxt
      Ymin(i,*) = Ymint & Ymax(i,*) = Ymaxt
    endfor ; i
  endelse ; centering
end ; Z, X, Y

5: begin ; Z, Xmin, Ymin, Xmax, Ymax
  X0 = reform(X0) & Y0 = reform(Y0) ; remove extraneous dimensions
  X1 = reform(X1) & Y1 = reform(Y1) ; remove extraneous dimensions
  Vsize = size(X0) & if not ((Vsize(Vsize(0)+2) eq Zsize(1)) or $
```

```
      ((Vsize(1) eq Zsize(1)) and (Vsize(2) eq Zsize(2)))) then begin
    msgText = 'Xmin must be of same size as Z(*,0) or Z(*,*)'
    if doStatus then begin
      message, msgText, /info & status = -1L & return
    endif else message, msgText
  endif
  if not (Vsize(Vsize(0)+2) eq Zsize(1)) then Xmin = X0 else $
   Xmin = rebin(X0,Zsize(1),Zsize(2),/sample)
  Vsize = size(Y0) & if not ((Vsize(Vsize(0)+2) eq Zsize(2)) or $
      ((Vsize(1) eq Zsize(1)) and (Vsize(2) eq Zsize(2)))) then begin
    msgText = 'Ymin must be of same size as Z(0,*) or Z(*,*)'
    if doStatus then begin
      message, msgText, /info & status = -1L & return
    endif else message, msgText
  endif
  if not (Vsize(Vsize(0)+2) eq Zsize(2)) then Ymin = Y0 else $
   Ymin =  rebin(reform(Y0,1,Zsize(2),/overwrite),Zsize(1),Zsize(2),/sa mple)
  Vsize = size(X1) & if not ((Vsize(Vsize(0)+2) eq Zsize(1)) or $
      ((Vsize(1) eq Zsize(1)) and (Vsize(2) eq Zsize(2)))) then begin
    msgText = 'Xmax must be of same size as Z(*,0) or Z(*,*)'
    if doStatus then begin
      message, msgText, /info & status = -1L & return
    endif else message, msgText
  endif
  if not (Vsize(Vsize(0)+2) eq Zsize(1)) then Xmax = X1 else $
   Xmax = rebin(X1,Zsize(1),Zsize(2),/sample)
  Vsize = size(Y1) & if not ((Vsize(Vsize(0)+2) eq Zsize(2)) or $
      ((Vsize(1) eq Zsize(1)) and (Vsize(2) eq Zsize(2)))) then begin
    msgText = 'Ymax must be of same size as Z(0,*) or Z(*,*)'
    if doStatus then begin
      message, msgText, /info & status = -1L & return
    endif else message, msgText
  endif
  if not (Vsize(Vsize(0)+2) eq Zsize(2)) then Ymax = Y1 else $
   Ymax =  rebin(reform(Y1,1,Zsize(2),/overwrite),Zsize(1),Zsize(2),/sa mple)
end ; Z, Xmin, Ymin, Xmax, Ymax

7: begin ; Z, Xcenter, Ycenter, Xminus, Yminus, Xplus, Yplus
  X0 = reform(X0) & Y0 = reform(Y0) ; remove extraneous dimensions
  X1 = reform(X1) & Y1 = reform(Y1) ; remove extraneous dimensions
  X2 = reform(X2) & Y2 = reform(Y2) ; remove extraneous dimensions
  Vsize = size(X0) & if not ((Vsize(Vsize(0)+2) eq Zsize(1)) or $
      ((Vsize(1) eq Zsize(1)) and (Vsize(2) eq Zsize(2)))) then begin
    msgText = 'Xcenter must be of same size as Z(*,0) or Z(*,*)'
    if doStatus then begin
      message, msgText, /info & status = -1L & return
    endif else message, msgText
  endif
```

