Ask Hjorth Larsen
asklarsen@gmail.com

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

May 31, 2017

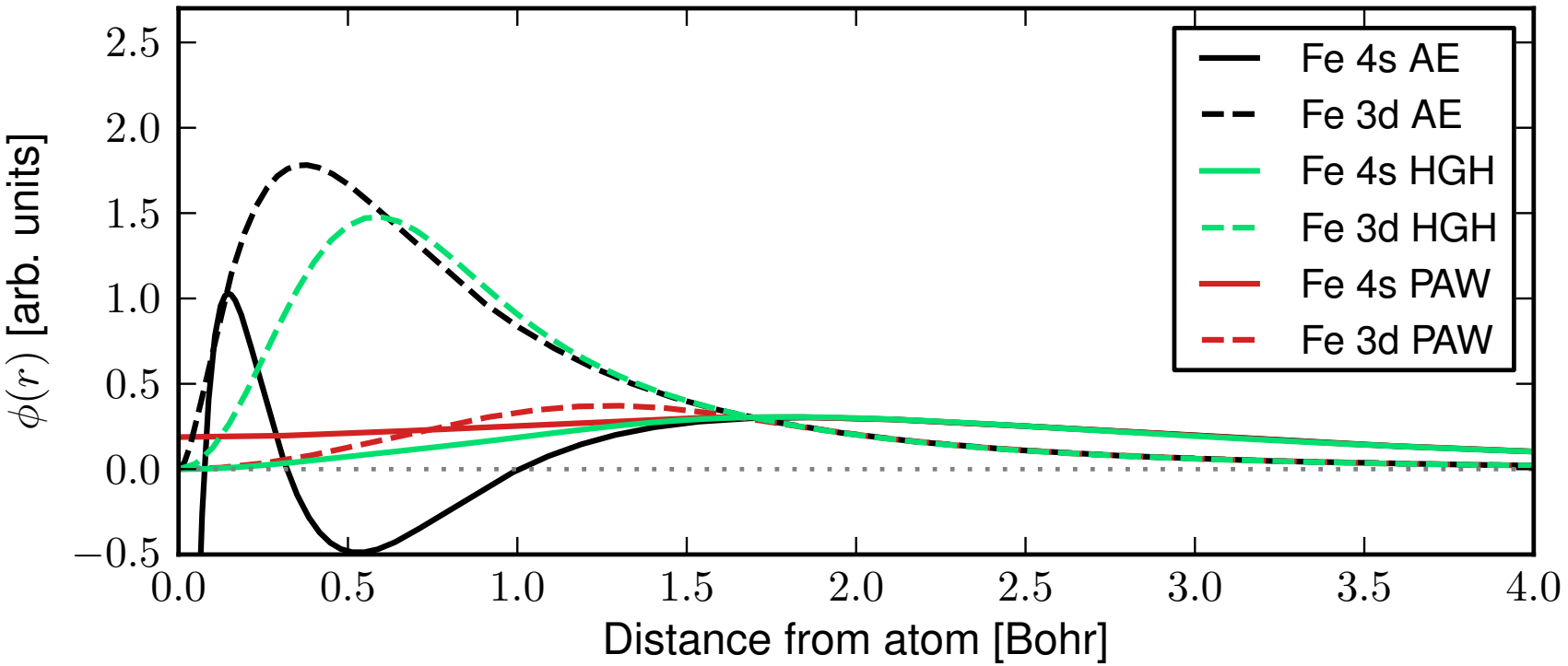
GPAW

- Density functional theory code written in Python and C
- `https://wiki.fysik.dtu.dk/gpaw/`
- Projector augmented wave method (Blöchl Phys. Rev. B 50, 17953, 1994)
- Calculations are written as Python scripts using ASE
- Three calculation modes: FD, LCAO, PW
- Parallelization with MPI; highly scalable (except PW)
- Python interfaces for important functions from MPI, BLAS, LAPACK, ScaLAPACK

The case for Python in DFT codes

- Probably more than 90 % of CPU time is spent in less than 10 % of the code
- Write 90 % in a nice language and 10 % in C/Fortran
- DFT codes do many non-computational things:
  - Glue components together (Poisson, eigensolver, mixing, etc.)
  - Read and write data
  - String processing (parse pseudopotentials, format output, ...)
  - Parallel operations (time spent waiting for hardware)
- Disadvantage: Mixing languages does have some extra complexity

PAW and the quest for smooth wavefunctions



- Wavefunctions oscillate and are expensive to represent
- Remove core electrons, replace nuclei with pseudopotentials
- PAW: Extension to pseudopotentials. Avoids norm-conservation, and retains “all-electron” information

