Technical Report
RAL-TR-97-060

PGXTAL – 3-D plotting with PGPLOT

D S Sivia

October 1997

COUNCIL FOR THE CENTRAL LABORATORY OF THE RESEARCH COUNCILS

Enquiries about copyright, reproduction and requests for additional copies of this report should be addressed to:

The Central Laboratory of the Research Councils
Library and Information Services
Rutherford Appleton Laboratory
Chilton
Didcot
Oxfordshire
OX11 0QX
Tel: 01235 445384 Fax: 01235 446403
E-mail library@rl.ac.uk

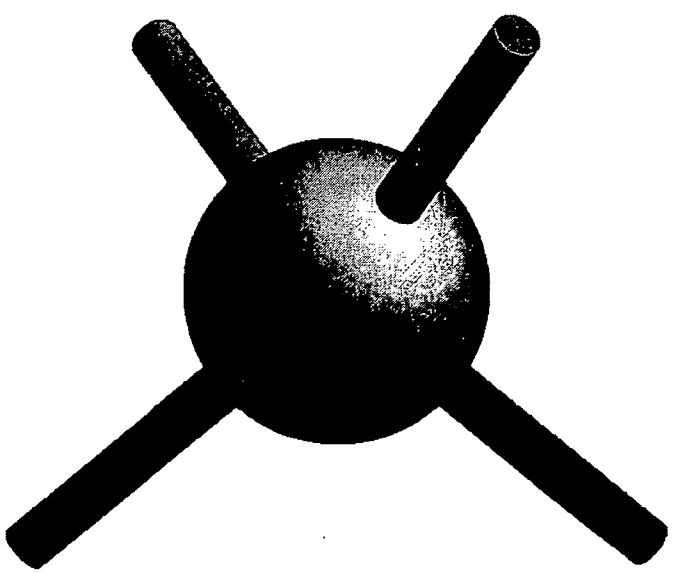
ISSN 1358-6254

PGXTAL

3-D plotting with PGPLOT

D. S. Sivia

October, 1997



CLRC, ISIS Facility
Rutherford Appleton Laboratory
Chilton, Oxon OX11 0QX, U.K.

Contents

1. Introduction

PGXTAL is a collection of (standard-77) FORTRAN subroutines designed to enable crystallographic structures, and density maps, to be displayed within PGPLOT. While several good commercial packages already existed for such plotting, it was felt desirable to also have "free-ware" (subject to the usual civilised conditions) for performing the task; the goal was to have something along the lines of Prof. Tim Pearson's graphics library, PGPLOT, which is widely used within academic circles (especially astronomy, where it all began). Although originating from a crystallographic interest, PGXTAL provides a basic tool-kit for general 3-D rendering. This manual serves both as a tutorial introduction and a reference document for PGXTAL, and for the associated PLOT2D and PGCELL routines.

Not only does PGXTAL try to emulate the structure and philosophy of PGPLOT, in terms of its usefulness and friendliness, it makes explicit use of its graphical infrastructure. As such, all copyright and other conditions that apply to PGPLOT must be respected (for details, see the web-site **http://astro.caltech.edu/~tjp/pgplot/**). While PGXTAL is intimately linked with PGPLOT, it is not an integral part of it — it's an addendum designed to bring some "space-age" functionality to a much-loved graphics library. To have such a high degree of dependency without adhering strictly to the conventions that would allow incorporation is, with hindsight, perhaps a mistake (resulting partly from "historical accidents", and partly from the nature of the problem) but, nevertheless, it all seems to work well with PGPLOT versions 5.0-5.1 (and even 4.9 if not VMS!) on the machines/operating-systems on which it has been tested: VAX, DEC-ALPHA (both OSF and VMS), Silicon Graphics and Linux.

The source-code for PGXTAL, and for the associated PLOT2D and PGCELL routines etc., can be obtained from the web-site **http://www.isis.rl.ac.uk/dataanalysis/dsplot/** . The related requirements for linking with your program are explained at the appropriate points in this document. Section 2 describes the routines for plotting 2-D data, $Z=f(x,y)$, including their 3-D rendering. Section 3 explains the general setup, calling sequence and scope of the PGXTAL library of 3-D rendering routines, and Section 4 gives a formal listing of their specifications. Details about the supporting subroutines, such as PGCELL, are given in the Appendix.

2. Plotting 2-dimensional data

Let's begin by describing a few subroutines designed to plot 2-D data: $Z=f(x,y)$. We should note that the term data is used rather loosely here, in that no account is taken of error-bars and so on; it would be more apt, therefore, to talk about a function defined numerically over a 2-dimensional grid of points. While subroutines for handling this case already exist in PGPLOT (such as PGCONT, PGHI2D and PGIMAG), a series of program-like routines, generically called PLOT2D, have been developed that provide easy access for such plotting, including the added ability to do 3-D rendering, that don't require explicit knowledge of PGPLOT.

2.1 A rectangular mapping

The simplest, and most common, case concerns the plotting of the function $Z=f(x,y)$ where the x and y coordinates are directly proportional to the values of the I and J indices,

respectively, of a FORTRAN array Z(I,J) which defines the data on a uniform rectangular grid of points. In order to accomplish this, all you have to do is make the following subroutine call in your program (*i.e.* no PGPLOT calls are required):

CALL PLOT(X, NX, Y, NY, Z, N1, N2, W, SIZE, IWIDTH, XLBL, YLBL, TITL)

Here the data are passed down in the two-dimensional FORTRAN array Z, with dimensions declared as (N1,N2), of which the first NX x NY elements are to be plotted; *in general, it's best to ensure that N1=NX and N2=NY since this is a requirement for using the 3-D rendering option.* The numerical values of the *x* and *y* coordinates for the grid-points should be given in the arrays X and Y, and W should be available for use as a work-space with at least NX elements. The character strings XLBL, YLBL and TITL refer to the desired annotation for the X-axis, Y-axis and title; their size and boldness is specified by the parameters SIZE (typically 1.5) and IWIDTH (typically 1 for an interactive device and 3 for a hardcopy). **Caution: the data in the Z-array may be changed on output!**

In order to use the above PLOT subroutine, your program needs to be linked with the object files **plot2db3**, **pgcell**, **dsqinf**, **pgxtal** and the PGPLOT library. Thus, for example, the link command on a VMS machine at the ISIS facility would take the form:

```
link program, ..., plot2db3, pgcell, dsqinf, pgxtal, pgplot/opt
```

where as the (digital) UNIX equivalent would be:

```
f77 program.f ... plot2db3.o pgcell.o dsqinf.o pgxtal.o -lpgplot -lX11
```

There are, in fact, a number of alternative options for the object files that could be linked, depending on the desired aesthetic properties of the plot: **pgcel0** could be used instead of pgcell, and plot2db3 could be replaced with **plot2d3**, **plot2db** or **plot2d**. The differences are described below:

pgcell : the input 2-D array Z is displayed through a linear interpolation to the resolution of the graphics device, so that a pleasant photograph-like output is obtained.

pgcel0 : the output is a faithful bin-like rendering of the input 2-D array Z, giving a rather boxy appearance for small NX and NY (essentially equivalent to using the PGPLOT routine PGIMAG).

plot2db3 : the tick-marks for the *x* and *y* axes are on the outside of the plotting box, and there is numerical annotation of the vertical *z*-axis.

plot2d3 : the tick-marks for the *x* and *y* axes are on the inside of the plotting box, and there is no numerical annotation of the vertical *z*-axis.

plot2db : same as plot2db3, except that there is no (proper) 3-D rending option; therefore, pgxtal does not need to be linked in with your program.

plot2d : same as plot2d3, except that there is no (proper) 3-D rending option; therefore, pgxtal does not need to be linked in with your program.

When you run your program, the PLOT subroutine will prompt you for various options before plotting the 2-dimensional data; the questions should all be self-explanatory, and also have sensible defaults (so that you can just hit the carriage-return, even if you're not sure). The first choice that you will be given is:

```
(0) Contour
(1) Surface
(2) Colour: Grey-Scale
(3) Colour: Heat
(4) Colour: Rainbow Spectrum
(5) Colour: BGYRW
(6) Colour: Serpent
(7) Colour: Read in from file

PLOT> Type ? :
```

The default option is 0, which simply gives a contour-plot of the function $Z = f(x,y)$, and should work on all graphics devices; you will then be asked whether you want linearly-spaced contours, and how many, or else given the opportunity to type in your preferred levels. All the other options (1 to 7) require a graphics device that can support at least 16 "colours"; you will be prompted for this just prior to the rendering:

```
Graphics device/type (? to see list, default /NULL):
```

Suitable choices include: X-Windows (/XW or /XS), PostScript (/CPS or /VCPS, which produces the file **pgplot.ps**), GIF (/GIF and /VGIF which produces the file **pgplot.gif**).

Option 2 produces a black-and-white photograph-like image, where as 3 to 6 render the different Z-levels with various colours depending on the table that is chosen (Heat, Rainbow, Blue-Green-Yellow-Red-White or Serpent). If you don't like any of the built-in colour-tables (taken from *STARLINK*, and stored in the include file COLTABS.INC), then you can provide your own in a file for which you will be prompted if you choose option 7; it should have a three-column ASCII format, listing the desired RGB values (all between 0.0 and 1.0) for the red, green and blue intensities of the colours. With options 2 to 7, you will have the opportunity to modify the colour-table at run-time through the use of a "contrast factor"; the default is 1.0, but you might find it helpful to use 0.7 (say) to highlight low-lying features or something like 1.3 when the dominant variations occur at the highest Z-values. You will also be able to overlay contours on the colour maps.

Option 1 generates a 3-D rendering of the function $Z=f(x,y)$. Its use entails compliance with some additional conditions (that are easily, and generally, satisfied): (i) the dimensional declaration N1 and N2 for the array Z in the call to the subroutine PLOT must be the same as the desired output NX and NY; and (ii) the numerical values of the x and y coordinates in the arrays X and Y must be in ascending order (so that X(NX) > X(1) and Y(NY) > Y(1)). You will be asked, at run-time, for your choices regarding the colouring, shading, orientation and Z-range of the 3-D surface that is to be rendered; the illumination is fixed to shine diagonally over your right-hand shoulder. As well as the "sensible" default settings, a nice effect with noisy data can often be achieved by viewing the surface end-on (tilt = 90°) and using a shaded colour-table (for example, Colour table = 4, No. of colour-bands = 16, not shiny, with Diffusiveness = 0.7).

2.2 A non-linear mapping

The more general case of plotting the function $Z=f(x,y)$ where the x and y coordinates are related non-linearly to the I and J indices of a FORTRAN array Z(I,J), which defines the data on a uniform rectangular grid of points, is accomplished by making the following subroutine call (instead of the previous one to PLOT):

CALL PLT2DX(Z, N1, N2, I1, I2, J1, J2, XMIN, XMAX, YMIN, YMAX, SIZE, IWIDTH, XLBL, YLBL, TITL)

Here the data are passed down in the two-dimensional FORTRAN array Z, with dimensions declared as (N1,N2), of which a subset of elements is to be plotted between (I1, J1) and (I2, J2); the range of the output, or *world*, x and y coordinates that is to be viewed lies between XMIN and XMAX, and YMIN and YMAX. The character strings XLBL, YLBL and TITL refer to the desired annotation for the X-axis, Y-axis and title; their size and boldness is specified by the parameters SIZE (typically 1.5) and IWIDTH (typically 1 for an interactive device and 3 for a hardcopy). We should again caution that the data in the Z-array may be changed on output! *The run-time prompts for the PLT2DX subroutine are identical to those of PLOT, except that the 3-D surface rendering option (1) is not operational.*

In order to use PLT2DX, your program will have to be linked with the object files **plot2dbx**, **pgcelx**, **pgcell**, **dsqinf** and the PGPLOT library. Thus, for example, the link command on a VMS machine at the ISIS facility would take the form:

```
link program, ..., plot2dbx, pgcelx, pgcell, dsqinf, pgplot/opt
```

where as the (digital) UNIX equivalent would be:

```
f77 program.f ... plot2dbx.o pgcelx.o pgcell.o dsqinf.o -lpgplot -lX11
```

There is the option of linking with the object file **plot2dx** instead of plot2dbx if you want the tick-marks for the x and y axes to be on the inside of the plotting box, rather than the outside, and if you don't want the colour-table to be annotated numerically.

