Franco Zavatti
Dipartimento di Astronomia, Universitá di Bologna

B O N G O

A Plotting Program under LINUX
with PostScript Output

Version 1.4

User Manual

Technical Report
November 2003

1. INTRODUCTION

The present version is running under Linux 2.4.22 (Mandrake 9.2) and is a Postscript driver which generates the output postscript file bongo.ps.

Bongo was born as an MS-DOS general plotting and data handling program, based on PLOTPE and ASMPLOT, Fortran and Assembler libraries described elsewhere (Rinaldi and Zavatti, 1991) and on a command parser. As can also be derived from its name, Bongo looks like the Tonry's Mongo in the user interface and in several commands.

The normal way Bongo operates, is to get a command, with its alphabetic or numerical parameters, if any, execute it and wait for the next command. Also, it can read from disk file a list of commands, load a command buffer, which can contain 250 lines as maximum, and sequentially execute it. After the commands are entered into the command buffer, they can be listed, edited or deleted (one or more a time).

The data Bongo can read are a virtually undefined number of files, each one beeing organized as a matrix whose maximum dimension is 7000 rows by 20 columns (the minimum column number is 1). Data are used to inizialize up to 4 internal vectors, namely XV, YV (abscissae and ordinates) and EXV, EYV (their relative error bars, if any); allowed operations apply to these vectors. Mathematical operations include arithmetics, logarithms, normalizations and transformation to hours and degrees. Polynomial fitting is also included up to the sixth order. All modified vectors can be saved on disk files either automatically or by giving suitable commands.

A list of Bongo commands is shown in Appendix B, with the number of parameters (9 means string) and one-line description. The commands are listed into the file CMDS.BGO, and Bongo routines validate a given command by comparing the input line to the content of this file. So, attribute of CMDS.BGO should be set to *read only* and the file not modified in any case.

On-line help is available for each command, with the limit that shortcuts are not allowed in the names of help files. For example, PL or PLAY are normally used in place of PLAYBACK, while Help PL gives an error. The correct command will be Help PLAYBACK. The help files are also collected in the file `bongol14.ps` (i.e. this file). If HELP command is entered without any parameter, a list af all available commands is displayed on the screen.

or

at the WWW URL: `http://tikal.bo.astro.it/ zavatti/bongo/bgo.html`

For any problem or comments, the author can be reached at the following mail and E-mail addresses:

Franco Zavatti
Dipartimento di Astronomia
Via Ranzani 1
40127 BOLOGNA (Italy)
Phone: +39 051 20 95712

e-mail: zavatti@tikal.bo.astro.it

2. INSTALLATION

Bongo requires some little operation to be installed correctly:

a) create the directory "bongo" where you prefer (say e.g. /home/bongo).
b) copy the tar file bongo14.tar.Z in this (sub)directory.
c) enter this directory (e.g. `cd /home/bongo`)
d) give the following command in order to uncompress and explode Bongo:

```
tar xZvf bongo14.tar.Z
```

e) Enter the following lines:

```
export BGODIR=/home/bongo
alias bongo='/home/bongo/bongo'
```

in any user's *.profile* or in */etc/profile* in order to allow the use of Bongo to a single user or to all users, respectively.

3. OPERATIONS

At the start, Bongo clears the screen (no welcome or copywrite message appears), shows the prompt '>' and waits for a command or reads the file *bongo.inp*, executes it and exits.

A typical *bongo.inp* file may be:

```
read file-name[.bon]
playback
end [one blank line]
```

After a command file has been created by an external text editor, the command `READ filename` can be given in order to load the existing list of commands, and inizialize the command buffer. Then enter PL. Default extension for `filename` is `.bon`.

Commands can be entered with the minimum number of characters needed to distinguish them one from another. An ambiguous command will be interpreted with the first matching occurrence in the file CMDS.BGO (Appendix B). For example, `IN 1` (which should mean `INSERT 1` in user's mind) will be traslated as `INFO`.

Bongo output is the PostScript file `bongo.ps`

As output on the screen, some information appears after PL and, as last output, the prompt '>' informs the user that the command END, OFF or QUIT must be given. After the command END, the message `bongo.ps has been written on disk` appears and Bongo stops. Now a PostScript viewer like Ghostview can show the graph and print the hardcopy.

Examples of Bongo output are shown in Figs. 9 – 18 (Figs. 1 – 8 refer to the MS-DOS version and are not shown here).

Character dimension can be changed in Bongo by the command TEXT, setting the first parameter to zero for 9-pixel, to one for 15-pixel and to two for 24-pixel characters of the only available Times-Roman font (see Fig.11 for examples of such a font).

Greek alphabet is available with the following limitation: any greek character is given by the corresponding latin character following the backslash symbol ('\' or ascii 92 decimal). The correspondence between latin and greek letters is shown in Fig.10.

Also exponents and deponents (like a^n or a_n) are available in the same way as greek letters, ^ (ascii 94D) and _ (ascii 95D) beeing respectively the special characters adopted.

• – Bongo include an "automatic" execution of one or more command files: by default Bongo reads the file *bongo.inp* in the current directory and executes the commands in it (i.e. commands like those given by the keyboard: read file.bon ; play; end ...). If *bongo.inp* does't exist or it is empty, Bongo jumps to the usual input through the keyboard, showing the prompt '>'.

•– Bongo also handles colors in the RGB space. The command RGB 0/1 allows the use of gray scale or color palette. See the command description for details. Postscript color printers are scarcely available, so the use of color in Bongo should be intended for display on a screen (via Ghostscript or Ghostview, say). Black and white output of color images can give unpleasant results, so the use of gray scale is strongly reccomanded for normal use. The color palette can be displayed by running the file Fig18.bon.

5. BUGS

Some bugs affect Bongo: when known, they have been reported in the respective help files.

Bongo is lacking about several capabilities which can be considered important tools in a general context, like selection of only orizontal or only vertical grid; MACRO command, to run subroutines; conditional jumps and loops within the command buffer (i.e. IF - ENDIF and DO - ENDDO); setting of user defined variables (something like the SET command); also, a maximum of 7000 data and 20 columns may be not sufficent for some application.

7. REFERENCES

Press,W.H., Flannery, B.P., Teukoslsky, S.A. and Vetterling, W.T., *Numerical Recipes*. Cambridge University Press, 1987.

PostScript Language Reference Manual, Addison-Wesley, 1985

PostScript Language Tutorial and Cookbook, Addison-Wesley, 1985

PostScript Language Program Design, Addison-Wesley, 1985

Rinaldi,W. and Zavatti, F., *Plotpe e Asmplot*, Dipartimento di Astronomia, Universitá di Bologna - Technical reports- December 1990

Zavatti, F., *Bongo ver. 2.0*, Dipartimento di Astronomia, Universitá di Bologna - Technical reports- September 1994

8. COMMANDS

A complete description of Bongo commands follows, but before that, three general statements :

1) Parameters may be separated by either comma or space(s).
2) The symbol % in the **first** column acts as a comment, i.e. the command parser does not interpret what follows.
3) A command line is analysed by the parser up to column 40. The remaining columns can be used for comments.

In Appendix B are listed commands not described in what follows. These are the interactive commands of the MS-DOS version (like ALPHA or CURSOR).

BIN

BIN

Format: BIN nbin sbin

Parameters: *nbin* - number of bins. If zero, nbin is computed from sbin. On exit *nbin* will be *nbin+1*. *sbin* - bins width in user units. If zero, sbin is computed from nbin.

Function: Bins a data vector according to nbin, sbin or both. Data vector is assumed to be XV. Maximum *nbin* allowed is 50. If this value is overcome, *nbin* is set to 50 and *sbin* re-computed. *nbin* and *sbin* cannot be **both** zero. In this case an error message is displayed. The main function of this command is to prepare data for the HISTO command. The commands [X or Y] SAVE save binned data on disk file

See also: HISTO, XSAVE, YSAVE, SAVE

Examples:

```
BIN 20 0.3    !  data into 20 bins, step 0.3
BIN 20 0      !  data into 20 bins, step from max, min and nbin
BIN 0 0.3     !  data into nbin bins, step 0.3 (nbin from max, min, sbin)
```

Bugs: none

BOX

BOX

Format: BOX p1 p2 p3 p4 or BOX p1 p2 or BOX

Parameters: p1,p2 are flags: if one of them is zero, numerical values along the corresponding axis are not written. p1 refers to X-axis, p2 to Y-axis. If both parameters are omitted, they are intended to be 1 1. p3,p4 are also flags: they refer to the drawing of the last numerical value on axis. If zero, the last value on the corresponding axis (p3 for x; p4 for y) is not written. Default values for p3 and p4 are 1 1.

Function: Draws a box according to data limits. If LIMITS hasn't been given, XCOL and YCOL are needed in order to compute data limits. Box **neither erases the screen, nor enters the graphic mode**. If interactive mode is used, the command MODE n must be supplied by the user; off-line mode (i.e. PLAYBACK) includes the MODE n command, were n is the default value or the last given mode. The values of p1 and/or p2 and p3 and/or p4 should be zero when multiple, joined plots are drawn. See Fig.14 as an examples of such application.

See also: CLABEL, XLABEL, YLABEL

Bugs: if X/Y/CLABEL are given **before** BOX, they will be written at location (0,0) because the scale of the plot is known after BOX.

