URL: http://cxc.harvard.edu/ciao3.4/curve.html
Last modified: December 2006

AHELP for CIAO 3.4

curve

Context: chips

Jump to: Description Examples ARGUMENT LIST Bugs See Also

Synopsis

Plots a curve or change its attributes (PLOT is equivalent).

Syntax

```
chips> CURVE <filename> # # #
chips> PLOT <filename> # # #

chips> CURVE <filename> [X #] [Y #] \
[{XERR # | XUP # | XDOWN # | XDN #}] \
[{E # | YERR # | YUP # | YDOWN # | YDN #}] \
[SCOL #]
chips> PLOT <filename> [X #] [Y #] \
[{XERR # | XUP # | XDOWN # | XDN #}] \
[{E # | YERR # | YUP # | YDOWN # | YDN #}] \
[SCOL #]

() = curve( x, y );
() = curve( x, y, yerr );
() = curve( x, y, yerrlo, yerrhi );
() = curve( x, y, xerrlo, xerrhi, yerrlo, yerrhi );

When used to plot data from a file CURVE and PLOT are synonyms but
curve() is the only valid S-Lang function name.
```

Description

Plotting data from a file

The CURVE command is used to plot data from either an ASCII data file or a FITS table file.

If you are only plotting x against y values − i.e. no errors − then you just need to specify the columns using the Data Model column filtering syntax (see "ahelp dmcols"). So

```
  chips> curve "lc.fits[cols time,rate]"
```

plots the rate column (Y axis) against the time column (X axis). For an ASCII file this would just be

```
chips> curve lc.dat
```

To include error bars you need to specify − after the filename − which columns contain the error information. If only symmetric errors on the Y axis are required then the simplest way is to say

```
chips> curve lc.fits 1 3 5
```

− which means that the X and Y columns are taken from the first and third column of lc.fits and the error on the Y value is taken from the fifth column. This can also be written as

```
  chips> curve lc.fits x 1 y 3 e 5
```

where we explicitly include the type for each column. If the X and Y columns are the first two columns in the file then this can be shortened to

```
chips> curve lc.fits e 5
```

(yerr can be used instead of e).

For non−symmetric Y errors, use the YUP and YDOWN (or YDN) labels to give the the number of the columns containing the low and high error values. The XERR, XUP, and XDOWN (or XDN) labels indicate the numbers of the columns which contain errors for the X axis.

The SCOL label is used to indicate the column from which the symbol type for each element is found. This column must be an integer and the values are discussed below.

Plotting data from S−Lang

The S−Lang function "curve()" can be used to plot two arrays against each other. So,

```
  variable x = [0:10:0.01];
  () = curve( x, sin(x) );
```

draws sin(x) for all points between 0 and 10 with a step−size of 0.01. As shown in the SYNTAX section above, the curve() command can also plot Y errors (symmetric or non−symmetric) and for the X axis too (although if X errors are given then the YERRLO, YERRHI, XERRLO, and XERRHI values must be given). So,

```
  variable x = [1:20];
  () = curve( x, x+2, sqrt(x+2) );
```

plots up a set of points with symmetric Y errors.

The curve() functions returns 0 on success, −1 on error. In the examples above we use "() =" to ignore this return value.

Changing how the plot looks

Whether a curve has been created from a file using the ChIPS CURVE command, or created from S−Lang using the curve() function, the same set of commands can be used to change its attributes.

Firstly, all curves are created with their attributes set to the values given in the ChIPS state object. Therefore, setting

```
  chips.symbolstyle = _chips->square;
```

(whether in the ChIPS resource file or at the ChIPS prompt) will cause any new curves to be drawn using an open square for the sybol. Note that this will not change the attributes of curves that have already been drawn.