In addition to the above object files, you must supply the subroutines **TRNL** and **TRNLB** that define your non-linear mapping:

SUBROUTINE TRNL(XW, YW, XI, YJ)

should return the (fractional) array indices XI and YJ which correspond to the x and y (world) coordinates XW and YW, and

SUBROUTINE TRNLB(XW, YW, XI, YJ)

should return the x and y (world) coordinates XW and YW that correspond to the (floated) array indices XI and YJ. Suppose, for example, that the I-index of the Z-array referred to a radius r and that the J-index corresponded to an angle θ; that is to say, in FORTRAN:

```
RADIUS = RSCL*FLOAT(I) + R0
THETA = TSCL*FLOAT(J) + T0
```

This would be mapped to our output Cartesian coordinates x and y according to the simple polar transformation: $x = r\cos(\theta)$ and $y = r\sin(\theta)$. Thus, the central part of subroutine TRNLB would take the form:

```
XW = (RSCL*XI + R0) * COS(TSCL*YJ + T0)
YW = (RSCL*XI + R0) * SIN(TSCL*YJ + T0)
```

where, after having been suitably initialised, the scaling and off-set constants RSCL, R0, TSCL and T0 could be stored in an appropriate COMMON BLOCK. The corresponding lines for the subroutine TRNL would be:

```
XI = (SQRT(XW**2+YW**2) - R0) / RSCL
YJ = (ATAN2(YW,XW) - T0) / TSCL
```

We should point out that PLT2DX simply carries out a point-to-point mapping, and does not take into account how the density of the points changes. Therefore, the Z-array should be pre-multiplied by the ***Jacobian*** if this is deemed to be appropriate.

2.3 Spherical plots

A specific example of a non-linear mapping that has been coded up (so that you don't need to write the TRNL or TRNLB subroutines for it) is for the case of plotting a 2-D function on the surface of a sphere: $Z=f(\phi,\theta)$, where ϕ and θ are the *longitude* and *co-latitude* respectively. In order to accomplish this, all you need to do is make the following call (instead of the previous one to PLT2DX):

```
CALL GLOBE(Z, THTMIN, PHIMIN, NTHT, NPHI, DTHT, DPHI, SIZE, IWIDTH,
           XLBL, YLBL, TITL)
```

Here the data are passed down in the two-dimensional FORTRAN array Z, with dimensions declared as (NPHI, NTHT), where the (1,1)-element corresponds to (ϕ=PHIMIN, θ=THTMIN) and the incremental constants for the longitude and co-latitude are given by DPHI and DTHT; *all the angles are assumed to be in degrees*. The character strings XLBL, YLBL and TITL refer to the desired annotation for the X-axis, Y-axis and title (although the first two are probably not very meaningful); their size and boldness is specified by the parameters SIZE (typically 1.5) and IWIDTH (typically 1 for an interactive device and 3 for a hardcopy). We should again caution that the data in the Z-array may be changed on output!

In order to use GLOBE, your program will have to be linked with the object files **globe**, **pgcelx**, **pgcell**, **dsqinf** and the PGPLOT library. Thus, for example, the link command on a VMS machine at the ISIS facility would take the form:

```
link program, ..., globe, pgcelx, pgcell, dsqinf, pgplot/opt
```

where as the (digital) UNIX equivalent would be:

```
f77 program.f ... globe.o pgcelx.o pgcell.o dsqinf.o -lpgplot -lX11
```

The run-time prompts for the GLOBE subroutine are identical to those of PLOT discussed in Section 2.1 (but with option 1 not operational), except for a couple of additional preliminary questions regarding the viewing of the sphere. These are to do with the distance to the sphere, which controls the perspective, and the orientation, which is fixed through the choice of three Euler angles.

2.4 A disconnected mapping

Very occasionally, the mapping of a disconnected space is required; this could be the case, for example, if the data are collected in several sets of detector-banks that are separated by large distances. In order to accomplish such a general 2-D colour plot, all you need to do is make the following call (*i.e.* no PGPLOT calls are required):

CALL PLT2DZ(ZMIN, ZMAX, XMIN, XMAX, YMIN, YMAX, SIZE, IWIDTH, XLBL, YLBL, TITL)

Here the function $Z=f(x,y)$ will be plotted in the x and y coordinate limits of XMIN and XMAX, and YMIN and YMAX, and in the Z-range ZMIN to ZMAX. The value of Z at a given point, with (world) coordinates XW and YW, should be returned by the subroutine **TRNL**, which you must supply, through a parameter ZCOLOR which is scaled so that it is 0.0 for ZMIN and 1.0 for ZMAX:

```
SUBROUTINE TRNL(XW, YW, ZCOLOR)
```

As usual, the character strings XLBL, YLBL and TITL refer to the desired annotation for the X-axis, Y-axis and title; their size and boldness is specified by the parameters SIZE (typically 1.5) and IWIDTH (typically 1 for an interactive device and 3 for a hardcopy).

In order to use PLT2DZ, your program will have to be linked with the object files **plot2dbz, pgcell, dsqinf** and the PGPLOT library. Thus, for example, the link command on a VMS machine at the ISIS facility would take the form:

```
link program, ..., plot2dbz, pgcell, dsqinf, pgplot/opt
```

where as the (digital) UNIX equivalent would be:

```
f77 program.f ... plot2dbz.o pgcell.o dsqinf.o -lpgplot -lX11
```

The run-time prompts are identical to those of PLOT discussed in Section 2.1, except that options 0 and 1 are not available and there is no opportunity for contour overlay.

2.5 Subroutine specifications

The formal calling specifications of the subroutines that have been discussed in this section, on the plotting of 2-D data, are listed below.

2.5.1 PLOT — plot a 2-D data array, with a simple rectangular mapping

```
      SUBROUTINE PLOT(X,NX,Y,NY,Z,N1,N2,W,SIZE,IWIDTH,XLBL,YLBL,TITL)
C     ------------------------------------------------------------
C
      REAL          X(*),Y(*),Z(N1,N2),W(*)
      CHARACTER*(*) XLBL,YLBL,TITL
C
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C
C Purpose
C      This subroutine plots "data" defined on a regularly-spaced
C   rectangular grid of points Z(I,J). With the default choice for the
C   PGCELL routine that is linked, the output is a linearly-interpolated
C   map (rather than coarse rectangular boxes).
C      Note that for the proper 3-D rendering option, it is required that
C   NX=N1 and NY=N2; in addition, the numerical values in the X and Y
C   arrays must be in ascending order.
C
C Parameters
C   ARGUMENT  TYPE  I/O  DIMENSION  DESCRIPTION
C    Z        R*4    I    N1 x N2   The rectangular "data"-array.
C    Z        R*4    O    N1 x N2   A scaled, and clipped, version of
C                                    the input array(!).
C    N1       I*4    I      -       The first dimension of array Z.
C    N2       I*4    I      -       The second dimension of array Z.
C    X        R*4    I      NX      Array of X-coordinates.
C    NX       I*4    I      -       Number of X-pixels to be plotted
C                                   (usually = N1, but must be <= N1).
C    Y        R*4    I      NY      Array of Y-coordinates.
C    NY       I*4    I      -       Number of Y-pixels to be plotted
C                                   (usually = N2, but must be <= N2).
C    W        R*4    I      NX      Work array, at least NX long.
C    SIZE     R*4    I      -       Character-size for plot (try 1.5).
C    IWIDTH   I*4    I      -       Line-width for plot (try 2).
C    XLBL     A*1    I     *(*)     Label for X-axis.
C    YLBL     A*1    I     *(*)     Label for Y-axis.
C    TITL     A*1    I     *(*)     Title for plot.
C
C Globals
C    COLTABS.INC
C
C History
C   Initial release.                                    DSS:  3 Jul 1992
C   Minor changes to conform with new PGCELL.           DSS:  6 Feb 1995
C   Put in option to over-lay contours.                 DSS: 21 Feb 1995
C   Now has proper 3-d surface rendering.               DSS: 27 Aug 1997
C   Fortran made LINUX-friendly!                        DSS: 15 Sep 1997
C-----------------------------------------------------------------------
```

In order to use the above PLOT subroutine, your program needs to be linked with the *object files* **plot2db3**, **pgcell**, **dsqinf**, **pgxtal** and the PGPLOT *library*. There are, in fact, a number of alternative options for the object files that could be linked, depending on the desired aesthetic properties of the plot: **pgcel0** could be used instead of pgcell, and plot2db3 could be replaced with **plot2d3**, **plot2db** or **plot2d** (the non-3 vesions don't require pgxtal). See section 2.1 for details.

2.5.2 PLT2DX — plot a 2-D data array, with a non-linear mapping

```
      SUBROUTINE PLT2DX(Z,N1,N2,I1,I2,J1,J2,XMIN,XMAX,YMIN,YMAX,
     *                  SIZE,IWIDTH,XLBL,YLBL,TITL)
C     ------------------------------------------------------------
C
      REAL          Z(N1,N2)
      CHARACTER*(*) XLBL,YLBL,TITL
      EXTERNAL      TRNL,TRNLB
C
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C
C Purpose
C      This subroutine takes "data" defined on a rectangular grid of
C    points Z(I,J) and plots out a linearly-interpolated map according
C    to a user-defined non-linear transform. To this end, the user must
C    supply the two subroutines TRNL and TRNLB; each will be called with
C    four arguments:
C        SUBROUTINE TRNL(XW,YW,XI,YJ)
C    should return the real (fractional) array indices XI and YJ which
C    correspond to the (real) world coordinates XW and YW;
C        SUBROUTINE TRNLB(XW,YW,XI,YJ)
C    should return the (real) world coordinates XW and YW corresponding
C    to the (real) array indices XI and YJ.
C      The appropriate Jacobian term should be factored into the array
C    Z(I,J) before it is passed down (by pre-multiplication).
C
C Parameters
C   ARGUMENT  TYPE  I/O  DIMENSION  DESCRIPTION
C    Z        R*4    I    N1 x N2   The rectangular "data"-array.
C    Z        R*4    O    N1 x N2   A scaled, and clipped, version of
C                                    the input array(!).
C    N1       I*4    I       -      The first dimension of array Z.
C    N2       I*4    I       -      The second dimension of array Z.
C    I1,I2    I*4    I       -      The inclusive range of the first
C                                    index (I) to be plotted.
C    J1,J2    I*4    I       -      The inclusive range of the second
C                                    index (J) to be plotted.
C    XMIN     R*4    I       -      Xmin for plot, in output units.
C    XMAX     R*4    I       -      Xmax for plot, in output units.
C    YMIN     R*4    I       -      Ymin for plot, in output units.
C    YMIN     R*4    I       -      Ymax for plot, in output units.
C    SIZE     R*4    I       -      Character-size for plot.
C    IWIDTH   I*4    I       -      Line-width for plot.
C    XLBL     A*1    I     *(*)     Label for X-axis.
C    YLBL     A*1    I     *(*)     Label for Y-axis.
C    TITL     A*1    I     *(*)     Title for plot.
C
C Globals
C    COLTABS.INC
C
C History
C   D. S. Sivia      15 Feb 1995  Initial release.
C   D. S. Sivia      21 Feb 1995  Put in option to over-lay contours.
C   D. S. Sivia      27 Aug 1997  No longer use non-standard Q-format.
C   D. S. Sivia      15 Sep 1997  Fortran made LINUX-friendly!
C-----------------------------------------------------------------------
```

To use the PLT2DX subroutine, your program needs to be linked with the *object files* **plot2dbx**, **pgcelx**, **pgcell**, **dsqinf** and the PGPLOT *library*; as a minor option, plot2dbx could be replaced with **plot2dx**. See section 2.2 for details.

