Supplementary information

1. Rotation of a group of atoms about a particular bond

Let **T** and **U** be the positions of two atoms which define the desired rotation axis, and **V** be an atom which is to be rotated by an angle τ about that axis. The arbitrary rotation about **TU** can be made up by rotating **TU** to the *x* axis, rotating by τ about the *x* axis, and then rotating the *x* axis back to **TU**. It is desired to find the unitary matrix $\boldsymbol{A}$ which rotates the vector **TU** to the *x* axis. $\boldsymbol{A}^{-1} = \boldsymbol{A}^{T}$ rotates the *x* axis to **TU**, so $\boldsymbol{A}$ can be expressed as being made up of three row vectors, $\boldsymbol{A}$ = [**a, b, a×b**], where **a** is the unit vector along **TU**, and **b** is an arbitrary unit vector perpendicular to **a**. **b** is constructed by choosing any vector **c** neither perpendicular nor parallel to **a**, and noting that $\mathbf{b_0} = \mathbf{a} - \mathbf{c}/(\mathbf{a}\cdot\mathbf{c})$ is a vector perpendicular to **a**; **b** is chosen as the unit vector along $\mathbf{b_0}$. The rotated position is therefore given by

$$\mathbf{V'} = \mathbf{T} + \boldsymbol{A}^{T}\, \boldsymbol{R}_x(\tau)\, \boldsymbol{A}\, (\mathbf{V} - \mathbf{T}), \text{ where} \qquad (8)$$

$$\boldsymbol{R}_x(\tau) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\tau & \sin\tau \\ 0 & -\sin\tau & \cos\tau \end{bmatrix}$$

2. Description of the program input file

The following commands (in bold) can be given in the input file. The information entered to the program is in italic and bold letters. The examples of the lines correspond to the input file of the compound **V** (*e.g.* Dapsone.in, see below).

```
**********************************************
title DAPSONE
lebailg dapsone.rfl 120
lambda 1.5406
lattice 25.515924 8.054261 5.756876 90.00 90.00 90.00
spacegroup P 21 21 21
symm 4 (+ means no inversion, - would include inversion)
1 0 0 0 1 0 0 0 1 0 0 0
-1 0 0 0 -1 0 0 0 1 0.5 0 0.5
-1 0 0 0 1 0 0 0 -1 0 0.5 0.5
1 0 0 0 -1 0 0 0 -1 0.5 0.5 0
molecule 1 17
8  O     -0.02767     -0.93565     -2.59261 1
8  O     -0.03679      1.49471     -2.28177 1
16 S      0.00202      0.21769     -1.69970 1
6  C      2.16235     -1.09625     -0.61692 1
6  C      1.41820      0.09781     -0.68522 1
6  C     -1.36235      0.03759     -0.62203 1
6  C     -1.96735     -1.20090     -0.41563 1
6  C     -1.84049      1.23179      0.01484 1
6  C      3.21167     -1.15162      0.24467 1
6  C      1.75104      1.16148      0.15913 1
6  C     -3.07036     -1.28719      0.41137 1
6  C     -2.94286      1.250015     0.88540 1
6  C     -3.54992     -0.1250061    1.08375 1
6  C      2.79461      1.03434      1.02056 1
6  C      3.5325007   -0.12454      1.13386 1
7  N     -4.66289     -0.23222      1.93045 1
7  N      4.58568     -0.19656      2.02987 1
params 8
0 360
0 360
0 360
0 1
0 1
0 1
0 360
0 360
structure 17
getmol 1
rot_body_var z  1
rot_body_var x  2
rot_body_var z  3
rot_axis 3 5 7 10 14 15 17 9 4
rot_axis 3 6 8 8 12 13 16 11 7
putmol 0 0 0 4 5 6
endstructure
random
anneal 50 0.8 2500 0.001
pattern
fullprofpcr dapsone.pcr
gsas dapsone.ato
cif dapsone.cif
end
**********************************************
```

2.1. Line-by-line description of the input file commands

title *text*: reads a title (maximum 100 characters).
Example:
title Dapsone

lebailf *filename.hkl n:* reads the Le Bail integrated intensities (I_{obs}) from the specified file written by FULLPROF (*filename*.hkl) with output option code Hkl=1 in *filename*.pcr. The first *n* reflections in the file are used. The widths listed in the file (FWHM) are used to determine the overlap coefficients according to Eq. 5 of the manuscript.
Example:
lebailf Dapsone.hkl 120

lebailg *filename.rfl n*: It reads the Le Bail data from a GSAS *filename*.rfl file, obtained from the GSAS Utilities menu, reflist, option R (one phase ASCII reflection file). The first *n* reflections in the file are used.
Example:
lebailg Dapsone.rfl 120

lambda *wavelength*: reads the X-ray wavelength in Ångströms.
Example:
lambda 1.5406

lattice *a b c* $\alpha\beta\gamma$: reads the unit cell parameters (in Ångströms and degrees) in the order specified.
Example:
lattice 25.515924 8.054261 5.756876 90.00 90.00 90.00

spacegroup *symbol*: it reads a string with the space group symbol. This is only used to write input files for the programs FULLPROF and MERCURY.