CIRCLE

CIRCLE

Format: CIRCLE p1 p2 p3 or CIRCLE p1

Parameters: p1= radius of the circle, in user coordinates, p2=start-angle (def. 0), p3=end-angle (def. 360)
Function: draws an arc of circle, centered at the current point, with radius p1, from the angle p2 to the angle p3, in the counter clock-wise direction. In the default situation (CIRCLE p1) a complete circle is drawn.
See also: ELLIPSE
Bugs: none

CLABEL **CLABEL**

Format: CLABEL string
Parameters: string, a text of 40 characters max.
Function: Writes at the upper left corner of the plot the label string, as a comment.
See also: BOX, LABEL, XLABEL, YLABEL
Bugs: this command must be given **after** BOX

COLOR **COLOR**

Format: COLOR *col*
Parameters: drawing color (0-15).
Function: sets to the value *col* the drawing color. Postscript driver changes *col* to a gray level (1-0) by $gray = 1. - col/15$. Note that in Postscript, gray=1 is white and gray=0 is black.
Bugs: none

COLUMNS **COLUMNS**

Format: COLUMNS *start col, end col*
Parameters: first and last column of a file to be plotted per rows.
Function: defines the *first* and the *last* column of a file that must be plotted "per rows".
Bugs: none

CONNECT **CONNECT**

Format: CONNECT
Parameters: none
Function: Connects data points with the last defined (or default) line type (LTYPE command).
See also: LTYPE
Bugs: none

CONTOUR

CONTOUR

Format: CONTOUR
Parameters: none
Function: draw iso-contours of the image read by RIMA file-name, according to values listed in the local file levels.dat. levels.dat contains in its first line a comment, in the second line the number of levels, followed by the levels, one per line.
Example: four levels must be plotted,

```
this file is levels.dat
4
123
257
333
38
```

See also: RIMA
Bugs: none. Note that the maximum number of levels is 50

DATA

DATA

Format: DATA file-name *or* DATA ?
Parameters: *file-name* and its path, if necessary. Max 40 characters.
Function: Opens the file with the data and inizializes string variable which contains the 255 characters max length data columns and comments. Data are assumed to be organized in columns, separated by space(s) or comma(s). If a column contains a non-digits element it will be transformed to 9999. Any row of such a column must contain non-numeric data. If *file-name* is ?, Bongo asks for file-name from the keyboard.
Bugs: none

DELETE

DELETE

Format: DELETE L1,L2 or DELETE L1
Parameters: L1, L2 starting and ending lines, or the only line to be deleted
Function: Deletes from the command buffer the lines from L1 to L2 (extrema included) or the line L1 and re-organizes the command buffer.
Bugs: none

DELTA

DELTA

Format: DELTA p1,p2 or DELTA
Parameters: p1, p2 percent of the axes lengths (X and Y respectively). Default values are 0.,0.

Function: Defines the amount, as percent of the axis length, of the user-scale displacement with respect the axes origin. In practice it is defined an area, where data are plotted, smaller than the one given by the command BOX.

Example: if PHYSICAL command has not been given (i.e. the limits of the plotting area are X1, X2 = 120, 520 and Y1, Y2 = 200, 600), DELTA 5 7 means that the new positions (X[1/2]DX, Y[1/2]1DY) of the first tick on X and Y axes are (in pixels):

X1DX=X1+(X2–X1+1)*p1/100 = 120+401*0.05 = 140
X2DX=X2–(X2–X1+1)*p1/100 = 520–401*0.05 = 500
Y1DY=Y1+(Y2–Y1+1)*p2/100 = 200+401*0.07 = 228
Y2DY=Y2–(Y2–Y1+1)*p2/100 = 600–401*0.07 = 572

X2DX and Y2DY being the position of the X and Y last tick.

See also: BOX, PHYSICAL, Fig. 16 (or fig16.bon)

Bugs: none

DIGITS

DIGITS

Format: DIGITS *ndx ndy* or DIGITS

Parameters: ndx- decimal digits on x-axis. ndy- decimal digits on y-axis

Function: Sets the number of decimal digits of axes numerical labels. Default values are 2 decimal digits for both axes. Default value(s) can be set also with both ndx and ndy less than zero.

Examples:

```
DIGITS !ndx=2, ndy=2
DIGITS 2 2 !as above
DIGITS -3 -5 !as above
DIGITS 0 1 !ndx=0 , ndy=1
```

Bugs: none

DOT

DOT

Format: DOT

Parameters: none

Function: draws at current point a symbol in the last-defined (or default) style. Use PTYPE command to change point style.

See also: DRAW, PTYPE, RELOCATE

Bugs: none

DRAW

DRAW

Format: DRAW X Y

Parameters: X,Y coordinates of the end of the segment in user coordinates.

Function: draws from the current point to X,Y a line with both the last- defined color and style. Use LTYPE command to change line aspect.
See also: DOT, LTYPE, RELOCATE
Bugs: none

ELLIPSE **ELLIPSE**

Format: ELLIPSE p1 p2 p3 p4 p5 or ELLIPSE p1 p2 p3 or ELLIPSE p1 p2
Parameters: p1= semi-major axis; p2=axial ratio; p3=rotation (degrees, def. 0), p4= start-angle(def. 0), p5=end-angle (def. 360). p1 is given in user coordinates, angles in counter clock-wise direction.
Function: draws an arc of ellipse, centered at the current point, with semi-major axis p1, axial ratio p2, turned of p3 degrees, from the angle p4 to the angle p5, in the counter-clock wise direction. If p2=1 a circle is drawn, also if the command CIRCLE is available. In the default situation (ELLIPSE p1 p2) a complete, unrotated ellipse is drawn.
See also: CIRCLE
Bugs: none

END **END**

Format: END
Parameters: none
Function: Exit from Bongo to Linux OS. It does not save the actual command buffer (use WRITE for that). END terminates also INSERT command. When used in BONGOPS, END closes the output file bongo.ps and writes on the screen the message "Bongo.ps written on disk". It also writes the logo '*Bongo-Linux 1.2*' in the bottom right corner of the output page. In practice, END exits Bongo and write the logo; OFF closes bongo.ps, doesn't either write the logo or exit Bongo; QUIT works like END but doesn't write the logo. END is for normal use; OFF for 'see, correct, run again' sessions and QUIT when plots must be inserted in a text, where the logo can give some problem. See also: QUIT, OFF Bugs: none
See also: OFF, QUIT, WRITE
Bugs: none

ERASE **ERASE**

Format: ERASE
Parameters: none
Function: Clears the actual PostScript page.
Bugs: none

EXCOL

EXCOL

Format: EXCOL n flag
Parameters: n number of a column of data file; flag (=1 draw; $\neq$ 1 not draw)
Function: Assigns the column #n of the data file to Bongo internal vector EXV (errors of XV vector). This command traces the last read error bars, associated to any next data vector XV, also if XV does'nt have errors. To avoid that, give again the command with *flag=0* after the right vectors have been drawn.
Example:

```
DATA file-name          !  reads input data
XCOL 1                  !  assigns column #1 to XV
EXCOL 3 1               !  assigns column #3 to EXV
POINTS                  !  error bars will be traced (flag=1)
EXCOL 3 0               !  sets off x-error bar drawing(flag=0)
```

See also: EYCOL
Bugs: none

EXOPER

EXOPER

Format: EXOPER opcode, constant
Parameters: code of selected operation, constant required by operation (or 1 or 0 as necessary)
Function: Applies the operation defined by opcode (with a constant if necessary) to EXV internal vector
use EXOPER BEFORE the corresponding XOPER
Available opcodes are:
1) vector+constant
2) vector*constant
3) vector/constant
4) LOG10(vector)*constant
5) LN(vector)*constant
6) vector**constant
7) vector/vector(constant)
8) vector-vector(constant)
9) 10**(vector*constant)
10) EXP(vector*constant)
11) vector/$\sum$(vector values)
12) vector to hh.mmss or dd.mmss
Modifications remain in effect until another EXOPER is given or command is cancelled. Log of negative number is set to -50. Codes 7 and 8 allow normalization with respect any element (set constant to the ordinal of the element) of the EXV vector. Code 11 allows normalization with respect the integral of vector. The file SUM.OUT, containing the

sum of the vector is also created. Code 12 assumes that data are expressed in terms of minor unit (e.g. in arcsec if degrees.minutesseconds are required)

Example: if data are D$\pm\sigma$, the command EXOPER 4 -2.5 gives:

```
A=D-σ
B=D+σ
Σ=-2.5*[LOG(A)-LOG(B)]/2
RELOCATE (-2.5*LOG(D)-Σ),Y
DRAW [-2.5*LOG(D)+Σ],Y
```

The same happens for any other operation.

Bugs: none

EXSAVE

EXSAVE

Format: EXSAVE

Parameters: none

Function: Saves the last-defined EXV vector on the disk file SAVE.NN (NN=01, 02 ..., 10). This is a working file: SAVE command saves all SAVE.NN into the file SAVEnn.SAV, in the same order the working files have been originally written. Only numerical values are saved, so, if you wish to retain SAVEnn.SAV, add any comment you need.

Example:

```
DATA file-name          !  reads input data
XCOL 1                  !  assigns the column #1 to XV
XOPER 4 1               !  decimal logarithm of XV
XSAVE                   !  writes logarithms to disk file SAVE.01
EXCOL 3                 !  assigns the column #3 to EXV
EXOPER 4 1              !  gets logarithm of errors
EXSAVE                  !  write log(errors) to disk file SAVE.02
SAVE 1                  !  creates SAVE01.SAV with column taken from
                        !  SAVE.01 and SAVE.02.
```

Bugs: none

EYCOL

EYCOL

Format: EYCOL n flag

Parameters: *n* number of a column of data file; *flag* (=1 draw; $\neq$ 1 not draw)

Function: Assigns the column #n of the data file to Bongo internal vector EYV (errors of YV vector). This command traces the last read error bars, associated to any next data vector YV, also if YV does'nt have errors. To avoid that, give again the command with *flag=0* after the right vectors have been drawn.

