AFLOWpi Developers Guide

Release

Andrew Supka

June 04, 2016

Contents:

CHAPTER 1

plot module

plot.**bands**(*calcs, yLim=[-10, 10], DOSPlot=''', LSDA=False, runlocal=False, postfix=''', tight_banding=False*)

Generates electronic band structure plots for the calculations in the dictionary of dictionaries of calculations with the option to have a DOS or PDOS plot to accompany it.

Parameters **calcs** (*dict*) – dictionary of dictionaries representing the set of calculations

Keyword Arguments

- **yLim** (*list*) – List or tuple of two integers for max and min range in horizontal axis of DOS plot
- **LSDA** (*bool*) – To plot DOS as spin polarized or not (calculation must have been done as spin polarized)
- **DOSPlot** (*str*) – DOS or the PDOS plots next to the eletronic band structure plot (assuming you ran either ppDOS or ppPDOS) (default: NONE)
- **postfix** (*str*) – Postfix to the filename of the plot
- **tight_banding** (*bool*) – Whether to treat the input data as from Quantum Espresso or WanT bands.x

Returns None

plot.**distortionEnergy**(*calcs1, xaxis, yaxis, zaxis='Energy', calcs=None, colorbarUnits=None, titleArray=None, plotTitle=None, xAxisStr=None, yAxisStr=None, fileName='distortionEnergy.pdf', percentage=False, runlocal=True*)

plot.**dos**(*calcs, yLim=[-10, 10], LSDA=False, runlocal=False*)

Generates DOS plots for the calculations in the dictionary of dictionaries of calculations

Parameters **calcs** (*dict*) – dictionary of dictionaries representing the set of calculations

Keyword Arguments

- **yLim** (*list*) – List or tuple of two integers for max and min range in horizontal axis of DOS plot
- **LSDA** (*bool*) – To plot DOS as spin polarized or not (calculation must have been done as spin polarized)
- **runlocal** (*bool*) – Do the plotting right now or if False do it when the calculations are running
- **postfix** (*str*) – Postfix to the filename of the plot
- **tight_banding** (*bool*) – Whether to treat the input data as from Quantum Espresso or WanT bands.x

Returns None

plot.**grid_plot**(*calcs*, *xaxis*, *yaxis*, *zaxis='Energy'*, *colorbarUnits=None*, *zaxis_title=None*, *plot_title=None*, *xAxisStr=None*, *yAxisStr=None*, *fileName='grid_plot.pdf'*, *runlocal=True*)

plot.**interpolatePlot**(*calcs*, *variable1*, *variable2*, *zaxis='Energy'*, *xaxisTitle=None*, *yaxisTitle=None*, *zaxisTitle=None*, *title=None*, *fileName='interpolatePlot.pdf'*, *delta=False*, *text_min=False*, *vhline_min=False*, *circle_min=False*, *delta_min=True*, *rel_center=False*, *plot_color='jet'*, *find_min=False*)

Takes a list of calculations and plots the energy of the calculations as a function of two input variables the first value is the baseline for the energy value and the energy plotted is the difference between that energy and the other energies in the grid

Parameters

- **calcs** (*dict*) – dictionary of dictionaries of calculations
- **variable1** (*str*) – a string of the variable in the calculations that you want as your x axis
- **variable2** (*str*) – a string of the variable in the calculations that you want to your y axis

Keyword Arguments

- **title** (*str*) – Title of plot (default: None)
- **zaxis** (*str*) – Choice out of the keywords in each calc to plot in the Z axis (default: Energy)
- **xaxisTitle** (*str*) – title of xaxis (default: same as variable1)
- **yaxisTitle** (*str*) – title of yaxis (default: same as variable2)
- **zaxisTitle** (*str*) – title of zaxis (default: same as zaxis)
- **fileName** (*str*) – Name (and path where default is directory where script is run from) of the output file (default: 'interpolatePlot.pdf')
- **delta** (*bool*) – Z-axis scale relative to its first value
- **delta_min** (*bool*) – Z-axis scale relative to its lowest value
- **text_min** (*bool*) – Display text of minimum value next to the minimum value if find_min=True
- **vhline_min** (*bool*) – Display text of minimum value next to the minimum value if find_min=True
- **circle_min** (*bool*) – Display text of minimum value next to the minimum value if find_min=True
- **delta_min** (*bool*) – Display text of minimum value next to the minimum value if find_min=True
- **rel_center** (*bool*) – Display text of minimum value next to the minimum value if find_min=True
- **plot_color** (*str*) – string of the matplotlib colormap to be used
- **find_min** (*bool*) – Interpolate between points and find value of variable1 and variable2 for the minimum value of Z axis

Returns None

plot.**interpolatePlot1D**(*calcs*, *variable1*, *yaxis='Energy'*, *xaxisTitle=None*, *yaxisTitle=None*, *title=None*, *fileName='interpolatePlot.pdf'*, *delta=False*, *circle_min=False*)

Takes a list of calculations and plots the energy of the calculations as a function of two input variables the first

value is the baseline for the energy value and the energy plotted is the difference between that energy and the other energies in the grid

Parameters

- **calcs** (*dict*) – dictionary of dictionaries of calculations
- **variable1** (*dict*) – a string of the variable in the calculations that you want as your x axis

Keyword Arguments

- **yaxis** (*str*) – Choice out of the keywords in each calc to plot in the Z axis (default: Energy)
- **xaxisTitle** (*str*) – title of xaxis (default: same as variable1)
- **yaxisTitle** (*str*) – title of yaxis (default: same as yaxis)
- **title** (*str*) – Title of plot (default: None)
- **fileName** (*str*) – Name (and path where default is directory where script is run from) of the output file (default: 'interpolatePlot.pdf')
- **delta** (*bool*) – Z-axis scale relative to its first value
- **circle_min** (*bool*) – Display text of minimum value next to the minimum value

plot.**opdos**(*calcs, yLim=[-10, 10], LSDA=False, runlocal=False, postfix='', scale=False*)

Generates electronic band structure plots for the calculations in the dictionary of dictionaries of calculations with the option to have a DOS or PDOS plot to accompany it.

Parameters **calcs** (*dict*) – dictionary of dictionaries representing the set of calculations

Keyword Arguments

- **yLim** (*list*) – List or tuple of two integers for max and min range in horizontal axis of DOS plot
- **LSDA** (*bool*) – To plot DOS as spin polarized or not (calculation must have been done as spin polarized)
- **runlocal** (*bool*) – Do the plotting right now or if False do it when the calculations are running
- **postfix** (*str*) – Postfix to the filename of the plot
- **tight_banding** (*bool*) – Whether to treat the input data as from Quantum Espresso or WanT bands.x

Returns None

plot.**phonon**(*calcs*, *runlocal=False*, *postfix=''*, *THz=True*)

plot.**radialPDF**(*calcs*, *atomNum*, *filterElement=None*, *runlocal=False*, *title=''*, ***kwargs*)

kwargs get passed to pyplot.hist

plot.**read_transport_datafile**(*ep_data_file*, *mult_x=1.0*, *mult_y=1.0*)

Arguments:

Keyword Arguments:

Returns:

plot.**transport_plots**(*calcs*, *runlocal=False*, *postfix=''*)

CHAPTER 2

prep module

prep.**ConfigSectionMap**(*section*, *option*, *configFile=None*)

prep.**__null__**(**args*, ***kwargs*)

prep.**addToBlockWrapper**(*oneCalc*, *ID*, *block*, *addition*)
Wraps AFLOWpi.prep.__addToBlock for use inside _calcs_container methods

Parameters

- **oneCalc** (*dict*) – a dictionary containing properties about the AFLOWpi calculation
- **ID** (*str*) – ID string for the particular calculation and step
- **block** (*str*) – string of the block in the _ID.py that the addition is to be added to for that step of workflow
- **addition** (*str*) – a string containing code to be written to the specific block in the _ID.py for each calculation

Returns None

prep.**askAFLOWpiVars**(*refAFLOWpiVars*)
Cycle on the keys of the refAFLOWpiVars dictionary and ask to define them

Parameters **refAFLOWpiVars** (*dict*) – the variables in the ref files that you need to input to run the calculation

Returns None

prep.**bands**(*calcs*, *dk=None*, *nk=None*)
Wrapper function to write the function AFLOWpi.prep.__oneBands to the _ID.py

Parameters **calcs** (*dict*) – a dictionary of dicionaries representing the set of calculations

Keyword Arguments

- **dk** (*float*) – distance between points for Electronic Band Structure calculation
- **nk** (*int*) – approximate number of k points to be calculated along the path

Returns The identical "calcs" input variable

prep.**bandsAflow**(*dk*, *LAT*)
Query aflow for band structure path and generate the path for band structure calculation

Parameters

- **dk** (*float*) – distance between k points along path in Brillouin Zone
- **LAT** (*int*) – bravais lattice number from Quantum Espresso convention

Keyword Arguments None

Returns **info** – some information nks (str): number of k points in path stringk (str): kpoint path string

Return type str

prep.**build_calcs**(*PARAM_VARS*, *build_type='product'*)

prep.**calcFromFile**(*aflowkeys*, *fileList*, *reffile=None*, *pseudodir=None*, *build=True*, *workdir=None*, *keep_name=False*, *clean_input=True*)

Reads in a string of an QE input file path, a string of an QE input, a file object of a QE input or a list of them and attempts to fill create a calculation from them. If they are missing things such as k_points card, they are automtically generated.

