Chapter 9

XDGRAPH - Visualising the Results

9.1 Overview

The graphics program differs from the rest of the package in one major way. To account for its interactive nature, it is not driven by the master file, but instead is controlled by a command language. The Tool Command Language (Tcl, by J.K.Ousterhout) was chosen because it provides a general scripting language in which special application-defined commands are easily integrated. The Tcl based toolkit (Tk) for the X11 Window System was then used to add a graphical user interface on top of the existing commands.

9.2 The Command Line Interface

XDGRAPH roughly follows the concept of Tk with its commands. For each type of high-level "object" one can work with (examples of objects are datasets, contour levels, etc.), a command exists to create this object (*e.g.* **dataset**, **contour**). This, in turn, creates a new command with the same name as the object. Actions on the object (apart from creation) are performed using that "object command". The different actions available for an object are called *subcommands*.

The name of an object must follow certain rules:

- All names start with a ":" (colon).
- Objects that are derived from other objects (*i.e.* from datasets) start with the name of that object, separated by a colon.

For example a valid name for a dataset would be :set1, and a possible name for a contour level group derived from that dataset would be :set1:plus. In addition to the "object oriented" commands, more "action oriented" commands are available. They are mostly implemented as Tcl-procedures.

XDGRAPH distinguishes between the creation of an object and the actual graphics output. The latter is considered an action you perform on this object by using the subcommand **plot** of the object.

So the general scheme to create a plot looks like this:

Load the data you want to visualize from a file

```
dataset :set001 -load xd_defden.grd
```

Create some graphical objects

```
contour :set001:plus -val {0.1 0.2 0.3 0.4 0.5}
contour :set001:zero  -val 0. -style dash
molecule :set001:mol
```

Display those objects

```
:set001:plus plot
:set001:zero plot
:set001:mol plot
```

9.3 The Graphical User Interface

The graphical user interface to XDGRAPH reflects the command structure described in the previous section. A dialog box exists for each object which allows to enter the options the respective command takes. Online help, access to reasonable default values, menu lists, file and colour browsers support the user entering required data.

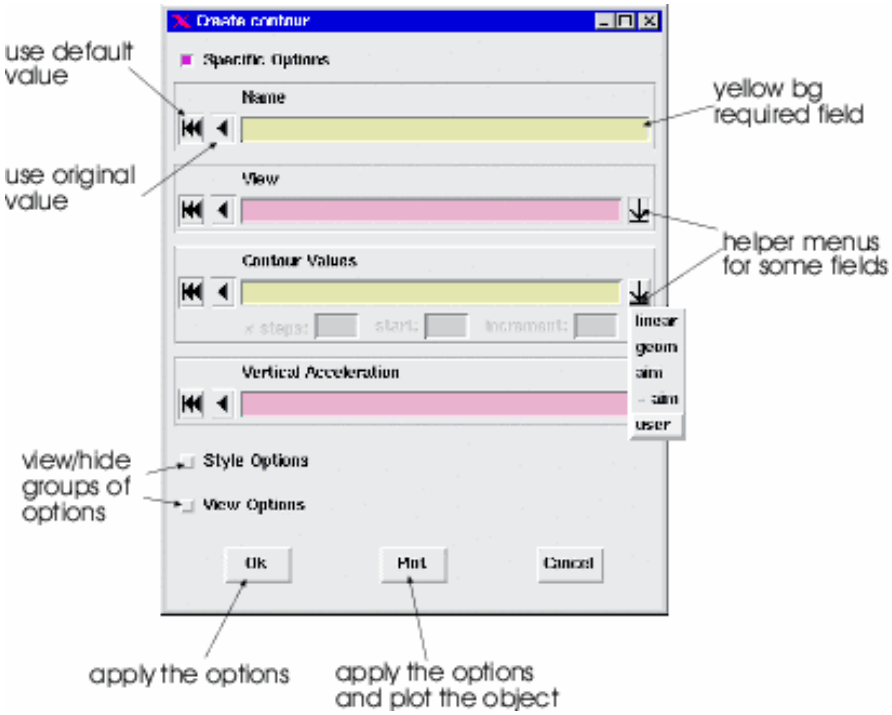

Figure 9-1 : A sample dialog box from the graphical user interface

is the example from the previous section, this time using the GUI:
(Menu entries are written as '**Menu::Submenu**')

Load the data you want to visualize from a file

Choose `File::Load Dataset` and fill in the necessary fields. You can get support to enter the file name from a directory browser. Which file selector is actually used depends on the Tcl/Tk release. Starting with 8.0, the internal file selector which comes with Tk is used. For older versions XD's own selector is used. You can force its use by setting the environment variable XD_USE_PRIVATE_FSBOX. A list of allowed file types is available as a menu. Use 'Ok' to finish this step.

