OctaDist Documentation

Release 3.0.0

OctaDist Development Team

Mar 13, 2021

User Documentation

OctaDist: A tool for computing the distortion parameters in coordination complexes.

OctaDist (**Octa**hedral **Dist**ortion calculator) is an inorganic chemistry and crystallography program for computing the distortion parameters, such as distance and angle distortions, in coordination complexes. For example, they are used for tracking structural change of the spin-crossover complex when the electronics spin-state changes from low-spin to high-spin, and vice versa. OctaDist can also be used to study other kind of the metal complex such as perovskite and metal-organic framework (MOF).

- Official homepage: https://octadist.github.io
- Github repository: https://github.com/OctaDist/OctaDist

CHAPTER 1

Program Status

OctaDist is maintained on Github version control system. All versions has been continuously tested using Travis CI. Currently, OctaDist project has two branches: Master (stable) and nightly-build (dev).

Branch	Version	Status
Master	3.0.0	Active
Nightly-build	3.1.0	Active

Note: OctaDist is open-source computer software and freely distributed under The GNU General Public License v3.0.

Tip: This documentation is generated be both user and reference code manuals. For more details, please go to the development page.

CHAPTER 2

Citation

Please cite this project when you use OctaDist for scientific publication.

```
Ketkaew, R.; Tantirungrotechai, Y.; Harding, P.; Chastanet, G.; Guionneau, P.;␣
↪Marchivie, M.; Harding, D. J.
OctaDist: A Tool for Calculating Distortion Parameters in Spin Crossover and␣
↪Coordination Complexes.
Dalton Trans., 2021,50, 1086-1096. https://doi.org/10.1039/D0DT03988H
```

BibTeX

```
@article{Ketkaew2021,
   doi = {10.1039/d0dt03988h},
   url = {https://doi.org/10.1039/d0dt03988h},
   year = {2021},
   publisher = {Royal Society of Chemistry ({RSC})},
   volume = {50},
   number = {3},
   pages = {1086--1096},
   author = {Rangsiman Ketkaew and Yuthana Tantirungrotechai and Phimphaka Harding␣
↪and Guillaume Chastanet and Philippe Guionneau and Mathieu Marchivie and David J.␣
↪Harding},
   title = {OctaDist: a tool for calculating distortion parameters in spin crossover␣
↪and coordination complexes},
   journal = {Dalton Transactions}
}
```

CHAPTER 3

Bug report

For reporting a bug in OctaDist, please submit issues on OctaDist Github issues page. We appreciate all help and contribution in getting program development.

CHAPTER 4

User Documentation

genindex, modindex, search

4.1 Getting Started

Welcome to the first section of the OctaDist documentation. Here you can find all information of OctaDist.

4.1.1 Why OctaDist?

Octahedral complex can be simply classified into two types: regular and distorted octahedron. The complexes with regular octahedral geometry (perfect octahedron) are expected to form, when all of the ligands are of the same kind. In contrast, if the ligands are of different kinds, the complex would turns the distorted octahedron instead. Octahedral distortion parameters has been widely used for determining the change of the distortion of the complexes.

Even though the people in community generally calculate the octahedral distortion parameters for their complexes, but they not used a certain way to do this. Moreover, there is no software for determining this kind of parameter yet. Therefore, we present the OctaDist program as a choice for those who are interested in this.

4.1.2 Features

Structural distortion analysis

- Determination of regular, irregular distorted, very distorted, and non-octahedral octahedral complexes
- **Calculation of octahedral distortion parameters**
  - Mean distance: d_{mean}
  - Distance distortion: ζ
  - Angle distortion: Σ
  - Torsional distortion: Θ

- Tilting distortion parameter: Δ

Molecular visualizations

- 3D modelling of complex
- Display of eight faces of octahedron
- Atomic orthogonal projection and projection plane
- Twisting triangular faces
- Molecular superposition (Overlay)

Tools and Utilities

- Structural parameters
- Surface area
- Scripting Run supported
- Relationship plot between parameters
- Least-squares plane of selected ligand atoms
- Jahn-Teller distortion parameters
- Root-mean-square deviation of atomic positions (RMSD)

Capabilities

- Cross-platform for both 32-bit and 64-bit systems
- Graphical user interface (GUI)
- Command line interface (CLI)
- User-friendly interactive scripting code
- User-adjustable program setting
- Simple and flexible processes of use
- On top of huge and complicated complexes
- Support for several output of computational chemistry software, including Gaussian, Q-Chem, ORCA, and NWChem

Architectures

- Python-based program binding to Tkinter GUI toolkit and tested on PyCharm (Community Edition)
- Encapsulation of data, variable, and function as Class/Object.

