

Python libraries for atomistic modeling and materials analysis

Andriy Zhugayevych (Instructor)

Sergey Levchenko (Co-Instructor)


Sergei Tretiak (Co-Instructor)

Dmitry Aksyonov (Co-Instructor)

Vasilii Vasilchenko (TA)

Approaches to materials modeling

Manual approach

- typing input files by hands
 - copy files to cluster and back by hands
 - (using ssh)
 - getting output data by opening files in text editors and copy/paste numbers
- 
- analysis of output data using Excel and Origin-like programs
 - relying on programs with graphical interface (in some cases its is efficient but in many cases you need to press many buttons every time to accomplish routine tasks)

Slow, easy to make a mistake, boring for repeating tasks

Approaches to materials modeling

Scripting

- all input files are created by scripts (only small changes are needed) files
 - file transfer is automatic
 - comprehensive and very flexible analysis specific for your needs easy reuse of code for routine tasks
-
- Information can be stored in database
 - Takes more time in the beginning, but then is much faster
 - Error-proof
 - More interesting



Libraries for atomistic materials modeling

- *ASE* <https://wiki.fysik.dtu.dk/ase/> a set of tools and Python modules for setting up, manipulating, running, visualizing and analyzing atomistic simulations.
- *Pymatgen* <https://pymatgen.org/> Python library for materials analysis.
- *SIMAN* <https://github.com/dimonaks/siman> Skoltech Python library for energy materials modeling.
- *MolMod* <http://zhugayevych.me/maple/MolMod/index.htm> Skoltech Maple library for molecules and materials modeling.
- *VTST* <https://theory.cm.utexas.edu/vtsttools/scripts.html> a set of Perl/Python scripts to perform common tasks to help with VASP calculations
- *and others* <https://google.ru> try to google before writing by yourself

How to use Python?

- Install jupyter and use either jupyter notebook or jupyter-lab for IPython
 - <https://jupyter.org/install.html>
 - Good for fast prototyping, education
- Use one of the IDE
 - Sublime text 3, PyCharm, etc
 - More flexible and powerful
 - For development of own tools, for production calculations

ASE - atomic simulation environment

- <https://wiki.fysik.dtu.dk/ase/>, free to use
- Good documentation with tutorials
- Great DFT book with examples written in ASE
<http://kitchingroup.cheme.cmu.edu/dft-book/dft.html>
- Interfaces for most popular modeling simulation codes for DFT, MD, etc (>30):



Requirements

- **Python** 3.6 or newer
- **NumPy** 1.11 or newer (base N-dimensional array package)
- **SciPy** 0.18 or newer (library for scientific computing)

Optional but strongly recommended:

- **Matplotlib** 2.0.0 or newer for plotting
- `tkinter` for `ase.gui`

Optional:

- **Flask** for `ase.db` web-interface
- **pytest** 3.6.1 or newer for running tests
- **pytest-xdist** 1.22.1 or newer for running tests in parallel
- **spglib** for certain symmetry-related features

Installation

- **Linux**

python-ase package or

```
pip3 install --upgrade --user ase
```

- **Mac OS**

```
brew install python
```

- **Windows:**

Use Anaconda

<https://www.anaconda.com/products/individual> and then follow instructions for installing python packages

Features of ASE

- Main features
 - The Atoms object
 - Set up molecules, crystals, surfaces and more using provided modules augmented by scripting
 - Use GUI to visualize structures
 - Read and write many formats (xyz, cube, xsf, cif, pdb, ...)
 - Call external codes from Python using the **ASE Calculator interface**
- Calculator object
 - A calculator can take Atoms as input and produce energies and forces as output
 - Most calculators call an external DFT code
 - Some calculators: Gaussian, VASP, Mopac, Abinit, EMT, GPAW, NWChem, There are 30+ calculators