Create and plot a simple graphical object

Choose `Create::Contour`. Again, a dialog box appears. See Figure 9-1 for details. Use 'Plot' to finish this step.

Create and plot the second group of contour lines

Same procedure as above. Open the 'Style Options' part of the dialog box and use the 'Linestyle' menu to specify dashed lines.

Finally, add a representation of the molecule to the graphics

Use `Create::Molecule`. Again, use 'Plot' to finish this step.

9.4 Running XDGRAPH

SYNOPSIS
xdgraph *(options) (tcl-script)*

OPTIONS
-d *driver*
select a driver (**tk** or **gt**)

9.5 Commands

Some of the options and subcommands are marked with a star (*). They are common to more than one command. Their descriptions can be found in Sections 9.6 & 9.7.

9.5.1 dataset

Create a dataset or list existing datasets.

SYNOPSIS
dataset *?dataSetPattern?*
dataset *name* **-load** *fileName options*
dataset *name* **-slice** *sourceDataSet options*

DESCRIPTION

This command has different uses, depending on the number of arguments. The first form of the command, with at most one argument, returns a list of existing datasets. If no pattern is specified all datasets are listed, otherwise only those matching the given pattern are returned. A list of special character sequences for pattern matching can be found in Section 9.11.3

The second form of the **dataset** command, which requires at least two arguments, creates a dataset. This can either be done by reading a file (requires **-load** as the first option) or by interpolating another dataset (if **-slice** is given as the first option). In either case, the first argument is the name of the new dataset.

The following set of options is available for the file reading version of the **dataset** command:

OPTIONS

-load *fileName*
The name of the file to be read. The GUI provides a file selector box for Unix users. *Note for VMS users:* Please read the section about file name syntax in Tcl in Section 9.11.4. Remember that square brackets [ ] and dollar signs have special meanings in Tcl.

-type *fileType*
The format (and to some extent the content type) of the file to be read. Valid file types are:
aim
A grid file format as used by some versions of the AIMPAC package.
xddata
This is used for xy-diagrams. It contains a list of arbitrary data points, with possibly multiple values per point. *Still experimental!*
xdgrid

Grid files contain data on a rectangular grid, together with a list of objects (atoms and critical points). Either 2- or 3-dimensional grids are possible. XDPROP and XDFOUR write grid files using this format.

xdpath

This type of file is written by XDPROP if a bond path calculation with algorithm 2 was done.

The following set of options is only available for the slicing version of the **dataset** command. This option is not yet available in the GUI.

OPTIONS

-slice *sourceDataSet*
-3point* *p1 p2 p3*
-nx *numXPoints*
-ny *numXPoints*
-nz *numXPoints*
-xmin *minXCoord*
-xmax *maxXCoord*
-ymin *minYCoord*
-ymax *maxYCoord*
-zmin *minZCoord*
-zmax *maxZCoord*

SUBCOMMANDS

configure* *options*
connections –auto
Generate a list of connections between atoms based on their distance.
contour
Returns a list of all contour level groups created from this dataset.
heightfield
Returns a list of all height field objects created from this dataset.
info*
Output some information about the dataset.
interpolate
Only available for slices. Re-calculate the data values by interpolating in the original dataset given when the slice was created. This subcommand must be used after **rotate** and **translate** subcommands.
isosurface
Returns a list of iso-surface objects created from this dataset.
molecule
Returns a list of molecule objects created from this dataset.
path
Returns a list of bond path objects created from this dataset.
property
Returns the property this dataset maps.
relief
Returns a list of relief objects created from this dataset.
remove*
Remove this dataset.
rotate *options*
Only available for slices. Rotate the slice. Note, that this command does not re-calculate the data values. You have to call the **interpolate** subcommand explicitly. Different options are available:
-eulerian *angle1 angle2 angle3*
-x *angle*
-y *angle*
-z *angle*

translate *{x y z}*
Only available for slices. Translate the slice. Note that this command does not recalculate the data values. You have to call the **interpolate** subcommand explicitly.

xydiagram
Returns a list of xy-diagram objects created from this dataset.

colorbg *min max*
Draw a filled rectangle on each grid point. The colour is chosen by mapping the given range (defaults to the whole range of values in the grid) to 256 colours in a colour map (Currently the colour map can't be changed by the user. A fixed map $xd_datadir/default.cmap is used.Values outside the given range are not filled.

