QUANTUM ESPRESSO and ASE

Keld, Jess, Joel, AJ, Chris & Johannes

Mar. 6th 2013

Big picture

- We aim at replacing DACAPO as our production code by quantum espresso.
- First results look promising both speed- and accuracy-wise,
- but we are still learning how to get the most out of the code,
- and the ASE-interface to espresso is in an early development stage.

quantum espresso

- Plane wave code.
- Can re-use Dacapo's ultrasofts but also newer (ultrasoft) pseudopotentials and PAW setups.
- Very efficient for small to medium-sized systems (i.e. slabs).
- Fast code; overall stable convergence.
- Advanced features, such electronic excitations, TDDFT, Car-Parrinello MD, DFPT, NMR spectra etc.
- Now also supports BEEF-vdW calculations
  with error estimates
  (and RPBE).

ASE-interface to quantum espresso

Aim: keep it similar to *e.g.* the JACAPO-interface but also support new features.

```
#!/usr/bin/env python
from ase import io
from ase.optimize import QuasiNewton
from espresso import espresso

atoms = io.read('slab.traj')

calc = espresso(pw=350,
                dw=3500,          #high density cut-offs
                kpts=(4,4,1),                #recommended
                nbands=-10,  #ten empty bands gpaw-style
                sigma=0.1,
                xc='BEEF',
                dipole={'status':True},
                outdir='calcdir') #instead of txt output

atoms.set_calculator(calc)

qn = QuasiNewton(atoms, trajectory='opt.traj')
qn.run(fmax=0.05)
```

espresso-internal unit cell relaxation

```
#!/usr/bin/env python
from ase.structure import bulk
from espresso import espresso

atoms = bulk('Mg', 'hcp', a=3.2, covera=1.625)
calc = espresso(pw=500,
                dw=5000,
                xc='pbesol',
                kpts=(16,16,10),
                nbands=-10,
                mode='vc-relax',     #internally relax unit
                cell_dynamics='bfgs',        #cell and atoms
                opt_algorithm='bfgs',
                cell_factor=5.,
                outdir='cellrelax')
atoms.set_calculator(calc)
atoms.get_potential_energy()#trigger launching of espresso
print 'Optimized unit cell:'
print calc.get_final_structure().cell
```

Useful commands

Submit espresso job:

```
esp-ver-bsub 6 -n 8 -o test.log -q suncat-test job.py
```

Check convergence:

```
grep "  total e" calcdir/log
```

Only energies at ionic steps:

```
grep "\! " calcdir/log
```