Subject: Error Bars in X & Y directions ...
Posted by shah on Tue, 14 Dec 1993 17:22:32 GMT
View Forum Message <> Reply to Message

Hello there,

This is in continuation with my query on the net for
generating XY plots with XY Error Bars. In response
to my initial post there were several e-mail messages
that I got. A summary of those responses is attached.
To those who responded (Jim Pendleton, William Thompson
and Jon Saken) I am extremely grateful; Thank You.

To move on. The problem can be solved in several ways.
There are programs in the various public domain
libraries which do it efficiently. For my purposes I
modified a program (ploterr.pro; yes, it has the same
name as the IDL library file, but it is much more powerful),
from the idlastro directory (address etc., in the FAQ of IDL).
In addition, there are the procedures EPLOT and EOPLOT. These
are also very good and are written by Jim Pendleton. For those
who are interested I am appending Mr Pendleton's mail message,
it includes the procedures (I hope it is all right to do so
because I have not taken his opinion on doing so).

Other than the above 2 methods I have written my own code.
After some excellent comments from Mr William Thompson it
is working quite well. However, the above alternatives are
better. My code is rather bare boned!

My thanks to all those who responded.

Copies of EPLOT, EOPLOT and PLOTERR are attached for those
who are interested.

Regards

Safwan SHAH

BioServe Space Technologies
Aerospace Engineering Sciences
University of Colorado at Boulder

==========

Jim Pendletons response...

==========

From PENDLETON@ossenu.astro.nwu.edu Fri Dec 3 08:46:12 1993

> From: shah@spot.Colorado.EDU (SSS)
> Subject:Help, somebody! Error Bars in both X and Y directions.
> Date: Fri, 3 Dec 1993 07:24:33 GMT
> Message-ID:<CHG78y.GnG@cnsnews.Colorado.EDU>

>
> Hello there,
>
> I have been fruitlessly trying to implement an IDL code to
> create errorbars in both x and y directions. However, my data
> on the ordinate (y axis) is at a very different range than the abcissae
> (x axis). For example, xrange=[0.75,1.25]; yrange=[0.01,0.05]. The
> following procedure, obviously wrong, is able to produce the
> correct x,y mapping but, alas, the errorbars do not correspond to
> their respective x and y scales.
>

Below are my procedures EPLOT and EOPLOT. Give 'em a try!