Secondly, the CURVE command can be used to change the attributes of a curve that has already been drawn. Each of the following commands can be preceeded by one − or both − of [D #] and [C #] to select the curve you wish to change. See "ahelp d" and "ahelp c" for more information on these two commands.

```
chips> [{CURVE | PLOT}] <color>
chips> [{CURVE | PLOT}] <curvestyle>
chips> [{CURVE | PLOT}] <linestyle>
chips> [{CURVE | PLOT}] SCALE [<coor>] <scale_value>
chips> [{CURVE | PLOT}] WIDTH <width_value>
chips> ERRS [<coor>] <color>
chips> ERRS [<coor>] <errorstyle>
chips> ERRS [<coor>] <errortype>
chips> ERRS [<coor>] <linestyle>
chips> ERRS [<coor>] {ABSOLUTE | RELATIVE}
chips> ERRS [<coor>] SIZE <errortick_size_value>
chips> SYMBOL <color>
chips> SYMBOL <symbolstyle>
chips> SYMBOL [SIZE] <symbol_size_value>
```

A description of the various arguments used in these calls can be found in the "ARGUMENT LIST" section at the end of this document.

Plotting Multiple Curves

Multiple curves may be plotted within the same drawing area. The limits of the drawing area are affected by the {CURVE | PLOT} command as follows:

- When the first curve is added, the limits of the drawing area are automatically sized to fit.
- When each subsequent curve is added, ChIPS automatically changes the limits of the drawing area. If the range and domain of the new curve overlaps with the range and domain of the current drawing area, the drawing area is resized, using as new limits the smallest minimum and the largest maximum values. If either the range or the domain of the new curve do not overlap with the range or domain of the current drawing area, the limits are set to display the new curve, and a warning message is given to indicate that the drawing area limits were changed significantly.

This automatic calculation of the plot limits can be stopped by using the LIMITS command to set the axis ranges to fixed values.

ChIPS converts double−precision numbers to floating−point values before plotting them, which can cause problems for values that are either too large or too small. The allowed range is approximately 1e−38 to 3e38 (for both positive and negative values).

Interrupting a Plot

Note that ChIPS does not recognize "CTRL−C" as an interrupt command. There is no way to cancel the drawing of a long plot once it has begun.

Example 1

```
chips> CURVE "lc.fits[cols time,rate]"
```

Here we plot the rate versus time columns from the FITS file "lc.fits". The Data model column filter "[cols time,rate]" is used to select the X and Y columns for the plot (since we do not explicitly set them they default to the first two columns in the file). The following commands are equivalent:

```
chips> D 1 CURVE "lc.fits[cols time,rate]"
chips> CURVE "lc.fits[cols time,rate]" X 1 Y 2
```

Example 2

```
chips> CLEAR
chips> CURVE "lc.fits[cols time,rate]"
chips> SYMBOL SQUARE
```

First we remove any existing data from ChIPS with the CLEAR command, otherwise the new data would be overplotted on any existing data. The data is ploted as in the previous example except that we then change the symbol to be a square.

Example 3

```
chips> CLEAR
chips> CURVE "lc.fits[cols time,rate]"
chips> SYMBOL SQUARE
chips> SYMBOL RED
```

This changes the symbol type to a red square.

Example 4

```
chips> CLEAR
chips> CURVE "lc.fits[cols time,rate]"
chips> SIMPLELINE
chips> SYMBOL NONE
```

The default behaviour for CURVE is to draw symbols at each point. Here we change it to drawing lines between each point and then remove the symbols.

Example 5

```
chips> CLEAR
chips> CURVE "lc.fits[cols time,rate]"
chips> SIMPLELINE
chips> RED
```

Here we draw a red line between the points but leave the symbol. Note that the symbol color has not been changed to red: this would be done by saying

```
  chips> SYMBOL RED
```

Example 6

```
chips> CLEAR
chips> CURVE "lc.fits[cols time,rate]"
chips> STEP
chips> RED
chips> SYMBOL YELLOW
```

There are two ways that points can be connected by lines: SIMPLELINE draws lines between the points whereas STEP (or its alias HIST) considers the data points to represent a histogram (the X values give the center of each bin).

The line color is changed to red and the symbol color is changed to yellow.

