Franco Zavatti
Dipartimento di Astronomia, Universitá di Bologna

B O N G O

A general MS-DOS Plotting Program
Version 2.2
with PostScript Driver BONGOPS 1.2

User Manual

Technical Report
October 1995

1.INTRODUCTION

Bongo is 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 700 rows by 10 columns (the minimum one is 700 by 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 *hidden* 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 TEX file BONGOM2.TEX (i.e. this paper). If help command is entered without any parameter, a list af all available commands is displayed on the screen.

Presently, two characters font files are available: 16x9 and 9x6 pixels characters set, including greek letters and some special symbols not in the ascii code. The sets can be displayed on the screen by the Bongo command files font16.bon and font9.bon (see also Figs. 1 and 2). Text drawing directions can be seen in Fig. 3 or running the command file trot.bon. Bongo also handles text as alphanumeric strings: dimension, rotation and color connot be set in this case, but, on the other hand, the writing speed is high. This kind of text is reccomended when old 8088 or 8086 Intel CPUs are available. More recent chips, as the 80x86, without 80x87 co-processor, require alpha text. Graphic text can be used for final outputs, by means of the command TTEXT.

Working output on printers is committed to DOS Print Screen (version 4.1 or higher) or to commercial software. For high quality output the included PostScript driver BONGOPS.EXE can be used.

Two examples of Bongo style are shown in Figs. 4 and 5. The first one concerns multiple plots within one box, at VGA resolution (640x480x16), while the other example shows the use of error bars, at the same resolution. Plots of histograms, by using the available choices in the HISTO command, are in Figs. 6,7 and 8. Example of high resolution graphs are also included in the distribution diskette (file MULTI7.BON).

Bongo is available through 'ftp anonymous' on the node ASTBO3 (or 137.204.64.4) into the directory [anonymous.pub.msdos.bongo]. The included read.me file illustrates the way to write the installation diskette (Bongo can be installed only from diskette).

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

Franco Zavatti
Dipartimento di Astronomia
Via Zamboni 33
40126 BOLOGNA (Italy)

`Internet:` zavatti@astbo4.bo.astro.it
`or` zavatti@tikal.bo.astro.it

2. INSTALLATION

Bongo can be installed by putting the distribution diskette into drive A:, setting the computer actual disk also to A: and typing:

A: > `INSTALL disk <return>`

to install Bongo into the hard disk *disk*, where *disk* may be any of C:, D:,···, Z.

The batch program INSTALL.BAT provides the creation of both the directory \BONGO\, containing the main program and related files (see Appendix C), and the subdirectory \BONGO\BGOHLP\ with the (on line) help files.

After the files have been copied, Install.bat asks to enter again the name of the disk where Bongo will be installed. A list of actually available VGA modes is then shown, and user proposed to give his/her own choice by typing the (decimal) code of the preferred graphical environment. Install.bat uses as default code, 18 decimal (12 Hex), i.e. the standard VGA mode, with a resolution of 640 rows by 480 columns and 16 colors. Any VGA card knows this mode and the lower ones, while the higher modes (called Super VGA or SVGA) are tipical of each manufacturer. So 18D is the better starting mode and its choice strongly recommended. To accept the default choice a *comma* can be typed, followed by <return>: Fortran allows such a way to confirm formerly given numerical value(s) (18 <return> can be also typed, of course).

the user is then asked to confirm or change the [path] filename of the default font file.

As last operation, the default text type must be given (or the available choice confirmed). Hit A (or a) for alphanumeric text, G (or g) for the graphic one. Some more words about SVGA codes: they are often listed in the VGA reference manual as a table of hexadecimal digits with related resolution, colors and text characters dimension. To change actually available SVGA codes, the file MODIVGA.PLT should be edited, referring to the above mentioned table, as far as the first column (and also the last one) is concerned. Also the help file \bongo\bgohlp\ mode.hlp should be updated in the same way.

BCONFIG.PLT is then written with the actual VGA code and font file, and both .PLT files (i.e. Bconfig and Modivga) are copied to the root of the chosen hard disk.

The file DSK.INI , containing the name of the hard disk where Bongo has been installed, is also created. This file **must** be copied to the directory Bongo is called from and modified any time Bongo directories are copied to another disk (only when copied, because Install provides by itself the creation of a correct DSK.INI).

The command `DEVICE =` *path* `Ansi.sys`, or any Ansi extension required by the actual VGA (e.g. Eansi.sys for the ET4000 chip) should be included in the Config.sys file.

