ASGL, Ver 1.3.2

PostScript Plotting Program

Andrej Šali

Mission Bay Byers Hall
1700 4th Street, Suite 503B
University of California, San Francisco
San Francisco, CA 94158-2330, USA
tel +1-415-514-4227, fax +1-415-514-4231
email sali@salilab.org
URL http://salilab.org/

November 9th, 2006

Contents

Part I

User's Guide

Chapter 1

Introduction

1.1 What is Asgl?

Purpose: Easy creation of PostScript files containing scientific plots.

Capabilities:

- 2D plots: scatter plots, curve plots, smoothed curve plots, transformation of axes, histograms, spectra, general axes, and plot labelling;
- 3D plots: density plots;
- molecule plots: ball-and-stick plots of a molecule in a Brookhaven Protein Databank format file; general labelling; virtual Calpha-Calpha bonds; stereo plots;
- the attributes of the objects to be drawn can be varied by the user; for example, the grayness and width of lines, the size of fonts, the grayness of histogram bars, *etc.*

Use:

Asgl is used via a steering file. A steering file contains commands specialized for creation of plots. This file is interpreted by Top which then calls appropriate Fortran subroutines to create the PostScript output file. See Chapter 4 for description of Top.

1.2 Using Asgl with the Top steering file

The Asgl program is run by:

```
asgl job
```

where `job` is the root of the steering file — the actual steering file must have an extension `.top`. By default, the output of the program is then written to the `job.ps` file that can be printed on the PostScript printer, previewed with a PostScript previewer, modified, or included in other documents, such as TeX files.

Asgl reads PostScript definitions of line types, font types and symbols from text file `src/psgl1.ini`. This file can be edited to extend the capabilities of Asgl. For example, line widths, grayness, dash-dot combinations, font types and sizes, symbol types and sizes can be customized in this way.

There are four coordinate systems (or windows) used by Asgl. The Base PostScript system corresponds to the PostScript device and has the Bounding Box (0,0,612,792). The Paper coordinate system is used to specify the Bounding Box and its orientation of the window on the paper that will contain the plot. It overlaps with the Base PostScript system except that it uses cm; its Bounding Box is (0,0,21.59,27.94). The World coordinate system is defined by the data to be plotted. It overlaps with the window specified in the Paper coordinate system. The Plot coordinate system is used for creating the plot. The origin of the Plot coordinate system is always (0,0), XMAX is always 1, and YMAX is such that it retains the X/Y aspect of the Paper coordinate system. It overlaps with the window specified in the Paper and World coordinate systems.

Probably the easiest way to use Asgl is to take an example Top file from `examples` sub-directory that is closest to what you need and edit it to suit your requirements exactly. The example Top files and their output are given in Chapter 3.

A complete list of Asgl commands is given in Chapter 2.

1.3 ASGL installation

```
                               INSTALLATION


                                ASGL 1.3.2

                           PostScript Plotting
                                       v
                    Copyright(c) 1989-2006 Andrej Sali
                            All Rights Reserved





Supported computers:

ASGL 1.3.2 runs on Silicon Graphics Iris 4D, Convex C2, Sun 4,
IBM RS/6000, DEC Decstation, DEC Alpha station, and Linux boxes.


Installation:


- If you have asgl-1.3.2-1.src.rpm, and are on a RedHat Linux (or similar)
  system, the following should generate a binary RPM from the source:
     rpmbuild --rebuild asgl-1.3.2-1.src.rpm
  (You may have to run this as root.)


- If you have asgl-1.3.2-1.i386.rpm (e.g. built from the .src.rpm, above)
  then the following should install it (run as root):
     rpm -ivh asgl-1.3.2-1.i386.rpm


- If you have the asgl-1.3.2.tar.gz distribution file:

  1) Unpack the distribution file:

     zcat asgl-1.3.2.tar.gz | tar xvf -

     The result will be a directory ./asgl-1.3.2


  2) Make sure the ASGL_EXECUTABLE_TYPE environment variable is defined
     for your architecture (see scripts/Makefile.include1 for a list):

          for sh :  ASGL_EXECUTABLE_TYPE=gfortran; export ASGL_EXECUTABLE_TYPE
```

```
     for csh:  setenv ASGL_EXECUTABLE_TYPE gfortran


    (These commands could also be put in your login script.)


3) Set the environment variable ASGLINSTALL to the directory where you
   want to have ASGL installed. Create this directory. For example,
   for the tcsh or csh shells:

   setenv ASGLINSTALL /usr/local/bin
   mkdir $ASGLINSTALL


4) Compile and install the program:

   make all; make install

   If you are not using one of the compilers listed above
   you will probably have to modify the Makefile in the
   src/ sub-directory. Hopefully, you won't have to modify
   the fortran code.


5) Change your login script to include the following (for csh or tcsh):

   # Root directory for installed ASGL:
   setenv ASGLINSTALL /usr/local/bin/asgl

   # ASGL executable type
   setenv ASGL_EXECUTABLE_TYPE gfortran

   # Set ASGL environment variables and update the command path:
   if (-e $ASGLINSTALL/setasgl) source $ASGLINSTALL/setasgl


6) Source your login script. You can now start using ASGL. See
   the examples in the examples/ directory. You may (but do not
   have to) delete the ASGL distribution directory.


7) To make and install the documentation, once ASGL is installed,
   run the following in the ASGL distribution directory:

   make installman

   In order to create the document, 'collect' must be built from
   'collect.f' in the doc/ directory. You may have to do this
```

```
   manually if the Makefile is unable to do it automatically
   (it uses 'f77 collect.f -o collect').


8) The ‘hosttype mechanism’ allows transparent ASGL installation
   and use of the same installation directory on some networks of
   different host types. For example, at Harvard we have an NFS
   and Yellow Pages network with only one home directory per user
   for almost all the machine types listed above. In such a case,
   to install ASGL on another host type in the same directory,
   you only have to login to that host, clean the distribution
   directory by ‘make distclean’ and do ‘make all’ again. Exactly
   the same commands (scripts) are used on all hosts to run the
   ASGL programs.
```

Chapter 2

Using Asgl

2.1 Asgl commands

2.1.1 #EPSF — produce encapsulated PostScript

Command: #EPSF [.ps | .eps]

Description: If the very first line in the steering file starts with the above in the first column, Asgl will create a PS file that conforms to the Encapsulated PostScript standard. In this case, only one page graphics per file can be produced. Optionally, you can also specify the extension for the PS filename. It is usually `.eps` for the Encapsulated PostScript, but you may want to keep it as `.ps` if you depend on it in some other programs. The default is `.ps`. You can only specify the extension when **#EPSF** is the first line. If you produce an EPSF file you can include it properly in the LaTeX document using:

2.1.2 SET STAMP_TEXT — stamp the page

Command: SET STAMP_TEXT = ⟨`string` : 1⟩ [`DEFAULT` | `NONE` `your text`], STAMP_SIZE = ⟨`real` : 1⟩

Description: If set to `DEFAULT` the page will be stamped at the bottom right corner with the Asgl version, date, time, filename, and page number. The default value is `DEFAULT` for PostScript and `NONE` for Encapsulated PostScript. The command is not allowed in Encapsulated PostScript.

2.1.3 READ_TABLE — read the Table array of data

Options:

FILE = ⟨`string` : 1⟩	`'counter'`	partial or complete filename
ADD_DATA = ⟨`logical` : 2⟩	`OFF OFF`	
POINT_MODULUS = ⟨`integer` : 1⟩	`1`	
ROW_RANGE = ⟨`integer` : 2⟩	`0 0`	

Description: This command will read a text file containing a rectangular array of values. The columns represent vectors that can be later used for plotting as X and Y coordinates. This array will be referred to as the Table array. If ADD_DATA[1] = `ON` new columns will be appended to the right of the existing columns. The new number of data points is the number of rows in the new file. If ADD_DATA[2] = `ON` new rows will be appended to the end of the existing rows. The new number of columns is the number of columns in the new file. The command will only read the points whose index satisfies $\mathrm{mod}(i-1, \mathsf{POINT\_MODULUS}) = 0$ and is within boundaries of ROW_RANGE. ROW_RANGE can be 0 or -999 to indicate that it is to be ignored. The total number of points read is approximately for a factor of POINT_MODULUS smaller than the total number of points between selected rows in a file. This option is useful for plotting very large files. The line can start with '#', in which case it will be ignored as far as the points are concerned. There can be string columns in the file that can contain data for point labelling by **PLOT2D**. Number/string column identification is achieved in one of the two ways: If there is a comment line '#COLUMNS' before any non-comment line, then that line is used for identification. The comment line must contain a series of 'N' for number columns and 'S' for string columns, in the correct order. Otherwise, the items from the first non-comment line are interpreted as numbers or strings. A string is everything that cannot be interpreted as a number. If you want blanks in strings, quote them in single quotes '. The last line starting with '#CAPTION_TEXT' is assigned to CAPTION_TEXT, which is useful for automated production of titles. Similarly for '#PRINT_TEXT' and '#DESCRIPTION'.

2.1.4 WRITE_TABLE — write the Table array of data

Options:

FILE = ⟨`string` : 1⟩	`'counter'`	partial or complete filename
NO_XY_SCOLUMNS = ⟨`integer` : 2⟩	`0 0`	
XY_SCOLUMNS = ⟨`integer` : 0⟩		

Description: It will write a text file containing an array of values, where columns are selected from the Table array using the NO_XY_SCOLUMNS and XY_SCOLUMNS variables. See **WORLD** for explanation of these variables.

2.1.5 READ_DPLOT — read the Density array of data

Options:

FILE = ⟨`string` : 1⟩	`'counter'`	partial or complete filename
DPLOT_FORMAT = ⟨`logical` : 1⟩	`OFF`	
DPLOT_COLUMN = ⟨`integer` : 1⟩	`1`	
DPLOT_SYMMETRIZE = ⟨`logical` : 1⟩	`OFF`	

DPLOT_FILL = ⟨real : 1⟩	0	
DPLOT_ORIENTATION = ⟨string : 1⟩	'XY'	YX

Description: It will read a text file containing the Density array of data. Two formats are possible. In free format (DPLOT_FORMAT=OFF), a text file must contain two integer dimensions (NX,NY) followed by data NX * NY values. The first line can contain NX and NY, at most. The following lines can contain a single row of an array, at most. The order of reading the elements is:

```
read(ioinp,*)NX,NY
do i = 1, NX
read(ioinp,*) (array(i,j),j=1,NY)
end do
```

This array can then be plotted using **DPLOT** command. The array(1,1) element will appear at the coordinate system origin, in the lower left corner of the coordinate system. When formatted input is used (DPLOT_FORMAT=ON), a text file contains any number of lines in the form:

I, J, number_1 number_2, ..., number_DPLOT_COLUMN, ...

These lines are read and the Density array is filled in on the fly as specified by the I and J indices and the value in column DPLOT_COLUMN. All other numbers in a line are ignored. The elements of the Density array not assigned explicitly are set to DPLOT_FILL.

With the formatted input, when DPLOT_SYMMETRIZE = ON, the array is also symmetrized, *i.e.* array(j,i)=array(i,j) for each line that is read in.

If the DPLOT_ORIENTATION is YX, instead of XY, the X and Y axes are swapped during reading in the data.

2.1.6 WRITE_DPLOT — write the Density array of data

Options:

FILE = ⟨string : 1⟩	'counter'	partial or complete filename
DPLOT_FORMAT = ⟨logical : 1⟩	OFF	
DPLOT_ORIENTATION = TYPEVALUES	DEFAULT	DESCRIPTION

Description: It will write a text file containing the Density array of data. Two formats are possible. In free format (DPLOT_FORMAT=OFF), the text file will contain two integer dimensions (NX,NY) followed by data NX * NY values. The first line will contain NX and NY, at most. The following lines will contain a single row of an array, at most. The order of writing the elements is:

```
read(ioinp,*)NX,NY
do i = 1, NX
read(ioinp,*) (array(i,j),j=1,NY)
end do
```

When formatted output is used (DPLOT_FORMAT=ON), the text file will contain a number of lines in the form:

I, J, ARRAY

If the DPLOT_ORIENTATION is YX, instead of XY, the X and Y axes are swapped during writing out the data.

2.1.7 WORLD — define the extent of your data

Options:

PERSPECTIVE = ⟨logical : 1⟩	OFF	
EYE_TO_SCR = TYPEVALUES	DEFAULT	DESCRIPTION
SCR_TO_TOP = TYPEVALUES	DEFAULT	DESCRIPTION
PAPER_WINDOW = ⟨real : 5⟩	7.0 15.8 17.0 23.0 0.0	
POSITION = ⟨integer : 2⟩	0 1	
WORLD_WINDOW = ⟨real : 4⟩	-999. -999. -999. -999.	
WORLD_FRACTION = ⟨real : 1⟩	-999	
NO_XY_SCOLUMNS = ⟨integer : 2⟩	0 0	
XY_SCOLUMNS = ⟨integer : 0⟩		
A4_WINDOW_MARGIN = ⟨logical : 1⟩	OFF	

Description: This command has to be called before any plotting is done. It calculates the extent of the plot and initializes the Plot coordinate system — it determines its position on a paper. If you selected automatic World bounds setting, you should have your data already read by the **READ_TABLE**, **READ_DPLOT**, or **READ_PDB** commands.

When plotting a PDB structure, the PERSPECTIVE, EYE_TO_SCR, and SCR_TO_TOP have to be set up at the time of drawing with the **BALL_STICK** command to properly scale the plot.

PAPER_WINDOW defines the position and orientation of a window on the paper. The format is (XMIN YMIN XMAX YMAX ORIENTATION). The origin for rotation is the origin of the Base PostScript coordinate system which is in the lower left corner of a paper. Units are cm and degrees. This window on the paper will contain the data from WORLD_WINDOW. Any data points outside this area will be clipped. If orientation of the window is 90° then the plot will be printed in the landscape mode.

In POSITION, if the first element is different from 0 then the PAPER_WINDOW is defined automatically using the position codes (1–37) irrespective of the value of PAPER_WINDOW. The following convention is used:

1 − 8 small 1-4, left column; 5-8, right column (8 plots / page)

9 − 11 medium small (3 plots / page)

12 − 13 medium (2 plots / page)

14 medium large (1 plots / page)

15 large, landscape (1 plots / page)

16 − 37 tiny, portrait (32 plots / page)

If the second element is 0, the aspect ratio between X and Y is set to 1.3333 (horizontal rectangle), if it is 1 the aspect ratio is 1.0 (square), if it is 2 the aspect ratio is 3.9 (very extended horizontal rectangle).

WORLD_WINDOW defines the coordinate axes in the World of the data to be plotted. These ranges will correspond to the PAPER_WINDOW window. However, if any of the four values is -999 (by default this is the case), the program will try to calculate that value from the data read in by the last data input command (**READ_TABLE**, **READ_DPLOT**, or **READ_PDB**).

WORLD_FRACTION defines the fraction of the number of the central points that are used to get the WORLD_WINDOW when calculated automatically. By default, this is 1. This is useful when there are a small number of outliers that you do not want to plot because the large scale would observe the relationship between the remaining points.

NO_XY_SCOLUMNS defines the numbers of selected columns in the Table array that are to be examined for the maximal and minimal values of X and Y, respectively, when calculating the real World extent of the graph. If any of the numbers is 0 then NO_XY_SCOLUMNS and XY_SCOLUMNS are set to reflect the values in XY_COLUMNS. Use this variable when automatic boundaries are required for multi-line plots and it is not clear which data vector should be used for bounds setup.

In multi-plot plots, do not forget that undefined NO_XY_SCOLUMNS and XY_SCOLUMNS are set automatically in the first **WORLD** command and will stay defined until reset with **SET** or **RESET** commands.

XY_SCOLUMNS defines the columns for the range searching (see also above). The first NO_XY_SCOLUMNS(1) integers define the X-columns in the Plotting array for X range and the last NO_XY_SCOLUMNS(2) integers are the Y columns for the Y range.

If A4_WINDOW_MARGIN is `ON` a line around the Bounding Box of the Base PostScript coordinate system is drawn, *i.e.* A4 paper is bounded.

2.1.8 ZOOM — zoom into your data

Options:

SCALE_WORLDX = ⟨`real : 1`⟩ `-999`

Description: SCALE_WORLDX if different from -999, it will find the x of the peak value in a histogram and then scale the number of bins in the histogram to the left and right from the peak by SCALE_WORLDX. It then condenses the plot array accordingly. If you follow by **WORLD** and plotting, you will get only a subset of data plotted.

2.1.9 AXES2D — draw coordinate axes, ticks, and labels

Options:

Y_SCALE = ⟨`string` : 1⟩	`'LEFT'`
CAPTION_XLEFT = ⟨`real` : 1⟩	`-0.15`
CAPTION_XRIGHT = ⟨`real` : 1⟩	`1.05`
TICK_FONT = ⟨`integer` : 1⟩	`3`
X_LABEL_STYLE = ⟨`integer` : 1⟩	`2`
Y_LABEL_STYLE = ⟨`integer` : 1⟩	`2`
X_LABELS = ⟨`string` : 0⟩	`''`
Y_LABELS = ⟨`string` : 0⟩	`''`
X_TICK = ⟨`real` : 0⟩	`-999. -999. -999.`
Y_TICK = ⟨`real` : 0⟩	`-999. -999. -999.`
X_LABEL_SHIFT = ⟨`real` : 1⟩	`0`
Y_LABEL_SHIFT = ⟨`real` : 1⟩	`0`
X_TICK_LABEL = ⟨`integer` : 2⟩	`-999 -999`
Y_TICK_LABEL = ⟨`integer` : 2⟩	`-999 -999`
X_TICK_DECIMALS = ⟨`integer` : 1⟩	`-999`
Y_TICK_DECIMALS = ⟨`integer` : 1⟩	`-999`
Y_AXIS_FACTOR = ⟨`real` : 1⟩	`1`
Y_AXIS_FACTOR = ⟨`real` : 1⟩	`1`
AXES2D_LINE_TYPE = ⟨`integer` : 1⟩	`2`
TICK_LINE_TYPE = ⟨`integer` : 1⟩	`2`
TICK_SMALL_LINE_TYPE = ⟨`integer` : 1⟩	`3`

EXPONENT = ⟨logical : 1⟩ OFF

Description: It will plot the axes of the coordinate system.

TICK_FONT sets the font size for labeling the ticks.

X_LABEL_STYLE and Y_LABEL_STYLE select the labelling regime:

1 labels for X,Y-ticks supplied explicitly to the routine.

2 labels calculated automatically.

3 labels not displayed at all.

X_TICK and Y_TICK define, in World coordinates, the position of the first tick on the X,Y-axes, the spacing between the ticks, and the rightmost tick position.

X_LABEL_SHIFT and Y_LABEL_SHIFT shift the X/Y-axes labels. Shifts are specified in the Plot coordinates.

X_TICK_LABEL and Y_TICK_LABEL set the index of the X/Y-axes ticks that are numbered first, and also every which tick from the first one on is numbered.

