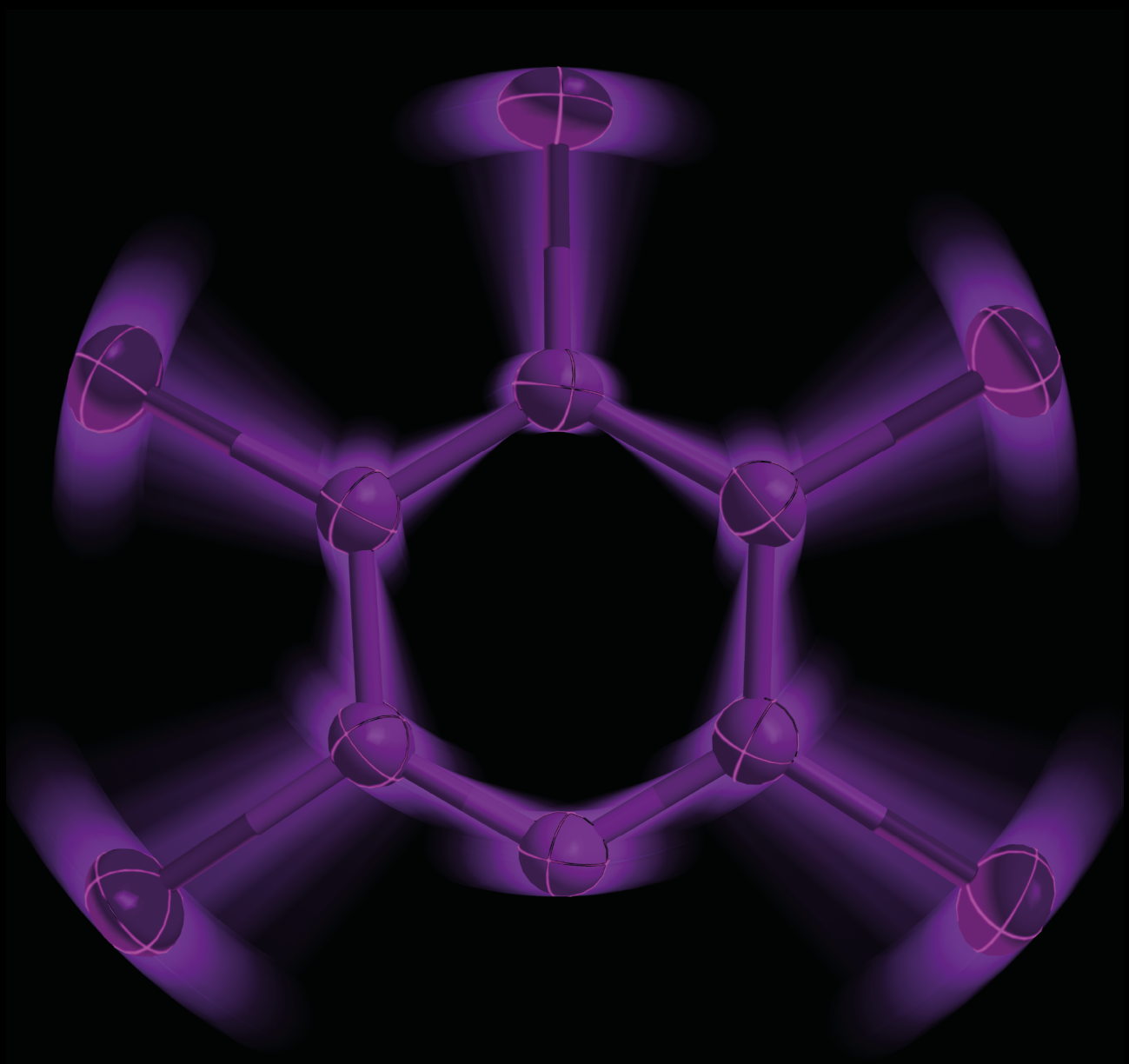
