# Contents

1 Introduction 3

2 Getting Started 5
   2.1 Requirements ........................................... 5
   2.2 Installation ............................................. 5

3 Examples 7
   3.1 Element and species classes ............................... 7
   3.2 List building .............................................. 8
   3.3 Neutral combinations ..................................... 8
   3.4 Compound electronegativity ................................ 9
   3.5 Interfacing to machine learning .......................... 10

4 smact Python package 11
   4.1 Submodules .............................................. 15
       4.1.1 smact.properties module ........................... 15
       4.1.2 smact.screening module ............................. 16
       4.1.3 smact.oxidation_states module ...................... 18
       4.1.4 smact.builder module ............................... 19
       4.1.5 smact.distorter module ............................. 19
       4.1.6 smact.lattice module ............................... 20
       4.1.7 smact.lattice_parameters module .................... 21
       4.1.8 smact.data_loader module ........................... 23

5 Indices and tables 27

Python Module Index 29

Index 31
View the code on Github here.

Contents:
**smact** is a collection of tools and examples for “low-fi” screening of potential semiconducting materials through the use of simple chemical rules.

**smact** uses a combination of heuristics and models derived from data to rapidly search large areas of chemical space. This combination of methods allows **smact** to identify new materials for applications such as photovoltaics, water splitting and thermoelectrics. Read more about **smact** in our publications:

- Computational Screening of All Stoichiometric Inorganic Materials
- Computer-aided design of metal chalcohalide semiconductors: from chemical composition to crystal structure
- Materials discovery by chemical analogy: role of oxidation states in structure prediction

This approach is heavily inspired by the work of Harrison\(^1\) and Pamplin\(^2\). The work is an active project in the Walsh Materials Design Group.

SMACT is now available via pip install smact.

We are also developing a set of Jupyter notebook examples here.

---


\(^2\) [http://dx.doi.org/10.1016/0022-3697(64)90176-3](http://dx.doi.org/10.1016/0022-3697(64)90176-3) Pamplin, B. R. *J. Phys. Chem. Solids* (1964) 7 675–684
2.1 Requirements

The main language is Python 3 and has been tested using Python 3.6+. Basic requirements are Numpy and Scipy. The Atomic Simulation Environment (ASE), spglib, and pymatgen are also required for many components.

2.2 Installation

The latest stable release of SMACT can be installed via pip which will automatically setup other Python packages as required:

```
pip install smact
```

Alternatively, the latest master branch from the Git repo can be installed using:

```
pip install git+git://github.com/WMD-group/SMACT.git
```

Then ensure that the location of smact is on your PYTHONPATH.

For developer installation SMACT can be installed from a copy of the source repository (https://github.com/wmd-group/smact); this will be preferred if using experimental code branches.

To clone the project from Github and make a local installation:

```
git clone https://github.com/wmd-group/smact.git
cd smact
pip install --user -e .
```

With -e pip will create links to the source folder so that that changes to the code will be immediately reflected on the PATH.
CHAPTER 3

Examples

Here we will give a demonstration of how to use some of smact’s features. For a full set of work-through examples in Jupyter notebook form check out the examples section of our GitHub repo. For workflows that have been used in real examples and in published work, visit our separate repository.

3.1 Element and species classes

The element and species classes are at the heart of smact’s functionality. Elements are the elements of the periodic table. Species are elements, with some additional information; the oxidation state and the coordination environment (if known). So for example the element iron can have many oxidation states and those oxidation states can have many coordination environments.

```python
import smact
iron = smact.Element('Fe')
print("The element %s has %i oxidation states. They are %s."
(iron.symbol, len(iron.oxidation_states), iron.oxidation_states))
The element Fe has 8 oxidation states. They are [-2, -1, 1, 2, 3, 4, 5, 6].
```

When an element has an oxidation state and coordination environment then it has additional features. For example the Shannon radius\(^1\) of the element, this is often useful for calculating radius ratio rules\(^2\), or for training neural networks\(^3\).

```python
iron_square_planar = smact.Species('Fe', 2, '4_n')
print('Square planar iron has a Shannon radius of %s Angstrom' % iron_square_planar.
˓→shannon_radius)
Square planar iron has a Shannon radius of 0.77 Angstrom
```

---


3.2 List building

Often when using `smact` the aim will be to search over combinations of a set of elements. This is most efficiently achieved by setting up a dictionary of the elements that you want to search over. The easiest way to achieve this in `smact` is to first create a list of the symbols of the elements that you want to include, then to build a dictionary of the corresponding element objects.

The list can be built by hand, or if you want to cover a given range there is a helper function.

```python
import smact

elements = smact.ordered_elements(13, 27)
print(elements)

['Al', 'Si', 'P', 'S', 'Cl', 'Ar', 'K', 'Ca', 'Sc', 'Ti', 'V', 'Cr', 'Mn', 'Fe', 'Co']
```

For doing searches across combinations of elements it is then quickest to load the element objects into a dictionary and search by key. This avoids having to repopulate the element class at each iteration of the search.

```python
element_list = smact.element_dictionary(elements)
print(element_list)

{'Al': <smact.Element at 0x10ecc5890>,
 'Ar': <smact.Element at 0x10ecc5cd0>,
 'Ca': <smact.Element at 0x10ecc5a10>,
 'Cl': <smact.Element at 0x10ecc5d90>,
 'Co': <smact.Element at 0x10ecc5f90>,
 'Cr': <smact.Element at 0x10ecc5ed0>,
 'Fe': <smact.Element at 0x10ecc5f50>,
 'K': <smact.Element at 0x10ecc5e90>,
 'Mn': <smact.Element at 0x10ecc5f10>,
 'P': <smact.Element at 0x10ecc5990>,
 'Sc': <smact.Element at 0x10ecc5e10>,
 'Si': <smact.Element at 0x10ecc5150>,
 'Ti': <smact.Element at 0x10ecc5dd0>,
 'V': <smact.Element at 0x10ecc5e50>}
```

3.3 Neutral combinations

One of the most basic tests for establishing sensible combinations of elements is that they should form charge neutral combinations. This is a straightforward combinatorial problem of comparing oxidation states and allowed stoichiometries.

$$\sum_i Q_i n_i = 0$$

where \(i\) are the elements in the compound and \(Q\) are the charges. We have a special function, `smact_filter`, which does this checking for a list of elements. The `smact_filter` also ensures that all elements specified to be anions have electronegativities greater than all elements specified to be cations.

