GPAW for developers

Jens Jørgen Mortensen

Department of Physics

Technical University of Denmark

Installation
Python tricks
Structure of the GPAW code
Setups
Parallelization
Optimization
Future work

Requirements for GPAW

- The Python interpreter - version 2.3 or better.
- A C-compiler.
- Non-standard Python packages:
  - Numeric.
  - ASE (http://wiki.fysik.dtu.dk/ase).
- Libraries:
  - BLAS.
  - LAPACK.
  - (MPI)

Installing GPAW

- System administrator:
  - Unpack the tar-file (get it from http://wiki.fysik.dtu.dk/gpaw/Download).
  - python setup.py install
- User:
  - svn checkout https://svn.berlios.de/svnroot/repos/gridpaw/trunk gpaw
  - python setup.py install –-home=$HOME
  - Put $HOME/lib/python in your $PYTHONPATH
- Developer:
  - svn checkout https://jensj@svn.berlios.de/svnroot/repos/gridpaw/trunk gpaw
  - python setup.py build_ext
  - Put $HOME/gpaw in your $PYTHONPATH

Setups:

- Unpack the tar-file from the wiki (or make your own setups).
- Point $GPAW_SETUP_PATH at the directory containing the setup files:
  - $GPAW_SETUP_PATH=$HOME/setups
  - $GPAW_SETUP_PATH=$HOME/mysetups:/usr/share/gpaw/setups

Python rules and tricks

1)Use an editor that knows about Python!
2)Never ever put tab's in Python code!
3)Always use four spaces for indentation!

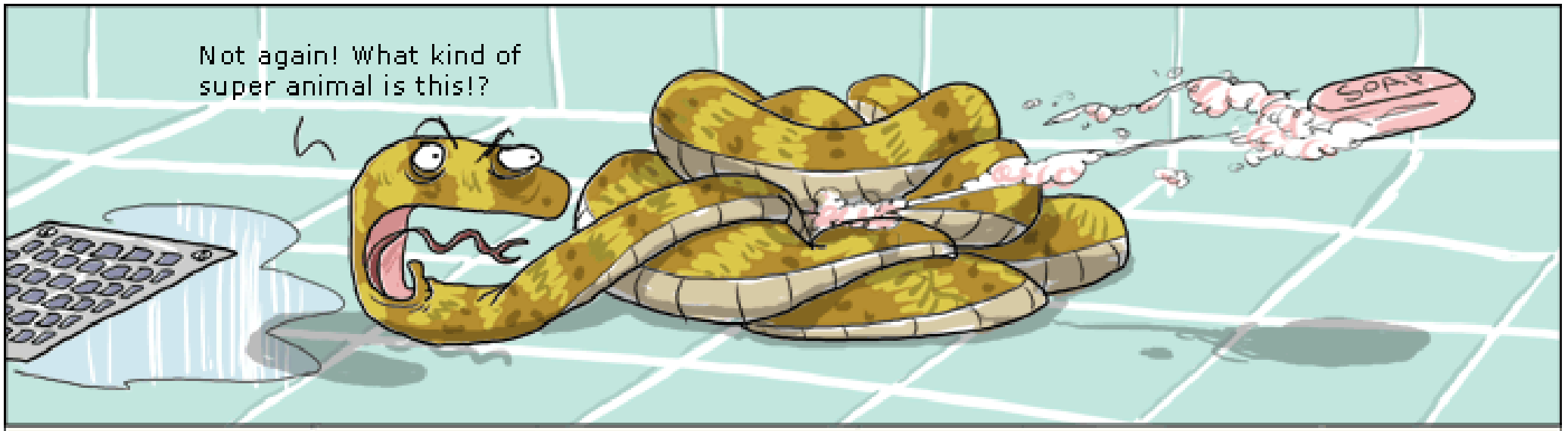
Turn on tab-completion for interactive Python sessions.