See also: EXCOL

Example:

```
DATA file-name          !  reads input data
```

```
LINES 4 130              !  data between 4th and 130th line
YCOL 4                   !  assigns column #4 to YV
EYCOL 3 1                !  assigns column #3 to EYV
POINTS                   !  error bars will be traced (flag=1)
EYCOL 3 0                !  sets off y-error bars drawing (flag=0)
```

Bugs: none

EYOPER **EYOPER**

Format: EYOPER opcode, constant

Parameters: code of selected operation, constant required by operation (or 1 or 0 as necessary)

Function: Applies the operation defined by opcode (with a constant if necessary) to EYV internal vector

use EYOPER BEFORE the corresponding YOPER

Available opcodes are:

1) vector+constant
2) vector*constant
3) vector/constant
4) LOG10(vector)*constant
5) LN(vector)*constant
6) vector**constant
7) vector/vector(constant)
8) vector-vector(constant)
9) 10**(vector*constant)
10) EXP(vector*constant)
11) vector/$\sum$(vector values)
12) vector to hh.mmss or dd.mmss

Modifications remain in effect until another EYOPER is given or command is cancelled by DELETE. Log of negative number is set to -50. Codes 7 and 8 allow normalization with respect any element (set constant to the ordinal of the element) of the EYV vector. Code 11 allows normalization with respect the integral of vector. The file SUM.OUT, containing the sum of the vector is also created. Code 12 assumes that data are expressed in terms of minor unit (e.g. in arcsec if degrees.minutesseconds are required)

Example: if data are D$\pm\sigma$, the command EYOPER 4 -2.5 gives:

A=D-σ
B=D+σ
Σ=-2.5*[LOG(A)-LOG(B)]/2
RELOCATE (-2.5*LOG(D)-Σ),Y
DRAW [-2.5*LOG(D)+Σ],Y

The same happens for any other operation.

Bugs: none

EYSAVE **EYSAVE**

Format: EYSAVE
Parameters: none
Function: Saves the last-defined EYV vector on the disk file SAVE.NN (NN=01, 02 ..., 10). This is a working file: SAVE command saves all SAVE.NN into the file SAVEnn.SAV, in the same order the working files have been originally written. Only numerical values are saved, so, if you wish to retain SAVEnn.SAV, add any comment you need.
Example:

```
DATA file-name          !  reads input data
YCOL 1                  !  assigns the column #1 to YV
YOPER 4 1               !  decimal logarithm of YV
YSAVE                   !  writes logarithms to disk file SAVE.01
EYCOL 3                 !  assigns the column #3 to EYV
EYOPER 4 1              !  gets logarithm of errors
EYSAVE                  !  writes log(errors) to disk file SAVE.02
SAVE 2                  !  creates SAVE02.SAV with columns taken from
                        !  SAVE.01 and SAVE.02.
```

Bugs: none

FILL **FILL**

Format: FILL
Parameters: none
Function: Flag; sets *on* the filling capability of Bongo. Any successive symbol or histogram bin will be filled with the tracing color.
See also: NOFILL
Bugs: none, but the areas (or symbols) exceeding box limits are not filled at all.

FIT **FIT**

Format: FIT deg wfl or FIT deg
Parameters: deg is the degree of polynomial. Max value is 6. If deg< 0 parameters are added to a log-file. wfl is a flag : wfl=0 no weights; wfl=1 weighted fit. EVY contains weights. If wfl is not written, its default value is zero. If wlf=1 but EVY has not been loaded, a warning message is written and the unweighted fit is computed.
Function: Computes the least squares polynomial fit of XV, YV data by
YC= a+bXV+cXV**2...+gXV**6.
On exit, YV is set to YC (i.e. YV contains the computed data). User must provide plotting instructions in order to display the fitting function. The output file POLY.OUT is created. Such file contains the degree of polynomial, the phrase `"weighting vector is EYV"` if wfl

≠0, the standard deviation of residuals, the covarince COV(XV,YV) and the standard deviations $\sigma(XV)$ and $\sigma(YV)$ of the data vectors, the correlation coefficient, if deg=1, the deg+1 parameters of the fitting function with the respective standard deviations, their covariance matrix and a table with an ordinal, the original abscissae, the original and the computed ordinates and the limits of the 1-σ confidence interval. POLY.OUT is overwritten by the next FIT n command.
If deg < 0 the file POLY.APP is opened with access=append and the main output parameters (s.d. of fit, coefficients and their errors) are added any time FIT is called.

Example:

```
YCOL 1                  !  assigns the column #1 to XV
YCOL 2                  !  assigns the column #2 to YV
PTYPE 4 3 2             !  sets point type
POINTS                  !  draws observed points
FIT 3                   !  computes 3rd degree unweighted fit
                        !  POLY.OUT is written
DATA file.ext           !  YV modified, so re-open data file #1 to YV
YCOL 2                  !  assigns again the column #2 to YV
EYCOL 9                 !  assigns the column #9 to EYV
FIT 4 1                 !  computes 4th degree weighted fit
                        !  POLY.OUT is over-written
LTYPE 1 2               !  defines line pattern and color
CONNECT                 !  draws line that fits given data
```

Bugs: none

GAUSS

GAUSS

Format: GAUSS average, dispersion

Parameters: Expectation value (or average) and dispersion (or square root of variance) of your data set.

Function: compute a **standard normal distribution** whose parameters are the same of actual data set (derived after XYSTAT, for example). Remember that "standard" means "with zero average"; if you want a gaussian centered at a value different from zero, **add** this value to the XV vector. After this command, the vectors XV and YV contain the normal distribution.

Example:

```
..          ..
gauss 47 9.68        define Xm=47 σ=9.68
xoper 1 47       add Xm to XV
ltype 1 15       define a solid, black line
connect       draw the graph
..          ..
```

Bugs: none

GRID

GRID

Format: GRID pattern, color, flag
Parameters: pattern and color of the grid, flag=1 draws also small ticks
Function: Draws a grid superimposed to the plot, corresponding to the principal ticks. If flag=1 the small ticks are also gridded. Available patterns are:

0 No LINE
1 '——————-'
2 '—— – ——'
3 '———— '
4 '– – – – '
5 '. . . . . .'
6 '. . . . . '
7 '— . — . — '

Examples:

```
GRID 6 8 2     draws a dotted (6) grid in gray (8), main ticks (≠ 1).
GRID 4 4 1     draws a dashed (4) grid in red (4), all ticks (1).
```

Bugs: none, but use the command before LABEL, so labels can overwrite the grid.

HELP

HELP

Format: (1) HELP or (2) HELP command-name
Parameters: none or the name of a command for detailed help.
Function: (1) lists all available commands in Bongo; (2) gives the help of the specific command. Use the COMPLETE name of the command, no abbreviations.
Example: HELP ALP is wrong (Help routine cannot find help file); HELP ALPHA is correct
Bugs: none

HISTO

HISTO

Format: HISTO wbin color flag or HISTO wbin color
Parameters: *wbin*- width of bins. If command BIN has been previously given, wbin is known and its value in this command is uninfluent. *color*- drawing (and filling, if any) color. *flag* - controls the histogram kind: if 0 (default), histogram start at the minimum of box y-value; if 1, histogram starts at the minimum of actual y data vector (histograms can be shifted up and down); if 2, histogram starts at zero (both positive and negative bins can be traced).
Function: draws an histogram of binned data given both as external file or after BIN command. Maximum allowed number of bins is 50.
Bugs: filling of bins exceeding box limits doesn't work

IDENT

IDENT

Format: IDENT or IDENT cc
Parameters: none or a 2-characters identification
Function: writes, at the upper right corner, date and time of the plot. Please note that date format is *weekday-mm-dd-hh:mm:ss-yyyy*. If the couple of characters 'cc' is given, it will be added to the above string.
Bugs: none

INFO

INFO

Format: INFO
Parameters: none
Function: Displays general information on Bongo (author address, references).
Bugs: none

INPUT

INPUT

Format: INPUT file name
Parameters: file name of data given by the keyboard
Function: Allows data to be entered by the keyboard from within Bongo. Data are not immediately available: they are written in the disk file `file name` and can be plotted by the normal sequence of commands (DATA, LINES, LIMITS, XCOL, YCOL, PTYPE, POINTS). Data couples can be separated by space or comma. Input of data ends with the couple 999,1. This command is intended as a purely manual one. It does not appear in the command buffer (e.g. after LIST) and is not executed by PLAYBACK, when given from within BONGO. Nevertheless, if INPUT appears in a command file created before entering BONGO, it will be executed after any PLAYBACK command. If `file name` is omitted, the DOS warning: `File name missing or blank - Please enter file name`, appears.
Example:

```
>INPUT myfile.dat
Comment:
enter 80-chars max comment
Data:
enter N couples of data, ending with 999,1
-0.1,23.7
-0.87 2.2
2056,0.0003
999,1           !  this end input and close myfile.dat
>
```

To plot *myfile.dat* (suppose another data file has been plotted, so that commands like LIMITS or BOX were already entered):

```
DATA myfile.dat
LINES 2 100
XCOL 1
YCOL 2
EXCOL 3 2    !  if above plot included error
EYCOL 3 2    !  bars, set them off
PTYPE 4,4,2
POINTS
```

Bugs: none

INSERT **INSERT**

Format: INSERT line-number
Parameters: line-number, the ordinal which identifies the command in the command buffer after LIST.
Function: Allows insertion of new command(s) AFTER the command defined by line-number. INSERT prompt is I:. END terminates insertion. Use INS 0 to add a command at the top of the buffer
Bugs: none

LABEL **LABEL**

Format: LABEL string
Parameters: string, a text of 40 characters max.
Function: Writes *string* at the last-defined position. Dimension, color and angle derive by the last TEXT command (or by default values).
See also: CLABEL, FONT, RELOCATE, XLABEL, YLABEL, TEXT, TTEXT
Bugs: none

LIMITS **LIMITS**

Format: LIMITS x1 x2 y1 y2
Parameters: first and last abscissa; first and last ordinate, in user coordinates.
Function: Defines the limits of plotting area in user coordinates. Scaling depends on these numbers.
Bugs: none

LINES **LINES**

Format: LINES l1 l2
Parameters: first and last line

Function: Defines the first and the last line of the actually open data file. Internal vectors XV and YV will be inizialized by data from I1 to I2, extrema included. If l1 or l2 or both are less than zero, Bongo asks for the input of l1 and l2 via the keyboard.

Bugs: none

LIST **LIST**

Format: LIST lin1 lin2 or LIST

Parameters: first and last line or none

Function: Lists the command buffer between lines lin1 and lin2, extrema included. No parameter means "all lines". List stops every 22 lines and shows: $< RET >$ to continue $< ESC >$ to exit. Hit Escape to stop listing, or any other key to continue.

