pyEMTO Documentation

Release 0.9.0

Henrik Levämäki, Matti Ropo

Dec 01, 2020

Contents

Note: This is pyEMTO's html documentation. pyEMTO is a toolkit used to:

1. **Easily generate input files and corresponding batch scripts for the EMTO (Exact Muffin-Tin Orbitals) solid-state physics software.**
2. **Automating routine tasks such as calculation of ground state volume and elastic constants.**
3. **Analysing the results.**

Development of this project started when the authors needed a fast and convenient way to generate large amounts of input files for various different types of calculations with the EMTO software. The design goal of pyEMTO is to minimise time spent on writing input files and other "repetitive" tasks in the most convenient and reliable way possible.

Contents:

CHAPTER 1

Features

- Written in Python.
- Similar in concept to the Atomic Simulation Environment (ASE).
- Integration with the cluster's job scheduler (currently only SLURM implemented).
- Generating large amounts of input files becomes easy.
- EMTO calculations controlled by Python language:
  - Full control over everything
- The goal is to minimise time spent on things that can reliably be automated.
- High-quality EOS module for Equation Of State fitting:
  - Morse
  - Murnaghan
  - Birch-Murnaghan
  - SJEOS
  - Vinet
  - Pourier-Tarantola
  - Anton-Schmidt
  - Taylor series
  - Polynomial fit
- Built-in functions for result analysis:
  - Determine ground state lattice constants, bulk moduli, energy and elastic constants.

CHAPTER 2

Installation

Install by running

```
pip install pyemto
```

or

```
git clone https://github.com/hpleva/pyemto.git
cd pyemto
python setup.py install
```

Optional: Try running some of the scripts from the *examples* page to verify the integrity of your pyEMTO installation.

CHAPTER 3

Example scripts

Copy-paste the text blocks below and save them as .py files to try them out.

3.1 Ground state volume of bcc bulk iron

```
#!/usr/bin/python

# An example script of how to automatically
# calculate the equilibrium volume and
# bulk modulus of bulk bcc Fe using pyEMTO.

import pyemto

emtopath = "/wrk/hpleva/pyEMTO_examples/fe" # Define a folder for our KGRN and KFCD␣
↪input and output files.
latpath  = "/wrk/hpleva/structures"         # Define a folder where the BMDL, KSTR␣
↪and KFCD input and output files
                                            # will be located.

fe = pyemto.System(folder=emtopath)  # Create a new instance of the pyemto System-
↪class.

# Let's calculate the equilibrium volume and bulk modulus of Fe.

# Initialize the system using the bulk.() function:

fe.bulk(lat     = 'bcc',    # We want to use the bcc structure.
        latpath = latpath, # Path to the folder where the structure files are located.
        afm     = 'F',      # We want to do a ferromagnetic calculation.
        atoms   = ['Fe'],   # A list of atoms.
        splts   = [2.0],    # A list of magnetic splittings.
        expan   = 'M',      # We want to use the double-Taylor expansion.
        sofc    = 'Y',      # We want to use soft-core approximation.
```

(continues on next page)

(continued from previous page)

```
        xc      = 'PBE',    # We want to use the PBE functional.
        amix    = 0.05,     # Density mixing.
        nky     = 21)       # Number of k-points.

sws = [2.6,2.62,2.64,2.66,2.68,2.70] # A list of Wigner-Seitz radii

# Generate all the necessary input files with this function:
#fe.lattice_constants_batch_generate(sws=sws)

#The newly created batch scripts are then submitted by hand.

# Analyze the results using this function once all the calculations
# have finished:
#fe.lattice_constants_analyze(sws=sws)

# This function combines the features of the previous two:
fe.lattice_constants_batch_calculate(sws=sws)
```

The output of this script should look like this:

```
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

*****lattice_constants_analyze*****



********************************
lattice_constants_analyze(cubic)
********************************
  SWS         Energy
2.600000  -2545.605517
2.620000  -2545.606394
2.640000  -2545.606680
2.660000  -2545.606435
2.680000  -2545.605705
2.700000  -2545.604616


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

Using morse function
```

(continues on next page)

(continued from previous page)

```
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
2.660000  -2545.606435  -2545.606426      0.000009       -0.003632
2.680000  -2545.605705  -2545.605716     -0.000011        0.004368
2.700000  -2545.604616  -2545.604612      0.000004       -0.001440


lattice_constants_analyze(cubic):
sws0 =      2.640005
B0   =    195.204183
E0   =  -2545.606678
```

3.2 Elastic constants of bcc bulk iron

```
#!/usr/bin/python

# An example script showing how to automatically calculate
# the elastic constants of bulk bcc Fe using pyEMTO.

import pyemto

emtopath = "/wrk/hpleva/pyEMTO_examples/fe_elastic_constants" # Define a folder for␣
↪our KGRN and KFCD input and output files.
latpath  = "/wrk/hpleva/structures"                           # Define a folder where␣
↪the BMDL, KSTR and KFCD input and output files
                                                              # will be located.

fe = pyemto.System(folder=emtopath)  # Create a new instance of the pyemto System-
↪class.

# Let's calculate the elastic constants of Fe.
```

(continues on next page)

(continued from previous page)

```
fe.bulk(lat     = 'bcc',    # We want to use the bcc structure.
        latpath = latpath, # Path to the folder where the structure files are located.
        afm     = 'F',      # We want to do a ferromagnetic calculation.
        atoms   = ['Fe'],   # A list of atoms.
        splts   = [2.0],    # A list of magnetic splittings.
        expan   = 'M',      # We want to use the double-Taylor expansion.
        sofc    = 'Y',      # We want to use soft-core approximation.
        xc      = 'PBE',    # We want to use the PBE functional.
        amix    = 0.05)     # Mixing parameter.

sws0 = 2.64 # Eq. WS-radius that we computed previously.
B0   = 195  # Eq. bulk modulus.

# Generate all the necessary input files with this function:
#fe.elastic_constants_batch_generate(sws=sws0)

#The newly created batch scripts are then submitted by hand.

# Analyze the results using this function once all the calculations
# have finished:
#fe.elastic_constants_analyze(sws=sws,bmod=B0)

# This function combines the features of the previous two:
fe.elastic_constants_batch_calculate(sws=sws0,bmod=B0)
```

The output of this script should look like this:

```
Submitted batch job 1021848
Submitted batch job 1021849
Submitted batch job 1021850
Submitted batch job 1021851
Submitted batch job 1021852
Submitted batch job 1021853
Submitted batch job 1021854
Submitted batch job 1021855
Submitted batch job 1021856
Submitted batch job 1021857
Submitted batch job 1021858
Submitted batch job 1021859

wait_for_jobs: Submitted 12 jobs
wait_for_jobs: Will be requesting job statuses every 60 seconds

0:01:00 {'RUNNING': 12} (  0% completion)
0:02:00 {'RUNNING': 12} (  0% completion)
0:03:00 {'RUNNING': 12} (  0% completion)
0:04:00 {'RUNNING': 12} (  0% completion)
0:05:00 {'RUNNING': 12} (  0% completion)
0:06:00 {'RUNNING': 12} (  0% completion)
0:07:00 {'RUNNING': 12} (  0% completion)
0:08:01 {'RUNNING': 12} (  0% completion)
0:09:01 {'RUNNING': 12} (  0% completion)
0:10:01 {'RUNNING': 12} (  0% completion)
0:11:01 {'RUNNING': 12} (  0% completion)
0:12:01 {'RUNNING': 12} (  0% completion)
0:13:01 {'RUNNING': 12} (  0% completion)
0:14:01 {'RUNNING': 12} (  0% completion)
```

(continues on next page)

(continued from previous page)

```
0:15:01 {'RUNNING': 12} (  0% completion)
0:16:01 {'RUNNING': 12} (  0% completion)
0:17:02 {'RUNNING': 12} (  0% completion)
0:18:02 {'RUNNING': 12} (  0% completion)
0:19:02 {'RUNNING': 12} (  0% completion)
0:20:02 {'RUNNING': 12} (  0% completion)
0:21:02 {'RUNNING': 12} (  0% completion)
0:22:02 {'RUNNING': 12} (  0% completion)
0:23:02 {'RUNNING': 12} (  0% completion)
0:24:02 {'RUNNING': 12} (  0% completion)
0:25:03 {'RUNNING': 12} (  0% completion)
0:26:03 {'RUNNING': 12} (  0% completion)
0:27:03 {'RUNNING': 12} (  0% completion)
0:28:03 {'RUNNING': 12} (  0% completion)
0:29:03 {'RUNNING': 12} (  0% completion)
0:30:03 {'RUNNING': 12} (  0% completion)
0:31:03 {'RUNNING': 12} (  0% completion)
0:32:03 {'RUNNING': 12} (  0% completion)
0:33:03 {'RUNNING': 12} (  0% completion)
0:34:04 {'RUNNING': 12} (  0% completion)
0:35:04 {'RUNNING': 12} (  0% completion)
0:36:04 {'RUNNING': 12} (  0% completion)
0:37:04 {'RUNNING': 12} (  0% completion)
0:38:04 {'RUNNING': 12} (  0% completion)
0:39:04 {'RUNNING': 12} (  0% completion)
0:40:04 {'RUNNING': 12} (  0% completion)
0:41:04 {'RUNNING': 12} (  0% completion)
0:42:05 {'RUNNING': 12} (  0% completion)
0:43:05 {'RUNNING': 12} (  0% completion)
0:44:05 {'RUNNING': 12} (  0% completion)
0:45:05 {'RUNNING': 12} (  0% completion)
0:46:05 {'RUNNING': 12} (  0% completion)
0:47:05 {'RUNNING': 12} (  0% completion)
0:48:05 {'RUNNING': 12} (  0% completion)
0:49:05 {'RUNNING': 12} (  0% completion)
0:50:05 {'RUNNING': 12} (  0% completion)
0:51:06 {'RUNNING': 12} (  0% completion)
0:52:06 {'RUNNING': 12} (  0% completion)
0:53:06 {'RUNNING': 12} (  0% completion)
0:54:06 {'RUNNING': 12} (  0% completion)
0:55:06 {'RUNNING': 12} (  0% completion)
0:56:06 {'RUNNING': 12} (  0% completion)
0:57:06 {'RUNNING': 12} (  0% completion)
0:58:06 {'RUNNING': 12} (  0% completion)
0:59:07 {'RUNNING': 11, 'COMPLETED': 1} (  8% completion)
1:00:07 {'RUNNING': 10, 'COMPLETED': 2} ( 16% completion)
1:01:07 {'RUNNING': 9, 'COMPLETED': 3} ( 25% completion)
1:02:07 {'RUNNING': 9, 'COMPLETED': 3} ( 25% completion)
1:03:07 {'RUNNING': 9, 'COMPLETED': 3} ( 25% completion)
1:04:07 {'RUNNING': 9, 'COMPLETED': 3} ( 25% completion)
1:05:07 {'RUNNING': 9, 'COMPLETED': 3} ( 25% completion)
1:06:07 {'RUNNING': 9, 'COMPLETED': 3} ( 25% completion)
1:07:08 {'RUNNING': 9, 'COMPLETED': 3} ( 25% completion)
1:08:08 {'RUNNING': 8, 'COMPLETED': 4} ( 33% completion)
1:09:08 {'COMPLETED': 6, 'RUNNING': 6} ( 50% completion)
1:10:08 {'COMPLETED': 6, 'RUNNING': 6} ( 50% completion)
1:11:08 {'COMPLETED': 6, 'RUNNING': 6} ( 50% completion)
```

(continues on next page)