3. OPERATIONS

To start Bongo from within its own directory \Bongo\, type `bongo <ret>`. Running from another directory, Bongo path should be known, either adding it to the PATH command in `autoexec.bat` or by a batch file (say bongo.bat) containing the following lines:

```
COPY c:\bongo\dsk.ini
c:\bongo\bongo,
```

where c: is the disk where Bongo has been installed.

This also automatically copies dsk.ini in the working directory.

At the start, Bongo clears the screen (no welcome or copywrite message appears), looks for the input file bongo.inp and execute instruction therein. If bongo.inp does not exist, Bongo shows the prompt '>' and waits for a command.

If the actual session is an interactive one, any command is executed just after <ret>. It should be remembered that interactive mode does not enter `MODE n` automatically (the default is `MODE 3`, i.e. alphanumeric screen), and then commands like BOX or CONNECT do not show anything. The ways to prepare and/or control a command file are:

a- write commands in alphanumeric mode and then give PL(ayback) command.

b- enter `MODE n` command and then give the commands, controlling their execution. Give PL to execute the command buffer again and show a clean plot.

In the other possible situation, i.e. when a command file has been already 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`.

4. POSTSCRIPT DRIVER

Bongo version 2.1 includes an off-line PostScript driver in the form of the executable program BONGOPS.EXE version 1.1 (June 1993).

Bongops works like Bongo: it reads a command file (previously tested by Bongo), can include, delete or edit command lines. Use PL(ayback) to start driver execution. Some information appears after PL and, as last output, the prompt '>' informs the user that the command END must be given. After the command END, the message *BONGO.PS* *has been written on disk* appears and Bongops stops. Now the copy of bongo.ps on the PostScript (laser) printer generates the final product.

Examples of BONGOPS output are shown in Figs.9 – 13.

User should remember that fine colors on the screen may give unpleasant gray tones on the printer.

Character dimension can be changed in Bongops 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).

Some problem is actually present when high resolution (800x600 and 1024x768) plots must be scaled within the 500x800 Postscript page. No automatic scaling is available, so user should modify the parameters of the command PHYSICAL.

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. A difference holds in this case between BONGO and BONGOPS, but it will hopefully be cancelled in the next version.

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.

5. BUGS

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

A bug, not reported in the help files, concerns the incapacity of correctly drawing lines at higher resolutions, with some SVGA card (e.g. with the Trident TR4000, but the drawing routines were originally developped on a TR3000, i.e. on practically the same chip, and there worked well, of course).

Bongo is lacking about several capabilities which can be considered important tools in a general context, like selection of orizontal or vertical grid; MACRO command, to run subroutines; conditional jumps and loops within command buffer (i.e. IF - ENDIF and DO - ENDDO); setting of user defined variables (something like SET command); also, a maximum of 700 data can be not sufficent for very many applications. Improvements in that sense (or, at least, for a part of above points) will be introduced in the next versions.

6. DIFFERENCES FROM EARLY VERSIONS

`from version 1.4`

- – New font file for small 9x6 pixels characters.
- – Real possibility of changing font files. Vers. 1.4 included the command FONT, but the gap between two letters was set to the fixed value of 9 pixels.
- – On-fly capability to select between quick-and-worse alphanumeric and slow-and-better graphic text (TTEXT command).
- – FIT includes now weights and errors on the parameters, and the capability to collect ALL outputs of a series of fits.

- – Histogram representation can be chosen among three modes: 0 (default) gives the normal drawing with the minimum set to the y lower limit of the box; 1 allows to shift histograms (minimum is set to data minimum); 2 draws histograms where bins have both positive and negative values.
- – Operations include normalization to the sum of YV data. This was intended mainly to normalize statistical distributions, but may be useful in many cases.
- – LIST command allows definition of the first and last lines. Listing of long buffers stops every 22 lines and user can exit from this command by ESC key or continue by any other key.