The projector augmented wave method

- Valence states represented by $|\psi\rangle = \hat{\mathcal{T}}\,|\tilde{\psi}\rangle$
- PAW transformation: $\hat{\mathcal{T}} = 1 + \sum_{ai} (|\phi_i^a\rangle - |\tilde{\phi}_i^a\rangle)\,\langle \tilde{p}_i^a|$
- Given atomic states $\phi_i^a(r)$, choose smooth functions $\tilde{\phi}_i^a(r)$ and localized "projectors" $\tilde{p}_i^a(r)$; this defines $\hat{\mathcal{T}}$.
- PAW transformation maps nice smooth pseudowavefunctions to ugly oscillating all-electron wavefunctions
- Calculate quantities from smooth functions plus atomic contributions:

$$\langle \phi|\hat{O}|\psi\rangle = \underbrace{\langle \tilde{\phi}|\hat{O}|\tilde{\psi}\rangle}_{\text{3D grid}} + \sum_{ai} \langle \tilde{\phi}|\tilde{p}_i^a\rangle \underbrace{\left\langle \begin{array}{c}\text{atomic}\\ \text{properties}\end{array}\right\rangle}_{\text{System independent}} \langle \tilde{p}_i^a|\tilde{\psi}\rangle$$

- Projector functions couple 3D calculation to atomic corrections

GPAW

Uniform real-space grids

- Accurate, parallelizes very well
- Grid causes “egg-box” effect

Numeric atomic orbitals

- Very fast – easy to do thousands of atoms
- Not as accurate; cannot easily reach complete basis set

Planewave mode

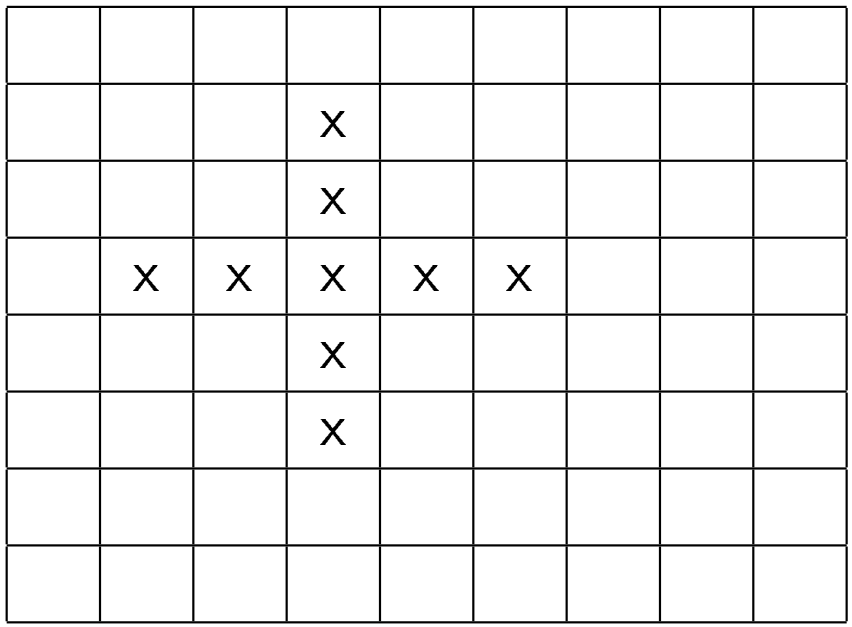
- Fast for small systems (crystals)
- Planewaves numerically well-behaved
- Not scalable

Real-space ("FD") mode

- `GPAW(mode='fd', h=0.18)`
- Iterative eigensolvers (solve "a bit" every iteration)
- 3D main CPU mesh: distribute $\psi_{kn}(\mathbf{r})$ over spins/k-points, domains and bands
- Wave functions on coarse grid, density/potential on fine grid
- Multigrid: Solve 2x2x2 grid; then on 4x4x4 grid; then on 8x8x8 grid; etc.
- Features: Real-time TDDFT, Linear response TDDFT

Stencils

- Differential operators represented by finite-difference stencils
- $\hat{O}\tilde{\psi}(x,y,z) = \sum_{ijk} w_{ijk}\tilde{\psi}(x + i\,\mathrm{d}x, y + j\,\mathrm{d}y, z + k\,\mathrm{d}z)$



More nearest neighbours $\rightarrow$ higher order $\rightarrow$ better convergence