```
if not (Vsize(Vsize(0)+2) eq Zsize(1)) then Xcenter = X0 else $
 Xcenter = rebin(X0,Zsize(1),Zsize(2),/sample)
Vsize = size(Y0) & if not ((Vsize(Vsize(0)+2) eq Zsize(2)) or $
     ((Vsize(1) eq Zsize(1)) and (Vsize(2) eq Zsize(2)))) then begin
  msgText = 'Ycenter must be of same size as Z(0,*) or Z(*,*)'
  if doStatus then begin
    message, msgText, /info & status = -1L & return
  endif else message, msgText
endif
if not (Vsize(Vsize(0)+2) eq Zsize(2)) then Ycenter = Y0 else $
   Ycenter =  rebin(reform(Y0,1,Zsize(2),/overwrite),Zsize(1),Zsize(2),/sa mple)
Vsize = size(X1) & if not ((Vsize(Vsize(0)+2) eq Zsize(1)) or $
 (Vsize(Vsize(0)+2) eq 1) or $ ; allow scalar
     ((Vsize(1) eq Zsize(1)) and (Vsize(2) eq Zsize(2)))) then begin
  msgText = 'Xminus must be of same size as Z(*,0) or Z(*,*)'
  if doStatus then begin
    message, msgText, /info & status = -1L & return
  endif else message, msgText
endif
if not (Vsize(Vsize(0)+2) eq Zsize(1)) then Xminus = X1 else $
 Xminus = rebin(X1,Zsize(1),Zsize(2),/sample)
Vsize = size(Y1) & if not ((Vsize(Vsize(0)+2) eq Zsize(2)) or $
 (Vsize(Vsize(0)+2) eq 1) or $ ; allow scalar
     ((Vsize(1) eq Zsize(1)) and (Vsize(2) eq Zsize(2)))) then begin
  msgText = 'Yminus must be of same size as Z(0,*) or Z(*,*)'
  if doStatus then begin
    message, msgText, /info & status = -1L & return
  endif else message, msgText
endif
if not (Vsize(Vsize(0)+2) eq Zsize(2)) then Yminus = Y1 else $
   Yminus =  rebin(reform(Y1,1,Zsize(2),/overwrite),Zsize(1),Zsize(2),/sa mple)
Vsize = size(X2) & if not ((Vsize(Vsize(0)+2) eq Zsize(1)) or $
 (Vsize(Vsize(0)+2) eq 1) or $ ; allow scalar
     ((Vsize(1) eq Zsize(1)) and (Vsize(2) eq Zsize(2)))) then begin
  msgText = 'Xplus must be of same size as Z(*,0) or Z(*,*)'
  if doStatus then begin
    message, msgText, /info & status = -1L & return
  endif else message, msgText
endif
if not (Vsize(Vsize(0)+2) eq Zsize(1)) then Xplus = X2 else $
 Xplus = rebin(X2,Zsize(1),Zsize(2),/sample)
Vsize = size(Y2) & if not ((Vsize(Vsize(0)+2) eq Zsize(2)) or $
 (Vsize(Vsize(0)+2) eq 1) or $ ; allow scalar
     ((Vsize(1) eq Zsize(1)) and (Vsize(2) eq Zsize(2)))) then begin
  msgText = 'Yplus must be of same size as Z(0,*) or Z(*,*)'
  if doStatus then begin
    message, msgText, /info & status = -1L & return
  endif else message, msgText
```

```
  endif
  if not (Vsize(Vsize(0)+2) eq Zsize(2)) then Yplus = Y2 else $
     Yplus =  rebin(reform(Y2,1,Zsize(2),/overwrite),Zsize(1),Zsize(2),/sa mple)

; remember X/Yminus and X/Yplus can be scalar as well as 2-dim arrays
  Xmin = Xcenter - Xminus & Xmax = Xcenter + Xplus
  Ymin = Ycenter - Yminus & Ymax = Ycenter + Yplus
end ; Z, Xcenter, Ycenter, Xminus, Yminus, Xplus, Yplus

else: begin
    msgText = 'Wrong number of arguments'
    if doStatus then begin
      message, msgText, /info & status = -1L & return
    endif else message, msgText
  end ; else
endcase

;##### remove "> 0." on next 30 lines for real Z < 0
;if (n_elements(minValue) gt 0) then minZ = minValue(0) else minZ = min(Z) > 0.
;if (n_elements(maxValue) gt 0) then maxZ = maxValue(0) else maxZ = max(Z) > 0.
if (n_elements(minValue) gt 0) then minZ = minValue(0) else minZ = min(Z)
if (n_elements(maxValue) gt 0) then maxZ = maxValue(0) else maxZ = max(Z)
if (n_elements(fillValue) gt 0) then begin
  fillZ = fillValue(0)
  wZfill = where(Z eq fillZ, wZfillc)
  if (wZfillc gt 0) then begin
    wnotZfill = where(Z ne fillZ, wnotZfillc)
    if (wnotZfillc gt 0) then begin
      if (n_elements(minValue) le 0) then minZ = min(Z(wnotZfill))
      if (n_elements(maxValue) le 0) then maxZ = max(Z(wnotZfill))
    endif
  endif
endif else fillZ = minZ - 1

isBad = (Z lt minZ) or (Z gt maxZ) or (Z eq fillZ)
wBad = where(isBad, wBadc)
if (wBadc gt 0) then begin
  wGood = where(isBad ne 1, wGoodc)
  if (wGoodc le 0) then begin ; quit here
    msgText = 'No good values to display'
    if doStatus then begin
      if not keyword_set(logY) then logY = 0
      plot, [0,1], [0,1], ytype=logY, /nodata, _Extra=extra
      message, msgText, /info & status = -1L & return
    endif else message, msgText
  endif
;  if (n_elements(minValue) eq 0) then minZ=min(Z(wGood)) > 0.
;  if (n_elements(maxValue) eq 0) then maxZ=max(Z(wGood)) > 0.
```

