Disordered Structure Refinement (DSR)

A program to easily refine disordered solvents with SHELXL-2013

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.

[Preface 1](#_Toc381792749)

[Program Overview 2](#_Toc381792750)

[Installation 2](#_Toc381792751)

[Windows 2](#_Toc381792752)

[Linux 2](#_Toc381792753)

[Explanations 3](#_Toc381792754)

[Command Syntax 3](#_Toc381792755)

[Example 4](#_Toc381792756)

[Command line Syntax 5](#_Toc381792757)

[General Procedure 5](#_Toc381792758)

[ShelXle Integration 5](#_Toc381792759)

[Olex2 Integration 6](#_Toc381792760)

[Database Format Definition 6](#_Toc381792761)

[Database Example 7](#_Toc381792762)

[Step by Step Example 7](#_Toc381792763)

[Step 0 7](#_Toc381792764)

[Step 1 8](#_Toc381792765)

[Step 2 9](#_Toc381792766)

[Step 3 10](#_Toc381792767)

[Import fragments from GRADE 10](#_Toc381792768)

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) in the DSR installation directory.

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 re-logon 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 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.

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.

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 and their neighboring atoms are restrained with DFIX and DANG respectively. Bonds to symmetry equivalent atoms are ignored.

RESI class num [alias]

Optional residue definition as in SHELXL.

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. In the replace-mode of the example, the atoms C5 and C2 are replaced by the atoms C2 and C3 from the fragment “Toluene”.

It is highly advised to use residues with DSR. They make a lot of things more easy 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. 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. Automatic fragment fit after the fragment transfer disabled.

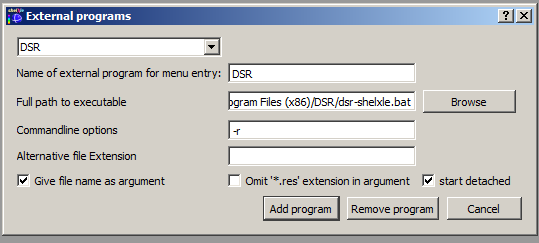
General Procedure

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 resulting 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 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.





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.

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. Linux users should not enable the “start detached” option.

Olex2 Integration

DSR copies the fragment to the clipboard during the fragment export with option "–c". It can afterwards be inserted in Olex2 via CTRL+V to use the match functionality. Fractional coordinates will be translated to Cartesian coordinates during the clipboard export.

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 of the fragment in the database.

Header with restraints <– At least one restraint must be present.

and comments.

FRAG 17 a  b  c  α  β  γ <- cell parameters of the fragment

Atom1 1 x y z <- atom coordinates of the fragment

Atom2 1 x y z

Atom3 1 x y z

...

</fragment name> <– End tag of the fragment in the database.

See more details in the dsr\_db.txt.

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

FEND

</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 command 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 sorting yourself. mol2 files can be easily generated with Avogadro

<http://sourceforge.net/projects/avogadro/>

and Mercury

http://www.ccdc.cam.ac.uk/Solutions/CSDSystem/Pages/Mercury.aspx.