pyEMTO

A Python based toolkit for EMTO

Henrik Levämäki
University of Turku
Matti Ropo
Tampere University of Technology
COMP Centre of Excellence, Aalto University

Mini-symposium, KTH, Stockholm
10.2.2015

What is pyEMTO?

- Collection of tools which are used to:
  1. Generating EMTO input files and corresponding batch scripts
  2. Running common tasks e.g. calculation of equilibrium volume and elastic constants
  3. Analyzing the results
- Written in Python
- Similar in concept to the Atomic Simulation Environment (ASE)

Why Python?

- Easy to write and learn
- $\Rightarrow$ Development with Python very fast
- Python interacts very well with Unix shell
- EMTO calculations controlled by Python scripts:
  1. Integration with the cluster's job scheduler (currently only SLURM implemented)
  2. Generating large amounts of input files becomes faster
  3. High-level algorithms possible (automatically find eq. volume etc.)

Motivation

Development began when we needed a tool that can generate hundreds/thousands of EMTO input files as easily as possible.

$\Rightarrow$ Doing the same by hand not so much fun.

The goal is to minimize time spent on "repetitive" tasks:

- Automate things that can be reliably automated
- Avoid typos in input files

FEATURES

Input file generation:

- Every EMTO sub-program has been implemented:
  - BMDL, KSTR, SHAPE, KGRN and KFCD
- Every possible EMTO input parameter is accessible via the "set_values" function:

```
some_system.emto.set_values(amix=0.02,
                            afm='M',
                            iex=10,
                            dx=0.015)
```

- All parameters have sensible default values (if possible)

FEATURES

Input file generation:

Example: Create input files for an hcp c/a -grid

```
latnames = ['hcp_ca1','hcp_ca2','hcp_ca3','hcp_ca4',
            'hcp_ca5','hcp_ca6','hcp_ca7']

dmaxs    = [2.39196429,2.41107143,2.43017857,2.44928571,
            2.46839286,2.4875,2.50660714]

for i in range(len(latnames)):
    structure.lattice.set_values(jobname=latnames[i],latpath=latpath,
                                 lat='hcp',kappaw=[0.0,-20.0],msgl=0,
                                 ca=cas[i],dmax=dmaxs[i])

    structure.lattice.bmdl.write_input_file(folder=latpath)
    structure.lattice.kstr.write_input_file(folder=latpath)
    structure.lattice.shape.write_input_file(folder=latpath)
    structure.lattice.batch.write_input_file(folder=latpath)
```

FEATURES

SLURM integration:

- pyEMTO knows when a specific task has finished running:

```
Submitted batch job 1021841
Submitted batch job 1021842
Submitted batch job 1021843
Submitted batch job 1021844
Submitted batch job 1021845
Submitted batch job 1021846

wait_for_jobs: Submitted 6 jobs
wait_for_jobs: Will be requesting job statuses every 60 seconds

0:01:00 {'RUNNING': 6} (  0% completion)
0:02:00 {'RUNNING': 6} (  0% completion)
0:03:00 {'RUNNING': 6} (  0% completion)
0:04:00 {'COMPLETED': 4, 'RUNNING': 2} ( 66% completion)
0:05:00 {'COMPLETED': 6} (100% completion)
completed 6 batch jobs in 0:05:00
```

FEATURES

SLURM integration:

- This feature can be used to join tasks to form more complicated operations:

```
# Calculate elastic constants based
# on the results of EOS calculations:

sws0,ca0,B0,e0,R0,cs0 =
ti_hcp.lattice_constants_batch_calculate(sws=sws_array)

# The script will wait here until the EOS calculations finish

ti_hcp.elastic_constants_batch_calculate(sws=sws0,bmod=B0,ca=ca0,
                                         R=R0,cs=cs0)
```

Features

Result analysis:

- High-quality EOS fitting module:
  - Morse, Murnaghan, Birch-Murnaghan, SJEOS, Vinet, Pourier-Tarantola, Anton-Schmidt, Taylor series, Polynomial
  - 2nd order polynomial fit for initial values increases robustness

FEATURES

Result analysis:

- High-quality EOS fitting module:

```
5.2.2015 -- 16:21:13
JOBNAM = fe1.00 -- PBE

Using morse function

Chi squared         = 2.7459052408e-10
Reduced Chi squared = 1.3729526204e-10
R squared           = 0.999907593903

morse parameters:

a      =     0.117551
b      =  -122.663728
c      = 32000.491593
lambda =     2.370150

Ground state parameters:

V0           =     2.640005 Bohr^3 (unit cell volume)
             =     2.640005 Bohr   (WS-radius)
E0           = -2545.606678 Ry
Bmod         =   195.204183 GPa
Grun. param. =     3.128604

sws            Einp           Eout        Residual       err (% * 10**6)
2.600000  -2545.605517  -2545.605515      0.000002       -0.000870
2.620000  -2545.606394  -2545.606400     -0.000006        0.002524
2.640000  -2545.606680  -2545.606678      0.000002       -0.000950
```

FEATURES

Result analysis:

- Obtain elastic constants:

```
***cubic_elastic_constants***

fe1.00

c11(GPa) = 299.60
c12(GPa) = 142.70
c44(GPa) = 105.95
c' (GPa) =  78.45
B  (GPa) = 195.00

Voigt average:

BV(GPa)  = 195.00
GV(GPa)  =  94.95
EV(GPa)  = 245.07
vV(GPa)  =   0.29

Reuss average:

BR(GPa)  = 195.00
GR(GPa)  =  92.92
ER(GPa)  = 240.55
vR(GPa)  =   0.29

Hill average:

BH(GPa)  = 195.00
GH(GPa)  =  93.93
```

Features

Result analysis:

- Utility functions for plotting:
  - EOS-curve + data points
  - Magnetic moments vs. volume
  - etc.

Now a brief real-time demonstration