The Atomic Simulation Environment: Overview and developments

Ask Hjorth Larsen
asklarsen@gmail.com

Nano-bio Spectroscopy Group
and ETSF Scientific Development Centre
Universidad del País Vasco UPV/EHU

FHI-aims developers' and users' meeting 2018

The Atomic Simulation Environment

ASE is a free (LGPLv2.1+) toolkit to set up and control atomistic calculations in a fully scripted environment using Python.

Main features

- The `Atoms` object: A collection of atoms
- Calculators: Capable of calculating energies and forces of atoms, often using an external code as backend
- Algorithms working with atoms/calculators: Structure optimization, molecular dynamics, basin hopping, minima hopping, nudged elastic band, ...
- Many utilities: Build crystals, surfaces, ...
- Read/write structures in many formats
- Also: GUI, command-line utilities

Example: Structure optimization with GPAW

```
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)
```

Example: Structure optimization with Espresso

```
from ase import Atoms
from ase.optimize import BFGS
from ase.calculators.espresso import Espresso

system = Atoms('H2O', positions=[[-1, 0, 0],
                                 [1, 0, 0],
                                 [0, 0, 1]])
system.center(vacuum=3.0)
system.calc = Espresso(
    ecutwfc=40., pseudo_dir='.', tprnfor=True,
    pseudopotentials={'H': 'H_ONCV_PBE-1.0.upf',
                      'O': 'O_ONCV_PBE-1.0.upf'})
opt = BFGS(system, trajectory='opt.traj',
           logfile='opt.log')
opt.run(fmax=0.05)
```

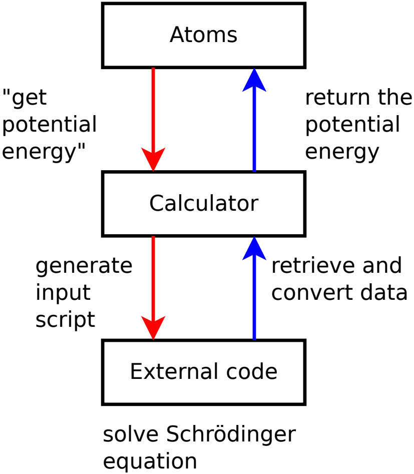
Example: Structure optimization with FHI-aims

```
from ase import Atoms
from ase.optimize import BFGS
from ase.calculators.aims import Aims

system = Atoms('H2O', positions=[[-1, 0, 0],
                                 [1, 0, 0],
                                 [0, 0, 1]])
species_dir = '/home/aimsuser/src/fhi-aims.171221_1/species
system.calc = Aims(xc='LDA',
                   command='aims',
                   species_dir=species_dir,
                   compute_forces=True)

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

Codes with ASE calculators

ASAP
Abinit
Atomistica
CP2K
Castep
DFTB+
Dacapo
ELK
Exciting
FHI-aims
Fleur
GPAW
Gaussian
Gromacs
Hotbit
JDFTx
LAMMPS
MOPAC
NWChem
Octopus
OpenKIM
OpenMX
OpenMX
QUIP
Quantum Espresso
Siesta
Turbomole
VASP
deMon
matscipy

Communication with external codes


Atoms
"get potential energy"
return the potential energy
Calculator
generate input script
retrieve and convert data
External code
solve Schrödinger equation

1) One process per calculation

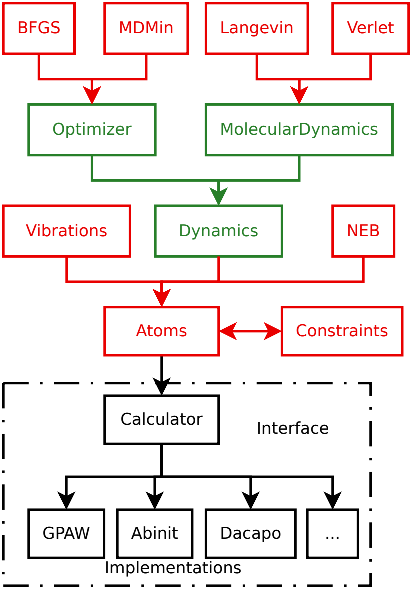
- ASE creates inputfile, runs code in subprocess (see figure)

2) Persistent subprocess

- External process remains alive over multiple calculations
- IO uses sockets, pipes or files

3) Within same process

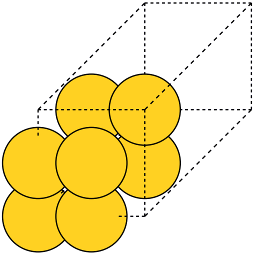
- Direct access to functions, data
- Requires Python bindings

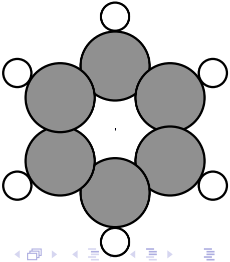


BFGS
MDMin
Langevin
Verlet
Optimizer
MolecularDynamics
Vibrations
Dynamics
NEB
Atoms
Constraints
Calculator
Interface
GPAW
Abinit
Dacapo
...
Implementations

Build and view structures

