JACK, A. & LEVITT, M. (1978). *Acta Cryst.* A**34**, 931–935.
JANIN, J. (1979). *Nature (London)*, **277**, 491–492.
JANIN, J., MILLER, S. & CHOTHIA, C. (1988). *J. Mol. Biol.* **204**, 155–164.
KARPLUS, M. & MCCAMMON, J. A. (1983). *Ann. Rev. Biochem.* **53**, 263–300.
LEE, B. & RICHARDS, F. M. (1971). *J. Mol. Biol.* **55**, 379–400.
LESSER, G. J. & ROSE, G. D. (1990). *Proteins: Struct. Funct. Genet.* **8**, 6–13.
LEVITT, M. (1982). *Ann. Rev. Biophys. Bioeng.* **11**, 251–271.
MILLER, S., LESK, A. M., JANIN, J. & CHOTHIA, C. (1987). *Nature (London)*, **328**, 834–836.
NÉMETHY, G., POTTLE, M. S. & SCHERAGA, H. A. (1983). *J. Phys. Chem.* **87**, 1883–1887.
OOI, T., OOBATAKE, M., NÉMETHY, G. & SCHERAGA, H. A. (1987). *Proc. Natl Acad. Sci. USA*, **84**, 3086–3090.
RICHARDS, F. M. (1977). *Ann. Rev. Biophys. Bioeng.* **6**, 151–176.
RICHMOND, T. J. (1984). *J. Mol. Biol.* **178**, 63–89.
RICHMOND, T. J. & RICHARDS, F. M. (1978). *J. Mol. Biol.* **119**, 537–555.
ROSE, G. D., GESELOWITZ, A. R., LESSER, G. J., LEE, R. H. & ZEHFUS, M. H. (1985). *Science*, **229**, 834–838.
ROSE, G. D. & LEE, R. H. (1986). *J. Biophys.* **49**, 83–85.
SHRAKE, A. & RUPLEY, J. A. (1973). *J. Mol. Biol.* **79**, 351–371.
SILLA, E., VILLAR, F., NILSSON, O., PASCUAL-AHNIR, J. L. & TAPIA, O. (1990). *J. Mol. Graph.* **8**, 168–172.
SIPPL, M. J. (1990). *J. Mol. Biol.* **213**, 859–883.
VINCENT, M. G. & PRIESTLE, J. P. (1985). *J. Appl. Cryst.* **18**, 185–188.
WARSHEL, A. & CREIGHTON, S. (1989). In *Computer Simulation of Biomolecular Systems*, edited by W. F. VAN GUNSTEREN & P. K. WEINER, pp. 120–138. Leiden: ESCOM Science Publishers B.V.
WEINER, S. J., KOLLMAN, P. A., NGUYEN, D. T. & CASE, D. A. (1986). *J. Comput. Chem.* **7**, 230–252.
WLODAWER, A., MACHMAN, J., GILLILAND, G. L., GALLAGHER, W. & WOODWARD, C. (1987). *J. Mol. Biol.* **198**, 469–480.
WODAK, S. J. & JANIN, J. (1980). *Proc. Natl Acad. Sci. USA*, **77**, 1736–1740.
WOLFENDEN, R., ANDERSON, L., CULLIS, P. M. & SOUTHGATE, C. C. B. (1981). *Biochemistry*, **20**, 849–855.

J. Appl. Cryst. (1991). **24**, 946–950

MOLSCRIPT: a program to produce both detailed and schematic plots of protein structures.

By PER J. KRAULIS,* *Department of Molecular Biology, Uppsala University, BMC, Box 590, S-751 24 Uppsala, Sweden*

(*Received* 1 *March* 1991; *accepted* 10 *April* 1991)

Abstract

The *MOLSCRIPT* program produces plots of protein structures using several different kinds of representations. Schematic drawings, simple wire models, ball-and-stick models, CPK models and text labels can be mixed freely. The schematic drawings are shaded to improve the illusion of three dimensionality. A number of parameters affecting various aspects of the objects drawn can be changed by the user. The output from the program is in PostScript format.