Parameters

- **aflowkeys** (*dict*) – a dictionary generated by AFLOWpi.prep.init
- **fileList** (*list*) – a string of an QE input file path, a string of an QE input, a file object of a QE input or a list of them

Keyword Arguments

- **reffile** (*str*) – a partially filled QE input file used in case the input(s) in fileList are missing. i.e. wfc cutoff. If the names of the Pseudopotential files are not included in the input(s) in fileList, they are chosen depending on the pseudodir chosen and included when the calculation set is formed.
- **workdir** (*str*) – a string of the workdir path that be used to override what is in the config file used when initating the AFLOWpi session
- **pseudodir** (*str*) – a string of the pseudodir path that be used to override what is in the config file used when initating the AFLOWpi session
- **build** (*bool*) – <DEFUNCT OPTION. NEEDS REMOVAL>

class prep.**calcs_container**(*dictionary*)

addToAll(*block=None*, *addition=None*)

bands(*dk=None*, *nk=100*)

Wrapper method to write call AFLOWpi.prep.bands for calculating the Electronic Band Structure.

Parameters **calcs** (*dict*) – a dictionary of dicionaries representing the set of calculations

Keyword Arguments

- **dk** (*float*) – the density in the Brillouin zone of the k point sampling along the entirety of the path between high symmetry points.
- **nk** (*int*) – the approximate number of sampling points in the Brillouin Zone along the entirety of the path between high symmetry points. Points are chosen so that they are equidistant along the entirety of the path. The actual number of points will be slightly different than the inputted value of nk. nk!=None will override any value for dk.

Returns None

change_input(*namelist=None*, *parameter=None*, *value=None*)

change_pseudos(*directory*)

conventional_cell_input()

crawl_min(*mult_jobs=False*, *grid_density=10*, *initial_variance=0.02*, *thresh=0.01*, *constraint=None*, *final_minimization='relax'*)

Wrapper method to call AFLOWpi.pseudo.crawlingMinimization in the high level user interface. Adds a new step to the workflow.

Parameters **self** – the _calcs_container object

Keyword Arguments

- **mult_jobs** (*bool*) – if True split the individual scf jobs into separate cluster jobs if False run them serially
- **grid_density** (*int*) – controls the number of calculations to generate for the minimization num_{calcs}=grid_density^{num_{parameters}-num_{constraints}}
- **initial_variance** (*float*) – amount to vary the values of the parameters from the initial value. i.e. (0.02 = +/-2% variance)
- **thresh** (*float*) – threshold for $DeltaX$ of the lattice parameters between brute force minimization iterations.
- **constraint** (*list*) – a list or tuple containing two entry long list or tuples with the first being the constraint type and the second the free parameter in params that its constraining for example in a orthorhombic cell: constraint=(["volume",'c'],) allows for A and B to move freely but C is such that it keeps the cell volume the same in all calculations generated by the input oneCalc calculation.
- **final_minimization** (*str*) – calculation to be run at the end of the brute force minimization options include "scf", "relax", and "vcrelax"

Returns None

dos(*kpFactor=2*, *project=True*)

Wrapper method to call AFLOWpi.prep.doss in the high level user interface. Adds a new step to the workflow.

Parameters **self** – the _calcs_container object

Keyword Arguments

- **kpFactor** (*float*) – factor to which the k-point grid is made denser in each direction
- **project** (*bool*) – if True: do the projected DOS after completing the DOS

Returns None

evCurve_min(*pThresh=25*, *final_minimization='relax'*)

get_initial_inputs()

increase_step(*func*)

items()

iteritems()

keys()

new_step(*update_positions=True*, *update_structure=True*, *new_job=True*, *ext=''*)

phonon(*nrx1=2*, *nrx2=2*, *nrx3=2*, *innx=2*, *de=0.01*, *mult_jobs=False*, *raman=False*, *LOTO=False*, *disp_sym=True*, *atom_sym=True*, *field_strength=0.01*)

pseudo_test_brute(*ecutwfc*, *dual=[]*, *sampling=[]*, *conv_thresh=0.01*, *constraint=None*, *initial_relax=None*, *min_thresh=0.01*, *initial_variance=0.05*, *grid_density=7*, *mult_jobs=False*, *options=None*)

relax()

resubmit(*reset=True*)

scf()

scfuj(*thresh=0.1*, *nIters=20*, *paodir=None*, *relax='scf'*, *mixing=0.0*)
Wrapper method to call AFLOWpi.scfuj.scfPrep and AFLOWpi.scfuj.run in the high level user interface. Adds a new step to the workflow.

Parameters **self** – the _calcs_container object

Keyword Arguments

- **thresh** (*float*) – threshold for self consistent hubbard U convergence
- **niters** (*int*) – max number of iterations of the acbn0 cycle
- **paodir** (*string*) – the path of the PAO directory. This will override an entry of the paodir in the AFLOWpi config file used for the session
- **mixing** (*float*) – the amount of the previous acbn0 U iteration to mix into the current (only needed when there is U val oscillation)

Returns None

split_chain(*update_positions=True*, *update_structure=True*)

submit()

transport(*temperature=[300], epsilon=True, run_bands=True*)
Wrapper method to call AFLOWpi.scfuj.prep_transport and AFLOWpi.scfuj.run_transport in the high level user interface. Adds a new step to the workflow.

Parameters **self** – the _calcs_container object

Keyword Arguments

- **epsilon** (*bool*) – if True episilon tensor will be computed
- **temperature** (*list*) – list of temperature(s) at which to calculate transport properties

Returns None

values()

vcrelax()

prep.**changeCalcs**(*calcs*, *keyword='calculation'*, *value='scf'*)
A Wrapper function that writes the function AFLOWpi.prep.__changeCalcs to the __ID.py

Parameters **calcs** (*dict*) – a dictionary of dicionaries representing the set of calculations

Keyword Arguments

- **keyword** (*str*) – a string which signifies the type of change that is to be made
- **value** – the value of the choice.

Returns The identical set of calculations as the input to this function

prep.**cleanCalcs**(*calcs*, *runlocal=False*)
Wrapper function for AFLOWpi.prep.__cleanCalcs

Parameters **calcs** (*dict*) – a dictionary of dicionaries representing the set of calculations

Keyword Arguments **runlocal** (*bool*) – a flag to choose whether or not to run the wrapped function now or write it to the _ID.py to run during the workflow

Returns None

prep.**construct_and_run**(*__submitNodeName__*, *oneCalc*, *ID*, *build_command=''*, *subset_tasks=[]*, *fault_tolerant=False*, *mult_jobs=True*, *subset_name='SUBSET'*, *keep_file_names=False*, *clean_input=True*)

this is a check to see if we're restarting when mult_jobs==True

prep.**doss**(*calcs*, *kpFactor=1.5*)

Wrapper function to write the functio n AFLOWpi.prep.__oneDoss to the _ID.py

Parameters **calcs** (*dict*) – a dictionary of dicionaries representing the set of calculations

Keyword Arguments **kpFactor** (*float*) – the factor to which we make each direction in the kpoint grid denser

Returns The identical "calcs" input variable

prep.**extractvars**(*refFile*)

Read refFile and return an empty dictionary with all the keys and None values

Parameters **refFile** (*str*) – filename of the reference input file for the frame

Returns A dictionary containing the keyword extracted from the reference input file as keys with None for values..

prep.**generateAnotherCalc**(*old*, *new*, *calcs*)

Modify the calculation in each subdir and update the master dictionary

Parameters

- **old** (*str*) – string to replace
- **new** (*str*) – replacement string
- **calcs** (*dict*) – dictionary of dictionaries of calculations

Returns A new set of calculations with a new ID of the hash of the new input strings

prep.**getMPGrid**(*primLatVec*, *offset=True*, *string=True*)

class prep.**init**(*PROJECT*, *SET=''*, *AUTHOR=''*, *CORRESPONDING=''*, *SPONSOR=''*, *config=''*, *workdir=None*)

from_file(*fileList*, *reffile=None*, *pseudodir=None*, *build=True*, *workdir=None*)

Reads in a string of an QE input file path, a string of an QE input, a file object of a QE input or a list of them and attempts to fill create a calculation from them. If they are missing things such as k_points card, they are automtically generated.