(continued from previous page)

```
1:12:08 {'COMPLETED': 6, 'RUNNING': 6} ( 50% completion)
1:13:08 {'COMPLETED': 6, 'RUNNING': 6} ( 50% completion)
1:14:08 {'COMPLETED': 6, 'RUNNING': 6} ( 50% completion)
1:15:09 {'COMPLETED': 6, 'RUNNING': 6} ( 50% completion)
1:16:09 {'COMPLETED': 6, 'RUNNING': 6} ( 50% completion)
1:17:09 {'COMPLETED': 6, 'RUNNING': 6} ( 50% completion)
1:18:09 {'COMPLETED': 6, 'RUNNING': 6} ( 50% completion)
1:19:09 {'COMPLETED': 7, 'RUNNING': 5} ( 58% completion)
1:20:09 {'COMPLETED': 7, 'RUNNING': 5} ( 58% completion)
1:21:09 {'COMPLETED': 8, 'RUNNING': 4} ( 66% completion)
1:22:09 {'COMPLETED': 10, 'RUNNING': 2} ( 83% completion)
1:23:10 {'COMPLETED': 10, 'RUNNING': 2} ( 83% completion)
1:24:10 {'COMPLETED': 10, 'RUNNING': 2} ( 83% completion)
1:25:10 {'COMPLETED': 11, 'RUNNING': 1} ( 91% completion)
1:26:10 {'COMPLETED': 12} (100% completion)
completed 12 batch jobs in 1:26:10

***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
EH(GPa)  = 242.81
vH(GPa)  =   0.29

Elastic anisotropy:

AVR(GPa)  =   0.01
```

3.3 Ground state volume and elastic constants of hcp bulk titanium

```
#!/usr/bin/python

# This script will automatically accomplish:

#  1. Calculate the equilibrium volume and bulk modulus of hcp Ti.
#  2. Calculate the elastic constants of hcp Ti.

import pyemto
import os
import numpy as np

# It is recommended to always use absolute paths
folder = os.getcwd()                # Get current working directory.
latpath = "/wrk/hpleva/structures"  # Folder where the structure files are located.
emtopath = folder+"/ti_hcp"         # Folder where the calculation take place.

ti_hcp=pyemto.System(folder=emtopath)

# Initialize the bulk system using the bulk() function:
ti_hcp.bulk(lat='hcp',
            latpath=latpath,
            atoms=['Ti'],
            sws=3.0,
            amix=0.02,
            efmix=0.9,
            expan='M',
            sofc='Y',
            xc='P07',           # Use PBEsol
            nky=31,             # k-points
            nkz=19,             # k-points
            runtime='24:00:00') # Allow large enough timelimit for SLURM

sws = np.linspace(2.9,3.1,7) # A list of 7 different volumes from 2.9 to 3.1


sws0,ca0,B0,e0,R0,cs0 = ti_hcp.lattice_constants_batch_calculate(sws=sws)
ti_hcp.elastic_constants_batch_calculate(sws=sws0,bmod=B0,ca=ca0,R=R0,cs=cs0)

# If the batch jobs are submitted by hand use these functions.

# To evaluate the results, comment out the _generate functions
# and uncomment the _analyze functions.

ti_hcp.lattice_constants_batch_generate(sws=sws)
#ti_hcp.lattice_constants_analyze(sws=sws)

# Results. These are inputed to the elastic_constants functions.
#sws0 =      3.002260
#ca0  =      1.610122
#B0   =    115.952318
#E0   =  -1705.738844
#R0   =      0.019532
#cs0  =    498.360422

#ti_hcp.elastic_constants_batch_generate(sws=sws0,ca=ca0)
```

(continues on next page)

(continued from previous page)

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

The output of this script should look like this:

```
Submitted batch job 1021973
Submitted batch job 1021974
Submitted batch job 1021975
Submitted batch job 1021976
Submitted batch job 1021977
Submitted batch job 1021978
Submitted batch job 1021979
Submitted batch job 1021980
Submitted batch job 1021981
Submitted batch job 1021982
Submitted batch job 1021983
Submitted batch job 1021984
Submitted batch job 1021985
Submitted batch job 1021986
Submitted batch job 1021987
Submitted batch job 1021988
Submitted batch job 1021989
Submitted batch job 1021990
Submitted batch job 1021991
Submitted batch job 1021992
Submitted batch job 1021993
Submitted batch job 1021994
Submitted batch job 1021995
Submitted batch job 1021996
Submitted batch job 1021997
Submitted batch job 1021998
Submitted batch job 1021999
Submitted batch job 1022000
Submitted batch job 1022001
Submitted batch job 1022002
Submitted batch job 1022003
Submitted batch job 1022004
Submitted batch job 1022005
Submitted batch job 1022006
Submitted batch job 1022007
Submitted batch job 1022008
Submitted batch job 1022009
Submitted batch job 1022010
Submitted batch job 1022011
Submitted batch job 1022012
Submitted batch job 1022013
Submitted batch job 1022014
Submitted batch job 1022015
Submitted batch job 1022016
Submitted batch job 1022017
Submitted batch job 1022018
Submitted batch job 1022019
Submitted batch job 1022020
Submitted batch job 1022021
()
wait_for_jobs: Submitted 49 jobs
wait_for_jobs: Will be requesting job statuses every 60 seconds
```

(continues on next page)

(continued from previous page)

```
0:01:00 {'RUNNING': 49} (  0% completion)
0:02:00 {'RUNNING': 49} (  0% completion)
0:03:00 {'RUNNING': 49} (  0% completion)
0:04:00 {'RUNNING': 49} (  0% completion)
0:05:01 {'RUNNING': 49} (  0% completion)
0:06:01 {'RUNNING': 49} (  0% completion)
0:07:01 {'RUNNING': 49} (  0% completion)
0:08:01 {'RUNNING': 49} (  0% completion)
0:09:02 {'RUNNING': 49} (  0% completion)
0:10:02 {'RUNNING': 49} (  0% completion)
0:11:02 {'RUNNING': 49} (  0% completion)
0:12:02 {'RUNNING': 49} (  0% completion)
0:13:02 {'RUNNING': 49} (  0% completion)
0:14:02 {'RUNNING': 49} (  0% completion)
0:15:02 {'RUNNING': 49} (  0% completion)
0:16:03 {'RUNNING': 49} (  0% completion)
0:17:03 {'RUNNING': 49} (  0% completion)
0:18:03 {'RUNNING': 49} (  0% completion)
0:19:03 {'RUNNING': 49} (  0% completion)
0:20:03 {'RUNNING': 49} (  0% completion)
0:21:03 {'RUNNING': 49} (  0% completion)
0:22:03 {'RUNNING': 49} (  0% completion)
0:23:03 {'RUNNING': 49} (  0% completion)
0:24:04 {'RUNNING': 49} (  0% completion)
0:25:04 {'RUNNING': 49} (  0% completion)
0:26:04 {'RUNNING': 48, 'COMPLETED': 1} (  2% completion)
0:27:04 {'RUNNING': 47, 'COMPLETED': 2} (  4% completion)
0:28:04 {'RUNNING': 46, 'COMPLETED': 3} (  6% completion)
0:29:04 {'RUNNING': 42, 'COMPLETED': 7} ( 14% completion)
0:30:04 {'RUNNING': 40, 'COMPLETED': 9} ( 18% completion)
0:31:05 {'COMPLETED': 12, 'RUNNING': 37} ( 24% completion)
0:32:05 {'COMPLETED': 15, 'RUNNING': 34} ( 30% completion)
0:33:05 {'COMPLETED': 15, 'RUNNING': 34} ( 30% completion)
0:34:05 {'COMPLETED': 15, 'RUNNING': 34} ( 30% completion)
0:35:05 {'COMPLETED': 18, 'RUNNING': 31} ( 36% completion)
0:36:05 {'COMPLETED': 29, 'RUNNING': 20} ( 59% completion)
0:37:05 {'COMPLETED': 44, 'RUNNING': 5} ( 89% completion)
0:38:06 {'COMPLETED': 49} (100% completion)
completed 49 batch jobs in 0:38:06

*****lattice_constants_analyze*****



*****************************
lattice_constants_analyze(hcp)
*****************************
  SWS        Energy0        c'a0
2.900000  -1705.733796  1.613939
2.933333  -1705.736608  1.612527
2.966667  -1705.738262  1.611196
3.000000  -1705.738843  1.610175
3.033333  -1705.738424  1.609326
3.066667  -1705.737080  1.608336
3.100000  -1705.734889  1.607505
```

(continues on next page)

(continued from previous page)

```
5.2.2015 -- 18:51:51
JOBNAM = ti1.00 -- P07

Using morse function

Chi squared         = 1.16375610448e-11
Reduced Chi squared = 3.87918701493e-12
R squared           = 0.999999464309

morse parameters:

a      =     0.757678
b      =   -15.168890
c      =    75.921091
lambda =     0.767287

Ground state parameters:

V0           =     3.002260 Bohr^3 (unit cell volume)
             =     3.002260 Bohr   (WS-radius)
E0           = -1705.738844 Ry
Bmod         =   115.952318 GPa
Grun. param. =     1.151798

sws            Einp           Eout          Residual       err (% * 10**6)
2.900000  -1705.733796  -1705.733796      0.000000       -0.000275
2.933333  -1705.736608  -1705.736609     -0.000001        0.000498
2.966667  -1705.738262  -1705.738263     -0.000001        0.000376
3.000000  -1705.738843  -1705.738842      0.000001       -0.000810
3.033333  -1705.738424  -1705.738423      0.000001       -0.000680
3.066667  -1705.737080  -1705.737082     -0.000002        0.001450
3.100000  -1705.734889  -1705.734889      0.000001       -0.000558


5.2.2015 -- 18:51:51
JOBNAM = ti1.00 -- P07

Using morse function

Chi squared         = 1.06749728331e-10
Reduced Chi squared = 3.5583242777e-11
R squared           = 0.999995355882

morse parameters:

a      =     0.748045
b      =   -15.100334
c      =    76.203651
lambda =     0.769114

Ground state parameters:

V0           =     3.005849 Bohr^3 (unit cell volume)
             =     3.005849 Bohr   (WS-radius)
E0           = -1705.736830 Ry
Bmod         =   114.888885 GPa
Grun. param. =     1.155920
```

(continues on next page)

(continued from previous page)

```
sws            Einp           Eout          Residual       err (% * 10**6)
2.900000  -1705.731449  -1705.731449     -0.000000        0.000021
2.933333  -1705.734368  -1705.734369     -0.000001        0.000770
2.966667  -1705.736134  -1705.736130      0.000004       -0.002260
3.000000  -1705.736814  -1705.736815     -0.000001        0.000665
3.033333  -1705.736497  -1705.736503     -0.000006        0.003550
3.066667  -1705.735275  -1705.735268      0.000007       -0.004035
3.100000  -1705.733178  -1705.733180     -0.000002        0.001288


5.2.2015 -- 18:51:51
JOBNAM = ti1.00 -- P07

Using morse function

Chi squared         = 3.64530335255e-11
Reduced Chi squared = 1.21510111752e-11
R squared           = 0.999998386204

morse parameters:

a      =     0.743238
b      =   -15.271044
c      =    78.441514
lambda =     0.775451

Ground state parameters:

V0           =     3.004114 Bohr^3 (unit cell volume)
             =     3.004114 Bohr   (WS-radius)
E0           = -1705.737897 Ry
Bmod         =   116.104730 GPa
Grun. param. =     1.164771

sws            Einp           Eout          Residual       err (% * 10**6)
2.900000  -1705.732644  -1705.732643      0.000001       -0.000875
2.933333  -1705.735527  -1705.735531     -0.000004        0.002534
2.966667  -1705.737255  -1705.737252      0.000003       -0.001704
3.000000  -1705.737891  -1705.737890      0.000001       -0.000729
3.033333  -1705.737524  -1705.737524     -0.000000        0.000183
3.066667  -1705.736229  -1705.736231     -0.000002        0.001216
3.100000  -1705.734082  -1705.734081      0.000001       -0.000625


5.2.2015 -- 18:51:51
JOBNAM = ti1.00 -- P07

Using morse function

Chi squared         = 1.18175890217e-11
Reduced Chi squared = 3.93919634056e-12
R squared           = 0.999999463488

morse parameters:

a      =     0.759517
```