`from version 1.5`

- – New command SMOOTH to filter YV data.
- – A bug in the ZERO command has been corrected
- – Greek alphabet includes the previously forgotten η and λ

`from version 1.6`

- – the decimal digits of the numbers on both x and y axes are user defined (default 2,2)
- – Symbol #6 (circle) can be filled
- – A user manual written in plain ascii text is available (BONGO2.DOC)

`from version 1.7`

- – A Postscript driver, the external program BONGOPS.EXE, is available
- – The COLOR command has been introduced for an overall definition of drawing color.
- – Text strings are allowed to have greek letters and both exponents and deponents (by means of the special characters \,^ ,_). All this applies only to the postscript driver Bongops.
- – The new WLINE command allows to define the line width. It works only for Bongops and is a dummy for Bongo.
- – The HARDCOPY command has been cancelled

`from version 2.0`

- – Operations between two internal vectors are now available. Only operations between XV and EXV (or YV and EYV) internal vectors are allowed so the following techique should be followed:
  1) Load the second vector into XV (or YV) and perform the needed operations, if any. Then resulting vector will be stored into XV (YV). Copy XV (YV) into EXV (EYV) (code 25).
  2) Load the first vector into XV (or YV) and perform the needed operations. Result will be into XV (YV). Do then the prescribed operation using an operation code larger than 20.
- – BOX command now controls the presence or not of the last numerical label on both axes. Useful for adiacent plots.

bullet – XZERO and YZERO commands have been added to draw both Y-and-X-axis parallel lines, respectively. ZERO command has been cancelled.

bullet – STAIRS command has been added in order to draw stairs-like histograms. This new command does not require wbin as parameter. Bin width is computed by difference between the 2nd and the 1st x coordinate of the data file.

`from version 2.1`

bullet – Bongo can now read a command file (i.e. bongo.inp) which contains the commands normally given by hand, like

READ file-name.bon
PLAY

In such a way a .bat file can be prepared, in order e.g. to plot a defined sequence of graphs from within a program.

7. REFERENCES

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

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

Wilton Richard, *Schede Grafiche*, ed. Gruppo Editoriale Jackson, Milano, 1989

PostScript Language Reference Manual, Addison-Wesley, 1985

PostScript Language Tutorial and Cookbook, Addison-Wesley, 1985

PostScript Language Program Design, Addison-Wesley, 1985

8. COMMANDS

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

1) Parameters may be separated by either comma or **one** space.
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.

ALPHA

ALPHA

Format: ALPHA
Parameters: none
Function: Sets video to alphanumeric mode. As MODE 3, but doesn't enter in the command buffer. Use this command before EDIT a command line.
Bugs: none

BCOLOR

BCOLOR

Format: BCOLOR ncol
Parameters: ncol, color code (0-15)
Function: Fills area with color ncol. Filling starts from the last defined current point. BCOLor should be used as first command.
Bugs: it works only for standard VGA modes, i.e. up to 640x480x16.

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

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 set to 1 1. The use of only one parameters is not allowed.Also p3, p4 are flags: if one of them is zero, the last numerical value on the corresponding axis is not written.
Function: Draws a (labelled) box according to data limits. If LIMITS hasn't been given, XCOL and YCOL are needed before BOX, in order to compute data limits. **Box neither erase the screen, nor enter 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, where n is the default mode as appears in BCONFIG.PLT. The values of *p1* and/or *p2* are zero when multiple, joined plots are drawn. See multi7.bon as an example of such an application at resolution of 1024x768.
Bugs: none

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: LABEL, XLABEL, YLABEL
Bugs: none

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

CONNECT

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

CURSOR

Format: CURSOR
Parameters: none
Function: Sets graphic cursor to ON. Use arrows to move, also along diagonals. Hit numbers (on the keyboard first line) to change speed. For max speed (20) hit ' (ascii 39). X and Y position (pixels) continuously appears on the upper right corner of the screen; user coordinates appear after pressing ESC key. The next ESC hitting produces the exit from the routine.
Bugs: none, but no control is made on actual screen physical limits.

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 80 characters length max data columns and comments. Data are assumed to be organized in columns, separated by space(s) or comma(s). If *file-name* is ?, Bongo asks for file-name from the keyboard. If the data file contains one or more non-numeric columns, these columns are translated as 9999. if ALL rows are non-numeric; if one or more rows contain either blank or numeric value, an error holds.
Bugs: none

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

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

DOS

Format: DOS
Parameters: none
Function: Opens a child process which allows one DOS commands a time. If the first DOS command is the word COMMAND, several commands can be given; EXIT allows to leave child process and return to Bongo.
Bugs: none, but a child process requires enough memory to run a copy of COMMAND.COM.

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

EDIT EDIT

Format: EDIT Line-number
Parameters: line-number; the line number,in the command buffer, to be edited
Function: Edit a line. Very, very raw editor. Use right arrow to set the cursor over the character to be changed. Space bar writes blanks (ascii 32). Use left arrow to go backward. Enter terminates the editing.
Bugs: left arrow doesn't work. In practice, carefully avoid the use of this key. Re-enter EDIT command if you are wrong.

