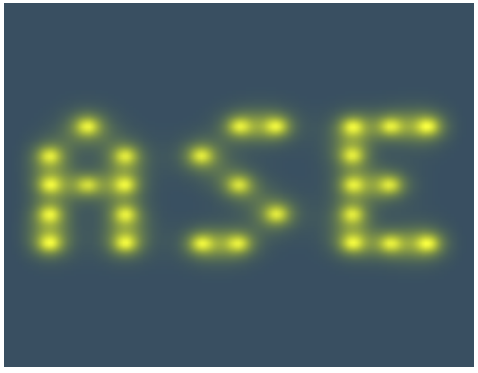
What's new in GPAW and ASE



- What is GPAW and ASE?
- GPAW and ASE news
- New `ase.db` module
- Infrastructure
  - git
  - web-page
  - testing
- Command-line tools
- Python versions
- Future work

GPAW and ASE is ...

ASE is: The Atomic Simulation Environment (ASE): a set of tools and Python modules for setting up, manipulating, running, visualizing and analyzing atomistic simulations.

GPAW is: An implementation of the "Projector augmented-wave method"

Wave functions are described using

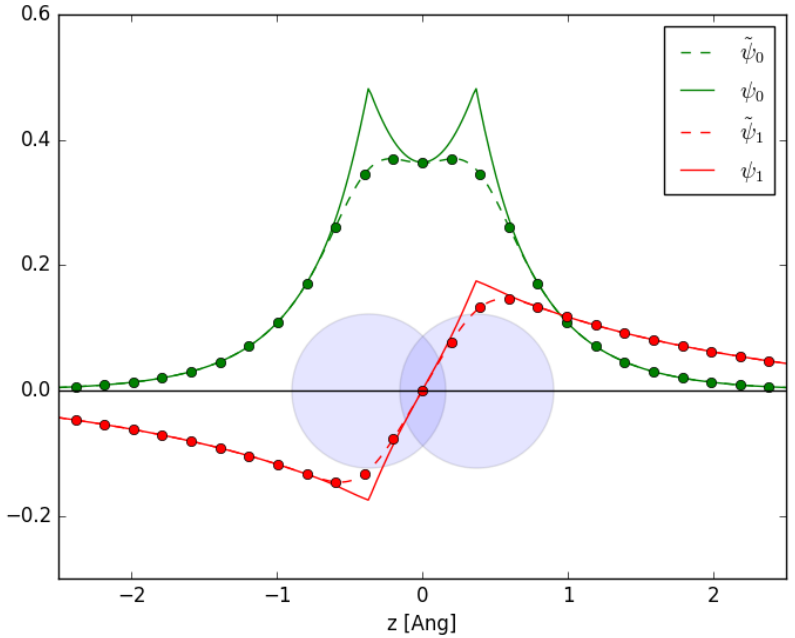
- real-space uniform grids (fd)
- atom-centered numerical basis functions (lcao)
- plane-waves (pw)

It's written in a combination of the Python and C languages, based on ASE and NumPy

Uses these libraries for the hard work: BLAS, LAPACK, ScaLAPACK, BLACS, MPI, FFTW, LIBXC

PAW

Exact* all-electron formalism, pseudo-potentials done right, access to full all-electron wave-functions, density and potential:

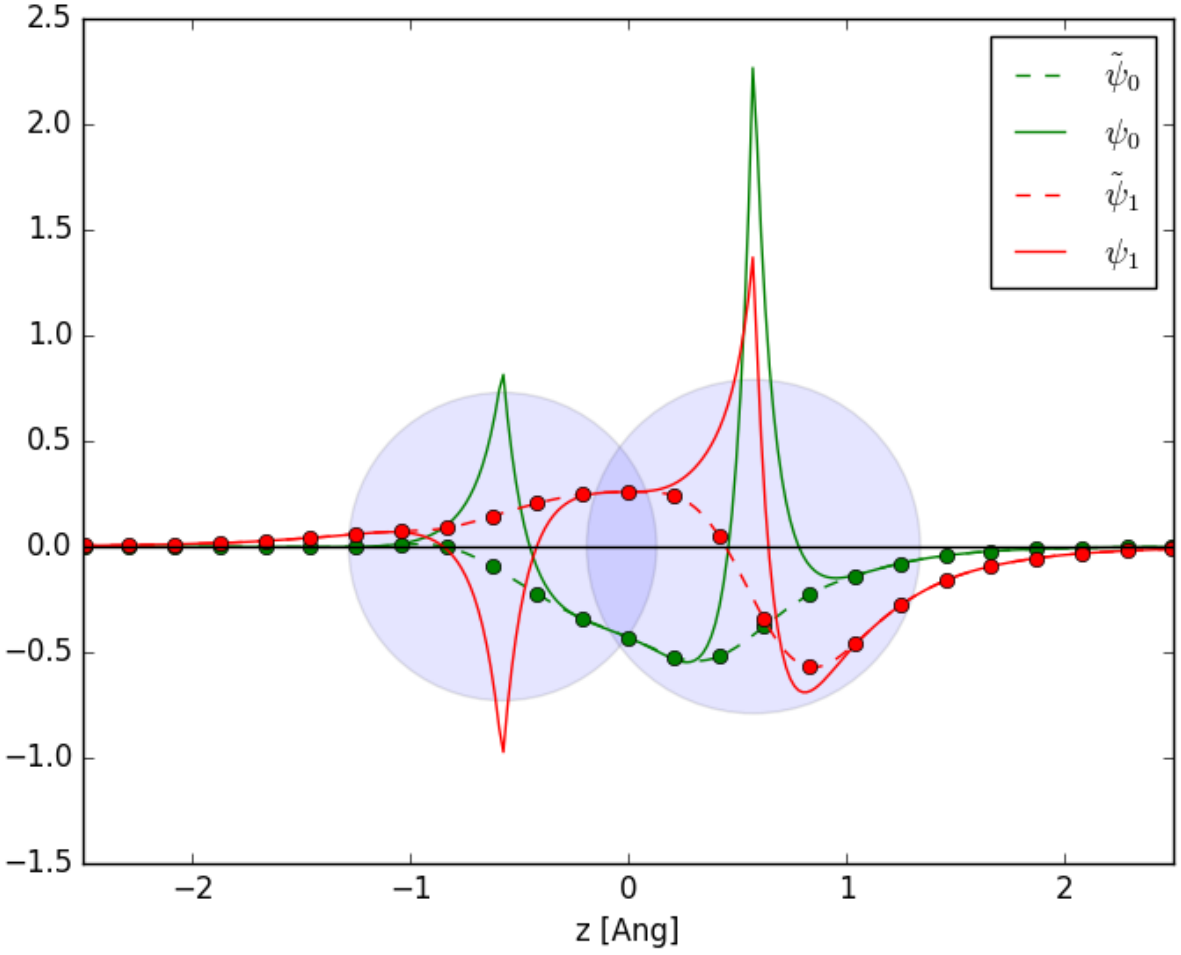


$$\psi = \tilde{\psi} + \sum_a \sum_i (\phi_i^a - \tilde{\phi}_i^a) \langle \tilde{p}_i^a | \tilde{\psi} \rangle$$

Peter Blöchl: "Projector augmented-wave method", Phys. Rev. B 50, 17953 (1994)

(*) frozen core and finite set of partial waves

More wave functions: CO molecule



GPAW news (since 2013)

- Time-propagation TDDFT with LCAO
- Complete rewrite of density response function code (GW, BSE, RPAc, ...)
- Non-selfconsistent spin-orbit coupling
- Non-symmorphic symmetries
- Orbital-free DFT
- Classical electrodynamics simulations
- Continuum solvent model
- Band structure unfolding
- Plane-wave mode is now a proper member of the family
- Stress tensor
- New web-page (mobile friendly and updated information about PAW datasets, `https://wiki.fysik.dtu.dk/gpaw/setups/Na.html`)

Pseudo-to-all-electron tool (ps2ae)

```
from gpaw.utilities.ps2ae import PS2AE
from gpaw import GPAW

calc = GPAW('something.gpw', txt=None)

# Transformer:
t = PS2AE(calc, h=0.05)

ps = t.get_wave_function(n, k, s, ae=False)
ae = t.get_wave_function(n, k, s, ae=True)
norm = t.gd.integrate(abs(ae)**2)
assert norm == 1
```

$$\psi = \tilde{\psi} + \sum_a \sum_i (\phi_i^a - \tilde{\phi}_i^a) \langle \tilde{p}_i^a | \tilde{\psi} \rangle$$

ASE news (since 2013)



- New calculator interfaces: JDFTx, CP2K, Octopus, Siesta
- New modules: `ase.ga`, `ase.db`, `ase.collections`
- The big rename: `ase.build`, `ase.geometry`, `ase.eos`, `ase.neighborlist`, `ase.spacegroup`
- New trajectory file format
- Phase-diagrams, Pourbaix diagrams
- Improved initial guess for NEB calculations

Updated `ase.units` module (CODATA-2014):

year	**bohr [Å]**	**hartree [eV]**
1986	0.52917726	27.211396
2014	0.52917721	27.211386

ASE-paper ...