2.5.3 GLOBE — plot a 2-D data array, on the surface of a sphere

```
      SUBROUTINE GLOBE(Z,THTMIN,PHIMIN,NTHT,NPHI,DTHT,DPHI,SIZE,IWIDTH,
     *                 XLBL,YLBL,TITLE)
C     ------------------------------------------------------------------
C
      REAL            Z(NPHI,NTHT)
      CHARACTER*(*)   XLBL,YLBL,TITLE
C
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C
C Purpose
C      This is a "front-end" for the globe-plotting routine PLTSPH; all
C    the angle parameters should be passed down in degrees. PLTSPH is
C    itself a globe-specific version of PLT2DX: it takes "data" defined
C    on a rectangular grid of points Z(I,J), corresponding to latitude
C    and longitude, and plots out a linearly-interpolated map on the
C    surface of a sphere.
C      The appropriate Jacobian term should be factored into the array
C    Z(I,J) before it is passed down (by pre-multiplication).
C
C Parameters
C   ARGUMENT  TYPE  I/O  DIMENSION  DESCRIPTION
C    Z        R*4    I  NPHI x NTHT The rectangular "data"-array.
C    Z        R*4    O  NPHI x NTHT A scaled, and clipped, version of
C                                    the input array(!).
C    THTMIN   R*4    I       -      The min. value of co-latitude theta.
C    PHIMIN   R*4    I       -      The min. value of longitude phi.
C    NPHI     I*4    I       -      The first dimension of array Z.
C    NTHT     I*4    I       -      The second dimension of array Z.
C    DTHT     R*4    I       -      The theta increment for array Z.
C    DPHI     R*4    I       -      The phi increment for array Z.
C    SIZE     R*4    I       -      Character-size for plot.
C    IWIDTH   I*4    I       -      Line-width for plot.
C    XLBL     A*(*)  I       -      The x-axis label for the plot.
C    YLBL     A*(*)  I       -      The y-axis label for the plot.
C    TITLE    A*(*)  I       -      The title for the plot.
C
C Globals
C    COLTABS.INC
C
C History
C   D. S. Sivia      21 Feb 1995  Initial release.
C   D. S. Sivia      27 Feb 1995  Corrected minor problem with contours.
C   D. S. Sivia      27 Aug 1997  No longer use non-standard Q-format.
C   D. S. Sivia      15 Sep 1997  Fortran made LINUX-friendly!
C-----------------------------------------------------------------------
```

In order to use the above GLOBE subroutine, your program needs to be linked with the *object files* **globe**, **pgcelx**, **pgcell**, **dsqinf** and the PGPLOT *library*. See section 2.3 for details.

2.5.4 PLT2DZ — plot a 2-D data array, with a disconnected mapping

```
      SUBROUTINE PLT2DZ(ZMIN,ZMAX,XMIN,XMAX,YMIN,YMAX,SIZE,IWIDTH,XLBL,
     •                  YLBL,TITL)
C     ------------------------------------------------------------------
C
      CHARACTER*(*) XLBL,YLBL,TITL
C
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C
C Purpose
C      This subroutine plots a 2D colour map given the users own
C    (peculiar) mapping. To this end, the following routine must be
C    supplied by the user:
C        SUBROUTINE TRNL(XW,YW,ZCOLOR)
C    should return the colour-scale, between 0.0 (for ZMIN) and 1.0
C    (for ZMAX), which corresponds to the (real) world coordinates XW
C    and YW.
C
C Parameters
C   ARGUMENT  TYPE  I/O  DIMENSION  DESCRIPTION
C    ZMIN     R*4    I       -      Lowest "intensity" for the plot.
C    ZMAX     R*4    I       -      Highest "intensity" for the plot.
C    XMIN     R*4    I       -      Xmin for plot, in output units.
C    XMAX     R*4    I       -      Xmax for plot, in output units.
C    YMIN     R*4    I       -      Ymin for plot, in output units.
C    YMIN     R*4    I       -      Ymax for plot, in output units.
C    SIZE     R*4    I       -      Character-size for plot.
C    IWIDTH   I*4    I       -      Line-width for plot.
C    XLBL     A*1    I     *(*)     Label for X-axis.
C    YLBL     A*1    I     *(*)     Label for Y-axis.
C    TITL     A*1    I     *(*)     Title for plot.
C
C Globals
C    COLTABS.INC
C    grpckg1.inc
C
C History
C   D. S. Sivia      15 Feb 1995  Initial release.
C   D. S. Sivia      21 Feb 1995  Put in option to over-lay contours.
C   D. S. Sivia       8 Aug 1995  Modified PLOT2DBX to PLOT2DBZ.
C   D. S. Sivia      28 Sep 1995  Fixed "bug" with the contrast-factor.
C   D. S. Sivia      27 Aug 1997  No longer use non-standard Q-format.
C   D. S. Sivia      15 Sep 1997  Fortran made LINUX-friendly!
C-----------------------------------------------------------------------
```

In order to use the above PLT2DZ subroutine, your program needs to be linked with the *object files* **plot2dbz**, **pgcell**, **dsqinf** and the PGPLOT *library*. See section 2.4 for details.

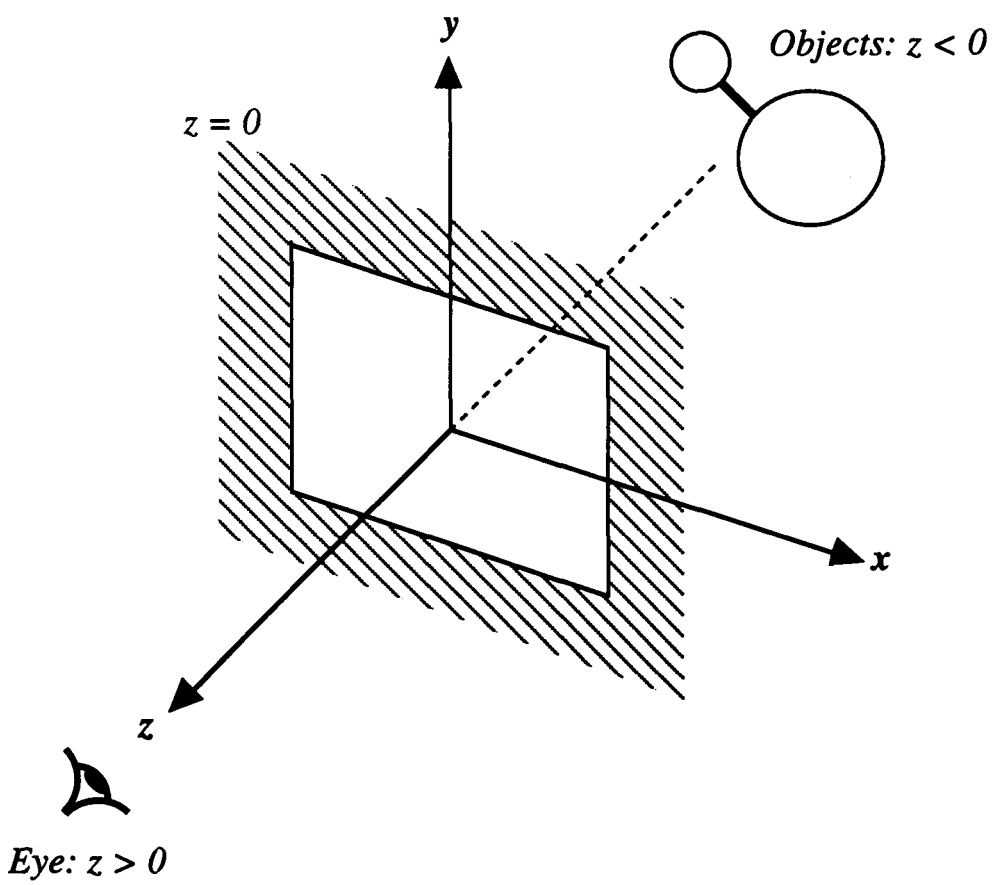
If you have to compile the FORTRAN file plot2dbz.f, note that you will need to have the PGPLOT include file grpckg1.inc in the directory (as well as the include file COLTABS.INC); make sure that it is consistent with the version of the PGPLOT library to which you are linking (it's slightly different between 5.0 and 5.1, for example)!

3. Plotting 3-dimensional data and objects

Now let's turn to the main body of the software designed to enable the 3-D rendering of objects, like ellipsoids and cylinders, and 3-dimensional data $\rho = f(x,y,z)$, within PGPLOT. This is contained in the file pgxtal.f, and comprises of a collection of FORTRAN subroutines that make up the PGXTAL library. The code should adhere to the 77 standard, apart from a system-specific random number generator RAN; if your compiler doesn't support this, you will have to provide a FUNCTION RAN(ISEED) which generates a *real* number between 0.0 and 1.0 (with the initial value of ISEED being a large odd *integer*). Unlike the 2-D plotting subroutines discussed in the previous section, which were program-like utilities, PGXTAL constitutes a set of building blocks that you must put together in the context of your own requirements. As such, it is very similar to PGPLOT itself; in many ways, it's an addendum to it!

3.1 Setting the scene

3-D rendering is rather like painting a picture: you look out at the world and put colour down onto a canvas in a way that, with suitable shading and perspective, reflects what you see. This analogy is central to the whole philosophy of PGXTAL, and we'll return to it periodically. In particular, we imagine that we are looking out at our world of selected 3-D objects through a rectangular window in an otherwise opaque (and infinite) wall; this is shown schematically in the diagram below:



In terms of a reference Carterisan coordinate system, this window on the world (which defines our canvas) is taken to be at $z = 0$. In accordance with the convention of a right-handed set, if left-to-right is the x-axis, and bottom-to-top is the y-axis, the z-axis must be out of the paper. That is to say, the view-point of the **EYE** must be at $z > 0$ and all the (seen) objects have $z < 0$. Thus if the objects were to be illuminated "head-on", then the direction of the **LIGHT** would be given by the vector (0,0,-1); a more pleasant shading effect is usually achieved when the light shines diagonally from over our right-hand shoulder, so that the corresponding direction is more like (-1,-1,-1).

While the relative distance between the eye and the objects controls the degree of the apparent perspective, it is the fraction of the visible scene that they occupy which determines their size. Moving the eye closer to the window has the paradoxical affect of making the objects look smaller, therefore, as a much larger solid angle can then be projected on to the canvas! The dimensions of the window are set by giving the lower and upper bounds of its x and y coordinates: XMIN, XMAX, YMIN and YMAX. This can be achieved by calling the PGPLOT routine PGENV (after first having called PGBEGIN or PGOPEN):

CALL PGENV(XMIN, XMAX, YMIN, YMAX, 1, -2)

or by using an equivalent set of constituent subroutines that allow greater flexibility; that is to say: PGPAPER, PGPAGE, PGVPORT, PGWINDOW and PGBOX. Having firmly fixed the coordinate system in this manner, we are now ready to turn our attention to the properties of the canvas upon which we shall be painting.