EXAMPLES
> dataset :ox -load xd_drho.grd
> dataset
< :ox

9.5.2 contour

This is used to visualize data on a 2-dimensional grid by drawing smooth lines connecting points of equal value. XDGRAPH handles groups of lines for different data values together as single objects. To get a complete contour map you usually create a few groups with different style options to distinguish different data ranges (for example positive and negative values.)

SYNOPSIS
contour *name ?options?*

OPTIONS

Specific Options
-plane* *plane*
This is used to select a plane from a 3-dimensional grid.

-values *valueList*
A list of values, for which contours are to be drawn. When used with a **configure** subcommand, the list overwrites the previous one, while when used with **append** the new values are appended to the existing list (appended, not merged!). Note that this is a 'list' in the Tcl sense:
contour :d1:clg -values 0. 1. 2. *is not correct !!*
contour :d1:clg -values {0. 1. 2.} this is the correct usage using curly braces {}

-vertac *vertAcc*
Vertical accentuation - this scales the data values to z coordinates. It is useful to prevent clipping and when adding contour lines to a height field. Currently only used when the OpenGL driver is in effect.

-view *view*
The view this contour group should use. See Section 9.5.9 for details about the default behaviour, which should be reasonable for most simple applications.

Style Options

-foreground | **-fg*** *colour* | *{colour1 colour2}*
This option takes either a single colour (see Section 9.6 on different ways to specify a colour) or a list consisting of two colour values. When two colours are given, the first one is assigned to the first contour level and the second one to the last contour level in the group. Intermediate values are interpolated. Note that the way the colours are specified influences intermediate colours. (See Section on common options).

-style* *lineStyle*
The line style (solid, dotted, etc.) used for this contour group.

View Options
See Section 9.6 for a list of possible options.

SUBCOMMANDS

configure* *options*
Change options for an existing contour line group. See previous section for a list of possible options.
append *options*
This is like **configure**, except that any **-values** given with **append** are added to the contour group, while those given with **configure** overwrite the previous list.
clear*
Remove the contour group from its view.
info*
Output some information about the contour group.
plot*
Display this contour group, adding it to its view.
remove
Delete the contour group.

9.5.3 height field

This is used to visualize data on a 2-dimensional grid by drawing an open surface on the grid which is elevated according to the data values. Creates a smooth surface from data on a rectangular grid where the height corresponds to the data values. **NOTE** *This option is NOT available in Windows versions of XDGRAPH.*
.

SYNOPSIS
heightfield *fieldheightField ?options?*

OPTIONS

Specific Options
-cutoff *cutOff* | {*lowCut highCut*}
Limits the maximum and minimum elevation. If only one value is given, it specifies the upper cutoff limit. The lower cutoff value is - cutoff in this case.
-map *name*
For future use.
-plane *n*
This is used to select a plane from a 3-dimensional grid.
-vertac *vertAcc*
This scales the data values to z coordinates. Reasonable values depend on the mapped property.
-view *view*
The view this contour group should use. See Section 9.5.9 for details about the default behaviour, which should be reasonable for most simple applications.

Style Options
-foreground | **-fg*** *colour* | {*colour1 colour2*}
The colour of the surface, changed by lighting calculations. Currently only one colour is used.
-polygon* *polygonMode*

View Options
See section 9.6 for a list of possible options.

SUBCOMMANDS

configure* *options*
Change options for an existing height field object. See previous section for a list of possible options.

clear*
Remove the height field from its view.

info*
Output some information about the height field.

plot*
Display this height field, adding it to its view.

remove*
Delete the height field.