Example:
spacegroup P 21 21 21

symm *n*: reads the number of space group symmetry operations *n* (excluding inversion and operations related by it) and *n* following lines containing the signs and translations of each symmetry operation, as described bellow. A minus (–) sign before *n* indicates a centrosymmetric space group.

The application of each symmetry operation is done in two steps. The new coordinates are calculated adding translations and multiplying by –1 or 1 to get correct signs in each component. If each atomic position is a vector **A** with 3 components, the coordinates related by the symmetry operation contained in the vector **E** are:

$$\mathbf{E} = \mathbf{T} + \boldsymbol{S}\,\mathbf{A} \qquad (9)$$

T is a vector containing the fractional unit cell translations specified in the symmetry operation to be added to the original coordinates. The elements of the line are S_{11}, S_{12}, S_{13}, S_{21}, S_{22}, S_{23}, S_{31}, S_{32}, S_{33}, T_1, T_2, T_3.

Example:
symm 4

1 0 0	0 1 0	0 0 1	0 0 0
-1 0 0	0 -1 0	0 0 1	0.5 0 0.5
-1 0 0	0 1 0	0 0 –1	0 0.5 0.5
1 0 0	0 -1 0	0 0 –1	0.5 0.5 0

which represent the 4 symmetry operations: x, y, z; -x+1/2, -y, z+1/2; -x, y+1/2, -z+1/2; x+1/2, -y+1/2,-z

molecule *n m*: Each molecule is read into the control file as atomic number, identifying label, Cartesian coordinates in Ångströms and occupancy factor, following a command line molecule *n m*, where *m* specifies the number of atoms in the n^{th} molecule.

Example:
molecule 1 17

8	O	-0.02767	-0.93565	-2.59261 1
8	O	-0.03679	1.49471	-2.28177 1
16	S	0.00202	0.21769	-1.69970 1
6	C	2.16235	-1.09625	-0.61692 1
6	C	1.41820	0.09781	-0.68522 1
6	C	-1.36235	0.03759	-0.62203 1
6	C	-1.96735	-1.20090	-0.41563 1
6	C	-1.84049	1.23179	0.01484 1
6	C	3.21167	-1.15162	0.24467 1
6	C	1.75104	1.16148	0.15913 1
6	C	-3.07036	-1.28719	0.41137 1
6	C	-2.94286	1.250015	0.88540 1
6	C	-3.54992	-0.1250061	1.08375 1
6	C	2.79461	1.03434	1.02056 1
6	C	3.5325007	-0.12454	1.13386 1
7	N	-4.66289	-0.23222	1.93045 1
7	N	4.58568	-0.19656	2.02987 1

params *n*: it specifies the number of parameters. After this line has to be specified in *n* following lines the variation range for each parameter. (In general 0 to 360 degrees for angles and 0 to 1 for molecular positions).

Example:
params 8
0 360
0 360
0 1
0 1
0 1
0 360
0 360

structure *n*: This command initiates the script used to define how the structure is defined by the adjustable parameters. *n* is the total number of atoms in the asymmetric unit defined for simulated annealing search.

Example:
structure 17

The action of each of the parameters $\{P_i\}$ defining the structure is specified by the user through a script that is included in the control file. This script is used by the program to generate the atomic coordinates for each trial set $\{P_i\}$ encountered in the simulated annealing procedure. It contains lines to enter each molecule into a buffer, subject it to the specified internal rotations followed by overall rotations of the entire molecule, and finally translate it into fractional coordinates in the crystallographic unit cell. In cases where two molecules are chemically identical, but crystallographically distinct ($Z' > 1$), the same starting (Cartesian) molecular coordinates can be read in repeatedly and distinct sets of adjustable parameters used to locate each inequivalent molecule. For salts or other systems with more than one independent fragment, each fragment is handled in succession, with independent lines (molecule *n m*) in the control script. The action of each of the commands is briefly described here.

getmol *n:* reads the *n*th molecule into the working buffer.

Example:
getmol 1

rot_body_var *q P* rotates the entire molecule in the buffer about the axis specified by *q* {x,y,z}, by an angle specified by parameter number *P*. Here, the Cartesian coordinates are defined with respect to the crystal lattice as follows: *x* is parallel to *a*, *y* is in the *ab* plane, with $y \cdot b > 0$. These axes, and the senses of rotation are the same as is used in GSAS, so that a rigid molecule or fragment can be conveniently refined in that program. However, the order of rotations is reversed (equivalently, in GSAS the reference axes are rotated along with the molecule), so that rotations defined in *PSSP* as successive lines "rot_body_var x P_1, rot_body_var z P_2, rot_body_var x P_3", would be defined in *GSAS* as "axis x, parameter number P_3, axis z, parameter number P_2, axis x, parameter number P_1".