X_TICK_DECIMALS and Y_TICK_DECIMALS set the number of decimal places used in the automatic calculation of the X/Y-axes tick labels. If 0, only a dot will appear after an integer. If −1 only an integer will appear.

If Y_SCALE is RIGHT it will plot the Y-axis ticks and their numbers on the right side of the plot. Default is LEFT.

CAPTION_XLEFT and CAPTION_XRIGHT are returned by **AXIS2D** for later use by the **CAPTION** command in placing the captions next to the Y-axis. If these automatic values are not good you can correct them manually (rarely needed).

X_AXIS_FACTOR is used to scale the X-label ticks before the label is written out. If EXPONENT is ON then 'En' is added to the tick label where n = nint[log 10(X_AXIS_FACTOR)]. If EXPONENT is OFF then 'En' is not added and could be included in the axis label with the **CAPTION CAPTION_POSITION = 4 or 5** command.

2.1.10 PLOT2D — draw a 2D line or scatter plot

Options:

PLOT2D_SYMBOL_TYPE = ⟨integer : 1⟩ 0

PLOT2D_LINE_TYPE = ⟨integer : 1⟩ 1

XY_COLUMNS = ⟨integer : 2⟩ 1 2

POINT_FONT = ⟨integer : 1⟩ 6

LABEL_FONT = ⟨integer : 1⟩ 6

LABEL_LOCATION = ⟨integer : 1⟩ 2

LABEL_COLUMN = ⟨integer : 1⟩	0	
COLOR_COLUMN = ⟨integer : 1⟩	0	
COLOR_RANGE = ⟨real : 2⟩	-999 -999	
COLOR_STYLE = ⟨string : 1⟩	'GRAY'	(GRAY BLACK)

Description: This command will plot a line and/or the points defined by the selected columns in the Table array.

PLOT2D_SYMBOL_TYPE defines the symbol to be plotted for every point. If 0 nothing is plotted. If set to -1, then the centered integer indices are plotted for each point.

PLOT2D_LINE_TYPE defines a line type to be plotted between successive points in the Table array. If 0 no line is plotted — used for scatter plots.

XY_COLUMNS selects X and Y columns in the Table array. If any of the two columns is not defined, it is substituted by a vector $1, 2, \ldots, N$.

POINT_FONT selects the font for the point symbol in the case where PLOT2D_SYMBOL_TYPE = -1.

If LABEL_COLUMN is a string column, then the labels in that column are drawn for each point. If LABEL_LOCATION is 1, the label is centered on the point, if 2 the label is to the right of the point. LABEL_FONT is the font type for the labels.

If COLOR_COLUMN is defined, it is used to color the symbols from red, yellow, green to blue in the range from COLOR_RANGE[1] to [2]. The HSB color convention is used (hue from 0.0 to 0.5). If COLOR_STYLE is `GRAY`, gradual coloring is done; if BLACK, the values within the range are of color 0 (red), and those outside the range are color 0.5 (blue).

2.1.11 SPECTRUM — draw a bar code plot

Options:

PLOT2D_LINE_TYPE = ⟨integer : 1⟩	1
BAR_XSHIFT = ⟨real : 1⟩	0.0
BAR_WIDTH = ⟨real : 1⟩	1.0
XY_COLUMNS = ⟨integer : 2⟩	1 2

Description: This command plots an energy spectrum looking like a vertical bar code.

PLOT2D_LINE_TYPE defines the line type to be used for horizontal energy levels.

XY_COLUMNS selects the Y column in the Table array that specifies energy levels. Note that Y column is plotted, not X column. This is to be consistent with the **WORLD** command.

BAR_XSHIFT specifies the starting X of the energy levels in the World coordinates.

BAR_WIDTH defines a relative width of the bars in the World coordinates.

2.1.12 HIST2D — draw a 2D histogram

Options:

BAR_GRAYNESS = ⟨`real` : 0⟩	`0.7`
BAR_WIDTH = ⟨`real` : 1⟩	`1.0`
BAR_LINE_TYPE = ⟨`integer` : 1⟩	`2`
NO_XY_SCOLUMNS = ⟨`integer` : 2⟩	`0 0`
XY_SCOLUMNS = ⟨`integer` : 0⟩	
XY_COLUMNS = ⟨`integer` : 2⟩	`1 2`
BAR_XSHIFT = ⟨`real` : 1⟩	`0.0`

Description: This command plots a histogram of X and Y columns in the Table array.

BAR_GRAYNESS defines the grayness of the bar on the scale from 0.0 (black) to 1.0 (white).

BAR_WIDTH defines a relative width of the bars where 1.0 would make two neighbouring bars touch each other. If less than 1 there is empty space between bars.

BAR_LINE_TYPE defines a linetype used to border the bar in the Π shaped way. If linetype is 0 bordering is not done.

In NO_XY_SCOLUMNS, the first element has to be 1 because there can only be one X-column. The second element is the number of Y-columns. It is 1 for normal histograms and more than 1 for a histogram where bars are stacked on top of each other to get a stacked bar at a single X value.

XY_SCOLUMNS specifies X-column and Y-columns in the Table array for the histogram. The dimension of XY_SCOLUMNS has to be NO_XY_SCOLUMNS(1) + NO_XY_SCOLUMNS(2). The default values for NO_XY_SCOLUMNS and XY_SCOLUMNS (when the inputs are 0) are obtained from XY_COLUMNS. If the X-column is not defined, it is substituted by a vector 1,2,...,N. The X coordinate of the bar specifies its mid-point (not the left edge, for example). X-interval corresponding to one bar is always calculated automatically as the difference between the first and last X divided by the number of bars less 1. This works well when you have equal spacing between the points on X axis. If not you can always correct the bar width using BAR_WIDTH.

XY_COLUMNS is used only when default values for XY_SCOLUMNS are required.

BAR_XSHIFT shifts the bars for this amount along the X-axis in real world coordinates. This is to allow the plotting of several bars at the same X without modifying the data files and/or for correcting the placement of bars (eg, centering on the tick, right edge on the tick, ...).

2.1.13 DPLOT — draw a density plot

Options:

BAR_LEGEND = ⟨logical : 1⟩	ON	
BAR_LEGEND_PLACES = ⟨integer : 2⟩	7 1	
DPLOT_GRAYNESS = ⟨real : 2⟩	1.0 0.0	
DPLOT_COLOR = ⟨real : 6⟩	0 0 0 0 0 1	
DPLOT_LINE_TYPE = ⟨integer : 1⟩	3	
DPLOT_PART = ⟨string : 1⟩	'FULL'	LOWER | FULL
DPLOT_BOUNDS = ⟨real : 2⟩	-999. -999.	
DPLOT_STYLE = ⟨string : 1⟩	'GRAY'	GRAY | BLACK | COLOR
NUMBER_DENSITY_PLOT = ⟨logical : 1⟩	OFF	
NUMBER_PLACES = ⟨integer : 2⟩	5 2	pre- and post-decimal point places
POINT_FONT = ⟨integer : 1⟩	6	
PRINT_FONT = ⟨integer : 1⟩	4	
PRINT_DXY = ⟨real : 2⟩	0 0	
DPLOT_PART = ⟨string : 1⟩	'FULL'	LOWER | FULL
DPLOT_STYLE = ⟨string : 1⟩	'GRAY'	GRAY | BLACK | COLOR
NUMBER_DENSITY_PLOT = ⟨logical : 1⟩	OFF	

Description: This command will do a density plot of the Density array read in by the **READ_DPLOT** command.

If BAR_LEGEND is ON it will plot a gray scale code to the right of the plot.

BAR_LEGEND_PLACES sets the number of pre- and post-decimal point places for the labelling of the bar legend.

DPLOT_GRAYNESS defines the grayness of the smallest and largest value to be plotted, respectively. If you want small to be white, set the first element larger than the second one; they have to lie between 0 and 1, per PostScript convention.

DPLOT_COLOR defines the RGB triplets of the smallest and largest RGB value to be plotted, respectively. For example, 0 0 0 1 1 1 is equivalent to setting DPLOT_GRAYNESS to 0 1; 0 0 0 1 0 0 is equivalent to spanning from black to red. To span from small to large in white to blue, use 1 1 1 0 0 1. Instead of pure blue 0 0 1, you could use a nicer darker blue of 0.1 0.1 0.7.

DPLOT_LINE_TYPE defines the line type for the mesh plotted on the density plot.

DPLOT_PART selects the part of the Density array to be plotted.

DPLOT_BOUNDS sets the real World bounds on the values of the Density array corresponding to the DPLOT_GRAYNESS interval.

DPLOT_STYLE selects whether the DPLOT_BOUNDS range is to be (i) colored in RGB color (`COLOR`), (ii) shadowed with various degrees of gray (`GRAY`), or (iii) every cell within the range is to be shaded by the first bound of DPLOT_GRAYNESS and all other cells by the second bound of DPLOT_GRAYNESS.

If NUMBER_DENSITY_PLOT is set to `ON` the number is plotted for each cell instead of the gray rectangle. This number shows the height of the function. You can use the same mechanism as for `GRAY` to show only numbers in certain range.

NUMBER_PLACES sets the number of spaces before and after the decimal point when NUMBER_DENSITY_PLOT = `ON`.

POINT_FONT sets the font type for the numbers when NUMBER_DENSITY_PLOT = `ON`.

PRINT_FONT sets the font type for the numbers for the bar legend.

PRINT_DXY will offset the X and Y of the number printed in each cell (in World coordinates).

2.1.14 FILL_COLUMN — fill a column with values

Options:

COLUMN = ⟨`integer` : 1⟩ `0`

COLUMN_PARAMETERS = ⟨`real` : 2⟩ `1 1`

Description: It will fill column COLUMN of the data table with values equal to COLUMN_PARAMETERS[1] + (i-1) COLUMN_PARAMETERS[2], where i runs from 1 to the number of points in the table. These values can be used for plotting or to help with axis labeling.

2.1.15 CAPTION — place a caption next to an axis

Options:

CAPTION_POSITION = ⟨`integer` : 1⟩ `1`

CAPTION_FONT = ⟨`integer` : 1⟩ `3`

CAPTION_TEXT = ⟨`string` : 1⟩ `'undefined'`

CAPTION_XLEFT = ⟨`real` : 1⟩ `-0.15`

CAPTION_XRIGHT = ⟨real : 1⟩ 1.05

Description: This command puts text at pre-specified positions around the plot.

The sequence of **CAPTION** commands for the same type of a caption is important. The captions will be placed around the graph starting with the position closest to the graph. Therefore, a subtitle should be done before the title, but the X-title should be done before the X-subtitle. CAPTION_XLEFT is the X-position in the Plot coordinates of the leftmost part of the Y-captions (by default -0.15). **AXES2D** returns the precise values for CAPTION_XLEFT, so use **CAPTION** after **AXES2D** without specifying CAPTION_XLEFT. CAPTION_XRIGHT is a similar variable that is used when Y_SCALE = `RIGHT`.

The following positions with respect to the graph are available:

1 above graph, centered

2 below graph, centered

3 left of graph, centered,

4 below graph, right

5 left of graph, top

6 right of graph, center

7 right of graph, top

8 left, top corner of graph

9 right, top corner of graph

10 right, bottom corner of graph

11 left, bottom corner of graph

12 center, top of graph

2.1.16 RESET_CAPTIONS — reset caption positioning

Command: RESET_CAPTIONS

Description: Resets the positions for the subsequent captions. This command has to be executed after the **WORLD** and before the **AXES2D** commands.

2.1.17 LINE2D — draw a line

Options:

LINE2D_XY1 = ⟨real : 2⟩ `0.0 0.0`

LINE2D_XY2 = ⟨real : 2⟩ `0.0 0.0`

LINE2D_GRAYNESS = ⟨real : 1⟩ `0.0`

LINE2D_WIDTH = ⟨real : 1⟩ `1.0`

LINE2D_WIDTH_SCALING = 'Y' 'X' | 'Y'
⟨string : 1⟩

LINE2D_LINE_TYPE = ⟨integer : 1⟩ 0

CLIP = ⟨logical : 1⟩ ON

Description: Plots a line specified in World coordinates. It uses LINE2D_LINE_TYPE if defined (i.e. in 1..LINTYPS, where LINTYPS is the number of line types read from file 'psgl1.ini'), otherwise it uses LINE2D_WIDTH and LINE_GRAYNESS. If LINE2D_WIDTH_SCALING is X (case insensitive), the line width is specified in the units of the X-axis, otherwise in the units of the Y-axis. If CLIP is off, the line can occur outside the plot area, otherwise it is clipped.

2.1.18 SET_RECORD — sets RECORD to selected Table elements

Options:

NO_XY_SCOLUMNS = ⟨integer : 2⟩ 0 0

XY_SCOLUMNS = ⟨integer : 0⟩

Description: This command puts the selected elements in the first line of the Table data into the RECORD variable. RECORD can then be used in captioning the plot or plots.

This command is useful in combination with the **SELECT_DATA** command in order to produce many different captions in one set of plots, from one data file.

2.1.19 PRINT — print a text

Options:

PRINT_XY = ⟨real : 2⟩ 1.0 1.0

PRINT_DXY = ⟨real : 2⟩ 0 0

COORDINATES = ⟨string : 1⟩ 'WORLD' 'WORLD' | 'PLOT'

PRINT_FONT = ⟨integer : 1⟩ 4

PRINT_ANGLE = ⟨real : 1⟩ 0.0

PRINT_HORIZ = ⟨integer : 1⟩ 2

PRINT_VERT = ⟨integer : 1⟩ 2

PRINT_TEXT = ⟨string : 1⟩ 'undefined'

PRINT_MODE = ⟨string : 1⟩	'XY'	XY | POINT
PRINT_POINT = ⟨integer : 1⟩	1	
XY_COLUMNS = ⟨integer : 2⟩	1 2	

Description: Prints text positioned in World (COORDINATES = WORLD) or Plot (COORDINATES = PLOT) coordinates at PRINT_XY (PRINT_MODE = XY). Alternatively, if PRINT_MODE = POINT, the coordinates of the point PRINT_POINT in the columns XY_COLUMNS are used. If point index is out of range, the last point is used. In either case, (X,Y) is translated for PRINT_DXY. PRINT_ANGLE is a rotation of the text, PRINT_HORIZ is 1 for left justified, 2 for centered and 3 for right justified, PRINT_VERT is 1 for bottom aligned, 2 for center aligned and 3 for top aligned.

For all text printing, including the one submitted to the **CAPTION** command, the following conventions hold:

- Superscript: embedded in _
- Subscript : embedded in ∧
- Greek char : embedded in @

Multi-level embedding is allowed.

Special characters can be printed by using the PostScript character codes. For example, to print the Åcharacter, use '
305'.

2.1.20 NEW_PAGE — start a new page

Command: NEW_PAGE NO_COPIES ⟨integer : 1⟩

Description: Advances to the next page: the next plot will be on the new page. It can only be used when Encapsulated PostScript (EPSF) is not selected.

2.1.21 ARROW — draw an arrow

Command: ARROW [options]

ARROW_POSITION = ⟨real : 4⟩	X1 Y1 X2 Y2
ARROW_SHAPE = ⟨real : 3⟩	tail thickness, arrow width, arrow length

Description: Draws an arrow from (X1,Y1) to (X2,Y2) (tail to tip), in World coordinates (COORDINATES = WORLD) or in Plot coordinates (COORDINATES = PLOT). The elements of ARROW_SHAPE are tail thickness, arrow width, arrow length in Plot coordinates.

2.1.22 POSTSCRIPT — write a PostScript command

Command: POSTSCRIPT POSTSCRIPT_TEXT = ⟨string : 1⟩

Description: Takes the POSTSCRIPT_TEXT string as a literal PostScript command and writes it to the output PS file.

2.1.23 TRANSFORM — transform Table or Density array data

Options:

TRF_TYPE = ⟨`string` : 1⟩	`'LOGARITHMIC1'`
TRF_PARAMETERS = ⟨`real` : 0⟩	`0.0 1.0`
TRF_UNDEFINED = ⟨`real` : 1⟩	`-999999`

Description: Transforms the columns of the Table array selected by XY_SCOLUMNS array. If NO_XY_SCOLUMNS are both 0, it transforms the Density array instead. You must be careful with labelling the ticks (**AXES2D**) and scaling (**WORLD**) when using this option since it transforms the data itself not only plotting of them.

TRF_PARAMETERS sets any parameters that may be required for the transformation.

TRF_TYPE selects the type of transformation:

`LOGARITHMIC1`: $Y = \ln[\mathrm{TRF\_PARAM}(1) + (Y - YMIN) \cdot \mathrm{TRF\_PARAM}(2)]$

`LOGARITHMIC2`: $Y = \ln[\mathrm{TRF\_PARAM}(1) + Y \cdot \mathrm{TRF\_PARAM}(2)]$

`LOGARITHMIC3`: $Y = \log10[\mathrm{TRF\_PARAM}(1) + (Y - YMIN) \cdot \mathrm{TRF\_PARAM}(2)]$

`LOGARITHMIC4`: $Y = \log10[\mathrm{TRF\_PARAM}(1) + Y \cdot \mathrm{TRF\_PARAM}(2)]$

`CUMULATIVE`: $Y_i = \sum_{k=1,i} Y_k$

`LINEAR`: $Y = \mathrm{TRF\_PARAM}(1) + Y \cdot \mathrm{TRF\_PARAM}(2)$

`INVERSE`: $Y = \mathrm{TRF\_PARAM}(1) + \mathrm{TRF\_PARAM}(2)$ / Y

`EXPONENTIAL`: $Y = \mathrm{TRF\_PARAM}(1) + \exp[\mathrm{TRF\_PARAM}(2) + \mathrm{TRF\_PARAM}(3)]$

`NORMALIZE`: $Y = Y / \sum_i Y_i$

YMIN is the smallest value in all selected columns of the Table array or the smallest value in the Density array, as appropriate. `CUMULATIVE` is not available for transformation of the Density array.

Any transformation that is undefined (for example, a logarithm of a non-positive argument, or a division by zero) is assigned a value TRF_UNDEFINED.

2.1.24 RESET — reset TOP

Command: RESET

Description: Reads in the `top.ini` file again. Resets all parameters to its default values. Use **RESET** before plotting the second, third, *etc* plot in the same TOP program to prevent surprises resulting from the non-default values of arguments such as X_TICK, XY_COLUMN, *etc* which were possibly set automatically when producing the first plot.

2.1.25 GET_BARS — calculate histogram bars

Options:

BAR_DX = ⟨real : 1⟩	1.0
WORLD_WINDOW = ⟨real : 4⟩	-999. -999. -999. -999.
XY_COLUMNS = ⟨integer : 2⟩	1 2

Description: Transforms the current X vector into a histogram by counting how many X coordinates fall in a certain interval. The interval size must be specified explicitly by the real World range BAR_DX. The centers of these intervals are returned in X and the heights of the bars in Y. Note that the current data in the Table array is erased, histogram coordinates are copied to the current X and Y. Only XMIN and XMAX are used to determine the X-range. If any of these two is undefined (-999) default values are supplied as the largest and smallest X in the data with an added DX on both sides. XMAX is corrected so that XMAX = XMIN + (N+1)*BAR_DX where N is the number of points (bars) in the new Table array. The first bin starts at XMIN+0.5*DX and the last bin ends at XMAX-0.5*DX. For example, if you have points with X-coordinates ranging from 0 to 100, you could set WORLD_WINDOW to -5 0 105 0 and BAR_DX to 10, which will create bins 0–9.999, 10–19.999, ..., 90–100.0; they will be centered on 5, 15, 25, etc. There will also be half of the bin width on each side of the histograms when it is plotted (from -5 to 105 on the X-axis).

