dfttk Documentation

Release 0.3.0

Yi Wang, Mingqing Liao, Brandon J. Bocklund, Shunli Shang, Lon

Jan 19, 2021

CONTENTS:

CHAPTER

ONE

LICENSE

CHAPTER

TWO

DFTTK: DENSITY FUNCTIONAL THEORY TOOL KIT

2.1 The goal and solution

The goal of DFTTK is to make high-throughput first-principles calculations as simple as possible. The density functional theory (DFT) based software VASP is employed to perform first-principles calculations. In addition thermodynamic properties via the quasiharmonic approach, we proposed that any property, as long as it is dependent on the volume or stain, can be predicted using a quasi-static approach implemented by our group according to (i) the predicted property-volume/strain relationship from first-principles calculations directly and (ii) the volume/strain-temperature relationship of materials from the quasiharmonic approach.

2.2 High-throughput calculations

By its definition, the term of "first-principles" represents a philosophy that the prediction is to be based on a basic, fundamental proposition or assumption that cannot be deduced from any other proposition or assumption. This implies that the computational formulations are based on the most fundamental theory of quantum mechanics - Schrödinger equation or density functional theory (DFT) and the inputs to the calculations must be based on well-defined physical constants – the nuclear and electronic charges. In another word, once the atomic species of an assigned material are known, the theory should predict the energy of all possible crystalline structures, without invoking any phenomenological fitting parameters.

However, to perform DFT calculations in reality, it still needs the user to have extensive experiences on a variety of parameter choices and a lot of human handling on numerical or system exceptions. In the last decade, we have been working on solving the problem by integrating our experiences accumulated on high-throughout DFT calculations into a software package named as DFTTK (DFT based toolkits) and opened to the community (https://www.dfttk.org).

2.3 The main functions of DFTTK

- Structure maker by protype and elemental substitution;
- Robust 0 K equilibrium volume optimization;
- Robust 0 K energy-volume curve optimization;
- Quasiharmonic phonon calculation;
- Born effective charge calculation;
- Elastic constant calculations.
- MongoDB database management
- Thermodynamic calculations and figure plots

2.4 DFTTK features

To perform DFT calculation using DFTTK, the user only needs to name the structure file called POSCAR by VASP, either prepared by user or produced by DFTTK by elemental substation on given prototype. DFTTK is developed on atomate from the [Materials Project](https://materialsproject.org/) which is built on three open-source Python libraries. The main benefits of atomate are its flexibility and data management platform, in particular the numerical convergence control and computational exception handling. DFTTK is able to predict properties at finite temperatures by phonon or Debye model for both stoichiometric and solution phases, featured by:

- High-throughput DFT calculation and postprocess;
- Postprocess plenty of data stored in MongoDB with one simple command;
- Compatible with Yphon package and phonopy;
- Can recover data from certain fizzled calculations;
- Can account thermal electron contribution to thermodynamic properties;
- Can calculate thermodynamic properties at 0 K and a few tenth K;
- Can perform doping calculations for semiconductors or thermoelectric materials under rigid band approximation;
- Can account the effect of thermal expansion/temperature on Seebeck coefficient, Lorenz number, thermal carrier concentrations;
- Automatic plot figures for more than 20 thermodynamic properties in the publishable resolution, including atomic volume, free energy, entropy, enthalpy, linear thermal expansion coefficient, isobartic specific heat, constant volume specific heat, lattice only specific heat, bulk modulus, Debye temperature, Seebeck coefficient, Lorenz number, absolute thermal electric force, etc.

CHAPTER THREE

INSTALLATION

- Requirements

  DFTTK requires YPHON, MongodB, Python 3. Python 2 support for NumPy ends 2019-01-01.

- Release version

```
pip install dfttk
mkdir dfttk #if dfttk folder not exist.
cd dfttk
dfttk config -mp -aci #a folder named "config" will be created where running␣
↪environmental info saved
```

- Development version

```
git clone https://github.com/PhasesResearchLab/dfttk.git
cd dfttk
pip install -e .
cd dfttk
dfttk config -mp -aci #a folder named "config" will be created where running␣
↪environmental info saved
```

3.1 YPHON

To postprocess the finite properties, the Yphon package is required. Yphon can be installed by run

```
cd ~
git clone https://github.com/yiwang62/YphonPackage
#Note: Usually the precompiled binaries should be executable in the common Linux/Unix␣
↪environment. If not, do the following:
```

```
cd YphonPackage/YPHON/YPHON
make
#Note: If errors reported in the compiling stage, insert one line #define R_OK 1␣
↪after #include
```

For csh user: the command search path should be changed by inserting line below into the .cshrc (.tcshrc) file

```
set path = (. ~/YphonPackage/YPHON/YPHON $BIN_PATH $path)
```

For bsh user: the command search path should be changed by inserting the lines below into the .bash_profile (.bashrc) file

```
PATH=.:~/YphonPackage/YPHON/YPHON:$BIN_PATH:$PATH
export PATH
```

3.2 Connect DFTTK to MongoDB server

Ask the MongoDB system manager for a json file named `db.json` to get your DFTTK results saved in MongoDB database. The `db.json` file contains something similiar to the following lines which should saved under the "dfttk/config" folder that was created by "dfttk config -mp -aci" command mentioned above.

```
{
    "database": "userid-results",
    "collection": "tasks",
    "admin_user": "userid",
    "admin_password": "BeFihJ2mrKGm",
    "readonly_user": "userid-ro",
    "readonly_password": "QIvaUT9ca6H8",
    "host": "146.186.149.69",
    "port": 27018,
    "aliases": {}
}
```