```
  if (n_elements(minValue) le 0) then minZ=min(Z(wGood))
  if (n_elements(maxValue) le 0) then maxZ=max(Z(wGood))
  if (n_elements(fillValue) le 0) then fillZ = minZ - 1
endif

if not keyword_set(logZ) then logZ = 0
doColorBar = 0
flipColorBar = 0 ; flip color bar if cScale is inverted
if (n_elements(cscale) ne 0) then begin
  doColorBar = 1
  ; check accuracy of cscale
  if n_elements(cscale) ne 2 then begin
     msgText = 'Error in cscale dimensions, no colorbar plotted'
     if doStatus then begin
      message, msgText, /info & status = -1L & return
    endif else message, msgText
  endif
  if logZ then begin ; check if cscale is less than 0 and minValue
    if (n_elements(minValue) ne 0) then minCscale = minValue(0) > 0 $
              else minCscale = 0
    wcs = where(cscale le minCscale, wcsc)
    if wcsc gt 0 then begin
      ws = where(Z gt minCscale, wsc)
      if wsc gt 0 then cscale(wcs) = min(Z(ws)) else cscale(wcs) = minCscale
    endif ; bad cscale
  endif ; logZ
  doCheck = 0
  if doCheck then begin ; check for exceeding maxValue or less than minValue
    if (n_elements(maxValue) ne 0) then begin
      wcs = where(cscale gt maxValue(0), wcsc) ; could be "ge"
      if wcsc gt 0 then begin
        ws = where(Z le maxValue(0), wsc) ; could be "lt"
        if wsc gt 0 then cscale(wcs) = max(Z(ws)) else cscale(wcs) = maxValue(0)
      endif
    endif
    if (n_elements(minValue) ne 0) then begin
      wcs = where(cscale lt minValue(0), wcsc) ; could be "le"
      if wcsc gt 0 then begin
        ws = where(Z ge minValue(0), wsc) ; could be "gt"
        if wsc gt 0 then cscale(wcs) = min(Z(ws)) else cscale(wcs) = minValue(0)
      endif
    endif
  endif ; doCheck min/max Values in cscale
  minZ = min(cscale) & maxZ = max(cscale)
  if (cscale(0) gt cscale(1)) then flipColorBar = 1
  if (n_elements(fillValue) le 0) then fillZ = minZ - 1
endif else begin; colorBar without cscale
  if keyword_set(colorbar) then begin
```

```
    doColorBar = 1
    cscale = [minZ, maxZ]
  endif
endelse

minZ1 = minZ & maxZ1 = maxZ
Zt = Z
if (Zsize(Zsize(0)+1) ne 1) then begin ; not Byte array
;  if keyword_set(logZ) then begin
  if logZ then begin
    wh = where(Z le 0, wc)
    if (wc eq 0) then begin
      Zt = alog10(Z)
    endif else begin
      Zt = Z*0 ; all 0's
      wh = where(Z gt 0, wc)
      if (wc gt 0) then Zt(wh) = alog10(Z(wh))
    endelse
    if (minZ le 0.) then minZ1 = 0. else minZ1 = alog10(minZ)
    if (maxZ le 0.) then maxZ1 = 0. else maxZ1 = alog10(maxZ)
  endif
  Zt = bytscl(Zt, min=minZ1, max=maxZ1, top=!d.n_colors-3)+1B
   ; reserve black and white at ends of colorscale
endif
if (wBadc gt 0) then Zt(wBad) = 0B ; minZ1

if (n_elements(deviceType) ne 0) then if (deviceType eq 2) then $
   Zt = (!d.n_colors-1B) - Zt ; invert grayscale for Postscript
if flipColorBar then Zt = (!d.n_colors-1B) - Zt ; invert for inverted cscale

xmargin = !x.margin
;ymargin = !y.margin
if doColorBar then if (!x.omargin(1)+!x.margin(1)) lt 14 then !x.margin(1) = 14
;pPosition = !p.position ; save for later
;w = where(pPosition eq 0, wc)
;if (n_elements(position) ne 0) then positiont = position else $
;    if (wc ne 4) then positiont = pPosition else $
;  positiont = [0.1, 0.1, 0.88, 0.9]
if not keyword_set(logY) then logY = 0