Example 7

```
chips> REDRAW OFF
chips> CLEAR
chips> CURVE "lc.fits[cols time,rate]"
chips> STEP
chips> RED
chips> SYMBOL YELLOW
chips> REDRAW ON
```

This has the same result as the previous example except that the plot is not changed until the REDRAW ON command is called. See "ahelp redraw" for more information.

Example 8

```
chips> CLEAR
chips> chips.curvestyle = _chips->step
chips> chips.curvecolor = _chips->red
chips> chips.symbolcolor = _chips->yellow
chips> CURVE "lc.fits[cols time,rate]"
```

The previous examples all used the CURVE command to change the attributes of an existing plot. The ChIPS state object (see "ahelp chips") provides a way of setting the attributes for any new plots. With the above commands the result is the same plot as previous (a set of points connected as if a histogram with a red line and yellow symbols). The difference is that if another curve is plotted it will also be drawn with the same settings.

Example 9

```
chips> chips_clear
chips> lc = readfile("lc.fits")
chips> () = curve(lc.time,lc.rate)
chips> STEP
chips> RED
chips> SYMBOL YELLOW
```

Here we have read in the contents of "lc.fits" using the Varmm function readfile (see "ahelp readfile") and have used the S−Lang function curve() to plot the time and rate columns of the file. Once the curve has been created it can be manipulated as any other curve.

Example 10

```
chips> chips_clear
chips> chips.curvestyle = _chips->step
chips> chips.curvecolor = _chips->red
chips> chips.symbolcolor = _chips->yellow
chips> lc = readfile("lc.fits")
chips> () = curve(lc.time,lc.rate)
```

The curve() command also follows the settings of the ChIPS state object when creating the plot.

Example 11

```
chips> CURVE data/exampleB.dat
```

Plots the ASCII data file data/exampleB.dat using the default settings, where column 1 is Xx and column 2 is y. The following commands are equivalent:

```
chips> D 1 CURVE data/exampleB.dat
chips> CURVE data/exampleB.dat X 1 Y 2
```

Example 12

```
chips> CURVE "data/exampleB.fits[cols col1,col2]"
```

The <virtual_file_syntax> argument is used to plot columns col1 and col2 from a FITS file. Again, the default setting are used, i.e. the first column listed in the DM filter (col1) is x and the second (col2) is y. The following commands are equivalent:

```
chips> D 1 CURVE "data/exampleB.fits[cols col1,col2]"
chips> CURVE "data/exampleB.fits[cols col1,col2]" X 1 Y 2
```

Example 13

```
chips> CURVE "data/exampleB.fits[cols col1,col2,col6]" 1 2 3
```

Columns named col1, col2, and col6 are plotted from the FITS data file data/exampleB.fits. The first column listed in the DM filter is x, the second is y, and the third is the symmetric error bar in y. The following commands are equivalent:

```
chips> D 1 CURVE "data/exampleB.fits[cols col1,col2,col6]" 1 2 3
chips> CURVE "data/exampleB.fits[cols col1,col2,col6]" X 1 Y 2 E 3
chips> CURVE "data/exampleB.fits[cols col1,col2,col6]" X 1 Y 2 YERR 3
```

Example 14

```
chips> CURVE data/exampleB.dat X 1 Y 2 YUP 7 YDOWN 8
```

The ASCII data file data/exampleB.dat is plotted, with column 1 as x, column 2 as y, column 7 as the positive error in y, and column 8 as the negative error in y. The following commands are equivalent:

```
chips> D 1 CURVE data/exampleB.dat X 1 Y 2 YUP 7 YDOWN 8
chips> CURVE data/exampleB.dat YUP 7 YDOWN 8
```

Example 15

```
chips> CURVE "data.fits[cols col1, col2, col7, col8]" X 1 Y 2
       YUP 3 YDOWN 4
```

The same as the previous example, but the data is read from a FITS file. The following commands are equivalent:

```
chips> D 1 CURVE "data.fits[cols col1, col2, col7, col8]" X 1 Y 2
       YUP 3 YDOWN 4
chips> CURVE "data.fits[cols col1, col2, col7, col8]" YUP 3 YDOWN 4
```

Example 16

```
chips> CURVE data/exampleB.dat X 1 Y 2 XUP 4 XDOWN 5 YERR 6
```

This command plots x, y, the positive and negative errors in x, and the symmetric error in y. The following commands are equivalent:

```
chips> D 1 CURVE data/exampleB.dat X 1 Y 2 XUP 4 XDOWN 5 YERR 6
chips> CURVE data/exampleB.dat XUP 4 XDOWN 5 YERR 6
```

Example 17

```
chips> CURVE data/exampleA.dat
chips> CURVE data/exampleB.dat
```

Two curves are plotted and numbered 1 and 2, respectively.