Parameters

- **aflowkeys** (*dict*) – a dictionary generated by AFLOWpi.prep.init
- **fileList** (*str*) – a string of an QE input file path, a string of an QE input, a file object of a QE input or a list of them

Keyword Arguments

- **reffile** (*str*) – a partially filled QE input file used in case the input(s) in fileList are missing. i.e. wfc cutoff. If the names of the Pseudopotential files are not included in the input(s) in fileList, they are chosen depending on the pseudodir chosen and included when the calculation set is formed.
- **workdir** (*str*) – a string of the workdir path that be used to override what is in the config file used when initating the AFLOWpi session

- **pseudodir** (*str*) – a string of the pseudodir path that be used to override what is in the config file used when initating the AFLOWpi session
- **build** (*bool*) – <DEFUNCT OPTION. NEEDS REMOVAL>

items()

iteritems()

keys()

load(*step=1*)

Loads the calc logs from a given step

Parameters **step** (*int*) – the step of the calculation for whose calclogs are to be loaded

Returns **calcs** – the loaded calc logs

Return type dict

scfs(*allAFLOWpiVars*, *refFile*, *name='first'*, *pseudodir=None*, *build_type='product'*, *build=True*, *run=True*)

A wrapper method to call AFLOWpi.prep.scfs to form the calculation set. This will also create directory within the set directory for every calculation in the set.

Parameters

- **allAFLOWpiVars** (*dict*) – a dictionary whose keys correspond to the keywords in the reference input file and whose values will be used to construct the set of calculations
- **refFile** (*str*) – a filename as a string, a file object, or a string of the file that contains keywords to construct the inputs to the different calculations in the set

Keyword Arguments

- **pseudodir** (*str*) – path of the directory that contains your Pseudopotential files The value in the AFLOWpi config file used will override this.
- **build_type** (*str*) – how to construct the calculation set from allAFLOWpiVars dictionary:

  zip | The first calculation takes the first entry from the list of

  each of the keywords. The second calculation takes the second and so on. The keywords for all lists in allAFLOWpiVars must be the same length for this method.

  product | Calculation set is formed via a "cartesian product" with

  the values the list of each keyword combined. (i.e if allAFLOWpiVars has one keyword with a list of 5 entires and another with 4 and a third with 10, there would be 2000 calculations in the set formed from them via product mode.
- **build** (*bool*) – <DEFUNCT OPTION. NEEDS REMOVAL>
- **run** (*bool*) – <DEFUNCT OPTION. NEEDS REMOVAL>

Returns A dictionary of dictionaries containing the set of calculations.

status(*status={}*, *step=0*, *negate_status=False*)

Loads the calc logs from a given step

Parameters **step** (*int*) – The step of the calculation for whose calclogs are to be loaded. If no step is specified then it will default to load calculations from all steps with the chosen status.

Keyword Arguments

- **status** (*dict*) – key,value pairs for status type and their value to filter on. i.e. status={'Finished':False}
- **negate_status** (*bool*) – filter on the opposite of the status filters

Returns **calcs** – the loaded calcs for one or more steps with the given status

Return type dict

values()

prep.**init__**(*PROJECT*, *SET=''*, *AUTHOR=''*, *CORRESPONDING=''*, *SPONSOR=''*, *config=''*, *workdir=None*)

Initializes the frame

Parameters

- **PROJECT** (*str*) – Name of project
- **SET** (*str*) – Name of set
- **author** (*str*) – Name of author
- **CORRESPONDING** (*str*) – Name of corresponding
- **SPONSOR** (*str*) – Name of sponsor

e.g. initFrame('LNTYPE','', 'MF', 'marco.fornari@cmich.edu','DOD-MURI'). Return the AFLOKEYS dictionary.

prep.**line_prepender**(*filename*, *new_text*)

prepends a file with a new line containing the contents of the string new_text.

Parameters

- **filename** (*str*) – string of the filename that is to be prepended
- **new_text** (*str*) – a string that is one line long to be prepended to the file

Returns None

prep.**loadlogs**(*PROJECT=''*, *SET=''*, *logname=''*, *config=None*)

prep.**lockAtomMovement**(*calcs*)

A Wrapper function that writes the function AFLOWpi.prep.__freezeAtoms to the __ID.py

Parameters **calcs** (*dict*) – a dictionary of dicionaries representing the set of calculations

Returns None

prep.**maketree**(*calcs*, *pseudodir=None*, *build=True*, *workdir=None*)

Make the directoy tree and place in the input file there

Parameters **calcs** (*dict*) –

- Dictionary of dictionaries of calculations

Keyword Arguments

- **pseudodir** (*str*) – path of pseudopotential files directory
- **workdir** (*str*) – a string of the workdir path that be used to override what is in the config file used when initating the AFLOWpi session
- **build** (*bool*) – <DEFUNCT OPTION. NEEDS REMOVAL>

Returns None

prep.**modifyCalcs**(*old*, *new*, *calcs*)
Modify the calculation in each subdir and update the master dictionary

Parameters

- **old** (*str*) – string to replace
- **new** (*str*) – replacement string
- **calcs** (*dict*) – dictionary of dictionaries of calculations

prep.**modifyInputPrefixPW**(*calcs*, *pre*)
A Wrapper function that is used to write a function to the __ID.py

Parameters **calcs** (*dict*) – a dictionary of dicionaries representing the set of calculations

Returns None

prep.**modifyNamelistPW**(*calcs*, *namelist*, *parameter*, *value*, *runlocal=False*)
A Wrapper function that is used to write the function AFLOWpi.prep.__modifyNameListPW to the _ID.py. If the value is intended to be a string in the QE input file, it must be dually quoted i.e. value="'scf"' will become 'scf' in the input file.

Parameters

- **calcs** (*dict*) – a dictionary of dicionaries representing the set of calculations
- **namelist** (*str*) – a string of the fortran namelist that the parameter is in
- **parameter** (*str*) – a string of the parameter name
- **value** – the value of that parameter

Keyword Arguments **runlocal** (*bool*) – a flag to choose whether or not to run the wrapped function now or write it to the _ID.py to run during the workflow.

Returns Either the identical set of calculations if runlocal == False or the set of calculations with the parameter's value changed in their oneCalc['_AFLOWPI_INPUT_'] if runlocal==True

prep.**newstepWrapper**(*pre*)
A function that wraps another function that is to be run before the a certain function runs. Its use must be in the form:

@newstepwrapper(func) def being_wrapped(oneCalc,ID,*args,**kwargs)

where func is the function that is to be run before and being_wrapped is the function being wrapped. oneCalc and ID must be the first two arguments in the function being wrapped. additional arguments and keyword arguments can follow.

Parameters **pre** (*func*) – function object that is to be wrapped before another function runs

Returns the returned values of function being wrapped or the execution of the function is skipped entirely.

class prep.**plotter**(*calcs*)
Class for adding common plotting functions from AFLOWpi.plot module to the high level user interface.

bands(*yLim=[-10, 10], DOSPlot='', LSDA=False, runlocal=False, postfix=''*)
Wrapper method to call AFLOWpi.plot.bands in the high level user interface.

Parameters **self** – the plotter object

Keyword Arguments

- **yLim** (*list*) – a tuple or list of the range of energy around the fermi/Highest occupied level energy that is to be included in the plot.

- **DOSPlot** (*str*) – a string that flags for the option to have either a DOS plot share the Y-axis of the band structure plot.

