Pg — A Tcl pgplot extension

Version 0.5 3 April 1995

Paul Alexander, Cavendish Laboratory, Cambridge, UK

1 Baisc PG commands

1.1 arrow – draw an arrow

```
pg arrow x1 y1 x2 y2 ?standard options?

Arguments:

  X1,Y1       : world coordinates of arrow tail
  X2,Y2       : world coordinates of arrow head
```

Draw an arrow from the point with world-coordinates (X1,Y1) to (X2,Y2). The size of the arrowhead at (X2,Y2) is determined by the current character size. The default size is 1/40th of the smaller of the width or height of the view surface. The appearance of the arrowhead (shape and solid or open) is controlled by following parameters:

```
arrow size line colour
```

See also: style, standard options

1.2 buffer – buffer output

```
pg buffer boolean
pg bbuf
pg ebuf

Arguments:

  None

pg buffer 1 and pg bbuf initiate buffering
pg buffer 0 and pg ebuf end buffering
```

Graphical output commands in an internal buffer when buffering is enabled and flushed to the output from buffering is ended or the buffer is emptied with "pg update". Buffering can greatly improve the efficiency of plotting.

See also: update

Example:

```
  # start buffering
  pg buffer 1

  # draw some graphics
    pg line 0 0 1 1 -colour 2
    pg line 1 0 0 1 -colour 3

  # end buffering and flush
    pg buffer 0
```

1.3 box – Annotate the viewport with frame

```
pg box ?box options? ?standard options?
```

```
Arguments:

  None
```

Annotate the viewport with frame, axes, numeric labels, etc.

Box options:

```
 -xopts           : string of options for X (horizontal) axis of
                    plot. Options are single letters, and may be in
                    any order (see below).
 -xticks          : world coordinate interval between major tick marks
                    on X axis. If xtick=0.0, the interval is chosen by
                    PGBOX, so that there will be at least 3 major tick
                    marks along the axis.
 -xdivisions      : the number of subintervals to divide the major
                    coordinate interval into. If xtick=0.0 or xdivisions=0,
                    the number is chosen by automatically.
 -yopts           :
 -yticks          : as above but for the Y-axis
 -ydivisions      :
 -radec           : label box with RA/DEC axes instead of default.
```

Options (for parameters xopt and yopt):

```
 A : draw Axis (X axis is horizontal line Y=0, Y axis is vertical
     line X=0).
 B : draw bottom (X) or left (Y) edge of frame.
 C : draw top (X) or right (Y) edge of frame.
 G : draw Grid of vertical (X) or horizontal (Y) lines.
 I : Invert the tick marks; ie draw them outside the viewport
     instead of inside.
 L : label axis Logarithmically (see below).
 N : write Numeric labels in the conventional location below the
     viewport (X) or to the left of the viewport (Y).
 P : extend ("Project") major tick marks outside the box (ignored if
     option I is specified).
 M : write numeric labels in the unconventional location above the
     viewport (X) or to the right of the viewport (Y).
 T : draw major Tick marks at the major coordinate interval.
 S : draw minor tick marks (Subticks).
 V : orient numeric labels Vertically. This is only applicable to Y.
     The default is to write Y-labels parallel to the axis
```

To get a complete frame, specify BC in both xopt and yopt. Tick marks, if requested, are drawn on the axes or frame or both, depending which are requested. If none of ABC is specified, tick marks will not be drawn.

For a logarithmic axis, the major tick interval is always 1.0. The numeric label is 10**(x) where x is the world coordinate at the tick mark. If subticks are requested, 8 subticks are drawn between each major tick at equal logarithmic intervals.

See also: standard options

1.4 circle – draw a filled or outline circle

```
pg circle xcent ycent radius ?standard options?
```

```
Arguments:
```

```
 XCENT    : world x-coordinate of the center of the circle.
 YCENT    : world y-coordinate of the center of the circle.
 RADIUS   : radius of circle (world coordinates).
```

Draw a circle. The action of this routine depends on the setting of the Fill-Area Style attribute. If Fill-Area Style is SOLID (the default), the interior of the circle is solid-filled using the current Color Index. If Fill-Area Style is HOLLOW, the outline of the circle is drawn using the current line attributes (color index, line-style, and line-width).