XY_COLUMNS selects the X and Y columns. If Y column exists before the call, this routine also calculates the average and standard deviaton of the Y-values in each bin. These are returned in the two columns after the currently existing columns.

2.1.26 GET_DENSITY — calculate Density plot data

Options:

BAR_DX = ⟨real : 1⟩	1.0
BAR_DY = ⟨real : 1⟩	1
WORLD_WINDOW = ⟨real : 4⟩	-999. -999. -999. -999.
XY_COLUMNS = ⟨integer : 2⟩	1 2

Description: Transforms the current X and Y vectors into a density plot (3D histogram) by counting how many X and Y coordinates fall in a certain interval. The interval sizes must be specified explicitly by the real parameters BAR_DX and BAR_DY. The heights of the density plot are returned in the Density array. All four values in WORLD_WINDOW are used to determine the X-range. If any of them is undefined (-999) default values are supplied as the largest and smallest X and Y in the data with an added DX/DY on all sides. This makes the WORLD exactly superposable to the world from the 1 .. NBARX, 1..NBARY density plot. This world goes from 0,0 to NBARX+1,NBARY+1.

The center of the first bin is at XMIN+DX,YMIN+DY and the center of the last bin is at XMAX-DX,YMAX-DY. This gives a 0.5*DX, 0.5*DY margin around the plotted bins. XMAX (and YMAX) is corrected so that XMAX = XMIN + (N+1)*BAR_DX where N is the number of points (bars) in the new Density array. XY_COLUMNS selects the X and Y columns.

2.1.27 SHUFFLE_DPLOT — re-organize the Density plot data

Options:

SHUFFLE_OPERATION = ⟨string : 1⟩	'TRANSPOSE'	REVERT_X, REVERT_Y

Description: The transpose operation only has sense for square matrices. The other two commands invert the order of the points along the X and Y coordinates, respectively. To exchange the orientation of the X and Y axes as they appear on the plot, you should use the DPLOT_ORIENTATION option of the **READ_DPLOT** command.

2.1.28 DENSITY_TO_XY — Density to XY data

Command: DENSITY_TO_XY

Description: Copies the current density array to the XY table.

2.1.29 XY_TO_DENSITY — XY to density data

Command: XY_TO_DENSITY

Description: Copies the current XY table to the density array.

2.1.30 LEGEND — draw a legend

Options:

SYMBOL = ⟨string : 1⟩	'LINE'	LINE | POINT | BAR
LEGEND_POSITION = ⟨integer : 1⟩	7	
PLOT2D_LINE_TYPE = ⟨integer : 1⟩	1	
PLOT2D_SYMBOL_TYPE = ⟨integer : 1⟩	0	
BAR_LINE_TYPE = ⟨integer : 1⟩	2	
BAR_GRAYNESS = ⟨real : 0⟩	0.7	
= ⟨string : 1⟩	'description'	DESCRIPTION

DESCRIPTION_FONT = ⟨integer : 1⟩ 5

Description: Plots the legend for objects (curves, points, and bars). The legend is in the form 'symbol symbol_description'. Successive **LEGEND** commands print legends one per line, starting at the top. **RESET_LEGEND** has to be issued before **LEGEND**. Note that you can use this command to place text right to the plot if you specify invisible object types (0). The value of the SYMBOL keyword determines the type of the object symbol in the legend (POINT, LINE, or BAR). DESCRIPTION is the text describing the symbol. LEGEND_POSITION can have the same values as CAPTION_POSITION.

2.1.31 RESET_LEGEND — reset legend positioning

Command: RESET_LEGEND

Description: Resets the starting Y position of the next legend to the initial value at the top of the plot.

2.1.32 PLOT_ERROR_BARS — draw error bars

Options:

XY_COLUMNS = ⟨integer : 2⟩	1 2	
ERROR_COLUMN = ⟨integer : 1⟩	0	
ERROR_SIGN = ⟨string : 1⟩	'UP DOWN'	'UP' | 'UP DOWN' | 'DOWN'
PLOT2D_SYMBOL_TYPE = ⟨integer : 1⟩	0	
PLOT2D_LINE_TYPE = ⟨integer : 1⟩	1	
POINT_FONT = ⟨integer : 1⟩	6	

Description: This command plots error bars. It relies on X and Y columns selected by the XY_COLUMNS argument (as in the PLOT2D command), and on the ERROR_COLUMN argument which selects the column in the Table array that contains the error in Y-coordinate for each (X,Y) point. The graphical attributes are the same as those used for the **PLOT2D** command. The error-bar is a vertical line with two short horizontal caps on top and bottom. The error line extends for 'error' units below the central point and for 'error' units above the central point; thus, the total height of the error bar is twice the error. PLOT2D_SYMBOL_TYPE selects the symbol type used to plot at the center of the error bar. If
PLOT2D_SYMBOL_TYPE is -1 then POINT_FONT font is used to print the points indices instead of point symbols. PLOT2D_LINE_TYPE selects the line type used for plotting the error bars. IF ERROR_SIGN contains word `UP`, the error value will be added to the y-value. If it contains `DOWN`, the error value will be subtracted from the y-value. ERROR_SIGN contains both by default, `UP DOWN`.

2.1.33 SELECT_DATA — select some rows in Table array

Command: SELECT_DATA SELECT_COLUMN = ⟨`integer` : 1⟩, SELECT_RANGE = ⟨`real` : 2⟩

Description: This command selects the lines of the Table array that have a value in a specified column in a specified range. This is useful when you want to plot only a subset of data, for example make a (X,Y) plot at a fixed Z.

2.1.34 SWITCH_PS — open a new PostScript file

Command: SWITCH_PS FILE = ⟨`string` : 1⟩

Description: This command will close the current PS file and open another one for subsequent output.

2.1.35 FIT — non-linear least-squares fit of data

Options:

XY_COLUMNS = ⟨`integer` : 2⟩	`1 2`	
ERROR_COLUMN = ⟨`integer` : 1⟩	`0`	
FIT_MODEL = ⟨`string` : 1⟩	`'POLYNOMIAL'`	EXPONENTIAL NORMAL LOGNORMAL LOGARITHMIC POLYGAUSS
FIT_PARAM_INITIAL = ⟨`real` : 0⟩	`1 1 1 1 1 1 1 1 1`	
FIT_PARAM_INDICES = ⟨`integer` : 0⟩	`1 2`	
FIT_CUTOFFS = ⟨`real` : 2⟩	`0.0001 0.0001`	
FIT_WORLD = ⟨`logical` : 1⟩	`OFF`	
WORLD_WINDOW = ⟨`real` : 4⟩	`-999. -999. -999. -999.`	
NO_SPLINE_POINTS = ⟨`integer` : 1⟩	`400`	

Description: This command does a non-linear least-squares fitting of the (X,Y) data in columns XY_COLUMNS to the model selected by FIT_MODEL. The standard errors of the data points can be specified in column ERROR_COLUMN. If this column index is undefined, all errors are set to 1. FIT_PARAM_INITIAL specifies the initial estimates for all the parameters. FIT_PARAM_INDICES specifies the indices of parameters to be actually varied in the fitting. FIT_CUTOFFS specifies the convergence criterion: when the absolute and relative change in the chi-square is smaller than specified in two consecutive cycles, the fitting exits. If FIT_WORLD is `OFF`, the Y-column contains the function values for the corresponding X. If FIT_WORLD is `ON`, the X column is changed to NO_SPLINE_POINTS points that equidistantly span WORLD_WINDOW X-range and Y-column is filled with the function values calculated at these new X-points. Note that Xmin and Xmax of the World are specified as the first and third element of WORLD_WINDOW.

2.1.36 FIT2 — non-linear least-squares fit of data

Command: FIT2 [options]

Description: Downward compatible with **FIT**, except that it can also do multidimensional least-squares fitting. Should replace **FIT** with time.

2.1.37 SMOOTH_TABLE — smooth the Table array data

Options:

SMOOTH_TYPE = ⟨string : 1⟩	'SPLINE'	SPLINE | AVERAGE
NO_SPLINE_POINTS = ⟨integer : 1⟩	400	
XY_COLUMNS = ⟨integer : 2⟩	1 2	
NO_XY_SCOLUMNS = ⟨integer : 2⟩	0 0	
XY_SCOLUMNS = ⟨integer : 0⟩		
HALF_WINDOW = ⟨integer : 1⟩	3	
PLOT_POINTS = ⟨integer : 1⟩	0	

Description: Smoothes data in the Table array using one of the two methods selected by SMOOTH_TYPE: the cubic spline or running average method. The main appearance difference between the two methods is that spline smoothing goes through the original points (if they were evenly distributed), whereas window averaging changes the original points.

In the spline method, NO_SPLINE_POINTS is the number of points to be calculated by spline smoothing. It is returned in PLOT_POINTS. Columns X and Y are selected by XY_COLUMNS; note the difference with the running average method.

In the running average method, the weight of point j is proportional to (HALF_WINDOW$-|i-j|+1$) where i is the central point and HALF_WINDOW is the number of points left and right of point i that are in the window. Only the columns specified by NO_XY_SCOLUMNS and XY_SCOLUMNS are smoothed; note the difference with the spline method. PLOT_POINTS is not changed.

2.1.38 READ_PDB — read a Brookhaven molecular structure

Command: READ_PDB FILE = ⟨string : 1⟩, PDB_FORMAT = ⟨string : 1⟩

Description: Reads in the PDB file according to the format specified. Format can be `PDB` or `XYZ`. The former is the Brookhaven Protein Databank format. The latter consists only of x,y,z for one atom per line.

2.1.39 WRITE_PDB — write a Brookhaven molecular structure

Command: WRITE_PDB FILE = ⟨`string` : 1⟩

Description: Writes out the PDB file.

2.1.40 MAKE_BONDS — make a list of bonds

Options:

BOND_TYPE = ⟨`string` : 1⟩	`'COVALENT'`	
COVALENT_BOND = ⟨`real` : 2⟩	`-999 -999`	
BOND_COLOR = ⟨`real` : 1⟩	`1.0`	
BOND_LINE = ⟨`integer` : 1⟩	`2`	
BOND_TAPER = ⟨`real` : 1⟩	`0.3`	
BOND_WIDTH_FACTO = TYPEVALUES	`DEFAULT`	DESCRIPTION
ADD_BONDS = ⟨`logical` : 1⟩	`ON`	

Description: Constructs the bonds between the currently selected bonding set of atoms and either adds them to the list of existing bonds or creates this list from scratch, depending on the logical variable ADD_BONDS.

BOND_COLOR sets the color of the current bonds. The coloring scheme is from PostScript: 0 for black and 1 for white; intermediate values select various shades of gray.

COVALENT_BOND defines the distance cuttoff between which the atom pair is recognized as bonded to each other. If any of the two values is undefined (i.e. -999), then the covalent bond exists when a distance between the two atoms is less than 0.55 times the sum of the two van der Waals radii and more than half of that value. The radii are obtained from the PDB atom names by first recognizing an extended atom type in the residue, and then looking into the radii library file for all atom types in all residue types. Hydrogen atoms are treated separetely — they are just recognized as such directly and their radii assigned accordingly.

BOND_LINE sets the line type for drawing the current bonds. If the line type is negative than only the line of the type -BOND_LINE is drawn, not the polygon with caps.

BOND_WIDTH_FACTOR sets the bond width. If too large (approx 1), bond coordinates may be undefined.

BOND_TAPER is a factor that is multiplied by the delta Z of the bond and its width to increase the width of the nearer end, and decrease the width of the further end. It should be 0 or larger than 0.

2.1.41 LABEL_ATOMS — label the selected atoms

Options:

LABEL_STYLE = ⟨string : 0⟩	`'RESIDUE_TYPE1 RESIDUE_NUMBER'`	
LABEL_FONT = ⟨integer : 1⟩	`6`	
LABEL_LOCATION = ⟨integer : 1⟩	`2`	

Description: Labels the currently selected labelling set of atoms with the currently selected labelling style. Labelling style is a string and can be any sequence of the items listed above, in any order.

2.1.42 DEFAULT_ATOM_COLOR — color all the atoms

Command: DEFAULT_ATOM_COLOR

Description: Assigns default attributes to all the atoms: atom color and atom lines. Useful to do after **SELECT_ATOMS** which assigns a specific color and line; if you need generic coloring for different atom types.

2.1.43 SELECT_ATOMS — select atoms in a molecule

Options:

SELECTION_SEARCH = ⟨string : 1⟩	`'SEGMENT'`	SEGMENT | SPHERE
FROM = ⟨string : 1⟩	`'ALL'`	ALL DISPLAYING BONDING LABELLING
FOR = ⟨string : 0⟩	`'DISPLAYING BONDING LABELLING'`	
RES_TYPE = ⟨string : 1⟩	`'ALL'`	
ATOM_TYPE = ⟨string : 1⟩	`'ALL'`	
SELECTION_MODE = ⟨string : 1⟩	`'ATOM'`	RESIDUE | ATOM
SELECTION_STATUS = ⟨string : 1⟩	`'INITIALIZE'`	REMOVE ADD
RADIUS = ⟨real : 1⟩	`1.5`	
RADIUS_FACTOR = ⟨real : 1⟩	`0.150`	
ATOM_COLOR = ⟨real : 1⟩	`1.0`	
ATOM_LINE = ⟨integer : 1⟩	`2`	
SELECTION_SEGMENT = ⟨string : 2⟩	`'X' 'X'`	

SELECTION_STEP = ⟨integer : 1⟩ 1

SPHERE_CENTER = ⟨string : 2⟩ 'undefined' undefined

SLAB = ⟨real : 5⟩ 9999 9999 0 0 0

Description: Adds atoms to, removes atoms from, or initializes the ASGL atom sets selected by FOR. SELECTION_STATUS determines whether the selected atoms are added (ADD), removed (REMOVED), or a set is initialized and then the selected atoms are added (INITIALIZE). There are three sets of atoms in ASGL: (1) the atoms that are selected for display (**BALL_STICK**), (2) the atoms that are selected for calculation of bonds (**MAKE_BONDS**) and (3) the atoms that are selected for labelling (**LABEL_ATOMS**). The FOR variable is a scalar string that can contain any combination of the three selections: The selection of atoms is a hierarchical process: First, FROM specifies which of the three atom sets are used for scanning; if FROM = ALL, all atoms in the input pdb file are used for scanning. ATOM_TYPE and RES_TYPE constrain scanning to specified atom and residue types, respectively. They can contain several atom and residue types in one quoted string. SELECTION_MODE determines whether only an atom satisfying all search criteria is to be selected or all atoms in a residue of a found atom are to be selected.

The SEARCH_STRING specifies SEGMENT or SPHERE search; this determines the other possible arguments.

The SEGMENT search scanns only a single segment specified by the beginning and ending residue number (as found in the input atom file), SEGMENT_RANGE. The value of the residue number can be X, which implies the first or last residue, as appropriate. SELECTION_STEP is a step in the residue index used in scanning for the atoms. This is useful in labelling only every 5-th CA atom, for example.

The SPHERE search scanns only over those atoms that are closer than SPHERE_RADIUS to the SPHERE_CENTER atom, after the center atom was translated by (xtrans, ytrans, ztrans) specified in SLAB. If the first element of SPHERE_CENTER is INDEX then the second element is an atom index of the center atom; otherwise, the first and second element are the residue number (as in the input atom file) and the atom type, respectively. SLAB specifies the interval on Z-axis relative to Z of the translated central atom that imposes another condition on the selected atoms: $Zcen + dz1 < Z + ztrans < Zcen + dz2$. This is useful to make less crowded plots. Larger Z is on front, so dz1 specifes the plane that is further away than the dz2 plane. To get any atoms, $dz1 < dz2$.

The radii of the currently selected display set of atoms is set to RADIUS * RADIUS_FACTOR. If the final atom radii is 0, the atom is not drawn. If RADIUS is undefined (-999) then the van der Waals radius of that atom type is used.

ATOM_COLOR sets the color of the currently selected display set of atoms. The coloring scheme is from PostScript: 0 for black and 1 for white; intermediate values select various shades of gray.

ATOM_LINE sets the line type for drawing the currently selected display set of atoms.

2.1.44 BALL_STICK — draw a molecule

Options:

PERSPECTIVE = ⟨logical : 1⟩ OFF

EYE_TO_SCREEN = ⟨real : 1⟩ 30.0

SCREEN_TO_TOP = ⟨real : 1⟩ 0.0

Description: Plots a ball-and-stick plot of the currently selected display set of atoms and of the selected bonds.

2.1.45 ROTATE_MOL — rotate the molecule using rotation matrix

Command: ROTATE_MOL ROTATION_MATRIX = ⟨real : 9⟩

Description: Rotates the whole molecule according to the rotation matrix.

2.1.46 ROTATE_MOL_AXIS — rotate the molecule around the axis

Command: ROTATE_MOL_AXIS AXIS ⟨real : 3⟩, DEGREES = ⟨real : 1⟩

Description: Rotates the whole molecule around the axis AXIS for DEGREES degrees.

2.1.47 TRANSLATE_MOL — translate the molecule

Command: TRANSLATE_MOL SHIFT = ⟨real : 3⟩

Description: Translates the whole molecule for SHIFT Ångstroms.

2.1.48 CENTER_MOL — center the molecule

Command: CENTER_MOL

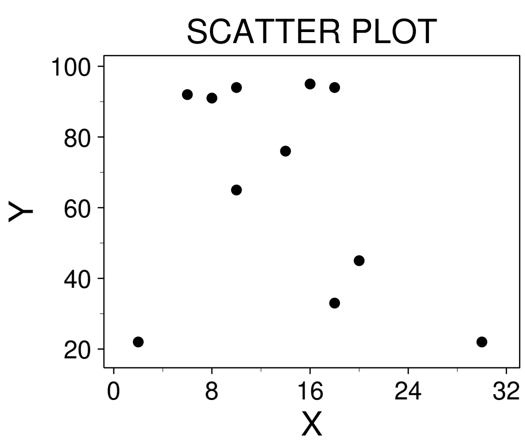
Description: Center the whole molecule.