  Options include: “” | A blank string (default) will cause No Density of

  States plotted alongside the Band Structure

  “APDOS” | Atom Projected Density of States “DOS” | Normal Density of States

- **LSDA** (*bool*) – Plot the up and down of a spin polarized orbital projected DOS calculation.
- **runlocal** (*bool*) – a flag to choose whether or not to run the wrapped function now or write it to the _ID.py to run during the workflow
- **postfix** (*str*) – a string of an optional postfix to the plot filename for every calculation.

Returns None

dos (*yLim=[-10, 10], LSDA=False, runlocal=False, postfix=''*)
Wrapper method to call AFLOWpi.plot.dos in the high level user interface.

Parameters **self** – the plotter object

Keyword Arguments

- **yLim** (*list*) – a tuple or list of the range of energy around the fermi/Highest occupied level energy that is to be included in the plot.
- **LSDA** (*bool*) – Plot the up and down of a spin polarized DOS calculation.
- **runlocal** (*bool*) – a flag to choose whether or not to run the wrapped function now or write it to the _ID.py to run during the workflow
- **postfix** (*str*) – a string of an optional postfix to the plot filename for every calculation.

Returns None

opdos (*yLim=[-10, 10], LSDA=False, runlocal=False, postfix=''*)
Wrapper method to call AFLOWpi.plot.opdos in the high level user interface.

Parameters **self** – the plotter object

Keyword Arguments

- **yLim** (*list*) – a tuple or list of the range of energy around the fermi/Highest occupied level energy that is to be included in the plot.
- **LSDA** (*bool*) – Plot the up and down of a spin polarized orbital projected DOS calculation.
- **runlocal** (*bool*) – a flag to choose whether or not to run the wrapped function now or write it to the _ID.py to run during the workflow
- **postfix** (*string*) – a string of an optional postfix to the plot filename for every calculation.

Returns None

phonon (*runlocal=False*, *postfix=''*, *THz=True*)

transport (*runlocal=False*, *postfix=''*)
Wrapper method to call AFLOWpi.plot.epsilon in the high level user interface.

Parameters **self** – the plotter object

Keyword Arguments

- **nm** (*bool*) – whether to plot in nanometers for spectrum or eV for energy
- **runlocal** (*bool*) – a flag to choose whether or not to run the wrapped function now or write it to the _ID.py to run during the workflow

Returns None

prep.**prep_split_step**(*calcs*, *subset_creator*, *subset_tasks=[]*, *mult_jobs=False*, *substep_name='SUBSET'*, *keep_file_names=False*, *clean_input=True*)

prep.**remove_blank_lines**(*inp_str*)
Removes whitespace lines of text

Parameters **inp_str** (*str*) – input string of text

Returns the same string with blank lines removed

prep.**runAfterAllDone**(*calcs*, *command*, *faultTolerant=True*)
Adds a command to the BATCH command block at the end of each calculation's _ID.py for all calculations in the set. Used to execute a command over all calculations in particular step have completed.

Parameters **calcs** (*dict*) – a dictionary of dicionaries representing the set of calculations

Keyword Arguments

- **command** (*str*) – the text to be added to the BATCH block
- **faultTolerant** (*bool*) – a flag to choose if we return True if some of the calculations ran but did not complete successful

Returns None

prep.**scfs**(*aflowkeys*, *allAFLOWpiVars*, *refFile*, *pseudodir=None*, *build_type='product'*, *build=True*)
Read a reference input file, and construct a set of calculations from the allAFLOWpiVars dictionary defining values for the keywords in the reference input file. This will also create directory within the set directory for every calculation in the set.

Parameters

- **allAFLOWpiVars** (*dict*) – a dictionary whose keys correspond to the keywords in the reference input file and whose values will be used to construct the set of calculations
- **refFile** (*str*) – a filename as a string, a file object, or a string of the file that contains keywords to construct the inputs to the different calculations in the set

Keyword Arguments

- **pseudodir** (*str*) – path of the directory that contains your Pseudopotential files The value in the AFLOWpi config file used will override this.
- **build_type** (*str*) – how to construct the calculation set from allAFLOWpiVars dictionary:

  zip | The first calculation takes the first entry from the list of

  each of the keywords. The second calculation takes the second and so on. The keywords for all lists in allAFLOWpiVars must be the same length for this method.

  product | Calculation set is formed via a "cartesian product" with

  the values the list of each keyword combined. (i.e if allAFLOWpiVars has one keyword with a list of 5 entires and another with 4 and a third with 10, there would be 2000 calculations in the set formed from them via product mode.

- **build** (*bool*) – <DEFUNCT OPTION. NEEDS REMOVAL>

Returns A dictionary of dictionaries containing the set of calculations.

prep.**totree**(*tobecopied*, *calcs*, *rename=None*, *symlink=False*)

Populate all the subdirectories for the calculation with the file in input

Parameters

- **tobecopied** (*str*) – filepath to be copied to the AFLOWpi directory tree
- **calcs** (*dict*) – Dictionary of dictionaries of calculations

Keyword Arguments

- **rename** (*bool*) – option to rename the file/directory being moves into the AFLOWpi directory tree
- **symlink** (*bool*) – whether to copy the data to the AFLOWpi directory tree or to use symbolic links

Returns None

prep.**unlockAtomMovement**(*calcs*)

A Wrapper function that writes the function AFLOWpi.prep.__unfreezeAtoms to the __ID.py

Parameters **calcs** (*dict*) – a dictionary of dicionaries representing the set of calculations

Returns None

prep.**updateStructs**(*calcs*, *update_structure=True*, *update_positions=True*)

A Wrapper function that writes the function AFLOWpi.prep.__oneUpdateStructs to the __ID.py

Parameters **calcs** (*dict*) – a dictionary of dicionaries representing the set of calculations

Keyword Arguments

- **update_structure** (*bool*) – if True update the cell parameter if possible from the output of previous calculations in the workflow.
- **update_positions** (*bool*) – if True update the atomic positions if possible from the output of previous calculations in the workflow.

Returns The identical set of calculations as the input to this function

prep.**updatelogs**(*calcs*, *logname*, *runlocal=False*)

prep.**varyCellParams**(*oneCalc*, *ID*, *param=()*, *amount=0.15*, *steps=8*, *constraint=None*)

Forms and returns a set of calcs with varied cell params must be in A,B,C, and, in degrees,alpha,beta,gamma and then returns it.

Parameters

- **oneCalc** (*dict*) – a dictionary containing properties about the AFLOWpi calculation
- **ID** (*str*) – ID string for the particular calculation and step

Keyword Arguments

- **param** (*tuple*) – the params assoc. with the amount and step. i.e. (‘celldm(1)’,’celldm(3))
- **amount** (*float*) – percentage amount to be varied up and down. i.e (0.04,0.02,0.01)
- **steps** (*int*) – how many steps to within each range. i.e (4,5,7)
- **constraint** (*list*) – a list or tuple containing two entry long list or tuples with the first being the constraint type and the second the free parameter in params that its constraining for example in a orthorhombic cell: constraint=([”volume”,’c’],) allows for A and B to move

freely but C is such that it keeps the cell volume the same in all calculations generated by the input oneCalc calculation.

Returns A dictionary of dictionaries representing a new calculation set

prep.**writeToScript**(*executable*, *calcs*, *from_step=0*)

Generates calls on several functions to set up everything that is needed for a new step in the workflow. The mechanics of the _ID.py are written to it here.