END END

Format: END
Parameters: none
Function: Exit from Bongo to DOS. 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 write on the screen the message "Bongo.ps written on disk".
Bugs: none

ERASE ERASE

Format: ERASE
Parameters: none
Function: Clears the actual graphic screen.
Bugs: none

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

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

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

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

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

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

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

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 provides 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 deg+1 parameters of the fitting function with the respective errors and a table with an ordinal, the original abscissae and the computed ordinates (not the original ones). POLY.OUT is overwritten by the next FIT n command. If needed, change its name by DOS 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

FONT

Format: FONT font-file name

Parameters: path and file name of a new font file

Function: makes available another (existing) font file. Text is written with actual font style, until a FONT command is given. Actually available fonts are shown in Figs. 1 and 2. The name of the font files, like pxp.fnt, indicates the dimension of character matrix, as rows by columns.

Bugs: none

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

Format: (1) HELP or (2) HELP command-name

Parameters: none or the name of a command for detailed help.

Function: (1) lists all commands available 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

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

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 *dd-mm-yyyy*. If two characters 'cc' are given, they are added to the above string.

Bugs: none

INFO

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

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.
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

MODE MODE

Format: MODE n
Parameters: decimal value of the VGA emulation code.
Function: Sets VGA video card to one of the modes listed in the file MODIVGA.PLT. This command works well also with the indigenous Olivetti or AT&T 6300 and the Olivetti EGC 16 colors card. MODIVGA.PLT can be edited in order to add your own graphic card or mode. The main VGA modes (decimal) are listed below:

Mode	Rows	Cols	Physical	Window	Char.h.	Maxcol	Description
3							TEXT
4	320	200	100 300	50 170	8	4	CGA

5	320	200	100	300	50	170	8	4	CGA
6	640	200	100	600	50	170	8	4	CGA
13	320	200	100	300	50	170	8	16	EGA
14	640	200	100	600	50	170	8	16	EGA
15	640	350	100	500	50	290	14	2	EGA MONO
16	640	350	100	600	50	290	14	16	EGA
17	640	480	100	600	130	430	16	2	VGA MONO
18	640	480	100	600	130	430	16	16	VGA default
19	320	200	100	300	50	170	8	256	VGA
64	640	400	100	600	100	350	16	2	OLIVETTI
41	800	600	100	500	100	500	8	16	VGA ET4000
42	640	400	100	500	100	350	16	256	VGA ET4000
96	640	480	100	500	100	430	16	256	VGA ET4000
55	1024	768	100	650	100	650	16	16	VGA ET4000
56	1024	768	100	650	100	650	16	256	VGA ET4000
222	650	800	150	550	150	550	16	10	POSTSCRIPT

Bugs: none

NOFILL

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

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 1.2*' 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

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

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

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 1.2*' 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

RESET RESET

Format: RESET
Parameters: none
Function: Resets Bongo graphics to default parameters, as listed in MODIVGA.-PLT and defined by CONFIG.PLT. Clears the screen. This command is added to the command buffer so that, at any PL command, it will be executed again.
Bugs: none

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

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

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.

see also: BIN, HISTO Bugs: filling of bins exceeding box limits doesn't work

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

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*. See Fig. 2 or run the file trot.bon. When Postscript driver runs, dim=0 means 9-pixel characters; dim=1 means 15-pixel characters; dim=2 means 24-pixel characters. Bongops is insensitive of the FONT command: the selection of character dimension is made only by TEXT. In order to save the compatibility with Bongo, within the latter one dim=0 is the same of dim=1. 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

TTEXT

Format: TTEXT tsel

Parameters: text type is A (or a) and G (or g)

Function: Defines if text will be written as alphanumeric strings or drawn as graphic symbols. The first mode is very quick but rotation, dimension and color changes are not allowed. Also, it is difficult to compute a correct positioning, due to transformation of pixel positions into row and column. Alpha mode is suggested as default in PCs without mathematical coprocessor, or equipped by the old 8088 or 8086 CPUs. The graphic mode is the preferred one when 80386 + 80387 or 80486DX are available. This mode is of course allowed also with slower machines, but now most of time is spent waiting the drawing of digits and labels.

Example:

```
TTEXT A !turns the text to alphanumeric
TTEXT G !turns the text to graphic
```

Bugs: none, but ylabel is positioned as graphic text requires (see e.g. TEST.BON with TTEXT A).