Bugs: none

LTICKS **LTICKS**

Format: LTICKS nx, ny

Parameters: Number of x and y intervals

Function: Defines the large ticks, i.e. the number of intervals x and y axes are divided in. In correspondence of such ticks, the axes are labelled with numerical values.

See also: STICKS

Bugs: none

LTYPE **LTYPE**

Format: LTYPE line-code, color

Parameters: numerical code for line pattern, color

Function: Defines line pattern and color. Lines are effectively drawn by the command CONNECT. Available codes and patterns are:

0 No LINE
1 '——————-'
2 '—— – ——'
3 '————— '
4 '– – – – '
5 '. . . . . .'
6 '. . . . . '
7 '— . — . — '

Ltype remains in effect until another Ltype is given.

See also: CONNECT

Bugs: none

NOFILL **NOFILL**

Format: NOFILL
Parameters: none
Function: Flag; sets to OFF the Bongo filling capability.
See also: FILL
Bugs: none

OFF **OFF**

Format: OFF
Parameters: none
Function: Closes the output file bongo.ps and prepares itself to open another bongo.ps. It doesn't write the logo '*Bongo-Linux 1.4*' in the bottom right corner of the page. In practice this command is useful to display bongo.ps via Ghostview without exit from the actual Bongo session, like a sort of 'interactive' session. In practice, END exits Bongo and write the logo; OFF closes bongo.ps, doesn't either write the logo or exit Bongo; QUIT works like END but doesn't write the logo. END is for normal use; OFF for 'see, correct, run again' sessions and QUIT when plots must be inserted in a text, where the logo can give some problem.
See also: END, QUIT, WRITE
Bugs: none

PAUSE **PAUSE**

Format: PAUSE
Parameters: none
Function: Stops the execution of the next command until a key is pressed. An acustic signal outlines the presence of this command. No message is written to the screen.
Bugs: none

PBOX **PBOX**

Format: PBOX xor yor
Parameters: X and Y screen coordinates in pixels
Function: Defines the origin of BOX. X and Y are the coordinates of the bottom left corner of the plotting region.
Bugs: none

PCOM **PCOM**

Format: PCOM
Parameters: none
Function: displays the values of the variables in the common blocks of both Plotpe and Bongo libraries. This command has been written for author's use, so not all may be clear; nevertheless PCOM can be useful in some cases in order to control input values.
Bugs: none

PHYSICAL **PHYSICAL**

Format: PHYSICAL xstart xend ystart yend
Parameters: screen coordinates (pixels) of the plotting area
Function: defines the physical screen coordinates of the plotting area. Corresponds to the combination PBOX+XAXIS+YAXIS.
Bugs: none

PLAYBACK **PLAYBACK**

Format: PLAYBACK
Parameters: none
Function: Executes the command buffer from the first to the last command. No jump is allowed.
Bugs: none

POINTS **POINTS**

Format: POINTS
Parameters: none
Function: Draws a symbol, following the last PTYPE command style or the default settings, at the x and y user coordinates defined in the XV and YV internal vectors.
See also: PTYPE
Bugs: none

PTYPE **PTYPE**

Format: PTYPE code,dimension,color
Parameters: numerical code for symbol shape, dimension (side or radius) in pixels, color.

Function: Defines symbol code, dimension and color. Symbols are effectively drawn by the command POINTS. Available codes are:

0 No Symbol
1 +
2 x
3 Triangle
4 Square
5 Diamond
6 Circle
7 Dot (1 pixel)
8 Asterisk
9 Star (min dim: 3 pixels)

Ptype remains in effect until another Ptype is given.

See also: POINTS

Bugs: none

QUIT **QUIT**

Format: QUIT

Parameters: none

Function: Exits from Bongo to Linux OS. Does not save the actual command buffer (use WRITE for that). In Bongo-Linux (i.e. Postscript) closes the output file bongo.ps. Doesn't write the logo ' *Bongo-Linux 1.4*' in the bottom right corner of the page. In practice, END exits Bongo and write the logo; OFF closes bongo.ps, doesn't either write the logo or exit Bongo; QUIT works like END but doesn't write the logo. END is for normal use; OFF for 'see, correct, run again' sessions and QUIT when plots must be inserted in a text, where the logo can give some problem.

See also: END, OFF, WRITE

Bugs: none

READ **READ**

Format: READ file-name

Parameters: file-name, 40 characters max string containing the name of a list of Bongo commands

Function: Reads a list of commands from a disk file and inizializes the command buffer. The default extension of file-name is `.bon`.

Examples:

```
READ cream.xyz     !look for the command file cream.xyz
READ cream         !look for the command file cream.bon
```

Bugs: none, but remember to give the full path, if necessary

RELOCATE **RELOCATE**

Format: RELOCATE *xuser yuser*
Parameters: x and y user coordinates
Function: sets the current point to *xuser, yuser*, i.e. defines where the next writing operation begins on the screen.
Bugs: none

RGB **RGB**

Format: RGB p1
Parameters: p1=1 means that the rgb color space will be used in what follows; if p1=0 the graphs are drawn with the normal gray scale palette.
Function: makes available the Red, Green, Blue color space. After RGB 1 color palette is used (please run fig18.bon and display the result to show this palette). To write text in color, use the command TEXT.
See also: COLOR, LTYPE, PTYPE, TEXT, fig18.bon
Bugs: none

RIMA **RIMA**

Format: RIMA file-name
Parameters: file-name containing a real matrix
Function: Load the image to be traced by CONTOUR. The image must be written as REAL*4 (100x100) max. The first row contains NY and NX, of columns and rows, respectively.
Example:

```
.
.
real aa
.
open (12,file='matrix.dat')
write(12,*)ny,nx
do i=1,ny
write(12,*) (aa(i,j), j=1,nx)
enddo
close (12)
.
.
.
```

See also: CONTOUR, WRIMA
Bugs: none. In one case I've had an "end-of-file" error, but the graph was fine. If an error happens, try to plot the contour levels.

SAVE **SAVE**

Format: SAVE n
Parameters: ordinal to distinguish saved files
Function: Saves all SAVE.NN into the file SAVEnn.SAV, in the same order the working files have been originally written. Only numerical values are saved, so, if you wish to retain SAVEnn.SAV, add any comment you need. Any SAVE command resets to 01 the NN of SAVE.NN file.
See also: EXSAVE, XSAVE, EYSAVE, YSAVE
Examples:
1)

```
DATA file-name          !  reads input data
XCOL 1                  !  assigns the column #1 to XV
YCOL 3                  !  assigns the column #3 to YV
XSAVE                   !  writes XV to disk file SAVE.01
YSAVE                   !  writes YV to disk file SAVE.02
SAVE 1                  !  creates SAVE01.SAV with columns taken from
                        !  SAVE.01 and SAVE.02.
```

2)

```
DATA file-name          !  reads input data
XCOL 1                  !  assigns the column #1 to XV
YCOL 3                  !  assigns the column #3 to YV
XSAVE                   !  writes XV to disk file SAVE.01
YSAVE                   !  writes YV to disk file SAVE.02
SAVE 1                  !  creates SAVE01.SAV with columns taken from
                        !  SAVE.01 and SAVE.02.
DATA file-name1         !  reads new input data
XCOL 1                  !  assigns the column #1 to XV
YCOL 3                  !  assigns the column #3 to YV
XSAVE                   !  writes XV to disk file SAVE.01
YSAVE                   !  writes YV to disk file SAVE.02
SAVE 2                  !  creates SAVE02.SAV with columns taken from
                        !  SAVE.01 and SAVE.02.
```

3)

```
DATA file-name1         !  reads input data
XCOL 1                  !  assigns the column #1 to XV
YCOL 3                  !  assigns the column #3 to YV
XSAVE                   !  writes XV to disk file SAVE.01
YSAVE                   !  writes YV to disk file SAVE.02
DATA file-name2         !  reads another input data
XCOL 1                  !  assigns the column #1 to XV
YCOL 3                  !  assigns the column #3 to YV
XSAVE                   !  writes XV to disk file SAVE.03
YSAVE                   !  writes YV to disk file SAVE.04
SAVE 10                 !  creates SAVE10.SAV with columns taken from
```

```
!  SAVE.01, SAVE.02, SAVE.03 and SAVE.04.
```

Bugs: none

SMOOTH **SMOOTH**

Format: SMOOTH step
Parameters: step - width of smoothing window, in # of data points.
Function: Computes the smoothing function of the last defined YV data vector. Smoothing values substitute the original YV data. Plotting of the smoothed data is the same as in FIT and is left to user. Smoothing routine is taken from *Numerical Recipes*.
See also: FIT
Example:

```
DATA file-name          !  reads input data
XCOL 1                  !  assigns the column #1 to XV
XOPER 4 1               !  decimal logarithm of XV
YCOL 3                  !  assigns the column #3 to YV
PTYPE 3 3 3             !  defines symbol aspect
POINTS                  !  draws symbols for original data
SMOOTH 6                !  smooths YV with a six-points window
LTYPE 1 3               !  defines line type
CONNECT                 !  draws line through smoothed data
```

Bugs: none

SORT **SORT**

Format: SORT
Parameters: none
Function: sorts XV vector in ascending order and rearranges YV consequently. Original XV and YV are modified.
Bugs: EXV and EYV error vectors are not sorted.