9.5.4 iso-surface

This is used to visualize data on a 3-dimensional grid by drawing smooth surfaces connecting points of equal value. Surfaces are represented by triangles which can be rendered using solid planes, lines or points. The later two options make it possible to see surfaces inside of one another. **NOTE** *This option is NOT available in Windows versions of XDGRAPH.*

SYNOPSIS

isosurface *name ?options?*

OPTIONS

Specific Options

-values *valueList*
A list of values for which iso-surfaces are to be drawn. When used with a **configure** subcommand, the list overwrites the previous one, while when used with **append** the new values are appended to the existing list (appended, not merged!).

-view *view*
The view these iso-surfaces should use. See Section 9.5.9 for details about the default behaviour, which should be reasonable for most simple applications.

Style Options

-foreground | -fg* *colour*
The colour used two draw this iso-surface.

-style* *lineStyle*
The line style (solid, dotted, etc.) used in case the polygonMode is set too **line.**

-polygon* *polygonMode*
How to render this iso-surface - **fill** (solid), **line** (lines) or **point** (points)

View Options
See Section 9.6 or a list of possible options.

SUBCOMMANDS

configure* *options*
Change options for an existing iso-surface. See previous section for a list of possible options.

append *options*
This is like **configure,** except that any **-values** given with **append** arc added to the iso-surfaces, while those given with **configure** overwrite the previous list.

clear*
Remove the iso-surface from its view.

info*
Output some information about the iso-surface.
plot*
Display this iso-surface, adding it to its view.
remove*
Delete the iso-surface.

9.5.5 molecule

The molecule as read from a grid or bond path file is visualized according to the view type used. For a contour or bond path view a line drawing is used, while for an iso-surface view a ball-and-stick model is used.

SYNOPSIS
molecule *name ?options?*

OPTIONS

Specific Options
-atoms *drawAtoms*
Include or exclude atoms from the display. *drawAtoms* is a boolean (**on**, **off**, **yes**, **no**).
-bonds *drawBonds*
Include or exclude bonds from the display. *drawBonds* is a boolean
-label *drawLabels*
Include or exclude labels from the display. *drawLabels* is a boolean
-view *view*
The view this molecule should use.
-zlimit *zlimit*
Exclude atoms further away from the plane than *zlimit* Angstrom.

Style Options
Not yet implemented.

View Options
See section 9.6 for a list of possible options.

SUBCOMMANDS

configure* *options*
Change options for an existing molecule. See previous section for a list of possible options.
clear*
Remove the molecule from its view.
info*
Output some information about the molecule.
plot*
Display this molecule, adding it to its view.
remove*
Delete the molecule.

9.5.6 path

This is used to visualize data from a bond path calculation.

SYNOPSIS
path *name ?options?*

OPTIONS

Specific Options
-view *view*
The view this bond path plot should use. See section 9.6 for a list of possible *view* options.

Style Options
-foreground | -**fg*** *colour*
The colour used for this bond path plot.
-style* *lineStyle*
The line style (solid, dotted, etc.) used for this bond path plot.

SUBCOMMANDS

configure* *options*
Change options for an existing bond path plot. See previous section for a list of possible options.
clear*
Remove the bond path plot from its view.
info*
Output some information about the bond path plot.
plot*
Display this bond path plot, adding it to its view.
remove*
Delete the bond path plot.

9.5.7 relief

Create a relief plot, *i.e.* the data is visualized as view of a landscape, using the value on each grid point as its height. The transformation into the display plane is chosen by giving a viewpoint. Currently, a number of restrictions apply:
- you can't select a plane from a 3-dimensional grid and
- you can't choose in which direction lines are drawn (currently always along *x* and *y*).

SYNOPSIS
relief *name ?options?*

OPTIONS

Specific Options
-cutoff *cutoffValue | {highCutoff lowCutoff}*
When used with one value, this option limits the absolute value of any data point to *cutoffValue.* When a list with two values is given, the high and low cutoff values can be given separately.
-eye *{x y z}*
The eye-point is a point in 3d-space from where the relief is viewed. The viewer is always looking across the map to the corner 0., 0., 0. This is a parallel projection, so only the ratio of the three numbers is used. The default value is (1. 1. 0.6).