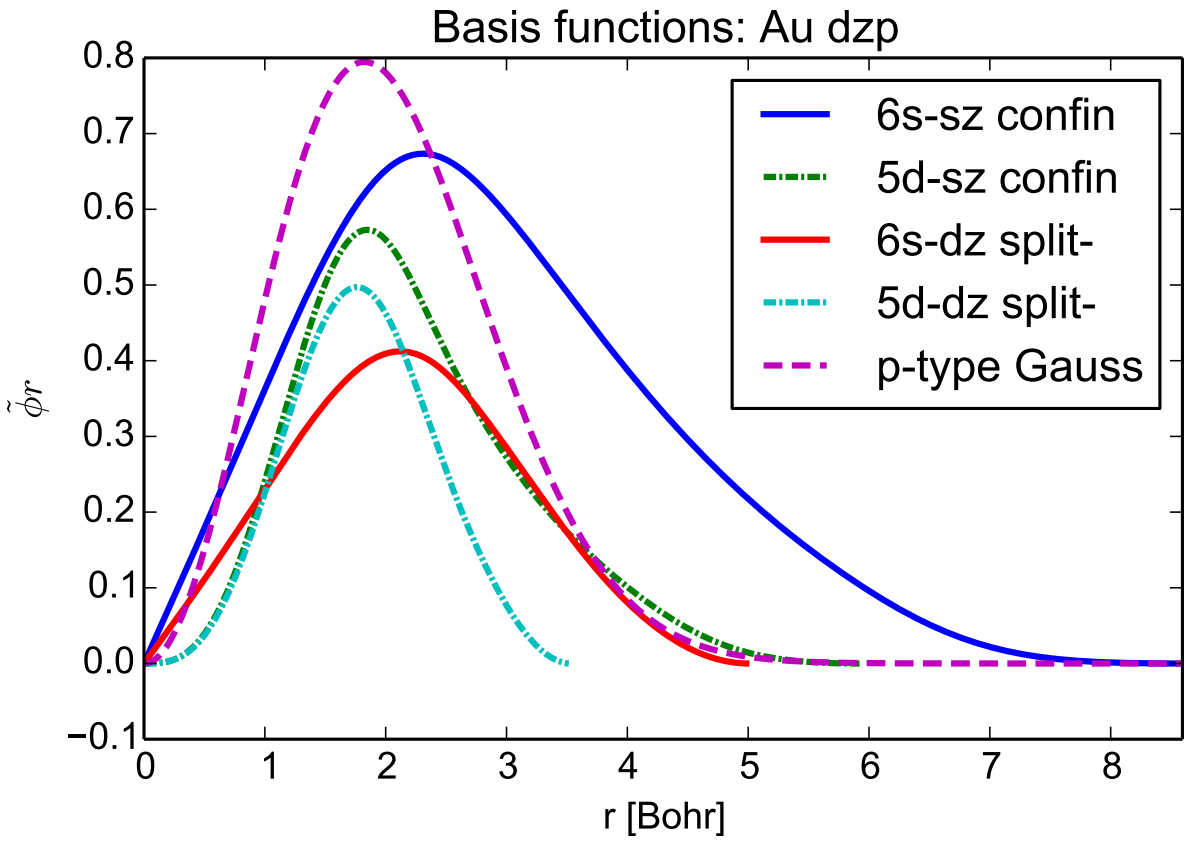
PW mode

- `GPAW(mode=PW(400))`
- Same eigensolvers as FD mode (iterative)
- (Almost) no egg-box effect
- Parallelize over spins/k-points and states ("bands")
- Only mode supporting stresses
- Features: RPA linear response, G_0W_0 (not selfconsistent but under development), Bethe–Salpeter equation