1.5 draw – draw a line from the current pen position to a point

```
pg draw x y ?standard options?

Arguments:
  X      : world x-coordinate of the end point of the line.
  Y      : world y-coordinate of the end point of the line.
```

Draw a line from the current pen position to the point with world-coordinates (X,Y). The line is clipped at the edge of the current window. The new pen position is (X,Y) in world coordinates.

1.6 line – draw a line joining two points

```
pg line x1 y1 x2 y2 ?standard options?

Arguments:
  X1,Y1      : world coordinates of the start point of the line.
  X2,Y2      : world coordinates of the end point of the line.

pg draw x y ?standard options?
```

Draw a line from the current pen position to the point with world-coordinates (X,Y). The line is clipped at the edge of the current window. The new pen position is (X,Y) in world coordinates.

1.7 enquire – enquiry function

```
pg enquire option

Arguments

   option     :  line fill width font colour color
                 size top window viewport vport range cursor
```

Enquire information from the plotting system. The information is returned in the form of a standard Tcl list.

```
  line           return current line-style index
  fill           return current fill style (0/1)
  width          return current line width
  font           return current font number
  colour         return current colour index (ci)
  color          return current colour index (ci)
  size           return current chatracter size
  top            return current error-bar "top" size
  window         return a list giving the current window {wx1 wx2 wy1 wy2}
  viewport unit} unit is the units of the required information:
  vport    unit}    0     normalized device coordinates
                    1     inches
                    2     millimeters
                    3     pixels
                 return a list giving the current viewport {vx1 vx2 vy1 vy2}
  range x1 x2    return a suggested range {rx1 rx2} for data running from
                 x1 to x2 inclusive
  cursor ?x y?  return the position of the graphics cursor as selected
                 by the user; if given, (x, y) give the initial display
                 position of the cursor; returned is {x y ch}, the cursor
                 coordinates and character typed by the user.
```

1.8 error – error bar

```
pg error p e1 e2 ?-x? ?-y? ?standard options?

Arguments:

  p           :   coordinate of point
  e1, e2      :   limits of the error bar.
```

Plot an error bars. This command draws an error bar only; to mark the data point in the middle of the error bar, an additional call to pg point is needed. If ”-x” is specified then the error bar is drawn in the X direction, if ”-y” is specified then in the Y diection (default is the Y direction). The error bar is drawn at point p from e1 to e2; for example the following will draw an error bar centred at 0.5,0.6 with an error of 0.05 in Y:

```
  pg error 0.5 0.55 0.65 -y
```

1.9 move – move current drawing position

```
pg move x y

Arguments:

  X, Y    :  corrdinates to which drawing position is moved
```

Primitive routine to move the ”pen” to the point with world coordinates (X,Y). No line is drawn.

1.10 label – write text at position relative to viewport

```
pg label "text" ?-side side? ?-displacement disp? ?-coordinate coord?
                ?-justify just? ?standard options?

Arguments:
   text           : text for the label
   -side          : option specifying side (see below)
   -displacement  : displacement of text from "side" (see below)
   -coordinate    : location of text on "side" (see below)
   -justify       : justofication of text string (see below)
```

Write text at a position specified relative to the viewport (outside or inside). This routine is useful for annotating graphs. The text is written using the current values of attributes color-index, line-width, character-height, and character-font although these can be overidden on the command line:

Parameter values:

```
   side   : must include one of the characters 'B', 'L', 'T',
            or 'R' signifying the Bottom, Left, Top, or Right
            margin of the viewport. If it includes 'LV' or
            'RV', the string is written perpendicular to the
            frame rather than parallel to it.
   disp   : the displacement of the character string from the
            specified edge of the viewport, measured outwards
            from the viewport in units of the character
            height. Use a negative value to write inside the
            viewport, a positive value to write outside.
   coord  : the location of the character string along the
            specified edge of the viewport, as a fraction of
            the length of the edge.
   just   : controls justification of the string parallel to
            the specified edge of the viewport. If
            JUST = 0.0, the left-hand end of the string will
            be placed at COORD; if JUST = 0.5, the center of
            the string will be placed at COORD; if JUST = 1.0,
            the right-hand end of the string will be placed at
            at COORD. Other values between 0 and 1 give inter-
            mediate placing, but they are not very useful.
   text  :  the text string to be plotted. Trailing spaces are
            ignored when justifying the string, but leading
            spaces are significant.
```

1.11 page – advance to new page

```
pf page
```

```
Arguments:

  None
```

Advance plotter to a new (sub-)page, clearing the screen if necessary. If the "prompt state" is ON (see PGASK), confirmation is requested from the user before clearing the screen. For an explanation of sub-pages, see PGBEG. PGPAGE does not change the window or the position of the viewport relative to the (sub-)page.

1.12 paper – change the size of the view surface ("paper size")

```
pg page width aspect

Arguments:

   width  : the requested width of the view surface in inches;
            if WIDTH=0.0, PGPAP will obtain the largest view
            surface available consistent with argument ASPECT.
   aspect : the aspect ratio (height/width) of the view
            surface; e.g., ASPECT=1.0 gives a square view
            surface, ASPECT=0.618 gives a horizontal
            rectangle, ASPECT=1.618 gives a vertical rectangle.
```

This routine changes the size of the view surface to a specified width and aspect ratio (height/width), in so far as this is possible on the specific device. It is always possible to obtain a view surface smaller than the default size; on some devices (e.g., printers that print on roll or fan-feed paper) it is possible to obtain a view surface larger than the default. If this routine is used, it must be called immediately after PGBEGIN.

1.13 point – draw a graph markers

```
pg point x y ?standard options?

Arguments:

  X, Y :    world coordinate of point to mark
```

The marker is drawn using the current values of attributes color-index, line-width, and character-height (character-font applies if the symbol number is ¿31). If the point to be marked lies outside the window, no marker is drawn. These attributes and the marker symbol to use may be specified using the standard options.

1.14 polygon – fill a polygonal area with shading

```
pgpolygon n {x(i) y(i)} ?standard options?
```

```
Arguments:

  n             :   number of vertices
  {x(i) y(i)}   :   "N" pairs of real values specifying the
                     vertices
```

Fill-area primitive routine: shade the interior of a closed polygon in the current window. The action of this routine depends on the setting of the Fill-Area Style attribute. If Fill-Area Style is SOLID (the default), the interior of the polygon is solid-filled using the current Color Index. If Fill-Area Style is HOLLOW, the outline of the polygon is drwan using the current line attributes (color index, line-style, and line-width). Other values of the Fill- Area attribute may be allowed in future, e.g., for shading with patterns or hatching. The number of vertices of the poygon must be spcified followed by "n" pairs of X,Y coordinates. Following these you may specify standard options to control line still, fill mode etc.

1.15 text – write text at arbitrary position and angle

```
pg text x y "text" ?-rotation angle? ?-justify just?
                   ?standard options?
```

```
Arguments:
   X    : world x-coordinate.
   Y    : world y-coordinate. The string is drawn with the
          baseline of all the characters passing through
          point (X,Y); the positioning of the string along
          this line is controlled by argument FJUST.
   TEXT : text to plot
```

Primitive routine for drawing text. The text may be drawn at any angle with the horizontal, and may be centered or left- or right- justified at a specified position. Text is drawn using the current values of attributes color-index, line-width, character-height, and character-font. Text is NOT subject to clipping at the edge of the window.