Set $PYTHONSTARTUP=$HOME/.pythonrc and put this in the startup file:

```
import rlcompleter
import readline
readline.parse_and_bind('tab: complete')
```

alias p=python

www.wulffmorgenthaler.com



Python attacks a bar of soap

GPAW's directory structure

```
/gpaw/setup.py
      config.py
      README.txt
      gpaw/paw.py
           nucleus.py
           ...
           mpi/...
           atom/...
           utilities/...
           eigensolvers/...
      c/_gpaw.c
        ...
        bmgs/...
      test/test.py
           ...
      demo/H.py
           ...
      tools/gpaw-setup
      scripts/moleculetest.py
```

The Paw.find_ground_state() SCF loop

Before entering the loop we must have a density, a potential and an orthonormalized set of wave functions:

$$n_\sigma(\vec{r}) = \tilde{n}_\sigma(\vec{r}) + \sum_a \sum_{i_1 i_2} D^a_{\sigma i_1 i_2} \left[\phi^a_{i_1}(\vec{r} - \vec{R}^a) \phi^a_{i_2}(\vec{r} - \vec{R}^a) - \tilde{\phi}^a_{i_1}(\vec{r} - \vec{R}^a) \tilde{\phi}^a_{i_2}(\vec{r} - \vec{R}^a) \right]$$

$$\hat{H}_\sigma = -\frac{1}{2}\nabla^2 + \tilde{v}_\sigma(\vec{r}) + \sum_a \sum_{i_1 i_2} |\tilde{p}^a_{i_1}\rangle \Delta H^a_{\sigma i_1 i_2} \langle \tilde{p}^a_{i_2}|$$

$$\langle \tilde{\Psi}_{\sigma \vec{k} n} | \hat{S} | \tilde{\Psi}_{\sigma \vec{k} m} \rangle = \delta_{nm}, \quad \hat{S} = 1 + \sum_a \sum_{i_1 i_2} |\tilde{p}^a_{i_1}\rangle \Delta S^a_{i_1 i_2} \langle \tilde{p}^a_{i_2}|$$

