Interface Builder ver. alpha - user manual

Jakub Kamiński
Department of Mathematics, University of California Los Angeles

November 14, 2014

1 Introduction

This document provides manual for the Interface Builder. The program is still in development stage as new features are being constantly added. At this stage the focus is more on the underlying theory and algorithms, whereas end user experience is put to the background. Once the development part is considered complete, the complete user interface will be created.

The program execution is controlled with the external input file. The compatibility between all the keywords combination has not been throughly tested, and some might lead to error.

The code does not feature exepction handling, therefore when it crashes it returns reference to the part of the code where the error occurred. One such instance might be, for example, when Miller indices referring to the plane the does not exist (for the given structure) are specified - the code will stop without giving detailed reason. This functionality will be added later.

2 Installation

2.1 Python

The program is written in Python 2.7. It uses standard Python modules + Numpy. In most modern Linux distributions and MacOS Python is pre-installed together with Numpy package. On Windows machines, Python and Numpy require additional installation.

2.2 Program execution

This manual will cover usage of Interface Builder in Unix based environments. Before first execution, make sure that the script is executable:

`chmode +x IfaceBuilder.py`

To execute the program, in the same directory issue:

`./IfaceBuilder.py options.dat`

where `options.dat` is the file with options controling program execution.

3 Program options

The options to the program are supplied in `options.dat` file. **The structure of the `options.dat` is very rigid, order of options matter, all the supplied keywords are case sensitise.** Again, there are no error handling routines avaialble in the code, therefore mistakes in the options file will not be detected. More modern implementation allowing for more freedom will be done later, once the code will reach end of development cycle.

3.1 Structure of `options.dat` file

The rigid file structure is given as follows:

```
substrate     Si.cif      # CIF file for Substrate
substrateIDX  100         # Miller indices for Substrate
deposit       SiO2.cif    # CIF file for Deposit
depositIDX    100         # Miller indices for Deposit
maxArea       250         # Maximum area of lattices
areaThresh    5.0         # Threshold for area misfit    [in %]
vecThresh     5.0         # Threshold for vectors misfit [in %]
angleThresh   5.0         # Threshold for angle misfit   [in %]
capAtmSub     Cl          # Atom to cap the substrate exposed surface or None
capAtmDep     H           # Atom to cap deposit exposed surface or None
fparam        1           # Parameter f for scoring function
nLayer        18.5        # Thickness of Deposit/Substrate slabs [A]
noConf        8           # Desired number of configurations to return
subAtRad      2.20        # Substrate atomic radius
depAtRad      2.20        # Deposit atomic radius
skipStep1     True       #Skip automatic generation of slabs distance True/False
```

We will follow explaining each option one by one.

- `substrate Si.cif`
  Name of the CIF file with substrate structure. The .cif file needs to be in the same directory as the IfaceBuilder.py

- `substrateIDX 100`
  Miller indices of substrate plane, for instance `100, 110, 111, 212,...`

- `deposit` and `depositIDX`
  Same as `substrate` and `substrateIDX` but for Deposit.

- `maxArea 250`
  Maximum allowed interface area. In Angtroms.

- `areaThresh 5.0`, `vecThresh 5.0`, `angleThresh 5.0`
  Maximum thresholds for Area, Lattice vectors and Angles between lattice vectors given in %. Decreasing those numbers will give fewer candidate structures, but with better fit. Decreasing those number will give give more candidate structure, but with bigger misfit.

- `capAtmSub Cl`
  Capping atom for the exposed surface of Substrate. Capping atoms are placed in the positions of the atoms that would be there in the true material. This implementation follows the new method to calculate the polar surface energies that is still under development, therefore this feature is also not very universal. Setting this option to `None` disables capping atoms.

- `capAtmDep H`
  Same as `capAtmSub` but for Deposit.

- `fparam 1`
  Parameter `f` for Scoring function

- `nLayer 18.5`
  Thickness of the Deposit and Substrate. In Angstroms. Manipulating this value for non-polar materials, one can find exposed surfaces that are "better" terminated than the other ones, i.e. having only one dangling bond instead of two.

- `noConf 8`
  Limits the number of output structures to the given number. If the given number is bigger than the actual number of structures, all the structures will be printed.

- `subAtRad 2.20`
  Atom radii of the atoms in Substrate. This is needed for the SCORING function.

- `depAtRad 2.20`
  Atom radii of the atoms in Deposit. This is needed for the SCORING function.

- `skipStep1 True`
  Setting this option to `False` will use SCORING function to determine the optimal distance between the Substrate and Deposit slabs. It increases the time to output structures considerablt. If it is set to `True`, the array with the desired distances between Substrate and Deposit slabs needs to be specified. Up to today, this is not possible to do from the level of the options file, but it requires manual inspection of `IfaceBuilder.py` script. This is not as hard as it seems.

  To do so, open the `IfaceBuilder.py` with text editor and go to line no. `3798`. The following variable should be defined there:

  ```
  if skipStep1:
          bondlist = [0.9,1.6,1.9,1.4,2.0]
  ```

  The numbers in the `bondlist` array are multipliers of average of atomic radii $r_{avg} = \frac{r_{Sub}+r_{Dep}}{2}$ (r_{Sub} and r_{Dep} given in `subAtRad` and `depAtRad`). Specifying, for instance, multiplier to `0.9` means that the Substrate and Deposit will be placed $0.9r_{avg}$ apart. The length of `bondlist` depends on number of alignments, that are created automatically, and can be found by running code once and noting the printed number.

4 Output

The program outputs following directories. The *main* directory:

`SiO2100-Si100`

The name of this directory is created automatically based on the names of the CIF input files and specified Miller indices. In the example above, it is interface between SiO_2(100) and Si(100).

The second output directory:

`SCORE`

contains results from Scoring function analysis. The filenames follow the pattern:

`OUTPUT-{D,S,SD}-{0,1,2...,n}.txt`

where `D,S,SD` corresponds to Deposit, Substrate, Substrate-and-Deposit respectively. The numbers from `0,1,2,...n` correspond to alignment number.

4.1 Structure of the *main* output directory

The main output directory contains subdirectories numbered from `0,...,n` where `n` is the number of alignments.

Each such directory contains files with the following extensions:

- `.xyz` - cartesian coordinates of interface
- `.in` - coordinates in input file format for FHI-AIMS DFT code
- `.gin` - coordinates in input file format for Gulp MM code
- `.idx` - miscellaneous information, such as number of atoms in comples:subsrtate:deposit; indices of atom of the exposed surfaces; area of the interface

In each file name, lable `-D` or `-S` denotes that those are coordinates for Deposit or Substrate respectively (no such label means it is the whole interface). The number from `0,....,m` is the number of the given candidate structure. The upper limit `m` is given in `options.dat` file in `noConf` line.