Parameters:

```
   ANGLE  : angle, in degrees, that the baseline is to make
            with the horizontal, increasing counter-clockwise
            (0.0 is horizontal).
   JUST   : controls horizontal justification of the string.
            If JUST = 0.0, the string will be left-justified
            at the point (X,Y); if JUST = 0.5, it will be
            centered, and if JUST = 1.0, it will be right
            justified. [Other values of FJUST give other
            justifications.]
```

1.16 rectangle – draw a rectangle, using fill-area attributes

```
pg rectangle x1 y1 x2 y2 ?standard options?

Arguments:
   X1, X2 : the horizontal range of the rectangle.
   Y1, Y2 : the vertical range of the rectangle.
```

This routine can be used instead of pg polygon for the special case of drawing a rectangle aligned with the coordinate axes; only two vertices need be specified instead of four. On most devices, it is faster to use pg rectangle than pgpolygon for drawing rectangles. The rectangle has vertices at (X1,Y1), (X1,Y2), (X2,Y2), and (X1,Y2).

1.17 style

```
pg style ?standard options? ?-rbg red green blue?
                            ?-hls hue saturation lightness?
                            ?-save? ?-restore?
```

This command sets attributes for subsequent plotting. The attributes set by this command will be used except when attributes are specified explicitly on the command line to a drawing command. In addition to the standard options the following can be set:

```
  -save
```

This option saves the current attributes in a private storage area. They can be restored by calling pg style -restore. Attributes saved are: character font, character height, color index, fill-area style, line style, line width, pen position, arrow-head style. Color representation is not saved.

Calls to pg -save and pg -restore should be paired. Up to 20 copies of the attributes may be saved. pg -restore always retrieves the last-saved values (first-in first-out stack).

```
  -restore
```

This option restores the attributes saved in the last call to pg style -save.

```
  -rgb ci red green blue

  Parameters:
    ci        colour index
    red       red intensity
    blue      blue intensity
    green     green intensity
```

Set color representation: i.e., define the color to be associated with a color index. Ignored for devices which do not support variable color or intensity. Color indices 0-15 have predefined color representations, but these may be changed with the -rgb option. Color indices 16-maximum have no predefined representations: if these indices are used, PGSCR must be called to define the representation. On monochrome output devices (e.g. VT125 terminals with monochrome monitors), the monochrome intensity is computed from the specified Red, Green, Blue intensities as 0.30*R + 0.59*G + 0.11*B, as in US color television systems, NTSC encoding. Note that most devices do not have an infinite range of colors or monochrome intensities available; the nearest available color is used. Examples: for black, set CR=CG=CB=0.0; for white, set CR=CG=CB=1.0; for medium gray, set CR=CG=CB=0.5; for medium yellow, set CR=CG=0.5, CB=0.0.

```
  -hls ci ch cl cs

  Parameters:
   CI       : the color index to be defined, in the range 0-max.
              If the color index greater than the device
              maximum is specified, the call is ignored. Color
              index 0 applies to the background color.
   CH       : hue, in range 0.0 to 360.0.
   CL       : lightness, in range 0.0 to 1.0.
   CS       : saturation, in range 0.0 to 1.0.
```

Set color representation: i.e., define the color to be associated with a color index. Reference: SIGGRAPH Status Report of the Graphic Standards Planning Committee, Computer Graphics, Vol.13, No.3, Association for Computing Machinery, New York, NY, 1979.

1.18 standard options

```
Options:  -font -size -colour -line -width -arrow -top -binned


  -font font

  Parameters:
    font     character font
\begin{verbatim}
```

```
Set the Character Font for subsequent text plotting. Four different
fonts are available:
\begin{verbatim}
    1: (default) a simple single-stroke font ("normal" font)
    2: roman font
    3: italic font
    4: script font
```

This call determines which font is in effect at the beginning of each text string. The font can be changed (temporarily) within a text string by using the escape sequences
fn,
fr,
fi, and
fs for fonts 1, 2, 3, and 4, respectively.

```
  -size size

  Parameters:
  size      new character size (dimensionless multiple of
            the default size).
```

Set the character size attribute. The size affects all text and graph markers drawn later in the program. The default character size is 1.0, corresponding to a character height about 1/40 the height of the view surface. Changing the character size also scales the length of tick marks drawn by pb box and terminals drawn by pg error.

```
  -colour ci

  Parameters:
  ci         the color index to be used for subsequent plotting
             on the current device (in range 0-max). If the
             index exceeds the device-dependent maximum, the
             default color index (1) is used.
```

Set the Color Index for subsequent plotting, if the output device permits this. The default color index is 1, usually white on a black background for video displays or black on a white background for printer plots. The color index is an integer in the range 0 to a device-dependent maximum. Color index 0 corresponds to the background color; lines may be ”erased” by overwriting them with color index 0 (if the device permits this).