Chapter 3

Asgl examples

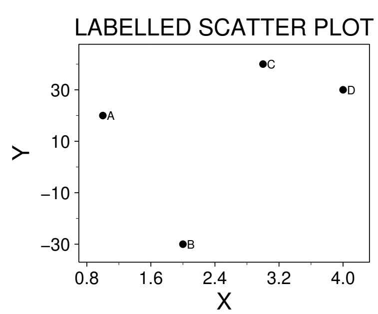
3.0.1 Scatter plots — `scatter.top`

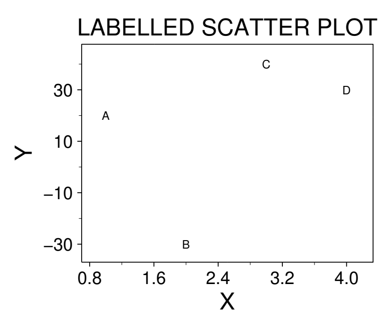
```
#EPSF .ps

READ_TABLE FILE 'scatter.dat'

SET POSITION 12 0
WORLD

AXES2D

RESET_CAPTIONS
CAPTION CAPTION_POSITION 1, CAPTION_FONT 2, CAPTION_TEXT 'SCATTER PLOT'
CAPTION CAPTION_POSITION 2, CAPTION_FONT 2, CAPTION_TEXT 'X'
CAPTION CAPTION_POSITION 3, CAPTION_FONT 2, CAPTION_TEXT 'Y'

SET PLOT2D_SYMBOL_TYPE 4, PLOT2D_LINE_TYPE = 0
PLOT2D
```



3.0.2 Labelled scatter plots — `labls.top`

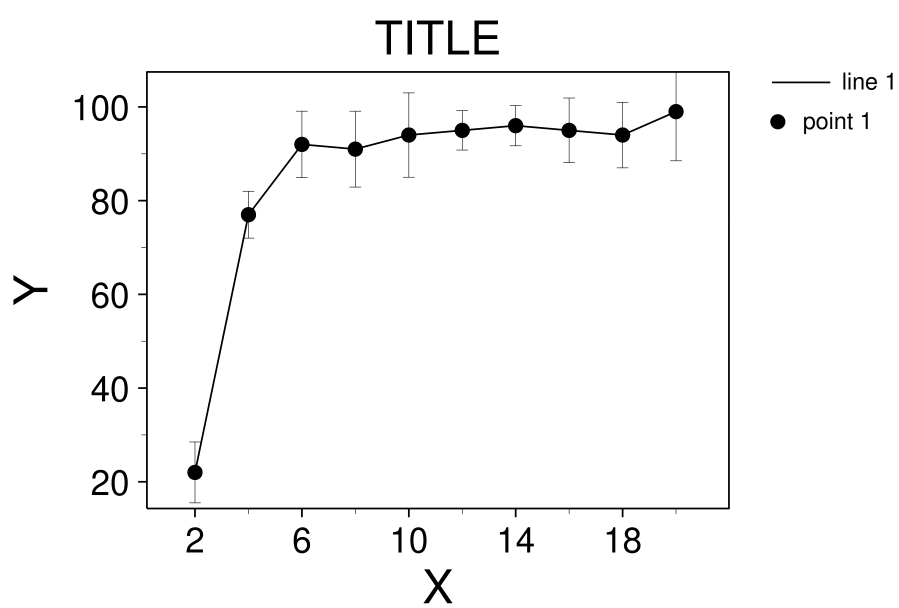
```
#EPSF .ps

READ_TABLE FILE 'labls.dat'

SET POSITION 1 0
WORLD

AXES2D

RESET_CAPTIONS
CAPTION CAPTION_POSITION 1, CAPTION_FONT 2, CAPTION_TEXT 'LABELLED SCATTER PLOT'
CAPTION CAPTION_POSITION 2, CAPTION_FONT 2, CAPTION_TEXT 'X'
CAPTION CAPTION_POSITION 3, CAPTION_FONT 2, CAPTION_TEXT 'Y'

SET PLOT2D_SYMBOL_TYPE 4, PLOT2D_LINE_TYPE = 0
PLOT2D LABEL_LOCATION = 2, LABEL_COLUMN = 3, LABEL_FONT = 5


SET POSITION 5 0
WORLD

AXES2D

RESET_CAPTIONS
CAPTION CAPTION_POSITION 1, CAPTION_FONT 2, CAPTION_TEXT 'LABELLED SCATTER PLOT'
CAPTION CAPTION_POSITION 2, CAPTION_FONT 2, CAPTION_TEXT 'X'
CAPTION CAPTION_POSITION 3, CAPTION_FONT 2, CAPTION_TEXT 'Y'

SET PLOT2D_SYMBOL_TYPE 0, PLOT2D_LINE_TYPE = 0
PLOT2D LABEL_LOCATION = 1, LABEL_COLUMN = 3, LABEL_FONT = 5
```





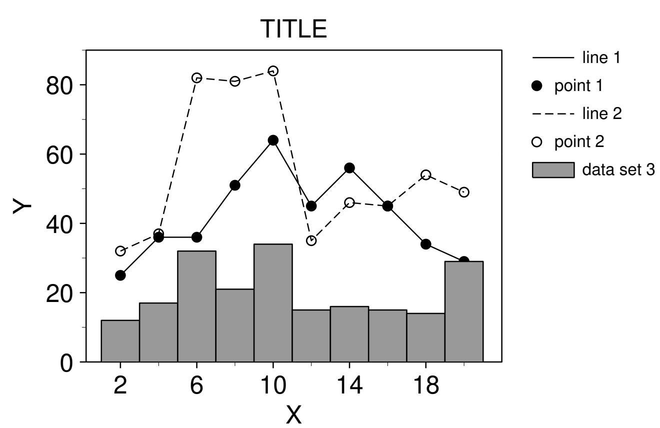
3.0.3 Error bars — `error.top`

```
#EPSF .ps

READ_TABLE FILE 'error.dat'
SET POSITION 12 0
WORLD

AXES2D

RESET_CAPTIONS
CAPTION CAPTION_POSITION 1, CAPTION_FONT 2, CAPTION_TEXT 'TITLE'
CAPTION CAPTION_POSITION 2, CAPTION_FONT 2, CAPTION_TEXT 'X'
CAPTION CAPTION_POSITION 3, CAPTION_FONT 2, CAPTION_TEXT 'Y'

RESET_LEGEND

SET PLOT2D_SYMBOL_TYPE 4, PLOT2D_LINE_TYPE = 2
LEGEND SYMBOL = LINE,  DESCRIPTION = 'line 1'
LEGEND SYMBOL = POINT, DESCRIPTION = 'point 1'
PLOT2D
PLOT_ERROR_BARS ERROR_COLUMN = 3, PLOT2D_LINE_TYPE = 3, ;
                PLOT2D_SYMBOL_TYPE = 3
```



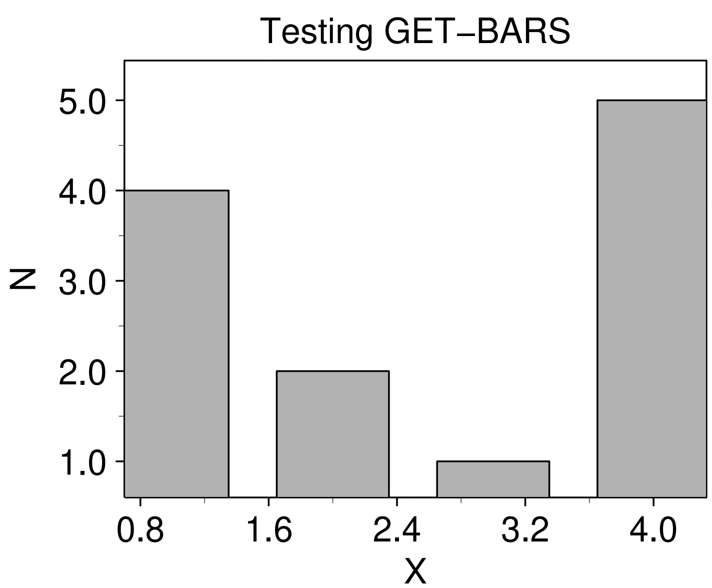
3.0.4 Mixed style plots — `mixed.top`

```
#EPSF .ps

READ_TABLE FILE 'hist1.dat'
SET XY_COLUMNS 1 2
SET POSITION 12 0
WORLD WORLD_WINDOW -999 0 -999 90

AXES2D

RESET_CAPTIONS
CAPTION CAPTION_POSITION 1, CAPTION_TEXT 'TITLE'
CAPTION CAPTION_POSITION 2, CAPTION_TEXT 'X'
CAPTION CAPTION_POSITION 3, CAPTION_TEXT 'Y'

RESET_LEGEND

SET PLOT2D_SYMBOL_TYPE 4, PLOT2D_LINE_TYPE = 2
LEGEND SYMBOL = LINE,  DESCRIPTION = 'line 1'
LEGEND SYMBOL = POINT, DESCRIPTION = 'point 1'
PLOT2D

READ_TABLE FILE 'hist2.dat'
SET PLOT2D_SYMBOL_TYPE 3, PLOT2D_LINE_TYPE = 5
LEGEND SYMBOL = LINE,  DESCRIPTION = 'line 2'
LEGEND SYMBOL = POINT, DESCRIPTION = 'point 2'
PLOT2D

READ_TABLE FILE 'hist3.dat'
SET BAR_GRAYNESS = 0.6, BAR_LINE_TYPE = 2
LEGEND SYMBOL = BAR,  DESCRIPTION = 'data set 3'
HIST2D
```



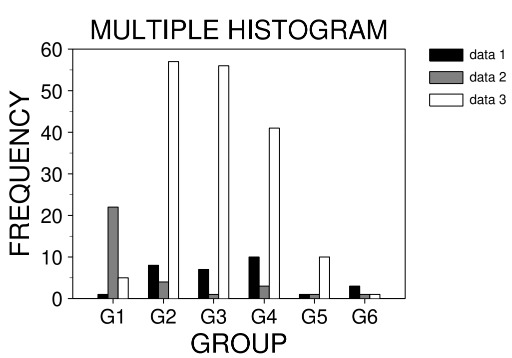
3.0.5 Histograms from raw data — `bars.top`

```
#EPSF .ps

READ_TABLE FILE 'bars.dat'
SET XY_COLUMNS 1 2
GET_BARS BAR_DX = 1.0

WORLD POSITION 12 0, WORLD_WINDOW = -999 -999 -999 -999
AXES2D
HIST2D BAR_WIDTH 0.7

RESET_CAPTIONS
CAPTION CAPTION_POSITION 1, CAPTION_TEXT 'Testing GET-BARS'
CAPTION CAPTION_POSITION 2, CAPTION_TEXT 'X'
CAPTION CAPTION_POSITION 3, CAPTION_TEXT 'N'
```



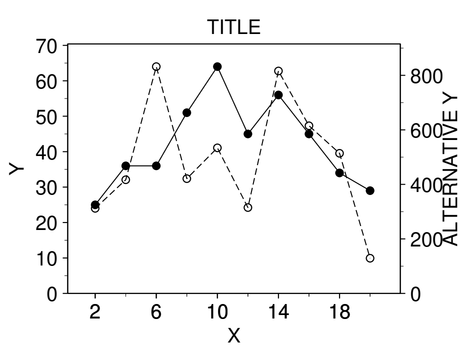
3.0.6 Multiple histograms — `mhist.top`

```
#EPSF .ps

READ_TABLE FILE 'mhist1.dat'
SET XY_COLUMNS 2 1
SET POSITION 12 0
SET WORLD_WINDOW -999 0 7 60
WORLD
SET X_TICK        1.2  1.0
SET Y_TICK_DECIMALS  -1
SET X_TICK_LABEL 1 1
SET Y_TICK_LABEL 1 2
SET BAR_WIDTH = 0.2

SET X_LABELS 'G1' 'G2' 'G3' 'G4' 'G5' 'G6', X_LABEL_STYLE = 1
AXES2D

RESET_CAPTIONS
CAPTION CAPTION_POSITION 1, CAPTION_FONT 2, CAPTION_TEXT ;
        'MULTIPLE HISTOGRAM'
CAPTION CAPTION_POSITION 2, CAPTION_FONT 2, CAPTION_TEXT 'GROUP'
CAPTION CAPTION_POSITION 3, CAPTION_FONT 2, CAPTION_TEXT 'FREQUENCY'

RESET_LEGEND
HIST2D BAR_GRAYNESS = 0.0
LEGEND SYMBOL = BAR,  DESCRIPTION = 'data 1'

READ_TABLE FILE 'mhist2.dat'
HIST2D BAR_GRAYNESS = 0.5, BAR_XSHIFT = 0.2
LEGEND SYMBOL = BAR,  DESCRIPTION = 'data 2'

READ_TABLE FILE 'mhist3.dat'
HIST2D BAR_GRAYNESS = 1.0, BAR_XSHIFT = 0.4
LEGEND SYMBOL = BAR,  DESCRIPTION = 'data 3'
```



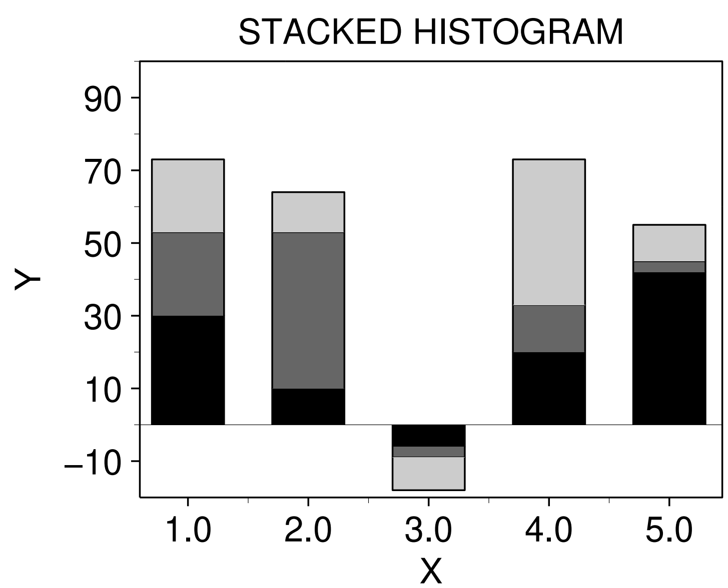
3.0.7 Y axis on the right — `alty.top`

```
#EPSF .ps

READ_TABLE FILE 'hist1.dat'
SET XY_COLUMNS 1 2
SET POSITION 12 0
WORLD WORLD_WINDOW -999 0 -999 -999

AXES2D

RESET_CAPTIONS
CAPTION CAPTION_POSITION 1, CAPTION_TEXT 'TITLE'
CAPTION CAPTION_POSITION 2, CAPTION_TEXT 'X'
CAPTION CAPTION_POSITION 3, CAPTION_TEXT 'Y'

SET PLOT2D_SYMBOL_TYPE 4, PLOT2D_LINE_TYPE = 2
PLOT2D

READ_TABLE FILE 'hist4.dat'
SET Y_TICK  -999 -999 -999
SET Y_TICK_DECIMALS  -999
SET Y_TICK_LABEL -999 -999
WORLD WORLD_WINDOW -999 0 -999 -999
SET PLOT2D_SYMBOL_TYPE 3, PLOT2D_LINE_TYPE = 5
PLOT2D

SET Y_TICK  -999 -999 -999
SET Y_TICK_DECIMALS  -999
SET Y_TICK_LABEL -999 -999
AXES2D Y_SCALE = 'RIGHT'
RESET_CAPTIONS
CAPTION CAPTION_POSITION 6, CAPTION_TEXT 'ALTERNATIVE Y'
```



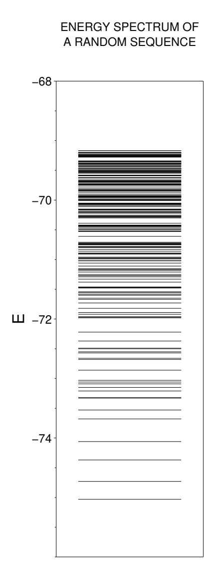
3.0.8 Stacked histograms — stkhist.top

```
#EPSF .ps

READ_TABLE FILE 'stkhist.dat'
SET NO_XY_SCOLUMNS = 1 3, XY_SCOLUMNS = 1   4   3   2
SET BAR_GRAYNESS =                        0.0 0.4 0.8
SET BAR_WIDTH = 0.6
SET POSITION 12 0
SET WORLD_WINDOW -999 -20 -999 100
WORLD
AXES2D

RESET_CAPTIONS
CAPTION CAPTION_POSITION 1, CAPTION_TEXT 'STACKED HISTOGRAM'
CAPTION CAPTION_POSITION 2, CAPTION_TEXT 'X'
CAPTION CAPTION_POSITION 3, CAPTION_TEXT 'Y'

HIST2D
```



3.0.9 Spectra plots — spectrum.top

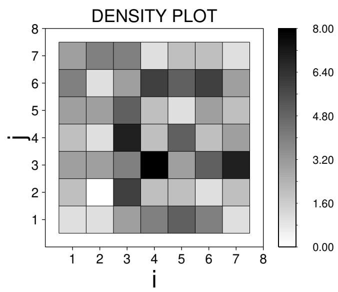
```
#EPSF .ps

SET WORLD_WINDOW = 0 -76 1 -68
SET PAPER_WINDOW = 8 2 14.5 23.0 0.0
SET X_LABEL_STYLE = 3
SET X_TICK          9 9
SET X_TICK_LABEL    9 9
SET Y_TICK       -76 0.5
SET Y_TICK_LABEL    5 4
SET Y_TICK_DECIMALS   -1
SET XY_COLUMNS 2 1
SET TICK_FONT = 2

WORLD
AXES2D

RESET_CAPTIONS
CAPTION CAPTION_POSITION 1, CAPTION_FONT = 2, CAPTION_TEXT ' '
CAPTION CAPTION_POSITION 1, CAPTION_FONT = 2, CAPTION_TEXT ' '
CAPTION CAPTION_POSITION 1, CAPTION_FONT = 2, ;
        CAPTION_TEXT 'A RANDOM SEQUENCE'
CAPTION CAPTION_POSITION 1, CAPTION_FONT = 2, ;
        CAPTION_TEXT 'ENERGY SPECTRUM OF'
CAPTION CAPTION_POSITION 3, CAPTION_FONT = 1, CAPTION_TEXT 'E'

SET BAR_WIDTH  = 0.70
SET BAR_XSHIFT = 0.15
READ_TABLE FILE = 'spectrum.dat'
SPECTRUM PLOT2D_LINE_TYPE = 2
```