ASE trajectory files

Warning

Convert your old trajectory files now!

You can identify and convert old trajectory files like this:

```
$ python -m ase.io.formats a.traj
a.traj: Old ASE pickle trajectory (trj+)
$ python -m ase.io.trajectory a.traj # convert
$ python -m ase.io.formats a.traj a.traj.old
a.traj:     ASE trajectory (traj)
a.traj.old: Old ASE pickle trajectory (trj)
```

New database module in ASE

In each row we have:

Taxonomy:

- `Atoms` object (positions, atomic numbers, ...)
- ID, user-name, creation and modified time
- Constraints
- Calculator name and parameters
- Energy, forces, stress tensor, dipole moment, magnetic moments

Folksonomy:

- Key-value pairs (string or number)

Additional stuff:

- Extra data (band structure, dos, ...)

`https://wiki.fysik.dtu.dk/ase/ase/db/db.html`

ase.db examples

Back-ends: JSON, SQLite3 and PostgreSQL.

```
from ase.db import connect
from ase import Atoms
con = connect('abc.db')
h = Atoms('H')
con.write(h)
from ase.calculators.emt import EMT
h.calc = EMT()
h.get_forces()
con.write(h, abc=42)
```

```
$ ase-db abc.db --columns +abc
id|age|formula|calculator|energy| fmax|pbc|abc
 1|15m|H      |          |      |     |FFF|
 2|15m|H      |emt       | 3.210|0.000|FFF| 42
Rows: 2
```

ase.db examples ...

```
$ ase-db abc.db abc=42 --json
{"1": {
 "calculator": "emt",
 "calculator_parameters": "{}",
 "cell": [[1, 0, 0], [0, 1, 0], [0, 0, 1]],
 "ctime": 16.422649618451555,
 "energy": 3.21,
 "forces": [[0.0, 0.0, 0.0]],
 "key_value_pairs": {"abc": 42},
 "mtime": 16.422670641858346,
 "numbers": [1],
 "pbc": [false, false, false],
 "positions": [[0.0, 0.0, 0.0]],
 "unique_id": "6db95e1595e205f4cfdf8801c1349",
 "user": "jensj"},
"ids": [1],
"nextid": 2}
```

https://cmr.fysik.dtu.dk/

Infrastructure

- Git
- GitLab
- Testing
- Web-page
- Distribution

Git and GitLab

Read this:

https://wiki.fysik.dtu.dk/ase/development/contribute.html

and watch this:

Introduction to Git with Scott Chacon of GitHub: `https://www.youtube.com/watch?v=ZDR433b0HJY`

- `https://gitlab.com/ase/ase`
- `https://gitlab.com/gpaw/gpaw`

Fork, clone, branch, push, create merge request (MR), code review, ...

Testing

ASE:

- GitLab-CI: Python 2.7
- Niflheim: Python 2.6

GPAW:

- GitLab-CI: Python 2.7 (one test)
- Niflheim (1, 2, 4, 8 cores): Python 2.6
- Niflheim (AGTS): Python 2.6

GPAW+ASE:

- My computer: Python 3.5
- Sphinx: Python 2.7

AGTS example

(Advanced GPAW Test System)
`https://wiki.fysik.dtu.dk/gpaw/tutorials/pbe0/pbe0.html`

```
$ ls doc/tutorials/pbe0/
eos.py  gaps.py  pbe0.rst  plot_a.py
submit.agts.py
```

This is what `submit.agts.py` looks like:

```
def agts(queue):
    queue.add('gaps.py', creates='gaps.csv')
    eos = queue.add('eos.py', ncpus=4)
    queue.add('plot_a.py', deps=eos,
              creates='si-a.png')
```

