Disordered Structure Refinement (DSR)

Enhanced modelling and refinement of disordered structures with SHELXL

Preface

Despite the success of X-ray crystallography, one of the remaining challenges in structure refinement is the proper description of disorder in crystal structures. The following user manual explains the usage of the program DSR. It allows for a semi-automatic modelling of disordered moieties. Its database contains molecular fragments and their corresponding restraints as well as a fitting procedure to place these fragments on the desired position in the unit cell.

The web page can be found at <https://www.xs3.uni-freiburg.de/research/dsr> and the development platform at <https://github.com/dkratzert/DSR>.

If you find any bugs in this program, have feature requests or just comments; please don’t hesitate to write an email to [dkratzert@gmx.de](mailto:dkratzert@gmx.de) to report these.

Please cite DSR as:

D. Kratzert, J.J. Holstein, I. Krossing, *J. Appl. Cryst.* **2015**. *48*, 933-938. [doi:10.1107/S1600576715005580](http://scripts.iucr.org/cgi-bin/paper?S1600576715005580)

**Disclaimer**

You are responsible for the correctness of restraints applied to your crystal structure. The restraints applied by DSR are only suggestions to stabilize a first model. DSR does not replace the judgment of an expert.

[Preface 1](#_Toc423536911)

[Program Overview 2](#_Toc423536912)

[Installation 3](#_Toc423536913)

[Windows 3](#_Toc423536914)

[Linux 3](#_Toc423536915)

[Mac OS X 3](#_Toc423536916)

[Explanations 4](#_Toc423536917)

[Command Syntax 4](#_Toc423536918)

[Example 5](#_Toc423536919)

[Command line Syntax 6](#_Toc423536920)

[General Procedure 6](#_Toc423536921)

[ShelXle Integration 6](#_Toc423536922)

[Database Format Definition 8](#_Toc423536923)

[Database Example 8](#_Toc423536924)

[Step by Step Example 9](#_Toc423536925)

[Step 0 9](#_Toc423536926)

[Step 1 10](#_Toc423536927)

[Step 2 10](#_Toc423536928)

[Step 3 11](#_Toc423536929)

[Import fragments from GRADE 12](#_Toc423536930)

[CF3-Groups 12](#_Toc423536931)

[Tips and Tricks 13](#_Toc423536932)

[Fragments Included In the Database (there might be even more already) 14](#_Toc423536933)

Program Overview

The program-package consists of a simple text-database with fragments of molecules and the DSR program itself. It acts as a preprocessor for SHELXL res-files. The user has to insert a special command line in the SHELXL .res file and the DSR program reads this information. The command lines main purpose is to tell DSR, how to orient a molecular fragment from the database in the unit cell. In practice, the user has to choose a minimum of three target positions (atoms or Q-peaks) in the structure and the corresponding atoms from the database fragment (source atoms) that should be placed on the target positions.



Installation

Windows

Execute the "DSR-setup-[version number].exe" and follow the instructions.   
DSR expects a shelxl.exe or xl.exe version 2013 or above in the system path. Windows XP users might have to install the “Microsoft Visual C++ 2010 Redistributable Package” vcredist\_x86.exe (You should not use Windows XP anyway!).

Linux

Install dsr-[version].deb or dsr-[version]-noarch.rpm according to the installation procedure of your LINUX distribution. Both packages depend on the xclip program. After a system restart DSR should work as shown in the command line syntax chapter later.  
DSR expects a shelxl or xl executable version 2013 or above in the system path.

For a manual installation create the directory /opt/DSR (or any other) and make the directory user accessible. Ex­tract the con­tent dsr\_linux-[version].tar.gz to /opt/DSR. Then copy /opt/DSR/setup/dsr.sh to /etc/profile.d. An installation to /usr/local/[DSR and bin] is of cause also possible. Therefore you have to adopt the paths in the dsr shell script. If you want to install DSR in another location, please edit the DSR\_DB\_DIR and DSR\_DIR variables in the /etc/profile.d/dsr.sh accordingly.

Mac OS X

Just open the .dmg file in the Finder and follow the instructions of the readme.txt included.

Explanations

Command Syntax

The DSR command has the following syntax:

REM DSR PUT/REPLACE fragment WITH atom1 atom2 atom3 ... ON atom2 atom3 atom4 ... PART n OCC mn RESI class num [alias] DFIX

The command is introduced with a REM because SHELXL should never interpret the DSR command line.