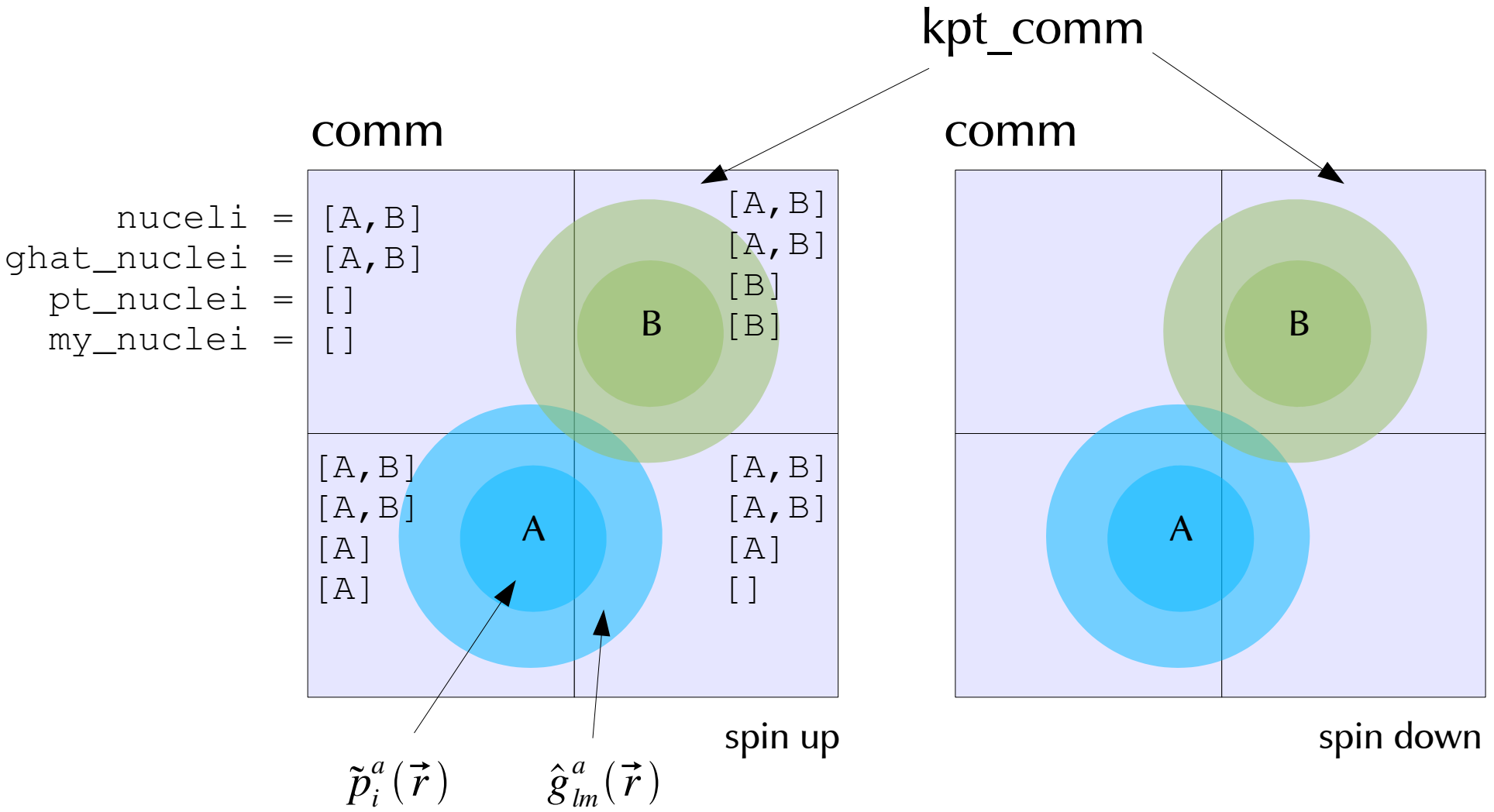
Inside the loop this equation is solved:

$$(\hat{H}_\sigma - \epsilon_{\sigma \vec{k} n} \hat{S}) \Psi_{\sigma \vec{k} n} = 0$$

Mapping between formulas and variable names

$D^a_{\sigma i_1 i_2}$	nuclei[a].D_sp[s, p]
$\tilde{n}_\sigma(\vec{r})$	nt_sG[s, nx, ny, nz] or nt_sg[s, nx, ny, nz]
Q^a_{lm}	nuclei[a].Q_L[L]
$\vec{F}^a = -\frac{\partial E}{\partial \vec{R}^a}$	F_ac[a, c]
$P_{\sigma \vec{k} n i} = \langle \tilde{p}^a_i \vert \tilde{\Psi}_{\sigma \vec{k} n} \rangle$	nuclei[a].P_uni[u, n, i]
$\tilde{\Psi}_{\sigma \vec{k} n}(\vec{r})$	kpt_u[u].psit_nG[n, nx, ny, nz]

Parallelization



Optimization

- All loops over grid points are already in C-code:
  - Numeric.
  - GPAW's own C-code.
  - BLAS/LAPACK/MPI.
- The inner loops of the parts with quadratic and cubic scaling has been somewhat optimized!

- GPAW's own C-code.
- Parallel runs:
  - ScaLAPACK?
  - Do more than one band at a time?
- Linear scaling parts of the code.

Work to be done:

- Make the Numeric to numpy transition.
- More output formats:
  - NetCDF.
  - Simple tar-file.
  - HDF?
- Rewrite parallel code:
  - Allow different number of grid points on different CPU's.
  - More efficient communication.
- Non-orthorhombic unit cells?
- Normconserving pseudopotentials?
- Text-output needs some love and care!
- Efficient storage of Gaunt-coefficients and other large tensors.

Wannier functions and linear scaling