Electronic structure calculations with the GPAW code

- History of GPAW
- New PAW setups for GPAW
- Plane-wave implementation
- Future work

The not very accurate history of GPAW

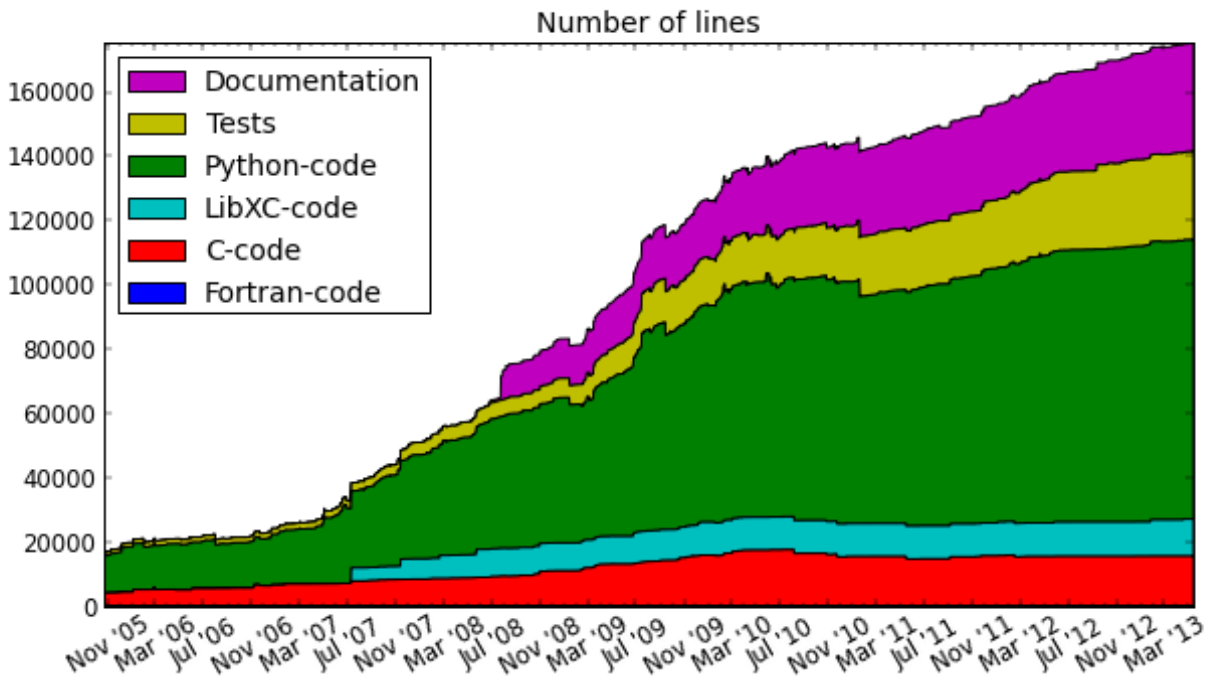
- Around 2002: We need an improved Dacapo:
  - Aim for big systems and massively parallel computers
  - Better parallelization - do everything directly in real-space
  - Based on the Projector-augmented wave method
- September 2003 - August 2005: Sponsored by The Carlsberg Foundation
- 2004: k-point sampling implemented
- October 2005: Finnish connection (TDDFT, GLLB, meta-GGA, ...)
- 2007: Work on LCAO basis set begins
- 2007: GPAW based on libxc
- 2009: Cray, BlueGene
- 2009: FFT implementation of the Rutgers-Chalmers vdW-DF
- 2010: Non-orthorhombic cells
- 2011: Linear dielectric response of an extended system
- 2012: Plane-wave basis added

GPAW Sprint (November 2007)

GPAW Sprint (November 2007) ...