3.2 Initialising, and revealing, the canvas

The first thing we have to do, before we can embark on any sort of 3-D rendering, is choose a suitable piece of paper for our drawing. Within the computer context of PGXTAL, this boils down to the task of initialising a *software buffer*. It is accomplished through a call to the subroutine **SBINIT**:

CALL SBINIT(RGB, IC, IBMODE, IBUF, MAXBUF)

Here RGB is a 3-element array that controls the colour of the background (the red, green and blue components, all between 0.0 and 1.0), and IC is an associated colour-index that will be discussed further in the next subsection (but is typically set to 17). MAXBUF is an integer that is returned by the routine, and tells you how many sheets of drawing paper you can have; this is of no consequence for simple plotting, but will be important if you're interested in playing a "movie" with PGXTAL (see section 3.7). IBUF indicates which piece of paper you wish to use, and should be set equal to 1 unless you're making a movie. IBMODE is a control parameter that should also generally be set to 1. A choice of 2 flags the fact that you may want to save an incomplete picture for subsequent use (which will reduce the size of MAXBUF), through a call to **SBFSAV** at the appropriate point. You can later continue to work on this picture by a calling SBINIT with IBMODE = 3, because this initialises the chosen canvas with the contents of the previously saved buffer. This mechanism enables complicated pictures that are similar apart from a few details to be generated quickly, by repeatedly starting from a common template. A

fancy alternative to beginning with a clean sheet of paper of a single given colour (IBMODE = 1) is provided by the subroutine **SBFBKG**, which allows you to have a shaded background.

Just as any great painting is not revealed until the picture is complete, so too is the case for our 3-D rendering. Within the context of PGXTAL, this unveiling is accomplished through a call to the subroutine **SBFCLS**:

CALL SBFCLS(IBUF)

IBUF is, of course, the integer that indicates which (of up to MAXBUF) canvases we wish to display. Since, in general, the painting will have been started with a call to SBINIT with IBUF=1, SBFCLS should also be called with this value. While it may be possible to add objects to the scene after having viewed the picture (if SBINIT has not been subsequently called with the same IBUF), and then redisplayed with SBFCLS, they cannot be removed from the painting because they loose their attributes as individual entities once they have been rendered. In any case, if a hardcopy graphics device has been chosen (for example /GIF or /CPS) then SBFCLS must be followed by a call to the PGPLOT subroutine PGEND or PGCLOSE.

3.3 Initialising the colour palettes

3-D rendering requires the ability to use different colours and shades; therefore, you must choose a graphics device that supports a good few them. The number available can be ascertained by a call to the PGPLOT subroutine PGQCOL:

CALL PGQCOL(ICMIN, ICMAX)

ICMIN is usually 0, and ICMAX should be at least around 30; the largest value of ICMAX is likely to be 255, thus giving access to 256 colours, as PGPLOT doesn't (currently) provide true-colour support (and PGXTAL would also have to be modified to use it).

PGXTAL has three subroutines that assign shades of colours to a given range of colour indices; the most commonly used one is **COLINT**:

CALL COLINT(RGB, IC1, IC2, DIFUSE, SHINE, POLISH)

Here RGB is a 3-element array that controls the colour of a "fully-lit" object: for example, (1.0,0.0,0.0) would be pure red, (0.0,0.0,1.0) pure blue, and (0.5,0.5,0.5) grey. Different shades of this RGB specification are assigned to colour indices IC1 to IC2, and this constitutes a *palette* that can subsequently be used to render all object having the same colour attributes. The actual shading can either be appropriate for a shiny (metallic) object or a diffusively-lit (matt) one, and is controlled by the parameters DIFUSE, SHINE and POLISH. Without true-colour support, it is recommended that SHINE=0.0 if DIFUSE>0.0 (but ≤ 1.0), and vice versa. SHINE determines the whiteness of the shiny reflections (or greyness if <1), and POLISH their size, where as DIFUSE specifies how dark an object should become if it is in the shade (1 means totally black); POLISH is generally set to 1.0 (even when SHINE=0.0, when it has no effect).

The other two PGXTAL subroutines that enable colour palettes to be set up are **COLSRF** and **COLTAB**; these are appropriate for the rendering of two and three-dimensional data (rather than objects), respectively, and will be discussed in sections 3.5. One thing they do have in

common with COLINT is the specification of the range of colour-indices, IC1 and IC2, that are to be used. It is best not to use the first 4 (usually 0 to 3), and preferably not the first 16 (0 to 15), as these are often pre-defined colours that are already being used for other purposes. Apart from one index that is required for the background (in SBINIT), the remainder (up to ICMAX) can be split evenly between the number of colour-types that are needed for the different objects.

3.4 Drawing objects

The objects that are to be drawn can placed in the scene by PGXTAL calls to a basic set of simple geometrical entities. Thus, for example, a sphere is painted by a call to the subrouitne **SBBALL**:

CALL SBBALL(EYE, CENTRE, RADIUS, IC1, IC2, LIGHT, LSHINE, X0, Y0, R0)

The location of the sphere is passed down in the 3-element array CENTRE, and its size is given by the parameter RADIUS. Remember, of course, that the PGXTAL viewing convention requires that the z-coordinate of CENTRE plus the RADIUS must be negative for this ball to be visible; that is, all objects have to be entirely "behind the wall with the window"! ***EYE and LIGHT give the coordinates of the vantage-point (with $z>0$) and the direction of the illumination, and should be the same for all the objects that are to be drawn.*** The sphere will be painted with shades taken from colour-indices IC1 to IC2, which should previously have been set up with an appropriate call to COLINT; if it is to be shiny, then LSHINE should be set to `.TRUE.` (otherwise `.FALSE.`). The position and size of the projection of the sphere on the canvas is returned in X0, Y0 and R0, should you need them.

Other PGXTAL subroutines for drawing 3-D objects include **SBELIP** (an ellipsoid), **SBPLAN** (a planar polygon), **SBROD** (a cylinder, with a polygon cross-section), **SBCONE** (a cone, with a polygon base) and **SBLINE** (a thin line); in addition, **SBTEXT** allows text to be written with perspective. This small collection of entities can form the building blocks of more complex objects: four triangular planes can be combined to give a tetrahedron; a rod and cone yield a 3-D arrow; and so on. There are also "see-through" variants of the sphere and plane drawing subroutines, called **SBTBAL** and **SBPLNT**. Thus, for example, the call to SBPLNT:

CALL SBPLNT(EYE, NV, VERT, IC1, IC2, LIGHT, ITRANS)

draws an NV-sided polygon, with vertices given in the 3xNV array VERT, with a *transparency* level of ITRANS: a value of 0 makes it completely opaque, so that it's just like calling SBPLAN, while the (usually) best option of 2 makes it 50% transparent (1 and 3 give a see-through level of 25% and 75%, respectively).

3.5 Rendering 2 and 3-dimensional data

In X-ray crystallography, if a molecular structure is not defined in terms of distinct atoms and bonds, then it tends to be represented as an electron density map. That is to say, the *unit cell* is divided into a uniform 3-D grid of discrete points and a density ρ assigned to each one of them: $\rho = \rho(i,j,k)$, where the (integer) array-indices i, j and k go from 0 to N1, 0 to N2 and 0 to N3 respectively. In fact, *the formal properties of a unit cell require that the density*

"wraps around", so that ρ (0,j,k)=ρ (N1,j,k) and so on. Such a three-dimensional set of data can be displayed by high-lighting a given iso-surface, ρ=constant, which can be accomplished within PGXTAL with a call to **SBSURF**:

CALL SBSURF(EYE, LATICE, DENS, N1, N2, N3, DSURF, IC1, IC2, LIGHT, LSHINE)

Here the parameters EYE, LIGHT, IC1, IC2 and LSHINE are as for SBBALL discussed in section 3.4, and DENS is the 3-dimensional density array ρ(0:N1,0:N2,0:N3) mentioned above. The size, skewness and orientation of the unit cell is given by the (x,y,z) coordinates of the origin **o** and the **a**, **b**, and **c** lattice-vectors passed down in the array LATICE; as usual, to be visible, every point in it must have $z<0$. Since electron densities are positive, by definition, the iso-surface to be displayed should also have ρ=DSURF>0.0; in essence, it is assumed that the "object" being viewed is opaque (or solid) for ρ < DSURF and transparent for ρ > DSURF. In a more general context, negative densities can be displayed quite easily by simply pre-multiplying DENS by -1! Indeed, even two (positive) iso-surfaces can be viewed simultaneously if a see-through variant **SBTSUR** is called for the one with the smaller value of DSURF; this subroutine is identical to SBSURF, except for the additional parameter ITRANS that controls the level of the transparency (just as in SBPLNT mentioned in the previous section).

An alternative way of visualising the 3-dimensional data ρ (i,j,k) is to draw a coloured contour-map of a 2-dimensional slice through the unit cell (in perspective); this is accomplished with a call to the subroutine **SBSLIC**:

CALL SBSLIC(EYE, LATICE, DENS, N1, N2, N3, DLOW, DHIGH, IC1, IC2, SLNORM, APOINT, ICEDGE)

The thin slice that is to be colour-contoured is defined by giving the (world) coordinates of a point within the chosen plane, APOINT, and the direction of the normal to it, SLNORM ; it is density-shaded according to the assignment of the colour-indices between IC1 and IC2, in the range DLOW < ρ < DHIGH. A suitable colour-table can be set up with a call to **COLTAB**:

CALL COLTAB(RGB, NCOL, ALFA, IC1, IC2)

which serves a function very similar to that of the subroutine COLINT discussed in section 3.3. Here RGB is a 3xNCOL array which lists the red, green and blue components of the colours (all between 0.0 and 1.0) that are to be associated with the range of densities to be displayed. The parameter ALPHA should normally be set to 1.0, but can be varied to modify the "contrast" of the input colour-table: a value of about 0.7 can be useful for highlighting variations in the density ρ around DLOW, where as 1.4 would be better if the dominant changes were centred about DHIGH. Finally, returning to SBSLIC, the integer ICEDGE controls whether or not a border is plotted around the perimeter of the slice being colour-contoured.

While on the topic of unit cells, we should also mention the subroutine **SBCPLN** : this allows an ordinary, semi-transparent, coloured plane to be drawn within the bounds of the lattice (so that it has nothing to do with densities *per se*).

A special case of the density maps considered above is a 2-D variant ρ = ρ (i,j) , where N3 =0; such a two dimensional lattice could arise, for example, when considering the surface

properties of a crystal. Apart from displaying this with simple colour-contours (using the PGPLOT routines PGCONT or PGIMAG etc.) there is the option of rendering it as a 3-D surface, by plotting ρ in a direction normal to that of the i and j axes. This can be accomplished within PGXTAL by calling the subroutine **SB2SRF**:

CALL SB2SRF(EYE, LATICE, DENS, N1, N2, DLOW, DHIGH, DVERT, IC1, IC2, NCBAND, LIGHT, LSHINE)

Most of the parameters are identical to those for SBSURF, apart the missing N3 and the related omission of the coordinates of the **c** lattice-vector in the array LATICE. There are, however, two additional variables DVERT and NCBAND; both relate to the "vertical" direction in which ρ is plotted. DVERT specifies the (world) length, or height, to be associated with the density range DLOW to DHIGH; and NCBAND gives the number of different colours to be used for representing the value of the density, so that each band will have (IC2-IC1+1)/NCBAND shades of light and dark. The corresponding initialisation of the colour-indices should previously have been carried out with a call to **COLSRF**:

CALL COLSRF(RGB, NCOL, ALFA, IC1, IC2, NCBAND, DIFUSE, SHINE, POLISH)

which is, in many ways, an amalgam of the subroutines COLINT and COLTAB discussed earlier. If NCBAND is small (like 1) then there is good shading for the illumination but poor density discrimination, while if it's large (like IC2-IC1+1) then there's a smooth vertical colour-table but no light and darkness variation; an intermediate compromise is probably best.