As input `smact_filter` takes:

- `els`: a tuple of the elements to search over (required)
- `threshold`: the upper limit of the stoichiometric ratios (default = 8)
• species_unique: whether or not we want to consider elements in different oxidation states as unique in our results (default is False).

We can look for neutral combos.

```python
import smact.screening

elements = ['Ti', 'Al', 'O']
space = smact.element_dictionary(elements)
# We just want the element items from the dictionary
eles = [e[1] for e in space.items()]
# We set a threshold for the stoichiometry of 4
allowed_combinations = smact.screening.smact_filter(eles, threshold=4)
print(allowed_combinations)
```

```
[(('Ti', 'Al', 'O'), (1, 3, 3)),
(('Ti', 'Al', 'O'), (2, 3, 4)),
(('Ti', 'Al', 'O'), (3, 1, 4)),
(('Ti', 'Al', 'O'), (1, 4, 4)),
(('Ti', 'Al', 'O'), (3, 1, 2)),
(('Ti', 'Al', 'O'), (3, 2, 4)),
(('Ti', 'Al', 'O'), (1, 2, 3)),
(('Ti', 'Al', 'O'), (1, 3, 4)),
(('Ti', 'Al', 'O'), (2, 4, 3)),
(('Ti', 'Al', 'O'), (2, 1, 3)),
(('Ti', 'Al', 'O'), (4, 2, 3)),
(('Ti', 'Al', 'O'), (1, 3, 2)),
(('Ti', 'Al', 'O'), (1, 2, 4)),
(('Ti', 'Al', 'O'), (3, 1, 3)),
(('Ti', 'Al', 'O'), (2, 1, 4)),
(('Ti', 'Al', 'O'), (1, 1, 3)),
(('Ti', 'Al', 'O'), (1, 2, 2)),
(('Ti', 'Al', 'O'), (1, 1, 2)),
(('Ti', 'Al', 'O'), (1, 1, 1)),
(('Ti', 'Al', 'O'), (2, 2, 3)),
(('Ti', 'Al', 'O'), (4, 1, 3)),
(('Ti', 'Al', 'O'), (1, 1, 3)),
(('Ti', 'Al', 'O'), (1, 4, 3)),
(('Ti', 'Al', 'O'), (2, 1, 2))]
```

There is an example of how this function can be combined with multiprocessing to rapidly explore large subsets of chemical space.

### 3.4 Compound electronegativity

One property that is often used in high-throughput screening where band alignment is important is the compound electronegativity. Ginley and Butler showed how the simple geometric mean of the electronegativities of a compound could be used to predict flat band potentials \(^4\). smact has a built in function to calculate this property for a given composition.

```python
import smact.properties

compound_electronegs = [smact.properties.compound_electroneg(elements = a[0], stoichs = a[1]) for a in ...]
```

3.5 Interfacing to machine learning

When preparing to do machine learning, we have to convert the compositions that we have into something that can be fed into an algorithm. Many of the properties provided in `smact` are suitable for this, one can take properties like electronegativity, mass, electron affinity etc etc (for the full list see `smact Python package`).

One useful representation that is often used in machine learning is the one-hot-vector formulation. A similar construction to this can be used to encode a chemical formula. A vector of length of the periodic table is set up and each element set to be a number corresponding to the stoichiometric ratio of that element in the compound. For example we could convert $\text{Ba}(\text{OH})_2$

```python
ml_vector = smact.screening.ml_rep_generator(['Ba', 'H', 'O'], stoichs=[1, 2, 2])
```

There is also an example demonstrating the conversion of charge neutral compositions produced by `smact` to a list of formulas using Pymatgen, or to a Pandas dataframe, both of which could then be used as input for a machine learning algorithm. For a full machine learning example that uses `smact`, there is a repository here which demonstrates a search for solar energy materials from the four-component (quaternary) oxide materials space.
smact Python package

The core module of smact contains classes which are used as fundamental data types within the smact package, as well as several utility functions. Particular attention is drawn to smact.element_dictionary(), which returns a dictionary of smact.Element objects indexed by their chemical symbols. Generating this dictionary once and then performing lookups is generally the fastest way of accessing element data while enumerating possibilities.

A collection of fast screening tools from elemental data

class smact.Element(symbol)
    Bases: object
    Collection of standard elemental properties for given element.
    Data is drawn from “data/element.txt”, part of the Open Babel package.
    Atoms with a defined oxidation state draw properties from the “Species” class.

    symbol
        Elemental symbol used to retrieve data
        Type  string

    name
        Full name of element
        Type  string

    number
        Proton number of element
        Type  int

    pauling_eneg
        Pauling electronegativity (0.0 if unknown)
        Type  float

    ionpot
        Ionisation potential in eV (0.0 if unknown)
Type float
e_affinity
   Electron affinity in eV (0.0 if unknown)
       Type float
dipol
   Static dipole polarizability in 1.6488e-41 C m^2 / V (0.0 if unknown)
       Type float
eig
   Electron eigenvalue (units unknown) N.B. For Cu, Au and Ag this defaults to d-orbital
       Type float
eig_s
   Eigenvalue of s-orbital
       Type float
SSE
   Solid State Energy
       Type float
SSEPauiling
   SSE based on regression fit with Pauling electronegativity
       Type float
oxidation_states
   Default list of allowed oxidation states for use in SMACT
       Type list
oxidation_states_sp
   List of oxidation states recognised by the Pymatgen Structure Predictor
       Type list
oxidation_states_icsd
   List of oxidation states that appear in the ICSD
       Type list
coord_envs
   The allowed coordination environments for the ion
       Type list
covalent_radius
   Covalent radius of the element
       Type float
mass
   Molar mass of the element
       Type float
crustal_abundance
   Crustal abundance in the earths crust mg/kg taken from CRC
       Type float
HHI_p
Herfindahl-Hirschman Index for elemental production
  Type  float

