Cgenarris Documentation

Marom group

June 15, 2020

Contents

1 Features

- Support for special positions.
- Parallelized using MPI.
- Fast and efficient structure checks.
- Can generate in all possible Z given Z' <= 1.

2 Installation

2.1 cgenarris_mpi.x

Requirements

Any MPI C compiler which supports ANSI C99 / GNU99 standard and v3+ MPI standard.

Using Makefile

1. Uncompress the tar file.
2. Execute 'make cgenarris_mpi' in the *src* folder.
3. This will create cgenarris_mpi.x which is the desired executable.

NOTE:

1. You may change the C compiler using the environment variable CC. You may also uncomment the first line of the makefile and set the compiler.
2. Remove object files using 'make clean'.

2.2 pygenarris_mpi

Requirements

1. Any MPI C compiler which supports ANSI C99 / GNU99 standard and v3+ MPI standard.
2. SWIG (Simplified Wrapper code and Interface Generator).
3. Numpy and Mpi4py.
4. Setuptools for installation through setup.py.

Method 1: Using Makefile

1. Paste the location of Python.h headerfile in the Makefile. (for Anaconda v3.7 it should be ' anaconda/include/python3.7m/ ')
2. Execute 'make pygenarris_mpi'.
3. This will create pygenarris_mpi.so library from which you can import pygenarris_mpi functions.

Method 2: Using Distutils

1. Execute ' python setup.py build_ext --inplace'.
2. This will create pygenarris_mpi.so library file from which you can import pygenarris_mpi functions.

NOTE

1. You may change the compiler using the environment variable CC.
2. Pygenarris can be built with both python2 and python3.
3. Import the module using *import _pygenarris_mpi* or *import pygenarris_mpi.*

2.3 pygenarris

All non-mpi utility functions except structure generator.

Method 1: Using Makefile

1. Paste the location of Python.h headerfile in the Makefile. (for Anaconda v3.7 it should be ' anaconda/include/python3.7m/ ')
2. Execute 'make pygenarris'.
3. This will create pygenarris.so library file from which you can import pygenarris_mpi functions.

3 Cgenarris

Cgenarris is the structure generator written in pure C. It can be compiled into a binary and can run without installing the full genarris python package. If you are interested in only random crystal structures, you can compile and run cgenarris_mpi.x. It is parallelised using MPI version (The openMP version is now deprecated). The settings for generation are read from control.in file and molecule geometry is read from *geometry.in* file found in the working directory. The output is printed in file named *geometry.out* file.

First, the generator first identifies space groups that are compatible with molecular symmetry and given number of molecules in the unit cell. Structures are generated sequentially from lowest space group to the highest with a given spacegroup sampling scheme. Cell volumes are sampled from a normal distribution. The attempted structures are checked for closeness of molecules. If an atom of a molecule is too close to its own periodic image or another atom of a different molecule in a cell, the structure is discarded. The closeness checks are controlled by the specific radius proportion (*sr*). If the generation of a space group fails after *max_attempts* times, the generator moves to the next higher space group. The generated structures are printed to the file *geometry.out* in FHI-aims geometry format.

Input

Geometry of the molecule is read from the file *geometry.in* from the working directory. Note that input file is **case sensitive**.

1. *num_structures* is the number of structures from each space group. Type: integer.
2. *Z* number of molecules in the conventional cell. Type: integer.
3. *volume_mean* is the mean of the normal distribution from which volume is sampled. Type: float.
4. *volume_std* is the standard deviation of the volume distribution. Type: float.
5. *sr* is specific radius proportion. See Genarris paper for definition. Type: float.
6. *tol* is the tolerance for special position generation and space group detection. Type: float.
7. *max_attempts* is the maximum number of attempts before moving to the next space group. Type: integer.
8. *spg_distribution_type* specifies how spacegroups are sampled. Currently there are four options. Type: string.
   (a) *standard*: Molecules in special positions are sampled lesser than *num_structures.* More precisely, population of special position = *num_structures/*(order of special position). Population of spacegroups with molecule in general position is *num_structures.* This is the recommended setting.
   (b) *uniform:* All spacegroups have equal population equal to *num_structures.*
   (c) *chiral:* Only spacegroups that do not change the handedness of molecules are sampled.
   (d) *csd:* Top ten most frequent spacegroups in the Cambridge Structural Database (CSD) are sampled.