-size *{hSteps vSteps}*
The number of lines to draw parallel to x and y, respectively. Defaults to the number of grid points.
-vertac *scaleFactor*
Gives a scale factor from data values to y-plot coordinates. This defaults to fitting the data to the plot size. Note this option is probably required if any of the *cutoff* options were used.

-view *view*
The view this contour group should use. See Section 9.5.9 for details about the default behaviour, which should be reasonable for most simple applications.

Style Options
-foreground | -fg* *colour*
The colour used for this relief plot.
-style* *lineStyle*
The line style (solid, dotted, etc.) used for this relief plot.

View Options
See section 9.6 for a list of possible options.

SUBCOMMANDS
configure* *options*
Change options for an existing relief plot. See previous section for a list of possible options.
clear*
Remove the relief plot from its view.
info*
Output some information about the relief plot.
plot*
Display this relief plot, adding it to its view.
remove*
Delete the relief plot.

9.5.8 xydiagram

Create an xy-diagram, *experimental.*

SYNOPSIS
xydiagram *name ?options?*

OPTIONS

Specific Options
-type point | line
-view
-x *i | varName*
-y *i | varName*

View Options
See Section 9.6 for a list of possible options.

SUBCOMMANDS
configure* *options*
clear*
info*
plot*
remove*

9.5.9 view

A view is used to create a connection between graphical objects (like contour lines or molecules) and the screen. It is an abstract object which manifests itself as an X11 window.

name: The name of the view object to create. Please note, that view names do not follow the rules for other objects in XDGRAPH. They do not have to start with a colon and they are not derived from any object.

type: The type of the view corresponds to the kind of objects shown. There is no separate view for molecules. Molecules may use any of the other view types (except **xydiag** and **relief**). The way molecules are represented depends on the type of view. E.g. in a **contour** view it is a simple line drawing whereas in a **surface** view a 3D ball and stick model is used.

Whenever you create a graphical object using a non-existing view (explicitly or implicitly) a view of the appropriate type is created automatically. Its name is derived from the dataset the object belongs to and the type of the object. For molecules a contour view is used. If you want to draw a molecule in another type of view, you must specify the name explicitly using the **-view** option. You can either create the view by hand or create the other object first and use the automatic name.

The following view types are available:
contour
height
path
relief
surface
xydiag

SYNOPSIS
view *name* **-type** *type ?options?*

OPTIONS

View Options
See Section 9.6 for a list of possible options.

SUBCOMMANDS
configure* *options*
Change options for an existing view. See previous section for a list of possible options.
matrix
Print, the 4 by 4 transformation matrix. This can be used to restore an orientation obtained using the mouse. This is currently implemented as **ddr**
objects
Returns a list of objects connected to this view.
rotate *options*
Rotate the view. **Not yet implemented!!!**
-x *angle*
-y *angle*
-z *angle*
translate {**x y z**}
Translate the view. **Not yet implemented!!!**

9.6 Common Options

View Options

These options are available for all graphics objects as well as for views. When used with graphics objects, they are applied to the related view, however. You cannot transform one object separate from another in the same view using these options.

Size and Scaling

-width *width*

-height *height*
-scale *scaleFactor* | **auto**

The first two options set the size of the view (in cm). If negative or not specified, this is calculated from *vrange* and *scaleFactor*. If no scale factor is given either, a default of 18cm is used. The scale factor is used to transform Angstrom to cm, a value of one meaning that 1cm in the plot corresponds to 1Å in the data. If the scale factor is less than or equal to zero or specified as **auto,** the data is scaled to fit into the view. This requires **-vrange** to be specified.

If all three options are given, the view might not be fully used or clipping might occur.

-vrange *{xrange yrange}*
For a grid file, this gives the length of the x and y axis of the grid (in Å).

Translation and Rotation
-matrix *matrix*
Transformation matrix, *matrix* is either a 3 by 3 or a 4 by 4 transformation matrix. This is mainly useful to restore a matrix from a previous run. (See section 9.5.9). **This option is not yet fully implemented !!!**
-origin *{x y z}*
Shift the objects before rotation.
-translate *{x y z}*
Shift the plot after rotation but before scaling. (So this is in Ångstrom.)