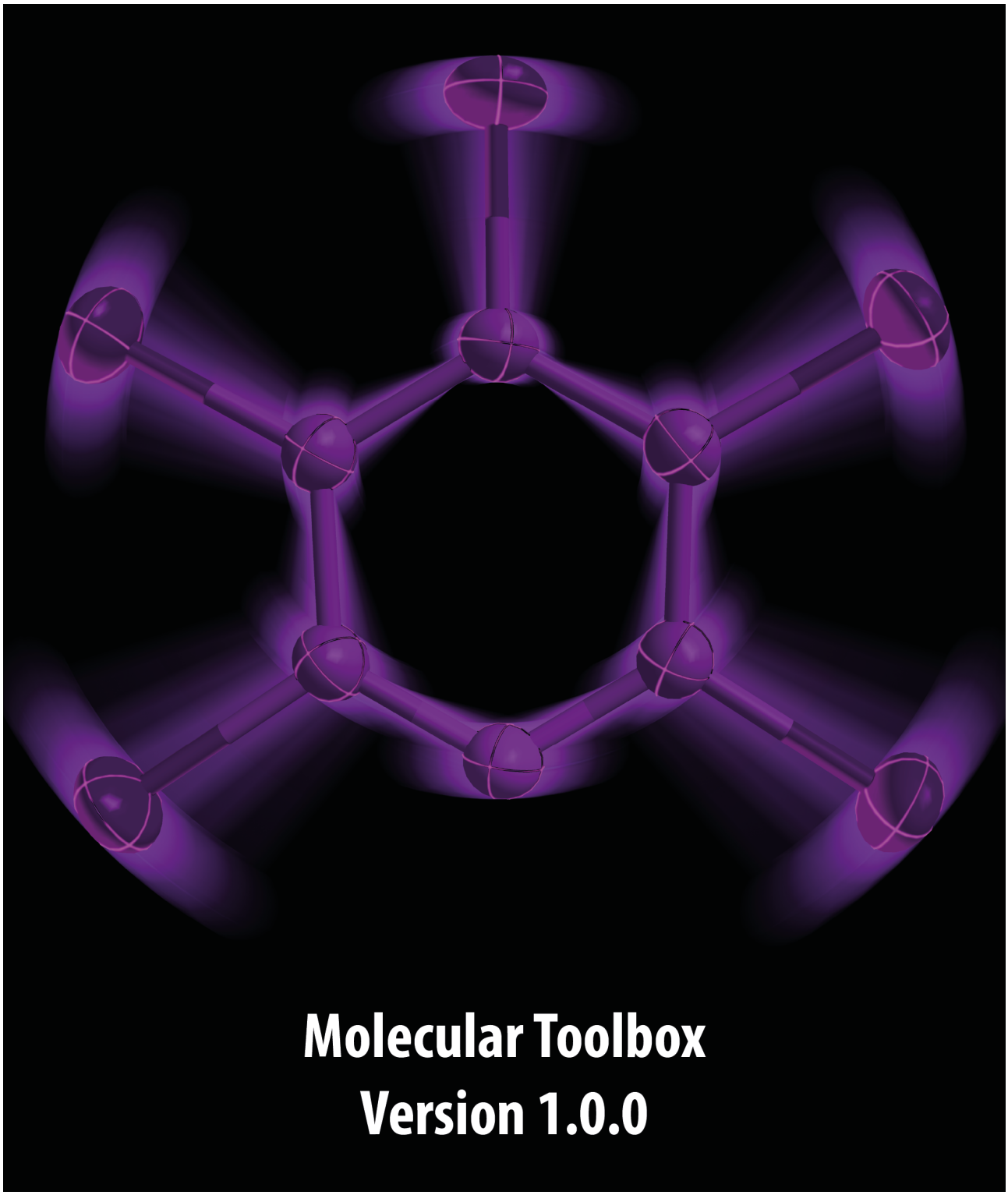