4.1.3 Distortion parameters

Mathematical expression of the octahedral distortion parameters are given by following equations

- ζ parameter[1]

$$\zeta = \sum_{i=1}^{6} |d_i - d_{mean}|$$

where d_i is individual M-X bond distance and d_{mean} is mean metal-ligand bond distance.

- Δ parameter[2]

$$\Delta = \frac{1}{6} \sum_{i=1}^{6} (\frac{d_i - d_{mean}}{d_{mean}})^2$$

where d_i is individual M-X bond distance and d_{mean} is mean metal-ligand bond distance.

- Σ parameter[3]

$$\Sigma = \sum_{i=1}^{12} |90 - \phi_i|$$

where ϕ_i in individual cis angle.

- Θ parameter[4]

$$\Theta = \sum_{i=1}^{24} |60 - \theta_i|$$

where θ_i is individual angle between two vectors of two twisting face.

4.1.4 System requirements

Minimum system requirements for OctaDist:

- Windows 7/8/10
- Linux (X11 Start)
- OS X 10.8+ and macOS 10.12+

4.2 Download OctaDist

4.2.1 Stable Version

The latest stable release of OctaDist is available for following OS and platforms:

[1] M. Buron-Le Cointe, J. Hébert, C. Baldé, N. Moisan, L. Toupet, P. Guionneau, J. F. Létard, E. Freysz, H. Cailleau, and E. Collet. - Intermolecular control of thermoswitching and photoswitching phenomena in two spin-crossover polymorphs. Phys. Rev. B 85, 064114.

[2] M. W. Lufaso and P. M. Woodward. - Jahn–Teller distortions, cation ordering and octahedral tilting in perovskites. Acta Cryst. (2004). B60, 10-20. DOI: 10.1107/S0108768103026661

[3] J. K. McCusker, A. L. Rheingold, D. N. Hendrickson. Variable-Temperature Studies of Laser-Initiated 5T2 → 1A1 Intersystem Crossing in Spin-Crossover Complexes: Empirical Correlations between Activation Parameters and Ligand Structure in a Series of Polypyridyl. Ferrous Complexes. Inorg. Chem. 1996, 35, 2100.

[4] M. Marchivie, P. Guionneau, J.-F. Létard, D. Chasseau. Photo-induced spin-transition: the role of the iron(II) environment distortion. Acta Crystal-logr. Sect. B Struct. Sci. 2005, 61, 25.

Platform	Version	Download	
		Full version	Lite version
Windows OS		Full (exe) / Full (zip)	Lite (exe) / Lite (zip)
Linux OS		Full (tar.gz)	N/A
macOS			
PyPI		`pip install octadist`	
Anaconda		`conda install -c rangsiman octadist`	

Note: Both full and lite versions of OctaDist are open-source and free to download under the GNU v.3 license. The full version contains all capabilities including standard calculations, structural analysis, and molecular visualization, whereas the lite version includes only standard calculations.

4.2.2 Development Version

An on-going development build of OctaDist, called `nightly-build` branch. The tarball can be downloaded at Dev-build (zip) or use the following command:

```
wget https://github.com/OctaDist/OctaDist/archive/nightly-build.zip
```

You can also use `pip` to install the latest development build version on your system using the following command:

```
pip install git+https://github.com/octadist/octadist.git@nightly-build
```

Note: Python version must be equal or higher than 3.5. See *Development* for more details.

4.2.3 Release Archives

The source code and executable of all version and release note can be found at

Version	Release date	Download link	Stats
2.6.1	Aug 24, 2019	dl-2.6.1	
2.6.0	Jun 22, 2019	dl-2.6.0	
2.5.4	Jun 10, 2019	dl-2.5.4	
2.5.3	May 22, 2019	dl-2.5.3	
2.5.2	May 6, 2019	dl-2.5.2	
2.5.1	May 1, 2019	dl-2.5.1	
2.5.0	Apr 25, 2019	dl-2.5.0	
2.4	Apr 21, 2019	dl-2.4	
2.3-beta	Apr 14, 2019	dl-2.3-beta	
2.3-alpha	Mar 6, 2019	dl-2.3-alpha	
2.2	Feb 9, 2019	dl-2.2	
2.1	Jan 25, 2019	dl-2.1	
2.0	Jan 24, 2019	dl-2.0	
1.3	Jan 20, 2019	dl-1.3	
1.2	Jan 12, 2019	dl-1.2	
1.1	Jan 8, 2019	dl-1.1	
1.0	Jan 8, 2019	dl-1.0	

Total download:

4.3 Install OctaDist

OctaDist is a cross-platform software which is available for Windows, Linux, and macOS; for both 32-bit and 64-bit systems. You can install OctaDist by several ways, depending on your system and purpose.