Jim Pendleton, Programmer Analyst/Technical Services Specialist
GRO/OSSE, Dept. Physics & Astronomy
Northwestern University
j-pendleton@nwu.edu (708) 491-2748   (708) 491-3135 [FAX]

```
Pro EOPlot, X, Y, YL = YL, YU = YU, XL = XL, XU = XU, BarSize = BarSize, $
 Relative_Errors = Relative_Errors, Help = Help
;+
;Module:
; EOPlot
;Author:
; Jim Pendleton, ST ScI
; 8/16/88
;Modified:
; 9/14/90 by JLP
;  Modified for IDL V2.
;Purpose:
; Given a set of (X,Y) data points, OPLOT them along with error
; bars, if errors are passed, on the currently active plot.  This
; procedure allows single-valued (non-array) X, Y, and errors.
;Input variables:
; X is the scalar or array of X-axis values
; Y is the corresponding scalar or array of Y-axis values
;Keyword parameters:
; YL is the scalar or array of lower bound errors in Y
; YU is the scalar or array of upper bound errors in Y
; XL is the scalar or array of lower bound errors in X
```

```
; XU is the scalar or array of upper bound errors in X
; Barsize is the normalized size of the bar end crosses, default = .005;
;  if 0 is input, then no end bars are plotted on the error bars
;Keywords:
; /Relative_Errors specifies whether the errors are relative
;  (e.g. Y +/- 3) or absolute bounds (e.g. -2 < Y < 5)
;Output variables:
; No values are modified or passed back from this routine.
;Calling sequence:
; EOPlot, X, Y, YL=YL, YU=YU, XL=XL, XU=XU, /Relative_Errors, $
;  BarSize=BarSize
;-
; Return immediately in case of any errors.
;
On_Error, 2
If (Keyword_Set(Help)) then Begin
 Doc_Library, 'EOPlot'
 Return
EndIf
;
; Save the input X and Y arrays
;
Xsave = X & Ysave = Y
;
; If a scalar point was passed, make it an array element.
;
If (N_elements(X) eq 1) then Begin
 X = [Xsave]
 Y = [Ysave]
EndIf
;
; Set a flag if the Relative_Errors keyword was specified.
;
Relative = Keyword_Set(Relative_Errors)
;
; If no X lower bounds were passed, make sure error bars won't be plotted.
;
If (N_elements(XL) eq 0) then Begin
 XL = X
 If (Relative) then XL(*) = 0
EndIf
;
; If no X upper bounds were passed, make sure error bars won't be plotted.
;
If (N_elements(XU) eq 0) then Begin
 XU = X
 If (Relative) then XU(*) = 0
EndIf
```

```
;
; If no Y lower bounds were passed, make sure error bars won't be plotted.
;
If (N_elements(YL) eq 0) then Begin
 YL = Y
 If (Relative) then YL(*) = 0
EndIf
;
; If no Y upper bounds were passed, make sure error bars won't be plotted.
;
If (N_elements(YU) eq 0) then Begin
 YU = Y
 If (Relative) then YU(*) = 0
EndIf
;
; Make the errors relative, regardless of how they were passed and use their
; absolute values.
;
If (Relative) then Begin
 XerrL = Abs(XL)
 XerrU = Abs(XU)
 YerrL = Abs(YL)
 YerrU = Abs(YU)
EndIf Else Begin
 XerrL = Abs(X - XL)
 XerrU = Abs(X - XU)
 YerrL = Abs(Y - YL)
 YerrU = Abs(Y - YU)
EndElse
;
; Only make X error bars where there are non-zero errors.
;
Xbars = Where((XerrL ne 0.) or (XerrU ne 0.))
If (Xbars(0) ne -1) then Begin
 NXbars = N_elements(Xbars) - 1
EndIf Else Begin
 NXBars = -1
EndElse
;
; Only make Y error bars where there are non-zero errors.
;
Ybars = Where((YerrU ne 0.) or (YerrL ne 0.))
If (Ybars(0) ne -1) then Begin
 NYbars = N_elements(Ybars) - 1
EndIf Else Begin
 NYBars = -1
EndElse
;
```

```
; Oplot the input data values
;
Oplot, X, Y
;
; If error bar end cross length was not passed, set a default size.
;
If (N_elements(BarSize) eq 0) then Begin
 BarSize = .005
EndIf
If (NXbars gt -1) then Begin
;
; We've got some error bars in X.  Loop over non-zero.
;
 For I = 0, NXbars Do Begin
;
; Get the index of the next point with error bars.
;
  J = Xbars(I)
;
; Determine the upper and lower limit of the error bar along X.
;
  XlimL = X(J) - XerrL(J)
  XlimU = X(J) + XerrU(J)
;
; Plot the error bar.
;
  Oplot, [XlimL, XlimU], [Y(J), Y(J)], Linestyle = 0, Psym = 0
  If (BarSize gt 0) then Begin
;
; Add cross bars perpendicular to the end of the error bar.
;
   If (!Y.Type eq 0) then Begin
;
; Linear scale along Y.  Convert to normalized coordinates, and create the
; cross bar end points.
;
    YMidNormal = !Y.S(1)*Y(J) + !Y.S(0)
    YBarEnds = [YMidNormal + BarSize, $
     YMidNormal - BarSize]
;
; Convert back to data coordinates.
;
    YBarEnds = (YBarEnds - !Y.S(0))/!Y.S(1)
   EndIf Else Begin
;
; Logarthmic scale along Y.  Convert to normalized coordinates, and create the