Molecular Toolbox
Version 1.0.0

The Dronskowski Group at RWTH Aachen University, Aachen, Germany, proudly presents:

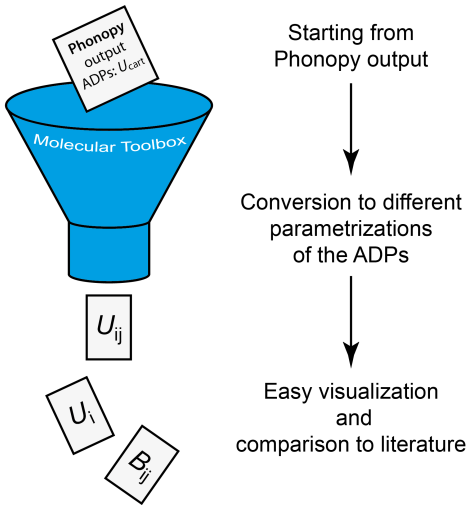


Contents

1 Introduction

1.1 Features of the Toolbox

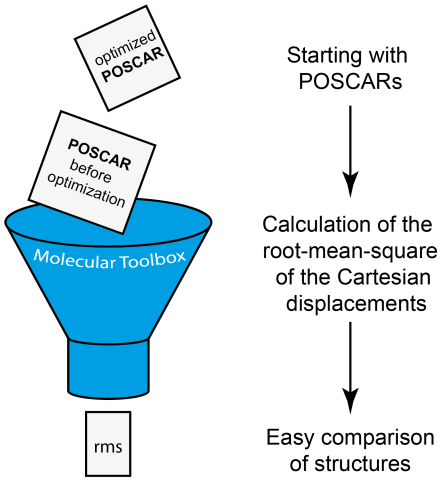
1. Conversion of Anisotropic Displacement Parameters to Different Parametrizations



This toolbox can convert anisotropic displacement parameters calculated with Phonopy referring to a Cartesian coordinate system (U_{cart}) to U_{cif}, B, U^*, β, U_i and U_{eq}. Moreover, U_{cif}, B, U^* and β from literature can be converted to U_{cif}, B, U^* and β.

This is all done according to:
R. W. Grosse-Kunstleve and P. D. Adams, *J. Appl. Crystallogr.*, **2002**, *35*, 477–480.
This article also includes the nomenclature of the different parametrizations.

2. Calculation of the Root-Mean-Square of the Cartesian Displacements



Moreover, this toolbox can also calculate the root-mean-square of the Cartesian displacements as defined in
J. George, V. L. Deringer, R. Dronskowski, *Inorg. Chem.*, **2015**, *54*, 956–962.

2 What to cite?

If you use the program to convert ADPs, please cite:

- R. W. Grosse-Kunstleve and P. D. Adams, *J. Appl. Crystallogr.*, **2002**, *35*, 477–480.
- J. George, A. Wang, V. L. Deringer, R. Wang, R. Dronskowski, U. Englert, *CrystEngComm*, **2015**, *17*, 7414–7422.

And please, don't forget to cite Phonopy

- A. Togo, I. Tanaka, *Scr. Mater.* **2015**, *108*, 1–5.

and the ADP calculation with Phonopy, if you have performed it:

- V. L. Deringer, R. P. Stoffel, A. Togo, B. Eck, M. Meven, R. Dronskowski, *CrystEngComm*, **2014**, *16*, 10907–10915..
- J. George, A. Wang, V. L. Deringer, R. Wang, R. Dronskowski, U. Englert, *CrystEngComm*, **2015**, *17*, 7414–7422.

If you use the program to calculate the root-mean-square of the Cartesian Displacements, please cite:

- J. George, V. L. Deringer, R. Dronskowski, *Inorg. Chem.*, **2015**, *54*, 956–962.

3 Getting Started

3.1 Install the MATLAB Toolbox

1. Open your MATLAB. Please use version 2015a or newer.
2. Browse with MATLAB's explorer to the folder with the Molecular-Toolbox files.
3. Double click on the **Molecular-Toolbox.mltbx**-file.
4. Click **Install**.
5. Check whether the Molecular-Toolbox is correctly installed. This is done by going to the **Home** tab, then switching to the **Environment** section, clicking on the **Add-Ons** icon and finally selecting **Manage Add-Ons**.
6. Run the scripts within the Matlab GUI or in the shell by typing:

```
matlab -nodisplay < NameOfTheScript.m
```

3.2 Run the Scripts

3.2.1 ADPs: Starting from Phonopy Output

In this section we explain what the script does to calculate the different parametrizations starting from the Phonopy output.

The script is called `ConvertADPsfromPhonopy.m` and is located in the provided folder `ConvertADPsfromPhonopy`. Please have a look at the specifications of the files `POSCAR` and `thermal_displacement_matrices.yaml` in the chapters 4.1 and 4.2. Make sure that you use a Phonopy version that produces the file `thermal_displacement_matrices.yaml` including the anisotropic displacement parameters in the correct format.