Output

1. A file named *geometry.out* with all the generated structures in FHI-aims geometry format. Standard output file has the log of generation.

Execution

You can run cgenarris_mpi.x using *mpirun -n #processors ./cgenarris_mpi.x* . *#processors* is the number of processors for running cgenarris.

4 Pygenarris_mpi

Pygenarris_mpi is a python API for C structure generator and associated functions. Import the module by:

import pygenarris_mpi

4.1 Generate a pool of random molecular crystals

mpi_generate_molecular_crystals_with_vdw_cutoff_matrix(filename, num_structures, Z, volume_mean, volume_std, sr, tol, max_attempts,spg_dist_type, world_comm)

Description

Generate random molecular crystals by space groups. First, the generator first identifies space groups that are compatible with molecular symmetry and given number of molecules in the unit cell. Structures are generated sequentially from lowest space group to the highest. Cell volumes are sampled from a normal distribution. The attempted structures are checked for closeness of molecules. If an atom of a molecule is too close to its own periodic image or another atom of a different molecule in a cell, the structure is discarded. The closeness checks are controlled by the specific radius proportion (sr). If the generation of a space group fails after max_attempts times, the generator moves to the next higher space group. The generated structures are printed to the file in FHI-aims geometry.in. format.

Input

1. Geometry of the molecule is read from the file *geometry.in* from the working directory.
2. *filename* is the name of the file to which generated structures are printed. Type: string
3. *num_structures* is the number of structures from each space group. Type: integer
4. *Z* number of molecules in the conventional cell. Type: integer
5. *volume_mean* is the mean of the normal distribution from which volume is sampled. Type: float
6. *volume_std* is the standard deviation of the volume distribution. Type: float
7. *sr* is specific radius proportion. See Genarris paper for definition. Type: float
8. *tol* is the tolerance for special position generation and space group detection. Type: float
9. *max_attempts* is the maximum number of attempts before moving to the next space group. Type: integer
10. *spg_dist_type* is space group distribution. Check see the input from cgenarris section. Type: string
11. *world_comm* is the MPI4PY communicator object.

Output

1. A file named *geometry.out* with all the generated structures in FHI-aims geometry format.

4.2 Identification of compatible space groups given molecular symmetry

find_allowed_positions_using_molecular_symmetry(point_group, Z, Z")

Description

This function finds the compatible space group positions using molecule's point group.

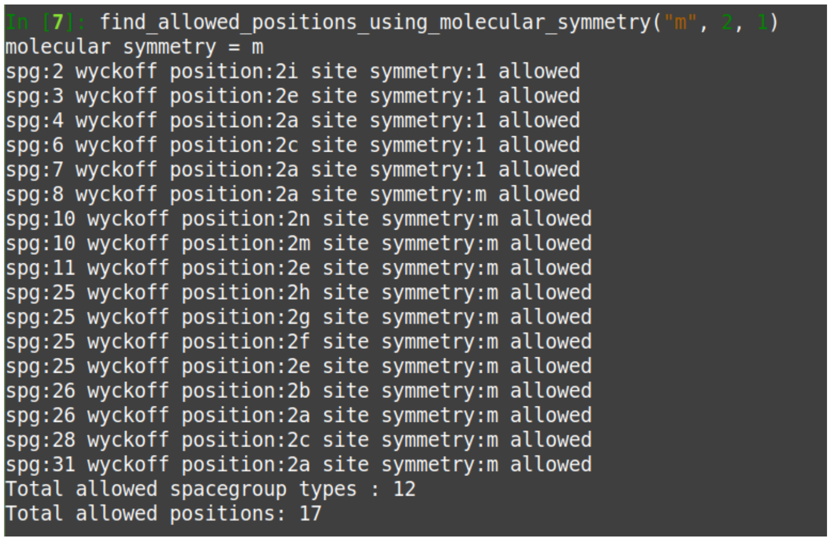
Input

1. *point_group* is the point_group of the molecule. Eg: "mmm" for tetracene. Type: String.
2. *Z* is the number of molecules in the conventional cell. Type: integer
3. *Z"* is the number of inequivalent molecules in the cell. *Not implemented! Set any integer.*