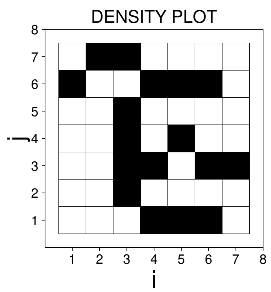


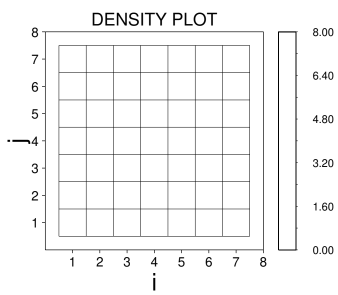
3.0.10 Density plots — `dplot.top`

```
#EPSF .ps

READ_DPLOT FILE = 'dplot.dat'
SET POSITION 9 1
SET X_TICK         1.  1.
SET Y_TICK         1.  1.
SET X_TICK_DECIMALS   -1
SET Y_TICK_DECIMALS   -1
SET X_TICK_LABEL 1 1
SET Y_TICK_LABEL 1 1
SET DPLOT_STYLE GRAY
SET DPLOT_GRAYNESS  1.0  0.0
SET DPLOT_PART FULL
SET DPLOT_LINE_TYPE 3
SET DPLOT_BOUNDS 0.0 8.0
SET BAR_LEGEND_PLACES 3 2

SET WORLD_WINDOW  -999 -999 -999 -999
WORLD
AXES2D

RESET_CAPTIONS
CAPTION CAPTION_FONT 2, CAPTION_POSITION 1, CAPTION_TEXT 'DENSITY PLOT'
CAPTION CAPTION_FONT 2, CAPTION_POSITION 2, CAPTION_FONT = 1, CAPTION_TEXT 'i'
CAPTION CAPTION_FONT 2, CAPTION_POSITION 3, CAPTION_FONT = 1, CAPTION_TEXT 'j'

DPLOT

RESET

READ_DPLOT FILE = 'dplot.dat'
SET POSITION 10 1

SET X_TICK         1.  1.
SET Y_TICK         1.  1.
SET X_TICK_DECIMALS   -1
SET Y_TICK_DECIMALS   -1
SET X_TICK_LABEL 1 1
SET Y_TICK_LABEL 1 1
SET DPLOT_STYLE BLACK
SET DPLOT_GRAYNESS  1.0  0.0
SET DPLOT_PART FULL
SET DPLOT_LINE_TYPE 3
SET DPLOT_BOUNDS 0.0 3.0

SET WORLD_WINDOW  -999 -999 -999 -999
WORLD
AXES2D

RESET_CAPTIONS
CAPTION CAPTION_FONT 2, CAPTION_POSITION 1, CAPTION_TEXT 'DENSITY PLOT'
```

```
CAPTION CAPTION_FONT 2, CAPTION_POSITION 2, CAPTION_FONT = 1, CAPTION_TEXT 'i'
CAPTION CAPTION_FONT 2, CAPTION_POSITION 3, CAPTION_FONT = 1, CAPTION_TEXT 'j'

DPLOT BAR_LEGEND = OFF

RESET

READ_DPLOT FILE = 'dplot.dat'
SET POSITION 11 1
SET X_TICK         1.  1.
SET Y_TICK         1.  1.
SET X_TICK_DECIMALS   -1
SET Y_TICK_DECIMALS   -1
SET X_TICK_LABEL 1 1
SET Y_TICK_LABEL 1 1
SET DPLOT_STYLE COLOR
# SET DPLOT_COLOR 0 0 0 0 0 1
SET DPLOT_COLOR 1 1 1   0.1 0.1 0.7
SET DPLOT_PART FULL
SET DPLOT_LINE_TYPE 3
SET DPLOT_BOUNDS 0.0 8.0
SET BAR_LEGEND_PLACES 3 2

SET WORLD_WINDOW  -999 -999 -999 -999
WORLD
AXES2D

RESET_CAPTIONS
CAPTION CAPTION_FONT 2, CAPTION_POSITION 1, CAPTION_TEXT 'DENSITY PLOT'
CAPTION CAPTION_FONT 2, CAPTION_POSITION 2, CAPTION_FONT = 1, CAPTION_TEXT 'i'
CAPTION CAPTION_FONT 2, CAPTION_POSITION 3, CAPTION_FONT = 1, CAPTION_TEXT 'j'

DPLOT
```

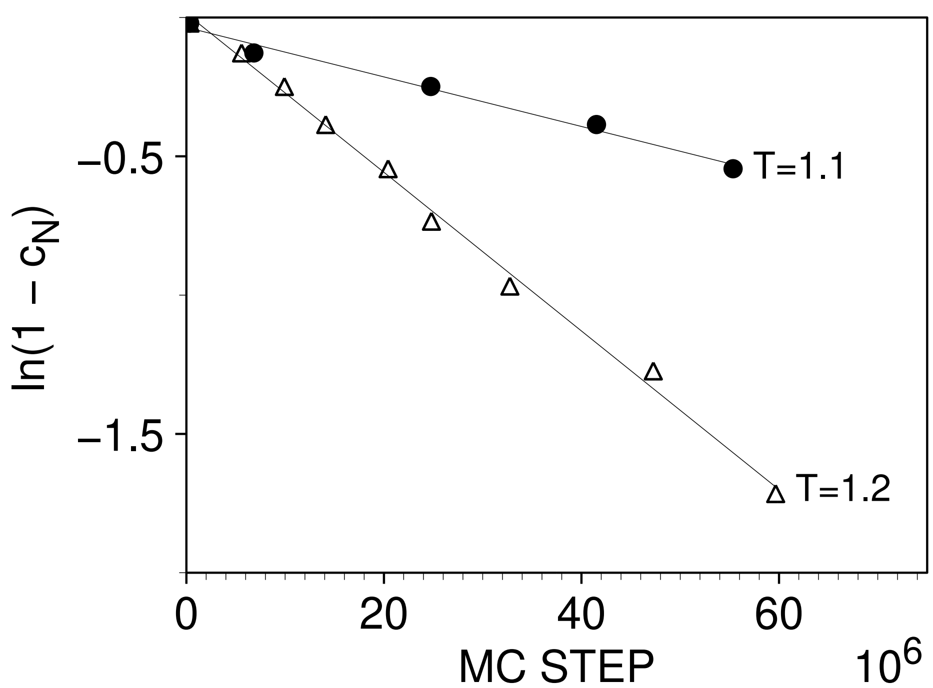

DENSITY PLOT
j
i
8.00
6.40
4.80
3.20
1.60
0.00


DENSITY PLOT
j
i


DENSITY PLOT
j
i
8.00
6.40
4.80
3.20
1.60
0.00

3.0.11 Least-squares fitting — `dplot.top`

```
#EPSF

SET XY_COLUMNS = 3 5
SET X_TICK = 0 2 -999
SET X_TICK_LABEL = 1 10
SET Y_TICK = -2.0 0.5 -999
SET Y_TICK_LABEL = 2 2
SET WORLD_WINDOW = 0 -2.0 75 0
SET PRINT_DXY = 2 0, PRINT_FONT = 4, PRINT_VERT = 2, PRINT_HORIZ = 1
SET PRINT_MODE = 'POINT'


READ_TABLE FILE = 'fit.dat'
SELECT_DATA SELECT_COLUMN = 2, SELECT_RANGE = 1.09 1.11
TRANSFORM TRF_TYPE = LINEAR, TRF_PARAMETERS = 0 0.000001, ;
          NO_XY_SCOLUMNS = 1 0, XY_SCOLUMNS = 3
SET POSITION 14 0
WORLD
AXES2D
PLOT2D PLOT2D_LINE_TYPE = 0, PLOT2D_SYMBOL_TYPE = 4
FIT FIT_MODEL = 'POLYNOMIAL', ERROR_COLUMN = 0, ;
    FIT_PARAM_INITIAL = 1 -1 0 0 0 0 0, FIT_PARAM_INDICES = 1 2
PLOT2D PLOT2D_LINE_TYPE = 3, PLOT2D_SYMBOL_TYPE = 0
RESET_CAPTIONS
PRINT PRINT_POINT = 0, PRINT_TEXT = 'T=1.1'
CAPTION CAPTION_POSITION 2, CAPTION_TEXT = 'MC STEP'
CAPTION CAPTION_POSITION 3, CAPTION_TEXT = 'ln(1 - c_N_)'
CAPTION CAPTION_POSITION 4, CAPTION_TEXT = '10^6^'


READ_TABLE FILE = 'fit.dat'
SELECT_DATA SELECT_COLUMN = 2, SELECT_RANGE = 1.2 1.2
TRANSFORM TRF_TYPE = LINEAR, TRF_PARAMETERS = 0 0.000001, ;
          NO_XY_SCOLUMNS = 1 0, XY_SCOLUMNS = 3
PLOT2D PLOT2D_LINE_TYPE = 0, PLOT2D_SYMBOL_TYPE = 7
FIT FIT_MODEL = 'POLYNOMIAL', ERROR_COLUMN = 0, ;
    FIT_PARAM_INITIAL = 0 -1 0 0 0 0 0, FIT_PARAM_INDICES = 1 2
PLOT2D PLOT2D_LINE_TYPE = 3, PLOT2D_SYMBOL_TYPE = 0
PRINT PRINT_POINT = 0, PRINT_TEXT = 'T=1.2'

# CAPTION CAPTION_POSITION 1, CAPTION_TEXT ='SEQ 43; CR MC; 70E6, B_o_=-2'
```

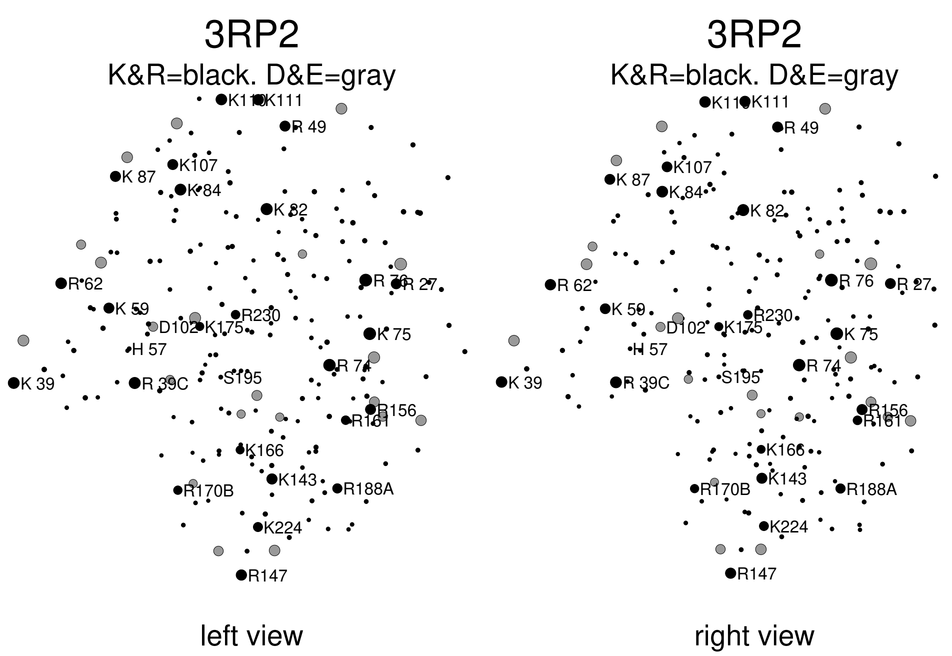


−0.5
−1.5
ln(1 − c_N)
0
20
40
60
MC STEP
10^6
T=1.1
T=1.2

3.0.12 Protein C_α plot — 3rp2.top

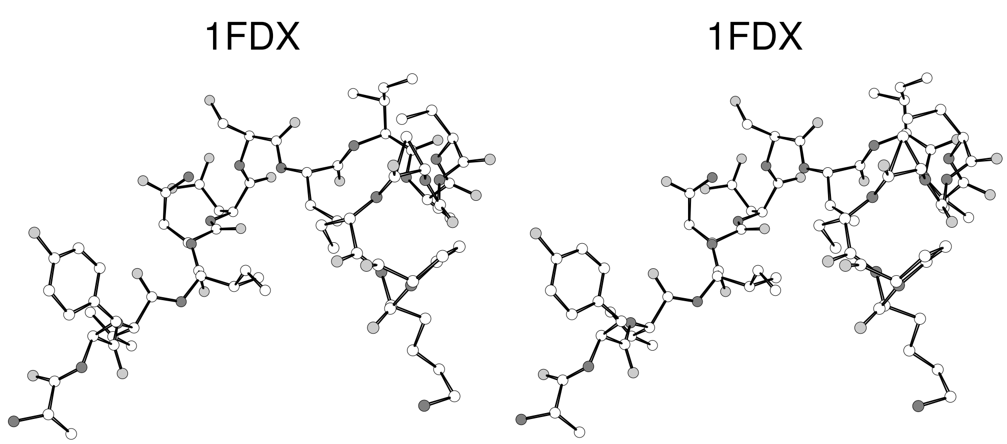
```
#EPSF .ps

# Set some general plot variables:
SET PERSPECTIVE   = 'ON', EYE_TO_SCREEN = 90.0, SCREEN_TO_TOP = 0.0
SET RADIUS_FACTOR = 0.12
# SET BOND_WIDTH_FACTOR = 0.2
SET ATOM_TYPE = 'CA'
SET FROM = 'ALL'
SET SELECTION_MODE = 'ATOM'
SET SELECTION_SEARCH = 'SEGMENT'

# Read and orient the PDB structure
READ_PDB FILE '3rp2.ca'

ROTATE_MOL ROTATION_MATRIX = -0.17317 -0.900  0.3992 ;
                             -0.5579   0.4237 0.7136 ;
                             -0.8116  -0.099 -0.5757

SET ATOM_LINE = 3, BOND_LINE = 3
SET SELECTION_SEGMENT = 'X' 'X'

# Label active site residues:
SELECT_ATOMS SELECTION_SEGMENT =  '57'  '57', RES_TYPE 'ALL', FOR = 'LABELLING',;
             SELECTION_STATUS = 'INITIALIZE'
SELECT_ATOMS SELECTION_SEGMENT = '102' '102', RES_TYPE 'ALL', FOR = 'LABELLING',;
             SELECTION_STATUS = 'ADD'
SELECT_ATOMS SELECTION_SEGMENT = '195' '195', RES_TYPE 'ALL', FOR = 'LABELLING',;
             SELECTION_STATUS = 'ADD'

# select the atoms for displaying and bonding; make virtual bonds
SET RADIUS = 1.5, ATOM_COLOR = 0.0
SELECT_ATOMS SELECTION_SEGMENT = 'X' 'X', RES_TYPE 'ALL',;
             FOR = 'DISPLAYING BONDING', SELECTION_STATUS = 'ADD'

# Colour positive charges black, label them, large spheres
SET RADIUS  = 3.5, ATOM_COLOR = 0.0
SELECT_ATOMS RES_TYPE 'ARG LYS', FOR = 'DISPLAYING LABELLING', ;
             SELECTION_STATUS = 'ADD'

# Colour negative charges gray, large spheres
SET RADIUS = 3.5, ATOM_COLOR = 0.6
SELECT_ATOMS RES_TYPE 'ASP GLU', FOR = 'DISPLAYING', ;
             SELECTION_STATUS = 'ADD'


MAKE_BONDS BOND_TYPE = 'SEQUENTIAL', ADD_BONDS = 'OFF'

# Draw a full page mono plot, with selected labels
SET PAPER_WINDOW  = 1.5 17 10.5 26 0
ROTATE_MOL_AXIS AXIS = 0 1 0, DEGREES = +3
WORLD
```

```
BALL_STICK
LABEL_ATOMS
# Title
RESET_CAPTIONS
CAPTION CAPTION_POSITION 1, CAPTION_FONT 3, CAPTION_TEXT ;
       'K&R=black. D&E=gray'
CAPTION CAPTION_POSITION 1, CAPTION_FONT 2, CAPTION_TEXT ;
        '3RP2'
CAPTION CAPTION_POSITION 2, CAPTION_FONT 3, CAPTION_TEXT 'left view'

# STOP

# Produce the right image on a new page for a stereo pair
SET PAPER_WINDOW  = 11 17 20 26 0
ROTATE_MOL_AXIS AXIS = 0 1 0, DEGREES = -6
WORLD
BALL_STICK
LABEL_ATOMS
RESET_CAPTIONS
CAPTION CAPTION_POSITION 1, CAPTION_FONT 3, CAPTION_TEXT ;
        'K&R=black. D&E=gray'
CAPTION CAPTION_POSITION 1, CAPTION_FONT 2, CAPTION_TEXT ;
        '3RP2'
CAPTION CAPTION_POSITION 2, CAPTION_FONT 3, CAPTION_TEXT 'right view'
```



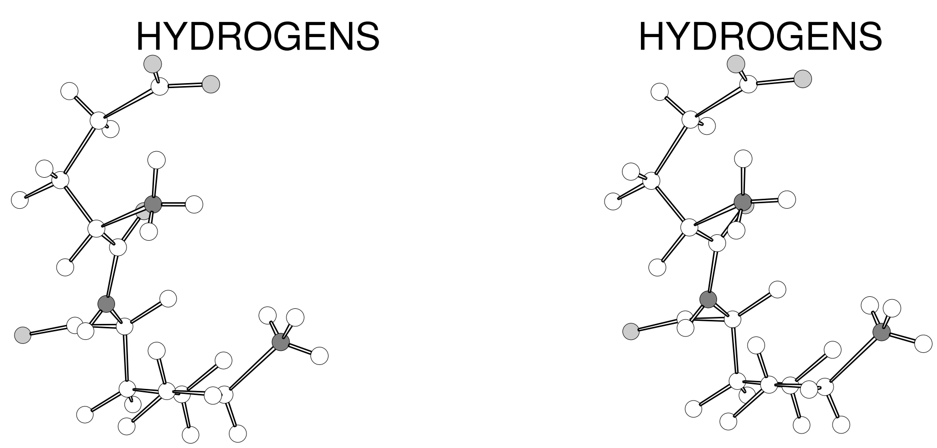
3.0.13 Protein heavy atom plot — `1fdx.top`

```
#EPSF .ps

# Set some general plot variables:
SET PERSPECTIVE = 'ON', EYE_TO_SCREEN = 90.0, SCREEN_TO_TOP = 0.0

# Read and orient the PDB structure
READ_PDB FILE '1fdx.atm'

# select the atoms for displaying and bonding; make bonds
SELECT_ATOMS SELECTION_SEGMENT = 'X' 'X', RES_TYPE 'ALL', ATOM_TYPE 'ALL', ;
             SELECTION_STATUS = 'INITIALIZE', FOR = 'DISPLAYING BONDING'
DEFAULT_ATOM_COLOR
MAKE_BONDS BOND_TYPE = 'COVALENT', ADD_BONDS = 'OFF'

# Draw a full page mono plot
SET PAPER_WINDOW  = 1.5 17 10.5 24 0
WORLD
BALL_STICK
LABEL_ATOMS
RESET_CAPTIONS
CAPTION CAPTION_POSITION 1, CAPTION_FONT 2, CAPTION_TEXT '1FDX'

SET PAPER_WINDOW  = 11 17 20 24 0
ROTATE_MOL_AXIS AXIS = 0 1 0, DEGREES = -6
WORLD
BALL_STICK
LABEL_ATOMS
RESET_CAPTIONS
CAPTION CAPTION_POSITION 1, CAPTION_FONT 2, CAPTION_TEXT '1FDX'
```