Web-pages

The new web-page uses a slightly tweaked Read-the-docs Sphinx-theme

- I love Sphinx: developer friendly, you can use `grep`, `find`, `git` and your favorite text editor (`http://www.sphinx-doc.org/`)
- Builds after every push
- It has never been easier:

```
$ pip install sphinx_rtd_theme --user
$ git clone https://gitlab.com/ase/ase.git
$ cd ase/doc
$ make
$ make browse
```

Distributing the code

ASE and GPAW is on PyPI: `https://pypi.python.org/`

Installation:

1) PIP:

```
pip install ase --user
pip install gpaw --user
```

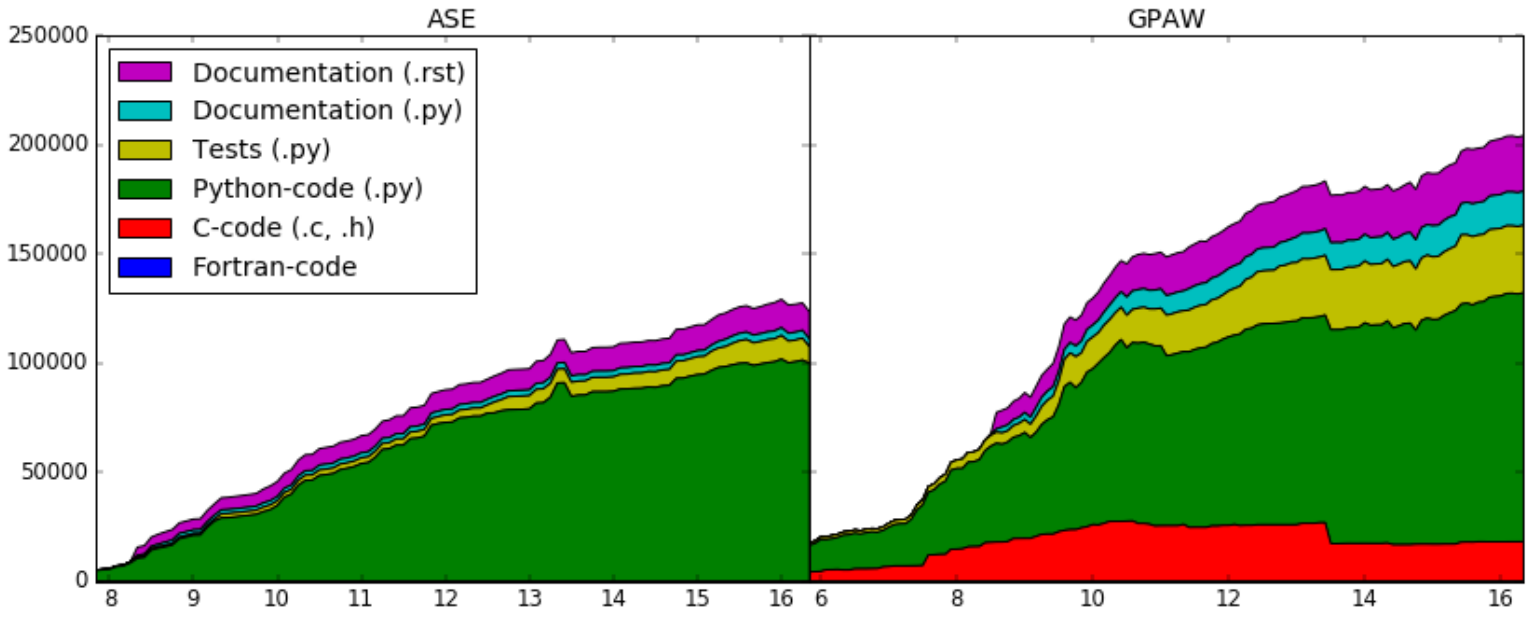
   (installs in `~/.local/`)

2) Clone from GitLab
3) Tar-files can be downloaded from PyPI
4) Binary wheels?

New release: adjust version number and do:

```
$ python setup.py sdist upload
```

Lines of code



Want to help develop ASE and GPAW? Learn Python, NumPy, Sphinx, Git and read 200000 lines of Python code!