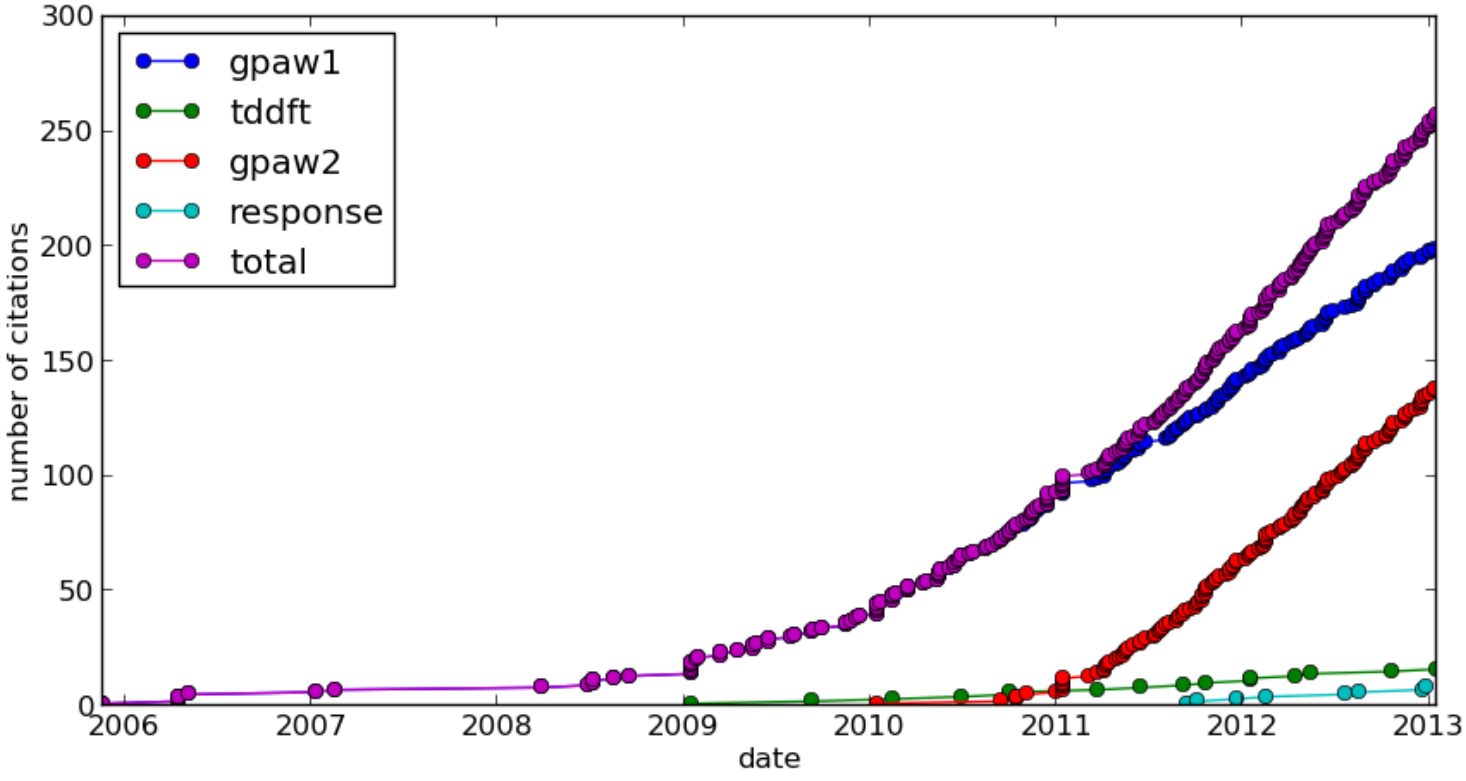
More trivia

- GPAW used to be called GridPAW
- The code started its life in CVS, then moved to SVN at berlios.de and then finally to our own SVN server
- The compiled part of GPAW was written in C++ in the beginning
- GPAW's web-pages used to be in a MoinMoin wiki
- We've used both Numeric, numarray and numpy as our Python array package



More numbers

cpu hours of tests: *4150*
number of committers: *43*
gpaw-users subscribers: *278*
gpaw-developers subscribers: *135*
gpaw-svncheckins subscribers: *22*



Homepage from 2004

[GridPAW]

documentation | manual | faq | tutorials | exercises

installation
mailing lists
developers
technology
bugs!

[CAMP]

[ASE-powered]

[Python Powered]

GridPAW

Grid-based Projector Augmented Wave method

GridPAW is a grid-based real-space implementation of Density Functional Theory (DFT) using the Projector Augmented Wave (PAW) method.
The PAW method[1]:

- gets rid of the core electrons
- works with soft valence wavefunctions
- is an all-electron method (frozen core approximation) - not a pseudopotential method

The use of regular 3D real-space grids for representing wavefunction, densities and potentials allows for:

- efficient multi-grid algorithms for solving Poisson and Kohn-Sham equations
- flexible boundary conditions
- efficient parallelization using real-space domain-decomposition

Warning
The code is at an early stage of development - lots of work needs to be done! The parallelization is not very efficient yet, and boundary conditions must be periodic.

Download page from 2004

[GridPAW]

documentation | manual | faq | tutorials | exercises

installation
mailing lists
developers
technology
bugs!

[CAMP]

[ASE-powered]

[Python Powered]

GridPAW

Requirements

1. Python 2.2 or later is required. Python is available from http://www.python.org.
2. ASE 0.8 or later.
3. Numeric Python (http://numpy.sf.net).
4. Scientific Python (http://starship.python.net/~hinsen/ScientificPython).
5. BLAS and LAPACK libraries.
6. An MPI library is required for parallel calculations.

Tip
You can check that you have everything needed by running `python requirements.py` from the source directory. This program should complain if some requirement is not fulfilled. If you get no complaints, and still have installation problems, please let us know!