If the requested color index is not available on the selected device, color index 1 will be substituted.

The assignment of colors to color indices can be changed with subroutine PGSCR (set color representation). Color indices 0-15 have predefined color representations (see the PGPLOT manual), but these may be changed with PGSCR. Color indices above 15 have no predefined representations: if these indices are used, PGSCR must be called to define the representation.

```
-fill fs

Parameters:
fs        the fill-area style to be used for subsequent
          plotting:
              FS = 1 => solid (default)
              FS = 2 => hollow
          Other values give an error message and are
          treated as 2.
```

Set the Fill-Area Style attribute for subsequent area-fill by polygon etc. At present only two styles are available: solid (fill polygon with solid color of the current color-index), and hollow (draw outline of polygon only, using current line attributes).

```
-line ls

Parameters:
 LS       : the line-style code for subsequent plotting
          (in range 1-5).
```

Set the line style attribute for subsequent plotting. This attribute affects line primitives only; it does not affect graph markers, text, or area fill.

Five different line styles are available, with the following codes:

```
1 (full line), 2 (dashed), 3 (dot-dash-dot-dash), 4 (dotted),
5 (dash-dot-dot-dot). The default is 1 (normal full line).
```

```
-width lw

Parameters:
 LW       : the number of strokes to be used
            (in range 1-201).
```

Set the line-width attribute. This attribute affects lines, graph markers, and text. Thick lines are generated by tracing each line with multiple strokes offset in the direction perpendicular to the line. The line width is specified by the number of strokes to be used, which must be in the range 1-201. The actual line width obtained depends on the device resolution.

```
-arrow fs angle vent


Parameters:
 FS      : FS = 1 => filled; FS = 2 => outline.
           Other values are treated as 2. Default 1.
 ANGLE   : the acute angle of the arrow point, in degrees;
           angles in the range 20.0 to 90.0 give reasonable
           results. Default 45.0.
 VENT    : the fraction of the triangular arrow-head that
           is cut away from the back. 0.0 gives a triangular
           wedge arrow-head; 1.0 gives an open >. Values 0.3
           to 0.7 give reasonable results. Default 0.3.
```

Set style for arrow heads.

```
  -top top

  Parameters:
   TOP     : The size of a drawn error-bar.


  -binned boolean

  Parameters:
   BOOLEAN : 0/1 to control whether plotted data have a continuous
             line (false, 0) or are plotted as binned data in the
             form of a histogram-type plot (true, 1).
```

1.19 stream – control the output stream

```
pg stream option id ?device?

Arguments:
  OPTION  : open close select
  ID      : the steam ID, an integer in the range 1-3.
  DEVICE  : a device specification in the form:
               file/device
            The parameter "file" may be omitted for an interactive device.
            Device is one of:
               /tektronix   /xwindow    /ps          /vps
               Tek Window   X window    Postscript   Postscript
               in xterm                 landscape    portrait.
```

Control the output stream for the graphics device.

```
   SELECT   : Select a plot stream.  This option will switch between
              one of the available streams (1-3).
   CLOSE    : Close the specified stream
   OPEN     : Open a device on the currently selected stream.
```

Examples:

```
# Open an X-window
   pg stream open 1 /xwindow

# Open a second stream for hardcopy postscript output
  pg stream select 2
  pg stream open 2 example.ps/ps
```

1.20 viewport – set the viewport for subsequent drawing

```
pg viewport x1 x2 y1 y2
pg vport    x1 x2 y1 y2

Arguments:
   X1  : x-coordinate of left hand edge of viewport, in NDC.
   X2  : x-coordinate of right hand edge of viewport, in NDC.
   Y1  : y-coordinate of bottom edge of viewport, in NDC.
   Y2  : y-coordinate of top  edge of viewport, in NDC.
```