Example 18

```
chips> C 1 DEL
You have no more curves in this drawing area.
chips> CURVE data/exampleB.dat 1 2 6
```

An existing curve is deleted before the ASCII data is plotted; column 1 is x, column 2 is y, column 6 is the symmetric error bar in Y. The following commands are equivalent:

```
chips> D 1 CURVE data/exampleB.dat 1 2 6
chips> CURVE data/exampleB.dat X 1 Y 2 E 6
chips> CURVE data/exampleB.dat X 1 Y 2 YERR 6
```

Example 19

```
chips> CURVE data/exampleB.dat
chips> D 1 C 1 CURVE HISTO
```

A single curve is plotted and the curve style is changed to histogram. The following commands are equivalent:

```
chips> C 1 CURVE HISTO
chips> CURVE HISTO
chips> PLOT HISTO
chips> HISTO
```

Example 20

```
chips> LONGDASH
chips> WIDTH 4.0
chips> RED
chips> SYMBOL UPTRI
chips> SYMBOL BLUE
chips> SYMBOL SIZE 5.0
```

Building on the previous example, various attributes to the curve are modified. The histogram line is changed from to a wide (width 4) line of long dashes. RED changes the color of the curve line, and the three SYMBOL commands change the type, color, and size of the plot symbols.

Example 21

```
chips> C 1 DEL
You have no more curves in this drawing area.
chips> CURVE data/exampleB.dat 1 2 6
chips> ERRS BAR
```

All existing curves are removed, then data with symmetric Y−axis error bars is plotted. The final command modifies these error bars such that they do not have ticks on the top or bottom of the bars.

Example 22

```
chips> CURVE data/exampleB.dat X 1 Y 2 XUP 4 XDOWN 5 YUP 7 YDOWN 8
chips> ERRS X UP
chips> ERRS X BAR
```

Data is plotted with asymmetric x− and y−axis error bars. The command ERRS X UP causes only the positive x errorbars to be plotted. Then the errors are changed so that they do not have ticks on the top or bottom of the bars.

Example 23

```
chips> CURVE data/exampleA.dat
chips> CURVE DATA data/exampleA.dat YUP 5 YDOWN 6
chips> CURVE data/exampleB.dat
chips> CLEAR
```

The first CURVE command adds curve number 1 to the drawing area, from file data/exampleA.dat. The CURVE DATA command alters the curve so that it includes error bars from the same file. The third command adds curve number 2 to the drawing area, from file data/exampleB.dat. All plotting objects are then deleted, creating a new blank drawing area.

Example 24

```
chips> CURVE data/exampleB.dat X 1 Y 2 YERR 6
chips> D 1 CURVE DATA data/exampleB.dat X 1 Y 2 YUP 7 YDOWN 8
```

X, y, and symmetric y errors are plotted from an ASCII data file. The curve is then altered to have separate positive and negative y errors plotted.