Installation

1. Get the latest version: GridPAW-0.4.tar.gz.
2. Unpack the tarball and go to the `GridPAW-0.4` directory:

Testing GPAW's PAW setups

Challenges:

- We need good benchmark results we can trust.
- Getting correct lattice constants or bond-lengths does not guarantee that the setup is good.
- Getting correct cohesive energy or atomization energy is a better test, but this involves atoms: not fun!
- Formation energies for bulk oxides would be a good test, but some of the oxide structures are quite complicated and contain many atoms.

FHI-aims:	All-electron full-potential density functional theory code using a numeric local orbital basis set. `https://aimsclub.fhi-berlin.mpg.de`
ELK:	All-electron full-potential linearised augmented-plane wave (FP-LAPW) code. `http://elk.sourceforge.net/`

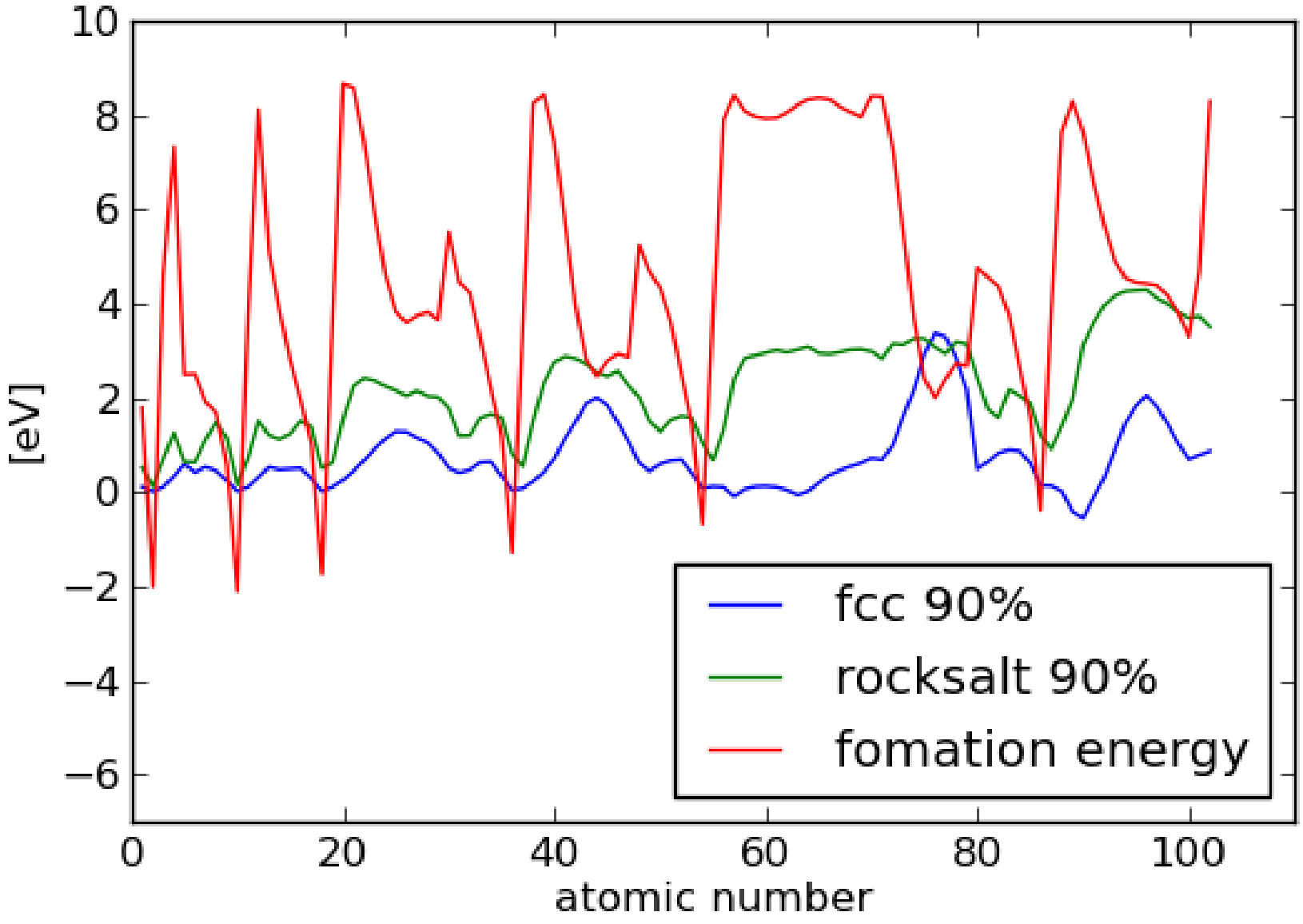
Strategy

- Optimize volume for fcc and rock-salt (oxygen + X) for all elements (from hydrogen to nobelium) using AIMS and ELK.
- Do non-relativistic calculations.
- Compress fcc and rock-salt structures to 90 % of equilibrium lattice constants.
- Use oxide formation energies and fcc and rock-salt compression energies as benchmark numbers.
- Simple tests make iterations faster.