(continues on next page)

(continued from previous page)

```
b      =   -15.168958
c      =    75.737639
lambda =     0.766269

Ground state parameters:

V0           =     3.003084 Bohr^3 (unit cell volume)
             =     3.003084 Bohr   (WS-radius)
E0           = -1705.738555 Ry
Bmod         =   115.894284 GPa
Grun. param. =     1.150586

sws            Einp           Eout          Residual       err (% * 10**6)
2.900000  -1705.733423  -1705.733423     -0.000000        0.000259
2.933333  -1705.736267  -1705.736265      0.000002       -0.001035
2.966667  -1705.737944  -1705.737946     -0.000002        0.001386
3.000000  -1705.738551  -1705.738550      0.000001       -0.000391
3.033333  -1705.738157  -1705.738156      0.000001       -0.000679
3.066667  -1705.736836  -1705.736837     -0.000001        0.000606
3.100000  -1705.734664  -1705.734664      0.000000       -0.000147


5.2.2015 -- 18:51:51
JOBNAM = ti1.00 -- P07

Using morse function

Chi squared         = 5.45131762039e-12
Reduced Chi squared = 1.81710587346e-12
R squared           = 0.999999749818

morse parameters:

a      =     0.772765
b      =   -15.129144
c      =    74.049208
lambda =     0.759794

Ground state parameters:

V0           =     3.002465 Bohr^3 (unit cell volume)
             =     3.002465 Bohr   (WS-radius)
E0           = -1705.738834 Ry
Bmod         =   115.954728 GPa
Grun. param. =     1.140627

sws            Einp           Eout          Residual       err (% * 10**6)
2.900000  -1705.733769  -1705.733768      0.000001       -0.000320
2.933333  -1705.736585  -1705.736586     -0.000001        0.000827
2.966667  -1705.738247  -1705.738246      0.000001       -0.000325
3.000000  -1705.738832  -1705.738831      0.000001       -0.000454
3.033333  -1705.738419  -1705.738419      0.000000       -0.000175
3.066667  -1705.737081  -1705.737082     -0.000001        0.000791
3.100000  -1705.734892  -1705.734891      0.000001       -0.000345


5.2.2015 -- 18:51:52
```

(continues on next page)

(continued from previous page)

```
JOBNAM = ti1.00 -- P07

Using morse function

Chi squared         = 1.19109949001e-11
Reduced Chi squared = 3.97033163337e-12
R squared           = 0.999999449336

morse parameters:

a      =     0.745414
b      =   -15.182081
c      =    77.304565
lambda =     0.773045

Ground state parameters:

V0           =     3.002133 Bohr^3 (unit cell volume)
             =     3.002133 Bohr   (WS-radius)
E0           = -1705.738728 Ry
Bmod         =   115.798751 GPa
Grun. param. =     1.160392

sws            Einp           Eout          Residual       err (% * 10**6)
2.900000  -1705.733697  -1705.733697      0.000000       -0.000064
2.933333  -1705.736504  -1705.736504      0.000000       -0.000199
2.966667  -1705.738150  -1705.738152     -0.000002        0.001152
3.000000  -1705.738728  -1705.738726      0.000002       -0.001245
3.033333  -1705.738305  -1705.738305      0.000000       -0.000241
3.066667  -1705.736961  -1705.736963     -0.000002        0.000983
3.100000  -1705.734771  -1705.734770      0.000001       -0.000387


5.2.2015 -- 18:51:52
JOBNAM = ti1.00 -- P07

Using morse function

Chi squared         = 1.38286217253e-11
Reduced Chi squared = 4.60954057509e-12
R squared           = 0.999999356009

morse parameters:

a      =     0.759904
b      =   -15.070597
c      =    74.720674
lambda =     0.764158

Ground state parameters:

V0           =     3.002206 Bohr^3 (unit cell volume)
             =     3.002206 Bohr   (WS-radius)
E0           = -1705.738293 Ry
Bmod         =   115.348605 GPa
Grun. param. =     1.147079
```

(continues on next page)

(continued from previous page)

```
sws            Einp           Eout          Residual       err (% * 10**6)
2.900000  -1705.733279  -1705.733278      0.000001       -0.000462
2.933333  -1705.736071  -1705.736073     -0.000002        0.001462
2.966667  -1705.737719  -1705.737717      0.000002       -0.001374
3.000000  -1705.738290  -1705.738290     -0.000000        0.000238
3.033333  -1705.737873  -1705.737873      0.000000       -0.000175
3.066667  -1705.736536  -1705.736537     -0.000001        0.000590
3.100000  -1705.734353  -1705.734353      0.000000       -0.000280


5.2.2015 -- 18:51:52
JOBNAM = ti1.00 -- P07

Using morse function

Chi squared         = 2.17769833254e-11
Reduced Chi squared = 7.25899444181e-12
R squared           = 0.99999898819

morse parameters:

a      =     0.831171
b      =   -14.866360
c      =    66.474685
lambda =     0.729592

Ground state parameters:

V0           =     3.002865 Bohr^3 (unit cell volume)
             =     3.002865 Bohr   (WS-radius)
E0           = -1705.737540 Ry
Bmod         =   114.985899 GPa
Grun. param. =     1.095434

sws            Einp           Eout          Residual       err (% * 10**6)
2.900000  -1705.732490  -1705.732491     -0.000001        0.000576
2.933333  -1705.735292  -1705.735289      0.000003       -0.001807
2.966667  -1705.736942  -1705.736944     -0.000002        0.001443
3.000000  -1705.737535  -1705.737536     -0.000001        0.000646
3.033333  -1705.737140  -1705.737138      0.000002       -0.001159
3.066667  -1705.735820  -1705.735820     -0.000000        0.000187
3.100000  -1705.733649  -1705.733649     -0.000000        0.000114


hcp_lattice_constants_analyze(hcp):
sws0 =      3.002260
c/a0 =      1.610122
B0   =    115.952318
E0   =  -1705.738844
R    =      0.019532
cs   =    498.360422
```

3.4 Ground state volume of fcc CoCrFeMnNi high-entropy alloy using DLM

```
#!/usr/bin/python

# Calculate the equilibrium volume and bulk modulus of
# fcc CoCrFeMnNi high-entropy alloy using DLM.

# !!! NOTE !!!
# This script DOES NOT automatically start running
# the batch scripts. It only generates the input files
# and batch scripts which the user runs by themselves
# !!! NOTE !!!

import pyemto
import os
import numpy as np

# It is recommended to always use absolute paths
folder = os.getcwd()                  # Get current working directory.
latpath = "/wrk/hpleva/structures"    # Folder where the structure output files are.
emtopath = folder+"/cocrfemnni_fcc"   # Folder where the calculations will be␣
↪performed.

cocrfemnni=pyemto.System(folder=emtopath)

sws = np.linspace(2.50,2.70,11) # 11 different volumes from 2.5 Bohr to 2.7 Bohr

# Set KGRN and KFCD values using a for loop.
# Use write_input_file functions to write input files to disk:

for i in range(len(sws)):
    cocrfemnni.bulk(lat='fcc',
                    jobname='cocrfemnni',
                    latpath=latpath,
                    atoms=['Co','Co','Cr','Cr','Fe','Fe','Mn','Mn','Ni','Ni'],
                    splts=[-1.0,1.0,-1.0,1.0,-1.0,1.0,-1.0,1.0,-1.0,1.0],
                    sws=sws[i],
                    amix=0.02,
                    efmix=0.9,
                    expan='M',
                    sofc='Y',
                    afm='M',          # Fixed-spin DLM calculation.
                    iex=7,            # We want to use self-consistent GGA (PBE).
                    nz2=16,
                    tole=1.0E-8,
                    ncpa=10,
                    nky=21,
                    tfermi=5000,
                    dx=0.015,         # Dirac equation parameters
                    dirac_np=1001,    # Dirac equation parameters
                    nes=50,           # Dirac equation parameters
                    dirac_niter=500)  # Dirac equation parameters

    cocrfemnni.emto.kgrn.write_input_file(folder=emtopath)
    cocrfemnni.emto.kfcd.write_input_file(folder=emtopath)
    cocrfemnni.emto.batch.write_input_file(folder=emtopath)
```

The output of this script should look like this:

```
25.3.2015 -- 14:57:04
JOBNAM = quick_fit -- N/A

Using morse function

Chi squared         = 4.62250902489e-09
Reduced Chi squared = 6.60358432126e-10
R squared           = 0.999931991714

morse parameters:

a      =     0.139363
b      =   -66.030179
c      =  7819.057118
lambda =     2.099339

Ground state parameters:

V0           =     2.604323 Bohr^3 (unit cell volume)
             =     2.604323 Bohr   (WS-radius)
E0           = -2558.658938 Ry
Bmod         =   184.105793 GPa
Grun. param. =     2.733677

sws            Einp           Eout        Residual       err (% * 10**6)
2.500000  -2558.650560  -2558.650581     -0.000021        0.008282
2.520000  -2558.653736  -2558.653710      0.000026       -0.010160
2.540000  -2558.656041  -2558.656024      0.000017       -0.006580
2.560000  -2558.657623  -2558.657612      0.000011       -0.004134
2.580000  -2558.658526  -2558.658555     -0.000029        0.011502
2.600000  -2558.658899  -2558.658926     -0.000027        0.010718
2.620000  -2558.658785  -2558.658792     -0.000007        0.002689
2.640000  -2558.658229  -2558.658212      0.000017       -0.006616
2.660000  -2558.657266  -2558.657242      0.000024       -0.009530
2.680000  -2558.655941  -2558.655930      0.000011       -0.004314
2.700000  -2558.654301  -2558.654322     -0.000021        0.008142
```

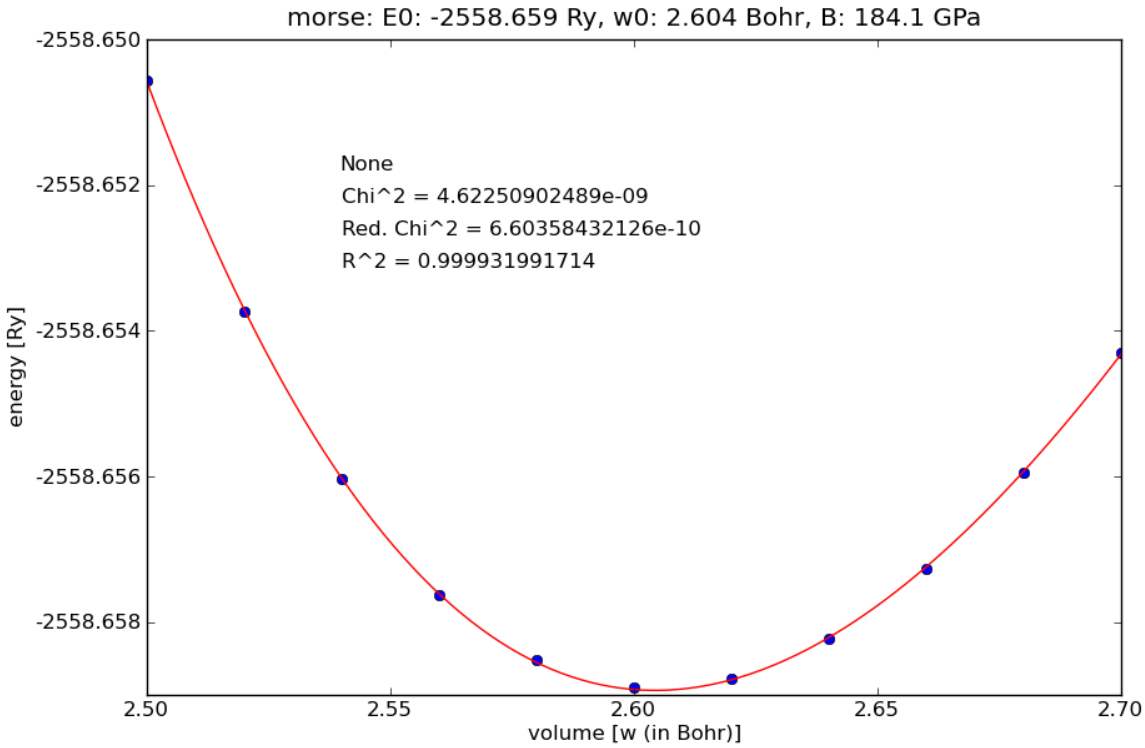

Fig. 1: WS-radius vs. energy curve of fcc CoCrFeMnNi.