First, the toolbox has to be imported:

```
import Molecular-Toolbox.*
```

Give the pathway of the `POSCAR` file you used for the phonon calculation with Phonopy.

```
FilenameOfYourPOSCAR='POSCAR';
```

Did you rename the `thermal_displacement_matrices.yaml`? How is it called at the moment? Make sure you used the `POSCAR` specified above to create this file.

```
FilenameOfYourPhonopyADPFile='thermal_displacement_matrices.yaml';
```

Define `TMIN`,`TMAX` and `TSTEP` as in the calculation with Phonopy. This is needed to read the `thermal_displacement_matrices.yaml` correctly.

```
TMIN=0;
TMAX=300;
TSTEP=10;
```

Here, name the output files:

```
CIFwithUsFilename='U.cif';
CIFwithBsFilename='B.cif';
MainAxisComponentsFilename='U1U2U3';
UstarFilename='Ustar';
BetasFilename='Betas';
UeqFilename='Ueq';
```

These are the commands to get the files written. Change these commands only if you know what you are doing!

```
try
  CIF=FromPhonopywriter(FilenameOfYourPOSCAR,FilenameOfYourPhonopyADPFile,TMIN,TMAX,TSTEP);
  CIF.cifwrite(CIFwithUsFilename);
  CIF.cifwriteWithBs(CIFwithBsFilename);
  CIF.writeU1U2U3inFile(MainAxisComponentsFilename);
  CIF.writeUstarinFile(UstarFilename);
  CIF.writeBetasinFile(BetasFilename);
  CIF.writeUeqinFile(UeqFilename);
  disp('No error. All files are created :).');
catch exception
    disp(exception.message)
end
clear;
```

3.2.2 ADPs: Starting from U_{cif}, B, β and U^*

In this section we explain what the script does to calculate various parametrizations from different ADP parametrizations.

The scripts are called `ConvertADPsfrom*_example.m` and are located in the folders `ConvertADPsfrom*`, the star * stands for Betas, Bs, Ucif, Ustar.

First, the toolbox has to be imported:

```
import Molecular-Toolbox.*
```

Then, specify the type of input data you wish to convert. You can choose between the following keywords: `Betas`, `Bcif`, `Ucif`, or `Ustar`. It will only work if the spelling is correct.

```
INPUTType='Ustar';
```

Then, give the name of the input file. The tensor elements have to be sorted in the following way: `E11 E22 E33 E23 E13 E12`. It will produce incorrect results if the matrix elements are sorted differently. A sample input can be found in each of the sample folders or in chapter 4.3.

```
NameOfFile='UstarInput';
```

Specify the lattice parameters of the structure: a, b, c, α, β, γ.

```
a=5.350524;
b=5.192441;
c=15.058446;
alpha=89.999991;
beta=99.578013;
gamma=89.999998;
```

Specify the temperature range by giving the starting temperature `TSTART`, the end temperature `TEND` and the step `TSTEP`.

```
TSTART=0;
TEND=10;
TSTEP=10;
```

Specify the number of atoms in the structure.

```
NumberofAtomsPerTemperature=22;
```

Last, name the output files here:

```
OUTPUTU1U2U3='U1U2U3';
OUTPUTUcif='Ucif';
OUTPUTBcif='Bcif';
OUTPUTUstar='UstarOutput';
OUTPUTBetas='Betas';
OUTPUTUeq='Ueq';
```

These are the commands to get the files written. Change these commands only if you know what you are doing.

```
try
  if strcmp(INPUTType,'Bcif')
    NewWriter=FromBcifwriter(a,b,c,alpha,beta,gamma,NameOfFile,TSTART,TEND,TSTEP,NumberofAtomsPerTemperature);
  elseif strcmp(INPUTType,'Betas')
    NewWriter=FromBetaswriter(a,b,c,alpha,beta,gamma,NameOfFile,TSTART,TEND,TSTEP,NumberofAtomsPerTemperature);
  elseif strcmp(INPUTType,'Ucif')
    NewWriter=FromUcifwriter(a,b,c,alpha,beta,gamma,NameOfFile,TSTART,TEND,TSTEP,NumberofAtomsPerTemperature);
  elseif strcmp(INPUTType,'Ustar')
```

```
    NewWriter=FromUstarwriter(a,b,c,alpha,beta,gamma,NameOfFile,TSTART,TEND,TSTEP,NumberofAtomsPerTemperature);
  end

  NewWriter.writeU1U2U3inFile(OUTPUTU1U2U3);
  NewWriter.writeUcifinFile(OUTPUTUcif);
  NewWriter.writeBcifinFile(OUTPUTBcif);
  NewWriter.writeUstarinFile(OUTPUTUstar);
  NewWriter.writeBetasinFile(OUTPUTBetas);
  NewWriter.writeUeqinFile(OUTPUTUeq);
  disp('No error. All files are created :).');
catch exception
    disp(exception.message)
end
clear;
```

3.2.3 RMS of the Cartesian Displacements

This section explains how the root-mean-square of the Cartesian displacements is calculated. The script is called `CalculateRMS.m` and lies in the folder `CalculateRMS`.

