The Atomic Simulation Environment

Ask Hjorth Larsen
and the ASE development team

Abinit developer meeting

April 17, 2013

The atomic simulation environment

- Collection of tools for atomistic simulations
- ASE facilitates calculations through other codes
- Written in Python; used by writing Python scripts
- Free software. GNU LGPL v2.1+.
- `https://wiki.fysik.dtu.dk/ase/`
- CAMPOS projects: `https://wiki.fysik.dtu.dk/`

Original citation

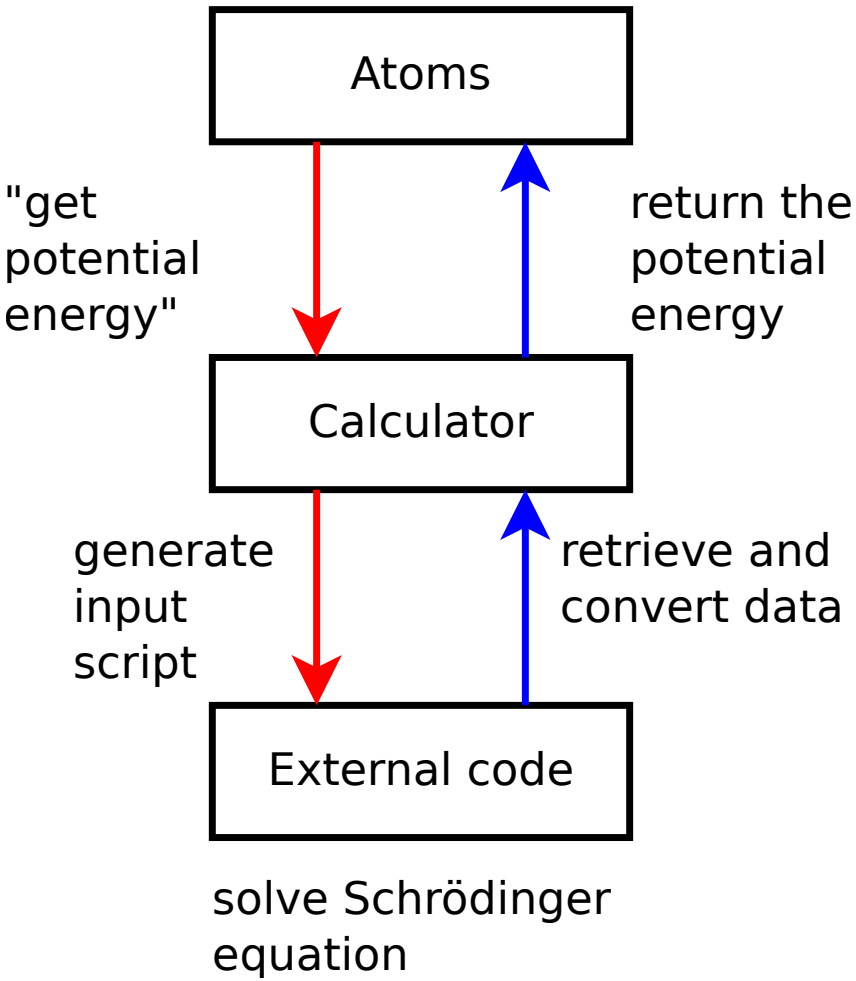
- S. R. Bahn and K. W. Jacobsen
  An object-oriented scripting interface to a legacy electronic structure code
  Comput. Sci. Eng., Vol. **4**, 56-66, 2002

Main functionality of ASE

- Defining structures (molecules, surfaces, ...)
- File I/O with many formats
- Algorithms for structure optimizations and dynamics
- Uniform calculator interfaces to external codes

Calculations as Python scripts

- Full programming environment
- Everything is scriptable
- Access to useful libraries (numpy, matplotlib, scipy, ...)



Interface through file I/O

- ASE creates inputfile, runs programme (see figure)

Calculator daemon

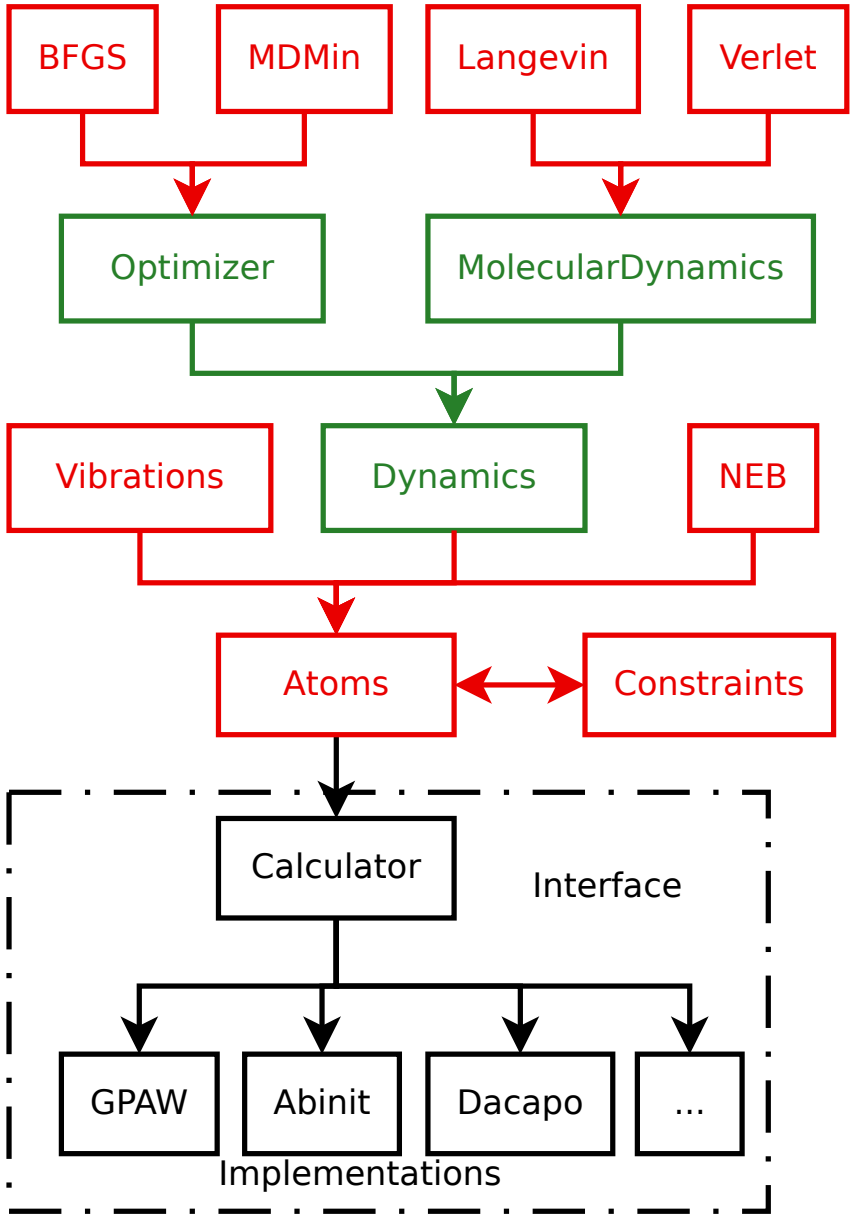
- Calculator runs in background
- Read/write using sockets