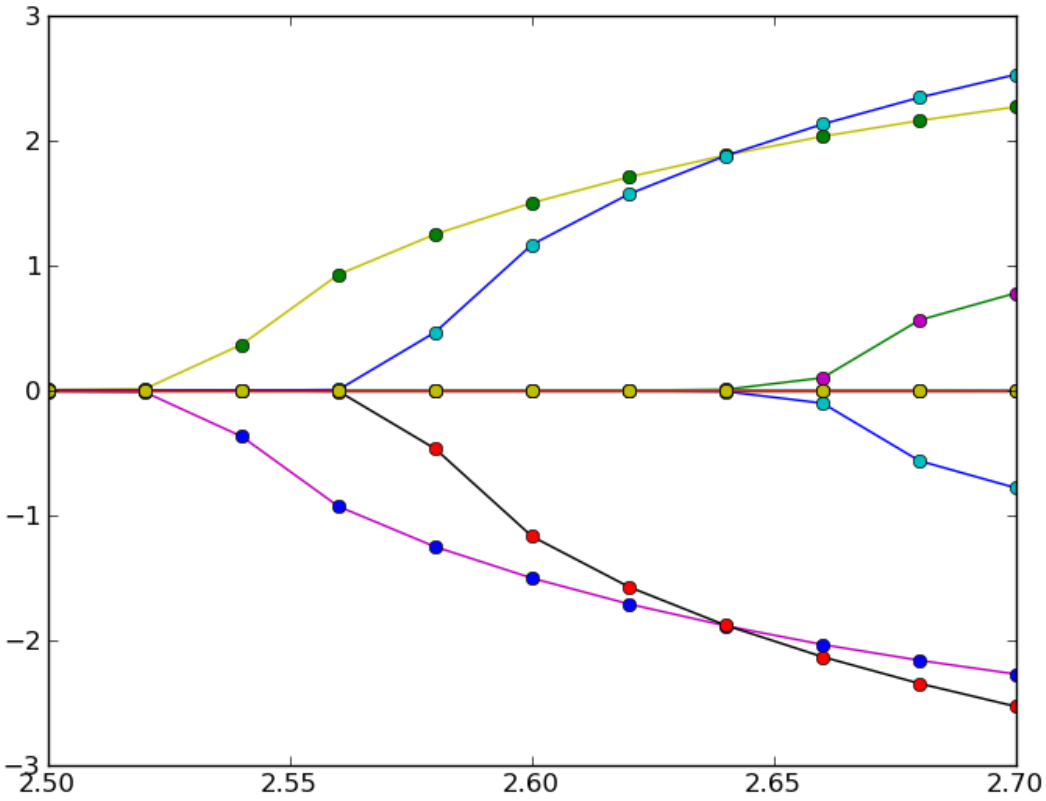

Fig. 2: WS-radius vs. magnetic moments of fcc CoCrFeMnNi.

CHAPTER 4

pyEMTO Code Reference

Descriptions of all modules, classes and functions can be found on these pages.

4.1 Main class

Below is a description of the code found in the main part of the package, system.py.

Created on Wed Dec 3 14:25:06 2014

@author: Matti Ropo @author: Henrik Levämäki

class system.**System**(*folder=None*, *EMTOdir=None*, *xc=None*)

Bases: object

The main class which provides the basis for the pyEMTO scripts.

Somewhere in the beginning of a pyEMTO script a new instance of the system class should be created. All subsequent communication with the newly created system should be through the class methods, which are described below.

Parameters

- **folder** (*str*) – Main folder where the input and output files will be stored. Use of absolute paths is recommended (Default value = current working directory)
- **EMTOdir** (*str*) – Path to the folder of the EMTO installation. This entry can and should be modified by the user inside the System.__init__ function (Default value = /home/user/EMTO5.8)
- **xc** (*str*) – Choice for the xc-functional can be set here. (Default value = PBE)

Returns None

Return type None

bulk(*jobname=None*, *lat=None*, *atoms=None*, *concs=None*, *splts=None*, *sws=None*, *latname=None*, *latpath=None*, *emtopath=None*, *ibz=None*, *bmod=None*, *xc=None*, *ca=None*, ***kwargs*)

Initializes the basic parameters for bulk systems.

Basic information concerning the system, such as the types of atoms and the crystal structure should be given to this function and it should be called right after the class instance has been created.

Parameters

- **jobname** – Name of the system (Default value = None)
- **lat** – The type of lattice structure (Default value = None)
- **atoms** – List of atoms in the system (Default value = None)
- **concs** – List of concentrations of the elements in the 'atoms' list. This information is only used in CPA calculations (Default value = None)
- **splts** – List of initial magnetic moments of the elements in the 'atoms' list (Default value = None)
- **sws** (*float*) – The Wigner-Seitz radius of the system (Default value = None)
- **latname** – The 'jobname' of the BMDL, KSTR and SHAPE output files. These structure output files have to be located in the 'latpath' directory and they have to be named jobname.extention (Default value = None)
- **latpath** – The absolute path to the folder where the 'bmdl', 'kstr' and 'shape' folders are located, which in turn contain the output files of the structure calculation (Default value = None)
- **emtopath** – The absolute path to the folder where the EMTO installation is located (Default value = None)
- **ibz** – The code number indicating the Bravais lattice that the crystal structure of the system has. For a list of possible values, please consult the EMTO manual (Default value = None)
- **bmod** – The bulk modulus can be inputed here and if it is given, it will be used by the elastic modulus routines (Default value = None)
- **xc** – The choice of the xc-functional. If None, PBE will be used as default (Default value = None)
- **ca** – The c/a ratio of hcp structures can be inputed here and if it is given, it will be used by the elastic modulus routines (Default value = None)
- ****kwargs** – Arbitrary other KGRN and KFCD input parameters can be given here as keyword arguments. They will be passed down to the self.emto.set_values() function

Returns None

Return type None

bulk_new(*jobname=None*, *lat=None*, *atoms=None*, *concs=None*, *splts=None*, *sws=None*, *latname=None*, *latpath=None*, *emtopath=None*, *ibz=None*, *bmod=None*, *xc=None*, *ca=None*, *iqs=None*, *its=None*, *itas=None*, ***kwargs*)

Initializes the basic parameters for bulk systems.

!!!A NEW VERSION OF THE OLD "bulk" ROUTINE!!!

Basic information concerning the system, such as the types of atoms and the crystal structure should be given to this function and it should be called right after the class instance has been created.

Parameters

- **jobname** – Name of the system (Default value = None)
- **lat** – The type of lattice structure (Default value = None)

- **atoms** – List of atoms in the system (Default value = None)
- **concs** – List of concentrations of the elements in the 'atoms' list. This information is only used in CPA calculations (Default value = None)
- **splts** – List of initial magnetic moments of the elements in the 'atoms' list (Default value = None)
- **sws** (*float*) – The Wigner-Seitz radius of the system (Default value = None)
- **latname** – The 'jobname' of the BMDL, KSTR and SHAPE output files. These structure output files have to be located in the 'latpath' directory and they have to be named jobname.extention (Default value = None)
- **latpath** – The absolute path to the folder where the 'bmdl', 'kstr' and 'shape' folders are located, which in turn contain the output files of the structure calculation (Default value = None)
- **emtopath** – The absolute path to the folder where the EMTO installation is located (Default value = None)
- **ibz** – The code number indicating the Bravais lattice that the crystal structure of the system has. For a list of possible values, please consult the EMTO manual (Default value = None)
- **bmod** – The bulk modulus can be inputed here and if it is given, it will be used by the elastic modulus routines (Default value = None)
- **xc** – The choice of the xc-functional. If None, PBE will be used as default (Default value = None)
- **ca** – The c/a ratio of hcp structures can be inputed here and if it is given, it will be used by the elastic modulus routines (Default value = None)
- ****kwargs** – Arbitrary other KGRN and KFCD input parameters can be given here as keyword arguments. They will be passed down to the self.emto.set_values() function

Returns None

Return type None

check_conv(*jobname*, *folder='./'*)

Checks the convergence of given KGRN and KFCD calculations by reading their output files.

Parameters

- **jobname** (*str*) – Name of the output files
- **folder** (*str*) – Name of the folder where the output files are located (Default value = "./")

Returns Convergence of KGRN (True/False), convergence of KFCD (True/False)

Return type boolean, boolean

check_str(*jobname*, *folder='./'*)

Checks if a KSTR calculation file exists and has converged.

Only KSTR file is checked because BMDL and SHAPE are fast to rerun in any case.

Parameters

- **jobname** (*str*) – Filename of the structure output file
- **folder** (*str*) – Folder where the output file is located (Default value = "./")

Returns True if KSTR has finished and False if not

Return type boolean

create_jobname(*jobname=None*)
Creates jobnames based on system information.

Creates a jobname based on the optional input prefix *jobname*. The format of the full jobname in this case will be: jobname_3.000000, where 3.000000 is the sws.

If *jobname* prefix is not given, full jobname is generated automatically based on system data self.sws, self.atoms and self.concs. The format of the jobname is: au0.50pd0.50_3.000000, where 0.50 are the atomic concentrations and 3.000000 is the sws.

Parameters **jobname** (*str*) – Optional prefix for the full jobname (Default value = None)

Returns Newly created full jobname

Return type str

elastic_constants_analyze(*sws=None*, *bmod=None*, *ca=None*, *R=None*, *cs=None*, *relax=True*, *debug=False*)
Analyzes the output files generated using the elastic_constants_batch_generate function.

The results are printed on screen.

Parameters

- **sws** (*float*) – WS-radius (Default value = None)
- **bmod** (*float*) – Bulk modulus (Default value = None)
- **ca** (*float*) – hpc c/a ratio (Default value = None)
- **R** (*float*) – Dimensionless quantity of hcp systems (Default value = None)
- **cs** (*float*) – Second order c/a-derivative of the energy (Default value = None)

Returns None

Return type None

elastic_constants_batch_calculate(*sws=None*, *bmod=None*, *ca=None*, *R=None*, *cs=None*)
Calculates the elastic constants of a system using the parallelism of the batch system.