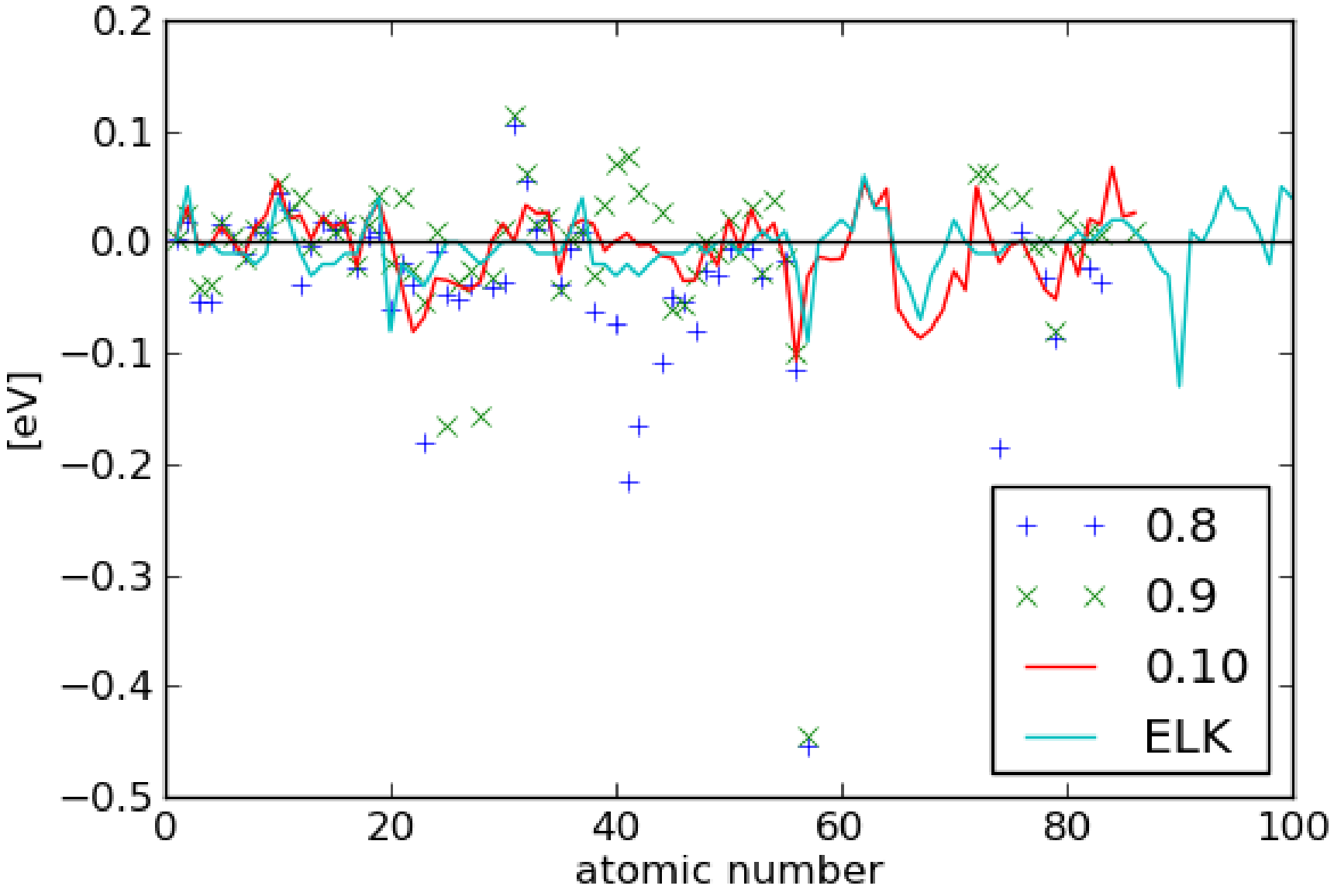
Note

This kind of work is extremely boring

AIMS reference energies



Oxide formation energies relative to AIMS