ARGUMENT LIST

The CURVE command (the ChIPS command not the S−Lang function) can take many arguments:

```
 chips> [D #] [C #] {CURVE | PLOT} DATA <filename> \
       [X #] [Y #]} [{XERR # | XUP # | XDOWN # | XDN #} \
       [{E # | YERR # | YUP # | YDOWN # | YDN #} [SCOL #]
```

The filename can include the path to the file (either relative or absolute) and − if a FITS file − any virtual file syntax as described in "ahelp dmsyntax". If a Data Model filter is used (i.e. something within "[]") then the whole file name has to be enclosed in "" characters.

```
       Argument: ABSOLUTE or ABS
       Description: errors are not relative to the data value,
                    they are plotted as absolute values
       Default: RELATIVE
```

```
       Argument: <color>
       Description: Color of the axes
       Options: BLACK, BLUE, CYAN, DEFAULT, GREEN, MAGENTA, RED, WHITE, YELLOW
       Default: DEFAULT (appears in ChIPS window as white; prints as black)
```

```
       Argument: <coor>
       Description: axis coordinates
       Options: X, Y
       Default: both X and Y
```

```
Argument: <curvestyle>
Description: style of curve
Options: NOLINE − no connecting line between data points,
         SIMPLELINE − line connecting data points,
         STEP − stepped connection between data points (alias for HISTO)
Default: NOLINE
```

```
Argument: E #
Options: column number for the symmetric error on y
Default: integer numbers
```

```
Argument: <errorstyle>
Description: style for errorbars
Options: STANDARD − errorbar with top and bottom ticks,
         BAR − errorbar only (no ticks on top or bottom)
Default: STANDARD
```

```
Argument: <errortick_size_value>
Description: size of top and bottom errorbar ticks
Options: real numbers
Default: 3.0
```

```
Argument: <errortype>
Description: type of errorbars
Options: BOTH − both positive and negative errorbars,
         NONE − no error bars,
         UP − positive errorbar only,
         {DOWN | DN} − negative errorbar only
Default: BOTH
```

```
Argument: <linestyle>
Description: style for lines
Options: SOLID, DOT, DASH, LONGDASH, DOTDASH, DOTLONGDASH, DASHLONGDASH
Default: SOLID
```

```
Argument: {RELATIVE | REL}
Options: errors are relative to the data value
Default: RELATIVE
```

```
Argument: <scale_value>
Description: plot scaling factor
Options: real numbers
Default: 1.0
```

```
Argument: SCOL #
Options: column number for the symbol specification
Default: integer numbers (see below for list)
```

```
Argument: <symbol_size_value>
Description: size of symbols factor
Options: real numbers
Default: 3.0
```

```
Argument: <symbolstyle>
Description: style for symbols
Options: BLOCK, CIRCLE, CROSS, DIAMOND, SOLIDDIAMOND, DOWNTRI,
         SOLIDDOWNTRI, UPTRI, SOLIDUPTRI, SQUARE, POINT, BIGPOINT,
         NONE
Default: CROSS
```

```
Argument: <width_value>
Description: curve thickness
Options: real numbers
Default: 1.0
```

```
Argument: X #
Description: column number for the X axis
Options: integer numbers
Default: X 1
```

```
Argument: XERR #
Description: column number for the symmetric error on x
Options: integer numbers
```

```
Argument: XUP #
Description: column number for the positive error on x
Default: integer numbers
```

```
Argument: {XDOWN | XDN} #
Description: column number for the negative error on x
Options: integer numbers
```

```
Argument: Y #
Description: column number for the Y axis
Options: integer numbers
Default: Y 2
```

```
Argument: YERR #
Description: column number for the symmetric error on y
Options: integer numbers
```

```
Argument: YUP #
Description: column number for the positive error on y
Options: integer numbers
```

```
Argument: {YDOWN | YDN} #
Description: column number for the negative error on y
Options: integer numbers
```

SCOL values

The following table lists the possible values for the SCOL column.

Value	Symbol type
1 − 19	dot
20 − 29	horizontal dash
30	open triangle
31	inverted Y
32	starred triangle
33	solid triangle
40	open square
41	X
42	starred square
43	solid square

AB	The two−digit number AB means that the symbol (a polygon) has A sides and that these sides should be connected according the the value of B: 0=open, 1=skeletal, 2=starred, and 3=solid. No symbol will be produced if B is greater than 3.

Bugs

See the bugs page for ChIPS on the CIAO website for an up−to−date listing of known bugs.

See Also

chips

contour, display, surface, viewpoint

URL: http://cxc.harvard.edu/ciao3.4/curve.html
Last modified: December 2006