PUT Put the fragment on there, ignoring atoms on this position.

REPLACE Replace the target atoms. Hydrogen atoms of a replaced target atom will be removed.

fragment The name of the desired molecule or fragment.

WITH Behind WITH are the source atoms. They are at least three atoms from the fragment.

ON Behind ON are the target atoms. They are at least three atoms or Q-peaks in the .res- file.

[atom n] Minimum three atoms each (including Q-peaks). Source and target have to include the same number of atoms and/or Q-peaks. Target atoms can be either regular atoms or atoms in residues. Atoms in residues can be addressed by the “\_” notation. C1\_2 would be atom C1 in residue number 2.

PART n Optional SHELXL PART definition.

OCC mn Optional occupancy and free variable definition for the fragment.

DFIX Optional, generates DFIX/DANG restraints instead of those from the database. All 1,2- and 1,3-distances in the fragment are restrained with DFIX and DANG respectively. DSR also searches for rings in the fragment and generates FLAT restraints for flat rings.

RESI class num [alias] Optional residue definition as in SHELXL.

SPLIT Only for a disordered CF3 group on two positions (CF6 fragment). Splits the pivot atom in two positions.

Use of Residues

To use the RESI command in DSR has several advantages. It places the fragment into a residue and therefore no renaming of the atoms in the fragment needs to be performed by DSR. If residues are used, the restraints like "SADI\_class Atoms" are inserted only once, since they act on the atoms in all residues with the same class together. Residues are especially useful if the same moiety is repeated several times in a crystal structure. Different moieties of the same residue class are distinguished by different residue numbers. Residue number must be unique in a .res file. The DSR command RESI without any further options is normally the best practice. DSR then uses the residue class name from the database and finds the next free residue number by itself. But the user can also specify a particular residue class and/or number after the RESI command, if desired.

The RESI option can be used in three ways:

1. If only a RESI command is given (best practice), the residue class is taken from the database entry and the residue number is automatically generated.
2. If RESI with only a number is given, DSR takes the residue class from the database with the given number.
3. RESI with a number and a class overwrites the information from the database and gives complete control over the residue.

A given class, number or alias always overwrites the information of the database. To use the RESI command, the user should have at least basic knowledge about the concept of residues in SHELXL. The manual on the SHELX website gives more detailed information: <http://shelx.uni-ac.gwdg.de/SHELX/wikis.php>

Example

The following command line can be inserted anywhere between the atoms of a res-file.

**REM DSR put toluene with C1 C2 C3 on Q1 C5 C2**

The command is always introduced with a REM. DSR is completely case insensitive. The DSR command line can be up to two lines with a trailing "=" for a continuation line like in SHELX. Please note that the second line of the DSR COMMAND after the "=" must begin with a leading whitespace.

The minimal requirement for DSR to work is rem dsr put/replace “fragment” with “three atoms/Q-peaks” on “three atoms/Q-peaks”.

The new molecule or fragment is placed in the line where the DSR command resides. DSR applies a new naming scheme to the fragment while inserting it into the res-file. Essentially it searches if any atom name from the database fragment is already used in the res-file. If this applies, the program places a suffix letter (A, B, ...) to the atom name in the res-file. This renaming is completely turned off if residues are used. Atoms of the new fragment are then addressed by their residue.

**put** DSR searches for the coordinates of the given atoms/Q-peaks and places the fragment on these coordinates leaving the given atoms in place. The above example will place the fragment on the coordinates of Q1, C5 and C2. The atoms C5 and C2 would remain where they were located before.

**replace** DSR searches for the coordinates of the given atoms/Q-peaks but in contrast to the former example, it replaces the target atoms and all atoms in 1.2 Å distance around each atom of the fitted fragment that are in PART 0. This mode is useful to quickly rename atoms from a solution by SHELXT.

It is highly advised to use residues with DSR. They make a lot of things easier and DSR takes care about all details regarding residues. Normally it is sufficient to simply use the RESI command without any options in DSR. This way, DSR takes the residue class from the database and finds the next residue number automatically. Restraints for the same residue class are only introduced once. Also the atoms in the fragment would not be renamed:

**REM DSR put toluene with C1 C2 C3 on Q1 C5 C2 RESI**

Command line Syntax

Following options are available in the Windows or Unix command line to control the behavior of DSR:

usage: dsr [-h] [-r "res file"] [-re "res file"] [-e "fragment"] [-c "fragment"] [-t] [-i "tgz file"] [-l] [-n]

optional arguments:

–h, --help Show a help message and exit.