ASE command-line tools

ASE has the following command line tools:

- `ase-gui`: simple graphical user interface
- `ase-build`: build simple molecule or bulk structure
- `ase-run`: run calculations with ASE's calculators
- `ase-db`: manipulation of databases

See the "command line tools" entry in the menu on the web-page (`https://wiki.fysik.dtu.dk/ase/cmdline.html`)

CLI examples

```
$ ase-build CO | ase-run nwchem -f 0.05
Running: CO
LBFGS:    0  08:36:52  -3041.669444      0.8139
LBFGS:    1  08:36:52  -3041.659936      1.5638
LBFGS:    2  08:36:53  -3041.672467      0.0457
$ ase-gui CO.traj@-1
```

```
$ ase-build -x fcc Cu -a 3.6
$ ase-run emt Cu.json --equation-of-state=5,2
Running: Cu
$ python -m ase.eos Cu.traj
# filename points  volume energy bulk modulus
#                 [Ang^3]   [eV]         [GPa]
Cu.traj          5  11.565 -0.007      134.419
```

GPAW command-line tool

GPAW has *one* command line tool: `gpaw`

```
$ gpaw test
$ gpaw install-data
$ gpaw info
$ gpaw run h2.traj -p xc=PBE -W h2.gpw
$ gpaw atom Si -f PBE
...
```

(`gpaw run` is the same as `ase-run gpaw`)

```
$ ase-build -x zincblende GaAs -a 5.6 zb.json
$ gpaw run -p kpts=2,2,2 zb.json
```

Python versions

Python 2.6:

- Released on October 1, 2008
- Being dropped by NumPy, SciPy, MatPlotLib, Sphinx, ...

Python 2.7:

- Released on July 3, 2010
- New stuff:
  - ordered-dictionary type
  - `argparse` module for parsing command lines
  - the `Counter` class in the collections module
  - `'{}:{}:{}'.format(2009, 04, 'Sunday')`
  - set literals (`{1,2,3}` is a mutable set)
  - dictionary and set comprehensions (`{i: i*2 for i in range(3)}`)

Python 3

We write code compatible with both 2 and 3.

- Use `print(...)` and sometimes add a:

```
from __future__ import print_function
```

- If you pickle something: use `open(..., 'rb')` or `open(..., 'wb')`
- Remember that `zip()`, `range()`, `dict.items()` and friends no longer return `list`s
- Use `ase.utils.basestring`

From my `.bashrc`:

```
alias p=python3
```

Start using Python 3 now (it takes more that a year to get used to the new print-function)
Everything seems to work. Except ASE's simple GUI)-:

Future

1) GPAW-1.0.1: bug-fix release (spin-orbit)
2) Self-consistent hybrid functionals in PW-mode with **k**-points
3) Rewrite of I/O stuff and new gpw file format
4) Various refactoring projects
5) ...

Things from 2013 talk

Important work:

- New PAW setups
- Robust eigensolver/density mixer
- GPAW-1.0?

Also important work, but not as important as the above:

- Better basis sets for lcao
- Parallelize plane-wave calculations over plane-waves
- Generalized multigrid solvers that can handle grid sizes of the form: $2^a 3^b 5^c 7^d$

Questions:

- When do we drop support for Python 2.4 and 2.5?
- Strategy for porting GPAW to Python 3?
- Switch from SVN to Bazaar and Launchpad?

Discussion (Thursday + Friday)

- Web-page improvements
- Mail lists?
- Supported Python version
- Future of `ase-gui`?
- Use libvdwxc by default?
- How to handle PAW + fully non-local XC functionals (like vdw-DF)
- Documentation for MAC users
- I/O and new gpw format
- Use setuptools for handling dependencies?
- New keywords for controlling parallelization?

Finally ...

Thank you for your attention

Questions?

Extra material

...

Python 3 examples

```
from __future__ import print_function
# print >> f, ...
print(..., file=f)
print('hello')
print('hello', name)
print('hello ...', end='')
print()

# keys = dct.keys()
# keys.sort()
keys = sorted(dct)
```

More examples

```
# import cPickle as pickle
# fd = open('things.pckl', 'w')
# pickle.dump(things, fd,
#             pickle.HIGHEST_PROTOCOL)
# things = pickle.load(open('things.pckl'))
import pickle
with open('things.pckl', 'wb') as fd:
    pickle.dump(things, fd,
                pickle.HIGHEST_PROTOCOL)
with open('things.pckl', 'rb') as fd:
    pickle.load(fd)
```

Even more examples

```
from __future__ import division
assert 1 / 2 == 0.5
assert 1 // 2 == 0

stuff = list(range(42))
things = list(zip('ABC', range(3)))
import numpy as np
stuff2 = np.arange(42)

# isinstance(x, int)
import numbers
isinstance(x, numbers.Integral)
```