
Ogre

Release 1.0

Shuyang Yang, Imanuel Bier

May 26, 2020

Contents

1	Introduction	1
2	Installation	2
3	Examples	3
3.1	Ogre	3
3.2	OgreSWAMP	3
4	Source Reference	4
4.1	Ogre	4
4.2	OgreSWAMP	5

CHAPTER 1

Introduction

Ogre is written in Python and interfaces with the FHI-aims code to calculate surface energies at the level of density functional theory (DFT). The input of Ogre is the geometry of the bulk molecular crystal. The surface is cleaved from the bulk structure with the molecules on the surface kept intact. A slab model is constructed according to the user specifications for the number of molecular layers and the length of the vacuum region. Ogre automatically identifies all symmetrically unique surfaces for the user-specified Miller indices and detects all possible surface terminations. Ogre includes utilities, OgreSWAMP, to analyze the surface energy convergence and Wulff shape of the molecular crystal.

CHAPTER 2

Installation

DO THIS

The examples are organized as follows

3.1 Ogre

3.2 OgreSWAMP

3.2.1 Wulffmaker

This will describe how this code utilizes Wulffmaker

This section is organized as follows

4.1 Ogre

```
class ogre.generators.OrganicSlabGenerator (initial_structure, miller_index, list_of_layers,  
                                           vacuum_size, supercell_size, work-  
                                           ing_directory)
```

Initialize the organic slab generator to cleave the surfaces for certain number of layers and Miller index.

Parameters

- **initial_structure** (*ASE Atoms structure.*) – The initial bulk structure to be cleaved.
- **miller_index** (*List[int]: [h, k, l]*) – The Miller index of the surface plane.
- **list_of_layers** (*List[int].*) – A list of layers to cleave.
- **vacuum_size** (*float*) – Height of vacuum size, unit: Angstrom. Note that the vacuum size would be added to both the bottom and the top of surface.
- **supercell_size** (*List[int]: [a, b, l]*) – Make a (a * b * l) supercell.
- **working_directory** (*str*) – The path to save the resulting slab structures.

```
cleave ()
```

Cleave the slab and repair the broken molecules.

List[List[structures]] list of list of slabs for the required list of layers. Each list contains one or multiple terminations.

```
ogre.generators.cleave_for_surface_energies (structure_path, structure_name, vac-  
                                             uum_size, list_of_layers, highest_index,  
                                             supercell_size, format_string)
```

Multiprocess launcher to cleave a surface plane with certain Miller index for different layers.

Parameters

- **structure_path** (*str*) – The path of initial bulk structure.
- **structure_name** (*str*) – The structure's name, used to create the directory.
- **highest_index** (*int*) – The highest value of Miller index used to calculate the Wulff shape.
- **miller_index** (*List[int]*: [*h*, *k*, *l*].) – The Miller index of the surface plane.
- **list_of_layers** (*List[int]*.) – A list of layers to cleave.
- **vacuum_size** (*float*) – Height of vacuum size, unit: Angstrom. Note that the vacuum size would be added to both the bottom and the top of surface.
- **supercell_size** (*List[int]*: [*a*, *b*, *c*]) – Make a (*a* * *b* * 1) supercell.
- **working_directory** (*str*) – The path to save the resulting slab structures.
- **format_string** (*str*) – The format of output file, could be "VASP", "FHI" or "CIF".

`ogre.generators.atomic_task` (*name*, *initial_structure*, *miller_index*, *list_of_layers*, *vacuum_size*, *supercell_size*, *format_string*)

Atomic task to cleave a surface plane with certain Miller index for different layers.

Parameters

- **initial_structure** (*ASE Atoms structure*.) – The initial bulk structure to be cleaved.
- **miller_index** (*List[int]*: [*h*, *k*, *l*].) – The Miller index of the surface plane.
- **list_of_layers** (*List[int]*.) – A list of layers to cleave.
- **vacuum_size** (*float*) – Height of vacuum size, unit: Angstrom. Note that the vacuum size would be added to both the bottom and the top of surface.
- **supercell_size** (*List[int]*: [*a*, *b*, *c*]) – Make a (*a* * *b* * 1) supercell.
- **working_directory** (*str*) – The path to save the resulting slab structures.

4.2 OgreSWAMP

- Demonstrate

class `ogre.utils.UniquePlanes` (*atoms*, *index=1*, *z_prime=1*, *symprec=0.001*, *verbose=True*, *force_hall_number=0*)

Given a ASE atoms crystal structure, finds and returns the unique planes. The algorithm first finds the space group for the system. More specifically, it finds the Hall Number which is particularly important for monoclinic systems where it's necessary to know the direction of the unique axis. Then, the symmetry operations of the space group are applied to miller index to identify how it transforms. If the miller index transforms to onto another, this these are necessarily specifying identical planes. This information is catalogued inside the class and a list of unique miller indices is returned.

You may also note, symmetry operations with translation components, including centering operations, screw axis, and glide planes create additional degenerate planes. For example, if the (100) plane is in the direction of a 2₁ screw, then the (100) and (200) planes are identical.