Parameters

- **calcs** (*dict*) – dictionary of dictionaries of calculations
- **executable** (*str*) – <DEFUNCT OPTION: HERE FOR LEGACY SUPPORT>
- ***args** – <DEFUNCT OPTION: HERE FOR LEGACY SUPPORT>

Keyword Arguments ****kwargs** – <DEFUNCT OPTION: HERE FOR LEGACY SUPPORT>

Returns A set of calculations for a new step in the workflow

pseudo module

pseudo.**brute_test**(*calcs*, *ecutwfc*, *dual=None*, *sampling=None*, *constraint=None*, *thresh=0.001*, *initial_variance=0.05*, *grid_density=10*, *mult_jobs=False*, *conv_thresh=0.01*, *calc_type='relax'*)

pseudo.**crawlingMinimization**(*calcs*, *options=None*, *faultTolerant=True*, *constraint=None*, *thresh=0.001*, *initial_variance=0.05*, *grid_density=10*, *mult_jobs=False*, *final_minimization='relax'*)

pseudo.**getMinimization**(*origCalcs*, *fitVars=None*, *options=None*, *runlocal=False*, *faultTolerant=True*, *minimize_var='Energy'*)

pseudo.**plot**(*resultList*, *xaxis=''*, *xtitle=None*, *ytitle=None*, *title=None*, *rename=None*, *file_name=''*)

takes in a list of dictionaries of dictionaries of calculations and generates a plot with the x axis being some value in the list 'key' and splits the calculations and plots them with each plot being a unique combination of the items in key that are not 'xaxis'

Parameters

- **resultList** (*list*) – list of dictionaries of dictionaries of calculations
- **xaxis** (*str*) – the keyword in oneCalc that you choose to be the x axis in your plots

Keyword Arguments

- **xtitle** (*str*) – title of the x axis of the plot (default: None)
- **ytitle** (*str*) – title of the y axis of the plot (default: None)
- **plotTitle** (*str*) – title of the plots (default: None)
- **rename** (*dict*) – a mapping of the names of the keywords of whose values used to generate the plot to some other name. ex. {' _AFLOWPI_ECUTW_':'wavefunction cutoff'}
- **file_name** (*str*) – use this instead of "PT_RESULTS.pdf" as filename of plot

Returns None

CHAPTER 4

retr module

retr.**abc2celldm**(*a=None*, *b=None*, *c=None*, *alpha=None*, *beta=None*, *gamma=None*, *ibrav=None*)

Convert a,b,c,alpha,beta,gamma into celldm for QE

Parameters **None** –

Keyword Arguments

- **a** (*float*) – length of a
- **b** (*float*) – length of b
- **c** (*float*) – length of c
- **alpha** (*float*) – Angle between axis b and c
- **beta** (*float*) – Angle between axis a and c
- **gamma** (*float*) – Angle between axis a and b
- **ibrav** (*int*) – ibrav to be used to convert for defaults

Returns **celldm_array** – an array of (ibrav,celldm(1),celldm(2),celldm(3),celldm(4),celldm(5),celldm(6))

Return type array

retr.**abc2free**(*a=None*, *b=None*, *c=None*, *alpha=None*, *beta=None*, *gamma=None*, *ibrav=None*, *returnString=True*)

Converts a,b,c,alpha,beta,gamma into QE convention primitive lattice vectors

Parameters **None** –

Keyword Arguments

- **a** (*float*) – length of a
- **b** (*float*) – length of b
- **c** (*float*) – length of c
- **alpha** (*float*) – Angle between axis b and c
- **beta** (*float*) – Angle between axis a and c
- **gamma** (*float*) – Angle between axis a and b
- **ibrav** (*int*) – bravais lattice type by QE convention
- **returnString** (*bool*) – return vectors as a string or as a numpy matrix

Returns Either a numpy.matrix or a string of the primitive lattice vectors generated from the celldm input

retr.**abcVol**(*a=None*, *b=None*, *c=None*, *alpha=None*, *beta=None*, *gamma=None*, *ibrav=None*)
Get volume from a,b,c,alpha,beta,gamma

Parameters **None** –

Keyword Arguments

- **a** (*float*) – length of a
- **b** (*float*) – length of b
- **c** (*float*) – length of c
- **alpha** (*float*) – Angle between axis b and c
- **beta** (*float*) – Angle between axis a and c
- **gamma** (*float*) – Angle between axis a and b
- **ibrav** (*int*) – ibrav to be used to convert for defaults

Returns **cell_vol** – volume of the cell

Return type float

retr.**attachPosFlags**(*positionString*, *flagString*)
Reattaches the flags to the end of the atomic position

Parameters

- **positionString** (*str*) – string of the atomic positions
- **flagString** (*str*) – string of the flags

Keyword Arguments **None**

Returns **positionString** – Positions with flags attached

Return type str

retr.**celldm2abc**(*ibrav=None*, *celldm1=None*, *celldm2=None*, *celldm3=None*, *celldm4=None*, *celldm5=None*, *celldm6=None*, *cosine=True*, *degrees=False*)
Convert celldm for espresso into A,B,C,and angles alpha,beta,gamma

Parameters **cellparamatrix** (*numpy.matrix*) – matrix of cell vectors

Keyword Arguments

- **cosine** (*bool*) – If True alpha,beta,gamma are cos(alpha),cos(beta),cos(gamma),
- **degrees** (*bool*) – If True return alpha,beta,gamma in degrees; radians if False

Returns **paramArray** – a list of the parameters a,b,c,alpha,beta,gamma generated from the input matrix

Return type list

retr.**celldm2free**(*ibrav=None*, *celldm1=None*, *celldm2=None*, *celldm3=None*, *celldm4=None*, *celldm5=None*, *celldm6=None*, *returnString=True*)
Converts QE's celldm format into QE convention primitive lattice vectors

Parameters **None** –

Keyword Arguments

- **ibrav** (*int*) – bravais lattice type by QE convention
- **celldm1** (*float*) – a
- **celldm2** (*float*) – b/a

- **celldm3** (*float*) – c/a
- **celldm4** (*float*) – Cosine of the angle between axis b and c
- **celldm5** (*float*) – Cosine of the angle between axis a and c
- **celldm6** (*float*) – Cosine of the angle between axis a and b
- **returnString** (*bool*) – return vectors as a string or as a numpy matrix

Returns Either a numpy.matrix or a string of the primitive lattice vectors generated from the celldm input

retr.**celldm2params**(*a*, *b*, *c*, *alpha*, *beta*, *gamma*, *k*, *v*)
DEFUNCT <Marked for removal>

Arguments:

Keyword Arguments:

Returns:

retr.**checkStatus**(*PROJECT*, *SET=''*, *config=''*, *step=0*, *status={}*, *negate_status=False*)
function that loads the calclogs of each of the calculations in your run and displays status information about them.

Parameters **PROJECT** (*str*) – Project name

Keyword Arguments

- **SET** (*str*) – Set name
- **config** (*str*) – Config file used
- **step** (*int*) – Which number step to seee (default all of them)
- **status** (*dict*) – dictionary containing the status and the value you want to see (ex. {'Error':True})
- **negate_status** (*bool*) – Whether to take the calculations with that status or without it

Returns calcsList (list) list of each step of the calculations you selected that satisfay the status criteria

retr.**chemAsKeys**(*calcs*)
For sets of calcs that only differ by chemistry you can replace the hash by the chemical name. May not always work. Especially if there are two compounds with swapped positions of atoms of different elements

Parameters **calcs** (*dict*) – dictionary of dictionaries of calculations

Keyword Arguments **None**

Returns **calcsCopy** – returns new dictionay with keys as the chemical stoiciometry

Return type dict

retr.**compMatrices**(*matrix1*, *matrix2*)
Compare two numpy matrices to see if they are equal or not

Parameters

- **matrix1** (*numpy.matrix*) – first matrix
- **matrix2** (*numpy.matrix*) – second matrix

Keyword Arguments **None**

Returns **equalBool** – True if they're equal False if they're not

Return type bool

retr.**convertFCC**(*oneCalc*, *ID*)

Parameters

- **oneCalc** (*dict*) – a dictionary containing properties about the AFLOWpi calculation
- **ID** (*str*) – ID string for the particular calculation and step

Keyword Arguments:

Returns:

retr.**detachPosFlags**(*positionString*)

Detach the position flags from the string of the atomic positions

Parameters **positionString** (*str*) – string of the atomic positions

Keyword Arguments **None**

Returns **positions** – atomic positions as a list flags (list): a list of the flags for by each position

Return type list

retr.**free2abc**(*cellparamatrix*, *cosine=True*, *degrees=True*, *string=True*, *bohr=False*)

Convert lattice vectors to a,b,c,alpha,beta,gamma of the primitive lattice

Parameters **cellparamatrix** (*numpy.matrix*) – matrix of cell vectors

Keyword Arguments

- **cosine** (*bool*) – If True alpha,beta,gamma are cos(alpha),cos(beta),cos(gamma),
- **degrees** (*bool*) – If True return alpha,beta,gamma in degrees; radians if False
- **string** (*bool*) – If True return a,b,c,alpha,beta,gamma as a string; if False return as a list
- **bohr** (*bool*) – If True return a,b,c in bohr radii; if False return in angstrom

Returns **paramArray** – a list of the parameters a,b,c,alpha,beta,gamma generated from the input matrix

Return type list

retr.**free2ibrav**(*cellparamatrix*, *ibrav=0*, *primitive=True*)

Convert lattice vectors to celldm

Parameters **cellparamatrix** (*numpy.matrix*) – matrix of cell vectors

Keyword Arguments

- **ibrav** (*int*) – Overrides the bravais lattice automatically detected (must be in QE convention if primitive)
- **primitive** (*bool*) – If True it will treat cellparamatrix as primitive lattice vectors

Returns **ibravStr** – string of the celldm used for QE pwscf input

Return type str

retr.**getBravaisLatticeName**(*bravaisLatticeNumber*)

Returns the name of the crystal system from the bravias lattice number

Parameters **braviasLatticeNumber** (*int*) – the number of the bravais lattice in QE convention

Keyword Arguments **None**

Returns A string of the name of the crystal system

retr.**getCellMatrixFromInput**(*inputString*, *string=False*, *scale=True*)
Get the primitive cell vectors from the input.