HHI_r
Hirfindahl-Hirschman Index for elemental reserves
  Type  float

Raises
  • NameError – Element not found in element.txt
  • Warning – Element not found in Eigenvalues.csv

class smact.Species(symbol, oxidation, coordination=4)
  Bases: smact.Element

Class providing data for elements in a given chemical environment

In addition to the standard properties from the periodic table (inherited from the Element class), Species objects use the oxidation state and coordination environment to provide further properties.

symbol
  Elemental symbol used to retrieve data

name
  Full name of element

oxidation
  Oxidation state of species (signed integer)

coordination
  Coordination number of species (integer)

pauling_eneg
  Pauling electronegativity (0.0 if unknown)

ionpot
  Ionisation potential in eV (0.0 if unknown)

e_affinity
  Electron affinity in eV (0.0 if unknown)

eig
  Electron eigenvalue (units unknown) N.B. For Cu, Au and Ag this defaults to d-orbital.

shannon_radius
  Shannon radius of Species.

ionic_radius
  Ionic radius of Species.

 Raises
  • NameError – Element not found in element.txt
  • Warning – Element not found in Eigenvalues.csv

smact.are_eq(A, B, tolerance=0.0001)
  Check two arrays for tolerance [1,2,3]==[1,2,3]; but [1,3,2]![1,2,3] :param A, B: 1-D list of values for approximate equality comparison :type A, B: lists :param tolerance: numerical precision for equality condition
**smact Documentation, Release 0.4.0**

- **Returns** boolean

  `smact.element_dictionary(elements=\text{None})`
  
  Create a dictionary of initialised smact.Element objects

  Accessing an Element from a dict is significantly faster than repeatedly initialising them on-demand within nested loops.

  **Parameters** elements (iterable of strings) – Elements to include. If None, use all elements up to 103.

  **Returns**

  Dictionary with element symbols as keys and smact.Element objects as data

- **Return type** dict

  `smact.lattices_are_same(lattice1, lattice2, tolerance=0.0001)`
  
  Checks for the equivalence of two lattices

  **Parameters** lattice1, lattice2 – ASE crystal class

  **Returns** boolean

  `smact.neutral_ratios(oxidations, stoichs=\text{False}, threshold=5)`
  
  Get a list of charge-neutral compounds

  Given a list of oxidation states of arbitrary length, yield ratios in which these form a charge-neutral compound. Stoichiometries may be provided as a set of legal stoichiometries per site (e.g. a known family of compounds); otherwise all unique ratios are tried up to a threshold coefficient.

  Given a list of oxidation states of arbitrary length it searches for neutral ratios in a given ratio of sites (stoichs) or up to a given threshold.

  **Parameters**

  - **oxidations** (list of ints) – Oxidation state of each site
  
  - **stoichs** (list of positive ints) – A selection of valid stoichiometric ratios for each site
  
  - **threshold** (int) – Maximum stoichiometry coefficient; if no ‘stoichs’ argument is provided, all combinations of integer coefficients up to this value will be tried.

  **Returns**

  - **exists** bool: True ifc any ratio exists, otherwise False
  
  - **allowed_ratios** list of tuples: Ratios of atoms in given oxidation states which yield a charge-neutral structure

  **Return type** (exists, allowed_ratios) (tuple)

  `smact.neutral_ratios_iter(oxidations, stoichs=\text{False}, threshold=5)`
  
  Iterator for charge-neutral stoichiometries

  Given a list of oxidation states of arbitrary length, yield ratios in which these form a charge-neutral compound. Stoichiometries may be provided as a set of legal stoichiometries per site (e.g. a known family of compounds); otherwise all unique ratios are tried up to a threshold coefficient.

  **Parameters**

  - **oxidations** – list of integers
  
  - **stoichs** – stoichiometric ratios for each site (if provided)
  
  - **threshold** – single threshold to go up to if stoichs are not provided

14 Chapter 4. smact Python package
Yields tuple – ratio that gives neutrality

\[
\text{smact.ordered_elements}(x, y)
\]
Return a list of element symbols, ordered by proton number in the range x -> y :param x,y: integers

Returns Ordered list of element symbols
Return type list

4.1 Submodules

4.1.1 smact.properties module

A collection of tools for estimating useful properties.

The “electronegativity of a compound” computed with \texttt{compound_electroneg()} is the rescaled geometric mean of electronegativity used in Nethercot’s recipe for estimating the photoelectric threshold:³

\[
\Phi^{AB} = 2.86(\chi_A\chi_B)^{1/2} + E_g/2.
\]

In other words, the computed group \(2.86(\chi_A\chi_B)^{1/2}\) is the mid-gap energy and the VBM/CBM positions can be estimated by subtracting/adding half of the band gap \(E_g\). This is an extension Mulliken’s electronegativity scale in which \(\chi_A = (I_A + E_A)/2\) (where \(I\) and \(E\) are respectively the ionisation potential and electron affinity.)²

\texttt{smact.properties.band_gap_Harrison}(anion, cation, verbose=False, distance=None)
Estimates the band gap from elemental data.

The band gap is estimated using the principles outlined in Harrison’s 1980 work “Electronic Structure and the Properties of Solids: The Physics of the Chemical Bond”.