3.6 A simple example

Before we give a "simple" example of the use of PGXTAL, let's outline the general structure of the calling sequence for the plotting subroutines.

```
CALL PGBEGIN
CALL PGENV
CALL SBFINT
CALL COLINT  and/or  COLTAB  and/or  COLSRF
•
CALL SBBALL  and/or  SBPLAN  and/or  SBSURF  and/or  whatever as appropriate
•
•
CALL SBFCLS
CALL PGEND
```

This can be stated in words as: (i) open the graphics device; (ii) define the viewing coordinates, by setting the size of the "window on the world"; (iii) initialise the canvas, or software buffer; (iv) initialise the colour palettes, or colour indices; (v) place all the objects in scene that are to be drawn; (vi) reveal the completed painting, or see the 3-D rendering, by closing the software buffer; (vii) close the graphics device.

A specific example of a FORTRAN program, which plots a shiny red ball with four tetradhedrally arranged green rods sticking out of it, is listed below:

```
      REAL EYE(3),LIGHT(3)
      REAL RGBBKG(3),RGBBAL(3),RGBROD(3)
      REAL CENTRE(3),VERT(3,4),VERT0(3,4)
C
      DATA EYE      /0.0,0.0,100.0/
      DATA LIGHT    /-1.0,-1.0,-0.5/
      DATA RGBBKG   /0.25,0.25,0.25/
      DATA RGBBAL   /1.00,0.00,0.00/
      DATA RGBROD   /0.00,1.00,0.00/
      DATA CENTRE   /0.0,0.0,0.0/
      DATA VERT0    /+0.5,-0.5,-0.5,
     *               -0.5,+0.5,-0.5,
     *               -0.5,-0.5,+0.5,
     *               +0.5,+0.5,+0.5/
      DATA ZSHIFT   /-0.8/
      DATA PI       /3.141592654/
C
      CENTRE(3)=CENTRE(3)+ZSHIFT
      ROT=-15.0*PI/180.0
      CALL ROTYSZ(VERT0,ROT,ZSHIFT,VERT,4)
      XYRANG=EYE(3)*ABS(ZSHIFT)/SQRT(EYE(3)*(EYE(3)+2.0*ABS(ZSHIFT)))
      CALL PGBEGIN(0,'?',1,1)
      CALL PGENV(-XYRANG,XYRANG,-XYRANG,XYRANG,1,-2)
      CALL SBFINT(RGBBKG,16,1,1,MAXBUF)
      CALL COLINT(RGBBAL,17,48,0.0,1.0,1.0)
      CALL COLINT(RGBROD,49,80,0.5,0.0,1.0)
      CALL SBBALL(EYE,CENTRE,0.3,17,48,LIGHT,.TRUE.,X0,Y0,R0)
      DO 10 I=1,4
   10   CALL SBROD(EYE,CENTRE,VERT(1,I),0.05,49,80,LIGHT,36,.TRUE.)
      CALL SBFCLS(1)
      CALL PGEND
      END
C
      SUBROUTINE ROTYSZ(VERT0,ROT,ZSHIFT,VERT,NV)
C     ------------------------------------------
C
      REAL VERT0(3,*),VERT(3,*)
C
      SINROT=SIN(ROT)
      COSROT=COS(ROT)
      DO 10 I=1,NV
        VERT(1,I)=VERT0(1,I)*COSROT+VERT0(3,I)*SINROT
        VERT(2,I)=VERT0(2,I)
        VERT(3,I)=VERT0(3,I)*COSROT-VERT0(1,I)*SINROT+ZSHIFT
   10 CONTINUE
      END
```

In order to run the above program, you will have to link it with the *object file* **pgxtal** and the PGPLOT *library*.

The arrays that will be needed are declared, and initialised, at the top: EYE, giving the coordinates of the vantage-point; LIGHT, giving the direction of the illumination; RGBBKG, for the colour of the background (dark grey); RGBBALL and RGBROD, for the colours of the ball (red) and the rods (green); CENTRE and VERT for the location of the ball and the ends of the rods. In fact the last two arrays are not correct at the beginning, but are made so at the start of the program: the z-coordinate of CENTRE is displaced backwards from the origin, by an amount ZSHIFT, so that the ball (of radius 0.3) becomes visible; and VERT is initialised by rotating (by 15°) and z-displacing (by ZSHIFT) the locations of the ends of the four rods given in VERT0, which radiate tetrahedrally from the origin (or the centre of the ball), in the subroutine ROTYSZ.

The only other preliminary calculation that needs to be done is the evaluation of the size of the "window" through which we will be looking; this is chosen, in a slightly complicated way, to ensure that the object will always be in view but never too small.

The sequence in which the plotting subroutines themselves are called is just as outlined earlier: (i) a graphics device is opened with PGBEGIN; (ii) the size of the viewing window, and thence the reference coordinate system, is set up with PGENV; (iii) the 3-D software buffer 1 is initialised with the background colour, assigned to colour index 16, with SBFINT ; (iv) 32 shades of red for the ball are assigned to colour indices 17 to 48, and 32 shades of green for the rods are assigned to colour indices 49 to 80, with COLINT ; (v) the ball is placed in the scene with SBBALL, and the tetrahedral rods are put in place with four calls to SBROD ; (vi) the 3-D picture (in software buffer 1) is finally rendered with SBFCLS ; (vii) the graphics device closed with PGEND.

3.7 Playing a movie

The dependence of the subroutines SBFINT and SBFCLS on a parameter IBUF, that flags which one of a possible MAXBUF software buffers (or canvases) is being used, allows for the possibility of playing a "movie". Thus, for example, a little modification of the preceding program can make the tetrahedral object appear to rotate:

```
      CENTRE(3)=CENTRE(3)+ZSHIFT
      ROT=-15.0*PI/180.0
      CALL ROTYSZ(VERT0,ROT,ZSHIFT,VERT,4)
      XYRANG=EYE(3)*ABS(ZSHIFT)/SQRT(EYE(3)*(EYE(3)+2.0*ABS(ZSHIFT)))
      CALL PGBEGIN(0,'/XW',1,1)
      CALL PGPAPER(2.5,1.0)
      CALL PGPAGE
      CALL PGVPORT(0.0,1.0,0.0,1.0)
      CALL PGWINDOW(-XYRANG,XYRANG,-XYRANG,XYRANG)
      CALL PGBOX('BC',0.0,0,'BC',0.0,0)
      CALL SBFINT(RGBBKG,16,1,1,MAXBUF)
      CALL COLINT(RGBBAL,17,48,0.0,1.0,1.0)
      CALL COLINT(RGBROD,49,80,0.5,0.0,1.0)
      CALL SBBALL(EYE,CENTRE,0.3,17,48,LIGHT,.TRUE.,X0,Y0,R0)
      DO 10 I=1,4
10      CALL SBROD(EYE,CENTRE,VERT(1,I),0.05,49,80,LIGHT,36,.TRUE.)
      CALL SBFCLS(1)

      DROT=2.0*PI/FLOAT(MAXBUF)
      DO 30 IBUF=2,MAXBUF
        ROT=ROT+DROT
        CALL ROTYSZ(VERT0,ROT,ZSHIFT,VERT,4)
        CALL SBFINT(RGBBKG,16,1,IBUF,MAXBUF)
        CALL SBBALL(EYE,CENTRE,0.3,17,48,LIGHT,.TRUE.,X0,Y0,R0)
        DO 20 I=1,4
20        CALL SBROD(EYE,CENTRE,VERT(1,I),0.05,49,80,LIGHT,36,.TRUE.)
        CALL SBFCLS(IBUF)
30    CONTINUE

      DO 50 IROT=1,5
        DO 40 IBUF=1,MAXBUF
40        CALL SBFCLS(IBUF)
50    CONTINUE
      CALL PGEND
      END
```

The declarations and initialisations of the arrays, and the subroutine ROTYSZ, have been omitted here as they are the same as in the example of section 3.6. Indeed, the first part of the program is essentially identical, apart from the replacement of PGENV with its constituent PGPLOT routines. To be more specific, PGBEGIN, PGPAPER and PGVPORT are used to open an X-Window, of size 2.5" square, in a manner that utilises its entirety for the plotting. The first software buffer is then filled with the previous scene, and displayed.

Rather than closing the graphics device at this point, however, the remaining MAXBUF software buffers are then filled in turn with (uniformly) rotated views of the tetrahedral object, and displayed, in the middle part of the program. In the final step, once a complete set of projections covering 2π radians has been calculated, the painted canvases are displayed on the screen sequentially (five times) to yield a "real-time movie" of the spinning object.

4. The PGXTAL library — subroutine specifications

The subroutines in PGXTAL fall into four broad categories; a brief summary is given below:

(1) *Initialising, saving and rendering the software buffer*

SBFINT — Initialises a software buffer for drawing
SBFBKG — Sets a shaded background
SBFSAV — Save an incomplete picture-buffer for subsequent reuse
SBFCLS — Renders the picture in a software buffer to the chosen graphics device

(2) *Initialising the colour palettes*

COLINT — Initialises colour indices for geometrical objects and 3-D iso-surfaces
COLTAB — Initialises colour indices for a 2-D slice through a 3-D data array
COLSRF — Initialises colour indices for 3-D surface rendering of a 2-D data array

(3) *Drawing geometrical objects*

SBBALL — Plots a sphere
SBTBAL — Plots a semi-transparent sphere
SBPLAN — Plots a (convex) planar polygon
SBPLNT — Plots a semi-transparent (convex) planar polygon
SBROD — Plots a cylinder, with a polygon cross-section
SBCONE — Plots a cone, with a polygon base
SBELIP — Plots an ellipsoid
SBLINE — Draws a (thin) line
SBTEXT — Writes a text string with perspective (variable fonts, but thin lines)

(4) *Rendering 2 and 3-dimensional data*

SBSURF — Plots an iso-surface through a 3-D "unit cell" array of density
SBTSUR — Plots a semi-transparent iso-surface through a 3-D array of density
SBSLIC — Plots a density-coloured slice through a 3-D unit cell array of data
SBCPLN — Plots a light-shaded semi-transparent slice through a 3-D unit cell lattice
SB2SRF — Plots a 3-D surface for a 2-D (unit cell) array of data

Before listing the formal calling specifications for the subroutines in PGXTAL, let's reiterate that the code is believed to adhere to the 77 standard apart from a system-specific random number generator **RAN**. If your compiler doesn't support this, you'll have to provide a FUNCTION RAN(ISEED) which generates such a *real* number between 0.0 and 1.0; it should have a uniform distribution, and be initialised with a value of ISEED that is a large odd *integer*. In order to use the PGXTAL subroutines, your program needs to be linked with the object file **pgxtal** and the PGPLOT library. Thus, for example, the link command on a VMS machine at the ISIS facility would take the form:

```
link program, ..., pgxtal, pgplot/opt
```

where as the (digital) UNIX equivalent would be:

```
f77 program.f ... pgxtal.o -lpgplot -lX11
```

We should also state that if you have to compile the FORTRAN file pgxtal.f, you will need to have the PGPLOT include file **grpckg1.inc** in the directory; make sure that it is consistent with the version of the PGPLOT library to which you are linking (it's slightly different between 5.0 and 5.1)!