4.3.1 Windows

Most of the Windows end-users do not have Python installed on their OS, so we strongly suggest you download and use a ready-to-use OctaDist executable.

Running OctaDist can be completed in a few steps as follows:

1. Download program executable (*.exe) to your machine:

```
OctaDist-*-Win-x86-64.exe
```

2. Right click on program icon and select:

```
Run as administrator
```

3. Click:

```
Yes
```

Note: Windows Defender might recognize OctaDist as third-party software. For first time starting OctaDist in Windows, you should run it as an administrator with full rights.

4.3.2 Linux

OctaDist is available on Python package index library, which can be found at https://pypi.org/project/octadist.

The end-user can use `pip`, a Python package-management system, to find and install OctaDist and other dependencies simultaneously.

Installing OctaDist can be completed in a few steps as follows:

1. Use `pip` command to install OctaDist:

```
pip install octadist
```

2. Execute OctaDist GUI, just type:

```
octadist
```

or:

```
octadist_gui
```

3. If you want to run OctaDist with command-line, just type:

```
octadist_cli
```

4.3.3 macOS

Like Linux, installing OctaDist on macOS can be completed in a few steps as follows:

1. Press **Command** - **spacebar** to launch Spotlight and type `Terminal`, then double-click the search result.
2. Use `pip` command to install OctaDist:

   ```
   pip install octadist
   ```

3. Execute OctaDist GUI, just type:

   ```
   octadist
   ```

   or:

   ```
   octadist_gui
   ```

4. If you want to execute OctaDist with command-line, just type:

   ```
   octadist_cli
   ```

4.3.4 PyPI

The following commands are also useful for those who want to play with `pip`:

- Show info of package:

  ```
  pip show octadist
  ```

- Install requirements packages:

  ```
  pip install -r requirements.txt
  ```

- Install or upgrade to the latest version:

  ```
  pip install --upgrade octadist
  ```

- Install/upgrade/downgrade to a certain version, for example, version 3.0.0:

  ```
  pip install --upgrade octadist==3.0.0
  ```

- Install the package with a specific version of Python. for example:

  ```
  python3.7 -m pip install --upgrade --user octadidst
  ```

- Uninstall package:

  ```
  pip uninstall octadist
  ```

More details on installing Python package can be found its official website: https://packaging.python.org/tutorials/installing-packages.

4.3.5 Anaconda

OctaDist is also available on Anaconda cloud server. The channel of OctaDist is at https://anaconda.org/rangsiman/octadist.

- It can be installed on system using command:

```
conda install -c rangsiman octadist
```

- To update OctaDist to the latest version:

```
conda update -c rangsiman octadist
```

- You can also create a personal environment only for OctaDist. For example, the following commands will create new env called `newenv`, then activate to this new env, and then install OctaDist from conda server:

```
conda create -n newenv python=3.7
activate newenv
conda update --all
conda install -c rangsiman octadist
```

- To clean conda cache:

```
conda clean --all
```

Note: OctaDist package on Anaconda server has been imported from PyPI server.

4.3.6 Python Package

OctaDist is a Python package and can be directly implemented into other applications. For example, that OctaDist is a package may be useful for interactive python script.

1. Check if your system has all dependencies for OctaDist:

```
python CheckPyModule.py
```

2. Download the source code (*.tar.gz) to your machine, for example, at **Download** directory:

```
OctaDist-*-src-x86-64.tar.gz
```

3. Uncompress the tarball, using **tar**:

```
tar -xzvf OctaDist-*-src-x86-64.tar.gz
```

4. Move to OctaDist root directory, using **cd**:

```
cd OctaDist-*-src-x86-64
```

5. Execute program like a package (you have to stay outside **octadist** directory):

```
python -m octadist
```

or command-line:

```
python -m octadist_cli
```

Note: The PyPI channel of OctaDist is at https://pypi.org/project/octadist/.

Tip: PIP-compressed zip files of OctaDist are also available at https://pypi.org/project/octadist/#files.

4.4 Build OctaDist

This section will explain how to build OctaDist from source code. If you already have OctaDist installed on your system, this section may be skipped.

4.4.1 Prerequisites

This section will explain the dependency requirements for building OctaDist. As OctaDist is written in Python 3, you have to make sure that the version of Python on your system is equal or higher than 3.5. Check it by following command:

```
python --version
```

or

```
python3 --version
```

Tip: If you do have Python on the system, I would suggest you to read The Hitchhiker's Guide to Python. It is very useful!