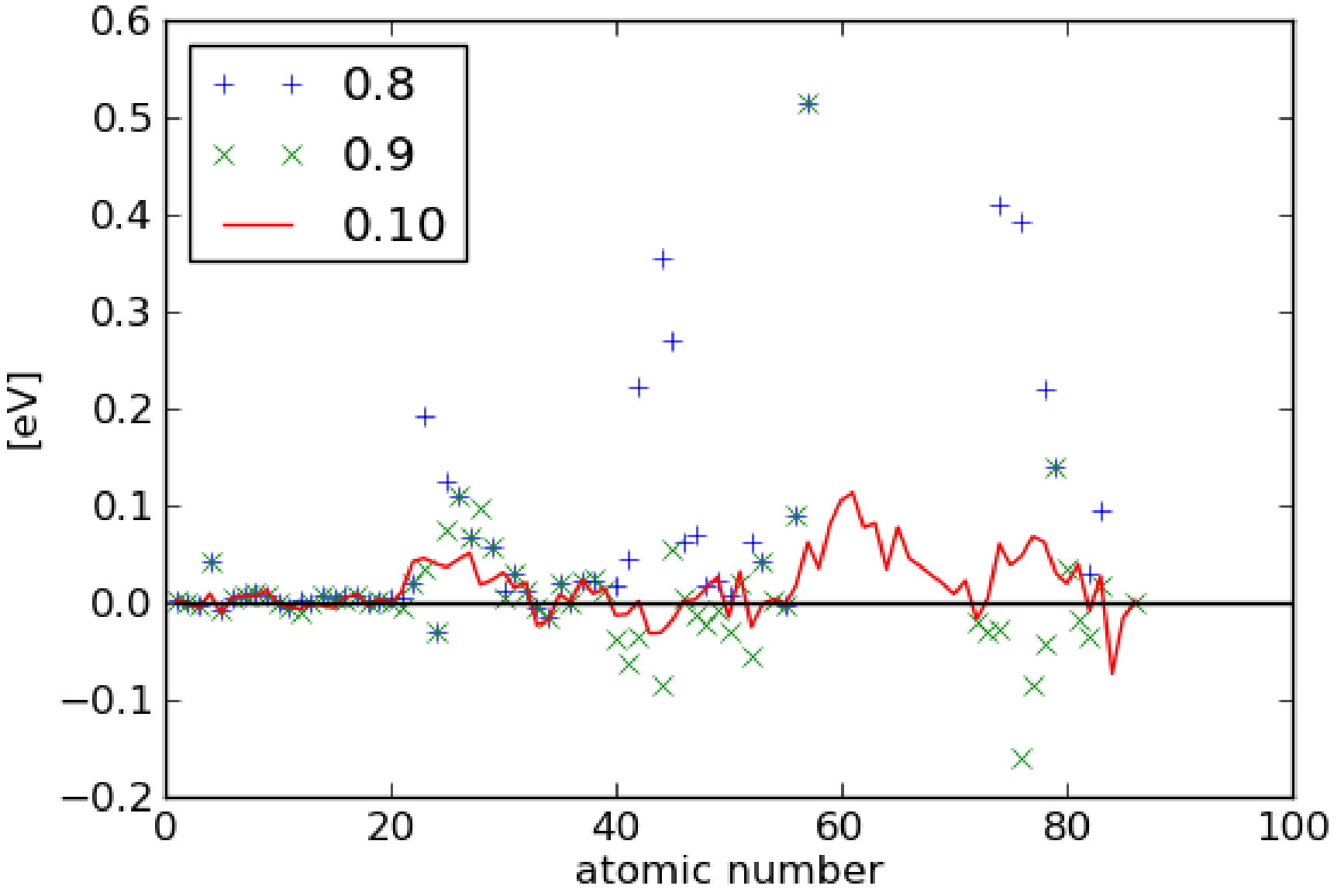
Ruthenium example

Adsorption energies in eV: Ru(001) + $\frac{1}{2}X_2$ - X/Ru(001)