; cross bar end points.
;
```

```
    YMidNormal = !Y.S(1)*Alog10(Y(J)) + !Y.S(0)
    YBarEnds = [YMidNormal + BarSize, $
     YMidNormal - BarSize]
;
; Convert back to data coordinates.
;
    YBarEnds = 10^((YBarEnds - !Y.S(0))/!Y.S(1))
   EndElse
;
; Plot the cross bars at the ends of the error bars.
;
   Oplot, [XlimL, XlimL], YBarEnds, Linestyle = 0, Psym = 0
   Oplot, [XlimU, XlimU], YBarEnds, Linestyle = 0, Psym = 0
  EndIf
 EndFor
EndIf
If (NYbars gt -1) then Begin
;
; We've got some error bars in Y.  Loop over non-zero.
;
 For I = 0, NYbars Do Begin
;
; Get the index of the next point with error bars.
;
  J = Ybars(I)
;
; Determine the upper and lower limit of the error bar along Y.
;
  YlimL = Y(J) - YerrL(J)
  YlimU = Y(J) + YerrU(J)
;
; Plot the error bar.
;
  Oplot, [X(J), X(J)], [YlimL, YlimU], Linestyle = 0, Psym = 0
  If (BarSize gt 0) then Begin
;
; Add cross bars perpendicular to the end of the error bar.
;
   If (!X.Type eq 0) then Begin
;
; Linear scale along X.  Convert to normalized coordinates, and create the
; cross bar end points.
;
    XMidNormal = !X.S(1)*X(J) + !X.S(0)
    XBarEnds = [XMidNormal + BarSize, $
     XMidNormal - BarSize]
;
; Convert back to data coordinates.
```

```
;
    XBarEnds = (XBarEnds - !X.S(0))/!X.S(1)
   EndIf Else Begin
;
; Logarthmic scale along X.  Convert to normalized coordinates, and create the
; cross bar end points.
;
    XMidNormal = !X.S(1)*Alog10(X(J)) + !X.S(0)
    XBarEnds = [XMidNormal + BarSize, $
     XMidNormal - BarSize]
;
; Convert back to data coordinates.
;
    XBarEnds = 10^((XBarEnds - !X.S(0))/!X.S(1))
   EndElse
;
; Plot the cross bars at the ends of the error bars.
;
   Oplot, XBarEnds, [YlimL, YlimL], Linestyle = 0, Psym = 0
   Oplot, XBarEnds, [YlimU, YlimU], Linestyle = 0, Psym = 0
  EndIf
 EndFor
EndIf
;
; Reset the X and Y values.
;
X = Xsave & Y = Ysave
Return
End


Pro EPlot, X, Y, YL = YL, YU = YU, XL = XL, XU = XU, BarSize = BarSize, $
 Relative_Errors = Relative_Errors, LogX = LogX, LogY = LogY, $
 XRange = XRange, YRange = YRange, Help = Help
;+
;Module:
; EPlot
;Author:
; Jim Pendleton, ST ScI
; 8/16/88
;Modified:
; 9/14/90 by JLP
;  Modified for IDL V2.
;Purpose:
; Given a set of (X,Y) data points, PLOT them along with error
; bars, if errors are passed, This procedure allows single-valued
; (non-array) X, Y, and errors.
;Input variables:
```

```
; X is the scalar or array of X-axis values
; Y is the corresponding scalar or array of Y-axis values
;Keyword parameters:
; YL is the scalar or array of lower bound errors in Y
; YU is the scalar or array of upper bound errors in Y
; XL is the scalar or array of lower bound errors in X
; XU is the scalar or array of upper bound errors in X
; Barsize is the normalized size of the bar end crosses, default = .005;
;  if 0 is input, then no end bars are plotted on the error bars
; XRange is the range along X to be plotted, with the default range
;  determined from the data values +/- error bars.
; YRange is the range along Y to be plotted, with the default range
;  determined from the data values +/- error bars.
;Keywords:
; /Relative_Errors specifies whether the errors are relative
;  (e.g. Y +/- 3) or absolute bounds (e.g. -2 < Y < 5)
; /LogX specifies if the X-axis values are to be plotted with a
;  logarithmic scale.
; /LogY specifies if the Y-axis values are to be plotted with a
;  logarithmic scale.
;Output variables:
; No values are modified or passed back from this routine.
;Calling sequence:
; EPlot, X, Y, YL=YL, YU=YU, XL=XL, XU=XU, /Relative_Errors, $
;  BarSize=BarSize, /LogX, /LogY
;-
; Return immediately in case of any errors.
;
On_Error, 2
If (Keyword_Set(Help)) then Begin
 Doc_Library, 'EPlot'
 Return
EndIf
;
; Save the input X and Y arrays
;
Xsave = X & Ysave = Y
;
; If a scalar point was passed, make it an array element.
;
If (N_elements(X) eq 1) then Begin
 X = [Xsave]
 Y = [Ysave]
EndIf
;
; Set a flag if the Relative_Errors keyword was specified.
;
Relative = Keyword_Set(Relative_Errors)
```

```
;
; If no X lower bounds were passed, make sure error bars won't be plotted.
;
If (N_elements(XL) eq 0) then Begin
 XL = X
 If (Relative) then XL(*) = 0
EndIf
;
; If no X upper bounds were passed, make sure error bars won't be plotted.
;
If (N_elements(XU) eq 0) then Begin
 XU = X
 If (Relative) then XU(*) = 0