–r "res file" res file with DSR command. Usually this option is used to process the SHELXL file with DSR.

–re "res file" Same as "-r", but a file called dsr\_class\_name.dfx or dsr\_class\_number\_name.dfx is written which includes the restraints for the fragment for the .res file "name" in the residue "class" and "number".

–e "fragment" Exports a fragment from the database to the file [fragment].res. It includes the minimal requirements to view the fragment in a 3D molecule viewer. If a PLATON executable and ImageMagic installation is in the system path, it also creates a .png-picture of the molecule.

–c "fragment" Exports the fragment to the clipboard with Cartesian coordinates. This fragment can for example be used for modelling in the program Olex2.

–t Inverts the current fragment. Available for fragment fit, import and export.

–I "GRADE file" Imports a molecular fragment from .tgz file of the Grade server http://grade.globalphasing.org/ into the dsr\_usr\_db.txt.

–l Displays all fragments in the database with the line numbers where they occur.

–s "string" Search the database for given string.

–n Only transfers the fragment. The fragment fit after the fragment transfer is disabled.

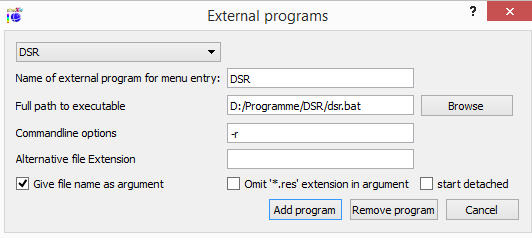
General Procedure

Edit the .res file according to the “Example” chapter. Run dsr −r filename.res. DSR will now insert the fragment, does a refinement with L.S. 0 to finally insert the fragment and removes the AFIX command. The resulting filename.res can now be reopened for further refinement. Before DSR changes anything in the .res file it creates a backup file in a subdirectory "dsrsaves" with the current date as file name.

ShelXle Integration

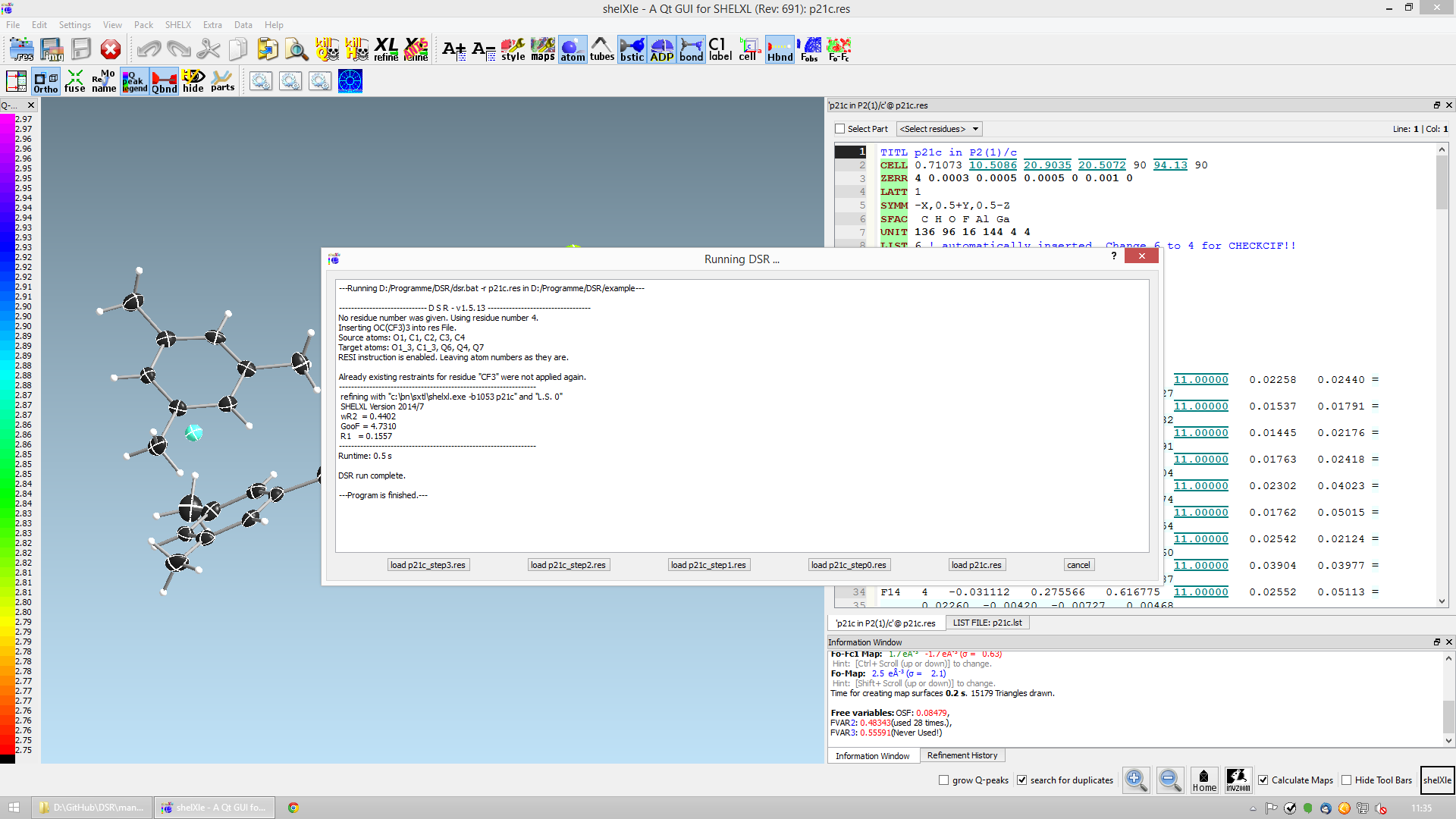
ShelXle has the ability to start external programs. You can integrate DSR as a task button in the menu “extra” -> “add external program”. Fill in the options as shown below and click “Add program”. The “Full path to executable” should be “C:\Program Files (x86)\DSR” on 64bit Windows systems and “C:\Program Files\DSR” on 32bit Windows systems. Do not use the “start detached” option. The output of DSR will hardly be visible if you enable it.