-3point *pi p2 p3*
An easy way to give origin and orientation. $\vec{p}_1$ is the origin, $\vec{p}_2 - \vec{p}_1$ gives the positive x axis $\vec{x}$. The *z* axis $\vec{z}$ is given by $\vec{x} \times (\vec{p}_3 - \vec{p}_1)$, and $\vec{z} \times \vec{x}$ is the y axis. Each of the points may be specified in one of the following ways:
{p_x p_y p_y}
A list, giving the coordinates directly.
label
The object label.

Style Options

-foreground or **-fg** *colour*
Sets the foreground colour(s) used to draw an object. The colour can be given in a number of ways.
name
XD's own database is used to convert colour names to RGB values when necessary, *(i.e.* not for Tk)
RGBtriple
X Window System style #rgb, #rrggbb #rrrgggbbb #rrrrggggbbbb with 'r', 'g', 'b' being hex digits
{**RGB** *red green blue}*
red, green, blue: [0., 1.|
{**HSV** *hue saturation value}*
hue: [0.,360.) 0. is red, saturation: [0., 1.], value: [0.,1.]
{**YUV** *luminance u v}*
luminance, u, v: [0.,1.], u, v: [-0.5,0.5]
-polygon *polygonMode*
How to draw polygons. One of the following:
solid
Draw solid, shaded faces.
line
Draw the shaded edges of the polygons.

point
Draw only the vertices of the polygons.

-style *lineStyle*
Sets the line style. *lineStyle* is either the keyword **solid** or a string build from the following elements.

" " (blank)
Empty space. You can use multiple blanks to add more space. For example the string "dot dot dot" would draw a line with three dots close to each other, separated by a larger space. If there are no trailing blanks given, a single one is added automatically.

dot
A dot.

dash
A dash.

long
A long dash.

Valid examples are: 'solid' (default), 'dot' or 'dot dash'.

Other Common Options

-plane *n*
This is used to select a plane from a 3-dimensional grid. For those grids, an xy-plane is plotted and the parameter *n* selects the n'th section along the *z-axis.* The first plane is numbered 1, which is also the default value for this option. *Rarely used.*

-view *view*
The view the object will use. See Section 9.5.9 for details about the default behaviour, which should be reasonable for most simple applications. This option is only meaningful while objects are created. Currently, the behaviour when used with the **configure** subcommand is undefined.

9.7 Common Subcommands

configure *options*
Change options for an existing object. The options that can be used are the same as for the command that is used to create the respective object type.

info
Show some information about the object.

plot
Plot the object.

clear
The object is removed from the display

remove
Delete the object. The object command is removed and the associated memory is released.

view
Return the view this object is using.

9.8 Toolbox

9.8.1 sleep

Sleep (do nothing) for *seconds* seconds.

SYNOPSIS
sleep *seconds*

9.8.2 plot

Plot all objects derived from the given dataset(s). Default are all datasets.

SYNOPSIS
plot *?datasets?*

9.8.3 clear

Clear all plotted objects derived from the given dataset(s). Default are all datasets.

SYNOPSIS
clear *?datasets?*

9.8.4 hardcopy

Dump the contents of the currently active display to file. The actual semantics of this command depend on the display driver in use.
When used with the Tk driver, the output will be a PostScript file. Width and height are standard Tk measures, they default to the size of the window. The default filename is `xdgraph.ps`.

Unfortunately, the *OpenGL* driver is only capable of outputing pixel oriented data. Currently, the file is written as a PPM file (Portable Pixel Map file). Conversion utilities to other pixel file formats are available from many ftp servers around the world. The default file name is `xdgraph.ppm`. Width and height are given in pixels. The default of 500 is only suitable for test purposes. The optimal value depends on your printer and the type of graphics shown. Start with values around 1500 for serious work. Use the Tk driver for line drawings such as contour maps. The PostScript output is much better suited for this purpose.

Linux users may find that the hardcopy option for the OpenGL driver is not functioning (an error message about no "visual for dump" is given). In this instance, hardcopy may be obtained with the Linux utility program import, by screen-grabbing the window. The image may be saved in several formats include PostScript and GIF, and may also be resized. See "man import" for further details of command syntax for import. The command line is given in any terminal window and the actual image is grabbed by then clicking on the window displaying the desired graphic with the mouse. For example, to save as a PostScript file

import image.ps *<CR> then click desired window with mouse*

or to save as a GIF with 150% expansion in image size

import image.gif -geometry 150% *<CR> then click desired window with mouse*

SYNOPSIS
hardcopy ?-file *filename? ?-***width** *width? ?-***height** *height?*

9.8.5 generate

Return a list of values. Very useful where values for contour levels have to be given, *type* maybe one of

lin
: Create a linear range of *nsteps* values, starting with *start,* adding *increment.*

geo
: Create a geometric range of *nsteps* values, starting with *start,* multiplying by *increment.*

aim

{ .001 .002 .004 .008 .02 .04 .08 .2 .4 .8 2. 4. 8. 20. 40. 80. 200. 400. 800.}

maim

As **aim,** but with negative sign.