Xminmax = [min([Xmin,Xmax], max=maxt), maxt]
Yminmax = [min([Ymin,Ymax], max=maxt), maxt]

; make any changes here also to plot command below polyfill
pmulti = !p.multi ; save so overwrite plot command will work
plot, Xminmax, Yminmax, $ ; position = positiont, $
        ytype=logY, /nodata, _Extra=extra
;        ytype=logY, /nodata, xstyle=1,   ystyle=1, _Extra=extra
```

```
;notime; ytype=logY, /nodata, xstyle=1+4, ystyle=1, ticklen=-0.01, _Extra=extra
;notime; timeaxis, tick=-1
  px=!x.window*!d.x_size
  py=!y.window*!d.y_size
  xWinsize=px(1)-px(0)
  yWinsize=py(1)-py(0)

if keyword_set(quick) and ((!d.name eq 'X') or (!d.name eq 'Z') or $
    (!d.name eq 'WIN') or (!d.name eq 'MAC') or (!d.name eq 'SUN')) then begin
  ; quick and dirty plotting
  tv,congrid(Zt,xWinsize,yWinsize),px(0),py(0)
endif else begin
  skipX = 1L & avgDeltaX = 1
  if keyword_set(reduce) and not (!d.flags and 1L) then begin
    ; not scalable pixels (Postscript)
    gaps = findGaps(Xmin(*,0), 1.1, avg=avgDeltaX)
; #### uses only first row of Xmin and assumes no fill data in Xmin
    skipX = long((!x.crange(1)-!x.crange(0)) / avgDeltaX / xWinsize) $ ; -1?
      > 1L < (Zsize(1)/2)
print, 'Spectrogram SkipX = ', skipX
  endif ; reduce
  if (n_elements(noclip) gt 0) then noclip = noclip(0) else noclip = 0

  for i = 0L, Zsize(1)-1, skipX do begin
    for j = 0L, Zsize(2)-1 do begin
      doPixel = 1 ; plot all data (no maximum value limit or below limit)
;##### why not use minZ, maxZ?
    if (n_elements(maxValue) ne 0) then if (Z(i,j) gt maxValue(0)) then doPixel=0
    if (n_elements(minValue) ne 0) then if (Z(i,j) lt minValue(0)) then doPixel=0
      if (n_elements(fillValue) ne 0) then if (Z(i,j) eq fillZ) then doPixel=0
      if (doPixel eq 1) then begin
;       polyfill, [0,1,1,0] * (Xmax(i,j) - Xmin(i,j)) + Xmin(i,j), $
        polyfill, [0,1,1,0] * ((Xmax(i,j)+(skipX-1L)*avgDeltaX) - Xmin(i,j)) $
                  + Xmin(i,j), $
        [0,0,1,1] * (Ymax(i,j) - Ymin(i,j)) + Ymin(i,j), $
        color = Zt(i,j), noclip=noclip
; #### problem when right on clip boundary? use noclip=1 for Z device
      endif
    endfor ; j
  endfor ; i
endelse

; replot so the box gets put back over the filled area
pmulti2 = !p.multi
!p.multi = pmulti ; restore from before first plot command
plot, Xminmax, Yminmax, $ ; position = positiont, $
   _Extra=extra, ytype=logY, /nodata, /noerase
;        ytype=logY, /nodata, xstyle=1,   ystyle=1, _Extra=extra
```

```
;notime; ytype=logY, /nodata, xstyle=1+4, ystyle=1, ticklen=-0.01, _Extra=extra
;notime; timeaxis, tick=-1
!p.multi = pmulti2 ; restore from after first plot command

if doColorBar then begin
  if (n_elements(ctitle) le 0) then ctitle = ''
  if (n_elements(cCharSize) le 0) then cCharSize = 0.
  xwindow = !x.window
  offset = 0.01
  colorbar, cscale, ctitle, logZ=logZ, cCharSize=cCharSize, $
   position=[!x.window(1)+offset,      !y.window(0),$
        !x.window(1)+offset+0.03, !y.window(1)]
  !x.window = xwindow
endif ; colorbar

;!p.Position = pPosition ; restore
!x.margin = xmargin
;!y.margin = ymargin
return
end ; spectrogram
```