This is a combination of the batch_generate and batch_analyze functions.

Parameters

- **sws** (*float*) – WS-radius (Default value = None)
- **bmod** (*float*) – Bulk modulus (Default value = None)
- **ca** (*float*) – hpc c/a ratio (Default value = None)
- **R** (*float*) – The dimensionless quantity of hcp systems (Default value = None)
- **cs** (*float*) – Second order c/a-derivative of the energy (Default value = None)

Returns None

Return type None

elastic_constants_batch_generate(*sws=None*, *ca=None*, *relax=True*)
Generates all the necessary input files based on the class data.

Parameters

- **sws** (*float*) – WS-radius (Default value = None)
- **ca** (*float*) – hpc c/a ratio (Default value = None)

Returns List of jobnames

Return type list(str)

elastic_constants_serial_calculate(*sws=None*, *bmod=None*, *ca=None*, *R=None*, *cs=None*)

Calculates elastic constants on one CPU without using the batch system.

At the end the results are printed on screen.

Parameters

- **sws** (*float*) – WS-radius (Default value = None)
- **bmod** (*float*) – Bulk modulus (Default value = None)
- **ca** (*float*) – hpc c/a ratio (Default value = None)

Returns None

Return type None

find_lc(*delta=0.01*, *prn=True*, *xc='PBE'*)

Computes initial estimates for the ground state quantities for cubic systems.

Parameters

- **delta** (*float*) – Step size for the volume vs. energy array (Default value = 0.01)
- **prn** (*boolean*) – True if results should be printed, False if not (Default value = True)
- **xc** (*str*) – Choice of the xc-functional (Default value = 'PBE')

Returns WS-radius, bulk modulus, c/a and energy

Return type float, float, float, float

find_lc_hcp(*delta=0.01*, *prn=True*, *xc='PBE'*)

Computes initial estimates for the ground state quantities for hcp systems.

Parameters

- **delta** (*float*) – Step size for the volume vs. energy array (Default value = 0.01)
- **prn** (*boolean*) – True if results should be printed, False if not (Default value = True)
- **xc** (*str*) – Choice of the xc-functional (Default value = 'PBE')

Returns WS-radius, bulk modulus, c/a and energy

Return type float, float, float, float

get_energy(*jobname=None*, *func='PBE'*, *folder=None*)

Extracts total energy from the KFCD output file.

Different total energies given by different xc-functionals can be selected using the *func* input parameter. Default value is 'PBE'.

Parameters

- **jobname** (*str*) – Name of the KFCD output file
- **func** (*str*) – Name of the xc-functional (Default value = "PBE")
- **folder** (*str*) – Name of the folder where the output file is located (Default value = "./")

Returns Total energy if it is found, otherwise return None

Return type float or None

get_fdos (*jobname=None*, *folder=None*)
Extract density of state (DOS) at fermi level from KGRN output

Parameters

- **jobname** (*str*) – Name of the KGRN output file
- **folder** (*str*) – Name of the folder where the output file is located (Default value = "./")

Returns DOS at fermi level

Return type float

get_jobs_status (*jobids=None*, *toplevel=True*)
Returns status of the jobs indicated (jobsdict or list of job ids) or all jobs if no jobids supplied. Set toplevel=False for job step data.

Parameters

- **jobids** (*dict or list*) – List of job IDs (Default value = None)
- **toplevel** (*boolean*) – (Default value = True)

Returns Job statuses

Return type list(str)

get_moments (*jobname=None*, *func='PBE'*, *folder=None*)
Extracts magnetic moments from the KFCD output file.

Parameters

- **jobname** (*str*) – Name of the KFCD output file
- **func** (*str*) – Name of the xc-functional (Default value = "PBE")
- **folder** (*str*) – Name of the folder where the output file is located (Default value = "./")

Returns Total energy if it is found, otherwise return None

Return type float or None

get_status_counts (*jobids=None*)
Returns the counts of all jobs by status category.

Parameters **jobids** – (Default value = None)

Returns

Return type

lattice_constants_analyze (*sws=None*, *ca=None*, *prn=True*, *debug=False*, *method='morse'*, *return_error=False*)
Analyzes the output files generated using the lattice_constants_batch_generate function.

The results are printed on screen.

Parameters

- **sws** (*list(float)*) – List of WS-radii (Default value = None)
- **ca** (*list(float)*) – List hpc c/a ratios (Default value = None)
- **prn** (*boolean*) – True if results should be printed on screen, False if not (Default value = True)

Returns Equilibrium WS-radius, c/a (only hcp), bulk modulus, energy, R (only hcp) and cs (only hcp)

Return type float, float (only hcp), float, float, float (only hcp), float (only hcp)

lattice_constants_batch_calculate(*sws=None*, *ca=None*)

Calculates the ground state WS-radius using the parallelism of the batch system by submitting one job for each entry in the *sws* list.

This is a combination of the batch_generate and batch_analyze functions. At the end results are printed on screen.

Parameters

- **sws** (*list(float)*) – List of WS-radii (Default value = None)
- **ca** (*float*) – hpc c/a ratio (Default value = None)

Returns WS-radius, c/a (only hcp), bulk modulus, energy, R (only hcp), cs (only hcp)

Return type float, float (only hcp), float, float, float (only hcp), float (only hcp)

lattice_constants_batch_generate(*sws=None*, *ca=None*, *auto_ca=False*)

Generates input files and writes them to disk.

Based on the input *sws* and *ca* lists jobnames are created and then corresponding input files are generated and written on disk. List of jobnames are returned.

Parameters

- **sws** (*list(float)*) – List of WS-radii (Default value = None)
- **ca** (*list(float)*) – List of hcp c/a ratios (Default value = None)

Returns List of jobnames

Return type list(str)

lattice_constants_serial_calculate(*sws=None*, *stype='simple'*, *rerun=False*, *skip=False*, *delta=0.01*, *refine=True*)

Calculates the equilibrium lattice constants on one CPU **WITHOUT** the batch system.

Also the eq. bulk modulus, c/a ratio and energy are returned.

Parameters

- **sws** (*float*) – Initial guess for the eq. WS-radius (Default value = None)
- **stype** (*str*) – Type of the energy minimisation algorithm (Default value = "simple")
- **rerun** (*boolean*) – True if (Default value = False)
- **skip** (*boolean*) – True if (Default value = False)
- **delta** (*float*) – Step size (Default value = 0.01)
- **refine** (*boolean*) – True if an optimized WS-radius vs. energy curve should be computed, False if not (Default value = True)

Returns WS-radius, c/a, bulk modulus, energy

Return type float, float, float, float

predict_next_sws(*swses*, *en*, *maxdelta=0.05*)

Predict next WS-radius based on a simple gradient descent algorithm.

Parameters

- **swses** (*list(float)*) – List of current WS-radii

- **en** (*list(float)*) – List of current energies
- **maxdelta** (*float*) – Maximum step size (Default value = 0.05)

Returns Next WS-radii and True if energy minimum has been found, False if not yet found

Return type float, boolean

print_sws_ens(*string*, *swses*, *energies*)
Prints the WS-radii and calculated energies of cubic systems

Parameters

- **string** (*str*) – Header for the printout
- **swses** (*list(float)*) – List of WS-radii
- **energies** (*list(float)*) – List of energies

Returns None

Return type None

print_sws_ens_hcp(*string*, *swses*, *energies*, *cas*)
Prints the WS-radii, ca/a ratios and calculated energies of hcp systems

Parameters

- **string** (*str*) – Header for the printout
- **swses** (*list(float)*) – List of WS-radii
- **energies** (*list(float)*) – List of energies
- **cas** (*list(float)*) – List of c/a ratios

Returns None

Return type None

refine_lc(*sws*, *delta=0.01*, *prn=True*, *xc='PBE'*)
Calculates a more accurate equilibrium volume for cubic systems.

Parameters

- **sws** (*float*) – WS-radius
- **delta** (*float*) – Step size for the volume vs. energy array (Default value = 0.01)
- **prn** (*boolean*) – True if results should be printed, False if not (Default value = True)
- **xc** (*str*) – Choice of the xc-functional (Default value = 'PBE')

Returns Ws-radius, bulk modulus, c/a and energy

Return type float, float, float, float

refine_lc_hcp(*sws*, *ca0*, *delta=0.01*, *prn=True*, *xc='PBE'*)
Calculates a more accurate equilibrium volume for hcp systems.

Parameters

- **sws** (*float*) – WS-radius
- **ca0** (*float*) – Previously computed eq. c/a ratio
- **delta** (*float*) – Step size for the volume vs. energy array (Default value = 0.01)
- **prn** (*boolean*) – True if results should be printed, False if not (Default value = True)

- **xc** (*str*) – Choice of the xc-functional (Default value = 'PBE')

Returns Ws-radius, bulk modulus, c/a and energy

Return type float, float, float, float

runemto(*jobname*, *folder='./'*, *EMTODIR=None*, *onlyKFCD=False*)
Run KGRN and KFCD **WITHOUT** using the batch system and check convergence.

Parameters

- **jobname** (*str*) – Name of the input files
- **folder** (*str*) – Name of the folder where the input files are located (Default value = "./")
- **EMTODIR** (*str*) – Path to the EMTO installation folder (Default value = None)
- **onlyKFCD** (*boolean*) – True if only KFCD needs to calculated, False if also KGRN (Default value = False)

Returns True if the calculations converged, False if not

Return type boolean

runlattice(*jobname=None*, *folder='./'*, *EMTODIR=None*)
Run BMDL, KSTR and SHAPE calculation **WITHOUT** using the batch system.

Parameters

- **jobname** (*str*) – Name of the input files (Default value = None)
- **folder** (*str*) – Name of the folder where the input files are located (Default value = "./")
- **EMTODIR** (*str*) – Path to the EMTO installation folder (Default value = None)

Returns None

Return type None

submit_jobs(*jobnames*, *folder=None*)
Takes a list of jobnames and submits the corresponding batch scripts to the batch system.

Parameters

- **jobnames** (*list (float)*) – List of jobnames
- **folder** (*str*) – Folder where the batch scripts are located (Default value = None)

Returns List of job ID numbers

Return type list(str)

wait_for_jobs(*jobsdict*, *restart_partition='general'*, *sleeptime=60*, *restart_z=None*, *restart_stragglers_after=0.75*, *kill_if_all_ssusp=False*)
Loops checking status until no jobs are waiting or running / all are finished.

wait/run states:

Key	Meaning	Description
CF	CONFIGURING	Job has been allocated resources, but are waiting for them to become ready for use (e.g. booting).
CG	COMPLETING	Job is in the process of completing. Some processes on some nodes may still be active.
PD	PENDING	Job is awaiting resource allocation.
R	RUNNING	Job currently has an allocation.
RS	RESIZING	Job is about to change size.
S	SUSPENDED	Job has an allocation, but execution has been suspended.

done states:

Key	Meaning	Description
CA	CANCELLED	Job was explicitly cancelled by the user or system administrator. The job may or may not have been initiated.
CD	COMPLETED	Job has terminated all processes on all nodes.
F	FAILED	Job terminated with non-zero exit code or other failure condition.
NF	NODE_FAIL	Job terminated due to failure of one or more allocated nodes.
PR	PREEMPTED	Job terminated due to preemption.
TO	TIMEOUT	Job terminated upon reaching its time limit.

Parameters

- **jobsdict** –
- **restart_partition** – (Default value = 'general')
- **sleeptime** – (Default value = 60)
- **restart_z** – (Default value = None)
- **restart_stragglers_after** – (Default value = 0.75)
- **kill_if_all_ssusp** – (Default value = False)

Returns None

Return type None

which_error(*jobname*, *folder='./'*)

Tries to determine the reason why a given KGRN calculation did not converge.