VWRITE

Format: VWRITE

Parameters: none

Function: Vectors WRITE. Display on the screen the last-defined XV and YV internal vectors.

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

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.
Bugs: none
See also: LABEL, CLABEL, FONT, TEXT, YLABEL, TTEXT

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
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) $\sum_1^i$ 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 distribution function (cumulative distribution) of an already 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.

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

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

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

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

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
Bugs: none.
See also: LABEL, CLABEL, XLABEL

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	7) vector/vector(constant)
2) vector*constant	8) vector-vector(constant)
3) vector/constant	9) 10**(vector*constant)
4) LOG10(vector)*constant	10) EXP(vector*constant)
5) LN(vector)*constant	11) vector/$\sum$(vector values)
6) vector**constant	12) vector to hh.mmss or dd.mmss
	13) $\sum_1^i$ vector/$\sum_1^N$ vector
21) vector1(vector2*const)	24) vector1=vector2
22) vector1*(vector2*const)	25) vector2=vector1

23) vector1/(vector2*const)

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 distribution function (cumulative distribution) of an already 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.

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

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, $\overline{x}$ 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

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

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

ZERO

Format: ZERO val or ZERO
Parameters: y user coordinate
Function: draws a full line at user coordinate y=val, from XMIN to XMAX, with the last-defined color. Val is defaulted to zero. Use LTYPE command to change drawing color.
Bugs: actually the last-defined pattern cannot be drawn.

Appendix A. The file TEST.BON

```
COLOR 15
WLINE 0.5
DIGITS 1 1
LIMITS 0 1.5 -8 1
BOX 1 1
CLABEL foc f/96
XLABEL R(arcsec)
grid 4 8 2
YLABEL μ (mag/arcsec²)

DATA PLOTTER.DAT
LINES 16,76
XCOLUMN 1
YCOLUMN 2

FILL
PTYPE 6 5 4
LTYPE 0 0
POINTS
RELOCATE 0.8 -1
DOT
NOFILL
RELOCATE 0.85 -1
LABEL circles
YCOLUMN 3
LTYPE 1 2
CONNECT
RELOCATE 0.75 -2.5
DRAW 0.85 -2.5
RELOCATE 0.90 -2.5
LABEL 3 Gauss fit
YCOL 4
LTYPE 0 0
PTYPE 1 2 11
POINTS
RELOCATE 0.7 -5
TEXT 2 14 0
LABEL 3 GAUSS FIT
TEXT 1 15 0
RELOCATE 1.32 0
LABEL (O-C)
RELOCATE 0.8 -1.72
DOT
RELOCATE 0.85 -1.7
LABEL crosses
```

```
RELOCATE 0.4 -11.4
LABEL Fig.4 - Output of TEST.BON
ID fz
```

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
ALPHA	0	enter alpha mode. Like MODE 3.
BCOLOR	1	fill screen with color *ncol* [0 - 15]
BIN	2	bin data vector. Needs *nbin, bin width*
BOX	4	draw box. If *p1* or *p2*=0 no values
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
CURSOR	0	set cursor on.
DATA	9	open and read *data file* as 80 chars strings
DELETE	2	delete lines *n1* to *n2*. Re-arrange the list.
DIGITS	2	set decimal digits for x and y axes. Default 2 2
DOS	0	allow DOS command(s). Needs enough memory
DOT	0	draw at current point a symbol defined by ptype
DRAW	2	draw from current point to *X, Y*
EDIT	1	edit the command line # *n* (->, space ,<RET>)
END	0	exit (from Ins or from Bongo) to DOS
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)
FONT	9	load new font file
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*
MODE	1	set graphic mode to *n*
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.)
RESET	0	set parameters to original values as in bconfig.plt
SAVE	1	save on disk-file savenn.sav all available save.nn files
SMOOTH	1	smooth YV vector with window= *step*
STAIRS	2	draw histogram of binned data. *color, flag*
STICKS	2	set small ticks *number* for X and Y axis
TEXT	3	set text dimension, color, angle or default values
TTEXT	9	set text type at alpha (A) or graphic (G) mode
VWRITE	2	display on screen of the last XV, YV, EXV, EYV 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
XZERO	1	draw a line at X=*value* from YMIN to YMAX
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 Y=*value* from XMIN to XMAX
WLINE	1	define the drawing line width (real numbers)
WRITE	9	write command buffer on *disk - file*