Parameters

• \textbf{Anion (str)} – Element symbol of the dominant anion in the system
• \textbf{Cation (str)} – Element symbol of the the dominant cation in the system
• \textbf{Distance (float or str)} – Nuclear separation between anion and cation i.e. sum of ionic radii
• \textbf{verbose (bool)} – An optional True/False flag. If True, additional
• \textbf{is printed to the standard output. [Default (information) – False]}

Returns: Band gap (float): Band gap in eV

\texttt{smact.properties.compound_electroneg}(verbose=False, elements=None, stoichs=None, source='Mulliken')
Estimate electronegativity of compound from elemental data.

Uses Mulliken electronegativity by default, which uses elemental ionisation potentials and electron affinities. Alternatively, can use Pauling electronegativity, re-scaled by factor 2.86 to achieve same scale as Mulliken method (Nethercot, 1974) DOI:10.1103/PhysRevLett.33.1088.

Geometric mean is used (n-th root of product of components), e.g.:

\[
X_{\text{Cu2S}} = (X_{\text{Cu}} \times X_{\text{Cu}} \times C_S)^{1/3}
\]

Parameters

• **elements** *(list)* – Elements given as standard elemental symbols.

• **stoichs** *(list)* – Stoichiometries, given as integers or floats.

• **verbose** *(bool)* – An optional True/False flag. If True, additional information is printed to the standard output. [Default: False]

• **source** – String ‘Mulliken’ or ‘Pauling’; type of Electronegativity to use. Note that in SMACT, Pauling electronegativities are rescaled to a Mulliken-like scale.

**Returns** Estimated electronegativity (no units).

**Return type** Electronegativity (float)

### smact.properties

**smact.properties.eneg_mulliken** *(element)*

Get Mulliken electronegativity from the IE and EA.

**Parameters**

- **symbol** *(smact.Element or str)* – Element object or symbol

**Returns** Mulliken electronegativity

**Return type** mulliken (float)

#### 4.1.2 smact.screening module

**smact.screening.eneg_states_test** *(ox_states, enegs)*

Internal function for checking electronegativity criterion

This implementation is fast as it ‘short-circuits’ as soon as it finds an invalid combination. However it may be that in some cases redundant comparisons are made. Performance is very close between this method and eneg_states_test_alternate.

**Parameters**

- **ox_states** *(list)* – oxidation states corresponding to species in compound
- **enegs** *(list)* – Electronegativities corresponding to species in compound

**Returns**

True if cations have higher electronegativity than anions, otherwise False

**Return type** bool

**smact.screening.eneg_states_test_alternate** *(ox_states, enegs)*

Internal function for checking electronegativity criterion

This implementation appears to be slightly slower than eneg_states_test, but further testing is needed.

**Parameters**

- **ox_states** *(list)* – oxidation states corresponding to species in compound
- **enegs** *(list)* – Electronegativities corresponding to species in compound

**Returns**

True if cations have higher electronegativity than anions, otherwise False

**Return type** bool

**smact.screening.eneg_states_test_threshold** *(ox_states, enegs, threshold=0)*

Internal function for checking electronegativity criterion
This implementation is fast as it ‘short-circuits’ as soon as it finds an invalid combination. However it may be that in some cases redundant comparisons are made. Performance is very close between this method and eneg_states_test_alternate.

A ‘threshold’ option is added so that this constraint may be relaxed somewhat.

**Parameters**

- **ox_states** *(list)* – oxidation states corresponding to species in compound
- **enegs** *(list)* – Electronegativities corresponding to species in compound
- **threshold** *(Option(float))* – a tolerance for the allowed deviation from the Pauling criterion

**Returns**

- **True if cations have higher electronegativity than** anions, otherwise False

```python
smact.screening.ml_rep_generator (composition, stoichs=None)
```

Function to take a composition of Elements and return a list of values between 0 and 1 that describes the composition, useful for machine learning.

The list is of length 103 as there are 103 elements considered in total in SMACT.

e.g. Li2O -> [0, 0, 2/3, 0, 0, 0, 0, 1/3, 0 . . . .]

Inspired by the representation used by Legrain et al. DOI: 10.1021/acs.chemmater.7b00789

**Parameters**

- **composition** *(list)* – Element objects in composition OR symbols of elements in composition
- **stoichs** *(list)* – Corresponding stoichiometries in the composition

**Returns**

- **List of floats representing the composition that sum** to one

```python
smact.screening.pauling_test (oxidation_states, electronegativities, symbols=[], repeat_anions=True, repeat_cations=True, threshold=0.0)
```

Check if a combination of ions makes chemical sense, (i.e. positive ions should be of lower electronegativity).

**Parameters**

- **ox** *(list)* – oxidation states of elements in the compound
- **paul** *(list)* – the corresponding Pauling electronegativities of the elements in the compound
- **symbols** *(list)* – chemical symbols of each site
- **threshold** *(float)* – a tolerance for the allowed deviation from the Pauling criterion
- **repeat_anions** – boolean, allow an anion to repeat in different oxidation states in the same compound
- **repeat_cations** – as above, but for cations

**Returns** True if positive ions have lower electronegativity than negative ions
smact Documentation, Release 0.4.0

Return type  bool

smact.screening.pauling_test_old(ox, paul, symbols, repeat_anions=True, repeat_cations=True, threshold=0.0)

Check if a combination of ions makes chemical sense, (i.e. positive ions should be of lower Pauling electronegativity). This function should give the same results as pauling_test but is not optimised for speed.

Parameters

- ox (list) – oxidation states of the compound
- paul (list) – the corresponding Pauling electronegativities of the elements in the compound
- symbols (list) – chemical symbols of each site.
- threshold (float) – a tolerance for the allowed deviation from the Pauling criterion
- repeat_anions – boolean, allow an anion to repeat in different oxidation states in the same compound.
- repeat_cations – as above, but for cations.

Returns  True if positive ions have lower electronegativity than negative ions

Return type  (bool)

smact.screening.smact_filter(els, threshold=8, species_unique=True)

Function that applies the charge neutrality and electronegativity tests in one go for simple application in external scripts that wish to apply the general ‘smact test’.