The reason for the crash will be printed on screen.

Parameters

- **jobname** (*str*) – Name of the KGRN output file
- **folder** (*str*) – Name of the folder where the output file is located (Default value = "./")

Returns None

Return type None

write_inputs(*folder=None*, *batch=True*)

Write kgrn and kfcd inputs files and possible batch file

4.2 latticeinputs package

4.2.1 latticeinputs.batch module

Created on Wed Dec 3 15:09:24 2014

@author: Matti Ropo @author: Henrik Levämäki

class latticeinputs.batch.**Batch**(*jobname_lat=None*, *lat=None*, *runtime=None*, *latpath=None*, *EMTOdir=None*, *runBMDL=None*, *runKSTR=None*, *runKSTR2=None*, *runSHAPE=None*, *kappaw=None*, *kappalen=None*, *slurm_options=None*, *account=None*)

Bases: object

Creates a batch script for running BMDL, KSTR and SHAPE calculations

This class is used to to create batch scripts for a supercomputer environment (EMTO 5.8). !!! Currently only SLURM is supported. !!!

Parameters

- **jobname_lat** – (Default value = None)
- **lat** – (Default value = None)
- **runtime** – (Default value = None)
- **latpath** – (Default value = None)
- **EMTOdir** – (Default value = None)
- **runBMDL** – (Default value = None)
- **runKSTR** – (Default value = None)
- **runKSTR2** – (Default value = None)
- **runSHAPE** – (Default value = None)
- **kappaw** – (Default value = None)
- **kappalen** – (Default value = None)

Returns None

Return type None

check_input_file()

Perform various checks on the class data to make sure that all necessary data exists before we attempt to write the input file to disk

Returns

Return type

output()

(self) -> (str)

Output first part of the kgrn input file in formated string

Returns

Return type

set_values (*key*, *value*)

Parameters

- **key** –
- **value** –

Returns

Return type

write_input_file (*folder=None*)
(self,str) ->(None)

Save batch input data to file named filename

Parameters **folder** – (Default value = None)

Returns

Return type

4.2.2 latticeinputs.bmdl module

Created on Wed Dec 3 15:09:24 2014

@author: Matti Ropo @author: Henrik Levämäki

class latticeinputs.bmdl.**Bmdl** (*jobname_lat=None*, *lat=None*, *latparams=None*, *latvectors=None*, *basis=None*, *msgl=None*, *nprn=None*, *bmdl_nl=None*, *lamda=None*, *amax=None*, *bmax=None*, *nqr2=None*, *ca=None*)

Bases: object

Contains information about BMDL input files for EMTO 5.8 program.

Parameters

- **jobname_lat** – (Default value = None)
- **lat** – (Default value = None)
- **latparams** – (Default value = None)
- **latvectors** – (Default value = None)
- **basis** – (Default value = None)
- **msgl** – (Default value = None)
- **nprn** – (Default value = None)
- **bmdl_nl** – (Default value = None)
- **lamda** – (Default value = None)
- **amax** – (Default value = None)
- **bmax** – (Default value = None)
- **nqr2** – (Default value = None)
- **ca** (*float*) – hcp c/a ratio (Default value = None)

Returns None

Return type None

check_input_file()
Perform various checks on the class data.

Makes sure that all necessary data exists before we attempt to write the input file to disk.

Returns None

Return type None

output()
Outputs BMDL input file as a formatted string

Creates a long string containing the batch job script which will later be written on disk.

Returns batch script input file

Return type str

set_values(*key*, *value*)

Parameters

- **key** –
- **value** –

Returns

Return type

write_input_file(*folder=None*)
Save BMDL input data to file named by self.jobname_lat

Parameters **folder** – (Default value = None)

Returns

Return type

4.2.3 latticeinputs.kstr module

Created on Wed Dec 3 15:00:14 2014

@author: Matti Ropo @author: Henrik Levämäki

class latticeinputs.kstr.**Kstr**(*jobname_lat=None*, *lat=None*, *latparams=None*, *ca=None*, *latvectors=None*, *basis=None*, *kappaw=None*, *dmax=None*, *msgl=None*, *nprn=None*, *lamda=None*, *amax=None*, *bmax=None*, *nqr2=None*, *mode=None*, *store=None*, *high=None*, *kstr_nl=None*, *nlh=None*, *nlw=None*, *nder=None*, *itrans=None*, *rwats=None*, *nghbp=None*, *awIQ=None*, *numvec_target=None*)
Bases: object

Contains information about KSTR input files for EMTO 5.8 program

Parameters

- **jobname_lat** – (Default value = None)
- **lat** – (Default value = None)
- **latparams** – (Default value = None)
- **ca** – (Default value = None)

- **latvectors** – (Default value = None)
- **basis** – (Default value = None)
- **kappaw** – (Default value = None)
- **dmax** – (Default value = None)
- **msgl** – (Default value = None)
- **nprn** – (Default value = None)
- **lamda** – (Default value = None)
- **amax** – (Default value = None)
- **bmax** – (Default value = None)
- **nqr2** – (Default value = None)
- **mode** – (Default value = None)
- **store** – (Default value = None)
- **high** – (Default value = None)
- **kstr_nl** – (Default value = None)
- **nlh** – (Default value = None)
- **nlw** – (Default value = None)
- **nder** – (Default value = None)
- **itrans** – (Default value = None)
- **rwats** – (Default value = None)
- **nghbp** – (Default value = None)
- **awIQ** – (Default value = None)

Returns None

Return type None

check_input_file()
Perform various checks on the class data.

Checking function to make sure that all necessary data exists before we attempt to write the input file to disk.

Returns None

Return type None

compute_num_of_vecs(*prims*, *basis*, *nghbp*, *dmax*)
Computes the number of vectors for a given dmax value.

finalize()
Re-initializes input parameters.

generate_prims(*angles*)
Generates the primitive lattice vectors from the angle and Bravais-lattice type information.

optimize_dmax(*prims*, *basis*)
Calculates the best possible dmax value, which gives the closest number of vectors given some target value (which is typically 80-90).

output(*index*)
Outputs KSTR input file as a formatted string.

Parameters **index** –

Returns KSTR input file string

Return type str

set_values(*key*, *value*)
Set input parameter values.

Parameters

- **key** –
- **value** –

Returns None

Return type None

write_input_file(*folder=None*)
Save KSTR input data to file named filename.

Parameters **folder** – (Default value = None)

Returns None

Return type None

4.2.4 latticeinputs.latticeinputs module

Created on Wed Dec 3 14:25:06 2014

@author: Matti Ropo @author: Henrik Levämäki

class latticeinputs.latticeinputs.**Latticeinputs**
Bases: object

Class which is used to communicate with the Bmdl, Kstr and Shape classes.

Returns None

Return type None

basis_transform(*basis*, *matrix*)
Calculates a basis vector transform given by the transformation matrix.

Parameters

- **basis** (*np.array*) – Basis vectors
- **matrix** (*np.array*) – Transformation matrix

distortion(*lat=None*, *dist=None*, *ca=None*, *index=None*, *deltas=None*, *dmaxs=None*, *relax=True*, *relax_index=None*, *basis=None*)
A function which sets various class data to create distorted lattice structures.

Distorted lattices are used to calculate elastic constants. An integer *index* is used to specify for which delta and dmax value we want to generate the distortion input file data.

Default naming convention for the structure files:

bcc bco	=	bcco
bcc fco	=	bccm
fcc bco	=	fccm
fcc fco	=	fcco

lat	Original undistorted lattice
dist	Which distortion we want to calculate. Possible values: 'ortho' or 'mono' for lat='bcc', 'fcc' or 'hcp'.
ca	c over a for hcp structures.
index	Index specifying an element in the delta array. Possible values: 0,1,2,3,4 or 5. 0 = No distortion.
delta	Array of displacement values. Optional, default value is good enough almost always (if not always).
dmax	Array of 'dmax' values for KSTR. They have to be chosen in such fashion so as to keep the number of lattice vectors constant (or as close to a constant as possible) for all of the distorted lattices. Optional, only needed when a custom delta array is used.

Parameters

- **lat** (*str*) – The original, undistorted lattice (Default value = None)
- **dist** (*str*) – The type of distortion (Default value = None)
- **ca** (*float*) – hcp c/a ratio (Default value = None)
- **index** (*int*) – Index for selecting a delta and dmax from the arrays (Default value = None)
- **deltas** (*np.array(float)*) – List of delta values (Default value = None)
- **dmaxs** (*np.array(float)*) – List of dmax values (Default value = None)

Returns None

Return type None

set_values(***kwargs*)

Passes various input parameters down to the Kgrn and Kfcd classes.

Parameters ****kwargs** –

Returns

Return type

write_structure_input_files(*jobname_lat=None*, *lat=None*, *folder=None*, ***kwargs*)

For a given lattice type, this function writes the corresponding structure input files into a given folder

Parameters

- **jobname_lat** (*str*) – Name of the job
- **lat** (*str*) – Name of the lattice
- **folder** (*str*) – Name of the folder

4.2.5 latticeinputs.shape module

Created on Wed Dec 3 15:10:00 2014

@author: Matti Ropo @author: Henrik Levämäki

class latticeinputs.shape.**Shape**(*jobname_lat=None*, *lat=None*, *lmax=None*, *nsr=None*, *nfi=None*, *ivef=None*, *msgl=None*, *nprn=None*)

Bases: object

Contains information about SHAPE input files for EMTO 5.8 program.

Parameters

- **jobname_lat** – (Default value = None)
- **lat** – (Default value = None)
- **lmax** – (Default value = None)
- **nsr** – (Default value = None)
- **nfi** – (Default value = None)
- **ivef** – (Default value = None)
- **msgl** – (Default value = None)
- **nprn** – (Default value = None)

Returns

Return type

check_input_file()

Perform various checks on the class data to make sure that all necessary data exists before we attempt to write the input file to disk

Returns

Return type

output()

Output SHAPE input file in formatted string.

Returns SHAPE input file as a string.

Return type str

set_values(*key*, *value*)

Parameters

- **key** –
- **value** –

Returns

Return type

write_input_file(*folder=None*)

Save SHAPE input data to file named filename

Parameters **folder** – directory to write (Default value = None)

Returns

Return type

4.3 emtoinputs package

4.3.1 emtoinputs.batch module

Created on Wed Dec 3 15:09:24 2014

@author: Matti Ropo @author: Henrik Levämäki

class emtoinputs.batch.**Batch**(*jobname=None*, *runtime=None*, *EMTOdir=None*, *emtopath=None*, *runKGRN=None*, *runKFCD=None*, *account=None*, *KGRN_file_type=None*, *KFCD_file_type=None*, *slurm_options=None*, *parallel=None*)

Bases: object

Creates a batch script for running KGRN and KFCD calculations on a supercomputer environment (EMTO 5.8).

!!! Currently only SLURM is supported. !!!