SYNOPSIS
generate *type ?nsteps ?start ?increment???*

EXAMPLES
```
contour :d1:plus -values [generate lin 10 0.1 0.1]
```
is the same as
```
contour :d1:plus -values {0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0}
```

9.9 Display Driver

Currently, XDGRAPH is interfaced to two different libraries for the actual display/plotting. One is the X11 Window System toolkit Tk already mentioned. It is the preferred display driver on any platform with X11 available. It requires the installation of Tk.

The other driver uses the *OpenGL* (or the Linux-clone MESA) libraries. *This driver is not available under Windows, because it requires the (Unix specific) GLX library.*

The Tk driver has some special features:

- The atomic labels in contour and path plots can be moved using the mouse. Move the cursor over the text and wait until the colour changes. Than press the left mouse button and drag the label over the display. Release the mouse button when you are satisfied with the placement. This does *not* move the marker of the atom, just the label.

9.10 Examples

The following examples can be found in the source directory $TOP/xdgraph/examples.

9.10.1 bw.tcl

This creates a contour plot from the data in file `xd_defden.grd`. The dataset is called :1. For positive and negative contour levels the **generate** command is used. The **-style** option is used to draw the zero level and negative levels with a different line style. The command **plot** is used to plot the three contour level groups together with a title, atom labels and some further info.

```
dataset   :1  -load xd_defden.grd
contour :1:plus  -val [generate lin  10    0.1     0.1]
contour :1:zero  -val 0.     -style dash
contour :1:minus -val [generate lin 10 -0.1  -0.1]   -style dot
plot
```

9.10.2 colramp.tcl

This creates a contour plot with coloured contour lines. Because two different colours are given for positive and negative contour level groups each contour level has a different colour. The YUV colour model is used to specify the colours because it is especially useful for interpolation. Unfortunately, it is not easy for humans to relate a specific colour to an YUV triple.

```
dataset   :1  -load xd_defden.grd
contour   :1:plus    -val   [generate lin 15    0.1    0.1]   \
-foreground {{YUV 0.6 0.6 0.1} {YUV 0.1 0.7 0.44}}
contour   :1:zero    -val 0.   -style dash
contour   :1:minus -val   [generate lin 10-0.1-0.1]   \
-fore {{YUV 0.9 0.2 0.5} {YUV 0.2 0.4 0.88}}
plot
```

9.11 A few words about Tcl

9.11.1 Syntax

The following list is derived from the Tcl man page. It gives almost all rules that define the syntax of the Tcl language. The examples mainly make use of two of the features: Quoting strings with curly braces ({ }) and *command substitution* with square brackets ([ ], which executes the enclosed string and substitutes it with the result).

- A Tcl script is a string containing one or more commands. Semi-colons and newlines are command separators unless quoted as described below. Close brackets are command terminators during command substitution (see below) unless quoted.

- A command is evaluated in two steps. First, the Tcl interpreter breaks the command into *words* and performs substitutions as described below. These substitutions are performed in the same way for all commands. The first word is used to locate a command procedure to carry out the command, then all of the words of the command are passed to the command procedure. The command procedure is free to interpret each of its words in any way it likes, such as an integer, variable name, list, or Tcl script. Different commands interpret their words differently.

- Words of a command are separated by white space (except for newlines, which are command separators).

- If the first character of a word is double-quote (" " ") then the word is terminated by the next double-quote character. If semi-colons, close brackets, or white space characters (including newlines) appear between the quotes then they are treated as ordinary characters and included in the word. Command substitution, variable substitution, and backslash substitution are performed on the characters between the quotes as described below. The double-quotes are not retained as part of the word.