Change the size and position of the viewport, specifying the viewport in normalized device coordinates. Normalized device coordinates run from 0 to 1 in each dimension. The viewport is the rectangle on the view surface "through" which one views the graph. All the PG routines which plot lines etc. plot them within the viewport, and lines are truncated at the edge of the viewport (except for axes, labels etc). The region of world space (the coordinate space of the graph) which is visible through the viewport is specified by a call to pg window. It is legal to request a viewport larger than the view surface; only the part which appears on the view surface will be plotted.

1.21 window – set window

```
pg window x1 x2 y1 y2

Arguments:
   X1    : the x-coordinate of the bottom left corner
           of the viewport.
   X2    : the x-coordinate of the top right corner
           of the viewport (note X2 may be less than X1).
   Y1    : the y-coordinate of the bottom left corner
           of the viewport.
   Y2    : the y-coordinate of the top right corner
           of the viewport (note Y2 may be less than Y1).
```

Change the window in world coordinate space that is to be mapped on to the viewport.

1.22 update – update display

```
pg update

Arguments:

  None
```

Update the graphics display: flush any pending commands to the output device. This routine empties the buffer created by PGBBUF, but it does not alter the PGBBUF/PGEBUF counter. The routine should be called when it is essential that the display be completely up to date (before interaction with the user, for example) but it is not known if output is being buffered.

1.23 curve – plot a curve or binned data

```
pg curve ?-data { parameters }? ?-spectrum { parameters }?
         ?-function { parameters } ? ?standard options?
```

A line is plotted using the current style attributes or those specified on the command line (see standard options). The data to be used for the plot may be specified either as a list of numbers on the command line, or as a spectral-format data file, or as a function. One of the specifications -data, -spectrum, -function, must be specified.

```
  -data { ?-errx? ?-erry? ?-err2x? ?-err2y? ?-values?
          ndata
          {x(i) y(i) ?errx1(i)? ?errx2(i)? ?erry1(i)? ?err21(i)?} }

        specify the data to plot.  If any of -errx, -erry, -err2x, -err2y
        are given as options then the corresponding error values must be
        given for each point. -errx, -erry specify one value for the
        error bar in either the X or Y directions, -err2x -err2y specifiy
        two values for the error a positive and negative error value
        respectively.  The parameter ndata speicifies the number of data
        points. For example to plot 5 data points with y-errors:

        pg curve \
            -data { -erry 5 0.0 1.24 0.03 0.2 2.43 0.04 0.3 6.52 0.12
                            0.5 3.25 0.09 0.75 1.02 0.02}

  -spectrum { -file filename ?-x column? ?-y column?
              ?-errx column?  ?-erry column?
              ?-err2x column? ?-err2y column? }

         specify a spectral-type file to plot.  A spectral file is
```

an ascii text file with multiple-column data. At the top of the
file any number of header keywords may be specified begin with
a percent character. Two keywords must be present:

```
%ndata    number of data in file
%ncols    number of columns in the file
```

A simple file might look like

```
%ndata 10
%ncols 2
1 .2
2 .2
3 .4
4 .9
5 .85
6 .7
7 .7
8 .3
9 .01
10 .0
```

The parameters are as follows:

```
    -file  file    : file name of spectral file
    -x  column     : column to use for x-axis [1]
    -y  column     : column to use for y-axis [2]
    -errx  column  : column to use for x-axis error [none]
    -erry  column  : column to use for y-axis error [none]
    -err2x column  : column to use for x-axis error (2nd values) [none]
    -err2y column  : column to use for y-axis error (2nd values) [none]
```

Example -- plot a spectral data file test.dat which has 3 columns

```
    pg curve -spectrum { -file test.dat -x 2 -y 3 -erry 1 }
```

-function { function }

specify a function to plot. The function can be given in a
natural algebraic form, but must use "x" for the dummy variable.

Example -- plot a simple function

```
    pg curve -function { sin(x)*cos(x) }
```