Parameters

- **jobname** (*str*) – Name for the KGRN and KFCD jobs. This will become the first part of the input and output file names.
- **runtime** (*str*) – Maximum running time for the individual batch jobs. The format of this entry should be 'xx:yy:zz', where xx is hours, yy is minutes and zz is seconds.
- **EMTOdir** (*str*) – Path to the EMTO installation (Default value = '$HOME/EMTO5.8')
- **emtopath** (*str*) – Path to the folder where the KGRN and KFCD input files are located
- **runKGRN** (*boolean*) – True if KGRN should be run, False if KGRN should not be run
- **runKFCD** (*boolean*) – True if KFCD should be run, False if KFCD should not be run

Returns None

Return type None

check_input_file()

Perform various checks on the class data to make sure that all necessary data exists before we attempt to write the input file to disk

output()

Output first part of the kgrn input file in formated string

Returns Batch job script file in the form of a long string

Return type str

set_values(*key*, *value*)

Parameters

- **key** –
- **value** –

Returns

Return type

write_input_file(*folder=None*)
(self,str) ->(None)

Save BMDL input data to file named filename

Parameters **folder** – (Default value = None)

Returns

Return type

4.3.2 emtoinputs.emtoinputs module

Created on Wed Dec 3 14:25:06 2014

@author: Matti Ropo @author: Henrik Levämäki

class emtoinputs.emtoinputs.**Emtoinputs**
Bases: object

Class which is used to communicate with the Kgrn and Kfcd classes.

Returns None

Return type None

set_values(***kwargs*)
Passes various input parameters down to the Kgrn and Kfcd classes

Parameters ****kwargs** – Keyword arguments

Returns None

Return type None

4.3.3 emtoinputs.kfcd module

Created on Wed Dec 3 15:05:43 2014

@author: Matti Ropo @author: Henrik Levämäki

class emtoinputs.kfcd.**Kfcd**(*jobname=None*, *latname=None*, *latpath=None*, *msgl=None*, *nprn=None*, *lmaxs=None*, *nth=None*, *kfcd_nfi=None*, *fpot=None*, *ovcor=None*, *ubg=None*, *DIR001=None*, *DIR002=None*, *DIR003=None*, *DIR004=None*, *DIR006=None*, *sws=None*, *CQNA=None*, *KFCD_file_type=None*)
Bases: object

Handles the information and writing of kfcd file.

Parameters

- **jobname** – (Default value = None)
- **latname** – (Default value = None)
- **latpath** – (Default value = None)
- **msgl** – (Default value = None)
- **nprn** – (Default value = None)
- **lmaxs** – (Default value = None)

- **nth** – (Default value = None)
- **kfcd_nfi** – (Default value = None)
- **fpot** – (Default value = None)
- **ovcor** – (Default value = None)
- **ubg** – (Default value = None)
- **DIR001** – (Default value = None)
- **DIR002** – (Default value = None)
- **DIR003** – (Default value = None)
- **DIR004** – (Default value = None)
- **DIR006** – (Default value = None)
- **sws** – (Default value = None)

Returns None

Return type None

check_input_file()
Perform various checks on the class data.

Makes sure that all necessary data exists before we attempt to write the input file to disk

Returns None

Return type None

output()
Outputs KFCD input file as a formatted string.

Outputs EMTO5.8 KFCD input file

Returns Formatted string

Return type str

set_values(*key*, *value*)
Changes values of the class variables.

Parameters

- **key** (*str*) – name of the variable
- **value** (*str, int or float*) – value of the variable

Returns None

Return type None

write_input_file(*folder=None*)
Writes input file to disk.

Save KFCD input data to file named filename.

Parameters **folder** (*str*) – Folder where the data will be written (Default value = None)

Returns None

Return type None

4.3.4 emtoinputs.kgrn module

Created on Wed Dec 3 14:48:25 2014

@author: Matti Ropo @author: Henrik Levämäki

class emtoinputs.kgrn.**Kgrn**(*jobname=None*, *latname=None*, *latpath=None*, *ibz=None*, *atoms=None*, *concs=None*, *iqs=None*, *its=None*, *itas=None*, *qtrs=None*, *splts=None*, *fixs=None*, *sm_ss=None*, *s_wss=None*, *ws_wsts=None*, *atconf=None*, *sws=None*, *strt=None*, *msgl=None*, *expan=None*, *fcd=None*, *func=None*, *niter=None*, *nlin=None*, *nprn=None*, *ncpa=None*, *mode=None*, *frc=None*, *dos=None*, *ops=None*, *afm=None*, *crt=None*, *lmaxh=None*, *lmaxt=None*, *kgrn_nfi=None*, *fixg=None*, *shf=None*, *sofc=None*, *kmsh=None*, *nkx=None*, *nky=None*, *nkz=None*, *fbz=None*, *kmsh2=None*, *ibz2=None*, *nkx2=None*, *nky2=None*, *nkz2=None*, *zmsh=None*, *nz1=None*, *nz2=None*, *nz3=None*, *nres=None*, *nzd=None*, *depth=None*, *imagz=None*, *eps=None*, *elim=None*, *amix=None*, *efmix=None*, *vmtz=None*, *mmom=None*, *tole=None*, *tolef=None*, *tolcpa=None*, *tfermi=None*, *nsws=None*, *dsws=None*, *alpcpa=None*, *efgs=None*, *hx=None*, *nx=None*, *nz0=None*, *stmp=None*, *iex=None*, *dirac_np=None*, *nes=None*, *dirac_niter=None*, *iwat=None*, *nprna=None*, *vmix=None*, *rwat=None*, *rmax=None*, *dx=None*, *dr1=None*, *test=None*, *teste=None*, *testy=None*, *testv=None*, *FOR001=None*, *DIR002=None*, *DIR003=None*, *FOR004=None*, *DIR006=None*, *DIR009=None*, *DIR010=None*, *DIR011=None*, *ncpu=None*, *CQNA=None*, *KGRN_file_type=None*, *setups=None*)

Bases: object

A class which contains all the KGRN input file related information.

Parameters

- **jobname** – (Default value = None)
- **latname** – (Default value = None)
- **latpath** – (Default value = None)
- **ibz** – (Default value = None)
- **atoms** – (Default value = None)
- **concs** – (Default value = None)
- **iqs** – (Default value = None)
- **its** – (Default value = None)
- **itas** – (Default value = None)
- **qtrs** – (Default value = None)
- **splts** – (Default value = None)
- **fixs** – (Default value = None)
- **sm_ss** – (Default value = None)
- **s_wss** – (Default value = None)
- **ws_wsts** – (Default value = None)
- **atconf** – (Default value = None)

- **sws** – (Default value = None)
- **strt** – (Default value = None)
- **msgl** – (Default value = None)
- **expan** – (Default value = None)
- **fcd** – (Default value = None)
- **func** – (Default value = None)
- **niter** – (Default value = None)
- **nlin** – (Default value = None)
- **nprn** – (Default value = None)
- **ncpa** – (Default value = None)
- **mode** – (Default value = None)
- **frc** – (Default value = None)
- **dos** – (Default value = None)
- **ops** – (Default value = None)
- **afm** – (Default value = None)
- **crt** – (Default value = None)
- **lmaxh** – (Default value = None)
- **lmaxt** – (Default value = None)
- **kgrn_nfi** – (Default value = None)
- **fixg** – (Default value = None)
- **shf** – (Default value = None)
- **sofc** – (Default value = None)
- **kmsh** – (Default value = None)
- **nkx** – (Default value = None)
- **nky** – (Default value = None)
- **nkz** – (Default value = None)
- **fbz** – (Default value = None)
- **kmsh2** – (Default value = None)
- **ibz2** – (Default value = None)
- **nkx2** – (Default value = None)
- **nky2** – (Default value = None)
- **nkz2** – (Default value = None)
- **zmsh** – (Default value = None)
- **nz1** – (Default value = None)
- **nz2** – (Default value = None)
- **nz3** – (Default value = None)

- **nres** – (Default value = None)
- **nzd** – (Default value = None)
- **depth** – (Default value = None)
- **imagz** – (Default value = None)
- **eps** – (Default value = None)
- **elim** – (Default value = None)
- **amix** – (Default value = None)
- **efmix** – (Default value = None)
- **vmtz** – (Default value = None)
- **mmom** – (Default value = None)
- **tole** – (Default value = None)
- **tolef** – (Default value = None)
- **tolcpa** – (Default value = None)
- **tfermi** – (Default value = None)
- **nsws** – (Default value = None)
- **dsws** – (Default value = None)
- **alpcpa** – (Default value = None)
- **efgs** – (Default value = None)
- **hx** – (Default value = None)
- **nx** – (Default value = None)
- **nz0** – (Default value = None)
- **stmp** – (Default value = None)
- **iex** – (Default value = None)
- **dirac_np** – (Default value = None)
- **nes** – (Default value = None)
- **dirac_niter** – (Default value = None)
- **iwat** – (Default value = None)
- **nprna** – (Default value = None)
- **vmix** – (Default value = None)
- **rwat** – (Default value = None)
- **rmax** – (Default value = None)
- **dx** – (Default value = None)
- **dr1** – (Default value = None)
- **test** – (Default value = None)
- **teste** – (Default value = None)
- **testy** – (Default value = None)

- **testv** – (Default value = None)
- **FOR001** – (Default value = None)
- **DIR002** – (Default value = None)
- **DIR003** – (Default value = None)
- **FOR004** – (Default value = None)
- **DIR006** – (Default value = None)
- **DIR009** – (Default value = None)
- **DIR010** – (Default value = None)
- **DIR011** – (Default value = None)

Returns None

Return type None

AtomOutput()
(self) -> str

Output of atomic lines in kgrn format

Returns A atomic lines for kgrn input

Return type str

Atomline(*atom*, *iq*, *it*, *ita*, *conc*, *Sm_s*, *S_ws*, *WS_wst*, *qtr*, *splt*, *fix*)
Prints atomic line in kgrn format

Parameters

- **atom** –
- **iq** –
- **it** –
- **ita** –
- **conc** –
- **Sm_s** –
- **S_ws** –
- **WS_wst** –
- **qtr** –
- **splt** –
- **fix** –

Returns

Return type

aconflines(*atype*)
Returns string containing electronic configuration lines

Contains the database of atomic orbital configuration information which is present in atomic block of the KGRN input file.

Parameters **atype** (*str*) – Name of the element in the periodic table

Returns A formatted string corresponding to the parameter atype

Return type str

check_input_file()
Perform various checks on the class data

Makes sure that all necessary data exists before we attempt to write the input file to disk.

Returns None

Return type None

create_atconf()
Constructs the self.atconf list out of the atomic information.

Returns None

Return type None

create_atomblock()
Constructs the KGRN input file atomblock if all the necessary parameters are present.

Returns None

Return type None

output()
(self) -> (str)

Output first part of the kgrn input file in formated string

Returns A first part of the kgrn input file

Return type str

set_values(*key*, *value*)

:param value:, so we have to use .any() :type value: :returns: :rtype:

write_input_file(*folder=None*)
(self,str) ->(None)

Save KGRN input data to file named filename

Parameters **folder** – (Default value = None)

Returns None

Return type

4.4 EOS package

4.4.1 EOS.EOS module

Created on Wed Dec 9 10:51:00 2014

@author: Matti Ropo @author: Henrik Levämäki

class EOS.EOS.**EOS**(*name*, *xc='PBE'*, *method='morse'*, *units='bohr'*, *warnings=True*)

Bases: object

Fit equation of state for bulk systems.

The following equations are available:

morse	PRB 37, 790 (1988)
sjeos	PRB 63, 224115 (2001)
taylor	A third order Taylor series expansion about the minimum volume
murnaghan	PRB 28, 5480 (1983)
birch	Intermetallic compounds: Principles and Practice, Vol I: Principles. pages 195-210
birchmurnaghan	PRB 70, 224107
pouriertarantola	PRB 70, 224107
vinet	PRB 70, 224107
antonschmidt	Intermetallics 11, 23-32 (2003)
oldpoly	A third order polynomial fit (alternative implementation)

Usage:

```
eos = EquationOfState(volumes, energies, eos='murnaghan')
v0, e0, B = eos.fit()
eos.plot()
```

Parameters

- **name** –
- **xc** – (Default value = 'PBE')
- **method** – (Default value = 'morse')
- **units** – (Default value = 'bohr')

Returns

Return type

angstrom2bohr(*V*)

Parameters **V** –

Returns

Return type

antonschmidt(*V*, *Einf*, *B*, *n*, *V0*)

From Intermetallics 11, 23-32 (2003)

Einf should be E_infinity, i.e. infinite separation, but according to the paper it does not provide a good estimate of the cohesive energy. They derive this equation from an empirical formula for the volume dependence of pressure,

E(vol) = E_inf + int(P dV) from V=vol to V=infinity

but the equation breaks down at large volumes, so E_inf is not that meaningful

n should be about -2 according to the paper.