STAIRS **STAIRS**

Format: STAIRS color flag or STAIRS color
Parameters: *color*- drawing (and filling, if any) color. *flag* - controls the histogram kind: if 0 (default), histogram start at the minimum of box y-value; if 1, histogram starts at the minimum of actual y data vector (histograms can be shifted up and down); if 2, histogram starts at zero (both positive and negative bins can be traced).
Function: draws a stair-shaped histogram of binned data given both as external file or after BIN command. Maximum allowed number of bins is 50.
Bugs: filling of bins exceeding box limits doesn't work

STICKS **STICKS**

Format: STICKS nx ny
Parameters: Number of x and y intervals
Function: Defines the small ticks, i.e. the number of intervals large ticks are divided in.
See also: LTICKS
Bugs: none

TEXT **TEXT**

Format: TEXT dim col angle *or* TEXT dim col *or* TEXT
Parameters: text dimension, color, angle
Function: Defines dimension of text characters, in units of actually loaded font, their color and drawing direction. Please note that orientation is *clockwise*. dim=0 means 9-pixel characters; dim=1 means 15-pixel characters; dim=2 means 24-pixel characters. These settings remain in effect until another TEXT command is given. If TEXT is entered with only *two* parameters or without at all, the default values (as listed in bconfig.plt) are set. Parameter *dim* can be only an integer ≥ 1.
Example:
```
if \ bongo \ 16x9.fnt has been loaded,
TEXT 1 4 45 means 16x9 pixels chars , red color, 45 degrees
TEXT 2 14 76 means 32x18 pixel chars , yellow color, 76 degrees
TEXT 1 1 means dim 1, blue(1) color, 0 degrees (default)
TEXT means dim 1, white(15) color, 0 degrees(defaults)
TEXT 0 15 draws in black color 9-pixel Times Roman font
```
Bugs: none

VERSION **VERSION**

Format: VERSION
Parameters: none
Function: displays on the screen the version number of the Bongo actually in use.
Bugs: none

VWRITE **VWRITE**

Format: VWRITE *p1,p2* or VWRITE
Parameters: *p1* first line and *p2* last line of the data vectors to be displayed. Default values are 1 and max number of lines
Function: Vectors WRITE. Display on the screen the last-defined XV, YV and EXV, EYV internal vectors from line *p1* to line *p2*

Bugs: none, but the listing may be continous. Use Pause key or Ctrl S code to stop it.

XAXIS **XAXIS**

Format: XAXIS axis-length
Parameters: x-axis length in pixels
Function: Sets X-axis length. If length overcomes the limits of actual graphic mode the plot is truncated. String from IDENT is written in the left side (wraparound).
Bugs: none

XCOLUMN **XCOLUMN**

Format: XCOLUMN n
Parameters: number of a column in the data file
Function: Assigns the column #n of the data file to Bongo internal vector XV. If LIMITS is not in effect, this command also computes max and min of the vector.
Bugs: none

XLABEL **XLABEL**

Format: XLABEL string
Parameters: string, a text of 40 characters max.
Function: Writes string centered below the X-axis of the plot. No fixed angle is given, so control the last given TEXT command and give a new TEXT with angle=0, if necessary.
See also: BOX, CLABEL, YLABEL
Bugs: This command must be given **after** BOX

XOPER **XOPER**

Format: XOPER opcode, constant
Parameters: code of selected operation, constant required by operation (or 1 or 0 as necessary)
Function: Applies the operation defined by opcode to the XV internal vector.
Available opcodes are:

1) vector+constant
2) vector*constant
3) vector/constant
4) LOG10(vector)*constant
5) LN(vector)*constant
7) vector/vector(constant)
8) vector-vector(constant)
9) 10**(vector*constant)
10) EXP(vector*constant)
11) vector/$\sum$(vector values)

6) vector**constant

12) vector to hh.mmss or dd.mmss
13) $\sum_1^i$ vec/$\sum_1^N$ vec

21) vector1(vector2*const)
22) vector1*(vector2*const)
23) vector1/(vector2*const)
24) vector1=vector2
25) vector2=vector1

Original data vector will be modified by ANY of the above operations. Modifications remain in effect until another XOPER is given or command is cancelled. Log of negative number is set to -50.

Codes 7 and 8 allow normalization with respect any element (set constant to the ordinal of the element) of the X vector.

Code 11 allows normalization with respect to the integral of the vector. The file SUM.OUT, containing the sum of the vector, is also created.

Code 12 assumes that data are expressed in terms of minor unit (e.g. in arcsec if degrees.minutesseconds are required)

Code 13 computes the empirical distribution function (cumulative distribution) of an early binned vector. The file SUM.OUT, containing the sum of the data vector is also created.

Codes above 20 allow operations between XV and EXV and/or between YV and EYV vectors. Results are copied into XV and YV vectors respectively (i.e. into vector1).

See also: EXOPER,EYOPER,YOPER

Example: only an example referred to 21-25 codes is presented. Two files must be read and three vectors extracted: one abscissa (common to both data vectors) and two ordinates (o1 and o2 from the files file1.dat and file2.dat, respectively). Ordinates must be arranged in order to give a new ordinate o3=log (o1)-log(o2). The command file is:

```
LIMITS x1 x2 y1 y2       !  limits in user coords
BOX                      !  draw axes and ticks
DATA [path]file2.dat     !  note:  first one is 2.nd file
LINES 12 200
XCOL 1                   !  load abscissa into XV
YCOL 2                   !  second ordinate o2 into YV
YOPER 4 1                !  log (o2)
YOPER 25 1               !  put log(o2) into EYV (i.e.  vect2=vect1)
DATA [path]file1.dat     !  read first file
LINES 12 200
YCOL 2                   !  first ordinate o1 into YV
YOPER 4 1                !  log (o1)
YOPER 21 -1              !  subtract:  YV=YV+(-1*EYV)
LTYPE 1 15               !  set white, solid line
CONNECT                  !  draw line
```

Bugs: none, but be sure that in code 23 vector2 is not a null vector.

XROW **XROW**

Format: XROW n
Parameters: number of a row in the data file
Function: Assigns the row #n of the data file to Bongo internal vector XV. If LIMITS is not in effect, this command also computes max and min of the vector.
Bugs: none

XSAVE **XSAVE**

Format: XSAVE
Parameters: none
Function: Saves the last-defined XV vector on the disk file SAVE.NN (NN=01, 02 ..., 10). This is a working file: SAVE command saves all SAVE.NN into the file SAVEnn.SAV, in the same order the working files have been originally written. Only numerical values are saved, so, if you wish to retain SAVEnn.SAV, add any comment you need (use DOS command to do that).
Example:

```
DATA file-name          !  reads input data
XCOL 1                  !  assigns the column #1 to XV
XOPER 4 1               !  decimal logarithm of XV
XSAVE                   !  writes logarithms to disk file SAVE.01
YCOL 3                  !  assigns the column #3 to YV
YOPER 7 1               !  gets YV(I)=YV(I)/YV(1)
YSAVE                   !  write normalized data to disk file SAVE.02
SAVE 8                  !  creates SAVE08.SAV with columns taken from
                        !  SAVE.01 and SAVE.02 respectively
```

Bugs: sometime a zero-length file SAVE.NN is created. Delete the file and try againg PL.

XSTAT **XSTAT**

Format: XSTAT
Parameters: none
Function: Computes and saves on the disk file STAT.OUT the number of data, mean value, standard deviation, variance, third moment(skewness) and forth moment (kurtosis) of the XV data vector. Third and fourth moments are the adimensional ones. This means that:

$$skewness = \frac{\sum (x-\overline{x})^3}{(N-1)*\sigma^3}$$

$$kurtosis = \frac{\sum (x-\overline{x})^4}{(N-1)*\sigma^4}$$

N beeing the number of data, $\overline{x}$ the mean value of XV and σ the standard deviation of residuals. STAT.OUT is a file with access=append,

so new data are appended to an existing file or a new one is created. The first line of any STAT.OUT file indicates what the following colums contain. No other written information is present in the output file: a new row with the above data is added any time XSTAT command is given.

See also: YSTAT

Example:

```
DATA file-name                    !  reads input data
XCOL 1                            !  assigns the column #1 to XV
XSTAT                             !  a new row of sta.out is added, for XV
XOPER 4 1                         !  decimal logarithm of XV
XSTAT                             !  a new row of sta.out is added, for new XV
YCOL 3                            !  assigns the column #3 to YV
YSTAT                             !  a new row of sta.out is added, for YV
YOPER 7 1                         !  gets YV(I)=YV(I)/YV(1)
YSTAT                             !  a new row of sta.out is added, for new YV
```

Bugs: none

XYSTAT **XYSTAT**

Format: XYSTAT

Parameters: none

Function: Computes and saves on the disk file STAT.OUT the number of data, mean value, standard deviation, variance, third moment(skewness) and forth moment (kurtosis) of the XV data vector if its probability (or frequency) distribution is known and contained into the vector YV. Third and fourth moments are the adimensional ones. This means that:

$$skewness = \frac{\sum (x_i - \overline{x})^3 * y_i}{\sigma^3(x)}$$

$$kurtosis = \frac{\sum (x_i - \overline{x})^4 * y_i}{\sigma^4(x)}$$

N beeing the number of data, $\overline{x}$ the mean value, σ the standard deviation of the XV vector and y_i the probabilities of x_i . STAT.OUT is a file with access=append, so new data are appended to an existing file or a new one is created. The first line of any STAT.OUT file indicates what the following colums contain. No other written information is present in the output file apart a new column where "XYSTAT" is shown: a new row with the above data is added any time XYSTAT command is given.

See also: XSTAT,YSTAT

Example:

```
DATA file-name                    !  reads input data
XCOL 1                            !  assigns the column #1 to XV
YCOL 3                            !  probabilities assigned to YV (3rd column)
```

```
XYSTAT                          !  a new row of sta.out is added, for XV
YOPER 7 1                       !  gets YV(I)=YV(I)/YV(1):  modify probabilities
XYSTAT                          !  a new row of sta.out is added, for new YV
```

Bugs: none

XZERO **XZERO**

Format: XZERO val or XZERO
Parameters: x user coordinate
Function: draws a full line at user coordinate x=val, from YMIN to YMAX (i.e. vertical line), with the last-defined color. Val is defaulted to zero. Use LTYPE command to change drawing color.
Bugs: A line shape different from *solid line* cannot be selected.

YAXIS **YAXIS**

Format: YAXIS axis-length
Parameters: y axis length in pixels
Function: Sets Y-axis length. If length overcomes y-dimension of actual graphic mode the plot is truncated.
Bugs: none

YCOLUMN **YCOLUMN**

Format: YCOLUMN n
Parameters: number of a column in the data file
Function: Assigns the column #n of the data file to Bongo internal vector YV. If LIMITS is not in effect, this command also computes max and min of the vector.
Bugs: none

YLABEL **YLABEL**

Format: YLABEL string
Parameters: string, a text of 40 characters max
Function: Writes string centered on the left side of the Y-axis of the plot. Angle has the value of 270 degrees
See also: BOX, LABEL, CLABEL, XLABEL
Bugs: This commando must be given **after** BOX

YOPER **YOPER**

Format: YOPER opcode, constant

Parameters: code of selected operation, constant required by operation (or 1 or 0 as necessary)

Function: Applies the operation defined by opcode to the YV internal vector.