EndIf
;
; If no Y lower bounds were passed, make sure error bars won't be plotted.
;
If (N_elements(YL) eq 0) then Begin
 YL = Y
 If (Relative) then YL(*) = 0
EndIf
;
; If no Y upper bounds were passed, make sure error bars won't be plotted.
;
If (N_elements(YU) eq 0) then Begin
 YU = Y
 If (Relative) then YU(*) = 0
EndIf
;
; Make the errors relative, regardless of how they were passed and use their
; absolute values.
;
If (Relative) then Begin
 XerrL = Abs(XL)
 XerrU = Abs(XU)
 YerrL = Abs(YL)
 YerrU = Abs(YU)
EndIf Else Begin
 XerrL = Abs(X - XL)
 XerrU = Abs(X - XU)
 YerrL = Abs(Y - YL)
 YerrU = Abs(Y - YU)
EndElse
;
; Only make X error bars where there are non-zero errors.
;
Xbars = Where((XerrL ne 0.) or (XerrU ne 0.))
If (Xbars(0) ne -1) then Begin
```

```
 NXbars = N_elements(Xbars) - 1
EndIf Else Begin
 NXBars = -1
EndElse
;
; Only make Y error bars where there are non-zero errors.
;
Ybars = Where((YerrU ne 0.) or (YerrL ne 0.))
If (Ybars(0) ne -1) then Begin
 NYbars = N_elements(Ybars) - 1
EndIf Else Begin
 NYBars = -1
EndElse
;
; Determine the plotting ranges in X and Y.
;
If (N_elements(XRange) eq 0) then XRange = [Min(X - XerrL), Max(X + XerrU)]
If (N_elements(YRange) eq 0) then YRange = [Min(Y - YerrL), Max(Y + YerrU)]
;
; Plot the input data values
;
Plot, X, Y, XRange = XRange, YRange = YRange, XType = Abs(Keyword_Set(LogX)), $
 YType = Abs(Keyword_Set(LogY))
;
; If error bar end cross length was not passed, set a default size.
;
If (N_elements(BarSize) eq 0) then Begin
 BarSize = .005
EndIf
If (NXbars gt -1) then Begin
;
; We've got some error bars in X.  Loop over non-zero.
;
 For I = 0, NXbars Do Begin
;
; Get the index of the next point with error bars.
;
  J = Xbars(I)
;
; Determine the upper and lower limit of the error bar along X.
;
  XlimL = X(J) - XerrL(J)
  XlimU = X(J) + XerrU(J)
;
; Plot the error bar.
;
  Oplot, [XlimL, XlimU], [Y(J), Y(J)], Linestyle = 0, Psym = 0
  If (BarSize gt 0) then Begin
```

```
;
; Add cross bars perpendicular to the end of the error bar.
;
   If (!Y.Type eq 0) then Begin
;
; Linear scale along Y.  Convert to normalized coordinates, and create the
; cross bar end points.
;
    YMidNormal = !Y.S(1)*Y(J) + !Y.S(0)
    YBarEnds = [YMidNormal + BarSize, $
     YMidNormal - BarSize]
;
; Convert back to data coordinates.
;
    YBarEnds = (YBarEnds - !Y.S(0))/!Y.S(1)
   EndIf Else Begin
;
; Logarthmic scale along Y.  Convert to normalized coordinates, and create the
; cross bar end points.
;
    YMidNormal = !Y.S(1)*Alog10(Y(J)) + !Y.S(0)
    YBarEnds = [YMidNormal + BarSize, $
     YMidNormal - BarSize]
;
; Convert back to data coordinates.
;
    YBarEnds = 10^((YBarEnds - !Y.S(0))/!Y.S(1))
   EndElse
;
; Plot the cross bars at the ends of the error bars.
;
   Oplot, [XlimL, XlimL], YBarEnds, Linestyle = 0, Psym = 0
   Oplot, [XlimU, XlimU], YBarEnds, Linestyle = 0, Psym = 0
  EndIf
 EndFor
EndIf
If (NYbars gt -1) then Begin
;
; We've got some error bars in Y.  Loop over non-zero.
;
 For I = 0, NYbars Do Begin
;
; Get the index of the next point with error bars.
;
  J = Ybars(I)
;
; Determine the upper and lower limit of the error bar along Y.
;
```

```
  YlimL = Y(J) - YerrL(J)
  YlimU = Y(J) + YerrU(J)
;
; Plot the error bar.
;
  Oplot, [X(J), X(J)], [YlimL, YlimU], Linestyle = 0, Psym = 0
  If (BarSize gt 0) then Begin
;
; Add cross bars perpendicular to the end of the error bar.
;
   If (!X.Type eq 0) then Begin
;
; Linear scale along X.  Convert to normalized coordinates, and create the
; cross bar end points.
;
    XMidNormal = !X.S(1)*X(J) + !X.S(0)
    XBarEnds = [XMidNormal + BarSize, $
     XMidNormal - BarSize]
;
; Convert back to data coordinates.
;
    XBarEnds = (XBarEnds - !X.S(0))/!X.S(1)
   EndIf Else Begin
;
; Logarthmic scale along X.  Convert to normalized coordinates, and create the
; cross bar end points.
;
    XMidNormal = !X.S(1)*Alog10(X(J)) + !X.S(0)
    XBarEnds = [XMidNormal + BarSize, $
     XMidNormal - BarSize]
;
; Convert back to data coordinates.
;
    XBarEnds = 10^((XBarEnds - !X.S(0))/!X.S(1))
   EndElse
;
; Plot the cross bars at the ends of the error bars.
;
   Oplot, XBarEnds, [YlimL, YlimL], Linestyle = 0, Psym = 0
   Oplot, XBarEnds, [YlimU, YlimU], Linestyle = 0, Psym = 0
  EndIf
 EndFor
EndIf
;
; Reset the X and Y values.
;
X = Xsave & Y = Ysave
Return
```