Parameters

- els (tuple/list) – A list of smact.Element objects
- threshold (int) – Threshold for stoichiometry limit, default = 8
- species_unique (bool) – Whether or not to consider elements in different states as unique in the results.

Returns  Allowed compositions for that chemical system in the form [(elements), (oxidation states), (ratios)] if species_unique=True or in the form [(elements), (ratios)] if species_unique=False.

Return type  allowed_comps (list)

4.1.3 smact.oxidation_states module

smact.oxidation_states: Module for predicting the likelihood of species coexisting in a compound based on statistical analysis of oxidation states. It is possible to use the values obtained in the publication Materials Discovery by Chemical Analogy: Role of Oxidation States in Structure Prediction - DOI: 10.1039/C8FD00032H.

class smact.oxidation_states.Oxidation_state_probability_finder(probability_table=None)
Bases: object

Uses the model developed in the Faraday Discussions Paper (DOI:10.1039/C8FD00032H) to compute the likelihood of metal species existing in solids in the presence of certain anions.

compound_probability(structure, ignore_stoichiometry=True)

calculate overall probability for structure or composition.

Parameters
• **structure** (*pymatgen.Structure*) – Compound for which the probability score will be generated. Can also be a list of pymatgen or SMACT Species.

• **ignore_stoichiometry** (*bool*) – Whether to weight probabilities by stoichiometry. Defaults to false as described in the original paper.

**Returns** Compound probability

**Return type** compound_prob (float)

**get_included_species** ()

Returns a list of species for which there exists data in the probability table used.

**pair_probability** (*species1*, *species2*)

Get the anion-cation oxidation state probability for a provided pair of smact Species. I.e. \( P_{SA} = \frac{N_{SX}}{N_{MX}} \) in the original paper (DOI:10.1039/C8FD00032H).

**Parameters**

• **species1** (*smact.Species*) – Cation or anion species

• **species2** (*smact.Species*) – Cation or anion species

**Returns** Species-anion probability

**Return type** prob (float)

---

### 4.1.4 smact.builder module

A collection of functions for building certain lattice types. Currently there are examples here for the Perovskite and Wurzite lattice types, which rely on the Atomic Simulation Environment (ASE) `spacegroup.crystal()` function.

**smact.builder.cubic_perovskite** (*species*, *cell_par=[6, 6, 6, 90, 90, 90]*, *repetitions=[1, 1, 1]*)

Function to build a perovskite cell using the crystal function in ASE. 

**Parameters**

• **species** :str

• **cell_par**: list : Six floats/ints specifying 3 unit cell lengths and 3 unit cell angles.

• **repetitions**: list : Three floats specifying the expansion of the cell in x,y,z directions.

**Returns** SMACT Lattice object of the unit cell, ASE crystal system of the unit cell.

**smact.builder.wurtzite** (*species*, *cell_par=[2, 2, 6, 90, 90, 120]*, *repetitions=[1, 1, 1]*)

Function to build a wurzite cell using the crystal function is ASE.

**Parameters**

• **species** :str

• **cell_par**: list : Six floats/ints specifying 3 unit cell lengths and 3 unit cell angles.

• **repetitions**: list : Three floats specifying the expansion of the cell in x,y,z directions.

**Returns** SMACT Lattice object of the unit cell, ASE crystal system of the unit cell.

---

### 4.1.5 smact.distorter module

smact.distorter: Module for generating symmetry-unique substitutions on a given sub-lattice.

As input it takes the ASE crystal object (as built by smact.builder) and the sub-lattice on which substitutions are to be made. There is an example of how to use the code in Example_distort.py.

TODO: Add a functionality to check two Atoms objects against one another for equivalence.

**smact.distorter.build_sub_lattice** (*lattice, symbol*)

Generate a sub-lattice of the lattice based on equivalent atomic species.

**Parameters**

• **lattice** (*ASE crystal class*) – Input lattice
- **symbol** (*string*) – Symbol of species identifying sub-lattice

  **Returns**  
  sub_lattice: Cartesian coordinates of the sub-lattice of symbol

  **Return type**  
  list of lists

  smact.distorter.get_inequivalent_sites(*sub_lattice, lattice*)

  Given a sub lattice, returns symmetry unique sites for substitutions.

  **Parameters**

  - **sub_lattice** (*list of lists*) – array containing Cartesian coordinates of the sub-lattice of interest
  - **lattice** (*ASE crystal*) – the total lattice

  **Returns**  
  List of sites

  smact.distorter.get_sg(*lattice*)

  Get the space-group of the system.

  **Parameters**  
  lattice – the ASE crystal class

  **Returns**  
  integer number of the spacegroup

  **Return type**  
  sg (int)

  smact.distorter.make_substitution(*lattice, site, new_species*)

  Change atomic species on lattice site to new_species.

  **Parameters**

  - **lattice** (*ASE crystal*) – Input lattice
  - **site** (*list*) – Cartesian coordinates of the substitution site
  - **new_species** (*str*) – New species

  **Returns**  
  lattice

### 4.1.6 smact.lattice module

**class**  
smact.lattice.Lattice(*sites, space_group=1, strukturbericht=False*)

  Bases: object

  A unique set of Sites.

  Lattice objects define a general crystal structure, with a space group and a collection of Site objects. These Site objects have their own fractional coordinates and a list of possible oxidation states (see the Site class).

  Specific crystal structures with elements assigned to sites are “materials” and use the Atoms class from the Atomic Simulation Environment.

  **basis_sites**

  A list of Site objects [SiteA, SiteB, SiteC, ...]

  **comprising the basis sites in Cartesian coordinates**

  **space_group**

  Integer space group number according to the

  **International Tables for Crystallography.**

  **strukturbericht**

  Structurbericht identity, if applicable
Type e.g. ‘B1’

```python
lattice_vector_calc()
```

```python
class smact.lattice Site(position, oxidation_states=[0])
```

Bases: `object`

A single lattice site with a list of possible oxidation states.