Introduction

The complexity of protein structures makes it necessary to use simplified representations in order to present relevant features. The proper type of representation depends on exactly what aspect of the structure is of interest. For example, the active site or binding site of a protein may best be shown by ball-and-stick models, while the overall fold of a protein is best represented by schematic drawings made by hand (Brändén, Jörnvall, Eklund & Furugren, 1975; Holbrook, Liljas, Steindel & Rossmann, 1975; Richardson, 1981, 1985) or by a computer program (Priestle, 1988). The ability to combine different types of representations is important for a number of purposes, such as showing the location of important residues or ligands in the overall structure. This type of more complicated plot is usually best prepared with a computer program (Lesk & Hardman, 1982, 1985).

Our aim was to create a program that could draw schematic pictures of proteins similar to those popularized by Jane Richardson, as well as more detailed representations such as ball and stick. Features such as shading, gray scale and text labels were deemed necessary to produce high-quality plots suitable for publication and educational purposes. The strategy was to utilize the recent advances in printing technology, mainly laser printers and the PostScript language (Adobe Systems Inc., 1985). Another important aim was to design a powerful and yet conceptually simple user interface.

The program

The *MOLSCRIPT* program reads an input script file that specifies the coordinate file(s), the desired view obtained by transforming the atomic coordinates within a fixed coordinate system, which graphics parameters to change from the default values, and what graphical objects to create from the atomic coordinates. Fig. 1(*a*) is a plot of the C-terminal fragment of cellobiohydrolase I (Kraulis *et al.*, 1989) created by *MOLSCRIPT* from the input script file shown in Fig. 1(*b*). The input file is free format, may contain

* Current address: Department of Biochemistry, University of Cambridge, Tennis Court Road, Cambridge CB2 1QW, England.

comments, and must adhere to a well defined syntax. Fig. 2 shows different views of the B2 subunit of ribonucleotide reductase (Nordlund, Sjöberg & Eklund, 1990) that illustrate some capabilities of *MOLSCRIPT*.

Different kinds of representations are available in *MOLSCRIPT*. These include the basic wire-drawing, ball-and-stick and CPK models. Bonds in the wire-drawing and ball-and-stick models are drawn between atoms that are closer to each other than a given cut-off distance. In some cases this criterion is inadequate. For example, in the Fe center of B2 (Fig. 2*c*) it is impossible to obtain all bonds between the Fe atoms and their ligands without also getting spurious bonds between, for example, the O atoms within the carboxylate groups when using the simple distance cut-off criterion. In *MOLSCRIPT* this problem is solved by providing an alternative method to define which bonds to draw. This uses two different sets of atoms and requires that a drawn bond must connect an atom within one set to an atom within the other set. By using different cut-off distances with these two methods, any desired set of bonds can be drawn.

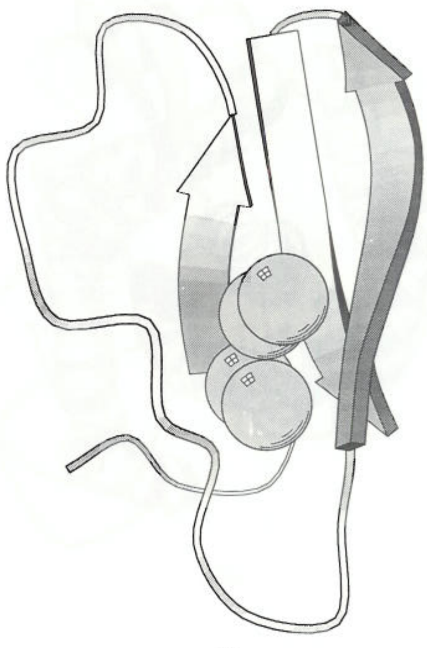
Schematic drawings of protein structures use helical ribbons for α helices, thick arrows for β strands and cylindrical coils for loops. These graphical objects are created directly from the Cα-atom coordinates of the protein (see Fig. 3 and below), not by fitting ideal secondary structure elements onto the structure. Consequently, irregularities in the secondary structure elements will be visible to some degree in the schematic drawings. In most cases this is actually a useful feature rather than a problem, since the drawing is closer to reality while still being simple. For example, in Fig. 2(*c*), the helix ribbon close to residue Glu 204 is considerably wider than normal, reflecting a real irregularity in the helix at this point.