Direct linking

- Everything within one process $\rightarrow$ efficient *and* nice
- Also rather complicated

High-level algorithms

- Structure optimization (BFGS, FIRE, MDMin, minima hopping)
- Minimum-energy path (NEB)
- Molecular dynamics (Verlet, Nosé–Hoover, Langevin, Berendsen, ...)
- Vibrational analysis
- Electron transport


BFGS
MDMin
Langevin
Verlet
Optimizer
MolecularDynamics
Vibrations
Dynamics
NEB
Atoms
Constraints
Calculator
Interface
GPAW
Abinit
Dacapo
...
Implementations

ASE calculators


Also: Gaussian, Mopac

Demonstration: calculate energy and structure

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

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

system.center(vacuum=3.0)
calc = GPAW(mode='lcao', basis='dzp', h=.2)
system.set_calculator(calc)
#E = system.get_potential_energy()

opt = BFGS(system, trajectory='opt2.traj')
opt.run(fmax=.05)
```

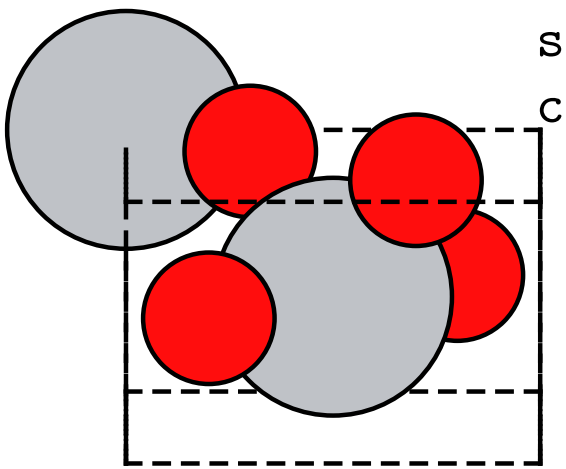
Demonstration: Abinit through ASE

```
from ase import Atoms
from ase.optimize import BFGS
from ase.calculators.abinit import Abinit
from ase.units import Ry

system = Atoms('H2O',
               positions=[[-1,0,0],[1,0,0],[0,0,1]])
system.center(vacuum=3.0)
calc = Abinit(ecut=25 * Ry, toldff=1e-4)
system.set_calculator(calc)
opt = BFGS(system, trajectory='opt2.traj')
opt.run(fmax=.05)
```

Bulk rutile

```
from ase.lattice.spacegroup import crystal
a = 4.6
c = 2.95
rutile = crystal(['Ti', 'O'],
                 basis=[(0, 0, 0),
                        (0.3, 0.3, 0.0)],
                 spacegroup=136,
                 cellpar=[a, a, c, 90, 90, 90])
```



The ASE team—or rather parts of it

Andrew Peterson
Anthony Goodrow
Ask Hjorth Larsen
Carsten Rostgaard
Christian Glinsvad
David Landis
Elvar Örn Jónsson
Felix Hanke
George Tritsaris
Heine Anton Hansen
Ivano Castelli
Jakob Blomquist
Jakob Schiøtz
Janne Blomqvist
Janosch Michael Rauba
Jens Jørgen Mortensen
Jesper Friis
Jesper Kleis
Jingzhe Chen
John Kitchin
Jonas Bjork
Jon B. Maronsson
Jussi Enkovaara
Karsten Wedel Jacobsen
Kristen Kaasbjerg
Lars Grabow
Marcin Dulak
Marco Vanin
Markus Kaukonen
Mattias Slabanja
Michael Walter
Mikkel Strange
Poul Georg Moses
Tao Jiang
Thomas Olsen
Troels Kofoed Jacobsen

Lead developer: Jens Jørgen Mortensen

Group leader: Karsten Wedel Jacobsen

Conclusion

- Pluggable calculators
- Python scripting
- Structure generation
- Free / open source software

Links and info

- CAMPOS: https://wiki.fysik.dtu.dk/
- ASE: https://wiki.fysik.dtu.dk/ase/
- Mailing list: ase-users@listserv.fysik.dtu.dk
- IRC: #gpaw on irc.freenode.net
- Python: http://www.python.org/

Thanks for listening