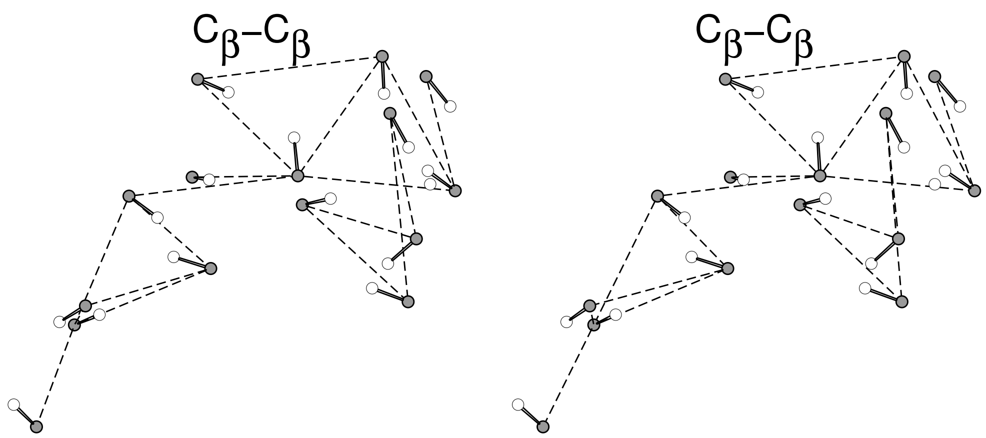
3.0.14 Protein H atom plot — hydr.top

```
#EPSF .ps

# Set some general plot variables:
SET PERSPECTIVE = 'ON', EYE_TO_SCREEN = 90.0, SCREEN_TO_TOP = 0.0

# Read and orient the PDB structure
READ_PDB FILE 'hydr.pdb'

# select the atoms for displaying and bonding; make bonds
SELECT_ATOMS SELECTION_SEGMENT = 'X' 'X', RES_TYPE 'ALL', ATOM_TYPE 'ALL', ;
             SELECTION_STATUS = 'INITIALIZE', FOR = 'DISPLAYING BONDING'
DEFAULT_ATOM_COLOR
MAKE_BONDS BOND_TYPE = 'COVALENT', ADD_BONDS = 'OFF'

# Draw a full page mono plot
SET PAPER_WINDOW  = 1.5 17 10.5 24 0
WORLD
BALL_STICK
LABEL_ATOMS
RESET_CAPTIONS
CAPTION CAPTION_POSITION 1, CAPTION_FONT 2, CAPTION_TEXT 'HYDROGENS'

SET PAPER_WINDOW  = 11 17 20 24 0
ROTATE_MOL_AXIS AXIS = 0 1 0, DEGREES = -6
WORLD
BALL_STICK
LABEL_ATOMS
RESET_CAPTIONS
CAPTION CAPTION_POSITION 1, CAPTION_FONT 2, CAPTION_TEXT 'HYDROGENS'
```



3.0.15 Protein CB–CB contacts — cbeta.top

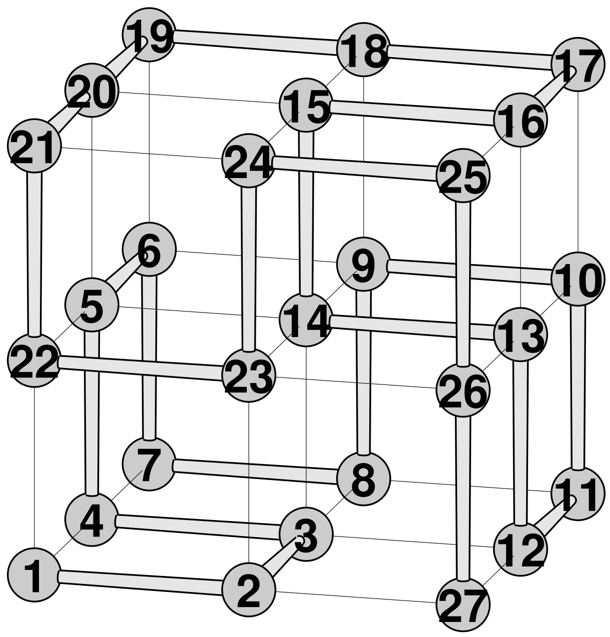
```
#EPSF .ps

# Set some general plot variables:
# SET PERSPECTIVE = 'ON', EYE_TO_SCREEN = 90.0, SCREEN_TO_TOP = 0.0

# Read and orient the PDB structure
READ_PDB FILE '1fdx.atm'

# select CA atoms
SELECT_ATOMS SELECTION_SEGMENT = 'X' 'X', RES_TYPE 'ALL', ATOM_TYPE 'CA', ;
             SELECTION_STATUS = 'INITIALIZE', FOR = 'DISPLAYING'
DEFAULT_ATOM_COLOR

# select CB atoms:
SELECT_ATOMS SELECTION_SEGMENT = 'X' 'X', RES_TYPE 'ALL', ATOM_TYPE 'CB', ;
             SELECTION_STATUS = 'ADD', FOR = 'DISPLAYING', ATOM_COLOR = 0.6

# make CA-CA bonds:
SELECT_ATOMS SELECTION_SEGMENT = 'X' 'X', RES_TYPE 'ALL', ATOM_TYPE 'CA', ;
             SELECTION_STATUS = 'INITIALIZE', FOR = 'BONDING'
MAKE_BONDS BOND_TYPE = 'SEQUENTIAL', ADD_BONDS = 'OFF'

# make CA-CB bonds:
SELECT_ATOMS SELECTION_SEGMENT = 'X' 'X', RES_TYPE 'ALL', ATOM_TYPE 'CA CB', ;
             SELECTION_STATUS = 'INITIALIZE', FOR = 'BONDING'
MAKE_BONDS BOND_TYPE = 'COVALENT', ADD_BONDS = 'ON'


# make CB-CB `bonds'
SELECT_ATOMS SELECTION_SEGMENT = 'X' 'X', RES_TYPE 'ALL', ATOM_TYPE 'CB', ;
             SELECTION_STATUS = 'INITIALIZE', FOR = 'BONDING'
MAKE_BONDS BOND_TYPE = 'COVALENT', ADD_BONDS = 'ON', ;
           COVALENT_BOND = 5.5 7.5, BOND_LINE = -5

# Draw a full page mono plot
SET PAPER_WINDOW  = 1.5 17 10.5 24 0
WORLD
BALL_STICK
LABEL_ATOMS
RESET_CAPTIONS
CAPTION CAPTION_POSITION 1, CAPTION_FONT 2, CAPTION_TEXT 'C_@b@_-C_@b@_'

SET PAPER_WINDOW  = 11 17 20 24 0
ROTATE_MOL_AXIS AXIS = 0 1 0, DEGREES = -6
WORLD
BALL_STICK
LABEL_ATOMS
RESET_CAPTIONS
CAPTION CAPTION_POSITION 1, CAPTION_FONT 2, CAPTION_TEXT 'C_@b@_-C_@b@_'
```


Cβ–Cβ
Cβ–Cβ

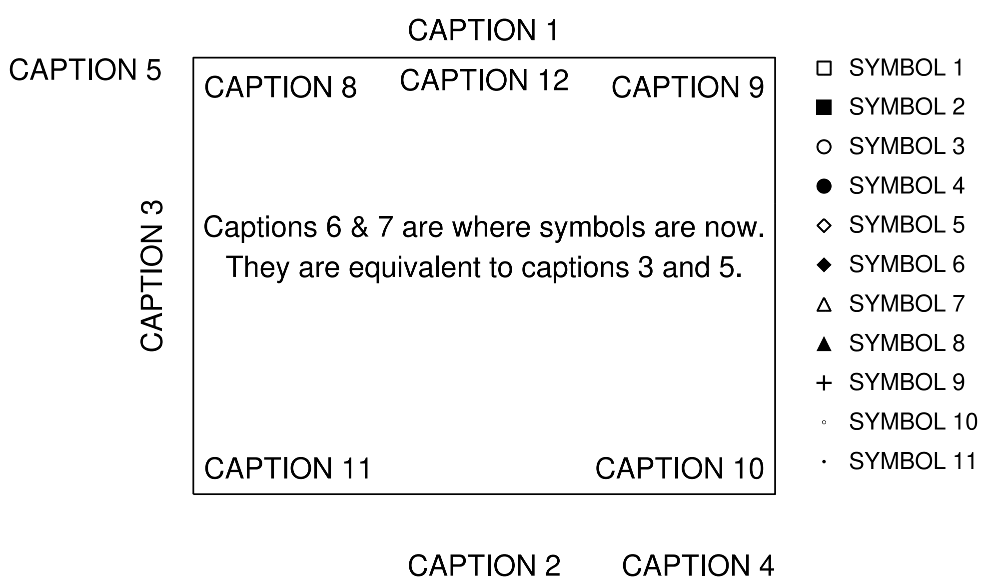
3.0.16 Lattice model plot — `lattice.top`

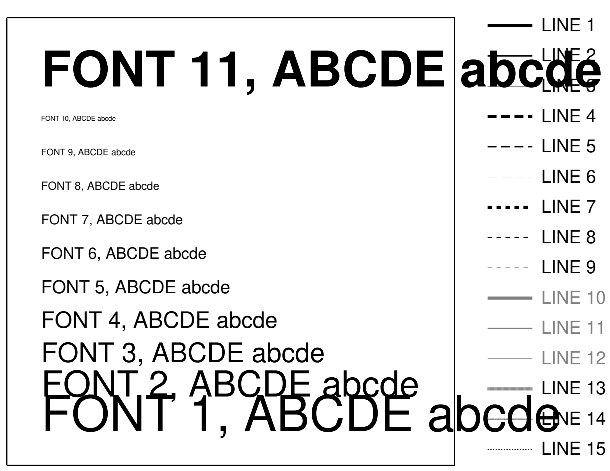
```
#EPSF .ps

# Set some general plot variables:
SET ATOM_TYPE = 'ALL'
SET FROM = 'ALL'
SET SELECTION_MODE = 'ATOM'
SET SELECTION_SEARCH = 'SEGMENT'
SET SELECTION_SEGMENT = 'X' 'X'
SET LABEL_LOCATION = 1
SET LABEL_STYLE = 'ATOM_INDEX'
SET LABEL_FONT = 12

# Read and orient the PDB structure
READ_PDB PDB_FORMAT = 'XYZ', FILE 'lattice.dat'
# READ_PDB FILE 'lattice.atm'

ROTATE_MOL_AXIS AXIS = 1 0 0, DEGREES = -90
ROTATE_MOL_AXIS AXIS = 0 1 0, DEGREES = -15
ROTATE_MOL_AXIS AXIS = 1 0 0, DEGREES =  15

SELECT_ATOMS SELECTION_STATUS='INITIALIZE', FOR = 'BONDING'
SET BOND_WIDTH_FACTOR = 0.10, BOND_COLOR = 0.00, BOND_LINE = -3
SET COVALENT_BOND = 0.9 1.1
MAKE_BONDS BOND_TYPE = 'COVALENT', ADD_BONDS = 'OFF'

SET RADIUS_FACTOR = 0.08, ATOM_COLOR = 0.8, ATOM_LINE = 2
SELECT_ATOMS SELECTION_STATUS='INITIALIZE', FOR = 'DISPLAYING BONDING LABELLING'

SET BOND_WIDTH_FACTOR = 0.320, BOND_COLOR = 0.9, BOND_LINE = 2
MAKE_BONDS BOND_TYPE = 'SEQUENTIAL', ADD_BONDS = 'ON'

SET POSITION = 14 1
WORLD
BALL_STICK
LABEL_ATOMS

POSTSCRIPT POSTSCRIPT_TEXT = ;
           '/font11 {/Helvetica-Bold findfont 0.125 scalefont setfont} def'

RESET_CAPTIONS
CAPTION CAPTION_POSITION = 5, CAPTION_FONT = 11, CAPTION_TEXT = '(a)'
```

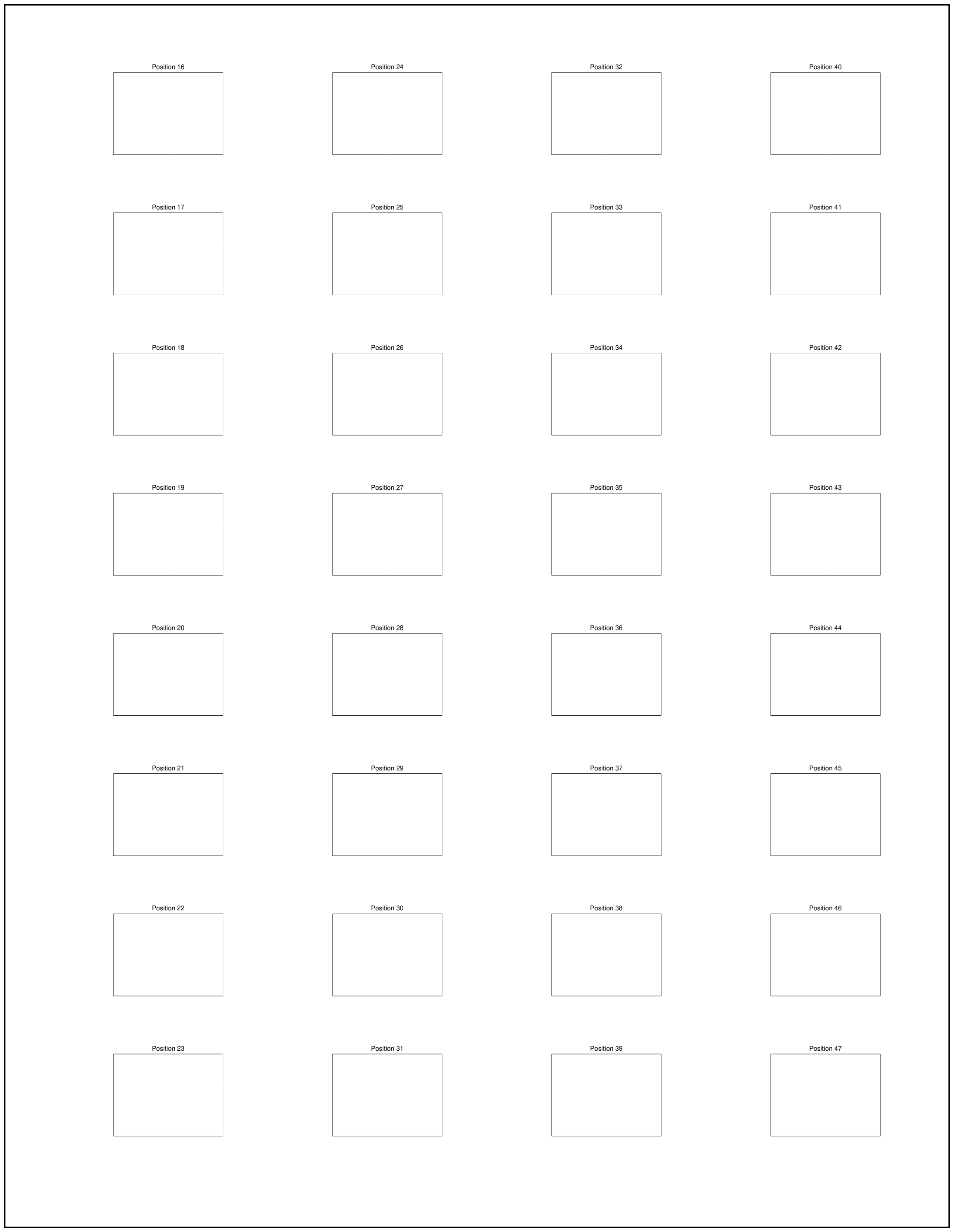
(a)