The Site object is primarily used within Lattice objects.

```python
position
A list of fractional coordinates [x,y,z]
```

```python
oxidation_states
A list of possible oxidation states e.g. [-1,0,1]
```

### 4.1.7 `smact.lattice_parameters` module

This module can be used to calculate roughly the lattice parameters of a lattice type, based on the radii of the species on each site.

```python
smact.lattice_parameters.b10(shannon_radius)
The lattice parameters of Litharge
```  
**Parameters**

- `shannon_radius` *(list)* – The radii of the a,b ions

**Returns**

- Float values of lattics constants and angles (a, b, c, alpha, beta, gamma)

**Return type** *(tuple)*

```python
smact.lattice_parameters.b2(shannon_radius)
The lattice parameters of b2.
```  
**Parameters**

- `shannon_radius` *(list)* – The radii of the a,b ions

**Returns**

- Float values of lattics constants and angles (a, b, c, alpha, beta, gamma)

**Return type** *(tuple)*

```python
smact.lattice_parameters.bcc(covalent_radius)
The lattice parameters of the A2.
```  
**Parameters**

- `shannon_radius` *(list)* – The radii of the a ions

**Returns**

- Float values of lattics constants and angles (a, b, c, alpha, beta, gamma)

**Return type** *(tuple)*

```python
smact.lattice_parameters.bct(covalent_radius)
The lattice parameters of the bct.
```  
**Parameters**

- `shannon_radius` *(list)* – The radii of the a ions

**Returns**

- Float values of lattics constants and angles (a, b, c, alpha, beta, gamma)

**Return type** *(tuple)*

```python
smact.lattice_parameters.cubic_perovskite(shannon_radius)
The lattice parameters of the cubic perovskite structure.
```  
**Parameters**

- `shannon_radius` *(list)* – The radii of the a,b,c ions

**Returns**

- Float values of lattics constants and angles (a, b, c, alpha, beta, gamma)

**Return type** *(tuple)*

## 4.1. Submodules
The lattice parameters of the diamond.

**Parameters**
- **shannon_radius** *(list)* – The radii of the a ions

**Returns**
- float values of lattics constants and angles (a, b, c, alpha, beta, gamma)

**Return type**
- (tuple)

The lattice parameters of the A1.

**Args**:
- **shannon_radius** *(list)* : The radii of the a ions

**Returns**
- float values of lattics constants and angles (a, b, c, alpha, beta, gamma)

**Return type**
- (tuple)

The lattice parameters of the hcp.

**Parameters**
- **shannon_radius** *(list)* – The radii of the a ions

**Returns**
- float values of lattics constants and angles (a, b, c, alpha, beta, gamma)

**Return type**
- (tuple)

The lattice parameters of rocksalt.

**Parameters**
- **shannon_radius** *(list)* – The radii of the a,b ions

**Returns**
- float values of lattics constants and angles (a, b, c, alpha, beta, gamma)

**Return type**
- (tuple)

The stuffed wurtzite structure (e.g. LiGaGe) space group P63/mc.

**Parameters**
- **shannon_radius** *(list)* – The radii of the a,b,c ions

**Returns**
- float values of lattics constants and angles (a, b, c, alpha, beta, gamma)

**Return type**
- (tuple)

The lattice parameters of the wurtzite structure.

**Parameters**
- **shannon_radius** *(list)* – The radii of the a,b ions

**Returns**
- float values of lattics constants and angles (a, b, c, alpha, beta, gamma)

**Return type**
- (tuple)

The lattice parameters of Zinc Blende. 

**Args**:
- **shannon_radius** *(list)* : The radii of the a,b ions

**Returns**
- float values of lattics constants and angles (a, b, c, alpha, beta, gamma)

**Return type**
- (tuple)
4.1.8 smact.data_loader module

Provide data from text files while transparently caching for efficiency.

This module handles the loading of external data used to initialise the core smact.Element and smact.Species classes. It implements a transparent data-caching system to avoid a large amount of I/O when naively constructing several of these objects. It also implements a switchable system to print verbose warning messages about possible missing data (mainly for debugging purposes). In general these functions are used in the background and it is not necessary to use them directly.

\texttt{smact.data_loader.float_or_None}(x)

Cast a string to a float or to a None.

\texttt{smact.data_loader.lookup_element_data}(symbol, copy=True)

Retrieve tabulated data for an element.

The table “data/element_data.txt” contains a collection of relevant atomic data. If a cache exists in the form of the module-level variable _element_data, this is returned. Otherwise, a dictionary is constructed from the data table and cached before returning it.

\textbf{Parameters}

- \textbf{symbol} (str) – Atomic symbol for lookup
- \textbf{copy} (Optional(bool)) – if True (default), return a copy of the data dictionary, rather than a reference to the cached object – only used copy=False in performance-sensitive code and where you are certain the dictionary will not be modified!

\textbf{Returns} (dict) [Dictionary of data for given element, keyed by] column headings from data/element_data.txt.

\texttt{smact.data_loader.lookup_element_hhis}(symbol)

Retrieve the HHI_R and HHI_p scores for an element.

\textbf{Parameters} symbol – the atomic symbol of the element to look up.

\textbf{Returns}

(HHI_p, HHI_R)

Return None if values for the elements were not found in the external data.

\textbf{Return type} tuple

\texttt{smact.data_loader.lookup_element_oxidation_states}(symbol, copy=True)

Retrieve a list of known oxidation states for an element. The oxidation states list used is the SMACT default and most exhaustive list.

\textbf{Parameters}

- \textbf{symbol} (str) – the atomic symbol of the element to look up.
- \textbf{copy} (Optional(bool)) – if True (default), return a copy of the oxidation-state list, rather than a reference to the cached data – only use copy=False in performance-sensitive code and where the list will not be modified!

\textbf{Returns}

List of known oxidation states for the element.

