Masci-tools Documentation

Release 0.2

Jens Broeder

Jun 19, 2019

Contents

This package was developed in the process of developing the AiiDA-FLEUR and AiiDA_KKR plugins to AiiDA. It contains helper functions that can help with common pre and proprocessing steps of the FLEUR and KKR codes developed at the Forschungszentrum Jülich (see also the juDFT website for more information).

If you use this package please cite: ...

CHAPTER 1

Requirements to use this code:

- lxml
- ase
- masci-tools

CHAPTER 2

Installation Instructions:

Install from pypi the latest release:

```
$ pip install masci-tools
```

or from the aiida-fleur source folder any branch:

```
$ pip install .
# or which is very useful to keep track of the changes (developers)
$ pip install -e .
```

CHAPTER 3

Acknowledgments:

We acknowledge partial support from the EU Centre of Excellence "MaX – Materials Design at the Exascale" (http://www.max-centre.eu). (Horizon 2020 EINFRA-5, Grant No. 676598) We thank the AiiDA team for their help and work. Also the vial exchange with developers of AiiDA packages for other codes was inspireing.

CHAPTER 4

User's Guide

CHAPTER 5

Developer's Guide

CHAPTER 6

Module reference (API)

6.1 Source code Documentation (API reference)

6.1.1 Visualisation and Plotting

masci_tools.vis.kkr_plot_shapefun.**change_zoom**(*ax, zoom_range, center=[0, 0, 0]*)

Change the zoom of a 3d plot

Author Philipp Ruessmann

Parameters

- **ax** – axis which is zoomed
- **zoom_range** – range to which the image is zoomed, total range from center-zoom_range to center+zoom_range
- **center** – center of the zoomed region (optional, defaults to origin)

masci_tools.vis.kkr_plot_shapefun.**plot_shapefun**(*pos*, *out*, *mode*)

Creates a simple matplotlib image to show the shapefunctions given it's positions in the unit cell, the atoms's vertices in *ut* and the plotting mode

Author Philipp Ruessmann

Parameters

- **pos** – positions of the centers of the cells
- **verts** – array of vertices of the shapefunction (outlines of shapes)
- **mode** – 'all' or 'single' determines whether or not all shapes are combined in a single figure or plotted as individual figures

Returns ax return the axis in which the plot was done (useful to pass to 'change_zoom' and 'zoom_in' functions of this module

masci_tools.vis.kkr_plot_shapefun.**zoom_in**(*ax*, *atm*, *pos*, *zoom_range=10*)
Zoom into shapefun of a single atom

Author Philipp Ruessmann

Parameters

- **ax** – axis in which shapefun plot is found
- **atm** – atom index whose shapefunction is zoomed
- **pos** – array of positions of centers of the shapes (needed to shift center of zommed region to correct atom
- **zoom_range** – range of the zoomed region (optional, defaults to 10)

6.1.2 I/O helper and output file parsers

Here commonly used functions that do not need aiida-stuff (i.e. can be tested without a database) are collected.

masci_tools.io.common_functions.**angles_to_vec**(*magnitude*, *theta*, *phi*)
convert (magnitude, theta, phi) to (x,y,z)

theta/phi need to be in radians!

Input can be single number, list of numpy.ndarray data Returns x,y,z vector

masci_tools.io.common_functions.**convert_to_pystd**(*value*)
Recursively convert numpy datatypes to standard python, needed by aiida-core. Usage:

```
converted = convert_to_pystd(to_convert)
```

where *to_convert* can be a dict, array, list, or single valued variable

masci_tools.io.common_functions.**get_corestates_from_potential**(*potfile='potential'*)
Read core states from potential file

masci_tools.io.common_functions.**get_ef_from_potfile**(*potfile*)
extract fermi energy from potfile

masci_tools.io.common_functions.**get_highest_core_state**(*nstates*, *energies*, *lmoments*)
Find highest lying core state from list of core states, needed to find and check energy contour

masci_tools.io.common_functions.**interpolate_dos**(*dosfile*, *return_original=False*)
interpolation function copied from complexdos3 fortran code

Principle of DOS here: Two-point contour integration for DOS in the middle of the two points. The input DOS and energy must be complex. Parameter deltae should be of the order of magnitude of eim:

```
      <-2*deltae->   _
           /\        |     DOS=(n(1)+n(2))/2 + (n(1)-n(2))*eim/deltae
          /  \       |
        (1)  (2)   2*i*eim=2*i*pi*Kb*Tk
        /      \     |
       /        \    |
------------------------ (Real E axis)
```

Parameters **input** – either absolute path of 'complex.dos' file or file handle to it

Returns E_Fermi, numpy array of interpolated dos

Note output units are in Ry!

masci_tools.io.common_functions.**open_general**(*filename_or_handle*, *iomode='r'*)
Open a file directly from a path or use a file handle if that is given. Also take care of closed files by reopenning them. This is intended to be used like this:

```
f = open_general(outfile)
with f: # make sure the file is properly closed
    txt = f.readlines()
```

masci_tools.io.common_functions.**vec_to_angles**(*vec*)
converts vector (x,y,z) to (magnitude, theta, phi)

Io routines for band structure files

masci_tools.io.io_fleur_bands.**read_fleur_banddos_hdf**(*filepath*)
Reads in the banddos.hdf file from the FLEUR code

returns a dictionary containing all datasets with multidim numpy arrays and also containing the attributes of the groups