1) vector+constant
2) vector*constant
3) vector/constant
4) LOG10(vector)*constant
5) LN(vector)*constant
6) vector**constant
7) vector/vector(constant)
8) vector-vector(constant)
9) 10**(vector*constant)
10) EXP(vector*constant)
11) vector/$\sum$(vector values)
12) vector to hh.mmss or dd.mmss
13) $\sum_1^i$ vector/$\sum_1^N$ vector

21) vector1(vector2*const)
22) vector1*(vector2*const)
23) vector1/(vector2*const)
24) vector1=vector2
25) vector2=vector1

Original data vector will be modified by ANY of the above operations. Modifications remain in effect until another YOPER is given or command is cancelled. Log of negative number is set to -50.

Codes 7 and 8 allow normalization with respect any element (set constant to the ordinal of the element) of the X vector.

Code 11 allows normalization with respect to the integral of the vector. The file SUM.OUT, containing the sum of the vector, is also created.

Code 12 assumes that data are expressed in terms of minor unit (e.g. in arcsec if degrees.minutesseconds are required)

Code 13 computes the empirical distribution function (cumulative distribution) of an early binned vector. The file SUM.OUT, containing the sum of the data vector is also created.

Codes above 20 allow operations between XV and EXV and/or between YV and EYV vectors. Results are copied into XV and YV vectors respectively (i.e. into vector1).

See also: EYOPER,EXOPER,XOPER

Example: only an example referred to 21-25 codes is presented. Two files must be read and three vectors extracted: one abscissa (common to both data vectors) and two ordinates (o1 and o2 from the files file1.dat and file2.dat, respectively). Ordinates must be arranged in order to give a new ordinate o3=log (o1)-log(o2). The command file is:

```
LIMITS x1 x2 y1 y2       !  limits in user coords
BOX                      !  draw axes and ticks
DATA [path]file2.dat     !  note:  first one is 2.nd file
LINES 12 200
XCOL 1                   !  load abscissa into XV
YCOL 2                   !  second ordinate o2 into YV
YOPER 4 1                !  log (o2)
YOPER 25 1               !  put log(o2) into EYV (i.e.  vect2=vect1)
DATA [path]file1.dat     !  read first file
LINES 12 200
```

```
YCOL 2                  !  first ordinate o1 into YV
YOPER 4 1               !  log (o1)
YOPER 21 -1             !  subtract:  YV=YV+(-1*EYV)
LTYPE 1 15              !  set white, solid line
CONNECT                 !  draw line
```

Bugs: none, but be sure that in code 23 vector2 is not a null vector.

YROW **YROW**

Format: YROW n
Parameters: number of a row in the data file
Function: Assigns the row #n of the data file to Bongo internal vector YV. If LIMITS is not in effect, this command also computes max and min of the vector.
Bugs: none

YSAVE **YSAVE**

Format: YSAVE
Parameters: none
Function: Saves the last-defined Y vector on the disk file SAVE.NN (NN=01, 02 ..., 10). This is a working file: SAVE command saves all SAVE.NN into the file SAVEnn.SAV, in the same order the working files have been originally written. Only numerical values are saved, so, if you wish to retain SAVEnn.SAV, add any comment you need (use DOS command to do that).
Example:

```
DATA file-name          !  reads input data
XCOL 1                  !  assigns the column #1 to XV
XOPER 4 1               !  decimal logarithm of XV
XSAVE                   !  writes logarithms to disk file SAVE.01
YCOL 3                  !  assigns the column #3 to YV
YOPER 7 1               !  gets YV(I)=YV(I)/YV(1)
YSAVE                   !  write normalized data to disk file SAVE.02
SAVE 3                  !  creates SAVE03.SAV with column taken from
                        !  SAVE.01 and SAVE.02 respectively
```

Bugs: sometime a zero-length file SAVE.NN is created. Delete the file and try againg PL.

YSTAT **YSTAT**

Format: YSTAT
Parameters: none

Function: Computes and saves on the disk file STAT.OUT the number of data, mean value, standard deviation, variance, third moment(skewness) and forth moment (kurtosis) of the YV data vector. Third and fourth moments are the adimensional ones. This means that:

$$skewness = \frac{\sum(y-\overline{y})^3}{(N-1)*\sigma^3}$$

$$kurtosis = \frac{\sum(y-\overline{y})^4}{(N-1)*\sigma^4}$$

N beeing the number of data, $\overline{y}$ the mean value of YV and σ the standard deviation of residuals. STAT.OUT is a file with access=append, so new data are appended to an existing file or a new one is created. The first line of any STAT.OUT file indicates what the following colums contain. No other written information is present in the output file: a new row with the above data is added any time YSTAT command is given.

See also: XSTAT

Example:

```
DATA file-name                    !  reads input data
XCOL 1                            !  assigns the column #1 to XV
XSTAT                             !  a new row of sta.out is added, for XV
XOPER 4 1                         !  decimal logarithm of XV
XSTAT                             !  a new row of sta.out is added, for new XV
YCOL 3                            !  assigns the column #3 to YV
YSTAT                             !  a new row of sta.out is added, for YV
YOPER 7 1                         !  gets YV(I)=YV(I)/YV(1)
YSTAT                             !  a new row of sta.out is added, for new YV
```

Bugs: none

YZERO **YZERO**

Format: YZERO val or YZERO

Parameters: y user coordinate

Function: draws a full line at user coordinate y=val, from XMIN to XMAX (i.e. a horizontal line), with the last-defined color. Val is defaulted to zero. Use LTYPE command to change drawing color.

Bugs: a line shape different from *solid line* cannot be selected.

WLINE **WLINE**

Format: WLINE line width

Parameters: line-width (real numbers, e.g. 0.5, 1.0, 1.5 ...)

Function: Defines the width of the drawing line. This value remains in in effect until another WL command is given.

Examples:

```
WLINE 0.5          !  the most common value
WL 1.5             !well visible lines or dots
```
Bugs: none

WRIMA **WRIMA**

Format: WRIMA
Parameters: none
Function: writes to the screen the image read by RIMA. The aim of this command is to verify modifications in the image.
Bugs: none

WRITE **WRITE**

Format: WRITE file-name
Parameters: [*path*]output file name
Function: Saves on the disk file *file-name* the actual command buffer. If file-name exists, it is overwritten.
Bugs: none

Appendix A. The file fig11.bon

```
COLOR 15
wl 0.5
DATA plotter.dat
LINES 16,76
LIMITS 0 1.5 -8,1
DIGITS 1 1
sticks 3 4
BOX 1 1
CLABEL foc f/96 PSF
XLABEL R(arcsec)
YLABEL \m (mag/arcsec^ 2)
GRID 4 8 2
XCOLUMN 1
YCOLUMN 2
PTYPE 9 4 15
LTYPE 1 15
POINTS
RELOCATE 0.8,-1
DOT
RELOCATE 0.85,-1
LABEL stars
YCOLUMN 3
LTYPE 1 15
CONNECT
RELOCATE 0.75 -2.5
DRAW 0.85 -2.5
RELOCATE 0.90 -2.5
LABEL 3 Gauss fit
YCOL 4
fill
PTYPE 4 3 15
POINTS
RELOCATE 0.8 -5
text 2 9
LABEL 3 GAUSS FIT
text 1 15
RELOCATE 1.32 0
LABEL (O-C)
RELOCATE 0.8 -1.7
DOT
nofill
RELOCATE 0.85 -1.7
LABEL squares
relocate 0.4 -10.4
```

`label Fig.11 - Output of FIG11.BON`

In general, a Bongo command-file can be divided in three areas: the first one defines the general behaviours of the plot (screen mode, box color, line width [PS only], physical and/or user defined limits, labels); the second area concerns data handling (data reading, math operations, definitions of symbols and lines color and shape); the third one refers to writing of comments or labels and user identification with date and time of the plot.

Appendix B. The file CMDS.BGO (73 commands)

Command	# Params	Description
BIN	2	bin data vector. Needs *nbin, bin width*
BOX	4	draw box. If *p1* or *p2*=0 no values
CIRCLE	3	draw a circle with radius *p1* from angle *p2* to *p3*, at current point
CLABEL	9	write a *comment*, 40 chars max, at top left
COLOR	1	set drawing *color*, or gray level in Postscript
CONNECT	0	connect data points by line LTYPE
CONTOUR	0	draw iso contours of the image by RIMA. Needs levels.dat
DATA	9	open and read *data file* as 80 chars strings
DELETE	2	delete lines *n1* to *n2*. Re-arrange the list.
DELTA	2	percent shift *dx, dy* of ticks respect to axes. Def. 0. 0.
DIGITS	2	set decimal digits for x and y axes. Default 2 2
DOT	0	draw at current point a symbol defined by ptype
DRAW	2	draw from current point to *X, Y*
ELLIPSE	5	draw an ellipse at current point: *p1*=s-axis;*p2*,a-ratio;*p3*=rot;*p4*=sangle;*p5*=ea
END	0	exit (from Ins or from Bongo) to OS
ERASE	0	clear graphic screen
EXCOL	2	assign to EXV column # *n* of data file. Drawing *flagged*
EXOPER	2	apply *opcode* and *constant* to EXV
EXSAVE	0	save on disk the last EXV vector
EYCOL	2	assign to EYV column # *n* of data file. Drawing *flagged*
EYOPER	2	apply *opcode* and *constant* to EYV
EYSAVE	0	save on disk the last EYV vector
FILL	0	set filling ability ON
FIT	2	polynomial fit (max *degree* = 6). *Weights* (1) or not (0)
GAUSS	2	copute gaussian given: *average and sigma*
GRID	3	draw grid: *pattern, color, flag* (=1 small ticks)
HELP	9	display help of a command (*name*); list commands (*none*)
HISTO	3	draw histogram with *bin width, color, flag*
IDENT	9	write *date, time, 2 chars user - id* on top right.
INFO	0	information about Bongo, References, Author's address.
INPUT	9	data input from keyboard. Write a *disk file*
INSERT	1	insert command AFTER line *#n* of command buffer
LABEL	9	write *text* at last cursor position
LIMITS	4	*limits of the plot* in user coordinates
LINES	2	*first* and *last* line of data vector
LIST	0	list the command buffer
LTICKS	2	number of *main (large) ticks* on x, y axis
LTYPE	2	set *line shape* and *color*
NOFILL	0	set filling OFF
OFF	0	close bongo.ps but not the bongo session. No logo
PAUSE	0	stop execution of command buffer. Hit a key to restart
PBOX	2	define *origin* of BOX in screen coordinates
PCOM	0	display data of Plotpe and Bongo common blocks
PHYSICAL	4	define *screen coords of box XS, XE, YS, YE*
PLAYBACK	0	execute actual command buffer
POINTS	0	draw points at current point and change it
PTYPE	3	define symbol: *shape, dimension, color*