Returns None if oxidation states for the Element were not found in the external data.

\textbf{Return type} list
smact.data_loader.lookup_element_oxidation_states_icsd(symbol, copy=True)

Retrieve a list of known oxidation states for an element. The oxidation states list used contains only those found in the ICSD (and judged to be non-spurious).

Parameters

- `symbol (str)` – the atomic symbol of the element to look up.
- `copy (Optional(bool))` – if True (default), return a copy of the oxidation-state list, rather than a reference to the cached data – only use copy=False in performance-sensitive code and where the list will not be modified!

Returns

List of known oxidation states for the element.

Return None if oxidation states for the Element were not found in the external data.

Return type list

smact.data_loader.lookup_element_oxidation_states_sp(symbol, copy=True)

Retrieve a list of known oxidation states for an element. The oxidation states list used contains only those that are in the Pymatgen default lambda table for structure prediction.

Parameters

- `symbol (str)` – the atomic symbol of the element to look up.
- `copy (Optional(bool))` – if True (default), return a copy of the oxidation-state list, rather than a reference to the cached data – only use copy=False in performance-sensitive code and where the list will not be modified!

Returns

List of known oxidation states for the element.

Return None if oxidation states for the Element were not found in the external data.

Return type list

smact.data_loader.lookup_element_shannon_radius_data(symbol, copy=True)

Retrieve Shannon radii for known states of an element.

Retrieve Shannon radii for known oxidation states and coordination environments of an element.

Parameters

- `symbol (str)` – the atomic symbol of the element to look up.
- `copy (Optional(bool))` – if True (default), return a copy of the data rather than a reference to the cached object --(dictionary,)--
- `use copy=False in performance-sensitive code and where (only) --(you)--`
- `are certain the dictionary will not be modified! (you)`

Returns

Shannon radii datasets.

Returns None if the element was not found among the external data.

Shannon radii datasets are dictionaries with the keys:

- `charge int charge`
- `coordination int coordination`
### crystal_radius

*float*

### ionic_radius

*float*

### comment

*str*

**Return type**  list

`smact.data_loader.lookup_element_sse2015_data(symbol, copy=True)`


**Parameters**

- **symbol** – the atomic symbol of the element to look up.
- **copy** – if True (default), return a copy of the data dictionary,
- **than a reference to a cached object -- only use(rather)–
- **in performance-sensitive code and where you are (copy=False)–
- **the dictionary will not be modified! (certain)–

**Returns**

- **SSE datasets for the element, or None** if the element was not found among the external data.

SSE datasets are dictionaries with the keys:

- **OxidationState**  *int*
- **SolidStateEnergy2015**  *float SSE2015*

**Return type**  list

`smact.data_loader.lookup_element_sse_data(symbol)`

Retrieve the solid-state energy (SSE) data for an element.

Taken from J. Am. Chem. Soc., 2011, 133 (42), pp 16852-16960, DOI: 10.1021/ja204670s

**Parameters**  *symbol* – the atomic symbol of the element to look up.

**Returns**

- **SSE datasets for the element, or None** if the element was not found among the external data.

SSE datasets are dictionaries with the keys:

- **AtomicNumber**  *int*
- **SolidStateEnergy**  *float SSE*
- **IonisationPotential**  *float*
- **ElectronAffinity**  *float*
- **MullikenElectronegativity**  *str*
- **SolidStateRenormalisationEnergy**  *float*

**Return type**  list

`smact.data_loader.lookup_element_sse_pauling_data(symbol)`

Retrieve Pauling SSE data
Retrieve the solid-state energy (SSEPaulting) data for an element from the regression fit when SSE2015 is plotted against Pauling electronegativity. Taken from J. Solid State Chem., 2015, 231, pp138-144, DOI: 10.1016/j.jssc.2015.07.037

Args: symbol (str) : the atomic symbol of the element to look up.

**Returns:** A dictionary containing the SSE2015 dataset for the element, or None if the element was not found among the external data.

```
smact.data_loader.set_warnings(enable=True)
```

Set verbose warning messages on and off.

In order to see any of the warnings, this function needs to be called _before_ the first call to the smact.Element() constructor.