Parameters **filepath** – path to the banddos.hdf file

Returns xcoord, bands, xlabels, band_character, band_char_label,

kpoints, weights, rep_cell, cell, positions, atomicnumbers, special_point_pos

Expected file content: datasets [u'bravaisMatrix',

u'numFoundEigenvals', u'specialPointIndices', u'lLikeCharge', u'positions', u'atomicNumbers', u'coordinates', u'weights', u'reciprocalCell', u'eigenvalues', u'specialPointLabels', u'equivAtomsGroup']

attributes: {u'lastFermiEnergy': array([0.20852455]),

u'maxL': array([3], dtype=int32), u'nAtoms': array([2], dtype=int32), u'nSpecialPoints': array([7], dtype=int32), u'nTypes': array([1], dtype=int32), u'neigd': array([19], dtype=int32), u'nkpt': array([100], dtype=int32), u'spins': array([1], dtype=int32), u'version': array([1], dtype=int32)}

IO routines for hdf

masci_tools.io.io_hdf5.**read_hdf**(*filepath*)
Reads in an hdf file and returns its context in a nested dictionary

!Only works for files with unique group and dataset names

class masci_tools.io.kkr_params.**kkrparams**(***kwargs*)
Class for creating and handling the parameter input for a KKR calculation Optional keyword arguments are passed to init and stored in values dictionary.

Example usage: params = kkrparams(LMAX=3, BRAVAIS=array([[1,0,0], [0,1,0], [0,0,1]]))

Alternatively values can be set afterwards either individually with params.set_value('LMAX', 3)

or multiple keys at once with params.set_multiple_values(EMIN=-0.5, EMAX=1)

Other useful functions: - print the description of a keyword: params.get_description([key]) where [key] is a string for a keyword in params.values - print a list of mandatory keywords: params.get_all_mandatory() - print a list of keywords that are set including their value: params.get_set_values()

Note: KKR-units (e.g. atomic units with energy in Ry, length in a_Bohr) are assumed except for the keys'<RBLEFT>', '<RBRIGHT>', 'ZPERIODL', and 'ZPERIODR' which should be given in Ang. units!

fill_keywords_to_inputfile(*is_voro_calc=False*, *output=u'inputcard'*)
Fill new inputcard with keywords/values automatically check for input consistency if is_voro_calc==True change mandatory list to match voronoi code, default is KKRcode

classmethod get_KKRcalc_parameter_defaults(*silent=False*)
set defaults (defined in header of this file) and returns dict, kkrparams_version

get_all_mandatory()
Return a list of mandatory keys

get_description(*key*)
Returns description of keyword 'key'

get_dict(*group=None*, *subgroup=None*)
Returns values dictionary.

Prints values belonging to a certain group only if the 'group' argument is one of the following: 'lattice', 'chemistry', 'accuracy',

'external fields', 'scf cycle', 'other'

Additionally the subgroups argument allows to print only a subset of all keys in a certain group. The following subgroups are available: in 'lattice' group: '2D mode', 'shape functions' in 'chemistry' group: 'Atom types', 'Exchange-correlation', 'CPA mode',

'2D mode'

in 'accuracy' group: 'Valence energy contour', 'Semicore energy contour', 'CPA mode', 'Screening clusters', 'Radial solver', 'Ewald summation', 'LLoyd'

get_missing_keys(*use_aiida=False*)
Find list of mandatory keys that are not yet set

get_set_values()
Return a list of all keys/values that are set (i.e. not None)

get_type(*key*)
Extract expected type of 'key' from format info

get_value(*key*)
Gets value of keyword 'key'

is_mandatory(*key*)
Returns mandatory flag (True/False) for keyword 'key'

items()
make kkrparams.items() work

read_keywords_from_inputcard(*inputcard=u'inputcard'*)
Read list of keywords from inputcard and extract values to keywords dict

Example usage p = kkrparams(); p.read_keywords_from_inputcard('inputcard')

Note converts '<RBLEFT>', '<RBRIGHT>', 'ZPERIODL', and 'ZPERIODR' automatically to Ang. units!

remove_value(*key*)
Removes value of keyword 'key', i.e. resets to None

set_multiple_values(***kwargs*)
Set multiple values (in example value1 and value2 of keywords 'key1' and 'key2') given as key1=value1, key2=value2

set_value(*key*, *value*, *silent=False*)
Sets value of keyword 'key'

classmethod split_kkr_options(*valtxt*)
Split keywords after fixed length of 8 :param valtxt: list of strings or single string :returns: List of keywords of maximal length 8

update_to_kkrimp()
Update parameter settings to match kkrimp specification. Sets self.__params_type and calls _update_mandatory_kkrimp()

update_to_voronoi()
Update parameter settings to match voronoi specification. Sets self.__params_type and calls _update_mandatory_voronoi()

masci_tools.io.kkr_read_shapefun_info.**read_shapefun**(*path='.'*)
Read vertices of shapefunctions with Zoom into shapefun of a single atom

Author Philipp Ruessmann

Parameters **path** – path where voronoi output is found (optional, defaults to './')

Returns pos positions of the centers of the shapefunctions

Returns out dictionary of the vertices of the shapefunctions

CHAPTER 7

Indices and tables

- genindex
- modindex
- search

Python Module Index

m

Index

S

U

V

Z