Install Python 3 on:

- Windows
- Linux
- macOS

The following third-party packages are used in OctaDist.

```
numpy
scipy
matplotlib
rmsd
pymatgen
```

Actually, if you use `pip` to install OctaDist, the required dependencies will be installed automatically. However, you can install these packages yourself. This can be done with only one step:

```
pip install -r requirements.txt
```

4.4.2 Build the tarball, wheel, and egg

- `.tar.gz` : the tarball (supported by PIP)
- `.whl` : wheel file (supported by PIP)
- `.egg` : cross-platform zip file (supported by easy_install)

1. Build source code:

```
python setup.py sdist bdist_wheel bdist_egg
```

2. Install OctaDist:

```
python setup.py install
```

   or:

```
pip install dist/*.tar.gz
```

3. Run test zip files:

```
python setup.py test
```

4. Installed library of OctaDist will be install at `build/lib/octadist` directory.
5. Standalone executable (binary) file will be automatically added to environment variables, you can start OctaDist by calling its names anywhere:
   - To start graphical-interface:

```
octadist
```

   - To start command-line:

```
octadist_cli
```

Note: More details on Python package can be found its official website: https://packaging.python.org/tutorials/installing-packages.

4.4.3 Compile OctaDist to EXE

Program source code can be compiled as a standalone executable file (*.exe). Compilation can be completed easily using PyInstaller.

1. Upgrade pip:

```
pip install pip --upgrade
```

2. Install the latest version of PyInstaller:

```
pip install pyinstaller --upgrade
```

3. Check the version of PyInstaller:

```
pyinstaller --version
```

4. Change directory to `octadist` subdirectory, where `main.py` is, for example:

```
cd OctaDist-*-src-x86-64/octadist/
```

5. Compile a standalone, like this:

```
pyinstaller --onefile --windowed -n OctaDist-*-src-x86-64 main.py
```

6. The standalone executable will be build in `dist` directory.

Note: Other useful options for building executable can be found at PyInstaller manual.

4.5 Run OctaDist

OctaDist supports both a graphical user interface (GUI) and a command line interface (CLI).

4.5.1 Run OctaDist GUI using EXE

If you have a standalone executable (.exe) of OctaDist GUI on your system, run OctaDist by double-clicking the `.exe` file as if you open other program.

Note: OctaDist can take time to launch the application, usually 5 - 10 seconds. However, if the program does not start, please restart your system and run it again.

4.5.2 Run OctaDist GUI on the terminal

Moreover, OctaDist can be called on the terminal such as `CMD`, `PowerShell`, and `Terminal` as long as it is added to environment variable, like this:

```
octadist
```

4.5.3 Run OctaDist CLI

You can execute command-line OctaDist interface by typing `octadist_cli` on the terminal. If it is executed without argument, the help docs will show by default.

```
(py37) user@Linux:~$ octadist_cli

# output
usage: octadist_cli [-h] [-v] [-a] [-c] [-g] [-i INPUT] [-o] [-s OUTPUT]
                    [--par PARAMETER [PARAMETER ...]] [--show MOL [MOL ...]]

Octahedral Distortion Calculator:
A tool for computing octahedral distortion parameters in coordination complex.
For more details, please visit https://github.com/OctaDist/OctaDist.
```

(continues on next page)

(continued from previous page)

```
optional arguments:
-h, --help            show this help message and exit
-v, --version         show program's version number and exit
-a, --about           show program info
-c, --cite            show how to cite OctaDist
-g, --gui             launch OctaDist GUI (this option is the same as
                        'octadist' command
-i INPUT, --inp INPUT
                        input structure in .xyz format
-o, --out             show formatted output summary
-s OUTPUT, --save OUTPUT
                        save formatted output to text file, please specify
                        name of OUTPUT file without '.txt' extension
--par PARAMETER [PARAMETER ...]
                        select which the parameter (zeta, delta, sigma, theta)
                        to show
--show MOL [MOL ...]  show atomic symbol (atom) and atomic coordinate
                        (coord) of octahedral structure

Rangsiman Ketkaew       Updated on August 2019    E-mail: rangsiman1993@gmail.com
```

Using OctaDist to calculate the distortion of structure can be done as follows:

```
# Compute parameters
octadist_cli -i INPUT.xyz

# Compute parameters and show formatted output
octadist_cli -i INPUT.xyz -o

# Compute parameters and save output as file
octadist_cli -i INPUT.xyz -s OUTPUT
```

Tip: On Windows, you can check whether OctaDist is added to environment variables by using `where` command:

```
where octadist
```