I find this equation does not fit volumetric data as well as the other eqution do.

Parameters

- **V** –
- **Einf** –
- **B** –
- **n** –
- **V0** –

Returns Energy

Return type float

ascii_plot(*x*, *y*, *z*, *title=''*)

birch(*V*, *E0*, *B0*, *BP*, *V0*)

From Intermetallic compounds: Principles and Practice, Vol. I: Principles Chapter 9 pages 195-210 by M. Mehl. B. Klein, D. Papaconstantopoulos paper downloaded from Web

case where n=0

Parameters

- **V** –
- **E0** –
- **B0** –
- **BP** –
- **V0** –

Returns

Return type

birchmurnaghan(*V*, *E0*, *B0*, *BP*, *V0*)

BirchMurnaghan equation from PRB 70, 224107

Parameters

- **V** –
- **E0** –
- **B0** –
- **BP** –
- **V0** –

Returns

Return type

bohr2angstrom(*WSrad*)

Parameters **WSrad** –

Returns

Return type

ca_fit(*x*, *y*, *n*, *debug=False*, *title=''*, *find_best_fit=False*)

Fits a polynomial to x vs. y data and calculates xmin and ymin from the curve.

Parameters

- **x** –
- **y** –
- **n** –

Returns

Return type

compute_eos_fit()
Performs the least-squares curve fitting for the chosen EOS function. Ground state quantities are returned.

compute_eos_quality()
Compute different estimates for the quality of the fit.

compute_initial_guess()
Calculates initial guessess for the fitting parameters.

distortion_fit(*x*, *y*, *num=2*, *title=''*, *ascii_art=False*)
Fits the distortion_poly function to the distortion data.

num [number of variables in the fitting function] 1 : E=a2*x**2 2 : E=a2*x**2 + a0 3 : E=a2*x**2 + a1*x + a0

The fit coefficient(s) and r-squared describing the accuracy of the fit are returned.

Parameters

- **x** –
- **y** –
- **num** – (Default value = 2)

Returns

Return type

distortion_poly1(*x*, *a2*)

One variable form of the 2nd order polynomial which is used to fitting distortion vs. energy data in order to find elastic constants.

Parameters

- **x** –
- **a2** –
- **a1** –

Returns

Return type

distortion_poly2(*x*, *a2*, *a0*)

Two variable form of the 2nd order polynomial which is used to fitting distortion vs. energy data in order to find elastic constants.

Parameters

- **x** –
- **a2** –

- **a0** –

Returns

Return type

distortion_poly3(*x*, *a2*, *a1*, *a0*)

Three variable form of the 2nd order polynomial which is used to fitting distortion vs. energy data in order to find elastic constants.

Parameters

- **x** –
- **a2** –
- **a1** –
- **a0** –

Returns

Return type

eos_output()

Construct a string for the EOS output message.

fit(*swses*, *energies*, *shift=False*, *show_plot=False*, *show_output=True*, *pressure=False*)

Calculate volume, energy, and bulk modulus.

About the units:

Input "swses" should be the WS-radii in bohr. Input "energies" should be the energies in Ry.

Volumes are always in Angstrom**3 Bulk moduli are always in GPa

Parameters

- **swses** (*list(float)*) – List of WS-radii
- **energies** (*list(float)*) – List of energies
- **shift** (*True or False*) – Shift the energies so that the lowest energy is zero. Improves fit quality. Sometimes instable.

Returns Eq. WS-rad, energy, bulk modulus, Gruneisen parameter

Return type float, float, float, float

fit_eval(*sws*)

Evaluate the fitting function at given points

Parameters **sws** –

Returns

Return type

morse(*sws*, *a0*, *b0*, *c0*, *l0*)

Parameters

- **w** –
- **a0** –
- **b0** –

- **c0** –
- **l0** –

murnaghan(*V*, *E0*, *B0*, *BP*, *V0*)
From PRB 28,5480 (1983)

Parameters

- **V** –
- **E0** –
- **B0** –
- **BP** –
- **V0** –

Returns

Return type

oldpoly(*V*, *c0*, *c1*, *c2*, *c3*)
polynomial fit, 3rd order

Parameters

- **V** –
- **c0** –
- **c1** –
- **c2** –
- **c3** –

Returns Energy

Return type float

parabola(*x*, *a*, *b*, *c*)
Parabola polynomial function

This function is used to fit the data to get good guesses for the equation of state fits.

A 4th order polynomial fit to get good guesses for was not a good idea because for noisy data the fit is too wiggly 2nd order seems to be sufficient, and guarentees a single minimum.

Parameters

- **x** –
- **a** –
- **b** –
- **c** –

Returns Energy

Return type float

plot(*filename=None*, *show=True*)
Plot fitted energy curve.

Uses Matplotlib to plot the energy curve. Use *show=True* to show the figure and *filename='abc.png'* or *filename='abc.eps'* to save the figure to a file.

Parameters

- **filename** – (Default value = None)
- **show** – (Default value = None)

Returns

Return type

static polyfit(*x*, *y*, *n*)

pouriertarantola(*V*, *E0*, *B0*, *BP*, *V0*)
Pourier-Tarantola equation from PRB 70, 224107

Parameters

- **V** –
- **E0** –
- **B0** –
- **BP** –
- **V0** –

Returns

Return type

predicted()
Evaluates the EOS function using the calc. EOS params.

Returns EOS function

Return type func

pressure(*sws*)
Pressure

print_eos_output()

relax_fit(*x*, *y*, *n*, *debug=False*, *title=''*)
Fits a polynomial to x vs. y data and calculates xmin and ymin from the curve.

Parameters

- **x** –
- **y** –
- **n** –

Returns

Return type

sjeos(*V*, *a*, *b*, *c*, *d*)
Stabilized jellium (SJEOS) fitting function from PRB 63, 224115 (2001).

Note: EOS parameters here differ from the PRB definitions by having V0 being included in them. a(PRB) = a/V0 b(PRB) = b/V0**(2/3) c(PRB) = c/V0**(1/3)

taylor(*V*, *E0*, *beta*, *alpha*, *V0*)
Taylor Expansion up to 3rd order about V0

Parameters

- **V** –
- **E0** –
- **beta** –
- **alpha** –
- **V0** –

Returns

Return type

vinet(*V*, *E0*, *B0*, *BP*, *V0*)

Vinet equation from PRB 70, 224107

Parameters

- **V** –
- **E0** –
- **B0** –
- **BP** –
- **V0** –

Returns

Return type

vol2wsrad(*V*)

Parameters **V** –

Returns

Return type

wsrad2vol(*WSrad*)

Parameters **WSrad** –

Returns

Return type

EOS.EOS.**curve_fit**(*f*, *x*, *y*, *p0*)

Fits an arbitraty function to data (x,y).

This part comes from: http://projects.scipy.org/scipy/browser/trunk/scipy/optimize/minpack.py

Parameters

- **f** –
- **x** –
- **y** –
- **p0** –

Returns

Return type

4.5 utilities package

4.5.1 utilities.utils module

Created on Wed Dec 3 14:25:06 2014

@author: Matti Ropo @author: Henrik Levämäki

utilities.utils.**distort**(*dist_mat*, *mat*)

Apply distortion matrix to lattice vectors or sites.

Coordinates are assumed to be Cartesian.

utilities.utils.**extrapolate_0k**(*a=None*, *Ta=None*, *bmod=None*, *TB=None*, *lat=None*, *TD=None*, *alpha=None*, *Talpha=None*, *B1=None*, *T0=None*, *ca=None*, *ZP=True*)

Extrapolates experimental data to zero Kelvin.

A function to extrapolate experimental room temperature lattice constants and bulk moduli to zero Kelvin. Zero point effects are also taken into account.

Input parameter	Description
a	Experimental lattice constant measured at temperature Ta (in Angstroms)
Ta	The temperature at which a was measured (in Kelvin)
bmod	Experimental bulk modulus measured at temperature TB (in GPa)
TB	The temperature at which bmod was measured (in Kelvin)
lat	Lattice structure
TD	Experimental Debye temperature of the substance
alpha	VOLUMETRIC!!! thermal expansion coefficient at temperature T (in 106 1/K)
Talpha	The temperature at which alpha was measured (in Kelvin)
B1	Pressure derivative of the bulk modulus
T0	The temperature to which we want to extrapolate (Optional argument, if not given assume 0K)
ca	c/a lattice parameter for hcp structures
ZP	Whether or not zero-point (ZP) corrections are performed. If one only wants to calculate the fractional volume change due to dropping temperature, ZP=False.

Parameters

- **a** – (Default value = None)
- **Ta** – (Default value = None)
- **bmod** – (Default value = None)
- **TB** – (Default value = None)
- **lat** – (Default value = None)
- **TD** – (Default value = None)
- **alpha** – (Default value = None)
- **Talpha** – (Default value = None)
- **B1** – (Default value = None)
- **T0** – (Default value = None)
- **ca** – (Default value = None)

- **ZP** – (Default value = True)

Returns ZPAE corrected 0K lattice constant (in Angstrom), ZPPE corrected 0K bulk modulus (in GPa)

Return type float,float

utilities.utils.**latparam_to_wsrad**(*a=None*, *lat=None*, *ca=None*, *c=None*)

Parameters

- **a** – (Default value = None)
- **lat** – (Default value = None)
- **ca** – (Default value = None)
- **c** – (Default value = None)

Returns

Return type

utilities.utils.**rotation_matrix**(*axis*, *theta*)

Return the rotation matrix associated with counterclockwise rotation about the given axis by theta radians.

Parameters

- **axis** –
- **theta** –

utilities.utils.**run_bash**(*cmd*)

Parameters **cmd** (*str*) – Command to run

Returns

Return type

utilities.utils.**run_emto**(*name*, *folder='./'*)

Submits a batch script to the queue.

Finds all the files in a folder that start with the name

Parameters

- **name** –
- **folder** – (Default value = "./")

Returns

Return type

utilities.utils.**submit_to_batch**(*folder*, *jobname*, *system='slurm'*)

Parameters

- **folder** –
- **jobname** –
- **system** – (Default value = 'slurm')

Returns

Return type

utilities.utils.**write_batch**(*folder*, *jobname*)

Parameters

- **folder** (*str*) – Folder to write batch file
- **jobname** (*str*) – Name of the job

Returns

Return type

utilities.utils.**write_job**(*folder*, *jobname*, *atom*, *conc*, *sws*)

Parameters

- **folder** –
- **jobname** –
- **atom** –
- **conc** –
- **sws** –

Returns

Return type

utilities.utils.**wsrad_to_latparam**(*sws*, *lat*, *ca=None*, *c=None*)

Parameters

- **sws** –
- **lat** –
- **ca** – (Default value = None)
- **c** – (Default value = None)

Returns

Return type

CHAPTER 5

pyEMTO presentation

Download pyEMTO presentation

CHAPTER 6

Indices and tables

- genindex
- modindex
- search

Python Module Index

Index

V

W