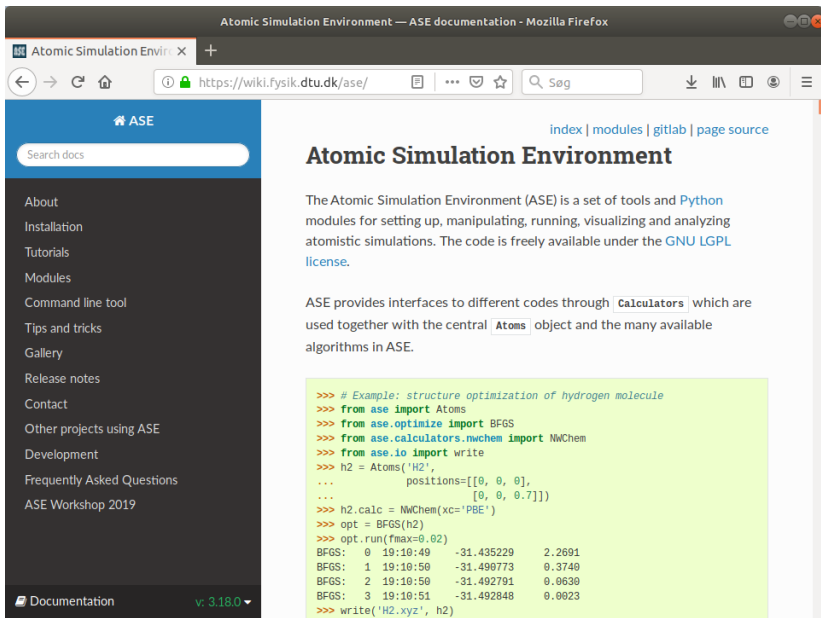


# ASE: The Atomic Simulation Environment

Ask Hjorth Larsen  
asklarsen@gmail.com

Simune Atomistics, San Sebastián, Spain  
and  
Nano-bio Spectroscopy Group  
and ETSF Scientific Development Centre  
Universidad del País Vasco UPV/EHU

August 30, 2019



Atomic Simulation Environment — ASE documentation - Mozilla Firefox

Atomic Simulation Environ X +

https://wiki.fysik.dtu.dk/ase/

ASE

Search docs

About

Installation

Tutorials

Modules

Command line tool

Tips and tricks

Gallery

Release notes

Contact

Other projects using ASE

Development

Frequently Asked Questions

ASE Workshop 2019

Documentation v. 3.18.0

[index](#) | [modules](#) | [gitlab](#) | [page source](#)

## Atomic Simulation Environment

The Atomic Simulation Environment (ASE) is a set of tools and [Python](#) modules for setting up, manipulating, running, visualizing and analyzing atomistic simulations. The code is freely available under the [GNU LGPL license](#).

ASE provides interfaces to different codes through [Calculators](#) which are used together with the central [Atoms](#) object and the many available algorithms in ASE.

```
>>> # Example: structure optimization of hydrogen molecule
>>> from ase import Atoms
>>> from ase.optimize import BFGS
>>> from ase.calculators.nwchem import NwChem
>>> from ase.io import write
>>> h2 = Atoms('H2',
...           positions=[[0, 0, 0],
...                      [0, 0, 0.7]])
>>> h2.calc = NwChem(xc='PBE')
>>> opt = BFGS(h2)
>>> opt.run(fmax=0.02)
BFGS:  0 19:10:49 -31.435229  2.2691
BFGS:  1 19:10:50 -31.490773  0.3740
BFGS:  2 19:10:50 -31.492791  0.0630
BFGS:  3 19:10:51 -31.492848  0.0023
>>> write('H2.xyz', h2)
```

# ASE: A Python library for working with atoms

## Main features

- ▶ The `Atoms` object
- ▶ Set up molecules, crystals, surfaces and more using provided modules augmented by scripting
- ▶ Read and write files (xyz, cube, xsf, cif, pdb, ...)
- ▶ Call external codes from Python using `Calculator`
- ▶ Visualisation: GUI, command-line tools

## Calculator

- ▶ Interface for calculating things: Energy, forces, etc.
- ▶ Most calculators call an external DFT code
- ▶ Calculators: GPAW, NWChem, Abinit, FHI-aims, VASP, ...

## Calculations written as Python scripts!

```
from ase import Atoms
from ase.optimize import BFGS
from gpaw import GPAW

system = Atoms('H2O', positions=[[ -1, 0, 0],
                                  [ 1, 0, 0],
                                  [ 0, 0, 1]])

system.center(vacuum=3.0)
system.calc = GPAW(mode='lcao', basis='dzp')

opt = BFGS(system,
            trajectory='opt.traj',
            logfile='opt.log')
opt.run(fmax=0.05)
```

## Scripting electronic structure calculations

- ▶ Workflow: Replace much ad-hoc scripting with more systematic tools
- ▶ No need for algorithms to be implemented *within* computational/DFT codes
- ▶ Hack or write your own algorithm!
- ▶ Batteries included: `ase.build.molecule`, `ase.build.bulk`, `ase.lattice`, `ase.io`, ...

Skim features on web page!