A new button now appears in the menu bar of ShelXle. Clicking on this button starts DSR and interprets the DSR command in the actual res-file. A similar button with the additional –t option before –r can also be useful if the fragment should be inverted.

Please don’t forget to save the res-file after you typed in the DSR command line. Otherwise DSR does not find the DSR command. In Addition the processed file has to be re-opened in ShelXle after the processing through DSR. Since version 681 of ShelXle, the console window has the possibility to open the processed res file directly.



Database Format Definition

The database format was deliberately kept very simple. It consists of a system database in the dsr\_db.txt and a user database in the dsr\_user\_db.txt. The system database is overwritten with every new program install while the user database will always stay untouched. So the user can easily add new fragments to its own dsr\_user\_db.txt database. The syntax mainly follows the SHELXL syntax:

**<fragment name>** <- Start tag. **RESI class** <- Required, defines the residue name of db entry. **restraints** <- Any restraints and comments following the SHELXL

syntax.You must enter at least one restraint!

e.g. RIGU C1 > C7 **FRAG 17 a b c alpha beta gamma** <- FRAG card with afix number and cell

parameters.

**Atom sfac-number coordinates** <- One isotropic atom per line following

SHELX syntax.

e.g.

**O1 1 1.2345 0.6734 0.8352** <- Either the atom type is recognized by

the atom name for positive Numbers in

the second column.

**C1 -6 0.2683 0.4783 0.1616**  <- Or the atom type is defined by the

negative atomic number in the second

column.

**</fragment name>**  <- End tag. Same as start tag but with /

- Anything not being an atom after FRAG is ignored.

- Fragment names CF3, CF6 and CF9 are reserved by DSR. Do not attempt to use them in database entries.

- Only lines beginning with valid SHELXL instructions are allowed in the header.

- Anything behind the 5th column in the atom list is ignored.

- Long lines can be wrapped with an equal sign (=) and a space character in the next line like in SHELXL, but the can also be of any length. All lines will be wrapped to fit in the SHELXL file automatically.

Database Example

A usual database entry looks like the following:

The restraints applied by DSR might be stricter than necessary. After introduction of a new fragment, the refinement can be proceeded as usual. In the course of you should review the restraints. Modifications to database fragments should always be done in the dsr\_user\_db.txt and not in the dsr\_db.txt. The user database will not be overwritten during updates. The fragment names must be unique in both databases. Every valid restraints from SHELXL can be used, even HFIX is possible.