For a complete list of space groups and hall numbers, please visit: http://pmsl.planet.sci.kobe-u.ac.jp/~seto/?page_id=37&lang=en

Examples

```
>>> up = UniquePlanes(atoms, index=1, symprec=1e-3)
>>> print(up.unique_idx)
```

Parameters

- **atoms** (*ase.atoms*) – Crystal structure to identify unique planes
- **index** (*int*) – Maximum miller index to use in plane creation.
- **z_prime** (*float*) – Number of molecules in the asymmetric unit. If the number is equal to 1, then (100) and (200) will necessarily be the same. However, if there is more than 1 molecule in the asymmetric unit, this is not generally the case.
- **symprec** (*float*) – Precision used within spglib for space group identification.
- **verbose** (*bool*) – True if the user would like informative print statements during operation.
- **force_hall_number** (*int*) – Can be used if the user would like to force a certain hall number to be attempted first. This is useful to avoid the automatic cell standardization methods that are performed by spglib. Hall number information with respect to the international tables is here: http://pmsl.planet.sci.kobe-u.ac.jp/~seto/?page_id=37&lang=en

prep_idx()

Prepares all possible miller indices using the maximum index.

get_cell (*atoms=None*)

Returns a cell tuple of the atoms object for use with spglib

get_hall_number (*atoms=None, symprec=0.001*)

Get Hall number using spglib. List here: http://pmsl.planet.sci.kobe-u.ac.jp/~seto/?page_id=37&lang=en

Parameters

- **atoms** (*ase.atoms*) – Crystal structure to identify space group
- **symprec** (*float*) – Precision used for space group identification.

miller_to_real (*miller_idx*)

Convert miller indices to real space vectors to apply symmetry operations in real space.

Parameters **miller_idx** (*np.float64[:,3]*) – Matrix of miller indices.

real_to_miller (*real_space*)

Convert real space vectors into miller indices.

Parameters **real_space** (*np.float64[:,3]*) – Matrix of real space vectors.

find_unique_planes (*hall_number, z_prime=1, mt=False*)

From hall number, calculates the unique planes.

Parameters

- **hall_number** (*int*) – Hall number of space group for unique plane identification. For a complete list of space groups and hall numbers, please visit: http://pmsl.planet.sci.kobe-u.ac.jp/~seto/?page_id=37&lang=en
- **z_prime** (*float*) – Number of molecules in the asymmetric unit. If the number is equal to 1, then (100) and (200) will necessarily be the same. However, if there is more than 1 molecule in the asymmetric unit, this is not generally the case.

- **mt** (*bool*) – This is really just an argument from testing the algorithm. If mt is True, then the reciprocal metric tensor is used to convert the miller indices into real space before applying symmetry operations. This is the physically correct this to do. mt should always be set to True. If False, symmetry operations are applied to the miller indices, which is not crystallographically correct.

`ogre.utils.convergence_plots(structure_name, scf_path, threshold=5, max_layers=-1, font-size=16, pbe=False, boettger=True, combined_figure=True)`
 Plot the surface convergence plots and calculate the surface energy for each surface.

Parameters

- **structure_name** (*str*) – Structure's name.
- **scf_path** (*str*) – The path of SCF data in json format, stored by scripts in scripts/
- **threshold** (*float*) – Threshold that determines the convergence. The threshold should be given as a percent.
- **max_layers** (*int*) – Maximum number of layers to use before an error is raised that the convergence tolerance was not reached. Default value of -1 indicates that all layers will be used.
- **fontsize** (*int*) – Font size to plot the convergence plots.
- **pbe** (*bool*) – If True, PBE results will be included in the final plots..
- **boettger** (*bool*) – If True, Boettger results will be included in the final plots.
- **combined_figure** (*bool*) – If True, will create one large combined figure. If False, figures are save separately.

Returns *dict* – Dictionary that contains the surface energy values for TS and MBD from the Linear and Boettger surface slab energy fitting methods.

`ogre.utils.wulffmaker.wulffmaker_index(miller_index)`
 Returns the string to be used for the Wulffmaker default index.

Parameters **miller_index** (*iterable*) – Any iterable that holds the miller indices as a iterable of length 3.

Returns *str* – String to be copied to wulffmaker for the miller indices.

`ogre.utils.wulffmaker.wulffmaker_gamma(energy)`
 Returns the string to be used for the Wulffmaker default gamma values.

Parameters **energy** (*iterable*) – Any iterable that holds the surfac energies

Returns *str* – String to be copied to wulffmaker for the surface energies.

`ogre.utils.wulffmaker.wulffmaker_color(miller_index)`
 Returns the string to be used as the Wulffmaker pickColor default display settings.

Parameters **miller_index** (*iterable*) – Any iterable that holds the miller indices as a iterable of length 3.

Returns *str* – String to be copied to wulffmaker for the surface colors.

```
ogre.utils.wulffmaker.miller_index_legend(miller_index, figname='legend.pdf', save-
fig_kw={'dpi': 400}, legend_kw={'borderpad':
1, 'columnspacing': 1, 'fontsize': 22,
'frameon': False, 'handletextpad': 0.05,
'labelspring': 1, 'loc': 'center', 'ncol':
9}, Line2D_kw={'color': 'w', 'marker': 's',
'markersize': 40}, figure_kw={'figsize': (40,
10)})
```

Create a legend for the miller indices.

Parameters

- **miller_index** (*iterable*) – Any iterable that holds the miller indices as a iterable of length 3.
- **figname** (*str*) – File name to save the figure as.
- **savefig_kw** (*dict*) – Key-word arguments to be passed to fig.savefig
- **legend_kw** (*dict*) – Key-word arguments passed into the matplotlib.axes.legend command
- **Line2D_kw** (*dict*) – Key-word arguments passed into the Line2D command which creates the shape used in the plot for each miller index. Should not use the markerfacecolor or label arguments because these are used by function.
- **figure_kw** (*dict*) – Key-word arguments passed to matplotlib.pyplot.figure command.