```python
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'))
```



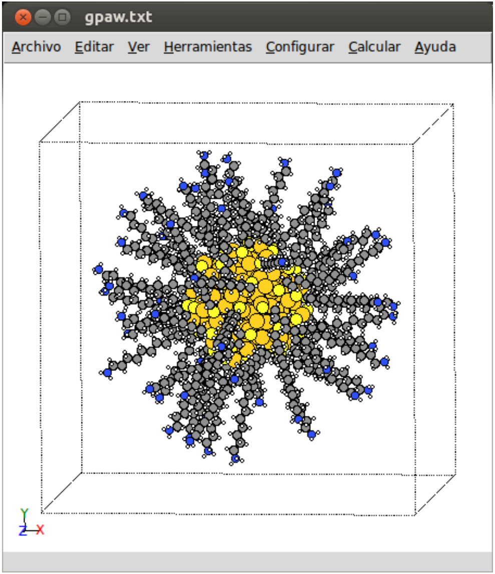


Command-line tools

- Type `ase help`
- Different subcommands: `build`, `gui`, `nomad-upload`, . . .

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



Database

- Store atoms objects alongside auxiliary information
- Backends: JSON, SQLite, PostgreSQL
- Good for everyday calculation workflow

```
import numpy as np
from ase.build import bulk
from ase.db import connect
from ase.calculators.emt import EMT

with connect('database.db') as con:
    for sym in ['Al', 'Cu', 'Pd', 'Ag', 'Au', 'Pd']:
        atoms = bulk(sym)
        atoms.calc = EMT()
        cell0 = atoms.cell.copy()
        for scale in np.linspace(0.95, 1.05, 11):
            atoms.cell = cell0 * scale
            atoms.get_potential_energy()
            # Add user-defined columns:
            con.write(atoms, sym=sym, scale=scale)
```

Database

Retrieve and print data from database