For Linux and macOS, use `which` command instead:

```
which octadist
```

or

```
type -P "octadist" && echo "It's in path" || echo "It's not in path"
```

4.6 Example Calculations

4.6.1 Supported File Format

- **CIF file format**

File extension: `.cif` (https://en.wikipedia.org/wiki/Crystallographic_Information_File)

Crystallographic Information File (CIF). Example CIF is below:

```
data_ADH041

###############
## ENTRY ##
###############

_entry.id              ADH041

###############
## ATOM_SITE ##
###############
loop_
_atom_site.id
_atom_site.label_atom_id
_atom_site.label_comp_id
_atom_site.label_asym_id
_atom_site.auth_seq_id
_atom_site.cartn_x
_atom_site.cartn_y
_atom_site.cartn_z
_atom_site.occupancy
_atom_site.B_iso_or_equiv
_atom_site.label_entity_id
_atom_site.label_seq_id
1    O5*   G A   1       7.231  -2.196  -5.399  1.00 22.25   1 1
2    C5*   G A   1       6.950  -3.464  -4.723  1.00 15.86   1 1
3    C4*   G A   1       8.299  -4.018  -4.302  1.00 15.20   1 1
...
```

- **XYZ file format**

File extension: `.xyz` (https://en.wikipedia.org/wiki/XYZ_file_format)

```
<number of atoms>
comment line
<element 1>  <X> <Y> <Z>
<element 2>  <X> <Y> <Z>
<element 3>  <X> <Y> <Z>
...
```

- **Output of computational chemistry programs**

File extension: `.out` and `.log`

1. Gaussian
2. NWChem
3. ORCA
4. Q-Chem

4.6.2 Running the tests

Example 1

Example 1 for running the test on OctaDist PyPI

```
import octadist as oc

# The first atom must be metal center atom of octahedral structure.
# If not, please see example_2.py for how to handle this issue.

atom = ['Fe', 'O', 'O', 'N', 'N', 'N', 'N']

coord = [[2.298354000, 5.161785000, 7.971898000],  # <- Metal atom
         [1.885657000, 4.804777000, 6.183726000],
         [1.747515000, 6.960963000, 7.932784000],
         [4.094380000, 5.807257000, 7.588689000],
         [0.539005000, 4.482809000, 8.460004000],
         [2.812425000, 3.266553000, 8.131637000],
         [2.886404000, 5.392925000, 9.848966000]]

dist = oc.CalcDistortion(coord)
zeta = dist.zeta             # Zeta
delta = dist.delta           # Delta
sigma = dist.sigma           # Sigma
theta = dist.theta           # Theta

print("\nAll computed parameters")
print("-----------------------")
print("Zeta  =", zeta)
print("Delta =", delta)
print("Sigma =", sigma)
print("Theta =", theta)

# All computed parameters
# -----------------------
# Zeta  = 0.22807256171728651
# Delta = 0.0004762517834704151
# Sigma = 47.926528379270124
# Theta = 122.688972774546
```

Example 2

Example 2 for running the test on OctaDist PyPI

```
import octadist as oc

atom = ['O', 'O', 'Fe', 'N', 'N', 'N', 'N']

coord = [[1.885657000, 4.804777000, 6.183726000],
         [1.747515000, 6.960963000, 7.932784000],
         [2.298354000, 5.161785000, 7.971898000],  # <- Metal atom
         [4.094380000, 5.807257000, 7.588689000],
         [0.539005000, 4.482809000, 8.460004000],
         [2.812425000, 3.266553000, 8.131637000],
         [2.886404000, 5.392925000, 9.848966000]]

# If the first atom is not metal atom, you can rearrange the sequence
# of atom in list using coord.extract_octa method.

atom_octa, coord_octa = oc.io.extract_octa(atom, coord)
```

(continues on next page)

(continued from previous page)

```
dist = oc.CalcDistortion(coord_octa)
zeta = dist.zeta              # Zeta
delta = dist.delta            # Delta
sigma = dist.sigma            # Sigma
theta = dist.theta            # Theta

print("\nAll computed parameters")
print("------------------------")
print("Zeta  =", zeta)
print("Delta =", delta)
print("Sigma =", sigma)
print("Theta =", theta)

# All computed parameters
# ------------------------
# Zeta  = 0.22807256171728651
# Delta = 0.0004762517834704151
# Sigma = 47.926528379270124
# Theta = 122.688972774546
```