<Toluene>

rem CCDC: BUWME

rem Name: Toluene, C7H8

RESI TOL

SAME C2 > C6 C1

SAME C1 C6 < C2 C7

HFIX 137 C7

FLAT C1 > C7

SIMU C1 > C7

RIGU C1 > C7

FRAG 17 11.430 12.082 15.500 106.613 100.313 90.68

C1 1 0.268330 0.478380 0.161680

C2 1 0.205960 0.555770 0.217990

C3 1 0.249400 0.600760 0.310040

C4 1 0.357300 0.568990 0.348900

C5 1 0.420800 0.492470 0.294060

C6 1 0.376630 0.447580 0.201340

C7 1 0.221500 0.430400 0.060360

</TOLUENE>

The syntax follows the SHELXL syntax. All entries between the start tag <Toluene> and the FRAG command are considered as the database entry header. Comments can be introduced with REM. All lines with non-SHELX commands are ignored.

If a "rem Name:" statement is given, the name after this statement is printed in the list of available fragments.

After FRAG until the end tag </TOLUENE> only atoms with SHELX syntax are accepted.

Step by Step Example

You can find the following example in the DSR install directory.

Step 0

* Open “p21c.res”.
* Residue number 3 turns out to be a part of a disorder.
* Apply a PART and the free variable 3 to this residue with “PART 1 31”:



**O1\_3**

**C1\_3**

**Q6**

**Q4**

**Q7**

RESI 3 CF3

SIMU O1\_3 > F9\_3

**PART 1 31**

O1 3 0.081800 0.235411 0.399962 11.00000 0.02269

C1 1 0.033372 0.232537 0.337248 11.00000 0.02263

C2 1 0.146121 0.230044 0.293483 11.00000 0.07563

F1 4 0.113536 0.207456 0.233219 11.00000 0.04423

F2 4 0.245556 0.200608 0.318034 11.00000 0.04443

F3 4 0.177383 0.298455 0.290893 11.00000 0.11659

C3 1 -0.047007 0.289792 0.317142 11.00000 0.08657

F4 4 -0.015523 0.342850 0.348530 11.00000 0.04231

F5 4 -0.064007 0.300531 0.253531 11.00000 0.03919

F6 4 -0.174400 0.269329 0.340816 11.00000 0.09549

C4 1 -0.042812 0.170935 0.325702 11.00000 0.11900

F7 4 -0.111534 0.156135 0.374179 11.00000 0.04207

F8 4 -0.120582 0.172784 0.270319 11.00000 0.05058

F9 4 0.058836 0.122805 0.319484 11.00000 0.09763

**PART 0**

RESI 0

Step 1

* Now you can insert the command line for DSR after the residue 3.
* The command is

**rem dsr put OC(CF3)3 with O1 C1 C2 C3 C4 on O1\_3 C1\_3 q6 q4 q7**

to place the fragment **OC(CF3)3** on the position of **O1\_3 C1\_3 q6 q4** and **q7**.

* In addition we want to have the fragment in a **PART 2** with the **occupancy** of **−31** and in a **residue**. DSR automatically finds a free residue number and uses a residue name from the database. All these options are placed in one line. Lines longer than 80 characters can be continued with "=" at the end and a whitespace before the first character in the next line, as usual in SHELXL.
* The complete command is then:

**rem dsr put OC(CF3)3 with O1 C1 C2 C3 C4 on O1\_3 C1\_3 q6 q4 q7 part 2 =**

**occ -31 resi**

* Save the res file after editing.

Step 2

* Now run "dsr -r p21c.res" on the Windows/Unix command line.
* DSR will run over the res file, insert the fragment and makes a L.S. refinement with L.S. 0. This finally inserts the fragment. You can see the status before the refinement in the "p21c‑step2.ins" file.



* Reopen the resulting res file.

Step 3

* The fragment turned out to be successfully fitted on its desired position:



* The previously used DSR command line is now commented out with REM and will not be recognized by DSR again.
* Now you can add/remove additional restraints and further refine the structure as usual. Already existing restraints for an existing residue class will not be inserted again, because they already act for all residues together.

