Disordered Solvent Refinement (DSR)

A program to easily refine disordered solvents with SHELXL-2013

Preface

The refinement of highly disordered solvents and moieties is an often boring or even uneasy task. Therefore these voids frequently are "squeezed out" of the X-ray data. This practice is often not necessary. To help with the routine work of refining disordered solvents the program DSR was developed in python.

This program is not in its final stage but it is already usable. If you find any bugs, feature requests or just comments, please don’t hesitate to write an email to [daniel.kratzert@ac.uni-freiburg.de](mailto:daniel.kratzert@ac.uni-freiburg.de) to report these.

[Preface 1](#_Toc373928087)

[Program Overview 2](#_Toc373928088)

[Installation 2](#_Toc373928089)

[Windows 2](#_Toc373928090)

[Linux 2](#_Toc373928091)

[Explanations 3](#_Toc373928092)

[Command Syntax 3](#_Toc373928093)

[Example 4](#_Toc373928094)

[Command line Syntax 5](#_Toc373928095)

[General Procedure 5](#_Toc373928096)

[ShelXle Integration 5](#_Toc373928097)

[Olex2 Integration 6](#_Toc373928098)

[Database Format Definition 6](#_Toc373928099)

[Database Example 7](#_Toc373928100)

[Step by Step Example 7](#_Toc373928101)

[Step 0 7](#_Toc373928102)

[Step 1 8](#_Toc373928103)

[Step 2 9](#_Toc373928104)

[Step 3 9](#_Toc373928105)

[Import fragments from Grade 10](#_Toc373928106)

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 in the SHELXL res file and the DSR program reads this information to place a molecule or fragment with the desired atoms on the position of the target atoms or Q-peaks, respectively, which are specified by the user. Bond restraints are applied from the database to the fragment. Future versions will also generate additional restraints automatically.

res-file with DSR command



fragment database

new res-file with fragment and restraints

apply new naming scheme

refinement with “L.S. 0”

RESI

DSR inserts the fragment

import from GRADE

Export to .res .png and Olex2-match

output .ins file

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. Poth 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 atom1 atom2 atom3 ... part n occ mn resi class num [alias]**

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 deleted.

**fragment** The name of the desired molecule or fragment.

**[atom n]** Minimum three atoms (including Q-peaks). Source and target have to include the same number of atoms or Q-peaks. Target atoms can be either normal 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 Switch on 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. It only works in combination with RESI.

**RESI class num [alias]**

Optional residue definition as in SHELXL. Residues can be used in three ways:

1) If only a RESI command is given, 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

If residues are used, the restraints like "SADI\_class Atoms" are inserted only once, since they act on the atoms in all residues of class class together.

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”.

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]] [-e [fragment]] [-l] [-n]

optional arguments:

-h, --help show this help message and exit

-r “res file” res file with DSR command

-e “fragment” export fragment from the database

-i “GRADE file” import a fragment from GRADE (needs .dfx and .mol2 file)

-l list names of all database entries

-n do not refine after fragment transfer

Usually the **−r** option is used to process the SHELXL file.

**−e** 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.

**-i** Import a molecular fragment from the Grade server. <http://grade.globalphasing.org/>

**–l** displays all Fragment in the database with the line numbers where they occur.

**–n** disables the refinement with “L.S. 0” after the fragment transfer.

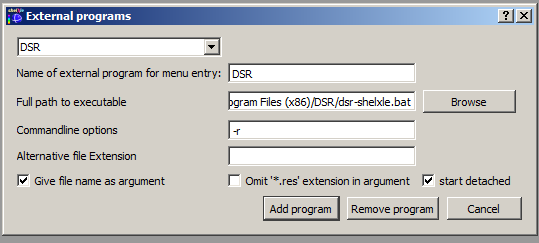
General Procedure

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.

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.

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 can be integrated into Olex2 via macros. Just copy the file custom.xld from the DSR program directory to the Olex2 program directory.

DSR can then be used as command in the Olex2 command line:

“dsr-r” executes dsr–r ‘filename’.res.

“dsr-l” lists the database fragments in the Olex2 console.

“dsr-e fragment” exports the fragment to a file and shows it in xp. Export only works in   
Windows 32bit at the moment. But you can also export outside of Olex2.

DSR copies the fragment to the clipboard during the fragment export. It can afterwards be inserted in Olex2 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.

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 every other refinement. In the course of you should review the restraints.