- If the first character of a word is an open brace ("{") then the word is terminated by the matching close brace ("}"). Braces nest within the word: for each additional open brace there must be an additional close brace (however, if an open brace or close brace within the word is quoted with a backslash then it is not counted in locating the matching close brace). No substitutions are performed on the characters between the braces except for backslash-newline substitutions described below, nor do semi-colons, newlines, close brackets, or white space receive any special interpretation. The word will consist of exactly the characters between the outer braces, not including the braces themselves.

- If a word contains an open bracket ("[") then Tcl performs *command substitution.* To do this it invokes the Tcl interpreter recursively to process the characters following the open bracket as a Tcl script. The script may contain any number of commands and must be terminated by a close bracket ("]"). The result of the script (i.e. the result of its last command) is substituted into the word in place of the brackets and all of the characters between them. There may be any number of command substitutions in a single word. Command substitution is not performed on words enclosed in braces.

- If a word contains a dollar-sign ("$") then Tcl performs *variable substitution:* the dollar-sign and the following characters are replaced in the word by the value of a variable.

- If a backslash ("\") appears within a word then *backslash substitution* occurs. In all cases but those described below the backslash is dropped and the following character is treated as an ordinary character and included in the word. This allows

characters such as double quotes, close brackets, and dollar signs to be included in words without triggering special processing. The following table lists the backslash sequences that are handled specially, along with the value that replaces each sequence.

\a Audible alert (bell) (0x7).
\b Backspace (0x8).
\f Form feed (0xc).
\n Newline (0xa).
\r Carriage-return (0xd).
\t Tab (0x9).
\v Vertical tab (0xb).
\<newline>*whiteSpace* A single space character replaces the backslash, newline, and all white space after the newline. This backslash sequence is unique in that it is replaced in a separate pre-pass before the command is actually parsed. This means that it will be replaced even when it occurs between braces, and the resulting space will be treated as a word separator if it is not in braces or quotes.
**** Backslash ("\").
ooo The digits *ooo* (one, two, or three of them)]give the octal value of the character.
x*hh* The hexadecimal digits *hh* give the hexadecimal value of the character. Any number of digits may be present.

Backslash substitution is not performed on words enclosed in braces, except for backslash-newline as described above.

- If a hash character ("#") appears at a point where Tcl is expecting the first character of the first word of a command, then the hash character and the characters that follow it, up through the next newline, are treated as a comment and ignored. The comment character only has significance when it appears at the beginning of a command.

- Each character is processed exactly once by the Tcl interpreter as part of creating the words of a command. For example, if variable substitution occurs then no further substitutions are performed on the value of the variable; the value is inserted into the word verbatim. If command substitution occurs then the nested command is processed entirely by the recursive call to the Tcl interpreter; no substitutions are performed before making the recursive call and no additional substitutions are performed on the result of the nested script.

- Substitutions do not affect the word boundaries of a command. For example, during variable substitution the entire value of the variable becomes part, of a single word, even if the variable's value contains spaces.

9.11.2 Some built-in commands

A very handy Tcl command is **source.** This is used to begin reading further commands from a file, switching back to stdin at the end of the file. For example, you could copy the file bw.tcl (see above) into your current directory and use it by typing "source bw.tcl" inside XDGRAPH.

9.11.3 Pattern Matching

* Matches any sequence of characters including a null string.
? Matches any single character.
[chars] Matches any character in the set given by *chars*. If a sequence of the form *x-y* appears in *chars,* then any character between *x* and *y,* inclusive, will match.
x Matches the single character *x.* This provides a way of avoiding the special interpretation of the characters *? [] \ in the pattern.

9.11.4 Notes for *Windows™* users

The *Windows™* release of XD contains an executable version of XDGRAPH which is linked with version 8.3 of the ***Tcl/tk*** library. The runtime libraries TCL83.DLL and TK83.DLL, as well as all the necessary Tcl/tk system files are supplied in the release, so there are no external dependencies. Unfortunately, the ***OpenGL*** driver is *not available* for this version of XDGRAPH, so several functions available in the Linux/Unix versions (like iso-surface plots) do not work in the *Windows™* version. An alternative program for viewing isosurfaces is *MoleCoolQT*, written by Christian B. Hübschle (Freie Universität Berlin), and freely available for academic users. See Section 13.3 for download details.