End

 == =
  Jon Saken....
 == =

From saken@stsci.edu Fri Dec 3 07:11:38 1993
Subject: Error plots
Status: OR

Hi,
 The ASTRON IDL library has a nice routine, PLOTERR, that does
just what you want. Unfortunately the IDL USERLIB also contains a
routine called PLOTERR which is not as useful, so you have to make
sure that the ASTRON directory appears first in your !PATH.

 == ===
  pcp2g
 == ===
From pcp2g@karma.astro.virginia.edu Fri Dec 3 06:59:45 1993

Hi! There is a large database of public IDL programs available
from the Goddard Space Flight Center archive, called MOOSE
routines. Here is the program for plotting error bars. I
don't use it, so I can;t explain it, but it will do both x
and y error bars.
=======================cut here==============================

```
PRO ploterr, x, y, xerr, yerr, NOHAT=hat, HATLENGTH=hln, ERRTHICK=eth, $
      ERRSTYLE=est, TYPE=itype, BACKGROUND=back, CHANNEL=chan,
CHARSIZE=chsiz, $
       CHARTHICK=chthck, COLOR=color, DATA=data, DEVICE=device, $
 FONT=font, LINESTYLE=linest, NOCLIP=noclip, NODATA=nodata, $
 NOERASE=noerase, NORMAL=normal, NSUM=nsum, PSYM=psym, $
 SUBTITLE=subtit, SYMSIZ=symsiz, T3D=t3d, THICK=thick, TICKLEN=ticklen,
$
 TITLE=title, XCHARSIZE=xchsiz, XMARGIN=xmargn, XMINOR=xminor, $
 XRANGE=xrange, XSTYLE=xstyle, XTICKLEN=xtickln, XTICKNAME=xticknm, $
 XTICKS=xticks, XTICKV=xtickv, XTITLE=xtitle, XTYPE=xtype, $
 YCHARSIZE=ychsiz, YMARGIN=ymargn, YMINOR=yminor, $
 YRANGE=yrange, YSTYLE=ystyle, YTICKLEN=ytickln, YTICKNAME=yticknm, $
 YTICKS=yticks, YTICKV=ytickv, YTITLE=ytitle, YTYPE=ytype
;+
; NAME:
; PLOTERR
```

```
; PURPOSE:
; Plot data points with accompanying X or Y error bars.
; CALLING SEQUENCE:
; ploterr, [ x,]  y, [xerr], yerr [, TYPE = type] [,/NOHAT,
HATLENGTH=hln,
;   ERRTHICK=eth, ERRSTYLE=est]
; INPUTS:
; X = array of abcissae.
; Y = array of Y values.
; XERR = array of error bar values (along X)
;       YERR = array of error bar values (along Y)
; OPTIONAL INPUT KEYWORD PARAMETERS:
; TYPE = type of plot produced.  The possible types are:
;  ITYPE = 0 : X Linear - Y Linear  (default)
;  ITYPE = 1 : X Linear - Y Log
;  ITYPE = 2 : X Log    - Y Linear
;  ITYPE = 3 : X Log    - Y Log
;       Actually, if 0 is specified, the XTYPE and YTYPE keywords
;       are used.  If these aren't specified, then a linear-linear
;       plot is produced.  This keyword is available to maintain
;       compatibility with the previous version of PLOTERR.
; NOHAT     = if specified and non-zero, the error bars are drawn
;           without hats.
; HATLENGTH = the length of the hat lines used to cap the error bars.
;           Defaults to !D.X_VSIZE / 100).
; ERRTHICK  = the thickness of the error bar lines.  Defaults to the
;           THICK plotting keyword.
; ERRSTYLE  = the line style to use when drawing the error bars.  Uses
;           the same codes as LINESTYLE.
;
; The keywords available to the PLOT procedure may also be specified
; (see Appendix D in the IDL User's Guide).  There are two that are
; specifically unavailable:  POSITION and CLIP.  To modify these, it
; is necessary to change the appropriate system variables.
; RESTRICTIONS:
; Arrays must not be of type string.  There must be enough points to
; plot.
;       If only three parameters are input, they will be taken as X, Y
and
;       YERR respectively.
; EXAMPLE:
;       Suppose one has X and Y vectors with associated errors XERR and
YERR
;       (1) Plot Y vs. X with both X and Y errors and no lines
connecting
;          the points
;                IDL> ploterr, x, y, xerr, yerr, psym=3
;       (2) Like (1) but plot only the Y errors bars and omits "hats"
```

```
;               IDL> ploterr, x, y, yerr, psym=3, /NOHAT
; PROCEDURE:
; A plot of X versus Y with error bars drawn from Y - YERR to Y + YERR
; and optionally from X - XERR to X + XERR is written to the output
device
; MODIFICATION HISTORY:
; William Thompson Applied Research Corporation  July, 1986
; DMS, April, 1989 Modified for Unix
; Michael R. Greason ST Systems
; May, 1991  Added most of the plotting keywords, put hats
;    on the error bars.
;       Wayne Landsman          Added call to PLOT_KEYWORDS   Jan, 1992
;       K. Venkatakrishna       Added option to plot xerr, May, 1992
;-
;   Check the parameters.
;
On_error, 2
np = N_params()
IF (np LT 2) THEN BEGIN
 print, "PLOTERR must be called with at least two parameters."
 print, "Syntax: ploterr, [x,] y, [xerr], yerr"
 RETURN
ENDIF
;
;   Interpret the keyword parameters.
plot_keywords, BACK=back,CHAN=chan,CHARSIZE=chsiz,CHARTHICK=chthck, $
  COLOR=color,DATA=data,DEVICE=device,FONT=font,LINESTYLE=line st, $
  NOCLIP=noclip,NODATA=nodata,NOERASE=noerase,NORMAL=normal,NS UM=nsum, $
 PSYM=psym,SUBTITLE=subtit,SYMSIZ=symsiz,T3D=t3d, $
  THICK=thick,TICKLEN=ticklen,TITLE=title,XCHARSIZE=xchsiz,XMA RGIN=xmargn
n, $
  XMINOR=xminor,XRANGE=xrange,XSTYLE=xstyle,XTICKLEN=xtickln,$
  XTICKNAME=xticknm,XTICKS=xticks,XTICKV=xtickv,XTITLE=xtitle, $