QUIT	0	like END, but doesn't write the logo
READ	9	read *file* with command list
RELOCATE	2	set current point to *x, y* (user coords.)
RGB	1	set(1) or reset(0) RGB colors. Reset means gray scale
RIMA	9	read a real image(max 100x100)
SAVE	1	save on disk-file savenn.sav all available save.nn files
SMOOTH	1	smooth YV vector with window= *step*
SORT	0	sort XV and YV in ascending order
STAIRS	2	draw stairs-histogram with *color*, *flag*
STICKS	2	set small ticks *number* for X and Y axis
TEXT	3	set text dimension, color, angle or default values
VERSION	0	show version number
VWRITE	2	display on screen of the last XV and YV from *p1* to *p2*
XAXIS	1	length of x-axis (pixels)
XCOLUMN	1	assign *column #n* to internal vector XV
XLABEL	9	define x-axis *label*
XOPER	2	apply operation *# m*, with *constant c*, to xcol
XSAVE	0	save on disk the last XV vector
XSTAT	0	save on disk n, mean, std. dev., var, 3.rd and 4.th moment of XV
XYSTAT	0	in stat.out n,ave,var,skew,kurt of XV if P(XV)=YV known
XZERO	1	draw a line at Y=*value* from XMIN to XMAX
YAXIS	1	*length* of y-axis in pixels
YCOLUMN	1	assign column *#n* to internal vector YV
YLABEL	9	define y-axis *label*
YOPER	2	apply operation *# m*, with *constant c*, to ycol
YSAVE	0	save on disk-file the last YV vector
YSTAT	0	save on disk n, mean, std. dev., var, 3.rd and 4.th moment of YV
YZERO	1	draw a line at X=*value* from YMIN to YMAX
WLINE	1	define the drawing line width (real numbers)
WRIMA	0	write on screen the image read by RIMA
WRITE	9	write command buffer on *disk - file*

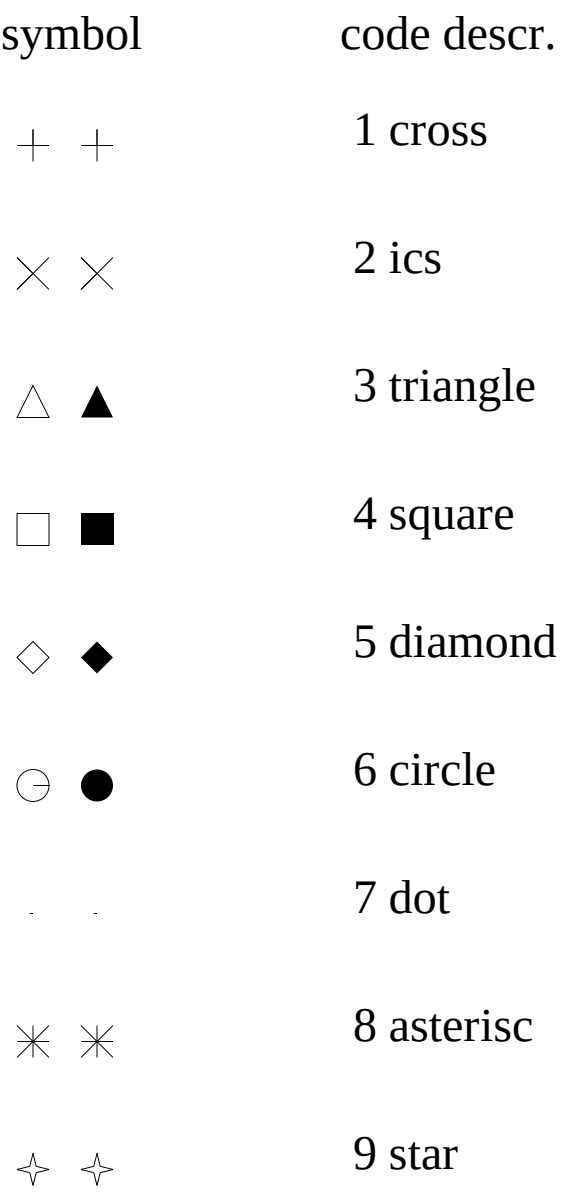

Fig. 9. 5-points symbols for Bongo.