4.1.1 SBFINT — Initialises a software buffer for drawing

```
      SUBROUTINE SBFINT(RGB,IC,IBMODE,IBUF,MAXBUF)
C     ------------------------------------------
C
      REAL RGB(*)
C
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C
C Purpose
C      Initialises the software buffer for crystal-plotting. It should
C    be called just once per plot (buffer), after PGWINDOW but before
C    any crystal-related routines.
C
C Parameters
C   ARGUMENT  TYPE  I/O  DIMENSION  DESCRIPTION
C    RGB      R*4    I       3      The RGB values for the background.
C    IC       I*4    I       -      The index for the background colour.
C    IBMODE   I*4    I       -      Buffering mode for initialisation:
C                                     1 = Ordinary, default.
C                                     2 = Will want to save later.
C                                     3 = Initialise from saved buffers.
C    IBUF     I*4    I       -      Software buffer to be used (>=1).
C    MAXBUF   I*4    O       -      Maximum number of buffers available.
C
C Globals
C    grpckg1.inc
C
C History
C   D. S. Sivia       4 Apr 1995  Initial release.
C   D. S. Sivia      15 Nov 1995  Allow initialisation to/from saved buffers.
C   D. S. Sivia       2 Aug 1996  Replaced pgplot.inc with SBFIN0
C-----------------------------------------------------------------------
```

4.1.2 SBFBKG — Sets a shaded background

```
      SUBROUTINE SBFBKG(IC1,IC2,ISHADE)
C     ---------------------------------
C
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C
C Purpose
C      Sets the shading for the background. This routine should be
C    called after SBFINT, and COLINT or COLTAB, but before any objects
C    are plotted.
C
C Parameters
C   ARGUMENT  TYPE  I/O  DIMENSION  DESCRIPTION
C    IC1,IC2  I*4    I       -      Lowest & highest colour-index to be
C                                   used for the shading.
C    ISHADE   I*4    I       -      Order of shading (IC1-->IC2 - IC1):
C                                      1 - Bottom to top.
C                                      2 - Left to right.
C                                      3 - Bottom-left to top-right.
C                                      4 - Top-left to bottom-right.
C                                      5 - Bottom, middle and top.
C                                      6 - Left, middle and right.
C                                      7 - Rectangular zoom to centre.
C                                      8 - Elliptical zoom to centre.
C
C History
C   D. S. Sivia      12 Oct 1995  Initial release.
C-----------------------------------------------------------------------
```

4.1.3 SBFSAV — Save an incomplete picture-buffer for subsequent reuse

```
      SUBROUTINE SBFSAV(IBUF)
C     -----------------------
C
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C
C Purpose
C      Save a rendered picture-buffer, and its Z-buffer, for subsequent
C    use in re-initialisation with SBFINT.
C
C Parameters
C   ARGUMENT  TYPE  I/O  DIMENSION  DESCRIPTION
C    IBUF     I*4    I       -      Software buffer to be saved (>=1).
C
C Globals
C    grpckg1.inc
C
C History
C   D. S. Sivia      15 Nov 1995  Initial release.
C-----------------------------------------------------------------------
```

4.1.4 SBFCLS — Renders the picture in a software buffer to the chosen graphics device

```
      SUBROUTINE SBFCLS(IBUF)
C     ----------------------
C
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C
C Purpose
C      Closes the software buffer for crystal-plotting, by outputting it
C    to the screen or writing out a postscript file (as appropriate).
C
C Parameters
C   ARGUMENT  TYPE  I/O  DIMENSION  DESCRIPTION
C    IBUF     I*4    I       -      Software buffer to be output (>=1).
C
C Globals
C    grpckg1.inc
C
C History
C   D. S. Sivia       4 Apr 1995  Initial release.
C---------------------------------------------------------------------
```

4.2.1 COLINT — Initialises colour indices for geometrical objects and 3-D iso-surfaces

```
      SUBROUTINE COLINT(RGB,IC1,IC2,DIFUSE,SHINE,POLISH)
C     -------------------------------------------------
C
      REAL RGB(*)
C
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C
C Purpose
C      Initialises a colour table for a geometrical object. In general,
C    it is recommended that SHINE = 0.0 if DIFUSE > 0.0 and vice versa.
C
C Parameters
C   ARGUMENT  TYPE  I/O  DIMENSION  DESCRIPTION
C    RGB      R*4    I       3      Red, green and blue intensity for
C                                   fully-lit non-shiny object (0-1).
C    IC1,IC2  I*4    I       -      Lowest & highest colour-index to be
C                                   used for shading.
C    DIFUSE   R*4    I       -      Diffusiveness of object (0-1).
C    SHINE    R*4    I       -      Whiteness of bright spot (0-1).
C    POLISH   R*4    I       -      Controls size of bright spot.
C
C History
C   D. S. Sivia       4 Apr 1995  Initial release.
C---------------------------------------------------------------------
```

4.2.2 COLTAB — Initialises colour indices for a 2-D slice through a 3-D data array

```
      SUBROUTINE COLTAB(RGB,NCOL,ALFA,IC1,IC2)
C     ----------------------------------------
C
      REAL RGB(3,*)
C
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C
C Purpose
C      Initialises a colour table for a "grey-scale" map.
C
C Parameters
C   ARGUMENT  TYPE  I/O  DIMENSION  DESCRIPTION
C    RGB      R*4    I   3 X NCOL   Red, green and blue intensity for
C                                    the colour table.
C    NCOL     I*4    I       -      No. of colours in the input table.
C    ALFA     R*4    I       -      Contrast-factor (linear=1).
C    IC1,IC2  I*4    I       -      Lowest & highest colour-index to be
C                                    used for the output.
C History
C   D. S. Sivia      30 Apr 1995  Initial release.
C-----------------------------------------------------------------------
```

4.2.3 COLSRF — Initialises colour indices for 3-D surface rendering of a 2-D data array

```
      SUBROUTINE COLSRF(RGB,NCOL,ALFA,IC1,IC2,NCBAND,DIFUSE,SHINE,
     *                  POLISH)
C     ------------------------------------------------------------
C
      REAL RGB(3,*)
C
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C
C Purpose
C      Initialises a colour table for a 3-D surface rendering of a 2-D
C   array of "data".
C
C Parameters
C   ARGUMENT  TYPE  I/O  DIMENSION  DESCRIPTION
C    RGB      R*4    I   3 X NCOL   Red, green and blue intensity for
C                                    the colour table.
C    NCOL     I*4    I       -      No. of colours in the input table.
C    ALFA     R*4    I       -      Contrast-factor (linear=1).
C    IC1,IC2  I*4    I       -      Lowest and highest colour-index to
C                                    be used for the rendering.
C    NCBAND   I*4    I       -      Number of colour-bands for the
C                                    height, so that the number of shades
C                                    per band = (IC2-IC1+1)/NCBAND.
C    DIFUSE   R*4    I       -      Diffusiveness of object (0-1).
C    SHINE    R*4    I       -      Whiteness of bright spot (0-1).
C    POLISH   R*4    I       -      Controls size of bright spot.
C
C History
C   D. S. Sivia      30 Oct 1995  Initial release.
C-----------------------------------------------------------------------
```

4.3.1 SBBALL — Plots a sphere

```
      SUBROUTINE SBBALL(EYE,CENTRE,RADIUS,IC1,IC2,LIGHT,LSHINE,X0,Y0,R0)
C     ------------------------------------------------------------------
C
      REAL    EYE(*),CENTRE(*),LIGHT(*)
      LOGICAL LSHINE
C
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C
C Purpose
C      This subroutine plots a shiny or matt coloured ball. All
C    (x,y,z) values are taken to be given in world coordinates. The
C    z-component of the eye-position should be positive and that of
C    the ball-centre should be negative (< -radius); the viewing-screen
C    is fixed at z=0.
C
C Parameters
C   ARGUMENT  TYPE  I/O  DIMENSION  DESCRIPTION
C    EYE      R*4    I       3      (x,y,z) coordinate of eye-position.
C    CENTRE   R*4    I       3      (x,y,z) coordinate of ball-centre.
C    RADIUS   R*4    I       -      Radius of ball.
C    IC1,IC2  I*4    I       -      Lowest & highest colour-index to be
C                                   used for shading.
C    LIGHT    R*4    I       3      (x,y,z) direction of flood-light.
C    LSHINE   L*1    I       -      Shiny ball if .TRUE., else diffuse.
C    X0,Y0    R*4    O       -      Centre of projected ball.
C    R0       R*4    O       -      Average radius of projected ball.
C
C History
C   D. S. Sivia       7 Apr 1995  Initial release.
C-----------------------------------------------------------------------
```

4.3.2 SBTBAL — Plots a semi-transparent sphere

```
      SUBROUTINE SBTBAL(EYE,CENTRE,RADIUS,IC1,IC2,LIGHT,LSHINE,X0,Y0,R0,
     •                  ITRANS)
C     ------------------------------------------------------------------
C
      REAL    EYE(*),CENTRE(*),LIGHT(*)
      LOGICAL LSHINE
C
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C
C Purpose
C      This subroutine plots a semi-transparent shiny or matt coloured
C    ball. All (x,y,z) values are taken to be given in world coordinates.
C    The z-component of the eye-position should be positive and that of
C    the ball-centre should be negative (< -radius); the viewing-screen
C    is fixed at z=0.
C
C Parameters
C   ARGUMENT  TYPE  I/O  DIMENSION  DESCRIPTION
C    All the arguments are just as in SBBALL except for:
C    ITRANS   I*4    I       -      Level of transparency:
C                                        1 = 25%; 2 = 50%; 3 = 75%.
C History
C   D. S. Sivia       8 Jul 1996  Initial release.
C-----------------------------------------------------------------------
```

4.3.3 SBPLAN — Plots a (convex) planar polygon

```
      SUBROUTINE SBPLAN(EYE,NV,VERT,IC1,IC2,LIGHT)
C     ------------------------------------------
C
      REAL EYE(*),VERT(3,*),LIGHT(*)
C
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C
C Purpose
C      This subroutine plots a diffusively-lit coloured plane; the user
C    must ensure that all the vertices lie in a flat plane, and that
C    the bounding polygon be convex (so that the angle at any vertex
C    <= 180 degs). All (x,y,z) values are taken to be given in world
C    coordinates. The z-component of the eye-position should be
C    positive and that of the vertices should be negative; the viewing-
C    screen is fixed at z=0.
C
C Parameters
C   ARGUMENT  TYPE  I/O  DIMENSION  DESCRIPTION
C    EYE      R*4    I       3      (x,y,z) coordinate of eye-position.
C    NV       R*4    I       -      No. of vertices (>=3).
C    VERT     R*4    I     3 x NV   (x,y,z) coordinate of vertices.
C    IC1,IC2  I*4    I       -      Lowest & highest colour-index to be
C                                   used for the shading.
C    LIGHT    R*4    I       3      (x,y,z) direction of flood-light.
C
C History
C   D. S. Sivia       4 Apr 1995  Initial release.
C-----------------------------------------------------------------------
```

4.3.4 SBPLNT — Plots a semi-transparent (convex) planar polygon

```
      SUBROUTINE SBPLNT(EYE,NV,VERT,IC1,IC2,LIGHT,ITRANS)
C     -------------------------------------------------
C
      REAL EYE(*),VERT(3,*),LIGHT(*)
C
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C
C Purpose
C      This subroutine plots a diffusively-lit, semi-transparent,
C    coloured plane; the use must ensure that all the vertices lie in a
C    flat plane, and that the bounding polygon be convex (so that the
C    angle at any vertex <= 180 degs). All (x,y,z) values are taken to
C    be given in world coordinates. The z-component of the eye-position
C    should be positive and that of the vertices should be negative; the
C    viewing-screen is fixed at z=0.
C
C Parameters
C   ARGUMENT  TYPE  I/O  DIMENSION  DESCRIPTION
C    All the arguments are just as in SBPLAN except for:
C    ITRANS   I*4    I       -      Level of transparency:
C                                        1 = 25%; 2 = 50%; 3 = 75%.
C History
C   D. S. Sivia      21 Aug 1995  Initial release.
C-----------------------------------------------------------------------
```