Import fragments from GRADE

GRADE from Global Phasing Ltd. <http://grade.globalphasing.org/> is a ligand restraint generator whose main source of restraint information is the Cambridge Structural Database (CSD) of small-molecule crystal structures, queried using the MOGUL program developed by the CCDC. Where small-molecule information is lacking, Grade uses quantum chemical procedures to obtain the restraint values.

Fragments obtained by GRADE can be imported to DSR with the –i command line option. The GRADE server outputs a .tgz file including several files. Execution of “dsr –i filename.tgz” will import a GRADE fragment from these files. The fragment gets imported to the “dsr-user-db.txt” in the DSR program directory. You also might need to change the fragment and residue name after the import. The best way is to supply the GRADE server with a .mol2 file. This way you can choose the atom names and their sorting yourself. mol2 files can be generated if you create a molecule with Avogadro (save as .res file) <http://sourceforge.net/projects/avogadro/> then you must rename the atoms and open the res file with mercury <http://www.ccdc.cam.ac.uk/Solutions/CSDSystem/Pages/Mercury.aspx>. Mercury can now save a .mol2 file for GRADE.

CF3-Groups

DSR is able to generate CF3 groups with the respective restraints automatically. The CF3 group can be modeled either as ordered group on one position (like an AFIX 130 would do), as well as a disordered group on two or three positions. The respective fragment names are CF3, CF6 and CF9. A disordered CF3 group on three positions would be modelled using:

REM DSR put CF9 on C1

The result is the following plus the respective free variables added to the FVAR line.

REM CF3 group made by DSR:

SUMP 1 0.0001 1 2 1 3 1 4

SADI 0.02 C1 F1B C1 F2B C1 F3B C1 F4B C1 F5B C1 F6B C1 F7B C1 F8B C1 F9B

SADI 0.04 F1B F2B F2B F3B F3B F1B F4B F5B F5B F6B F6B F4B F7B F8B F8B F9B =

F9B F7B

SADI 0.1 C2 F1B C2 F2B C2 F3B C2 F4B C2 F5B C2 F6B C2 F7B C2 F8B C2 F9B

RIGU C2 C1 F1B > F9B

PART 1 21

F1B 3 0.128560 -0.462510 0.547150 11.00000 0.04

F2B 3 0.047696 -0.372991 0.451178 11.00000 0.04

F3B 3 0.040064 -0.244730 0.559034 11.00000 0.04

PART 2 31

F4B 3 0.118185 -0.486145 0.500565 11.00000 0.04

F5B 3 0.020565 -0.289148 0.471484 11.00000 0.04

F6B 3 0.077571 -0.304938 0.585313 11.00000 0.04

PART 3 41

F7B 3 0.086249 -0.450791 0.462663 11.00000 0.04

F8B 3 0.017551 -0.238494 0.514080 11.00000 0.04

F9B 3 0.112520 -0.390946 0.580619 11.00000 0.04

PART 0

Existing fluorine atoms connected to the respective carbon atom will be deleted by DSR beforehand.

If you like DFIX instead of SADI, add DFIX to the DSR command line:

REM DSR put CF9 on C1 DFIX

The special command SPLIT in combination with the CF6 fragment tries to split the target carbon atom in two positions. The coordinates of the two positions are the principal axes of the carbon atom ellipsoid in its longest direction. The restraints will be adjusted accordingly.



Figure 1 CF3 group split on two positions (rem dsr put CF6 on C22 split).

You should never use CF9 just because it is possible! Often CF6 is sufficient. CF9 often just uses more least-squares parameters without improving the model.

Tips and Tricks

* Some crystal structure refinement programs like ShelXle do not always show the Q-peaks at the position where you expect them to be. Especially around special positions, the Q-peaks appear at symmetry equivalent positions. If you then fit fragments to the Q-peaks, the fitting position will not be as expected. You can avoid this by renaming the Q-peaks first (e.g. to carbon atoms), place them in the correct symmetry equivalent position (Move xy here in ShelXle) and start fitting with DSR.
* You can quickly rename molecular moieties using DSR in the replace mode. This is useful directly after the solution with SHELXT for example. In the replace mode, DSR replaces all atoms that are in PART 0 and that are 1.2 Å near the atoms of the placed fragment. Using replace mode prevents you from renaming every single atom.
* You want to know typical bond lengths of fragments included in the DSR database? Export the respective fragment and either see 1,2 and 1,3 distances in the restraints lists or let the SHELX GUI of your choice find out any distance of the respective atoms.

Fragments Included In the Database (there might be even more already)