ABCDEFGHIJKLMNOPQRSTUVWXYZ
ΑΒΧΔΕΦΓΗΙϑΚΛΜΝΟΠΘΡΣΤΥςΩΞΨΖ

abcdefghijklmnopqrstuvwxyz
αβχδεφγηιφκλμνοπθρστυϖωξψζ

Fig.10 - Correspondence between greek and latin alphabets

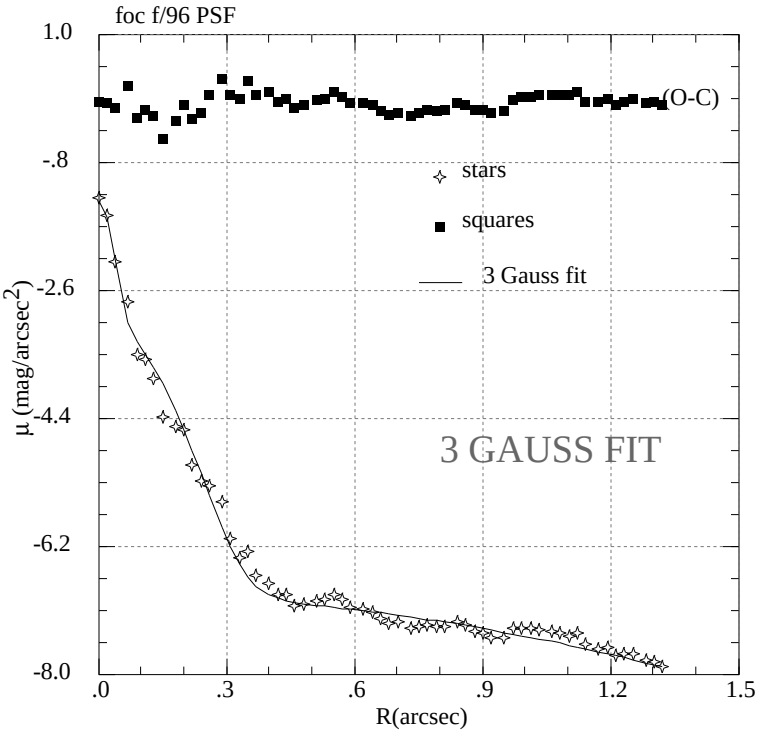

Fig.11 - Output of FIG11.BON

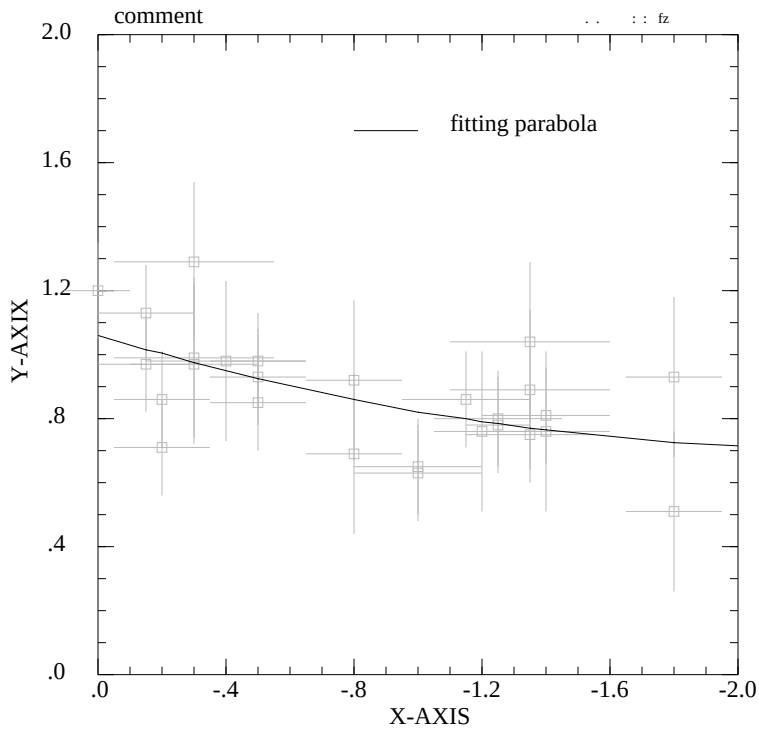

Fig.12 - Output of FIG12.BON

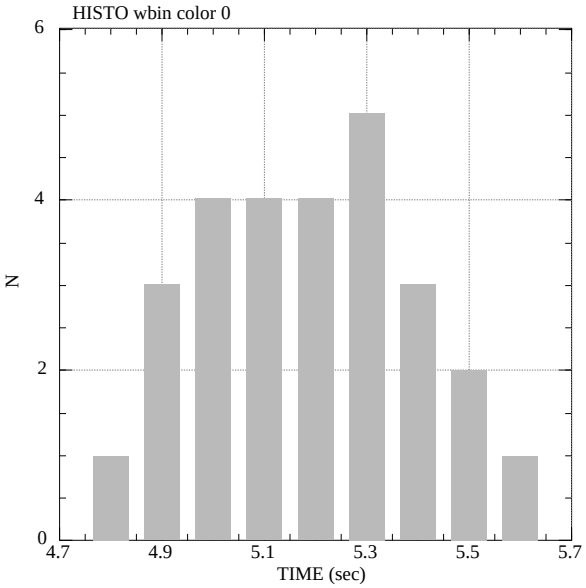

Fig.13 - HISTO with flag=0

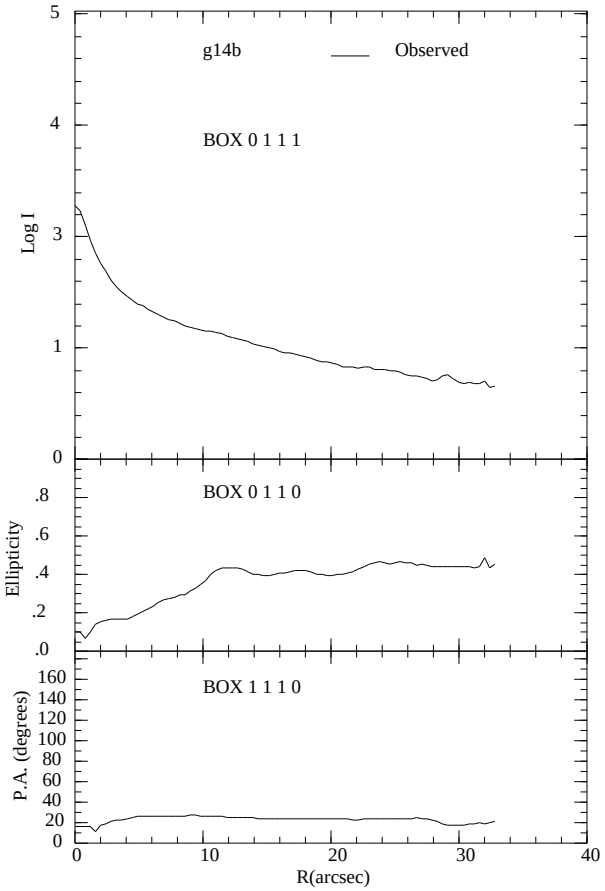

Fig.14 - Use of BOX p1 p2 p3 p4

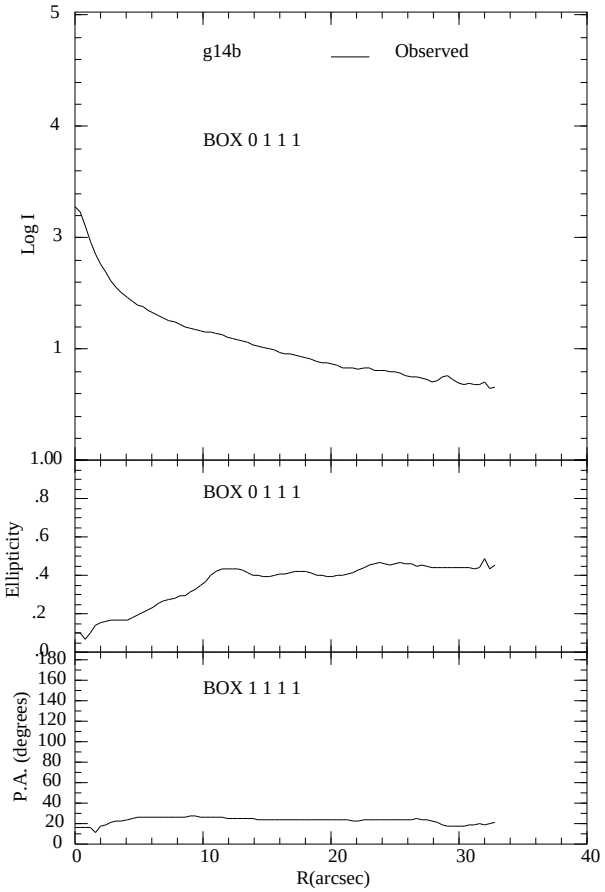

Fig.15 - Use of BOX p1 p2 p3 p4

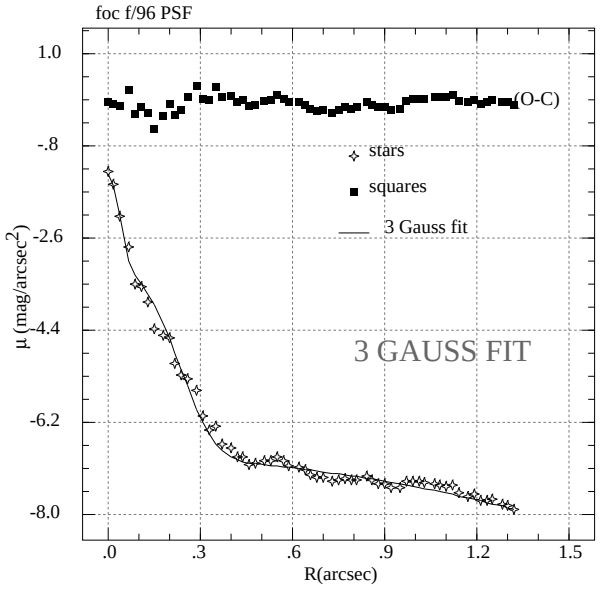

Fig.16 - Example of DELTA 5 5

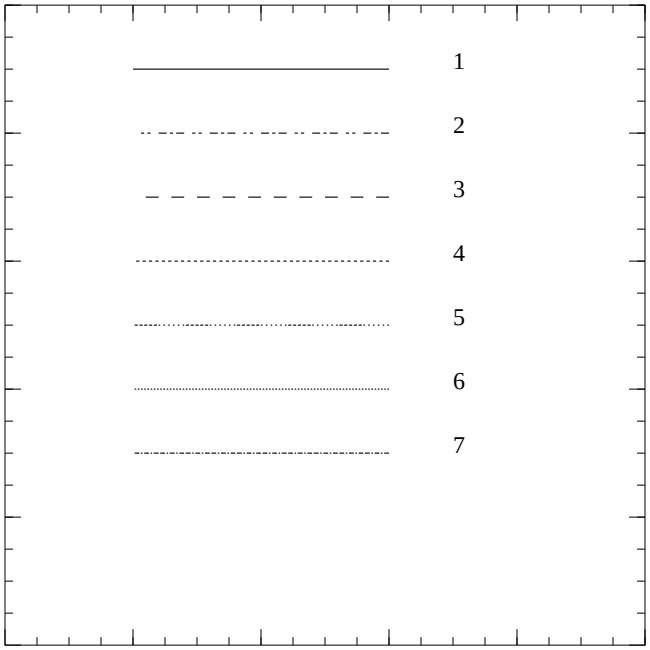

Fig. 17 - LTYPE: patterns and codes

Fig.18 - RGB/Gray colors and codes in Bongo

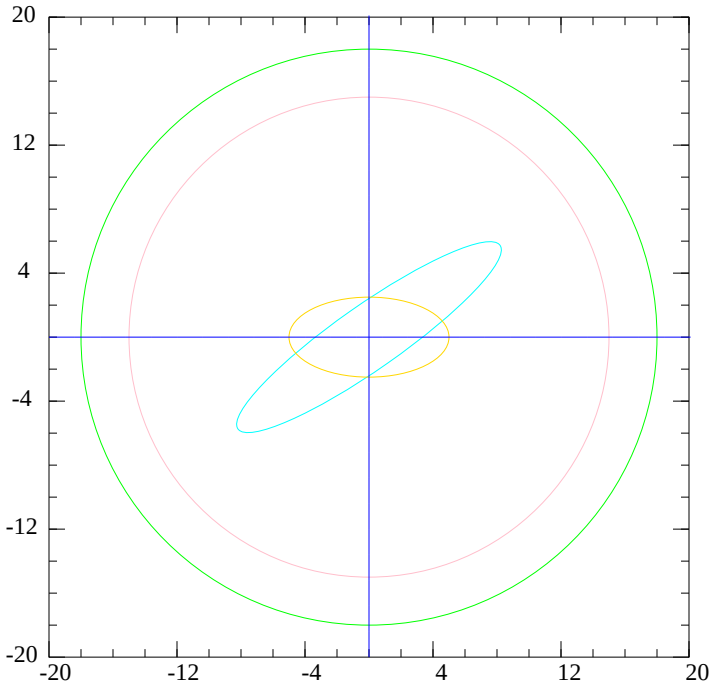

Fig.19 - Use of CIRCLE and ELLIPSE

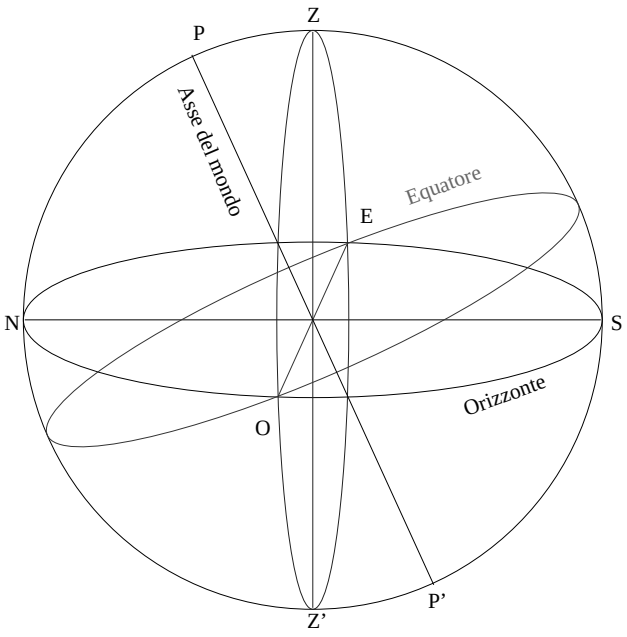

Fig. 20 - The celestial sphere