4.3.5 SBROD — Plots a cylinder, with a polygon cross-section

```
      SUBROUTINE SBROD(EYE,END1,END2,RADIUS,IC1,IC2,LIGHT,NSIDES,LEND)
C     ------------------------------------------------------------------
C
      REAL    EYE(*),END1(*),END2(*),LIGHT(*)
      LOGICAL LEND
C
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C
C Purpose
C      This subroutine plots a diffusively-shaded coloured rod. All
C    (x,y,z) values are taken to be given in world coordinates. The
C    z-component of the eye-position should be positive and that of
C    the rod-ends should be negative (< -radius); the viewing-screen
C    is fixed at z=0.
C
C Parameters
C   ARGUMENT  TYPE  I/O  DIMENSION  DESCRIPTION
C    EYE      R*4    I       3      (x,y,z) coordinate of eye-position.
C    END1     R*4    I       3      (x,y,z) coordinate of rod-end 1.
C    END2     R*4    I       3      (x,y,z) coordinate of rod-end 2.
C    RADIUS   R*4    I       -      Radius of cylindrical rod.
C    IC1,IC2  I*4    I       -      Lowest & highest colour-index to be
C                                   used for the shading.
C    LIGHT    R*4    I       3      (x,y,z) direction of flood-light.
C    NSIDES   I*4    I       -      The order of the polygon to be used
C                                   for the cross-section of the rod.
C    LEND     L*1    I       -      If true, plot the end of the rod.
C
C History
C   D. S. Sivia       4 Apr 1995  Initial release.
C----------------------------------------------------------------------
```

4.3.6 SBCONE — Plots a cone, with a polygon base

```
      SUBROUTINE SBCONE(EYE,BASE,APEX,RADIUS,IC1,IC2,LIGHT,NSIDES)
C     ------------------------------------------------------------
C
      REAL EYE(*),BASE(*),APEX(*),LIGHT(*)
C
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C
C Purpose
C      This subroutine plots a diffusively-shaded coloured right-angular
C    cone. All (x,y,z) values are taken to be given in world coordinates.
C    The z-component of the eye-position should be positive and that of
C    the base and apex of the cone should be negative (< -radius); the
C    viewing-screen is fixed at z=0.
C
C Parameters
C   ARGUMENT  TYPE  I/O  DIMENSION  DESCRIPTION
C    EYE      R*4    I       3      (x,y,z) coordinate of eye-position.
C    BASE     R*4    I       3      (x,y,z) coordinate of the centre of
C                                   the base of the cone.
C    APEX     R*4    I       3      (x,y,z) coordinate of the apex.
C    RADIUS   R*4    I       -      Radius of the base of the cone.
C    IC1,IC2  I*4    I       -      Lowest & highest colour-index to be
C                                   used for the shading.
C    LIGHT    R*4    I       3      (x,y,z) direction of flood-light.
C    NSIDES   I*4    I       -      The order of the polygon to be used
C                                   for the cross-section of the cone.
C
C History
C   D. S. Sivia      29 Jun 1995  Initial release.
C-------------------------------------------------------------------------
```

4.3.7 SBELIP — Plots an ellipsoid

```
      SUBROUTINE SBELIP(EYE,CENTRE,PAXES,IC1,IC2,LIGHT,LSHINE,ICLINE,
     *                  ANGLIN,X0,Y0,R0)
C     ---------------------------------------------------------------
C
      REAL    EYE(*),CENTRE(*),PAXES(3,*),LIGHT(*)
      LOGICAL LSHINE
C
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C
C Purpose
C      This subroutine plots a shiny or matt coloured elliptical ball.
C    All (x,y,z) values are taken to be given in world coordinates. The
C    z-component of the eye-position should be positive and that of
C    the ball-centre should be negative (< -radius); the viewing-screen
C    is fixed at z=0.
C
C Parameters
C   ARGUMENT  TYPE  I/O  DIMENSION  DESCRIPTION
C    EYE      R*4    I       3      (x,y,z) coordinate of eye-position.
C    CENTRE   R*4    I       3      (x,y,z) coordinate of ball-centre.
C    PAXES    R*4    I     3 x 3    Principal axes of the ellipsoid.
C    IC1,IC2  I*4    I       -      Lowest & highest colour-index to be
C                                   used for shading.
C    LIGHT    R*4    I       3      (x,y,z) direction of flood-light.
C    LSHINE   L*1    I       -      Shiny ball if .TRUE., else diffuse.
C    ICLINE   I*4    I       -      If >=0, colour index for lines on
C                                   surface of ellipsoid.
C    ANGLIN   R*4    I       -      Width of lines: +/- degs.
C    X0,Y0    R*4    O       -      Centre of projected ball.
C    R0       R*4    O       -      Average radius of projected ball.
C
C History
C   D. S. Sivia       8 Sep 1995  Initial release.
C-----------------------------------------------------------------------
```

4.3.8 SBLINE — Draws a (thin) line

```
      SUBROUTINE SBLINE(EYE,END1,END2,ICOL,LDASH)
C     ------------------------------------------
C
      REAL    EYE(*),END1(*),END2(*)
      LOGICAL LDASH
C
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C
C Purpose
C      This subroutine draws a straight line between two points. All
C    (x,y,z) values are taken to be given in world coordinates. The
C    z-component of the eye-position should be positive, while that
C    of both the ends should be negative; the viewing-screen is fixed
C    at z=0.
C
C Parameters
C   ARGUMENT  TYPE  I/O  DIMENSION  DESCRIPTION
C    EYE      R*4    I       3      (x,y,z) coordinate of eye-position.
C    END1     R*4    I       3      (x,y,z) coordinate of end-1.
C    END2     R*4    I       3      (x,y,z) coordinate of end-2.
C    ICOL     I*4    I       -      Colour-index for line.
C    LDASH    L*1    I       -      Dashed line if .TRUE. (else cont.).
C
C History
C   D. S. Sivia       4 Apr 1995  Initial release.
C-----------------------------------------------------------------------
```

4.3.9 SBTEXT — Writes a text string with perspective (variable fonts, but thin lines)

```
      SUBROUTINE SBTEXT(EYE,TEXT,ICOL,PIVOT,FJUST,ORIENT,SIZE)
C     ------------------------------------------------------
C
      REAL          EYE(*),PIVOT(*),ORIENT(3,2)
      CHARACTER*(*) TEXT
C
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C
C Purpose
C      Write a text string in 3-d perspective. All (x,y,z) values are
C    taken to be given in world coordinates. The z-component of the
C    eye-position should be positive and that of the text string should
C    be negative; the viewing-screen is fixed at z=0.
C
C Parameters
C   ARGUMENT  TYPE  I/O  DIMENSION  DESCRIPTION
C    EYE      R*4    I       3      (x,y,z) coordinate of eye-position.
C    TEXT     C*1    I       *      The text string to be written.
C    ICOL     I*4    I       -      Colour index for text.
C    PIVOT    R*4    I       3      (x,y,z) coordinate of pivot point.
C    FJUST    R*4    I       -      Position of pivot along the text:
C                                   0.0=left, 0.5=centre, 1.0=right.
C    ORIENT   R*4    I     3 x 2    (x,y,z) for X-length and Y-height
C                                   directions of the text.
C    SIZE     R*4    I       -      Height of the reference symbol "A".
C
C Globals
C    grpckg1.inc
C
C History
C   D. S. Sivia      14 Sep 1995  Initial release.
C-----------------------------------------------------------------------
```

Note: the character string may have the "\" PGPLOT control sequences to access different font styles, and to generate super and subscripts.

4.4.1 SBSURF — Plots an iso-surface through a 3-D array of data $\rho = \rho$ (i,j,k), corresponding to the electron-density in a unit cell

```
      SUBROUTINE SBSURF(EYE,LATICE,DENS,N1,N2,N3,DSURF,IC1,IC2,LIGHT,
     *                  LSHINE)
C     ---------------------------------------------------------------
C
      REAL    EYE(*),LATICE(3,*),DENS(0:N1,0:N2,0:N3),LIGHT(*)
      LOGICAL LSHINE
C
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C
C Purpose
C      This subroutine plots an iso-surface through a unit-cell of
C    density. All (x,y,z) values are taken to be given in world
C    coordinates. The z-component of the eye-position should be
C    positive and that of all the lattice-vertices should be negative;
C    the viewing-screen is fixed at z=0.
C
C Parameters
C   ARGUMENT  TYPE  I/O  DIMENSION  DESCRIPTION
C    EYE      R*4    I       3      (x,y,z) coordinate of eye-position.
C    LATICE   R*4    I     3 x 4    (x,y,z) coordinates of the origin
C                                   and the a, b & C lattice-vertices.
C    DENS     R*4    I     (N1+1)   The density at regular points within
C                        x (N2+1)   the unit cell, wrapped around so
C                        x (N3+1)   that DENS(0,J,K)=DENS(N1,J,K) etc..
C    N1,N2,N3 I*4    I       -      The dimensions of the unit-cell grid.
C    DSURF    R*4    I       -      Density for the iso-surface.
C    IC1,IC2  I*4    I       -      Lowest & highest colour-index to be
C                                   used for the shading.
C    LIGHT    R*4    I       3      (x,y,z) direction of flood-light.
C    LSHINE   L*1    I       -      Shiny surface if TRUE, else diffuse.
C
C History
C   D. S. Sivia       3 May 1995  Initial release.
C   D. S. Sivia      14 Jun 1996  Completely revised algorithm!
C-----------------------------------------------------------------------
```

4.4.2 SBTSUR — Plots a semi-transparent iso-surface through a 3-D array of data $\rho = \rho(i,j,k)$, corresponding to the electron-density in a unit cell

```
      SUBROUTINE SBTSUR(EYE,LATICE,DENS,N1,N2,N3,DSURF,IC1,IC2,LIGHT,
     *                  LSHINE,ITRANS)
C     ---------------------------------------------------------------
C
      REAL    EYE(*),LATICE(3,*),DENS(0:N1,0:N2,0:N3),LIGHT(*)
      LOGICAL LSHINE
C
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C
C Purpose
C      This subroutine plots a semi-transparent iso-surface through a
C    unit-cell of density. All (x,y,z) values are taken to be given in
C    world coordinates. The z-component of the eye-position should be
C    positive and that of all the lattice-vertices should be negative;
C    the viewing-screen is fixed at z=0.
C
C Parameters
C   ARGUMENT  TYPE  I/O  DIMENSION  DESCRIPTION
C    EYE      R*4    I       3      (x,y,z) coordinate of eye-position.
C    LATICE   R*4    I     3 x 4    (x,y,z) coordinates of the origin
C                                   and the a, b & C lattice-vertices.
C    DENS     R*4    I     (N1+1)   The density at regular points within
C                        x (N2+1)   the unit cell, wrapped around so
C                        x (N3+1)   that DENS(0,J,K)=DENS(N1,J,K) etc..
C    N1,N2,N3 I*4    I       -      The dimensions of the unit-cell grid.
C    DSURF    R*4    I       -      Density for the iso-surface.
C    IC1,IC2  I*4    I       -      Lowest & highest colour-index to be
C                                   used for the shading.
C    LIGHT    R*4    I       3      (x,y,z) direction of flood-light.
C    LSHINE   L*1    I       -      Shiny surface if TRUE, else diffuse.
C    ITRANS   I*4    I       -      Level of transparency:
C                                        1 = 25%; 2 = 50%; 3 = 75%.
C
C History
C   D. S. Sivia       9 Jul 1996  Initial release.
C-----------------------------------------------------------------------
```