Parameters

- **oneCalc** (*dict*) – a dictionary containing properties about the AFLOWpi calculation
- **ID** (*str*) – ID string for the particular calculation and step

Keyword Arguments

- **string** (*bool*) – Return a string or matrix
- **scale** (*bool*) – scale the vectors by alat

Returns **cellParamMatrix** – primitive lattice params

Return type numpy.matrix

retr.**getCellOutput**(*oneCalc*, *ID*)
retreives information about the structure and its chemistry and prints it into a file in the AFLOWpi folder inside the project/set directories

Parameters

- **oneCalc** (*dict*) – a dictionary containing properties about the AFLOWpi calculation
- **ID** (*str*) – ID string for the particular calculation and step

Keyword Arguments **None**

Returns **outputStr** – Output of the report for the calculation

Return type str

retr.**getCellVolume**(*oneCalc*, *ID*, *conventional=True*, *string=True*)
Gets the cell volume from output if avaiable and if not gets it from the input

Parameters

- **oneCalc** (*dict*) – a dictionary containing properties about the AFLOWpi calculation
- **ID** (*str*) – ID string for the particular calculation and step

Keyword Arguments

- **conventional** (*bool*) – return the volume of the primitive or conventional lattice
- **string** (*bool*) – to return as a string or as a float

Returns **vol** – Volume of the cell

Return type str

retr.**getCellVolumeFromVectors**(*cellInput*)
Calculates cell volume from basis vectors

Parameters **cellInput** (*numpy.matrix*) – basis set defining the cell

Keyword Arguments **None**

Returns **vol** – volume of the cell (may be not scaled. it depends on your input matrix)

Return type float

retr.**getForce**(*oneCalc*, *ID*)
Gets last entry of the total force in the calculation from the engine output.

Parameters

- **oneCalc** (*dict*) – a dictionary containing properties about the AFLOWpi calculation
- **ID** (*str*) – ID string for the particular calculation and step

Keyword Arguments None

Returns **force_string** – Total force from engine output

Return type str

retr.**getIbravFromVectors**(*cellVectors*)
Arguments:

Keyword Arguments:

Returns:

retr.**getPointGroup**(*oneCalc*, *ID*, *source='input'*)
NOT COMPLETED.

Parameters

- **oneCalc** (*dict*) – a dictionary containing properties about the AFLOWpi calculation
- **ID** (*str*) – ID string for the particular calculation and step

Keyword Arguments **source** (*string*) – either 'input' or 'output' to get point group from input or output atomic positions

Returns None

retr.**getPositionsFromInput**(*oneCalc*, *ID*)

Parameters

- **oneCalc** (*dict*) – a dictionary containing properties about the AFLOWpi calculation
- **ID** (*str*) – ID string for the particular calculation and step

Keyword Arguments:

Returns:

retr.**getPositionsFromOutput**(*oneCalc*, *ID*)
Get the atomic positions from the output. If atomic positons can not be read from output then the the positions from the input are returned.

Parameters

- **oneCalc** (*dict*) – a dictionary containing properties about the AFLOWpi calculation
- **ID** (*str*) – ID string for the particular calculation and step

Keyword Arguments None

Returns **pos** – atomic positions

Return type numpy.matrix

retr.**getRecipParams**(*oneCalc*)
reads the output from the SCF or relax calculation to get the reciprocal cell parameters produced by the calculation.

Parameters **oneCalc** (*dict*) – a dictionary of a single calculation

Keyword Arguments None

Returns **alat** – alat multiplier paramMatrix (numpy.matrix): matrix of reciprocal lattice vectors

Return type float

retr.**get_parameters**(*oneCalc*, *ID*, *conventional=True*)

retr.**glideXshiftY**(*symMatrix*, *cellMatrix*, *shift*)

retr.**glideXshiftZ**(*symMatrix*, *cellMatrix*, *shift*)

retr.**glideYshiftX**(*symMatrix*, *cellMatrix*, *shift*)

retr.**glideYshiftZ**(*symMatrix*, *cellMatrix*, *shift*)

retr.**glideZshiftX**(*symMatrix*, *cellMatrix*, *shift*)

retr.**glideZshiftY**(*symMatrix*, *cellMatrix*, *shift*)

retr.**grabEnergy**(*oneCalc*, *ID*)

Grabs energy for oneCalc and adds the keyword 'Energy' with the value of the energy grabbed from the output

Parameters

- **oneCalc** (*dict*) – a dictionary containing properties about the AFLOWpi calculation
- **ID** (*str*) – ID string for the particular calculation and step

Keyword Arguments **None**

Returns

oneCalc – a dictionary containing properties about the AFLOWpi calculation

with the 'Energy' keyword added

Return type dict

retr.**grabEnergyOut**(*calcs*)

Goes in every subdirectory of the calculation and searches for the final energy of the calculation and returns a new copy of the input dictionary that includes the final energy.

Parameters **calcs** (*dict*) – dictionary of dictionaries of calculations

Keyword Arguments **None**

Returns **calcs** – dictionary of dictionaries of calculations with energy added

Return type dict

retr.**ibrav2String**(*ibrav=None*, *celldm1=None*, *celldm2=None*, *celldm3=None*, *celldm4=None*, *celldm5=None*, *celldm6=None*)

DEFUNCT <CONSIDER FOR REMOVAL>

Arguments:

Keyword Arguments:

Returns:

retr.**inputDict2params**(*inputDict*)

Arguments:

Keyword Arguments:

Returns:

retr.**invertX**(*symMatrix*, *cellMatrix*)

Parameters

- **symMatrix** (*numpy.matrix*) – atomic positions in matrix form
- **cellMatrix** (*numpy.matrix*) – primitive lattice vectors in matrix form

Keyword Arguments:

Returns:

retr.**invertXYZ**(*symMatrix*, *cellMatrix*)

Parameters

- **symMatrix** (*numpy.matrix*) – atomic positions in matrix form
- **cellMatrix** (*numpy.matrix*) – primitive lattice vectors in matrix form

Keyword Arguments:

Returns:

retr.**invertY**(*symMatrix*, *cellMatrix*)

Parameters

- **symMatrix** (*numpy.matrix*) – atomic positions in matrix form
- **cellMatrix** (*numpy.matrix*) – primitive lattice vectors in matrix form

Keyword Arguments:

Returns:

retr.**invertZ**(*symMatrix*, *cellMatrix*)

Parameters

- **symMatrix** (*numpy.matrix*) – atomic positions in matrix form
- **cellMatrix** (*numpy.matrix*) – primitive lattice vectors in matrix form

Keyword Arguments:

Returns:

retr.**pw2cif**(*calcs*, *inpt=True*, *outp=True*, *runlocal=False*, *outputFolder=None*, *filePrefix=''*)

Writes a simple CIF file for engine input and outputs for viewing the structure

Parameters **calcs** (*dict*) – dictionary of dictionaries of calculations

Keyword Arguments

- **inpt** (*bool*) – Do CIF for input
- **outp** (*bool*) – Do CIF for output
- **outputFolder** (*str*) – Output directory for the CIF
- **filePrefix** (*str*) – Optional prefix to the CIF filenames

Returns None

retr.**rotateAlpha**(*symMatrix*, *cellMatrix*, *angle*)

Parameters

- **symMatrix** (*numpy.matrix*) – atomic positions in matrix form
- **cellMatrix** (*numpy.matrix*) – primitive lattice vectors in matrix form