Example:
rot_body_var x 1
rot_body_var z 2
rot_body_var x 3

rot_axis *a_1 a_2 P a_3 … a_n* specifies that atoms a_3 through a_n are to be rotated about the axis defined by atoms a_1 and a_2, by the angle specified by the parameter number *P*. A mathematical description of this algorithm is given in the first section of the supplementary information. See also figure 2 and the related manuscript text.

Example:
rot_axis 3 5 7 10 14 15 17 9 4
rot_axis 3 6 8 8 12 13 16 11 7

putmol *a_1 a_2 a_3 P_1 P_2 P_3* : a_1, a_2, a_3 are fractional coordinates to be added to the origin of the molecule *n*, which positional parameters have the numbers P_1 , P_2, P_3 .

Two ways to use this command are as follows:

The molecular origin is fixed. For example, the location of the molecules in the cell is known, the molecular origin (as defined in the copy of the molecule in Cartesian coordinates) is in 0.5 0.5 0.5, then no positional parameters are necessary. The command is putmol 0.5 0.5 0.5 0 0 0. In this way only the Eulerian and internal rotations are applied (through the commands rot_body_var and rot_axis). This may be used when molecules or fragments are located in special positions, in this case the occupancy factors (as entered in molecule *n m*) remain 1. No complete tests of this mode have been done.

Other utility of this command is when molecules are known to be around symmetry elements. For example, is known the molecules are very close to inversion centers that are coincident with the unit cell origin forming dimers through the inversion operation. Then is possible to work with the dimer instead of the single molecule, the dimer origin (as defined in the Cartesian coordinates in molecule *n m*) has to be defined so that the dimers located at 0, 0, 0 in all trial structures reproduce the inversion symmetry, which must not be included again. The command is putmol 0 0 0 0 0 0.

The molecular origin is unknown (this is the general case). The command is putmol 0 0 0 4 5 6. The last three values assigned to the parameters 4, 5 and 6 determined after structure solution runs can be used directly to insert the rigid bodies in the program GSAS for Rietveld refinements.

endstructure: indicates the end of the script of commands entering the copies of the molecules or fragments to the working buffer and applying intramolecular and external rotations and positioning in the unit cell.

Example:

endstructure

algorithm *n y*: selects the algorithm to be used through the algorithm number *n,* and the *y* parameter that governs the bias towards small values in the generation of increments in the structural parameter values in the simulated annealing cycles in algorithms 1 and 2 (step 1 of algorithm 2). If no value is specified the default values used for *n* and *y* are 2 and 1 respectively. We have used two algorithms:

Algorithm 1. In each trial solution all parameters are changed together. In this case, a random number *a* is chosen in the interval [0,1] with probability density $p(a) = a^{1/y-1} / y$ (by choosing a random number *b* uniformly on [0,1], and taking $a = b^y$). The integer parameter *y*, which governs the bias towards small values, is set in the control file. The increments in the parameter values are then given by, $\Delta_i = a\ R_i\ \delta_i$, where δ_i are chosen randomly from the interval [-½ , ½] and applied to $\{P_i\}$. Each parameter P_i has a natural range R_i, generally 0° to 360° for Eulerian angles and unrestricted rotations, and 0 to 1 for fractional coordinates, with periodic boundary conditions.

Algorithm 2. The moves alternate between the method described above, and a change (by choosing a random number uniformly on [0,1]) in a single parameter P_i, chosen at random among all defined parameters $\{P_i\}$.

Example:

algorithm 2 1

random: assigns a initial set of random numbers to the structural parameters. It is used to start a simulated annealing run.

Example:

random

enter_p n_1 n_2 …n_n: assigns a set of determined numbers n_1 n_2 …n_n to the structural parameters.

Example:

enter_p 358.2307 302.4828 29.0510 0.0963 0.5039 0.4963 329.7178 188.8820

anneal *T_{init} T_{fac} ncyc T_{final}*: simulated anneal command. T_{init} is the initial temperature, T_{fac} is the decrement factor between succesive temperatures, ncyc is the number of cycles (trial structures) postulated at each temperature and T_{final} is the final temperature.

Example:

anneal 50 0.8 2500 0.001

pattern: it writes a list with h k l, 2θ, $I_{obs.}$, $I_{calc.}$, $I_{calc.}$- $I_{obs.}$ and overlap coefficients as defined in Eq. 5 of the manuscript.

Example:

pattern

fullprofpcr *filename*: it writes a complete FULLPROF input file, however the space group symbol and other parameters must be checked. This file can be used as an input file for the visualization program ORTEP-3 for Windows.

Example:

fullprofpcr Dapsone.pcr.

gsas *filename*: it writes a text file that can be used to insert the fractional coordinates in the program GSAS. In the GSAS Expedt program, menu “edit atom parameters”, type @r and input the filename including the chosen extension.

Example:

Gsas dapsone.ato

cif *filename*: it writes a simple text file that can be used to view the crystal structure with the program MERCURY.

Example:

cif dapsone.cif

end: It indicates the end of the run.

Example:

end