First, the toolbox has to be imported:

```
import Molecular-Toolbox.*
```

The script does not work if the atoms of the two `POSCAR`s are sorted differently!

Just define the relative pathways of the sorted `POSCAR`s and the output file below. Then run the script.

```
PathwayPOSCAR1='POSCAR.vdw';
PathwayPOSCAR2='POSCAR_exp100K';
filenameOUTPUT='RMS';
```

If you do not know what you are doing, do not change anything here, please.

```
try
  RMS1=RMS(PathwayPOSCAR1,PathwayPOSCAR2);
  RMS1.printFileRMSandRMSxyz(filenameOUTPUT);
  disp('No error. All files are created :).');
catch exception
    disp(exception.message)
end
clear;
```

4 Requirements for the INPUT files

4.1 POSCAR

A sample POSCAR:

```
1  This is a sample POSCAR
2     1.00000000000000
3       5.3505    0.0000    0.0097
4      -0.0000    5.1924    0.0000
5      -2.5327    0.0000   14.8439
6     Cl   C    N
7      10    10     2
8  Direct
9    0.1149 0.6767 0.3328
10   0.1149 0.3232 0.8328
11   0.5321 0.1092 0.3241
12   0.5321 0.8907 0.8241
13   0.9618 0.1771 0.4921
14   0.9618 0.8228 0.9921
15   0.9686 0.8162 0.6599
16   0.9686 0.1837 0.1599
17   0.5316 0.3970 0.6530
18   0.5316 0.6029 0.1530
19   0.3582 0.7256 0.4215
20   0.3582 0.2743 0.9215
21   0.5410 0.9176 0.4169
22   0.5410 0.0823 0.9169
23   0.5430 0.6026 0.5637
24   0.5430 0.3973 0.0637
25   0.7340 0.9476 0.4921
26   0.7340 0.0523 0.9921
27   0.7373 0.7867 0.5677
28   0.7373 0.2132 0.0677
29   0.3619 0.5754 0.4928
30   0.3619 0.4245 0.9928
```

Requirements

- The scaling factor is not allowed to be smaller than 0.0 or equal to 0.0 (line 2).
- The names of the elements have to be included (line 6 of the sample `POSCAR`).
- No `Selective Dynamics` are allowed.
- Only `direct` coordinates are allowed (lines 8–30).
- If you calculate the RMS of the Cartesian displacements, the atoms in the two `POSCAR`s have to be sorted in the same way!
- If you have other questions about this format, please read the VASP documentation (`http://www.vasp.at`).

4.2 thermal_displacement_matrices.yaml

Sample thermal_displacement_matrices.yaml file from Phonopy

```
# Thermal displacement_matrices
natom:     22
cutoff_frequency: 0.130000
thermal_displacement_matrices:
- temperature:            0.0000000
  displacement_matrices:
  - [ 0.005083, 0.006011, 0.004118, −0.000079, −0.001577, −0.000813 ] # atom 1
  - [ 0.005083, 0.006011, 0.004118, 0.000079, −0.001577, 0.000813 ] # atom 2
  - [ 0.006039, 0.005437, 0.003949, 0.001616, −0.000284, −0.000838 ] # atom 3
  - [ 0.006039, 0.005437, 0.003949, −0.001616, −0.000284, 0.000838 ] # atom 4
  - [ 0.004167, 0.004238, 0.005192, 0.000015, −0.000068, −0.001359 ] # atom 5
  - [ 0.004167, 0.004238, 0.005192, −0.000015, −0.000068, 0.001359 ] # atom 6
  - [ 0.005127, 0.006339, 0.003925, 0.000279, −0.001566, −0.000697 ] # atom 7
  - [ 0.005127, 0.006339, 0.003925, −0.000279, −0.001566, 0.000697 ] # atom 8
  - [ 0.006152, 0.005091, 0.004119, 0.001507, 0.000065, −0.000875 ] # atom 9
  - [ 0.006152, 0.005091, 0.004119, −0.001507, 0.000065, 0.000875 ] # atom 10
  - [ 0.004275, 0.004507, 0.003681, 0.000350, −0.000623, −0.000852 ] # atom 11
  - [ 0.004275, 0.004507, 0.003681, −0.000350, −0.000623, 0.000852 ] # atom 12
  - [ 0.004361, 0.004422, 0.003636, 0.000622, −0.000489, −0.000861 ] # atom 13
  - [ 0.004361, 0.004422, 0.003636, −0.000622, −0.000489, 0.000861 ] # atom 14
  - [ 0.004474, 0.004318, 0.003685, 0.000601, −0.000371, −0.000833 ] # atom 15
  - [ 0.004474, 0.004318, 0.003685, −0.000601, −0.000371, 0.000833 ] # atom 16
  - [ 0.004111, 0.004176, 0.003762, 0.000408, −0.000442, −0.000887 ] # atom 17
  - [ 0.004111, 0.004176, 0.003762, −0.000408, −0.000442, 0.000887 ] # atom 18
  - [ 0.004316, 0.004425, 0.003633, 0.000461, −0.000630, −0.000810 ] # atom 19
  - [ 0.004316, 0.004425, 0.003633, −0.000461, −0.000630, 0.000810 ] # atom 20
  - [ 0.004931, 0.004991, 0.003964, 0.000758, −0.000766, −0.001472 ] # atom 21
  - [ 0.004931, 0.004991, 0.003964, −0.000758, −0.000766, 0.001472 ] # atom 22
```