Example 3

Example 3 for running the test on OctaDist PyPI

```
import octadist as oc

# You can also import your input file, like this:

file = r"../example-input/Multiple-metals.xyz"

# Then use coord.extract_file to extract all atomic symbols and coordinates,
# and then use coord.extract_octa for taking the octahedral structure.

atom_full, coord_full = oc.io.extract_coord(file)
atom, coord = oc.io.extract_octa(atom_full, coord_full)

dist = oc.CalcDistortion(coord)
zeta = dist.zeta              # Zeta
delta = dist.delta            # Delta
sigma = dist.sigma            # Sigma
theta = dist.theta            # Theta

print("\nAll computed parameters")
print("------------------------")
print("Zeta  =", zeta)
print("Delta =", delta)
print("Sigma =", sigma)
print("Theta =", theta)

# All computed parameters
# ------------------------
# Zeta  = 0.0030146365519487794
# Delta = 1.3695007180404868e-07
# Sigma = 147.3168033970211
# Theta = 520.6407679851042
```

Example 4

Example 4 for running the test on OctaDist PyPI

```
import octadist as oc

file = r"../example-input/Multiple-metals.xyz"

atom_full, coord_full = oc.io.extract_coord(file)

# If complex contains metal center more than one, you can specify the index metal
# whose octahedral structure will be computed.
# For example, this complex contains three metal atoms: Fe, Ru, and Rd.
# I add "2" as a second argument for choosing Ru as metal of interest.

atom, coord = oc.io.extract_octa(atom_full, coord_full, 2)

dist = oc.CalcDistortion(coord)
zeta = dist.zeta              # Zeta
delta = dist.delta            # Delta
sigma = dist.sigma            # Sigma
theta = dist.theta            # Theta

print("\nAll computed parameters")
print("-----------------------")
print("Zeta  =", zeta)
print("Delta =", delta)
print("Sigma =", sigma)
print("Theta =", theta)

# All computed parameters
# -----------------------
# Zeta  = 0.001616439510534251
# Delta = 3.5425830613072754e-08
# Sigma = 1.26579367508117
# Theta = 4.177042495798965
```

Example 5

Example 5 for running the test on OctaDist PyPI

```
import octadist as oc

file = r"../example-input/Multiple-metals.xyz"

atom_full, coord_full = oc.io.extract_coord(file)

# Graphical display for octahedral complex

my_plot = oc.draw.DrawComplex(atom=atom_full, coord=coord_full)
my_plot.add_atom()
my_plot.add_bond()
my_plot.add_legend()
my_plot.show_plot()
```

Example 6

Example 6 for running the test on OctaDist PyPI

```
import octadist as oc

file = r"../example-input/Multiple-metals.xyz"

atom_full, coord_full = oc.io.extract_coord(file)

# Display and automatically save image as .png file with user-specified name

my_plot = oc.draw.DrawComplex(atom=atom_full, coord=coord_full)
my_plot.add_atom()
my_plot.add_bond()
my_plot.add_legend()
my_plot.save_img()
my_plot.show_plot()

# Output image, Complex_saved_by_OctaDist.png, is stored at ../images directory
```

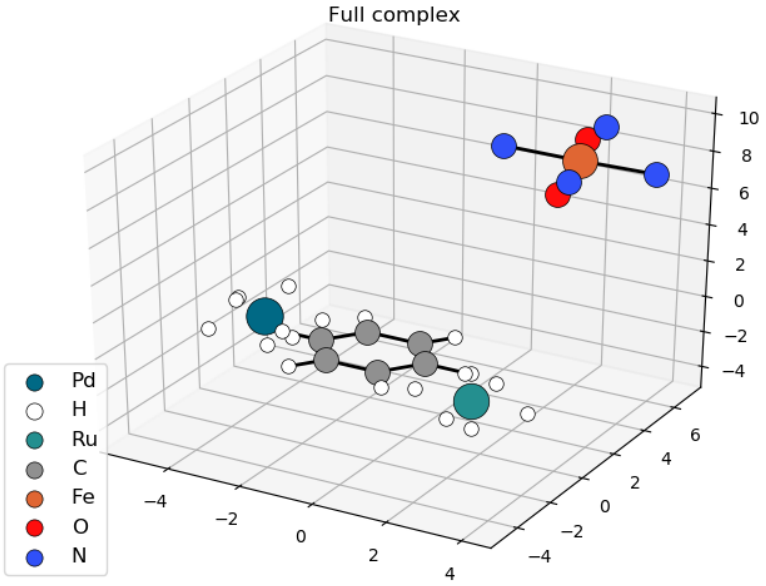

Fig. 1: Snapshot of structure saved by OctaDist.