The exact appearance of a graphical object is controlled by a number of parameters, collectively termed the graphics state. When *MOLSCRIPT* creates a graphical object, it is given dimensions, gray scale and shading according to the graphics state at that stage in the processing of the input script file. Any parameter can be changed at any point in the input script file. This allows the user considerable scope for fine tuning a plot, since different objects in the same plot can be given a different gray scale or shading. The basic wire model, for example, can be embellished by using different line widths, gray scale, dashing or depth cueing for different parts. *MOLSCRIPT* allows the use of colors defined either by the RGB (red–green–blue) or by the HSB (hue–saturation–brightness) system. Of course, the available printer or display device determines how colors are rendered.

Shading of surface segments is used to emphasise the three dimensionality of the objects used for the schematic representations (helix, strand, coil). The surface color of a graphical object is varied between black and the user-specified color as a function of the angle between the normal for each plane segment and the vector pointing towards the viewer (Foley & Van Dam, 1982). The parameters controlling the shading function are part of the graphics state, and hence can be changed by the user. During tests of the program, it was found that the shading of helix–ribbon plane segments became aesthetically more pleasing if the shading angle was corrected for the angle between the helix axis and the vector towards the viewer. In effect, the shading angle is computed as if the helix axis were in the plane of the paper (see Fig. 2).

For spheres (in CPK and ball-and-stick models) a very simple, yet effective, shading method is used. This method, which may be called Donald-Duck shading, uses three concentric arcs and an idealized reflection of a window to give the illusion of shadow and highlighting on a sphere (see Fig. 1*a*).

An important feature of *MOLSCRIPT* is the atom and residue selection mechanism by which a set of atoms or residues is given as argument to a command that produces graphical objects. This facility was inspired by a similar



(*a*)

```
! MolScript v1.1, script file

! CT-CBH1, secondary structure
! and view of cysteine sulphurs.

plot

  slab 11.5;                        ! For coil depth-cueing.

  read mol "ctcbh1.pdb";            ! Input coordinate file.

  transform atom *                  ! Transform all atoms by
    by centre position atom *       ! centering all atoms and
    by rotation z  80.0;            ! then rotate for good view.

  turn from 1 to 6;                 ! The command 'turn' gives a
  strand from 6 to 10;              ! coil that goes through the
  turn from 10 to 24;               ! CA atom positions.
  strand from 24 to 30;
  turn from 30 to 31;
  strand from 31 to 36;

  cpk atom SG;                      ! Cysteine sulphur atoms.

end_plot
```

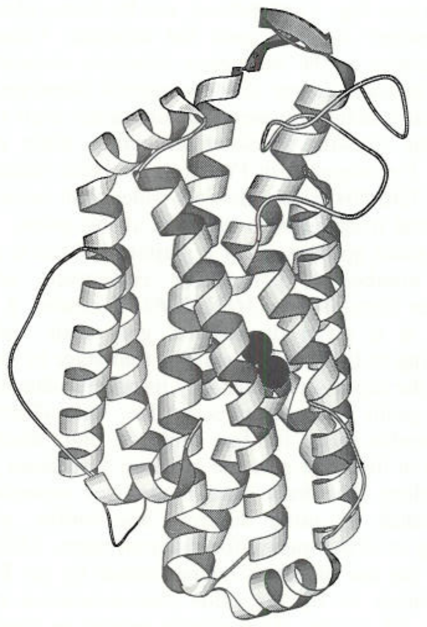
(*b*)

Fig. 1. (*a*) Plot of the C-terminal fragment of cellobiohydrolase I, CT-CBH1 (Kraulis *et al.*, 1989), using CPK representation for the S atoms of the four cysteines in this 36-residue peptide. (*b*) Input script file used to produce the picture in (*a*). No modification by hand was made to any of the printed plots shown in Figs. 1 or 2.