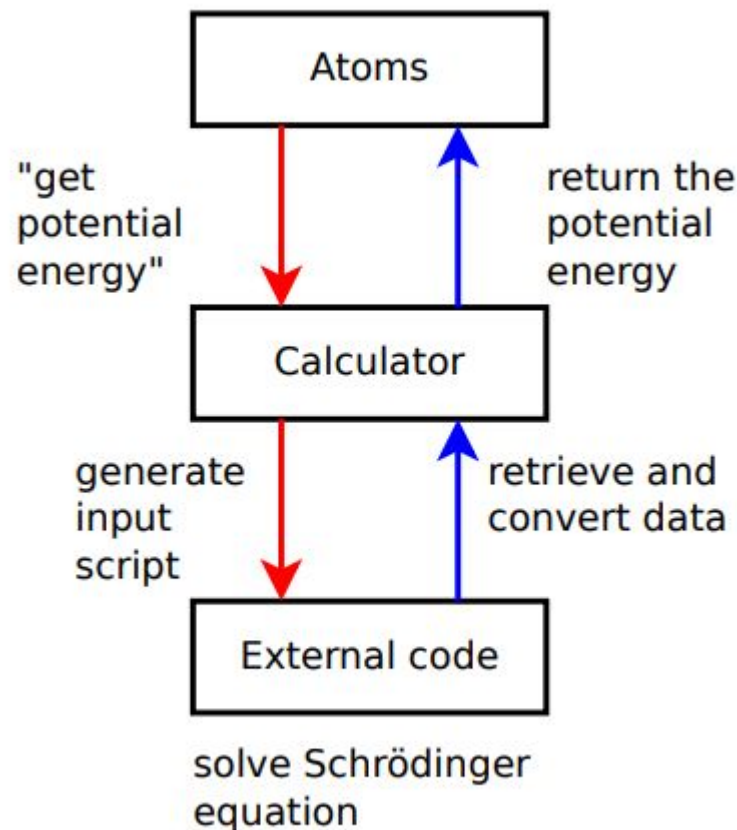
Classes for setting up structures

- `ase.build.molecule`
 - *G2 molecule test set*
- `ase.build.bulk`
- `ase.spacegroup.crystal`
- `ase.lattice.cubic, tetragonal, ...`
- `ase.build.surface`

Calculators

The **calculators** can be divided in four groups:

- **Asap**, **DFTK**, **GPAW**, and **Hotbit** **have their own native ASE interfaces.**
- ABINIT, AMBER, CP2K, CASTEP, deMon2k, DFTB+, ELK, EXCITING, FHI-aims, FLEUR, GAUSSIAN, Gromacs, LAMMPS, MOPAC, NWChem, Octopus, ONETEP, psi4, Q-Chem, Quantum ESPRESSO, SIESTA, TURBOMOLE and VASP, **have Python wrappers in the ASE package, but the actual FORTRAN/C/C++ codes are not part of ASE.**
- Pure python implementations included in the ASE package: EMT, EAM, Lennard-Jones and Morse.
- **Calculators** that wrap others, included in the ASE package ([see here](#))



Basic properties

- `atoms.get_potential_energy()`
- `atoms.get_forces()`
- `atoms.get_stress()`
- `atoms.get_dipole_moment()`
- `atoms.get_positions()`
- `calc.get_eigenvalues()`
- `calc.get_occupations()`
- `calc.get_pseudo_density()`
- `calc.get_ibz_k_points()`

Optimization algorithms

- Gradient-based structure optimizations with constraints
- Global optimizations: minima/basin hopping, genetic algorithm
- Molecular dynamics with different controls
- Saddle-point searches (for transition states)
- Vibrational modes (molecules and phonons)

Let's try some examples!