4.7 Benchmarks

4.7.1 1. Perfect octahedral complex

Perfect iron metal complex:

```
Perfect-octahedron.xyz
Atom                    Cartesian coordinate
Fe          0.200698080     0.706806270     0.000000000
```

(continues on next page)

(continued from previous page)

```
O          1.660698080     0.706806270     0.000000000
O          0.200698080     2.166806270     0.000000000
O          0.200698080     0.706806270     1.460000000
O         -1.259301920     0.706806270     0.000000000
O          0.200698080    -0.753193730     0.000000000
O          0.200698080     0.706806270    -1.460000000
```

- d_{mean} = 1.460000 Angstrom
- ζ = 0.000000 Angstrom
- Δ = 0.00000000
- Σ = 0.00000000 degree
- Θ = 0.00000000 degree

4.7.2 2. [Fe(1-bpp)2][BF4]2 complex in low-spin state

The XRD structure taken from Malcolm Halcrow's CCDC library:

```
[Fe(1-bpp)2][BF4]2-LS-Full.xyz
Atom                        Cartesian coordinate
Fe          4.067400000     7.204000000    13.611700000
N          4.303300000     7.375000000    11.729200000
N          3.832600000     6.971500000    15.492600000
N          5.882200000     6.446100000    13.431200000
N          3.300200000     5.382800000    13.631600000
N          4.805500000     8.931800000    14.271600000
N          2.318400000     8.016500000    13.115200000
```

- d_{mean} = 1.958109 Angstrom
- ζ = 0.203199 Angstrom
- Δ = 0.000348
- Σ = 86.081494 degree
- Θ = 281.231091 degree

4.7.3 3. [Fe(1-bpp)2][BF4]2 complex in high-spin state

The XRD structure taken from Malcolm Halcrow's CCDC library:

```
[Fe(1-bpp)2][BF4]2-HS-Full.xyz
Atom                        Cartesian coordinate
Fe          4.904900000     6.913500000    14.248000000
N          4.982200000     6.876500000    12.110900000
N          4.671400000     6.741200000    16.368500000
N          6.853500000     6.086400000    13.701700000
N          5.683000000     8.779200000    15.108200000
N          4.107600000     4.898400000    14.643100000
N          2.957100000     7.673300000    13.543900000
```

- d_{mean} = 2.178519 Angstrom
- ζ = 0.155914 Angstrom

- Δ = 0.000168
- Σ = 150.814795 degree
- Θ = 496.648479 degree

4.7.4 4. Very distorted structure

Highly distorted structure:

```
Fe-very-distorted-octa.xyz
Atom                        Cartesian coordinate
Fe         18.268051000    11.289120000     2.565804000
O          19.074466000     9.706294000     3.743576000
O          19.823874000    10.436314000     1.381569000
N          18.364987000    13.407634000     2.249608000
N          16.149538000    11.306661000     2.913619000
N          18.599941000    12.116308000     4.528988000
N          17.364238000    10.733354000     0.657318000
```

- d_{mean} = 2.149211 Angstrom
- ζ = 0.082408 Angstrom
- Δ = 0.000066
- Σ = 182.673342 degree
- Θ = 673.278321 degree

4.8 Error and Fixing

4.8.1 1. OctaDist Startup Slow on Windows?

Windows Defender slow down OctaDist by scanning its file. You can fix this annoying issue by excluding OctaDist out of process scan list.

Here are the steps for adding OctaDist to exclusion list:

1. Go to `Start` > `Settings` -> `Update & Security` -> `Virus & threat protection`
2. Under Virus & threat protection settings select `Manage settings`
3. Under Exclusions, select `Add or remove exclusions` and select `Add exclusion`
4. Specify the name of OctaDist executable, for example:

   ```
   OctaDist-3.0.0-Win-x86-64.exe
   ```

5. Close OctaDist and run it again.

4.8.2 2. Missing some packages

If error message says *ImportError:* or *ModuleNotFoundError:*, some important packages have not been installed. To install all required packages, stay at top directory of OctaDist and type this command:

```
pip install -r requirements.txt
```

4.8.3 3. MPL error

If program crashes with confusing errors messages, you may need to set *MPLBACKEND* environment variable before running the program, like this:

```
export MPLBACKEND=TkAgg
```

4.8.4 4. Cannot connect to X11 server

If you run GUI using *octadist* or *octadist_gui* and then it fails with the following error:

```
(py37) nutt@Ubuntu:~$ octadist

Program Starts >>>
... OctaDist 3.0.0 January 2021 ...
Traceback (most recent call last):
    File "/home/nutt/.local/bin/octadist", line 10, in <module>
    sys.exit(run_gui())
    File "/home/nutt/.local/lib/python3.7/site-packages/octadist/__main__.py", line␣
↪35, in run_gui
    app = octadist.main.OctaDist()
    File "/home/nutt/.local/lib/python3.7/site-packages/octadist/main.py", line 68,␣
↪in __init__
    self.master = tk.Tk()
    File "/usr/lib/python3.7/tkinter/__init__.py", line 2023, in __init__
    self.tk = _tkinter.create(screenName, baseName, className, interactive,␣
↪wantobjects, useTk, sync, use)
_tkinter.TclError: couldn't connect to display ":0"
```