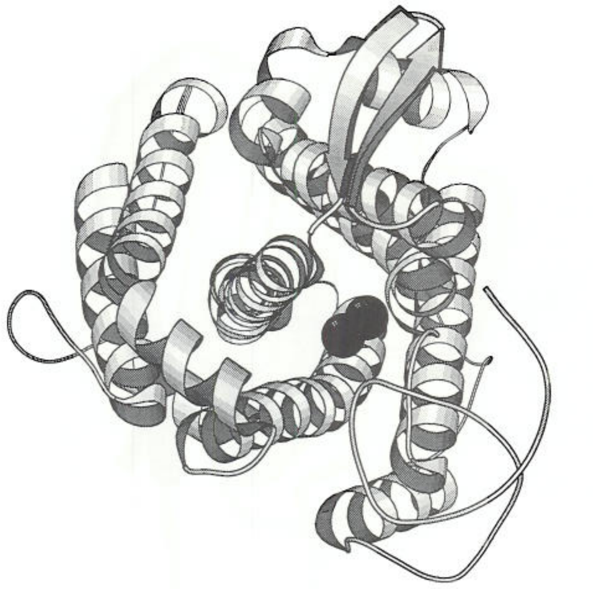
feature in the refinement program *X-PLOR* (Brünger, 1988). It is, for example, possible to make CPK or ball-and-stick models of any set of atoms, not just of predefined subsets. This allows *MOLSCRIPT* to be of use also for making plots of non-protein structures.

Text labels can be positioned either at explicit coordinates or at the positions of selected atoms. Labels can be manipulated in ways that allow output of Greek characters as part of a label (*e.g.* in atom names) and that allow the residue name or type of atoms to be inserted into the label for each specified atom. The size of label characters is depth cued, so that a label is smaller at larger distances from the viewer. The positions of labels can be fine tuned to avoid overlaps with other parts of the plot.

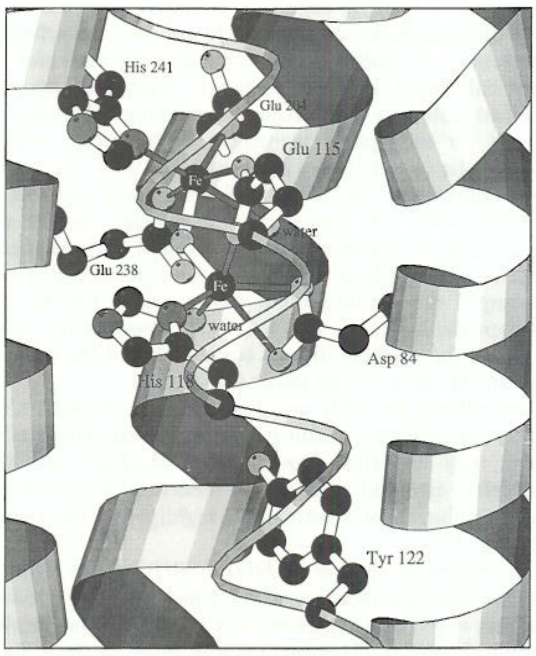
The graphical objects created within *MOLSCRIPT* can, in principle, be used as input for other rendering programs. In particular, there is an option in *MOLSCRIPT* to create files suitable for the *RASTER3D* program (written by David Bacon), which renders ray-traced pictures composed of spheres and triangular plane segments.