<Toluene>

rem CCDC: BUWME

RESI TOL

SAME C2 > C6 C1

SAME C1 C6 < C2 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 a the database entry header. Comments can be introduced with REM. All lines with non-SHELX command are ignored.

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

Step by Step Example

You can find this 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 the **residue** number **4** of **class** **CF3**. 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:

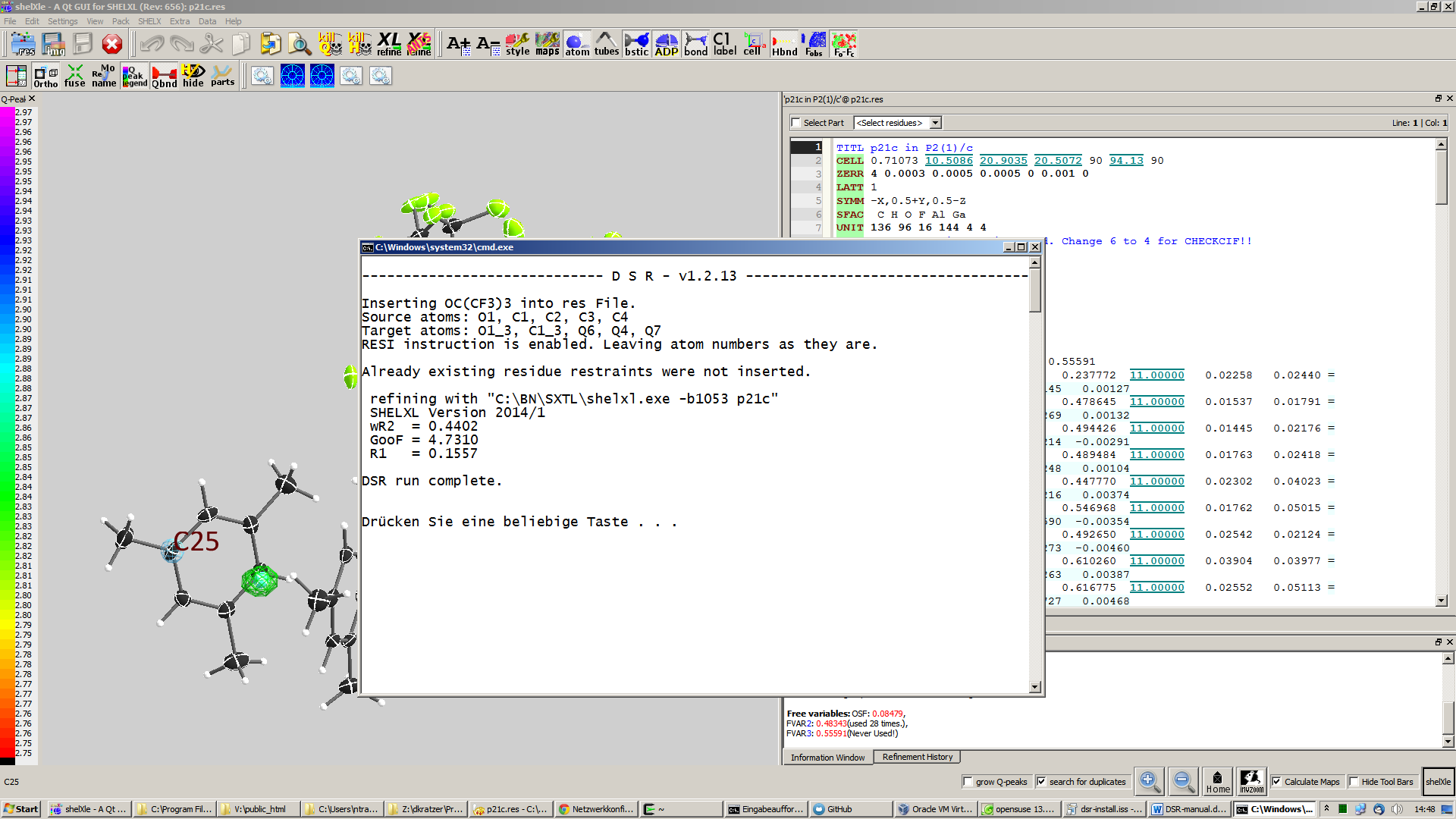
**rem dsr put OC(CF3)3 with O1 C1 C2 C3 C4 on O1\_3 C1\_3 q6 q4 q7 part 2 =**

**occ -31 resi 4 CF3**

* 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. 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. The Grade web server can be accessed here:

<http://grade.globalphasing.org/>