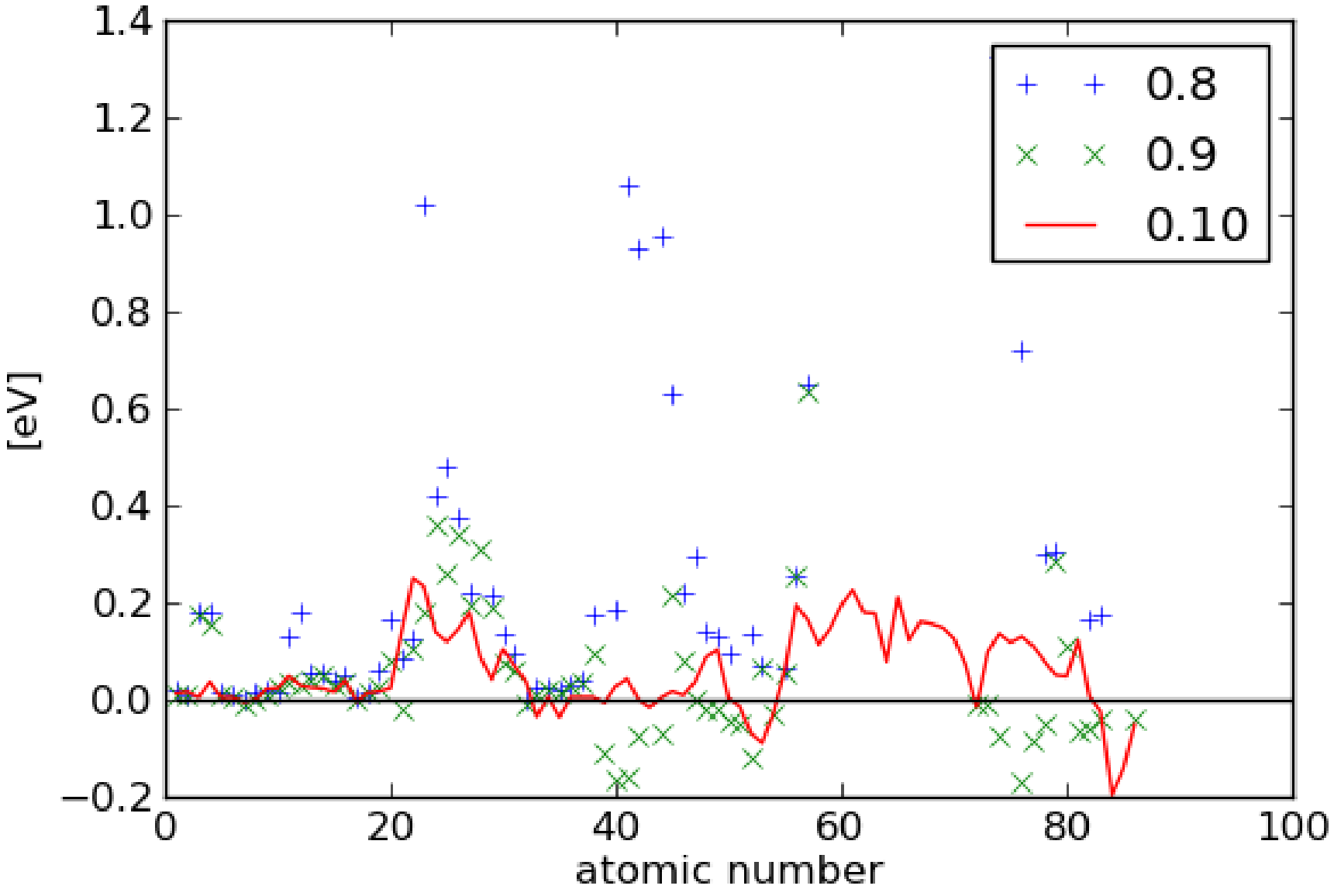
	0.8	0.9	0.10	Gajdoš *et al.*
O/Ru(001)	2.51	2.78	2.78	2.67
N/Ru(001)	0.46	0.88	0.93	0.94
H/Ru(001)	0.54	0.57	0.59	
NO - $\frac{1}{2}N_2$ - $\frac{1}{2}O_2$	0.96	0.95	0.96	0.95

M. Gajdoš, J. Hafner and A. Eichler, J.Phys.: Condens. Matter 18 (2006) 41-54

FCC compression energies relative to AIMS



Rock-salt compression energies relative to AIMS



PAW approximations

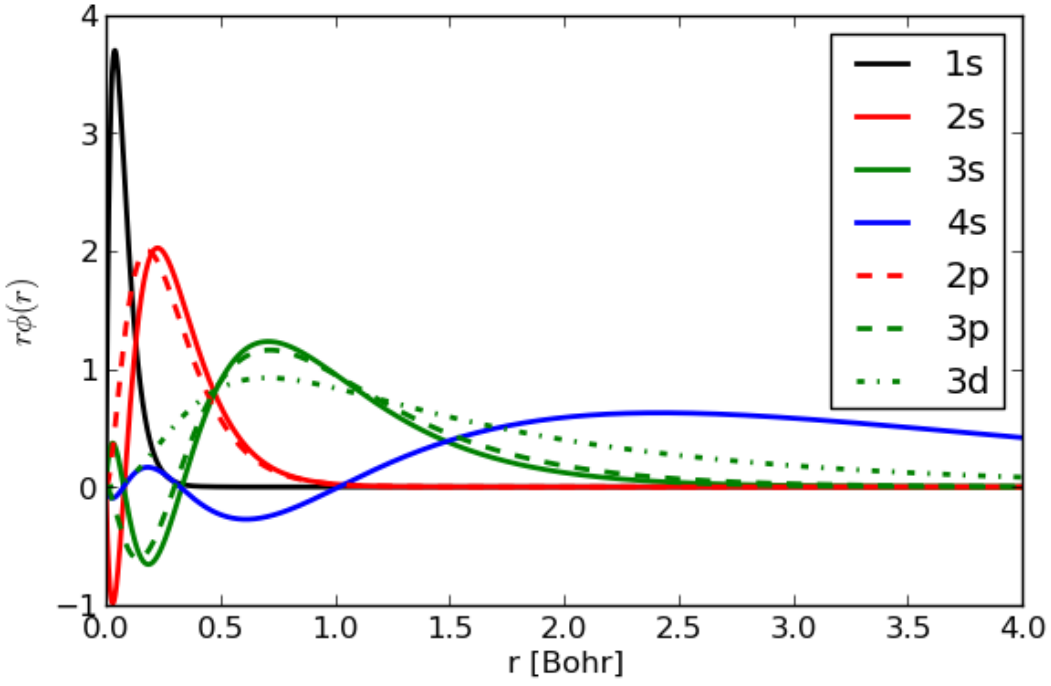
For each element we have to choose:

- which states to freeze and which to include as valence
- the number of projectors, partial waves and pseudo partial waves
- the local potential
- radii for projector functions, local potential and compensation charges

We also need to think about:

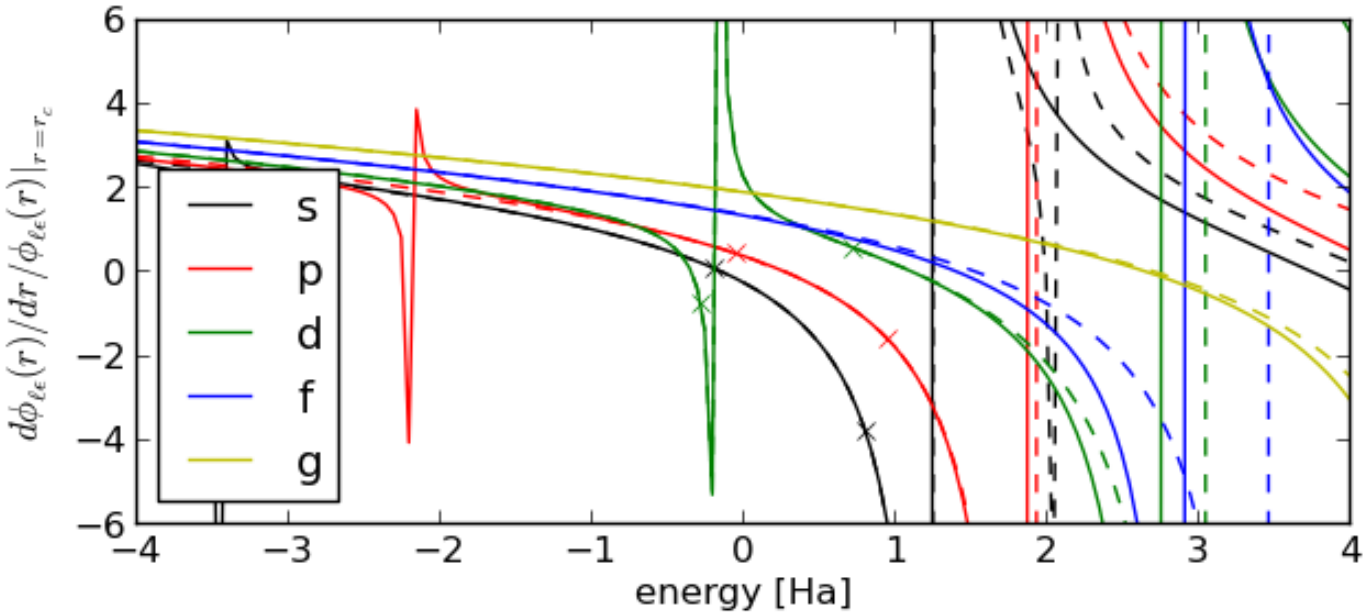
- convergence with respect to number of grid-points/plane-waves
- egg-box errors