(*a*)



(*b*)



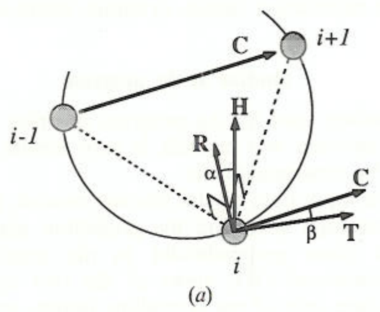
(*c*)

Fig. 2. (*a*) Plot of the B2 subunit of ribonucleotide reductase (Nordlund *et al.*, 1990) showing the shading of helices with the two Fe atoms as black spheres. (*b*) Same as (*a*), but rotated 90° around the *x* axis for a view into the α barrel, showing the location of the Fe atoms. Combined 90° views of a structure, such as (*a*) and (*b*), are often more effective than stereo pictures for conveying spatial relationships in a structure. (*c*) Close-up view of the Fe atoms in B2 and residues interacting with them. The residues are shown in ball-and-stick representation. The helix in front is shown as a coil to avoid obscuring details. In this plot it was necessary to use both methods available in *MOLSCRIPT* for defining which bonds to draw (see the text). White bonds use the single-atom-set method, while the gray bonds use the two-atom-sets method with a larger cut-off distance.

Implementation

The basic strategy in implementing *MOLSCRIPT* was to use the facilities of the PostScript language to maximum advantage. The PostScript imaging model states that any marks (lines, surface color) applied to the display surface completely obscures any previously applied marks at that position (Adobe Systems Inc., 1985). This is used in *MOLSCRIPT* to achieve hidden-surface removal by simply depth sorting all graphical segments and then writing the farthest segments first to the PostScript file. No special hidden-surface removal algorithm has been implemented in *MOLSCRIPT* beyond this trivial depth sort. Of course, this requires that the graphical segments are small enough not to overlap in complicated ways. Also, all graphical segments have to be stored in an intermediate representation within the program until an entire plot has been created.

The helix is created by first computing a curve using the Hermite spline function (Foley & Van Dam, 1982) based on the coordinate and helix tangent vector at each Cα atom. The ribbon segments are then made by translating this curve parallel to the helix axis. The helix tangent and axis vectors at each Cα atom are constructed geometrically from vectors involving the neighboring Cα atoms (see Fig. 3*a*).



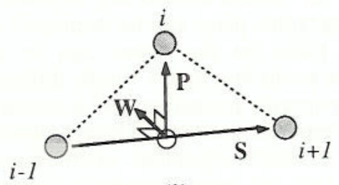


Fig. 3. (*a*) Construction of helix vectors for the atom C$\alpha(i)$. The vector **C** is the vector from atom C$\alpha(i-1)$ to atom C$\alpha(i+1)$. Vector **R** is perpendicular to the plane formed by the atoms C$\alpha(i-1)$, C$\alpha(i)$ and C$\alpha(i+1)$. Both **C** and **R** are normalized. The helix axis vector is then computed as $\mathbf{H} = (\cos\alpha)\mathbf{R} + (\sin\alpha)\mathbf{C}$ and the helix tangent vector as $\mathbf{T} = (\cos\beta)\mathbf{C} - (\sin\beta)\mathbf{R}$. The constants $\alpha = 32°$ and $\beta = 11°$ were determined empirically and are optimized for a regular α helix. Note that **C** and **R** are perpendicular but **H** and **T** are not. (*b*) The definition of vectors for atom C$\alpha(i)$ used to create the β-strand arrow. The vector **S** is the strand-arrow direction vector for atom C$\alpha(i)$. The arrow normal vector **P** at atom C$\alpha(i)$ is the vector from the midpoint between atoms C$\alpha(i-1)$ and C$\alpha(i+1)$ to atom C$\alpha(i)$. The vector **W** is used for creating the arrow planes and is simply perpendicular to **S** and **P**. Subsequent **P** vectors must not have an angle greater than 90° between them, otherwise they are turned around.