        XTYPE=xtype, YCHARSIZE=ychsiz,YMARGIN=ymargn, YMINOR=yminor, $
        YRANGE=yrange, YSTYLE=ystyle,YTICKLEN=ytickln,
YTICKNAME=yticknm, $
        YTICKS=yticks, YTICKV=ytickv,YTITLE=ytitle,YTYPE=ytype
;
;    Error bar keywords (except for HATLENGTH; this
;    one will be taken care of later, when it is
;    time to deal with the error bar hats).
;
IF (keyword_set(hat)) THEN hat = 0 ELSE hat = 1
IF (n_elements(eth) EQ 0) THEN eth = thick
IF (n_elements(est) EQ 0) THEN est = 0
;
;    Other keywords.
;
```

```
IF (keyword_set(itype)) THEN BEGIN
 CASE (itype) OF
    1 :  ytype = 1 ; X linear, Y log
    2 :  xtype = 1 ; X log, Y linear
    3 :  BEGIN  ; X log, Y log
   xtype = 1
   ytype = 1
   END
  ELSE :
 ENDCASE
ENDIF
;
;   If no x array has been supplied, create one.  Make
;   sure the rest of the procedure can know which parameter
;   is which.
;
IF np EQ 2 THEN BEGIN   ; Only Y and YERR passed.
 yerr = abs(y)
 yy = x
 xx = indgen(n_elements(yy))
       xerr = make_array(size=size(xx))

ENDIF ELSE IF np EQ 3 THEN BEGIN  ; X, Y, and YERR passed.
       yerr = abs(xerr)
       yy = y
       xx = x

ENDIF ELSE BEGIN                  ; X, Y, XERR and YERR passed.
 yerr = abs(yerr)
 yy = y
       xerr = abs(xerr)
 xx = x
ENDELSE
;
;   Determine the number of points being plotted.  This
;   is the size of the smallest of the three arrays
;   passed to the procedure.  Truncate any overlong arrays.
;

n = N_elements(xx) < N_elements(yy)

IF np GT 2 then n = n < N_elements(yerr)
IF np EQ 4 then n = n < N_elements(xerr)

IF n LT 2 THEN $
 message,'Not enough points to plot.'

xx = xx(0:n-1)
```

```
yy = yy(0:n-1)
yerr = yerr(0:n-1)
IF np EQ 4 then xerr = xerr(0:n-1)

;
;  If no y-range was passed via keyword or system variable, force one
;  large enough to display all the data and the entire error bars.
;
ylo = yy - yerr
yhi = yy + yerr
IF yrange(0) EQ yrange(1) THEN $
 yrange = [min(ylo), max(yhi)]
;
;       Similarly for x-range
;
if NP EQ 4 then begin
   xlo = xx - xerr
   xhi = xx + xerr
   IF xrange(0) EQ xrange(1) THEN xrange = [min(xlo), max(xhi)]
endif
;
;   Plot the positions.
;
plot, xx, yy, BACK=back,CHAN=chan,CHARSIZE=chsiz,CHARTHICK=chthck, $
  COLOR=color,DATA=data,DEVICE=device,FONT=font,LINESTYLE=line st, $
  NOCLIP=noclip,NODATA=nodata,NOERASE=noerase,NORMAL=normal,NS UM=nsum, $
 PSYM=psym,SUBTITLE=subtit,SYMSIZ=symsiz,T3D=t3d, $
  THICK=thick,TICKLEN=ticklen,TITLE=title,XCHARSIZE=xchsiz,XMA RGIN=xmargn
n, $
  XMINOR=xminor,XRANGE=xrange,XSTYLE=xstyle,XTICKLEN=xtickln,$
  XTICKNAME=xticknm,XTICKS=xticks,XTICKV=xtickv,XTITLE=xtitle, XTYPE=xtype
e, $
 YCHARSIZE=ychsiz,YMARGIN=ymargn, YMINOR=yminor,YRANGE=yrange, $
       YSTYLE=ystyle,YTICKLEN=ytickln, YTICKNAME=yticknm,YTICKS=yticks,
$
       YTICKV=ytickv,YTITLE=ytitle, YTYPE=ytype
;
; Plot the error bars.   Compute the hat length in device coordinates
;       so that it remains fixed even when doing logarithmic plots.
;
    data_low = convert_coord(xx,ylo,/TO_DEVICE)
    data_hi = convert_coord(xx,yhi,/TO_DEVICE)
    if NP EQ 4 then begin
    x_low = convert_coord(xlo,yy,/TO_DEVICE)
    x_hi = convert_coord(xhi,yy,/TO_DEVICE)
    endif
    ycrange = !Y.CRANGE   &  xcrange = !X.CRANGE
```

```
FOR i = 0, (n-1) DO BEGIN

    if (xtype EQ 0) then if $
         ( xx(i) LT xcrange(0) ) or ( xx(i) GT xcrange(1) ) then
goto,NOPLOT
    if (ytype EQ 0) then if $
          ( yy(i) LT ycrange(0) ) or ( yy(i) GT ycrange(1) ) then
goto,NOPLOT
 plots, [xx(i),xx(i)], [ylo(i),yhi(i)], LINESTYLE=est,THICK=eth
;                                              Plot X-error
bars
;
    if np EQ 4 then plots,
[xlo(i),xhi(i)],[yy(i),yy(i)],LINESTYLE=est,THICK=eth
 IF (hat NE 0) THEN BEGIN
  IF (N_elements(hln) EQ 0) THEN hln = !D.X_VSIZE/100.
  exx1 = data_low(0,i) - hln/2.
  exx2 = exx1 + hln
  plots, [exx1,exx2], [data_low(1,i),data_low(1,i)],  $
                    LINESTYLE=est,THICK=eth,/DEVICE
  plots, [exx1,exx2], [data_hi(1,i),data_hi(1,i)],  $
                     LINESTYLE=est,THICK=eth,/DEVICE
;
               IF np EQ 4 THEN BEGIN
                  IF (N_elements(hln) EQ 0) THEN hln = !D.Y_VSIZE/100.
                  eyy1 = x_low(1,i) - hln/2.
                  eyy2 = eyy1 + hln
                  plots, [x_low(0,i),x_low(0,i)], [eyy1,eyy2], $
                       LINESTYLE=est,THICK=eth,/DEVICE
                  plots, [x_hi(0,i),x_hi(0,i)], [eyy1,eyy2], $
                       LINESTYLE=est,THICK=eth,/DEVICE
               ENDIF
 ENDIF
    NOPLOT:
ENDFOR
;
RETURN
END

From pcp2g@karma.astro.virginia.edu Fri Dec  3 06:59:45 1993
Received: from virginia.edu (uvaarpa.Virginia.EDU) by spot.Colorado.EDU with SMTP id
AA22098
  (5.65c+/IDA-1.4.4/CNS-2.1 for <shah@spot.colorado.edu>); Fri, 3 Dec 1993 06:59:43 -0700
Received: from karma.astro.virginia.edu by uvaarpa.virginia.edu id aa29788;
          3 Dec 93 8:59 EST
Received: by karma.astro.Virginia.EDU (4.1/1.34)
 id AA18307; Fri, 3 Dec 93 08:59:39 EST
Date: Fri, 3 Dec 93 08:59:39 EST
```