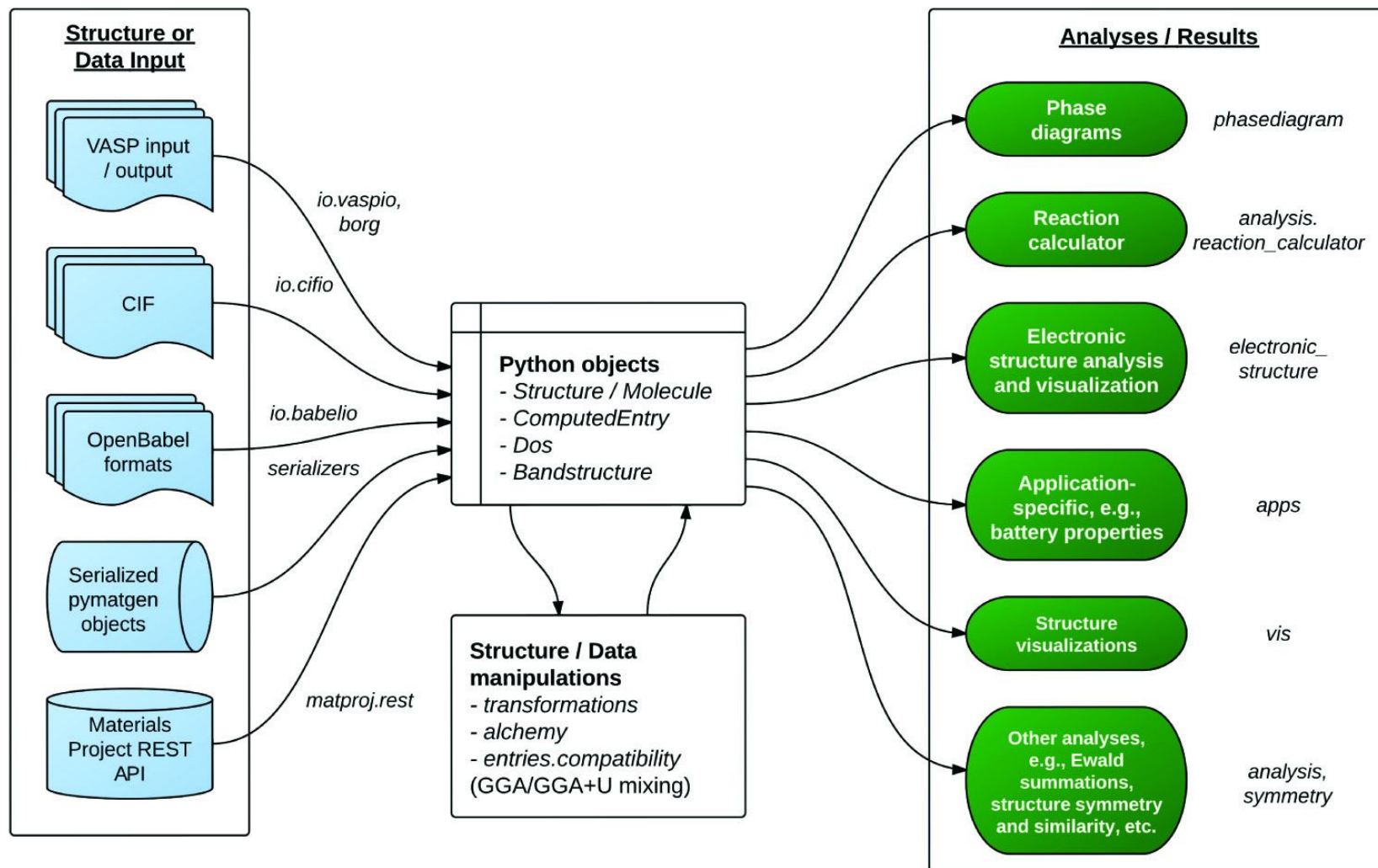
- We have special server with Jupyter Notebook (IPython):
<http://10.30.99.214:8000/>
- If you need access please write to d.aksenov@skoltech.ru The server will be available from November 17th (next wednesday). You need 10.30.99.214 address be added to your VPN (make a request now)
- Or try installing it on your personal computer

Pymatgen

Pymatgen (Python Materials Genomics) is a robust, open-source Python library for materials analysis. These are some of the main features:

- Highly flexible classes for the representation of Element, Site, Molecule, Structure objects.
- Extensive input/output support, including support for VASP (<http://cms.mpi.univie.ac.at/vasp/>), ABINIT (<http://www.abinit.org/>), CIF, Gaussian, XYZ, and many other file formats.
- Powerful analysis tools, including generation of phase diagrams, Pourbaix diagrams, diffusion analysis, reactions, etc.
- Electronic structure analysis, such as density of states and band structure.
- Integration with the Materials Project REST API, Crystallography Open Database and other external data sources.

Basic workflow with pymatgen



Installation

<https://pymatgen.org/installation.html>

```
pip3 install pymatgen
```

the requirements are similar to ase

to enable visualisation of structures from python

```
pip3 install chemview
```

```
pip3 install widgetsnbextension ipywidgets
```

```
sudo jupyter nbextension enable --py --sys-prefix  
widgetsnbextension
```

Basic objects

- `pymatgen.core.periodic_table`: Everything begins here, where the Element and Specie (Element with an oxidation state) objects are defined. Unlike typical implementations, pymatgen's Element object is rich, which means that each Element contains many useful properties associated with it, including atomic numbers, atomic masses, melting points, boiling points, just to name a few.
- `pymatgen.core.lattice`: This module defines a Lattice object, which essentially defines the lattice vectors in three dimensions. The Lattice object provides convenience methods for performing fractional to cartesian coordinates and vice versa, lattice parameter and angles computations, etc
- `pymatgen.core.sites`: Defines the Site and PeriodicSite objects. A Site is essentially a coordinate point containing an Element or Specie. A PeriodicSite contains a Lattice as well.
- `pymatgen.core.structure`: Defines the Structure and Molecule objects. A Structure and Molecule are simply a list of PeriodicSites and Site respectively.
- `pymatgen.core.composition`: A Composition is simply a mapping of Element/Specie to amounts.