Requirements

- Please make sure that you used the specified `POSCAR` to calculate the matrices!

4.3 Reading in U_{cif}, U^*, B, β

Sample file for reading in βs:

```
1  0
2  Cl1   0.003214 0.004401 0.000369 −0.000020 −0.000219 −0.000587
3  Cl2   0.003214 0.004401 0.000369 0.000020 −0.000219 0.000587
4  Cl3   0.004174 0.003981 0.000354 0.000414 0.000094 −0.000399
5  Cl4   0.004174 0.003981 0.000354 −0.000414 0.000094 0.000399
6  Cl5   0.002960 0.003103 0.000465 0.000004 0.000201 −0.000963
7  Cl6   0.002960 0.003103 0.000465 −0.000004 0.000201 0.000963
8  Cl7   0.003243 0.004641 0.000352 0.000072 −0.000225 −0.000461
9  Cl8   0.003243 0.004641 0.000352 −0.000072 −0.000225 0.000461
10 Cl9   0.004337 0.003727 0.000369 0.000386 0.000188 −0.000439
11 Cl10 0.004337 0.003727 0.000369 −0.000386 0.000188 0.000439
12 C1    0.002873 0.003300 0.000330 0.000090 −0.000001 −0.000563
13 C2    0.002873 0.003300 0.000330 −0.000090 −0.000001 0.000563
14 C3    0.002963 0.003237 0.000326 0.000160 0.000031 −0.000536
15 C4    0.002963 0.003237 0.000326 −0.000160 0.000031 0.000536
16 C5    0.003070 0.003161 0.000330 0.000154 0.000062 −0.000519
17 C6    0.003070 0.003161 0.000330 −0.000154 0.000062 0.000519
18 C7    0.002804 0.003057 0.000337 0.000105 0.000048 −0.000581
19 C8    0.002804 0.003057 0.000337 −0.000105 0.000048 0.000581
20 C9    0.002899 0.003240 0.000325 0.000118 −0.000004 −0.000519
21 C10  0.002899 0.003240 0.000325 −0.000118 −0.000004 0.000519
22 N1    0.003297 0.003654 0.000355 0.000195 −0.000024 −0.000954
23 N2    0.003297 0.003654 0.000355 −0.000195 −0.000024 0.000954
```

- First, the temperature is given (line 1).
- Then (lines 2–23, in this case), the tensor elements are given in the following order in each line after the name of the atom (e.g. Cl1):
  β_{11} β_{22} β_{33} β_{23} β_{13} β_{12}
- You could start with another temperature and tensor elements of the same structure in line 24.

Warning: If you don't use the order of the tensor elements described above, your results will be wrong!

5 Output files

5.1 ADPs: Starting from Phonopy Output

These are the files you will get:

- A CIF file in $P1$ space group including U_{cif}
- A CIF file in $P1$ space group including B
- A file containing the main-axis components U_1, U_2, U_3
- A file containing U_{eq}
- Files containing the tensor elements of β and U^*

5.2 ADPs: Starting from U_{cif}, B, β and U^*

These are the files you will get:

- A file containing the main-axis components U_1, U_2, U_3
- A file containing U_{eq}
- Files containing the tensor elements of U_{cif}, B, β and U^*

5.3 RMS of the Cartesian Displacements

- The file contains the rms, rms$_x$, rms$_y$ and rms$_z$ for all atoms and for each atom type.

6 Version history

1.0.0 (2016)	First Molecular-Toolbox version is released!

7 License of the Molecular Toolbox: BSD 3-Clause License