From: Esplanade <pcp2g@karma.astro.virginia.edu>
Message-Id: <9312031359.AA18307@karma.astro.Virginia.EDU>
To: SSS <shah@spot.colorado.edu>
Subject: Re: Help, somebody! Error Bars in both X and Y directions.
Status: OR

Hi! There is a large database of public IDL programs available
from the Goddard Space Flight Center archive, called MOOSE
routines. Here is the program for plotting error bars. I
don't use it, so I can;t explain it, but it will do both x
and y error bars.

```
PRO ploterr, x, y, xerr, yerr, NOHAT=hat, HATLENGTH=hln, ERRTHICK=eth, $
      ERRSTYLE=est, TYPE=itype, BACKGROUND=back, CHANNEL=chan,
CHARSIZE=chsiz, $
       CHARTHICK=chthck, COLOR=color, DATA=data, DEVICE=device, $
 FONT=font, LINESTYLE=linest, NOCLIP=noclip, NODATA=nodata, $
 NOERASE=noerase, NORMAL=normal, NSUM=nsum, PSYM=psym, $
 SUBTITLE=subtit, SYMSIZ=symsiz, T3D=t3d, THICK=thick, TICKLEN=ticklen,
$
 TITLE=title, XCHARSIZE=xchsiz, XMARGIN=xmargn, XMINOR=xminor, $
 XRANGE=xrange, XSTYLE=xstyle, XTICKLEN=xtickln, XTICKNAME=xticknm, $
 XTICKS=xticks, XTICKV=xtickv, XTITLE=xtitle, XTYPE=xtype, $
 YCHARSIZE=ychsiz, YMARGIN=ymargn, YMINOR=yminor, $
 YRANGE=yrange, YSTYLE=ystyle, YTICKLEN=ytickln, YTICKNAME=yticknm, $
 YTICKS=yticks, YTICKV=ytickv, YTITLE=ytitle, YTYPE=ytype
;+
; NAME:
; PLOTERR
; PURPOSE:
; Plot data points with accompanying X or Y error bars.
; CALLING SEQUENCE:
; ploterr, [ x,]  y, [xerr], yerr [, TYPE = type] [,/NOHAT,
HATLENGTH=hln,
;   ERRTHICK=eth, ERRSTYLE=est]
; INPUTS:
; X = array of abcissae.
; Y = array of Y values.
; XERR = array of error bar values (along X)
;       YERR = array of error bar values (along Y)
; OPTIONAL INPUT KEYWORD PARAMETERS:
; TYPE = type of plot produced.  The possible types are:
;  ITYPE = 0 : X Linear - Y Linear  (default)
;  ITYPE = 1 : X Linear - Y Log
;  ITYPE = 2 : X Log    - Y Linear
;  ITYPE = 3 : X Log    - Y Log
;       Actually, if 0 is specified, the XTYPE and YTYPE keywords
```

```
;       are used.  If these aren't specified, then a linear-linear
;       plot is produced.  This keyword is available to maintain
;       compatibility with the previous version of PLOTERR.
; NOHAT     = if specified and non-zero, the error bars are drawn
;           without hats.
; HATLENGTH = the length of the hat lines used to cap the error bars.
;           Defaults to !D.X_VSIZE / 100).
; ERRTHICK  = the thickness of the error bar lines.  Defaults to the
;           THICK plotting keyword.
; ERRSTYLE  = the line style to use when drawing the error bars.  Uses
;           the same codes as LINESTYLE.
;
; The keywords available to the PLOT procedure may also be specified
; (see Appendix D in the IDL User's Guide).  There are two that are
; specifically unavailable:  POSITION and CLIP.  To modify these, it
; is necessary to change the appropriate system variables.
; RESTRICTIONS:
; Arrays must not be of type string.  There must be enough points to
; plot.
;       If only three parameters are input, they will be taken as X, Y
and
;       YERR respectively.
; EXAMPLE:
;       Suppose one has X and Y vectors with associated errors XERR and
YERR
;       (1) Plot Y vs. X with both X and Y errors and no lines
connecting
;          the points
;                IDL> ploterr, x, y, xerr, yerr, psym=3
;       (2) Like (1) but plot only the Y errors bars and omits "hats"
;                IDL> ploterr, x, y, yerr, psym=3, /NOHAT
; PROCEDURE:
; A plot of X versus Y with error bars drawn from Y - YERR to Y + YERR
; and optionally from X - XERR to X + XERR is written to the output
device
; MODIFICATION HISTORY:
; William Thompson Applied Research Corporation  July, 1986
; DMS, April, 1989 Modified for Unix
; Michael R. Greason ST Systems
; May, 1991  Added most of the plotting keywords, put hats
;    on the error bars.
;       Wayne Landsman         Added call to PLOT_KEYWORDS   Jan, 1992
;       K. Venkatakrishna      Added option to plot xerr, May, 1992
;-
;   Check the parameters.
;
On_error, 2
np = N_params()
```

```
IF (np LT 2) THEN BEGIN
 print, "PLOTERR must be called with at least two parameters."
 print, "Syntax: ploterr, [x,] y, [xerr], yerr"
 RETURN
ENDIF
;
;   Interpret the keyword parameters.
plot_keywords, BACK=back,CHAN=chan,CHARSIZE=chsiz,CHARTHICK=chthck, $
  COLOR=color,DATA=data,DEVICE=device,FONT=font,LINESTYLE=line st, $
  NOCLIP=noclip,NODATA=nodata,NOERASE=noerase,NORMAL=normal,NS UM=nsum, $
 PSYM=psym,SUBTITLE=subtit,SYMSIZ=symsiz,T3D=t3d, $
  THICK=thick,TICKLEN=ticklen,TITLE=title,XCHARSIZE=xchsiz,XMA RGIN=xmargn
n, $
  XMINOR=xminor,XRANGE=xrange,XSTYLE=xstyle,XTICKLEN=xtickln,$
  XTICKNAME=xticknm,XTICKS=xticks,XTICKV=xtickv,XTITLE=xtitle, $