Creating structures and molecules

```
1 from pymatgen import Lattice, Structure, Molecule
2 #Silicon
3 coords = [[0, 0, 0], [0.75,0.5,0.75]]
4 lattice = Lattice.from_parameters(a=3.84, b=3.84,
   c=3.84, alpha=120, beta=90, gamma=60)
5 struct = Structure(lattice, ["Si", "Si"], coords)
6
7 #Methane molecule
8 coords = [[0.000000, 0.000000, 0.000000],
9           [0.000000, 0.000000, 1.089000],
10          [1.026719, 0.000000, -0.363000],
11          [-0.513360, -0.889165, -0.363000],
12          [-0.513360, 0.889165, -0.363000]]
13 methane = Molecule(["C", "H", "H", "H", "H"], coords)
```

Reading and writing structures

```
1. # Read a POSCAR and write to a CIF.
2. structure = Structure.from_file("POSCAR")
3. structure.to(filename="CsCl.cif")
4.
5. # Read an xyz file and write to a Gaussian Input file.
6. methane = Molecule.from_file("methane.xyz")
7. methane.to(filename="methane.gjf")
```

```
1. from pymatgen.io.vasp import Poscar
2. poscar = Poscar.from_file("POSCAR")
3. structure = poscar.structure
```

Input/output packages and module

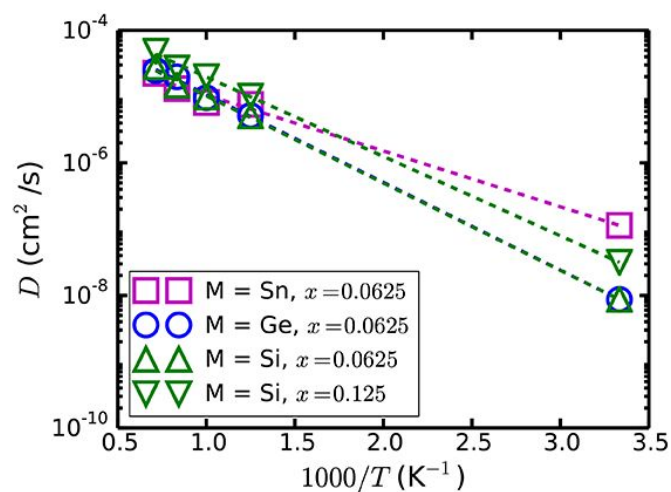
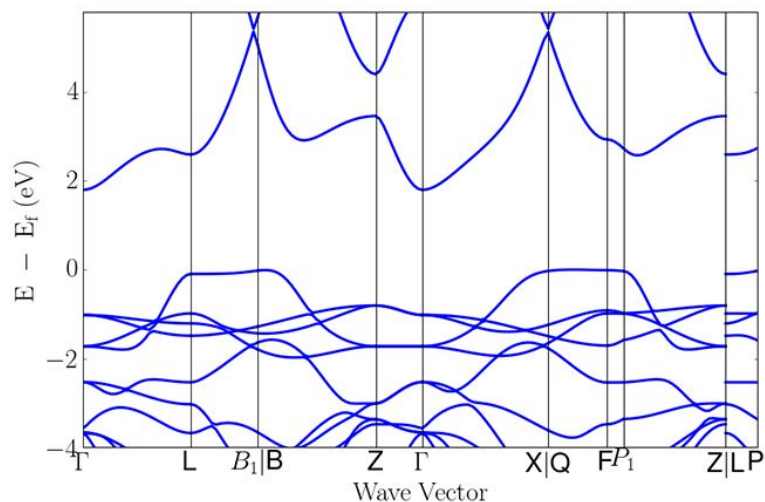
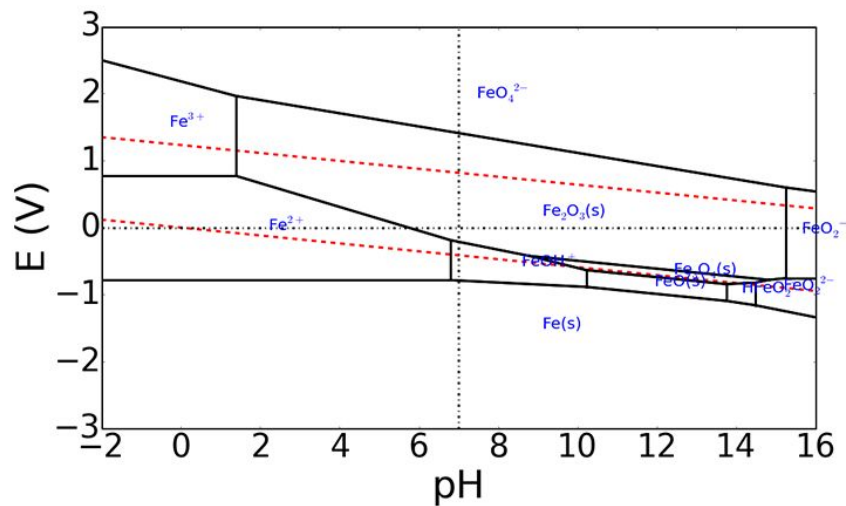
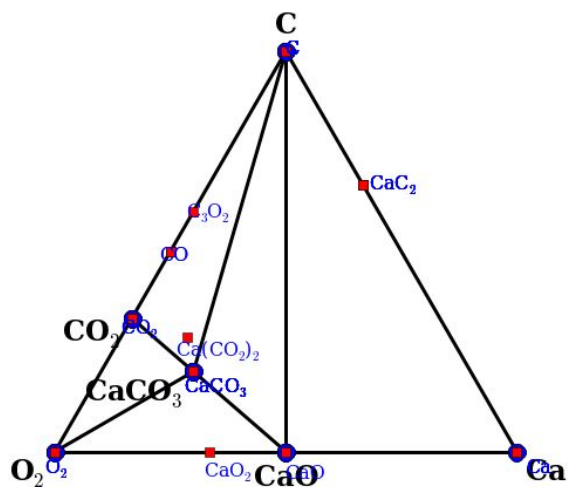
- [pymatgen.io.abinit package](#)
- [pymatgen.io.lammps package](#)
- [pymatgen.io.vasp package](#)
- [pymatgen.io.cif module](#) - read and write cif files
- [pymatgen.io.xyz module](#) - read and write xyz files
- [pymatgen.io.ase module](#) - interface with ase
- [pymatgen.io.atat module](#) - cluster expansion
- [pymatgen.io.babel module](#) - open Babel formats
- [pymatgen.io.gaussian module](#)
- [pymatgen.io.phonopy module](#) - phonons