Args: enable (bool) : print verbose warning messages.
CHAPTER 5

Indices and tables

- genindex
- modindex
- search
Python Module Index

S

smact, 11
smact.builder, 19
smact.data_loader, 23
smact.distorter, 19
smact.lattice, 20
smact.lattice_parameters, 21
smact.oxidation_states, 18
smact.properties, 15
smact.screening, 16
<table>
<thead>
<tr>
<th>A</th>
<th>are_eq() (in module smact), 13</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>b10() (in module smact.lattice_parameters), 21</td>
</tr>
<tr>
<td></td>
<td>b2() (in module smact.lattice_parameters), 21</td>
</tr>
<tr>
<td></td>
<td>band_gap_Harrison() (in module smact.properties), 15</td>
</tr>
<tr>
<td></td>
<td>basis_sites (smact.lattice.Lattice attribute), 20</td>
</tr>
<tr>
<td></td>
<td>bcc() (in module smact.lattice_parameters), 21</td>
</tr>
<tr>
<td></td>
<td>bct() (in module smact.lattice_parameters), 21</td>
</tr>
<tr>
<td></td>
<td>build_sub_lattice() (in module smact.distorter), 19</td>
</tr>
<tr>
<td>C</td>
<td>compound_electroneg() (in module smact.properties), 15</td>
</tr>
<tr>
<td></td>
<td>compound_probability() (smact.oxidation_states.Oxidation_state_probability_finder method), 18</td>
</tr>
<tr>
<td></td>
<td>coord_envs (smact.Element attribute), 12</td>
</tr>
<tr>
<td></td>
<td>coordination (smact.Species attribute), 13</td>
</tr>
<tr>
<td></td>
<td>covalent_radius (smact.Element attribute), 12</td>
</tr>
<tr>
<td></td>
<td>crustal_abundance (smact.Element attribute), 12</td>
</tr>
<tr>
<td></td>
<td>cubic_perovskite() (in module smact.builder), 19</td>
</tr>
<tr>
<td></td>
<td>cubic_perovskite() (in module smact.lattice_parameters), 21</td>
</tr>
<tr>
<td>D</td>
<td>diamond() (in module smact.lattice_parameters), 21</td>
</tr>
<tr>
<td></td>
<td>dipol (smact.Element attribute), 12</td>
</tr>
<tr>
<td>E</td>
<td>e_affinity (smact.Element attribute), 12</td>
</tr>
<tr>
<td></td>
<td>e_affinity (smact.Species attribute), 13</td>
</tr>
<tr>
<td></td>
<td>eig (smact.Element attribute), 12</td>
</tr>
<tr>
<td></td>
<td>eig (smact.Species attribute), 13</td>
</tr>
<tr>
<td></td>
<td>eig_s (smact.Element attribute), 12</td>
</tr>
<tr>
<td></td>
<td>Element (class in smact), 11</td>
</tr>
<tr>
<td></td>
<td>element_dictionary() (in module smact), 14</td>
</tr>
<tr>
<td></td>
<td>eneg_mulliken() (in module smact.properties), 16</td>
</tr>
<tr>
<td></td>
<td>eneg_states_test() (in module smact.screening), 16</td>
</tr>
<tr>
<td></td>
<td>eneg_states_test_alternate() (in module smact.screening), 16</td>
</tr>
<tr>
<td></td>
<td>eneg_states_test_threshold() (in module smact.screening), 16</td>
</tr>
<tr>
<td>F</td>
<td>fcc() (in module smact.lattice_parameters), 22</td>
</tr>
<tr>
<td></td>
<td>float_or_None() (in module smact.data_loader), 23</td>
</tr>
<tr>
<td>G</td>
<td>get_included_species() (smact.oxidation_states.Oxidation_state_probability_finder method), 19</td>
</tr>
<tr>
<td></td>
<td>get_inequivalent_sites() (in module smact.distorter), 20</td>
</tr>
<tr>
<td></td>
<td>get_sg() (in module smact.distorter), 20</td>
</tr>
<tr>
<td>H</td>
<td>hcp() (in module smact.lattice_parameters), 22</td>
</tr>
<tr>
<td></td>
<td>HHI_p (smact.Element attribute), 12</td>
</tr>
<tr>
<td></td>
<td>HHI_r (smact.Element attribute), 13</td>
</tr>
<tr>
<td>I</td>
<td>ionic_radius (smact.Species attribute), 13</td>
</tr>
<tr>
<td></td>
<td>ionpot (smact.Element attribute), 11</td>
</tr>
<tr>
<td></td>
<td>ionpot (smact.Species attribute), 13</td>
</tr>
<tr>
<td>L</td>
<td>Lattice (class in smact.lattice), 20</td>
</tr>
<tr>
<td></td>
<td>lattice_vector_calc() (smact.lattice.Lattice method), 21</td>
</tr>
<tr>
<td></td>
<td>lattices_are_same() (in module smact), 14</td>
</tr>
<tr>
<td></td>
<td>lookup_element_data() (in module smact.data_loader), 23</td>
</tr>
</tbody>
</table>
lookup_element_hhis() (in module smact.data_loader), 23
lookup_element_oxidation_states() (in module smact.data_loader), 23
lookup_element_oxidation_states_icsd() (in module smact.data_loader), 23
lookup_element_oxidation_states_sp() (in module smact.data_loader), 24
lookup_element_shannon_radius_data() (in module smact.data_loader), 24
lookup_element_sse2015_data() (in module smact.data_loader), 25
lookup_element_sse_data() (in module smact.data_loader), 25
lookup_element_sse_pauling_data() (in module smact.data_loader), 25
make_substitution() (in module smact.distorter), 20
mass (smact.Element attribute), 12
ml_rep_generator() (in module smact.screening), 17
name (smact.Element attribute), 11
name (smact.Species attribute), 13
neutral_ratios() (in module smact), 14
neutral_ratios_iter() (in module smact), 14
number (smact.Element attribute), 11
ordered_elements() (in module smact), 15
oxidation (smact.Species attribute), 13
Oxidation_state_probability_finder (class in smact.oxidation_states), 18
oxidation_states (smact.Element attribute), 12
oxidation_states (smact.lattice.Site attribute), 21
oxidation_states_icsd (smact.Element attribute), 12
oxidation_states_sp (smact.Element attribute), 12
pair_probability() (smact.oxidation_states.Oxidation_state_probability_finder method), 19
pauling_eneg (smact.Element attribute), 11
pauling_eneg (smact.Species attribute), 13
pauling_test() (in module smact.screening), 17
pauling_test_old() (in module smact.screening), 18
position (smact.lattice.Site attribute), 21

R
rocksalt() (in module smact.lattice_parameters), 22

S
set_warnings() (in module smact.data_loader), 26
shannon_radius (smact.Species attribute), 13
Site (class in smact.lattice), 21
smact (module), 11
smact.builder (module), 19
smact.data_loader (module), 23
smact.distorter (module), 19
smact.lattice (module), 20
smact.lattice_parameters (module), 21
smact.oxidation_states (module), 18
smact.properties (module), 15
smact.screening (module), 16
smact_filter() (in module smact.screening), 18
space_group (smact.lattice.Lattice attribute), 20
Species (class in smact), 13
SSE (smact.Element attribute), 12
SSEPauling (smact.Element attribute), 12
structurbericht (smact.lattice.Lattice attribute), 20
stuffed_wurtzite() (in module smact.lattice_parameters), 22
symbol (smact.Element attribute), 11
symbol (smact.Species attribute), 13

W
wurtzite() (in module smact.builder), 19
wurtzite() (in module smact.lattice_parameters), 22

Z
zincblende() (in module smact.lattice_parameters), 22