<https://wiki.fysik.dtu.dk/ase/>

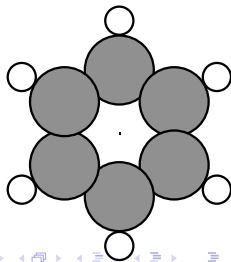
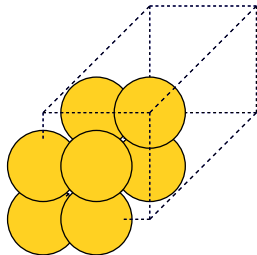
## Build and view structures

```
from ase import Atoms
from ase.visualize import view

a = 2.04
gold = Atoms('Au', pbc=True,
             cell=[[0, a, a],
                  [a, 0, a],
                  [a, a, 0]])

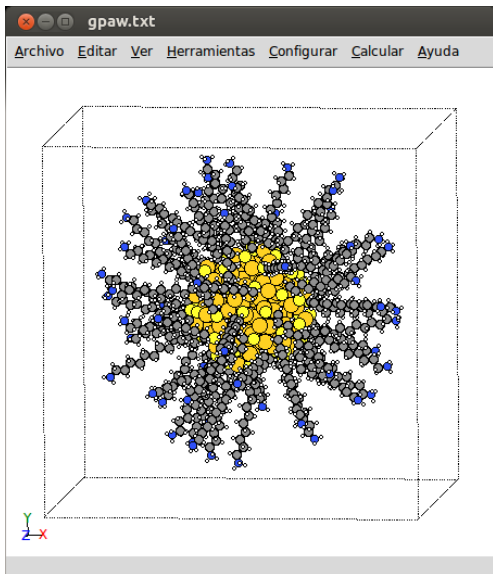
print(gold)
view(gold.repeat((2, 2, 2)))

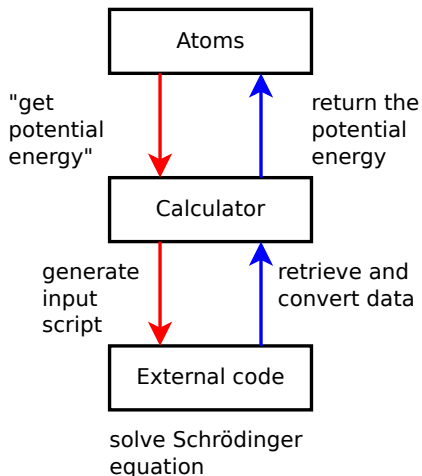
from ase.build import molecule
view(molecule('C6H6'))
```



## Try the ASE GUI

- ▶ Run ase gui  
(previously: ase-gui)
- ▶ Build nanoparticle or something else
- ▶ Select, move atoms  
(Ctrl+M)
- ▶ Save to your favourite format





## Interface through file I/O

- ▶ ASE creates inputfile, runs programme (see figure)

## Calculator daemon

- ▶ Calculator runs in background
- ▶ Read/write using sockets, pipes

## Direct linking

- ▶ Everything within one process  
→ efficient *and* nice
- ▶ Also rather complicated



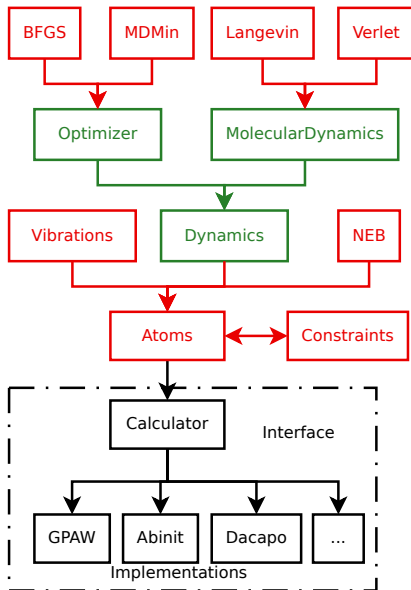
# Calculators

## Basic properties

- ▶ `atoms.get_potential_energy()`
- ▶ `atoms.get_forces()`
- ▶ `atoms.get_stress()`
- ▶ `atoms.get_dipole_moment()`

## Electronic structure calculators

- ▶ `calc.get_eigenvalues()`
- ▶ `calc.get_occupations()`
- ▶ `calc.get_pseudo_density()`
- ▶ `calc.get_ibz_k_points()`



## A bit of history

- ▶ Started as an object-oriented Python interface to the old ultrasoft pseudopotential planewave code Dacapo
- ▶ S.R. Bahn, K.W. Jacobsen, “An object-oriented scripting interface to a legacy electronic structure code”. *Computing in Science & Engineering*, 4(3):56–66, 2002.
- ▶ BDFL: Jens Jørgen Mortensen, DTU Physics
- ▶ Very many contributors
- ▶ Now has interfaces to many codes, and many tools.
- ▶ New reference paper: A.H. Larsen, J.J. Mortensen *et al.*, “The Atomic Simulation Environment – A Python library for working with atoms”: 2017 *J. Phys. Condens. Matter* 29 273002 (Available as Psi-k Highlight of the Month, January 2017)
- ▶ <https://wiki.fysik.dtu.dk/ase/>