        XTYPE=xtype, YCHARSIZE=ychsiz,YMARGIN=ymargn, YMINOR=yminor, $
        YRANGE=yrange, YSTYLE=ystyle,YTICKLEN=ytickln,
YTICKNAME=yticknm, $
        YTICKS=yticks, YTICKV=ytickv,YTITLE=ytitle,YTYPE=ytype
;
;    Error bar keywords (except for HATLENGTH; this
;    one will be taken care of later, when it is
;    time to deal with the error bar hats).
;
IF (keyword_set(hat)) THEN hat = 0 ELSE hat = 1
IF (n_elements(eth) EQ 0) THEN eth = thick
IF (n_elements(est) EQ 0) THEN est = 0
;
;    Other keywords.
;
IF (keyword_set(itype)) THEN BEGIN
 CASE (itype) OF
     1 :  ytype = 1 ; X linear, Y log
     2 :  xtype = 1 ; X log, Y linear
     3 :  BEGIN  ; X log, Y log
   xtype = 1
   ytype = 1
   END
  ELSE :
 ENDCASE
ENDIF
;
;   If no x array has been supplied, create one.  Make
;   sure the rest of the procedure can know which parameter
;   is which.
;
IF np EQ 2 THEN BEGIN   ; Only Y and YERR passed.
 yerr = abs(y)
```

```
 yy = x
 xx = indgen(n_elements(yy))
       xerr = make_array(size=size(xx))

ENDIF ELSE IF np EQ 3 THEN BEGIN  ; X, Y, and YERR passed.
       yerr = abs(xerr)
       yy = y
       xx = x

ENDIF ELSE BEGIN                  ; X, Y, XERR and YERR passed.
 yerr = abs(yerr)
 yy = y
       xerr = abs(xerr)
 xx = x
ENDELSE
;
;   Determine the number of points being plotted.  This
;   is the size of the smallest of the three arrays
;   passed to the procedure.  Truncate any overlong arrays.
;

n = N_elements(xx) < N_elements(yy)

IF np GT 2 then n = n < N_elements(yerr)
IF np EQ 4 then n = n < N_elements(xerr)

IF n LT 2 THEN $
 message,'Not enough points to plot.'

xx = xx(0:n-1)
yy = yy(0:n-1)
yerr = yerr(0:n-1)
IF np EQ 4 then xerr = xerr(0:n-1)

;
;  If no y-range was passed via keyword or system variable, force one
;  large enough to display all the data and the entire error bars.
;
ylo = yy - yerr
yhi = yy + yerr
IF yrange(0) EQ yrange(1) THEN $
 yrange = [min(ylo), max(yhi)]
;
;       Similarly for x-range
;
if NP EQ 4 then begin
   xlo = xx - xerr
   xhi = xx + xerr
```

```
   IF xrange(0) EQ xrange(1) THEN xrange = [min(xlo), max(xhi)]
endif
;
;   Plot the positions.
;
plot, xx, yy, BACK=back,CHAN=chan,CHARSIZE=chsiz,CHARTHICK=chthck, $
  COLOR=color,DATA=data,DEVICE=device,FONT=font,LINESTYLE=line st, $
  NOCLIP=noclip,NODATA=nodata,NOERASE=noerase,NORMAL=normal,NS UM=nsum, $
 PSYM=psym,SUBTITLE=subtit,SYMSIZ=symsiz,T3D=t3d, $
  THICK=thick,TICKLEN=ticklen,TITLE=title,XCHARSIZE=xchsiz,XMA RGIN=xmargn
n, $
  XMINOR=xminor,XRANGE=xrange,XSTYLE=xstyle,XTICKLEN=xtickln,$
  XTICKNAME=xticknm,XTICKS=xticks,XTICKV=xtickv,XTITLE=xtitle, XTYPE=xtype
e, $
 YCHARSIZE=ychsiz,YMARGIN=ymargn, YMINOR=yminor,YRANGE=yrange, $
       YSTYLE=ystyle,YTICKLEN=ytickln, YTICKNAME=yticknm,YTICKS=yticks,
$
       YTICKV=ytickv,YTITLE=ytitle, YTYPE=ytype
;
; Plot the error bars.   Compute the hat length in device coordinates
;       so that it remains fixed even when doing logarithmic plots.
;
    data_low = convert_coord(xx,ylo,/TO_DEVICE)
    data_hi = convert_coord(xx,yhi,/TO_DEVICE)
    if NP EQ 4 then begin
    x_low = convert_coord(xlo,yy,/TO_DEVICE)
    x_hi = convert_coord(xhi,yy,/TO_DEVICE)
    endif
    ycrange = !Y.CRANGE   &  xcrange = !X.CRANGE

FOR i = 0, (n-1) DO BEGIN

    if (xtype EQ 0) then if $
         ( xx(i) LT xcrange(0) ) or ( xx(i) GT xcrange(1) ) then
goto,NOPLOT
    if (ytype EQ 0) then if $
          ( yy(i) LT ycrange(0) ) or ( yy(i) GT ycrange(1) ) then
goto,NOPLOT
 plots, [xx(i),xx(i)], [ylo(i),yhi(i)], LINESTYLE=est,THICK=eth
;                                              Plot X-error
bars
;
    if np EQ 4 then plots,
[xlo(i),xhi(i)],[yy(i),yy(i)],LINESTYLE=est,THICK=eth
 IF (hat NE 0) THEN BEGIN
  IF (N_elements(hln) EQ 0) THEN hln = !D.X_VSIZE/100.
  exx1 = data_low(0,i) - hln/2.
  exx2 = exx1 + hln
```

```
  plots, [exx1,exx2], [data_low(1,i),data_low(1,i)],  $
                  LINESTYLE=est,THICK=eth,/DEVICE
  plots, [exx1,exx2], [data_hi(1,i),data_hi(1,i)],  $
                   LINESTYLE=est,THICK=eth,/DEVICE
;
              IF np EQ 4 THEN BEGIN
                IF (N_elements(hln) EQ 0) THEN hln = !D.Y_VSIZE/100.
                eyy1 = x_low(1,i) - hln/2.
                eyy2 = eyy1 + hln
                plots, [x_low(0,i),x_low(0,i)], [eyy1,eyy2], $
                     LINESTYLE=est,THICK=eth,/DEVICE
                plots, [x_hi(0,i),x_hi(0,i)], [eyy1,eyy2], $
                     LINESTYLE=est,THICK=eth,/DEVICE
              ENDIF
 ENDIF
   NOPLOT:
ENDFOR
;
RETURN
--
=========================== Department of Aerospace Engineering Sciences
BioServe Space Technologies      University of Colorado at Boulder
===========================        E-Mail: shah@spot.colorado.edu
```