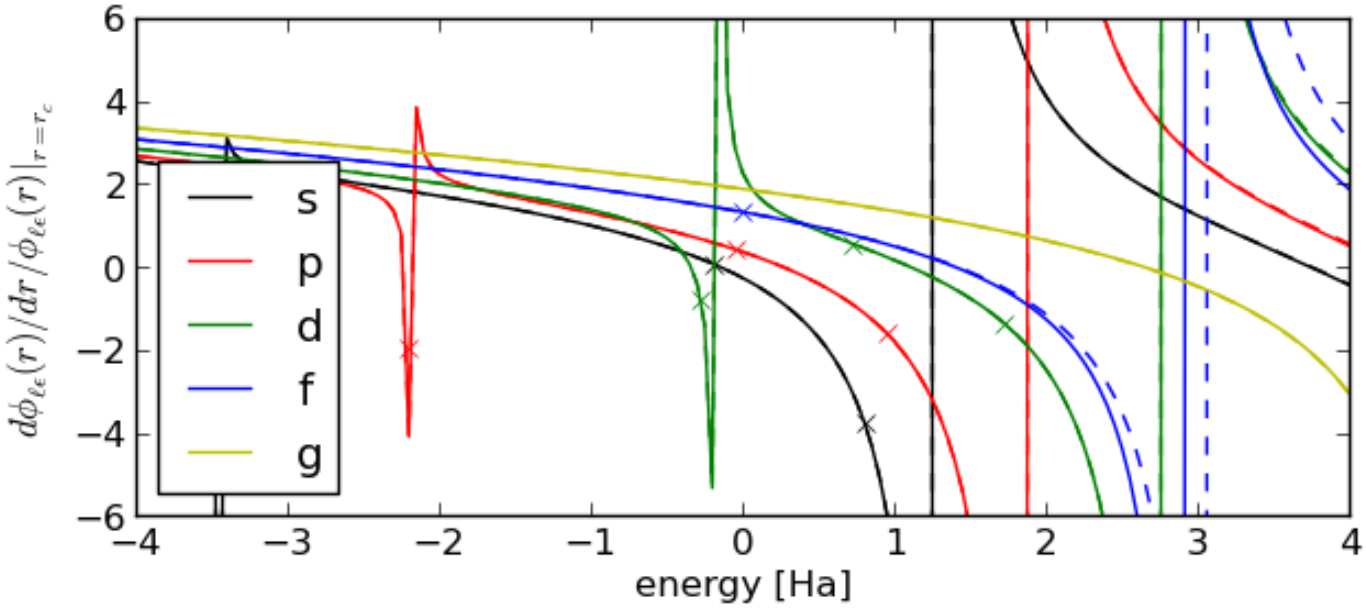
Iron example



	8e	16e	16e+	FHI-aims	ELK
oxide formation	3.58	3.56	3.59	3.60	3.60
fcc compression	1.35	1.32	1.26	1.27	
rock-salt compression	2.25	2.18	2.04	2.04	

Logarithmic derivatives at r=2.3 Bohr (8e and 16e+)





Plane-wave Mode

Advantages:

- Fast for not too large systems
- Fast convergence with respect to number of plane-waves
- Smaller memory footprint
- Simpler density mixing metric and preconditioning
- No egg-box error
- Simple implementation of stress tensor

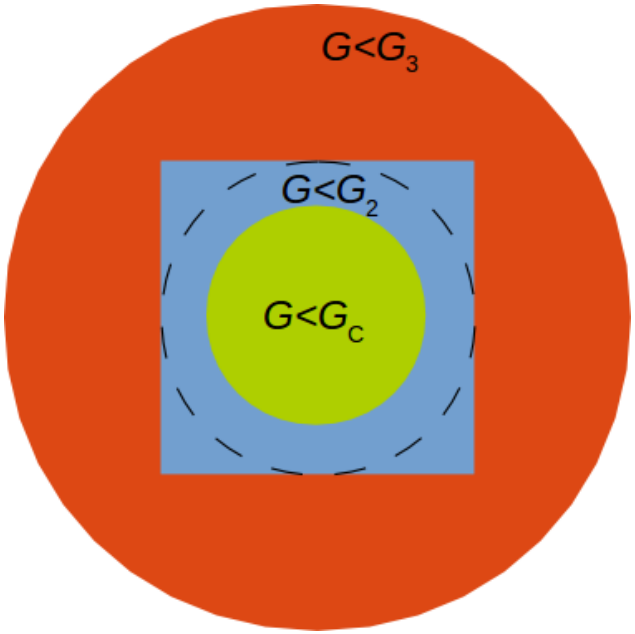
Poisson equation is solved in reciprocal space. Can also be used in lcao and fd mode:

```
calc = GPAW(..., realspace=False, ...)
```

Plane-wave implementation

It's based on FFTW and does the projector wave function overlaps in reciprocal space with ZGEMM.

$$\tilde{\psi}(\mathbf{r}) = \sum_{G<G_c} c_{\mathbf{G}} e^{i\mathbf{G}\cdot\mathbf{r}}.$$



- `mode=PW(ecut=400):ecut=`$\frac{1}{2}G_c^2$
- Zero-pad and do inverse FFT to real-space (grid-spacing: `h=`π/G_2).
- If `h` is not set, we choose $G_2 = \sqrt{2}G_c$ (and not $G_2 = 2G_c$).
- Using FFT's, the electron density is interpolated to a real-space grid of grid-spacing `h`/2 corresponding to $G_3 = 2G_2$.
- There is currently no way to control the value used for G_3.

The future

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: $2^a 3^b 5^c 7^d$

Thursday (May 23)

Activities for GPAW developers (we start at 9:00):

- Coordination of code development and discussions about the future: Quick tour of ongoing projects - what's the current status?
- Introduction to Sphinx and reStructuredText
- Introduction to testing of GPAW
- Hands on: Write new documentation/tutorials and how to make sure they stay up to date
- *Lunch*
- Status of unmerged branches: rpa-gpu-expt, cuda, lcaotddft, lrtddft_indexed, aep1, libxc1.2.0
- Questions open for discussion:
  - 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?
- Hands on: Write new documentation/tutorials --- continued
- Presentations of today's work (we stop at 15:00)

Finally ...

Thank you for your attention,
and don’t forget to give me your pdf file from your talk