Keyword Arguments:

Returns:

retr.**rotateBeta**(*symMatrix*, *cellMatrix*, *angle*)

Parameters

- **symMatrix** (*numpy.matrix*) – atomic positions in matrix form
- **cellMatrix** (*numpy.matrix*) – primitive lattice vectors in matrix form

Keyword Arguments:

Returns:

retr.**rotateGamma**(*symMatrix*, *cellMatrix*, *angle*)

Parameters

- **symMatrix** (*numpy.matrix*) – atomic positions in matrix form
- **cellMatrix** (*numpy.matrix*) – primitive lattice vectors in matrix form

Keyword Arguments:

Returns:

retr.**shiftCell**(*symMatrix*)

Parameters **symMatrix** (*numpy.matrix*) – atomic positions in matrix form

Keyword Arguments:

Returns:

retr.**shiftX**(*symMatrix*, *cellMatrix*, *shift*)

Parameters

- **symMatrix** (*numpy.matrix*) – atomic positions in matrix form
- **cellMatrix** (*numpy.matrix*) – primitive lattice vectors in matrix form

Keyword Arguments:

Returns:

retr.**shiftY**(*symMatrix*, *cellMatrix*, *shift*)

Parameters

- **symMatrix** (*numpy.matrix*) – atomic positions in matrix form
- **cellMatrix** (*numpy.matrix*) – primitive lattice vectors in matrix form

Keyword Arguments:

Returns:

retr.**shiftZ**(*symMatrix*, *cellMatrix*, *shift*)

Parameters

- **symMatrix** (*numpy.matrix*) – atomic positions in matrix form
- **cellMatrix** (*numpy.matrix*) – primitive lattice vectors in matrix form

Keyword Arguments:

Returns:

retr.**transform_input_conv**(*oneCalc*, *ID*)

Parameters

- **oneCalc** (*dict*) – a dictionary containing properties about the AFLOWpi calculation
- **ID** (*str*) – ID string for the particular calculation and step

Keyword Arguments:

Returns:

retr.**writeInputFromOutput**(*calcs*, *replace=False*, *runlocal=False*)

Parameters **calcs** (*dict*) – dictionary of dictionaries of calculations

Keyword Arguments

- **replace** (*bool*) – If true replace the input with updated atomic positions and cell parameters
- **runlocal** (*bool*) – run local or write to <ID>.py

Returns None

CHAPTER 5

run module

run.**addatexit__**(*command*, **args*, ***kwargs*)
Wrapper to add function to be run at exit

Parameters

- **command** (*func*) – function to be run
- ***args** – arguments for command

Keyword Arguments ****kwargs** – Keyword arguments for command

Returns None

run.**bands**(*calcs*, *engine=''*, *execPrefix=None*, *execPostfix=' '*, *holdFlag=True*, *config=None*)
Wrapper to set up Electronic Band Structure calculation

Parameters **calcs** (*dict*) – Dictionary of dictionaries of calculations

Keyword Arguments

- **engine** (*str*) – executable that you are calling to run the calculations
- **execPrefix** (*str*) – commands to go before the executable when run (ex. mpiexec nice -n 19 <executable>) (default = None)
- **execPostfix** (*str*) – commands to go after the executable when run (ex. <execPrefix> <executable> -ndiag 12 -nimage 2) (default = None)
- **holdFlag** (*bool*) – DEFUNCT. NEEDS REMOVAL
- **config** (*str*) – DEFUNCT. NEEDS REMOVAL

Returns None

run.**clean_cell_params**(*output*)
Parses the atomic shifts in a supercell from fd.x outputted pw.x input files to correct for formatting issues when they are imported to AFLOWpi

Parameters **output** (*str*) – pw.x input files generated by fd.x

Keyword Arguments **None**

Returns **output** – pw.x input files generated by fd.x (cleaned by AFLOWpi)

Return type str

run.**cleanup**(*calcs*)
Deletes all files a calculation set's directory tree that are prepended with '_'

Parameters **calcs** (*dict*) – Dictionary of dictionaries of calculations

Keyword Arguments None

Returns None

run.**dos**(*calcs*, *engine=''*, *execPrefix=None*, *execPostfix=None*, *holdFlag=True*, *config=None*)
Wrapper to set up DOS nscf calculation

Parameters **calcs** (*dict*) – Dictionary of dictionaries of calculations

Keyword Arguments

- **engine** (*str*) – executable that you are calling to run the calculations
- **execPrefix** (*str*) – commands to go before the executable when run (ex. mpiexec nice -n 19 <executable>) (default = None)
- **execPostfix** (*str*) – commands to go after the executable when run (ex. <execPrefix> <executable> -ndiag 12 -nimage 2) (default = None)
- **holdFlag** (*bool*) – DEFUNCT. NEEDS REMOVAL
- **config** (*str*) – DEFUNCT. NEEDS REMOVAL

Returns None

run.**emr**(*calcs*, *engine=''*, *execPrefix=None*, *execPostfix=None*, *holdFlag=True*, *config=None*)
Wrapper to set up GIPAW EMR calculation

Parameters **calcs** (*dict*) – Dictionary of dictionaries of calculations

Keyword Arguments

- **engine** (*str*) – executable that you are calling to run the calculations
- **execPrefix** (*str*) – commands to go before the executable when run (ex. mpiexec nice -n 19 <executable>) (default = None)
- **execPostfix** (*str*) – commands to go after the executable when run (ex. <execPrefix> <executable> -ndiag 12 -nimage 2) (default = None)
- **holdFlag** (*bool*) – DEFUNCT. NEEDS REMOVAL
- **config** (*str*) – DEFUNCT. NEEDS REMOVAL

Returns None

run.**generateSubRef**(*qsubRefFileString*, *oneCalc*, *ID*)
Reads in the reference cluster submission file specified in "jobreffile" in the config used. Tries to insert a few parameters.

OBSOLETE PLANNED FOR REMOVAL

Parameters

- **qsubRefFileString** (*str*) – string of the "reference" cluster submission file
- **oneCalc** (*dict*) – dictionary of one of the calculations
- **ID** (*str*) – ID of calculation

Keyword Arguments None

Returns **clusterTypeDict** – string cluster submission file

Return type dict

run.**gvectors**(*calcs*, *engine=''*, *execPrefix=None*, *execPostfix=None*, *holdFlag=True*, *config=None*)
Wrapper to set up GIPAW gvectors calculation

Parameters **calcs** (*dict*) – Dictionary of dictionaries of calculations

Keyword Arguments

- **engine** (*str*) – executable that you are calling to run the calculations
- **execPrefix** (*str*) – commands to go before the executable when run (ex. mpiexec nice -n 19 <executable>) (default = None)
- **execPostfix** (*str*) – commands to go after the executable when run (ex. <execPrefix> <executable> -ndiag 12 -nimage 2) (default = None)
- **holdFlag** (*bool*) – DEFUNCT. NEEDS REMOVAL
- **config** (*str*) – DEFUNCT. NEEDS REMOVAL

Returns None

run.**hyperfine**(*calcs*, *engine=''*, *execPrefix=None*, *execPostfix=None*, *holdFlag=True*, *config=None*, *isotope=()*)

Wrapper to set up GIPAW hyperfine calculation

Parameters **calcs** (*dict*) – Dictionary of dictionaries of calculations

Keyword Arguments

- **engine** (*str*) – executable that you are calling to run the calculations
- **execPrefix** (*str*) – commands to go before the executable when run (ex. mpiexec nice -n 19 <executable>) (default = None)
- **execPostfix** (*str*) – commands to go after the executable when run (ex. <execPrefix> <executable> -ndiag 12 -nimage 2) (default = None)
- **holdFlag** (*bool*) – DEFUNCT. NEEDS REMOVAL
- **config** (*str*) – DEFUNCT. NEEDS REMOVAL

Returns None

run.**nmr**(*calcs*, *engine=''*, *execPrefix=None*, *execPostfix=None*, *holdFlag=True*, *config=None*)

Wrapper to set up GIPAW NMR calculation

Parameters **calcs** (*dict*) – Dictionary of dictionaries of calculations

Keyword Arguments

- **engine** (*str*) – executable that you are calling to run the calculations
- **execPrefix** (*str*) – commands to go before the executable when run (ex. mpiexec nice -n 19 <executable>) (default = None)
- **execPostfix** (*str*) – commands to go after the executable when run (ex. <execPrefix> <executable> -ndiag 12 -nimage 2) (default = None)
- **holdFlag** (*bool*) – DEFUNCT. NEEDS REMOVAL
- **config** (*str*) – DEFUNCT. NEEDS REMOVAL