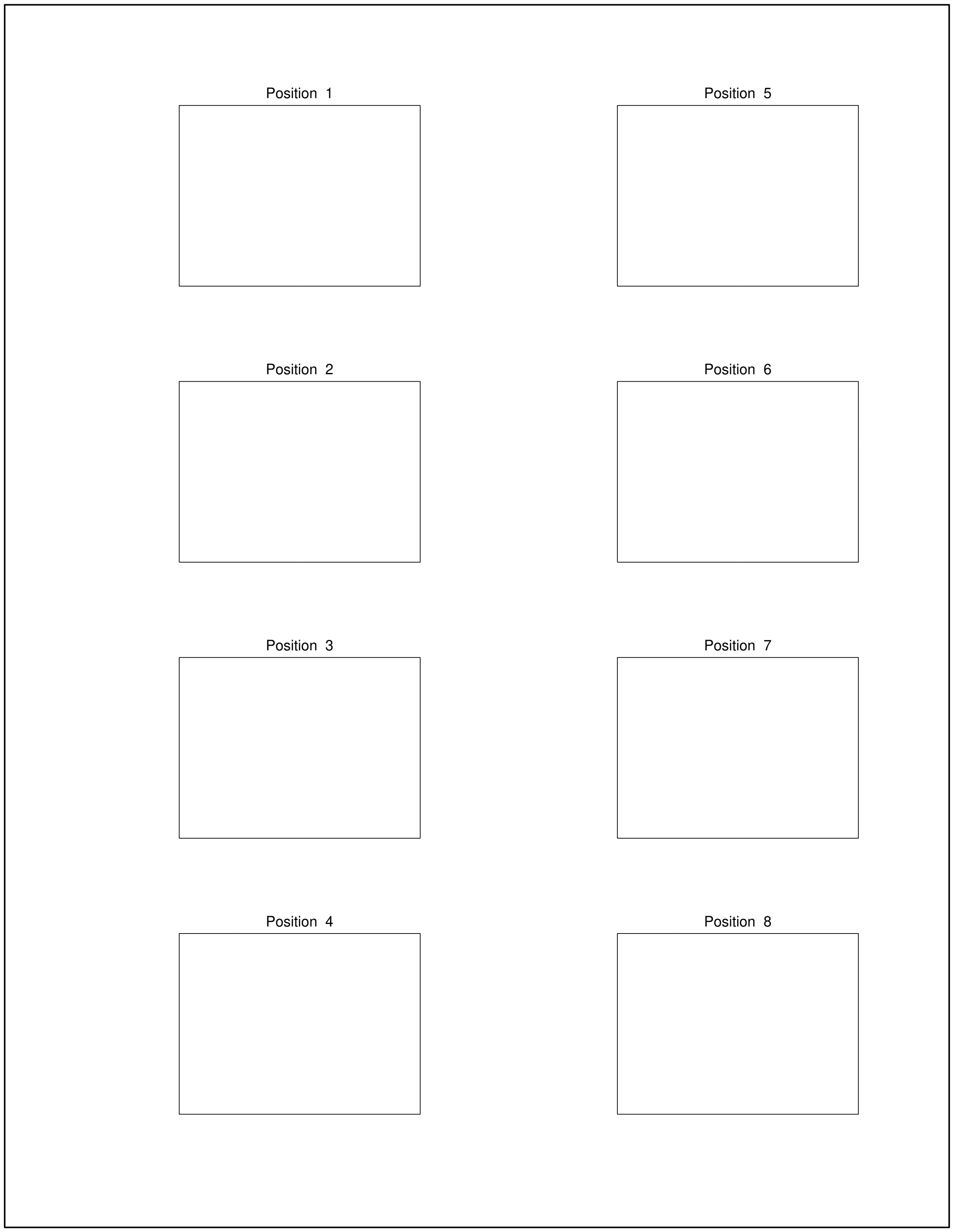


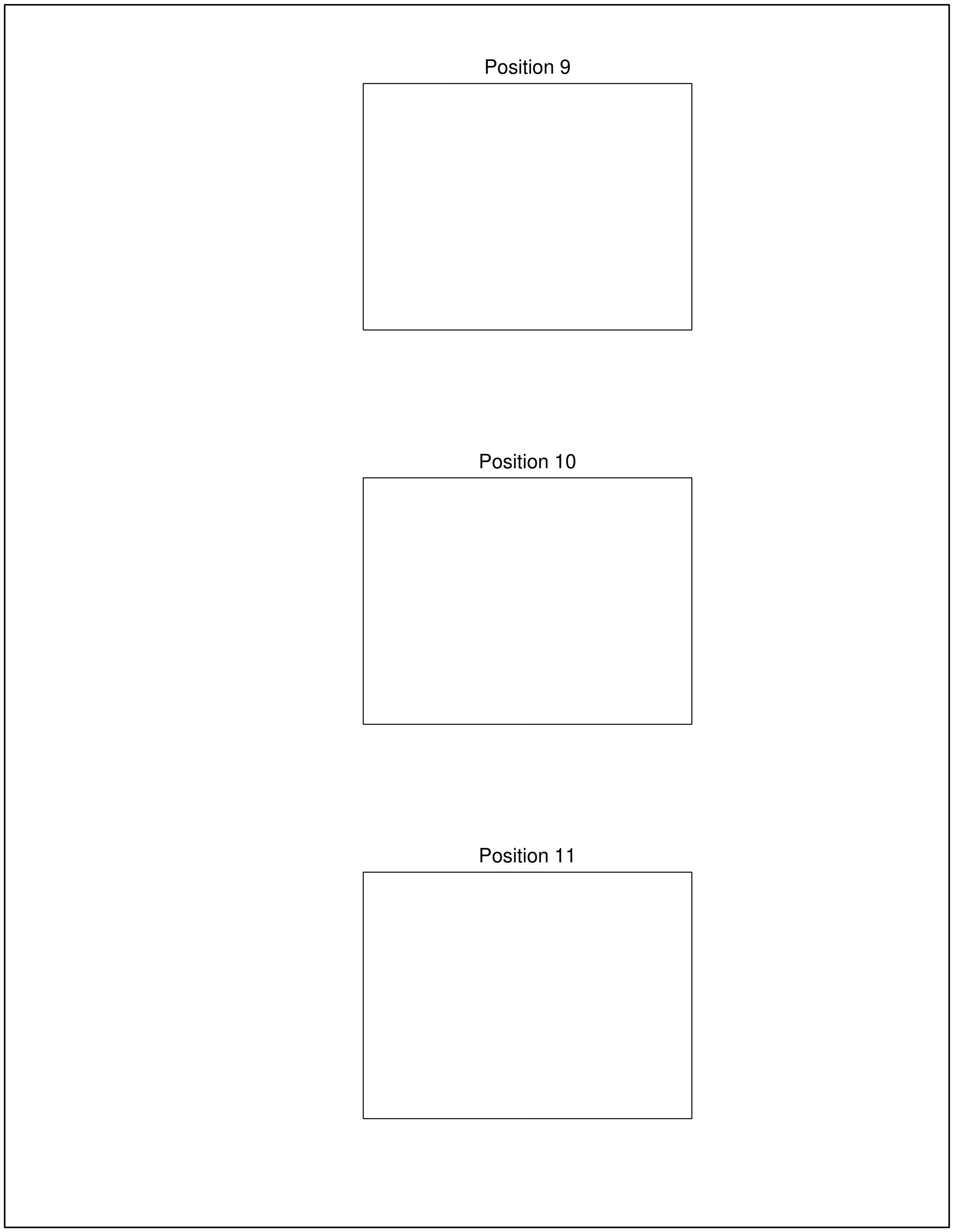
3.0.17 ASGL fonts, symbols, line types and standard plot positions

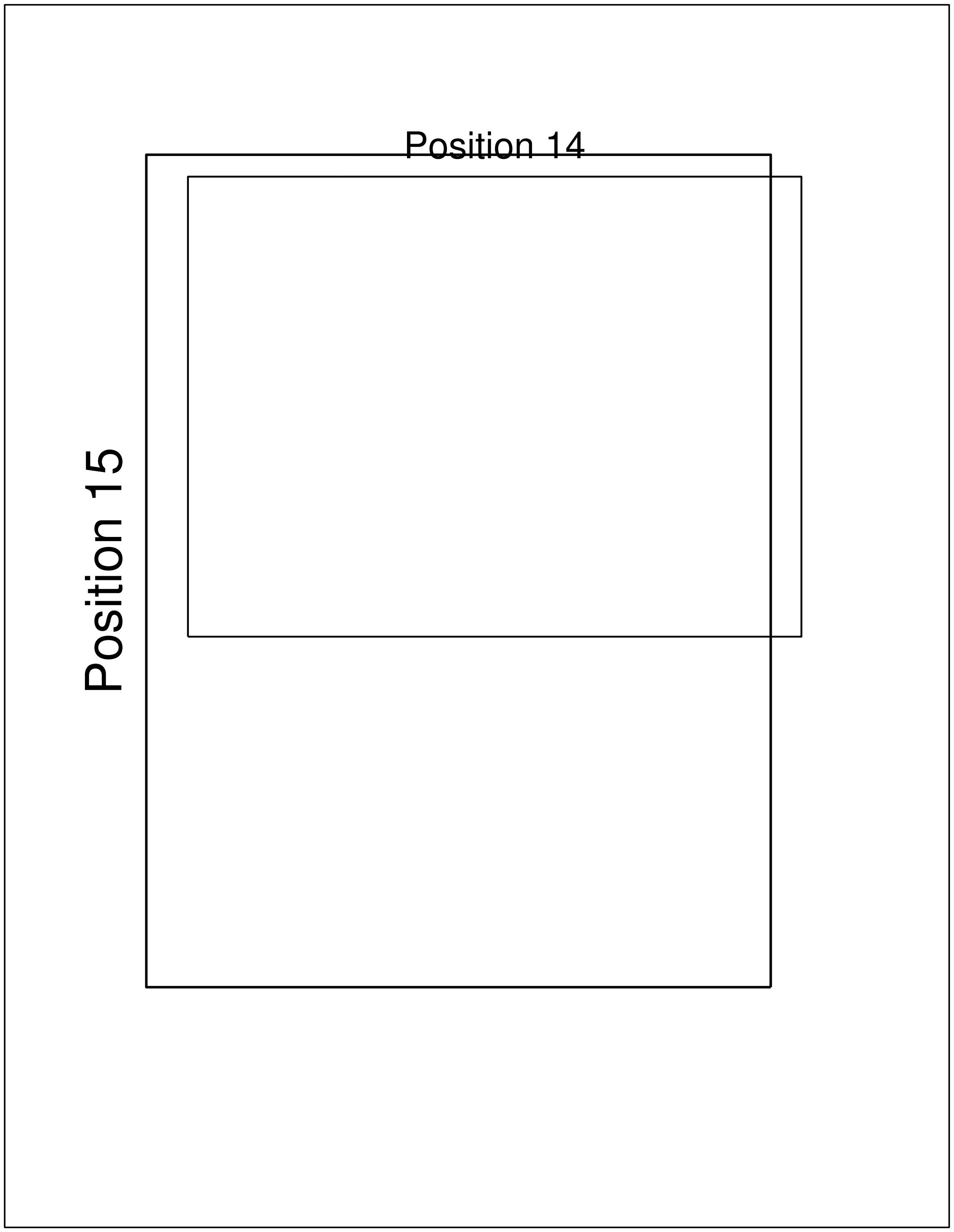





Position 16
Position 24
Position 32
Position 40
Position 17
Position 25
Position 33
Position 41
Position 18
Position 26
Position 34
Position 42
Position 19
Position 27
Position 35
Position 43
Position 20
Position 28
Position 36
Position 44
Position 21
Position 29
Position 37
Position 45
Position 22
Position 30
Position 38
Position 46
Position 23
Position 31
Position 39
Position 47


Position 1
Position 5
Position 2
Position 6
Position 3
Position 7
Position 4
Position 8


Position 9
Position 10
Position 11


Position 14
Position 15

Chapter 4

Top language

Top is an interpreter of a scripting language specialized for certain areas. Its use includes programs Modeller and Asgl. Its syntax resembles that of Fortran.

4.1 The source file

Each Top program or include file is stored in a file named `'program.top'`. The `.top` extension is mandatory.

The Top program consists of a series of commands. The order of commands is important. An example of the Top program that writes integers 1 to 10 to the output file is:

```
# Define a variable:
DEFINE_INTEGER VARIABLES = IVAR

# Open a file for appending
OPEN IO_UNIT = 21, OBJECTS_FILE = 'output.file', FILE_ACCESS = 'APPEND'

# Loop from 1 to 10:
DO  IVAR = 1, 10, 1
  # Append IVAR to the output file:
  WRITE IO_UNIT = 21, OBJECTS = IVAR
END_DO

# Close a file
CLOSE IO_UNIT = 11

# Exit:
STOP
```

There can be at most one command per line. Each command or line can be at most `LENACT` (512) characters long. The command can extend over several lines if a continuation character ';' is used to indicate the end of the current line. Everything on that line after the continuation character is ignored.

A comment character '#' can be used anywhere on the line to ignore everything that occurs after the comment character.

Blank lines are allowed. They are ignored.

TAB characters are replaced by blank characters.

TOP converts all commands to upper case, except for the string constants that are quoted in single quotes '. Thus, TOP is case insensitive, except for the quoted strings.

There are two groups of commands: flow control commands and commands that perform certain tasks. The next two sections describe the flow control commands and those 'performing' commands that are an integral part of TOP. There are also additional commands specific to each application of TOP, such as MODELLER and ASGL, which are described elsewhere.

The usual UNIX conventions are used for typesetting the rules. Table 4.1 explains the shorthand used to describe different variables and constants:

⟨integer : 1⟩	an integer variable or constant
⟨real : 1⟩	a real variable or constant
⟨string : 1⟩	a string variable or constant
⟨logical : 1⟩	a logical variable or constant
⟨var_ : 1⟩	prefix for a variable
⟨const_ : 1⟩	prefix for a constant
⟨variable : 1⟩	⟨var_integer : 1⟩ | ⟨var_real : 1⟩ | ⟨var_string : 1⟩ | ⟨var_logical : 1⟩
⟨constant : 1⟩	⟨const_integer : 1⟩ | ⟨const_real : 1⟩ | ⟨const_string : 1⟩ | ⟨const_logical : 1⟩
⟨number : 1⟩	⟨integer : 1⟩ | ⟨real : 1⟩
⟨quantity : 1⟩	⟨variable : 1⟩ | ⟨constant : 1⟩
⟨quantity : 0⟩	a vector of any length with elements ⟨quantity : 1⟩
⟨quantity : N⟩	a vector of N elements ⟨quantity : 1⟩

Table 4.1: *List of variable types in* TOP.

All the variables are formally vectors. When a variable is referred to in a scalar context its first element is used. All elements of one vector are of the same type. All variables, including a vector of the variable length, must have at least one element.

There are four different variable types: integer, real, string and logical.

The real constant is (FORTRAN real number representation):

```
[+|-][digits][.][digits][{e|E|d|D}{+|-}digits]
```

The integer constant is (FORTRAN integer number representation):

```
[+|-][digits]
```

The logical constant can be either `on` or `off` (case insensitive).

The string constant can contain any character except for a prime '. It can be optionally enclosed in primes. If it is not quoted it is converted to upper case and its extent is determined by the position of the blanks on each side of the contiguous string of non-blank characters.

4.2 TOP Commands

There are 'flow control' and 'performing' commands. If general, the 'performing' commands have the following syntax:

ACTION [**ASSIGNMENT**, **ASSIGNMENT**, ..., **ASSIGNMENT**]

ACTION specifies what action to take. **ASSIGNMENT** sets the variable to the specified value. The values assigned in this way are kept until the next assignment. For example, `CALL ROUTINE = 'routine_name', IVAR = 3` sets the integer variable **IVAR** to 3 and then calls routine `routine_name`; if **IVAR** is not changed in the routine, its value will remain to be 3 after the call to the routine.

There can be any number of assignments in a command. They must be separated by commas. The assignment is of the form:

⟨`variable` : 0⟩ = [-]⟨`quantity` : 0⟩

The '=' character is optional (can be replaced with a blank).

⟨`integer` : 1⟩ and ⟨`real` : 1⟩ can be assigned to each other. When a real number is assigned to an integer variable, the decimal places are ignored. That is, the result is the same as if the FORTRAN function `IFIX()` was used. There must be no space between the optional − and ⟨`quantity` : 0⟩. If a vector variable is assigned to a variable, all its elements are used.

Real, integer, and logical variables can also be assigned to a string variable. The conversion of a real variable to a string value is guided by the TOP variable NUMBER_PLACES which is of type ⟨`integer` : 2⟩. The first element of NUMBER_PLACES sets the number of places before the decimal point, and the second element the number of places after the decimal point. If the latter is −1, an integer number without a decimal point is obtained, if 0 there is a decimal point without any decimal places.

Assignments can follow any command, except **DO**, **END_DO**, **GO_TO**, **LABEL**, **STOP**, and **END_SUBROUTINE**.

4.2.1 DEFINE_INTEGER — define integer variables

Options:

VARIABLES = ⟨`string` : 0⟩	''	variable names

Description: This command defines user integer variables. All variables used in the TOP program must be defined. An exception are the pre-defined TOP variables listed at the end of this section.

4.2.2 DEFINE_LOGICAL — define logical variables

Options:

VARIABLES = ⟨`string` : 0⟩	''	variable names

Description: This command defines user logical variables.

4.2.3 DEFINE_REAL — define real variables

Options:

VARIABLES = ⟨`string` : 0⟩	''	variable names

Description: This command defines user real variables.

4.2.4 DEFINE_STRING — define string variables

Options:

VARIABLES = ⟨string : 0⟩	''	variable names

Description: This command defines user string variables.

4.2.5 SET — set variable

Command: SET [ASSIGNMENT, [ASSIGNMENT, . . . [ASSIGNMENT]]]

Description: This command sets the values of variables of any of the four types. See the description of **ASSIGNMENT** above.

There can be UNIX shell environment variables in any input or output filename. The environment variables have to be in the format `${VARNAME}` or `$(VARNAME)`. Also, four predefined macros are available for string variables:

Four predefined macros are available for string variables:

- `'${LIB}'` is expanded into `$LIB_APPLICATION` shell environment variable, where APPLICATION is the name-version of the program (*e.g.*, MODELLER5);
- `'${DIR}'` is expanded into the TOP variable DIRECTORY,
- `'${JOB}'` is expanded into the root of the TOP script filename,
- `'${DEFAULT}'` is expanded into `(ROOT_NAME)(FILE_ID)(ID1)(ID2)(FILE_EXT)`, where ROOT_NAME, FILE_ID, ID1, ID2, and FILE_EXT are TOP variables. FILE_ID is a string that may be set to `'default'`. In that case, a hard-wired short string is used instead of FILE_ID. Otherwise, the explicitly specified FILE_ID is applied instead. In any case, FILE_ID is not modified by the filename generation routine so that it can be used more than once without resetting it to the `'default'` value. Four digits are used for both ID1 and ID2. For example, `'2ptn.B99990001'` results from ROOT_NAME = `'2ptn'`, FILE_EXT = `'.B'`, ID1 = 9999, and ID2 = 1.

4.2.6 OPERATE — perform mathematic operation

Options:

OPERATION = ⟨string : 1⟩	'SUM'	operation to perform: SUM | MULTIPLY | DIVIDE | POWER | MOD
RESULT = ⟨string : 0⟩	''	variable name for the result of operation
ARGUMENTS = ⟨real : 0⟩	0.00	real arguments to the math operation

Description: This command performs a specified mathematical operation. There can be up to `MRPRM` (120) operands for the `'SUM'` and `'MULTIPLY'` operations, but only two for `'DIVIDE'` and `'POWER'`. The RESULT value has to be the name of a real variable.

4.2.7 STRING_OPERATE — perform string operation

Options:

OPERATION = ⟨string : 1⟩	'SUM'	operation to perform: CONCATENATE
RESULT = ⟨string : 0⟩	''	variable name for the result of operation
STRING_ARGUMENTS = ⟨string : 0⟩	''	arguments for string operation

Description: This command performs a specified string operation. There can be up to MSPRM (130) operands for the CONCATENATE operation. The RESULT value has to be a name of the string variable.

4.2.8 RESET — reset TOP

Description: This command resets the internal state of TOP and its predefined variables to their initial values. It does this by calling the initialization routine that reads the 'top.ini' file. This command also undefines all user defined variables.

4.2.9 OPEN — open input file

Options:

IO_UNIT = ⟨integer : 1⟩	21	IO unit for file operations
OBJECTS_FILE = ⟨string : 1⟩	'top.out'	filename
FILE_ACCESS = ⟨string : 1⟩	'SEQUENTIAL'	file access: SEQUENTIAL | APPEND
FILE_STATUS = ⟨string : 1⟩	'UNKNOWN'	file status: UNKNOWN | OLD | NEW

Description: This command opens a specified file on the specified I/O stream for formatted access. FORTRAN conventions apply to FILE_ACCESS and FILE_STATUS.

4.2.10 WRITE — write TOP objects

Options:

IO_UNIT = ⟨integer : 1⟩	21	IO unit for file operations
OBJECTS = ⟨string : 0⟩	''	variable names or constants
NUMBER_PLACES = ⟨integer : 2⟩	5 2	pre- and post-decimal point places
OUTPUT_DIRECTORY = ⟨string : 1⟩	''	output directory

Description: This command writes the specified objects to a single line which is then written to a selected I/O stream. Each element of the OBJECTS vector is first tested if it is a name of a variable of any type. If it is the contents of that variable is written out. If it is not, the element is treated as a string constant. The first and second element of NUMBER_PLACES set the numbers of places before and after the decimal point, respectively, for real and integer objects.

4.2.11 READ — read record from input file

Options:

IO_UNIT = ⟨integer : 1⟩	21	IO unit for file operations
RECORD = ⟨string : 1⟩	'undefined'	contents of the input line

Description: This command reads a line from the file on the I/O channel IO_UNIT. The line goes into the string variable RECORD.

4.2.12 CLOSE — close an input file

Options:

IO_UNIT = ⟨integer : 1⟩	21	IO unit for file operations

Description: This command closes a specified I/O stream.

4.2.13 WRITE_TOP — write the TOP program

Options:

FILE = ⟨string : 1⟩	'counter'	partial or complete filename
OUTPUT_DIRECTORY = ⟨string : 1⟩	''	output directory
FILE_ACCESS = ⟨string : 1⟩	'SEQUENTIAL'	file access: SEQUENTIAL | APPEND

Description: This command writes the current TOP program in memory to a specified file.