The β-strand arrow is made from direction and plane vectors computed as described in Fig. 3(*b*). The path of the arrow is a Hermite curve through the Cα-atom coordinates smoothed using an algorithm described by Priestle (1988). The number of smoothing iterations can be changed by the user.

The coil is also a Hermite spline curve through the Cα-atom coordinates which have been smoothed using Priestle's algorithm. The smoothing step can be bypassed if the user wishes the coil to pass through the real Cα atom positions (see Fig. 2*c*).

The *MOLSCRIPT* program can be viewed as a special computer language compiler. The 'source code' for *MOLSCRIPT* are the input script file and coordinate file, while the output PostScript file is the 'object code'. The final plot, produced by sending the PostScript file to, for example, a laser printer, can be seen as the 'executable'. The input script file parsing module in *MOLSCRIPT* is based on the principles for syntax definition and parsing as described by Wirth (1976). The basic graphics algorithms were taken from Foley & Van Dam (1982).

The program is written in Fortran77 and runs on a Silicon Graphics IRIS-4D system. It has been ported to other UNIX systems, such as the Alliant FX/40 and DECstation 3100, as well as to VAX/VMS systems with very few changes. The source code relies extensively on a number of general Fortran subroutine packages which were originally developed for other purposes (Kraulis, 1989).

The PostScript code produced by the program follows the structuring conventions as laid down by Adobe Systems Inc. (1985) and is suitable for direct output on, for example, a laser printer or for viewing on a display using a PostScript previewer program ('psview' on Silicon Graphics IRIS-4D systems).

The *MOLSCRIPT* program, with a manual and a set of example input script files, is available from the author.

The author thanks T. Alwyn Jones, Mats Kihlén, Ylva Lindqvist, Hans Eklund, Erling Wikman, Carl-Ivar Brändén and others at the Biomedical Centre, Uppsala, for support, ideas and comments. Hans Eklund provided the ribonucleotide reductase B2 coordinates. This work was supported by Nordisk Industrifond and the Swedish Natural Science Research Council (NFR).

References

ADOBE SYSTEMS INC. (1985). *PostScript Language Reference Manual*. Reading, MA: Addison-Wesley.
BRÄNDÉN, C.-I., JÖRNVALL, H., EKLUND, H. & FURUGREN, B. (1975). In *The Enzymes*, edited by P. D. BOYER, Vol. 11, pp. 103–190. New York: Academic Press.
BRÜNGER, A. T. (1988). *X-PLOR*. Version 1.5. Manual. Yale Univ., New Haven, Connecticut, USA.
FOLEY, J. D. & VAN DAM, A. (1982). *Fundamentals of Interactive Computer Graphics*. Reading, MA: Addison-Wesley.
HOLBROOK, J. J., LILJAS, A., STEINDEL, S. J. & ROSSMANN, M. G. (1975). In *The Enzymes*, edited by P. D. BOYER, Vol. 11, pp. 191–292. New York: Academic Press.
KRAULIS, P. J. (1989). *J. Magn. Reson.* **84**, 627–633.

Kraulis, P. J., Clore, G. M., Nilges, M., Jones, T. A., Pettersson, G., Knowles, J. & Gronenborn, A. M. (1989). *Biochemistry*, **28**, 7241–7257.

Lesk, A. M. & Hardman, K. D. (1982). *Science*, **216**, 539–540.

Lesk, A. M. & Hardman, K. D. (1985). *Methods Enzymol.* **115**, 381–390.

Nordlund, P., Sjöberg, B.-M. & Eklund, H. (1990). *Nature (London)*, **345**, 593–598.

Priestle, J. P. (1988). *J. Appl. Cryst.* **21**, 572–576.

Richardson, J. S. (1981). *Adv. Protein Chem.* **34**, 167–339.

Richardson, J. S. (1985). *Methods Enzymol.* **115**, 359–380.