LCAO mode

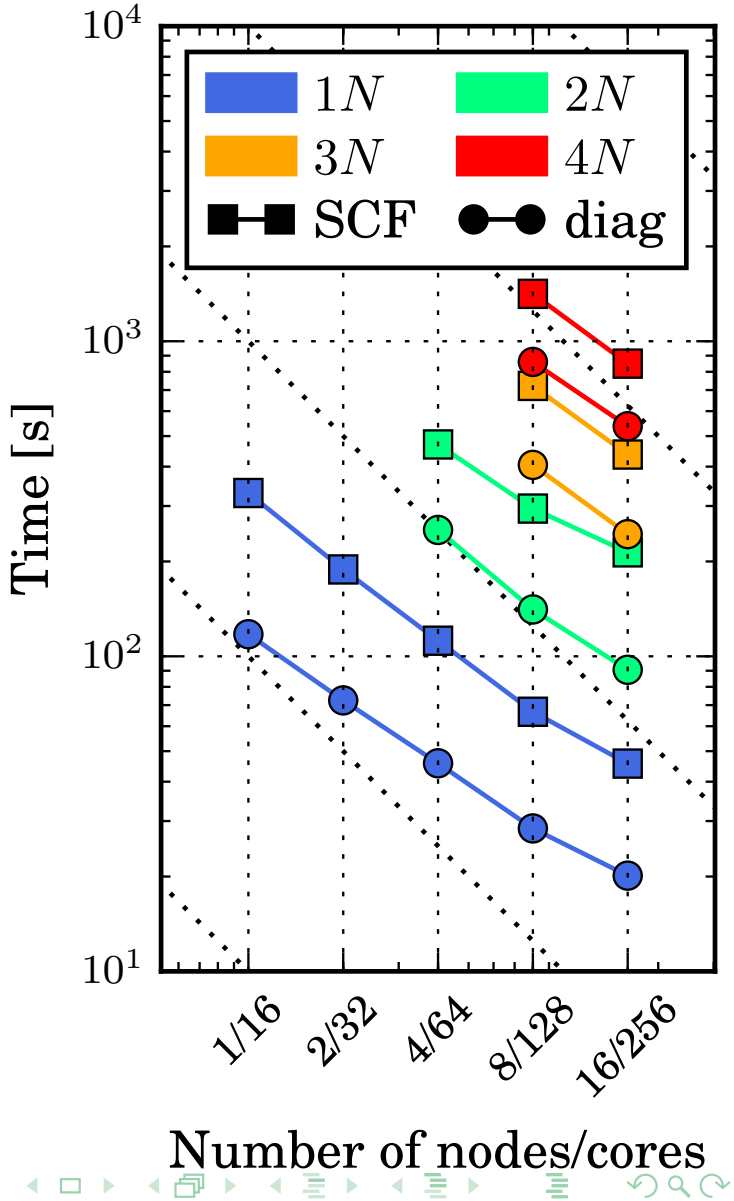
- `GPAW(mode='lcao', basis='dzp', h=0.18)`
- Wavefunctions represented as $|\tilde{\psi}_n\rangle = \sum_\mu |\Phi_\mu\rangle\, c_{\mu n}$
- $|\Phi_\mu\rangle$ are localized functions based on atomic orbitals
- Small basis set ($\sim$ 5–20 functions per atom) allows direct diagonalization
- Kohn–Sham equations explicitly solved as generalized eigenvalue problem: $\sum_\nu H_{\mu\nu} C_{\nu n} = \sum_\nu S_{\mu\nu} C_{\nu n} \epsilon_n$
- Parallelize over spins/k-points, domains, orbitals
- Parallelize linear algebra with BLACS/ScaLAPACK
- Features: Time-propagation TDDFT (LCAO-TDDFT)

Localized basis sets



Parallel scaling

- Parallel performance of GPAW/LCAO
- Timing per SCF step
- 2500–10000 atoms



Parallel scripting

- GPAW compiles MPI-enabled interpreter `gpaw-python`
- `mpirun -np 4 gpaw-python script.py`
- Each core has distinct "rank" but runs the same code

```
import numpy as np
from gpaw.mpi import world
print('hello {}/{}'.format(world.rank, world.size))
array = np.ones(3) * world.rank
big_array = None
if world.rank == 0:
    big_array = np.zeros(3 * world.size)
world.gather(array, 0, big_array)
if world.rank == 0:
    print(big_array)
world.sum(array)
print(array)
```

ASE

- Web page: `https://wiki.fysik.dtu.dk/ase/`
- Mailing list: `ase-users@listserv.fysik.dtu.dk`
- git: `https://gitlab.com/ase/ase`
- IRC: `#ase` on `irc.freenode.net`
- License: LGPLv2.1+

GPAW

- Web page: `https://wiki.fysik.dtu.dk/gpaw/`
- Mailing list: `gpaw-users@listserv.fysik.dtu.dk`
- git: `https://gitlab.com/gpaw/gpaw`
- IRC: `#gpaw` on `irc.freenode.net`
- License: GPLv3+

Contribute to ASE!

- `https://wiki.fysik.dtu.dk/ase/development/contribute.html`
- Create account on Gitlab
- Clone ASE (and learn git...)
- Hack
- Push changes to your own branch on Gitlab
- Create merge request

Things to do

- Report and fix bugs
- Add your favourite calculator
- Add your favourite file format
- Translate ASE GUI into Catalan

Ask not what free software can do for you;
ask what you can do for free software!
— Not quite JFK