```
from ase.db import connect

con = connect('database.db')
for row in con.select(sym='Au'):
    atoms = row.toatoms()
    print('{}: {:8.3f} {:8.3f} {:8.3f}'
          .format(row.id, row.scale,
                  atoms.get_volume(), row.energy))
```

- Also available: ase db command line tool
- Display and manipulate databases

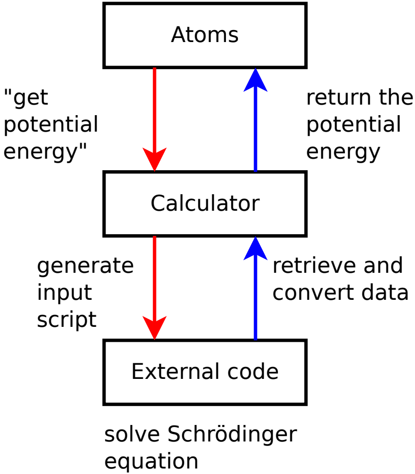
More on ASE

- 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.
- New reference paper: A.H. Larsen, J.J. Mortensen *et al.*, 2017 *J. Phys. Condens. Matter* **29** 273002, 2017. "The Atomic Simulation Environment – A Python library for working with atoms". (Also available as Psi-k Highlight of the Month, January 2017)

Development

- `https://gitlab.com/ase/ase`
- BDFL: Jens Jørgen Mortensen, DTU Physics
- Very large number of contributors ($\sim$ 150 committers)
- Many modules are maintained by different contributors
- “Scratch your own itch”

Communication schemes (again)



Atoms

"get potential energy"

return the potential energy

Calculator

generate input script

retrieve and convert data

External code

solve Schrödinger equation

Persistent calculators

- Standard “File I/O” calculator processes will end after each calculation
- Inefficient because: No reuse of density/wavefunctions, no wavefunction extrapolation
- With some codes the process can persist over multiple calculations, providing some mechanism to communicate new positions/forces
- Typical mechanisms are sockets or pipes

Support for i-PI socket protocol in ASE

About i-PI

- Molecular dynamics with many codes over socket interface
- `http://ipi-code.org/`
- Ceriotti, More, Manolopoulos, Comp. Phys. Comm. **185**, 1019–1026 (2014)
- Implements MD algorithms, “drivers”, server
- Interfaces to several codes using sockets
- Client codes: Quantum Espresso, FHI-aims, Siesta, DFTB+, Yaff, cp2k, Lammps, ASE, GPAW
- ASE now implements a server that can use the i-PI protocol

Sockets in ASE using the i-PI protocol

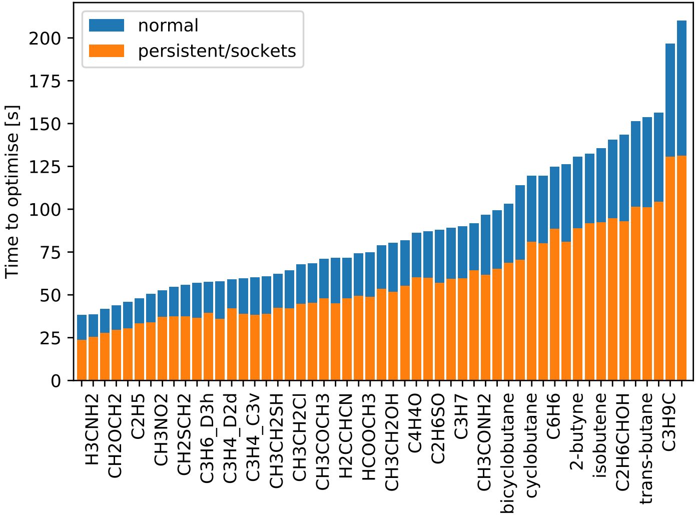
- ASE implements both server and client for the i-PI protocol
- The server can be used as a Calculator, `ase.calculators.socketio.SocketIOCalculator`
- The SocketIOCalculator wraps another (ordinary) calculator which is responsible for launching the client

Socket protocol

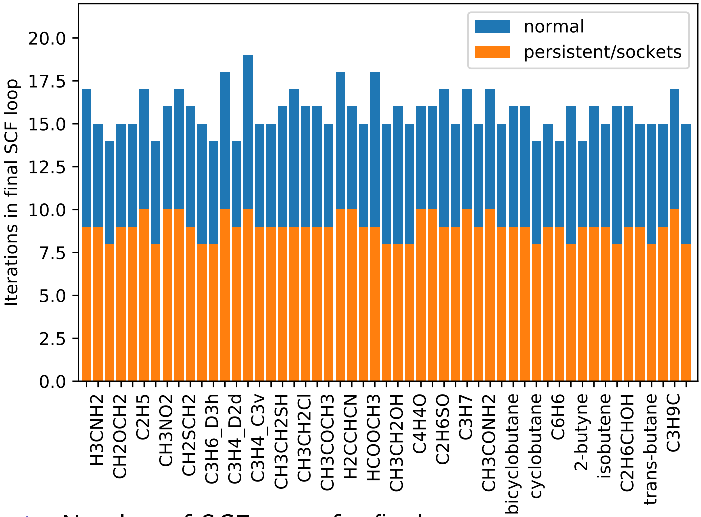
- Start server, listen for connection on socket
- When calculation is triggered, use wrapped calculator to launch subprocess
- Or: Start server on one computer, start client on different computer
- Client process connects to socket
- Server passes positions and cell to client; client passes energy, forces, and stress to server
- Other quantities (atomic species, input parameters in general) are communicated by other means and requires restart

Run FHI-aims with ASE using i-PI socket protocol:

```
from ase.build import molecule
from ase.optimize import BFGS
from ase.calculators.aims import Aims
from ase.calculators.socketio import SocketIOCalculator

port = 31415
atoms = molecule(’H2O’, vacuum=3.0)
atoms.rattle(stdev=0.1)
aims = Aims(command=’ipi.aims.171221_1.mpi.x’,
            use_pimd_wrapper=(’localhost’, port),
            compute_forces=True,
            xc=’LDA’,
            species_dir=’/home/aimsuser/species_dir’)
opt = BFGS(atoms, trajectory=’opt.aims.traj’)

with SocketIOCalculator(aims, port=port) as calc:
    atoms.calc = calc
    opt.run(fmax=0.05)
```



- Relaxation time for randomly perturbed molecules with Aims, no wavefunction extrapolation (some axis labels omitted)



- Number of SCF steps for final step
- About 40 % fewer iterations on average

Support in ASE for codes implementing i-PI clients

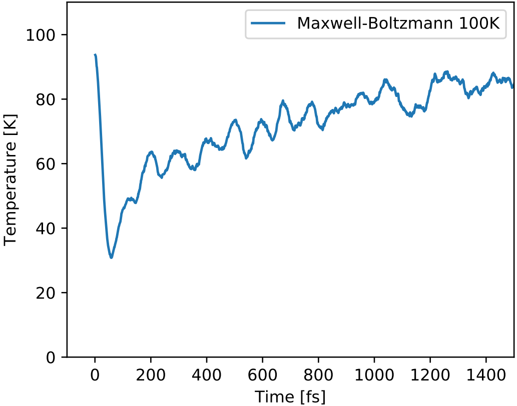
Program name	Supported by ASE calculator
Quantum Espresso	Yes
FHI-aims	Yes
Siesta	Yes
DFTB+	Yes, presumably (untested)
Yaff	No; there is no ASE calculator for Yaff
cp2k	No; ASE uses cp2k shell instead
Lammps	No; ASE uses lammpsrun/lammpslib instead
ASE	Yes - ASE provides a client as well
GPAW	Yes, using the ASE client

Planned/on-going developments for molecular dynamics

- ASE supports molecular dynamics with several thermostats (Langevin, Berendsen NPT/NVT, . . .
- High-level tools for initialization/calibration of MD runs
- High-level tools for process management, error handling, . . .

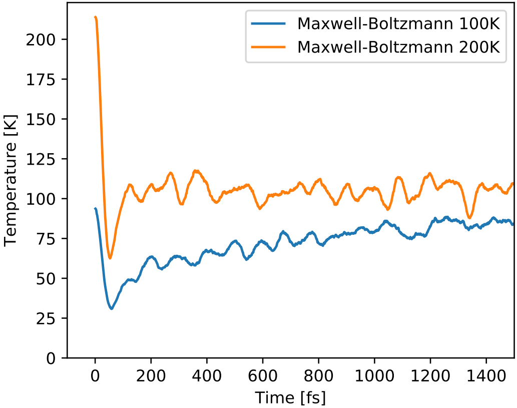
Thermalization in molecular dynamics

- How do we start an MD run at a given temperature?
- Maxwell–Boltzmann distribution ignores potential energy contribution in solids



Langevin thermostat at 100 K running on FCC structure of random alloy

Equilibrium positions, double temperature



- Almost good, but unphysical starting point

MD initialization from phonon modes

- Initialize displacements and velocities according to phonon modes ϵ_{ai}

$$\mathbf{R}_a = \sqrt{\frac{k_B T}{m_a}} \sum_i \frac{\epsilon_{ai}}{\omega_i} A_i \sin(P_i)$$

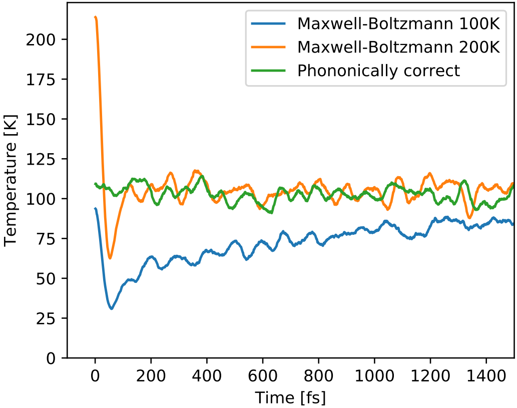
$$\mathbf{v}_a = \sqrt{\frac{k_B T}{m_a}} \sum_i \epsilon_{ai} A_i \cos(P_i)$$

- A_i are random normally distributed numbers
- P_i are uniformly random phases $0 \ldots 2\pi$

`http://ollehellman.github.io/program/extract_forceconstants.html`

More on phonons: See poster by Florian Knoop

MD initialization from phonon modes



- Starts at physically meaningful configuration

Concluding remarks

- Web page: `https://wiki.fysik.dtu.dk/ase/`
- Gitlab: `https://gitlab.com/ase/ase`
- Mailing lists, IRC: `https://wiki.fysik.dtu.dk/ase/contact.html`