Structure transformations

The `pymatgen.transformations` package is the standard package for performing transformations on structures.

- `pymatgen.transformations.site_transformations` module
- `pymatgen.transformations.standard_transformations` module
- `pymatgen.transformations.advanced_transformations` module
- `pymatgen.transformations.defect_transformations` module ([defect](#))

Analysis with pymatgen



Analysis tools

- Defects
 - create and analyze point defects
 - defects corrections for DFT calculations
 - defect concentration based on composition, temperature, and defect energies
 - defect phase diagrams
 - [create grain boundaries](#)
- Diffraction
 - calculation of diffraction patterns, neutron, TEM, XRD
- Elasticity
- Magnetism
- [pymatgen.analysis.adsorption module](#)
- [pymatgen.analysis.diffusion_analyzer module](#)
- [pymatgen.analysis.eos module](#) - equation of states
- [pymatgen.analysis.ewald module](#) - calculate electrostatic energy

Analysis tools

- [pymatgen.analysis.interface_reactions module](#)
- [pymatgen.analysis.molecule_matcher module](#)
- [pymatgen.analysis.molecule_structure_comparator module](#)
- [pymatgen.analysis.phase_diagram module](#)
- [pymatgen.analysis.piezo_sensitivity module](#)
- [pymatgen.analysis.pourbaix_diagram module](#) - potential/pH diagram

Analysis tools

- [pymatgen.analysis.quasiharmonic module](#) Quasi-harmonic Debye approximation
- [pymatgen.analysis.reaction_calculator module](#) chemical reactions
- [pymatgen.analysis.structure_analyzer module](#) Voronoi, coordination, type
- [pymatgen.analysis.structure_matcher module](#)
- [pymatgen.analysis.surface_analysis module](#) surface and adsorption related quantities
- [pymatgen.analysis.transition_state module](#) Henkelman's Transition State Analysis
- [pymatgen.analysis.wulff module](#) WulffShape construction

Set up for vasp and database for pymatgen

put potentials into `~/vasp_potentials/POT_GGA_PAW_PBE/`

run commands:

```
pmg config --add PMG_DEFAULT_FUNCTIONAL PBE_54
```

```
pmg config --add PMG_VASP_PSP_DIR  
/home/aksenov/vasp_potentials/
```

```
pmg config --add PMG_MAPI_KEY key_obtained on  
materialsproject.org
```

or edit `~/.pmgrc.yaml` - configuration file for pymatgen

Tutorials

<https://pymatgen.org/pymatgen.html> - see here for full capabilities

<http://matgenb.materialsvirtuallab.org/> - a collection of pymatgen tutorials

Let's try some examples
symmetries

Thank you for attention!

Sources

<http://dcwww.camd.dtu.dk/~askhl/files/python-ase-slides.pdf>