4.2.14 SYSTEM — execute system command

Options:

COMMAND = ⟨string : 1⟩	'nothing'	UNIX command

Description: This command executes the specified UNIX command.

4.2.15 INQUIRE — check if file exists

Options:

FILE = ⟨string : 1⟩	'counter'	partial or complete filename

Description: This command assigns 1 to FILE_EXISTS if the specified file exists, otherwise it assigns 0. You can use it with a subsequent **IF** command for the flow control.

4.2.16 GO_TO — jump to label

Command: GO_TO ⟨`string` : 1⟩

Description: The 'go_to' statement, which transfers execution to the Top statement occurring after the **LABEL** statement with the same name.

4.2.17 LABEL — place jump label

Command: LABEL ⟨`string` : 1⟩

Description: This command labels a target position for the **GO_TO** statement with the same name.

4.2.18 INCLUDE — include Top file

Options:

INCLUDE_FILE = ⟨`string` : 1⟩	`'__asgl'`	include file name

Description: This command includes a Top file INCLUDE_FILE. You do not have to specify the `.top` extension. First, the given filename is tried. Second, the directory specified in the `$BIN_APPLICATION` environment variable is prefixed and the open function is tried again. **INCLUDE** command is useful for including standard subroutines.

4.2.19 CALL — call Top subroutine

Options:

ROUTINE = ⟨`string` : 0⟩	`''`	subroutine name

Description: This command calls a Top subroutine ROUTINE.

4.2.20 SUBROUTINE — define Top subroutine

Options:

ROUTINE = ⟨`string` : 0⟩	`''`	subroutine name

Description: This command is the first Top statement for any routine. It has to have a matching **END_SUBROUTINE**. No nesting of subroutine definitions is allowed, although the definitions can be located anywhere in a file.

4.2.21 RETURN — return from Top subroutine

Description: This command will exit the execution from the current routine. It is optional.

4.2.22 END_SUBROUTINE — end definition of Top subroutine

Description: This command has to be present at the end of each routine. Possibly used instead of **RETURN** if **RETURN** not present.

4.2.23 DO — DO loop

Command: DO VAR = `START`, `END`, `STEP`
commands
END_DO

Description: Commas after `START` and `END` can be omitted. This loop is exactly like a Fortran `DO` loop except that real values are allowed for any of the four controlling variables. **VAR** must be a variable, while `START`, `END` and `STEP` can also be constants.

4.2.24 IF — conditional statement for numbers

Options:

OPERATION = ⟨`string` : 1⟩	`'SUM'`	`EQ | GT | LT | GE | LE | NE`

Description: This command performs conditional IF operation on two real arguments. The possible operations are equal (`EQ`), greater than (`GT`), less than (`LT`), greater or equal (`GE`), less or equal (`LE`), and not equal (`NE`). If the condition is true, the command specified in the THEN variable is executed. Otherwise the command in the ELSE variable is executed. Typically, these commands are **GO_TO** statements.

4.2.25 STRING_IF — conditional statement for strings

Options:

OPERATION = ⟨`string` : 1⟩	`'SUM'`	`EQ | NE | INDEX`
STRING_ARGUMENTS = ⟨`string` : 0⟩	`''`	arguments for string operation
THEN = ⟨`string` : 1⟩	`'undefined'`	statement when IF evaluates to T
ELSE = ⟨`string` : 1⟩	`'undefined'`	statement when IF evaluates to F

Description: This command performs conditional IF operation on two string arguments. The possible operations are equal (`EQ`), not equal (`NE`), and the Fortran `index()` function (`INDEX`), which returns true if there is 'argument2' substring within 'argument1'. If the condition is true, the command specified in the THEN variable is executed. Otherwise the command in the ELSE variable is executed. Typically, these commands are **GO_TO** statements.

4.2.26 STOP — exit Top

Description: This command stops the execution of the Top program.

4.3 Predefined Top variables

Name	Type
ARGUMENTS	⟨real : 0⟩
IO_UNIT	⟨integer : 1⟩
ID1	⟨integer : 1⟩
ID2	⟨integer : 1⟩
NUMBER_PLACES	⟨integer : 2⟩
FILE_EXISTS	⟨integer : 1⟩
OUTPUT_CONTROL	⟨integer : 4⟩
STOP_ON_ERROR	⟨integer : 1⟩
ERROR_STATUS	⟨integer : 1⟩
OBJECTS	⟨string : 0⟩
VARIABLES	⟨string : 0⟩
ROUTINE	⟨string : 1⟩
ROOT_NAME	⟨string : 1⟩
DIRECTORY	⟨string : 1⟩
FILE_ID	⟨string : 1⟩
OPERATION	⟨string : 1⟩
RESULT	⟨string : 1⟩
STRING_ARGUMENTS	⟨string : 0⟩
OBJECTS_FILE	⟨string : 1⟩
INCLUDE_FILE	⟨string : 1⟩
FILE	⟨string : 1⟩
RECORD	⟨string : 1⟩
THEN	⟨string : 1⟩
ELSE	⟨string : 1⟩
COMMAND	⟨string : 1⟩
FILE_EXT	⟨string : 1⟩
OUTPUT_DIRECTORY	⟨string : 1⟩
FILE_ACCESS	⟨string : 1⟩
FILE_STATUS	⟨string : 1⟩

Table 4.2: *Predefined* Top *variables*

Chapter 5

top.ini file

```
--- COMMANDS:
 1  no_action
 2  SET
 3  STOP
 4  LABEL
 5  GO_TO
 6  DEFINE_INTEGER
 7  DEFINE_REAL
 8  END_DO
 9  DO
10  CALL
11  RESET
12  WRITE
13  OPERATE
14  STRING_OPERATE
15  DEFINE_STRING
16  DEFINE_LOGICAL
17  SUBROUTINE
18  END_SUBROUTINE
19  INCLUDE
20  RETURN
21  READ
22  OPEN
23  CLOSE
24  IF
25  WRITE_TOP
26  SYSTEM
27  INQUIRE
28  STRING_IF
31  HIST2D
32  PLOT2D
33  READ_TABLE
34  RESET_CAPTIONS
35  WORLD
36  READ_DPLOT
```

```
37  SET_BLINES
38  LINE2D
39  PRINT
40  NEW_PAGE
41  ARROW
42  POSTSCRIPT
43  TRANSFORM
44  AXES2D
45  CAPTION
46  DPLOT
47  GET_BARS
48  READ_PDB
49  BALL_STICK
50  LABEL_ATOMS
51  MAKE_BONDS
52  DEFAULT_ATOM_COLOR
53  FIT
54  CENTER_MOL
55  SHUFFLE_DPLOT
56  ROTATE_MOL
57  ROTATE_MOL_AXIS
58  TRANSLATE_MOL
59  WRITE_TABLE
60  RESET_LEGEND
61  LEGEND
62  PLOT_ERROR_BARS
63  SELECT_DATA
64  WRITE_PDB
65  SPECTRUM
66  SWITCH_PS
67  SMOOTH_TABLE
68  TRANSLATE_TABLE
69  GET_DENSITY
70  WRITE_DPLOT
71  SELECT_ATOMS
72  FIT2
73  DENSITY_TO_XY
74  XY_TO_DENSITY
75  SET_RECORD
76  FILL_COLUMN
77  ZOOM
--- KEYWORDS:
 1  REAL    ARGUMENTS              0 0.00  # real arguments to the math operation
31  REAL    PAPER_WINDOW           5   7.0 15.8 17.0 23.0 0.0
32  REAL    WORLD_WINDOW           4   -999. -999. -999. -999.
33  REAL    X_TICK                 0   -999. -999. -999.
34  REAL    Y_TICK                 0   -999. -999. -999.
35  REAL    ARROW_SHAPE            3   0.002 0.012 0.036
36  REAL    BAR_GRAYNESS           0   0.7
```

```
37  REAL    BAR_WIDTH               1   1.0
38  REAL    DPLOT_GRAYNESS          2   1.0 0.0
39  REAL    DPLOT_BOUNDS            2   -999. -999.
40  REAL    TRF_PARAMETERS          0   0.0 1.0
41  REAL    PRINT_XY                2   1.0 1.0
42  REAL    PRINT_ANGLE             1   0.0
43  REAL    LINE2D_XY1              2   0.0 0.0
44  REAL    LINE2D_XY2              2   0.0 0.0
45  REAL    LINE2D_GRAYNESS         1   0.0
46  REAL    LINE2D_WIDTH            1   1.0
47  REAL    RADIUS_FACTOR           1   0.150
48  REAL    BOND_WIDTH_FACTOR       1   0.350
49  REAL    EYE_TO_SCREEN           1   30.0
50  REAL    SCREEN_TO_TOP           1   0.0
51  REAL    RADIUS                  1   1.5
52  REAL    COVALENT_BOND           2   -999 -999
53  REAL    ROTATION_MATRIX         9   1.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 1.0
54  REAL    AXIS                    3   0.0 1.0 0.0
55  REAL    DEGREES                 1   6.0
56  REAL    SHIFT                   3   30.0 0.0 0.0
57  REAL    CAPTION_XLEFT           1   -0.15
58  REAL    BAR_DX                  1  1.0
59  REAL    BAR_XSHIFT              1  0.0
60  REAL    SPHERE_RADIUS           1  10.0
61  REAL    SELECT_RANGE            2  1 1
62  REAL    TRANSLATION             1  0
63  REAL    BAR_DY                  1  1
64  REAL    DPLOT_FILL              1  0
65  REAL    CAPTION_XRIGHT          1  1.05
66  REAL    ARROW_POSITION          4  0 0 1 1
67  REAL    X_LABEL_SHIFT           1  0
68  REAL    Y_LABEL_SHIFT           1  0
69  REAL    SLAB                    5  9999 9999 0 0 0
70  REAL    BOND_TAPER              1  0.3
71  REAL    X_AXIS_FACTOR           1  1
72  REAL    Y_AXIS_FACTOR           1  1
73  REAL    BOND_COLOR              1  1.0
74  REAL    ATOM_COLOR              1  1.0
75  REAL    FIT_PARAM_INITIAL       0  1 1 1 1 1 1 1 1 1 1
76  REAL    FIT_CUTOFFS             2  0.0001 0.0001
77  REAL    TRF_UNDEFINED           1  -999999
78  REAL    PRINT_DXY               2  0 0
79  REAL    STAMP_SIZE              1   8
80  REAL    WORLD_FRACTION          1  -999
81  REAL    COLOR_RANGE             2  -999 -999
82  REAL    DPLOT_COLOR             6  0 0 0 0 0 1
83  REAL    SCALE_WORLDX            1  -999
84  REAL    COLUMN_PARAMETERS       2  1 1
 1  INTEGER IO_UNIT                 1  21 # IO unit for file operations
```

```
 2  INTEGER ID1                   1  1  # ID1 for filename construction
 3  INTEGER ID2                   1  1  # ID2 for filename construction
 4  INTEGER NUMBER_PLACES         2  5 2 # pre- and post-decimal point places
 5  INTEGER FILE_EXISTS           1 0    # an output flag: 0 | 1
 6  INTEGER OUTPUT_CONTROL        4 1 1 0 1 # selects output, flow-control msgs, warnings, errors
 7  INTEGER STOP_ON_ERROR         1 1 # whether to stop on error
31  INTEGER X_LABEL_STYLE         1   2
32  INTEGER Y_LABEL_STYLE         1   2
33  INTEGER X_TICK_DECIMALS       1   -999
34  INTEGER Y_TICK_DECIMALS       1   -999
35  INTEGER PLOT2D_SYMBOL_TYPE    1   0
36  INTEGER PLOT2D_LINE_TYPE      1   1
37  INTEGER XY_COLUMNS            2   1  2
38  INTEGER BAR_LINE_TYPE         1   2
39  INTEGER DPLOT_LINE_TYPE       1   3
40  INTEGER ERROR_COLUMN          1   0
41  INTEGER X_TICK_LABEL          2   -999 -999
42  INTEGER Y_TICK_LABEL          2   -999 -999
43  INTEGER NO_SPLINE_POINTS      1   400
44  INTEGER POSITION              2   0 1
45  INTEGER NO_COPIES             1   1
46  INTEGER NO_XY_SCOLUMNS        2   0  0
47  INTEGER XY_SCOLUMNS           0
48  INTEGER TICK_FONT             1   3
49  INTEGER POINT_FONT            1   6
50  INTEGER CAPTION_FONT          1   3
51  INTEGER CAPTION_POSITION      1   1
52  INTEGER PRINT_FONT            1   4
53  INTEGER PRINT_HORIZ           1   2
54  INTEGER PRINT_VERT            1   2
55  INTEGER BAR_LEGEND_PLACES     2   7 1
56  INTEGER LABEL_LOCATION        1   2
57  INTEGER LABEL_FONT            1   6
58  INTEGER LABEL_COLUMN          1   0
59  INTEGER POINT_MODULUS         1   1
60  INTEGER SELECT_COLUMN         1   1
61  INTEGER PLOT_POINTS           1   0
62  INTEGER HALF_WINDOW           1   3
63  INTEGER DESCRIPTION_FONT      1   5
64  INTEGER DPLOT_COLUMN          1   1
65  INTEGER SELECTION_STEP        1   1
66  INTEGER BOND_LINE             1   2
67  INTEGER ATOM_LINE             1   2
68  INTEGER FIT_PARAM_INDICES     0   1 2
69  INTEGER PRINT_POINT           1   1
70  INTEGER LINE2D_LINE_TYPE      1   0
71  INTEGER ROW_RANGE             2   0 0
72  INTEGER AXES2D_LINE_TYPE      1   2
73  INTEGER TICK_LINE_TYPE        1   2
```

```
74  INTEGER TICK_SMALL_LINE_TYPE    1   3
75  INTEGER LEGEND_POSITION         1   7
76  INTEGER COLOR_COLUMN            1   0
77  INTEGER COLUMN                  1   0
 1  STRING  OBJECTS                 0 '' # variable names or constants
 2  STRING  VARIABLES               0 '' # variable names
 3  STRING  ROUTINE                 0 '' # subroutine name
 4  STRING  ROOT_NAME               1 'undf' # root of a filename for filename construction
 5  STRING  DIRECTORY               1 '' # directory list
 6  STRING  FILE_ID                 1 'default' # file id for filename construction
 7  STRING  OPERATION               1 'SUM' # operation to perform: \V{SUM} | \V{MULTIPLY} | \V{DIVIDE} | \V{POWER} | \V{MOD}
 8  STRING  RESULT                  0 '' # variable name for the result of operation
 9  STRING  STRING_ARGUMENTS        0 '' # arguments for string operation
10  STRING  OBJECTS_FILE            1 'top.out' # filename
11  STRING  INCLUDE_FILE            1 '__asgl' # include file name
12  STRING  FILE                    1 'counter' # partial or complete filename
13  STRING  RECORD                  1 'undefined' # contents of the input line
14  STRING  THEN                    1 'undefined' # statement when IF evaluates to T
15  STRING  ELSE                    1 'undefined' # statement when IF evaluates to F
16  STRING  COMMAND                 1 'nothing' # UNIX command
17  STRING  FILE_EXT                1 '' # file extension for filename construction
18  STRING  OUTPUT_DIRECTORY        1 '' # output directory
19  STRING  FILE_ACCESS             1 'SEQUENTIAL'  # file access: \V{SEQUENTIAL} | \V{APPEND}
20  STRING  FILE_STATUS             1 'UNKNOWN'     # file status: \V{UNKNOWN} | \V{OLD} | \V{NEW}
31  STRING  SMOOTH_TYPE             1 'SPLINE'      # SPLINE | AVERAGE
32  STRING  DPLOT_PART              1 'FULL'        # LOWER | FULL
33  STRING  DPLOT_STYLE             1 'GRAY'        # GRAY | BLACK | COLOR
34  STRING  X_LABELS                0 ''
35  STRING  SELECTION_MODE          1 'ATOM'        # RESIDUE | ATOM
36  STRING  TRF_TYPE                1 'LOGARITHMIC1'
37  STRING  Y_LABELS                0 ''
38  STRING  FOR                     0 'DISPLAYING BONDING LABELLING'
39  STRING  SPHERE_CENTER           2 'undefined' undefined
40  STRING  CAPTION_TEXT            1 'undefined'
41  STRING  PRINT_TEXT              1 'undefined'
42  STRING  LABEL_STYLE             0 'RESIDUE_TYPE1 RESIDUE_NUMBER'
43  STRING  SELECTION_SEGMENT       2 'X' 'X'
44  STRING  RES_TYPE                1 'ALL'
45  STRING  ATOM_TYPE               1 'ALL'
46  STRING  BOND_TYPE               1 'COVALENT'
47  STRING  SYMBOL                  1 'LINE'      # LINE | POINT | BAR
48  STRING  DESCRIPTION             1 'description'
49  STRING  POSTSCRIPT_TEXT         1 'description'
50  STRING  Y_SCALE                 1 'LEFT'
51  STRING  SELECTION_SEARCH        1 'SEGMENT'   # SEGMENT | SPHERE
52  STRING  FROM                    1 'ALL' # ALL DISPLAYING BONDING LABELLING
53  STRING  SELECTION_STATUS        1 'INITIALIZE' # REMOVE ADD
54  STRING  FIT_MODEL               1 'POLYNOMIAL' # EXPONENTIAL NORMAL LOGNORMAL LOGARITHMIC POLYGAUSS
55  STRING  PRINT_MODE              1 'XY'   # XY | POINT
```

```
56  STRING  PDB_FORMAT              1 'PDB'  # PDB | XYZ
57  STRING  SHUFFLE_OPERATION       1 'TRANSPOSE' # REVERT_X, REVERT_Y
58  STRING  DPLOT_ORIENTATION       1 'XY' # YX
59  STRING  STAMP_TEXT              1 'DEFAULT'  # NONE | text of your choice
60  STRING  LINE2D_WIDTH_SCALING    1 'Y'  # 'X' | 'Y'
61  STRING  COORDINATES             1 'WORLD'  # 'WORLD' | 'PLOT'
62  STRING  ERROR_SIGN              1 'UP DOWN'  # 'UP' | 'UP DOWN' | 'DOWN'
63  STRING  COLOR_STYLE             1 'GRAY'        # (GRAY BLACK)
31  LOGICAL PERSPECTIVE             1 OFF  #
32  LOGICAL ADD_DATA                2 OFF OFF
33  LOGICAL BAR_LEGEND              1 ON   #
34  LOGICAL NUMBER_DENSITY_PLOT     1 OFF
35  LOGICAL DPLOT_FORMAT            1 OFF
36  LOGICAL DPLOT_SYMMETRIZE        1 OFF
37  LOGICAL A4_WINDOW_MARGIN        1 OFF
38  LOGICAL ADD_BONDS               1 ON
39  LOGICAL EXPONENT                1 OFF
40  LOGICAL FIT_WORLD               1 OFF
41  LOGICAL CLIP                    1 ON
42  LOGICAL STANDARD_ERROR          1 OFF
--- END OF FILE
```

Index