Output

1. Compatible Wyckoff positions and space groups are printed.

Example



```
In [7]: find_allowed_positions_using_molecular_symmetry("m", 2, 1)
molecular symmetry = m
spg:2 wyckoff position:2i site symmetry:1 allowed
spg:3 wyckoff position:2e site symmetry:1 allowed
spg:4 wyckoff position:2a site symmetry:1 allowed
spg:6 wyckoff position:2c site symmetry:1 allowed
spg:7 wyckoff position:2a site symmetry:1 allowed
spg:8 wyckoff position:2a site symmetry:m allowed
spg:10 wyckoff position:2n site symmetry:m allowed
spg:10 wyckoff position:2m site symmetry:m allowed
spg:11 wyckoff position:2e site symmetry:m allowed
spg:25 wyckoff position:2h site symmetry:m allowed
spg:25 wyckoff position:2g site symmetry:m allowed
spg:25 wyckoff position:2f site symmetry:m allowed
spg:25 wyckoff position:2e site symmetry:m allowed
spg:26 wyckoff position:2b site symmetry:m allowed
spg:26 wyckoff position:2a site symmetry:m allowed
spg:28 wyckoff position:2c site symmetry:m allowed
spg:31 wyckoff position:2a site symmetry:m allowed
Total allowed spacegroup types : 12
Total allowed positions: 17
```

4.3 Molecule closeness check using specific radius proportion

int check_structure_with_vdw_matrix(xtal, vdw_matrix)

Description

This function uses the built-in cgenarris structure checking function to see if the molecules are unphysically close. The vdW distance matrix defines the shortest distance between two atoms belonging to different molecules (which includes periodic images). The matrix should be symmetric and should be of size *total_atoms x total_atoms* where *total_atoms* is the number of atoms in a cell.

Input

First, you need to generate a SWIG object (C structure called crystal internally) of type crystal by:

1. *vdw_matrix* is a numpy array. Type: 2D numpy array of type "float32" (single precision) and size *total_atoms* x *total_atoms*.
2. create an object by *xtal = crystal().* [Or use the constructor generated automatically by SWIG; xtal = new_crystal().]
3. Then assign values using the function

- *create_crystal_from_array(xtal, lattice_vector, X, Y, Z, atoms, total_atoms, Z, spg)*
- *lattice vector* is a 3x3 numpy array which has the lattice vectors of the crystal in a row-wise form. Type: numpy 2D array.
- *X, Y, Z* are numpy arrays which the X, Y, and Z coordinates respectively. Type: numpy 1D array; length *total_atoms.*
- *atoms* contain the atom type. This is a string which defines the element at each coordinate. If an atom is represented by a single character, add a trailing space. Eg: for Carbon it should be "C " , for Bromine it should be "Br". Type: String of length 2 x *total_atoms*
- *total_atoms* is the number of atoms in the molecule. This should be the length of *X, Y, Z.* Type: integer.
- *Z* is the number of molecules in the unit cell. Type: integer
- *spg* is the space group (Not used by structure checker!). Type : integer
- IMPORTANT: It is assumed that coordinates are specified in molecule blocks. i.e, first molecule is first N coordinates, second molecule is from N+1 to 2N etc.
- IMPORTANT: Don't forget to free the swig object. This can be done by *delete_crystal(xtal).*

Output

Returns 0 if the structure is unphysical, returns 1 if the structure passes the test.

4.4 Number of space groups compatible with molecule's symmetry

int num_compatible_spacegroups(Z, tolerance)

Description

Function to get information about the allowed space groups and Wyckoff positions for a given molecule.

Input

1. Z is the number of molecules in the unit cell. Type: int.
2. *tolerance* is the tolerance for compatibility check. Type: float.
3. Molecule geometry is read from *geometry.in* file from the present working directory.

Output

1. Returns the allowed space groups as an integer.
2. Prints the detailed output to stdout which includes Wyckoff positions available for given Z .