4.4.3 SBSLIC — Plots a density-coloured slice through a 3-D array of data $\rho = \rho(i,j,k)$, corresponding to the electron-density in a unit cell

```
      SUBROUTINE SBSLIC(EYE,LATICE,DENS,N1,N2,N3,DLOW,DHIGH,IC1,IC2,
     •                  SLNORM,APOINT,ICEDGE)
C     ------------------------------------------------------------
C
      REAL EYE(*),LATICE(3,*),DENS(0:N1,0:N2,0:N3)
      REAL SLNORM(*),APOINT(*)
C
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C
C Purpose
C      This subroutine plots a "grey-scale" slice through a unit-cell
C    of density. All (x,y,z) values are taken to be given in world
C    coordinates. The z-component of the eye-position should be
C    positive and that of all the lattice-vertices should be negative;
C    the viewing-screen is fixed at z=0.
C
C Parameters
C   ARGUMENT  TYPE  I/O  DIMENSION  DESCRIPTION
C    EYE      R*4    I       3      (x,y,z) coordinate of eye-position.
C    LATICE   R*4    I     3 x 4    (x,y,z) coordinates of the origin
C                                   and the a, b & C lattice-vertices.
C    DENS     R*4    I     (N1+1)   The density at regular points within
C                        x (N2+1)   the unit cell, wrapped around so
C                        x (N3+1)   that DENS(0,J,K)=DENS(N1,J,K) etc..
C    N1,N2,N3 I*4    I       -      The dimensions of the unit-cell grid.
C    DLOW     R*4    I       -      Density for the lowest colour-index.
C    DHIGH    R*4    I       -      Density for the highest colour-index.
C    IC1,IC2  I*4    I       -      Lowest & highest colour-index to be
C                                   used for the shading.
C    SLNORM   R*4    I       3      (x,y,z) direction of the normal to
C                                   the slice to be "grey-scaled".
C    APONIT   R*4    I       3      (x,y,z) coordinate of a point within
C                                   the slice to be "grey-scaled".
C    ICEDGE   I*4    I       -      If >=0, it's the colour-index for the
C                                   boundary of the "grey-scaled" slice.
C
C History
C   D. S. Sivia      30 Apr 1995  Initial release.
C-----------------------------------------------------------------------
```

4.4.4 SBCPLN — Plots a light-shaded semi-transparent slice through a 3-D unit cell lattice

```
      SUBROUTINE SBCPLN(EYE,LATICE,IC1,IC2,LIGHT,SLNORM,APOINT,ICEDGE,
     *                  ITRANS)
C     ------------------------------------------------------------------
C
      REAL EYE(*),LATICE(3,*),LIGHT(*)
      REAL SLNORM(*),APOINT(*)
C
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C
C Purpose
C      This subroutine plots a diffusively-lit, semi-transparent,
C    coloured plane through a unit cell. All (x,y,z) values are taken to
C    be given in world coordinates. The z-component of the eye-position
C    should be positive and that of all the lattice-vertices should be
C    negative; the viewing-screen is fixed at z=0.
C
C Parameters
C   ARGUMENT  TYPE  I/O  DIMENSION  DESCRIPTION
C    EYE      R*4    I       3      (x,y,z) coordinate of eye-position.
C    LATICE   R*4    I     3 x 4    (x,y,z) coordinates of the origin
C                                   and the a, b & C lattice-vertices.
C    IC1,IC2  I*4    I       -      Lowest & highest colour-index to be
C                                   used for the shading.
C    LIGHT    R*4    I       3      (x,y,z) direction of flood-light.
C    SLNORM   R*4    I       3      (x,y,z) direction of normal to plane.
C    APONIT   R*4    I       3      (x,y,z) coordinate of a point within
C                                   the plane.
C    ICEDGE   I*4    I       -      If >=0, it's the colour-index for
C                                   the boundary of the plane.
C    ITRANS   I*4    I       -      Level of transparency:
C                                     0 = 0%; 1 = 25%; 2 = 50%; 3 = 75%.
C
C
C History
C   D. S. Sivia      26 Sep 1995  Initial release.
C-----------------------------------------------------------------------
```

Note: this subroutine really is only for crystallographic utility!

4.4.5 SB2SRF — Plots a 3-D surface for a 2-D (unit cell) array of data $\rho = \rho$ (i,j)

```
      SUBROUTINE SB2SRF(EYE,LATICE,DENS,N1,N2,DLOW,DHIGH,DVERT,IC1,IC2,
     •                  NCBAND,LIGHT,LSHINE)
C     ------------------------------------------------------------------
C
      REAL    EYE(*),LATICE(3,*),DENS(0:N1,0:N2),LIGHT(*)
      LOGICAL LSHINE
C
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C
C Purpose
C      This subroutine plots a 3-d surface given a 2-d unit-cell
C    of density. All (x,y,z) values are taken to be given in world
C    coordinates. The z-component of the eye-position should be
C    positive and that of all the lattice-vertices should be negative;
C    the viewing-screen is fixed at z=0.
C
C Parameters
C   ARGUMENT  TYPE  I/O  DIMENSION  DESCRIPTION
C    EYE      R*4    I       3      (x,y,z) coordinate of eye-position.
C    LATICE   R*4    I     3 x 3    (x,y,z) coordinates of the origin
C                                   and the a and b lattice-vertices.
C    DENS     R*4    I     (N1+1)   The density at regular points within
C                        x (N2+1)   the unit cell, wrapped around so
C                                   that DENS(0,J)=DENS(N1,J) etc..
C    N1,N2    I*4    I       -      The dimensions of the unit-cell grid.
C    DLOW     R*4    I       -      Lowest density to be plotted.
C    DHIGH    R*4    I       -      Highest density to be plotted.
C    DVERT    R*4    I       -      "Vertical" world-coordinate length
C                                   corresponding to density-range.
C    IC1,IC2  I*4    I       -      Lowest and highest colour-index to
C                                   be used for the rendering.
C    NCBAND   I*4    I       -      Number of colour-bands for the
C                                   height, so that the number of shades
C                                   per band = (IC2-IC1+1)/NCBAND.
C    LIGHT    R*4    I       3      (x,y,z) direction of flood-light.
C    LSHINE   L*1    I       -      Shiny surface if TRUE, else diffuse.
C
C History
C   D. S. Sivia        1 Jun 1995  Initial release.
C   D. S. Sivia       26 Oct 1995  Completely revised algorithm!
C-----------------------------------------------------------------------
```

Appendix: specifications of the support subroutines

A.1 PGCELL — Plots a linearly mapped 2-D array of data

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
      SUBROUTINE PGCELL(A,IDIM,JDIM,I1,I2,J1,J2,FG,BG,TR,NCOLS,R,G,B)
C     ------------------------------------------------------------
C
      REAL A(IDIM,JDIM),TR(6)
      REAL R(0:NCOLS-1),G(0:NCOLS-1),B(0:NCOLS-1)
C
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C Purpose
C     This subroutine is designed to do the job of CELL_ARRAY in GKS;
C   that is, it shades elements of a rectangular array with the
C   appropriate colours passed down in the RGB colour-table. Essentially,
C   it is a colour version of PGGRAY.
C     The colour-index used for particular array pixel is given by:
C        Colour Index = NINT{[A(i,j)-BG/(FG-BG)]*FLOAT(NCOLS-1)} ,
C   with truncation at 0 and NCOLS-1, as necessary.
C     The transform matrix TR is used to calculate the (bottom left)
C   world coordinates of the cell which represents each array element:
C        X = TR(1) + TR(2)*I + TR(3)*J
C        Y = TR(4) + TR(5)*I + TR(6)*J  .
C
C Parameters
C   ARGUMENT  TYPE  I/O  DIMENSION  DESCRIPTION
C    A        R*4    I   IDIMxJDIM  The array to be plotted.
C    IDIM     I*4    I       -      The first dimension of array A.
C    JDIM     I*4    I       -      The second dimension of array A.
C    I1,I2    I*4    I       -      The inclusive range of the first
C                                   index (I) to be plotted.
C    J1,J2    I*4    I       -      The inclusive range of the second
C                                   index (J) to be plotted.
C    FG       R*4    I       -      The array value which is to appear
C                                   with shade 1 ("foreground").
C    BG       R*4    I       -      The array value which is to appear
C                                   with shade 0 ("background").
C    TR       R*4    I       6      Transformation matrix between array
C                                   grid and world coordinates.
C    NCOLS    I*4    I       -      Number of colours in colour-table.
C    R        R*4    I      NCOLS   Red intensity for colour-table.
C    G        R*4    I      NCOLS   Green intensity for colour-table.
C    B        R*4    I      NCOLS   Blue intensity for colour-table.
C
C Globals
C    grpckg1.inc
C
C History
C   D. S. Sivia       3 Jul 1992  Initial release.
C   D. S. Sivia       6 Feb 1995  Now uses GRGRAY approach instead of PGPOLY
C   D. S. Sivia       1 Aug 1996  Replaced pgplot.inc with DSQINF!
C-----------------------------------------------------------------------
```

A program that uses the PGCELL subroutine will need to be linked with the *object files* **pgcell** (or pgcel0), **dsqinf** and the PGPLOT *library*. If you have to compile the FORTRAN file pgcell.f (or pgcel0.f), you must have the "right" version of the PGPLOT include file grpckg1.inc in the directory!

A.2 PGCELX — Plots a non-linearly mapped 2-D array of data

```
      SUBROUTINE PGCELX(A,IDIM,JDIM,I1,I2,J1,J2,FG,BG,TRNL,NCOLS,R,G,B)
C     ---------------------------------------------------------------
C
      REAL     A(IDIM,JDIM)
      REAL     R(0:NCOLS-1),G(0:NCOLS-1),B(0:NCOLS-1)
      EXTERNAL TRNL
C
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C
C Purpose
C      This subroutine is a non-linear version of PGCELL. That is to
C   say, it paints a linearly-interpolated version of the rectangular
C   grid A(i,j), where the colour-index used for particular array pixel
C   is given by:
C         Colour Index = NINT{[A(i,j)-BG/(FG-BG)]*FLOAT(NCOLS-1)} ,
C   with truncation at 0 and NCOLS-1, as necessary, according to the
C   user-provided non-linear transform TRNL. The subroutine TRNL will
C   be called with four arguments:
C         SUBROUTINE TRNL(XW,YW,XI,YJ)
C   and should return the real (fractional) array indices XI and YJ
C   which correspond to the (real) world coordinates XW and YW. The
C   appropriate Jacobian term should be factored into the array A(i,j)
C   before it is passed down (by pre-multiplication).
C
C Parameters
C   ARGUMENT  TYPE  I/O  DIMENSION  DESCRIPTION
C    A        R*4    I   IDIMxJDIM  The array to be plotted.
C    IDIM     I*4    I       -      The first dimension of array A.
C    JDIM     I*4    I       -      The second dimension of array A.
C    I1,I2    I*4    I       -      The inclusive range of the first
C                                   index (I) to be plotted.
C    J1,J2    I*4    I       -      The inclusive range of the second
C                                   index (J) to be plotted.
C    FG       R*4    I       -      The array value which is to appear
C                                   with shade 1 ("foreground").
C    BG       R*4    I       -      The array value which is to appear
C                                   with shade 0 ("background").
C    TRNL     R*4    I       -      Subroutine which performs the inverse
C                                   of the desired non-linear transform.
C    NCOLS    I*4    I       -      Number of colours in colour-table.
C    R        R*4    I      NCOLS   Red intensity for colour-table.
C    G        R*4    I      NCOLS   Green intensity for colour-table.
C    B        R*4    I      NCOLS   Blue intensity for colour-table.
C
C Globals
C    grpckg1.inc
C
C History
C   D. S. Sivia      14 Feb 1995  Initial release.
C   D. S. Sivia       1 Aug 1996  Replaced pgplot.inc with DSQINF!
C-----------------------------------------------------------------------
```

A program that uses the PGCELX subroutine will need to be linked with the *object files* **pgcelx, dsqinf** and the PGPLOT *library*. If you have to compile the FORTRAN file **pgcelx.f**, you must have the "right" version of the PGPLOT include file **grpckg1.inc** in the directory!