Returns None

run.**pdos**(*calcs*, *engine=''*, *execPrefix=None*, *execPostfix=''*, *holdFlag=True*, *config=None*)

Wrapper to set up DOS projection calculation

Parameters **calcs** (*dict*) – Dictionary of dictionaries of calculations

Keyword Arguments

- **engine** (*str*) – executable that you are calling to run the calculations

- **execPrefix** (*str*) – commands to go before the executable when run (ex. mpiexec nice -n 19 <executable>) (default = None)
- **execPostfix** (*str*) – commands to go after the executable when run (ex. <execPrefix> <executable> -ndiag 12 -nimage 2) (default = None)
- **holdFlag** (*bool*) – DEFUNCT. NEEDS REMOVAL
- **config** (*str*) – DEFUNCT. NEEDS REMOVAL

Returns None

run.**prep_fd**(*__submitNodeName__*, *oneCalc*, *ID*, *nrx1=2*, *nrx2=2*, *nrx3=2*, *innx=2*, *de=0.01*, *atom_sym=True*, *disp_sym=True*)

Generates input files for fd.x, fd_ifc.x, and matdyn.x for the finite difference phonon calculations.

Parameters

- **__submitNodeName__** (*str*) – String of hostname that cluster jobs should be submitted from
- **oneCalc** (*dict*) – dictionary of one of the calculations
- **ID** (*str*) – ID of calculation

Keyword Arguments

- **nrx1** (*int*) – supercell size for first primitive lattice vector
- **nrx2** (*int*) – supercell size for second primitive lattice vector
- **nrx3** (*int*) – supercell size for third primitive lattice vector
- **innx** (*int*) – how many differernt shifts in each direction for finite difference phonon calculation
- **de** (*float*) – amount to shift the atoms for finite differences

Returns None

run.**reset_logs**(*calcs*)

Removes log files from AFLOWpi directory

Parameters **calcs** (*dict*) – Dictionary of dictionaries of calculations

Keyword Arguments **None**

Returns None

run.**resubmit**(*calcs*)

Stages loaded calculation set to be resubmitted on a cluster

Parameters **calcs** (*dict*) – Dictionary of dictionaries of calculations

Keyword Arguments **None**

Returns None

run.**scf**(*calcs*, *engine=''*, *execPrefix=None*, *execPostfix=None*, *holdFlag=True*, *config=None*)

Wrapper to set up self-consitent calculation

Parameters **calcs** (*dict*) – Dictionary of dictionaries of calculations

Keyword Arguments

- **engine** (*str*) – executable that you are calling to run the calculations
- **execPrefix** (*str*) – commands to go before the executable when run (ex. mpiexec nice -n 19 <executable>) (default = None)

- **execPostfix** (*str*) – commands to go after the executable when run (ex. <execPrefix> <executable> -ndiag 12 -nimage 2) (default = None)
- **holdFlag** (*bool*) – DEFUNCT. NEEDS REMOVAL
- **config** (*str*) – DEFUNCT. NEEDS REMOVAL

Returns:

run.**submit**()
sets global __submit__flag__ so calculations will start when user script completes

Parameters **None** –

Keyword Arguments **None**

Returns None

run.**submitFirstCalcs__**(*calcs*)
Submits the first step of a calculation's pipeline

Parameters **calcs** (*dict*) – Dictionary of dictionaries of calculations

Keyword Arguments **None**

Returns None

run.**testOne**(*calcs*, *calcType='scf'*, *engine=''*, *execPrefix=None*, *execPostfix=None*, *holdFlag=True*, *config=None*)
Run all the calculation in the dictionary with a specific engine

Parameters **calcs** (*dict*) – Dictionary of dictionaries of calculations

Keyword Arguments

- **engine** (*str*) – executable that you are calling to run the calculations
- **execPrefix** (*str*) – commands to go before the executable when run (ex. mpiexec nice -n 19 <executable>) (default = None)
- **execPostfix** (*str*) – commands to go after the executable when run (ex. <execPrefix> <executable> -ndiag 12 -nimage 2) (default = None)

Returns None

run.**write_fdx_template**(*oneCalc*, *ID*, *nrx1=2*, *nrx2=2*, *nrx3=2*, *innx=2*, *de=0.01*, *atom_sym=True*, *disp_sym=True*)
Generates input files for fd.x, fd_ifc.x, and matdyn.x for the finite difference phonon calculations.

Parameters

- **oneCalc** (*dict*) – dictionary of one of the calculations
- **ID** (*str*) – ID of calculation

Keyword Arguments

- **nrx1** (*int*) – supercell size for first primitive lattice vector
- **nrx2** (*int*) – supercell size for second primitive lattice vector
- **nrx3** (*int*) – supercell size for third primitive lattice vector
- **innx** (*int*) – how many differernt shifts in each direction for finite difference phonon calculation
- **de** (*float*) – amount to shift the atoms for finite differences

Returns None

CHAPTER 6

scfuj module

scfuj.**WanT_bands**(*oneCalc*, *ID=None*, *eShift=10.0*, *nbnd=None*)
Make input files for WanT bands calculation

Parameters **calc_copy -- dictionary of dictionaries of calculations** (-) –

scfuj.**WanT_dos**(*oneCalc, ID=None, eShift=10.0, temperature=300.0, energy_range=[-10.0, 10.0]*)
Make input files for WanT bands calculation

Parameters **calc_copy -- dictionary of dictionaries of calculations** (-) –

scfuj.**WanT_epsilon**(*oneCalc, ID=None, eShift=10.0, temperature=300.0, energy_range=[0.1, 12.5]*)
Make input files for WanT bands calculation

Parameters **calc_copy -- dictionary of dictionaries of calculations** (-) –

scfuj.**acbn0**(*oneCalc*, *projCalcID*, *byAtom=False*)

scfuj.**checkOscillation**(*ID*, *oneCalc*, *uThresh=0.001*)

scfuj.**chkSpinCalc**(*oneCalc*, *ID=None*)
Check whether an calculation is spin polarized or not.

Arguments:

–oneCalc : dictionary of a single calculation.

scfuj.**chk_species**(*elm*)

scfuj.**evCurveMinimize**(*calcs*, *config=None*, *pThresh=10.0*, *final_minimization='vc-relax'*)

scfuj.**getU_frmACBN0out**(*oneCalc*, *ID*, *byAtom=False*)

scfuj.**maketree**(*oneCalc*, *ID*, *paodir=None*)
Make the directoy tree and place in the input file there

Parameters **calcs -- Dictionary of dictionaries of calculations** (-) –

Keyword Arguments

- **- pseudodir – path of pseudopotential files directory**
- **- paodir - path of pseudoatomic orbital basis set**

scfuj.**nscf_nosym_noinv**(*oneCalc*, *ID=None*, *kpFactor=1.5*)
Add the ncsf input to each subdir and update the master dictionary

Parameters

- **calc_copy -- dictionary of one calculation** (-) –
- **kpFactor -- multiplicative factor for kpoints from SCF to DOS (default** (-) –

2.

scfuj.**projwfc**(*oneCalc*, *ID=None*)
Run projwfc on each calculation

Parameters oneCalc -- dictionary of a single calculation (-) –

scfuj.**run**(*calcs*, *uThresh=0.001*, *nIters=20*, *mixing=0.0*)

scfuj.**run_transport**(*__submitNodeName__*, *oneCalc*, *ID*, *run_scf=True*, *run_transport_prep=True*, *run_bands=True*, *epsilon=True*, *temperature=300*)

scfuj.**scfprep**(*calcs*, *paodir=None*)

scfuj.**transport_prep**(*oneCalc*, *ID*)
sets up the environment do do scf->nscf->projwfc to get overlap for transport calcs

scfuj.**updateUvals**(*oneCalc*, *Uvals*, *ID=None*)
Modify scf input file to do a lda+u calculation.

Parameters

- **oneCalc -- Dictionary of one calculation** (-) –
- **Uvals -- Dictionary of Uvals** (-) –

CHAPTER 7

Indices and tables

- genindex
- modindex
- search

p

r

s