The above message implies that your system cannot connect to X11 server used for displaying the GUI of program. This error usually happens on Debian or Ubuntu (and Windows Subsystem for Linux on Windows). So, you need to install X11 server as follows:

X11 Client Installation

To install the *xauth* package, use *apt-get*:

```
sudo apt-get install xauth
```

X11 Server Installation

To install a minimal X11 on Ubuntu Server edition:

```
sudo apt-get install xorg
sudo apt-get install openbox
```

Tip: If you find any issues, do not hesitate to let us know. Your suggestions would help OctaDist getting improved.

4.9 Modules

4.9.1 Program structure

OctaDist is composed of the following modules:

Function	Description
main	Main program
calc	Calculating distortion parameters
draw	Displaying molecule
elements	Atomic properties
linear	Built-in mathematical functions
io	Manipulating atomic coordinates
plane	Manipulate projection plane
plot	Plotting graph and chart
popup	Error, warning, and info messages
projection	2D & 3D vector projections
scripting	Interactive code Console
structure	All data about structure
tools	Analysis tools by 3rd-party libraries
util	Frequently-used functions e.g. find atomic bonds

4.9.2 Application Program Interface (API)

API version	Description
octadist_gui	Graphical user interface (__main__.py)
octadist_cli	Command line interface (octadist_cli.py)

4.9.3 Source code

octadist.main

octadist.gui

octadist.cli

octadist.calc

octadist.draw

octadist.elements

octadist.io

octadist.linear

octadist.plane

octadist.plot

octadist.popup

octadist.projection

octadist.scripting

octadist.structure

octadist.tools

octadist.util

4.10 Development

OctaDist is written entirely in Python 3 binding to Tkinter toolkit. We have been developing OctaDist with the ease of use and flexibility. In the current version, it supports both of a graphical user interface (GUI) and a command line interface (CLI) version. The first one is mainly developed for the general end-users who are not familiar with command line, while the latter is primarily developed as a package which is appropriate for those who works with CLI. Having designed as a third party package, the command-line OctaDist version is an smart assistant helping with a wide range of your problems.

4.10.1 Contribution

To give a contribution on program development, please pull request on the OctaDist Github.

```
git clone https://github.com/OctaDist/OctaDist.git
git checkout nightly-build
git pull origin nightly-build
```

4.10.2 OctaDist Testing

When you have finished editing the source code of the program, you can use `setuptools` for testing OctaDist such as build and install. A `setup.py` file in top-level directory provides software testing as follows:

```
pip setup.py build
pip setup.py install
pip setup.py test
```

4.10.3 Bug report

If you found a bug in OctaDist, please submit it on issues page. We appreciate all help and contribution in getting program development.

4.10.4 Code maintenance

The source code of OctaDist is maintained on Github version control system. Both master revision and nightly development build have been being tested and deployed on Travis CI, a continuous integration service.

Source code on Github:

- Master (stable) version : github.com/OctaDist/OctaDist
- Nightly build version : github.com/OctaDist/OctaDist/tree/nightly-build

Tip: For OctaDist download stats, please go to https://octadist.github.io/stats.html.

4.11 Authors

The program is actively developed in international collaboration between the members of the Computational Chemistry Research Unit at Thammasat University, the Functional Materials & Nanotechnology CoE at Walailak University, Thailand, and the Switchable Molecules and Materials group at University of Bordeaux, France.

- **Rangsiman Ketkaew (Thammasat University, Thailand)** E-mail: rangsiman1993@gmail.com
- **Yuthana Tantirungrotechai (Thammasat University, Thailand)** E-mail: yt203y@gmail.com
- **David J. Harding (Walailak University, Thailand)** E-mail: hdavid@mail.wu.ac.th
- **Phimphaka Harding (Walailak University, Thailand)** E-mail: kphimpha@mail.wu.ac.th
- **Mathieu Marchivie (University of Bordeaux, France)** E-mail: mathieu.marchivie@icmcb.cnrs.fr

4.12 License

Authors Rangsiman Ketkaew, Yuthana Tantirungrotechai, David J. Harding, Phimphaka Harding, Mathieu Marchivie

Version 3.0.0 of 2021