Wirth, N. (1976). *Algorithms + Data Structures = Programs*. Englewood Cliffs, NJ: Prentice-Hall.

J. Appl. Cryst. (1991). **24**, 950–955

EPITAX: a computer-assisted graphical representation of the bicrystallography of general interfaces.

By F. Guillet* and S. Hagège,* *Ecole Nationale Supérieure de Chimie de Paris, Laboratoire de Métallurgie Structurale*, 11 *Rue P. et M. Curie*, 75231 *Paris, France*

(*Received* 17 *January* 1991; *accepted* 10 *April* 1991)

Abstract

A program has been developed to help design a geometric representation of the atomic structure of interfaces between two crystalline structures. The C-language program run on a graphics-oriented computer has proved to be a simple precise tool to use. It provides a clear picture of the atomic positions at the interface and, simultaneously, a stereographic projection and diffraction pattern for each crystal and bicrystal. Examples of multiple twinning in the sphalerite structure and of various metal–ceramic interfaces are presented.

Introduction

The atomic structure of intergranular and interphase boundaries has been extensively studied and reviewed periodically (*cf.* Bourret, 1990, for a most recent review). Every analysis of a theoretical or experimental result has to go through geometric modeling of the interface in which the crystal structure of each phase, the mutual orientation relationship (rotation, translation) between the two phases and the orientation and position of the boundary plane has to be fully determined and graphically represented. Rulers and protractors on tracing paper are the tools usually used for this representation. Furthermore, in non-cubic crystals, the irrationality of the lattice parameters introduces a supplementary imprecision in the drawing.

Traditionally, one has to represent each crystal by its projection on a lattice plane, in order to have an 'end-on' view of the boundary. Firstly, the projection often requires several equivalent planes in order to have a full representation of the structure. Secondly, the two projections have to be rotated and translated with great precision. Thirdly, on this dichromatic pattern (*i.e.* the superimposition of the two projections) the boundary plane has to be positioned, and, fourthly, the excess atoms removed.

The logic of this procedure has never been questioned, but its realization has always been tedious and time consuming. The purpose of the computer program is to carry out the same procedure in a precise, easy to use and efficient way. Once installed on a routine basis, it can be used on its own to provide a clear picture of the interface. The input data (rotation axis and rotation angle) can be provided, for instance, by a transmission-electron-microscopy study. Eventually, the atomic position around the interface could be used as a starting configuration for relaxation calculation (static, dynamic, electronic).

* Also at Centre National de la Recherche Scientifique, Centre d'Etudes de Chimie Métallurgique, 15, Rue Georges Urbain, 94407 Vitry-sur-Seine Cedex, France.

Outline of the program

The procedure used by the program in order to build an interface between two crystals is very similar to the one followed by experimentalists.

Once the input data crystal files are created, the first step of the procedure is to find the projection plane for each crystal (a plane perpendicular to the interface plane). Three-dimensional (3D) views of the two crystals, each showing eight cells of the crystalline lattice, are displayed. A view can be rotated around any crystallographic axis; any crystallographic plane can be displayed; the indices of the current plane on the screen can be calculated. A stereographic projection and a simple diffraction diagram (including systematic extinctions) of a current view can be obtained on separate windows. These commands allow the user to try different crystalline orientations if an exact relationship has not been completely determined experimentally.

The next step is the superimposition of the two projections. Since the number of atoms displayed in the 3D view is small (64, excluding additional atoms inside the unit cell), a larger 2D view has to be calculated. The program determines two directions generating a plane and the number of equivalent planes to display in order for the projection to be complete. This procedure makes it possible to keep track of the first atomic plane built (which can be arbitrarily placed at altitude zero) and therefore to display the 2D unit cell of this plane. The number of atoms displayed on a 2D view may vary from 100 to 2000 atoms per crystal.

Once the projections are superimposed, any of them can be rotated around any atomic position of the same crystal or translated in any direction. Since the interface plane is