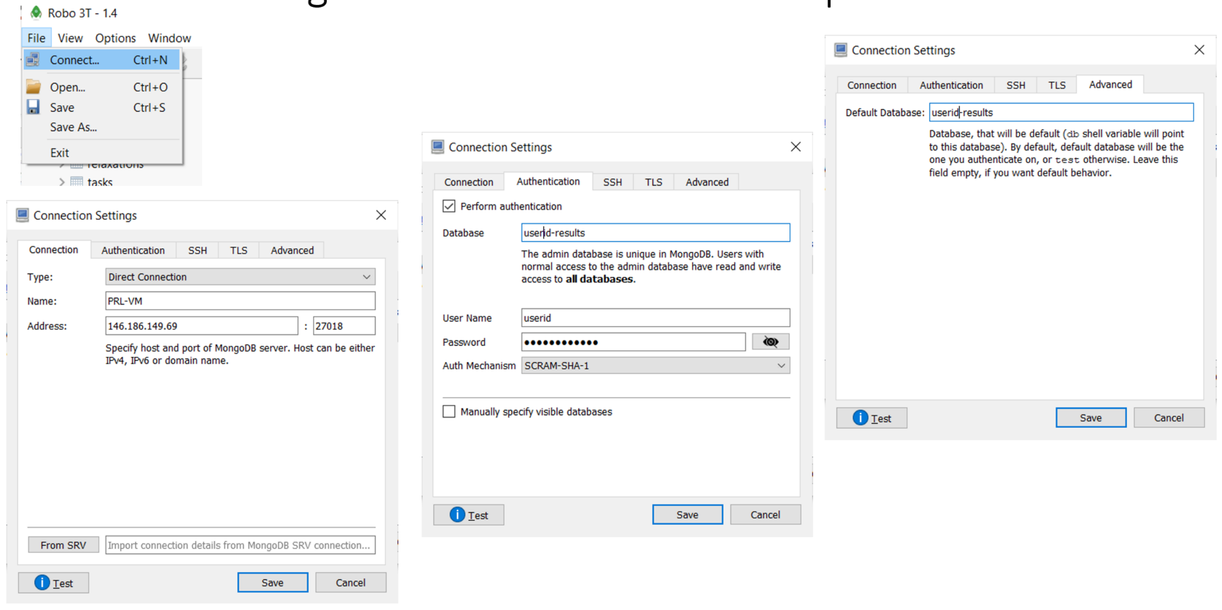
3.3 Access MongoDB database from desktop

One can download robo3T from https://robomongo.org/. After install it one use the information from the `db.json` file to setup robo3T connection as indicated in the following figure

#note

1. `PSU-VM` is an arbitray name for the connection that you want to use;
2. `146.186.149.69` is the ip address of one's MongoDB server;
3. One need to replace `userid` by one's own usedid provided by one's MongoDB system manage

Access MongoDB database from desktop: robo3t



Connection Settings

Connection | Authentication | SSH | TLS | Advanced

Type: Direct Connection

Name: PRL-VM

Address: 146.186.149.69 : 27018

Specify host and port of MongoDB server. Host can be either IPv4, IPv6 or domain name.

From SRV | Import connection details from MongoDB SRV connection...

Test | Save | Cancel

Connection Settings

Connection | Authentication | SSH | TLS | Advanced

☑ Perform authentication

Database: userid-results

The admin database is unique in MongoDB. Users with normal access to the admin database have read and write access to **all databases**.

User Name: userid

Password: ●●●●●●●●●●●●

Auth Mechanism: SCRAM-SHA-1

☐ Manually specify visible databases

Test | Save | Cancel

Connection Settings

Connection | Authentication | SSH | TLS | Advanced

Default Database: userid-results

Database, that will be default (db shell variable will point to this database). By default, default database will be the one you authenticate on, or test otherwise. Leave this field empty, if you want default behavior.

Test | Save | Cancel

CHAPTER

FOUR

EXAMPLES

The Examples folder is designed to keep the data for user to test the DFTTK package. The two subfolder are:

```
- dir ``Al/`` - contains all data for test run for Al
- dir ``ZrSiO4/`` - contains all data for test run for ZrSiO4
- dir ``ExptData/`` - contains a json file ``ExptData.json`` which saves the␣
↪experimental thermodynamic data for a collection of materials.
```

Within the `Al` or the `ZrSiO4` subfolder, one see two subfloders

- dir `input/` - contain input setup files using `Al` as the example on E-V, phonon, and thermodynamic property calculations
- Al_Fm-3m_225PBE/ - contain outputs by postprocessing data that saved in MongoDB by the above `Al` example.

For the data within Al_Fm-3m_225PBE/

- dir `Yphon/` - all data input/output for Yphon, e.x., hessian matrix (superfij.out), calculated phonon dos
- dir `figures/` - plots in png format for most of the thermodynamic properities
- file `readme` - extensive summary of the calculated results in json format
- file `fvib_ele` - tablated data containing the calculated thermodynamic properties
- file `fvib_eij` - tablated data containing the calculated thermal expansion coefficient tensor
- file `record.json` - SGTE fitting record for heat capacity, Gibbs energy, enthalpy, and entropy at given temperature range

To run the Example for the VASP calculation, say `Al`, run

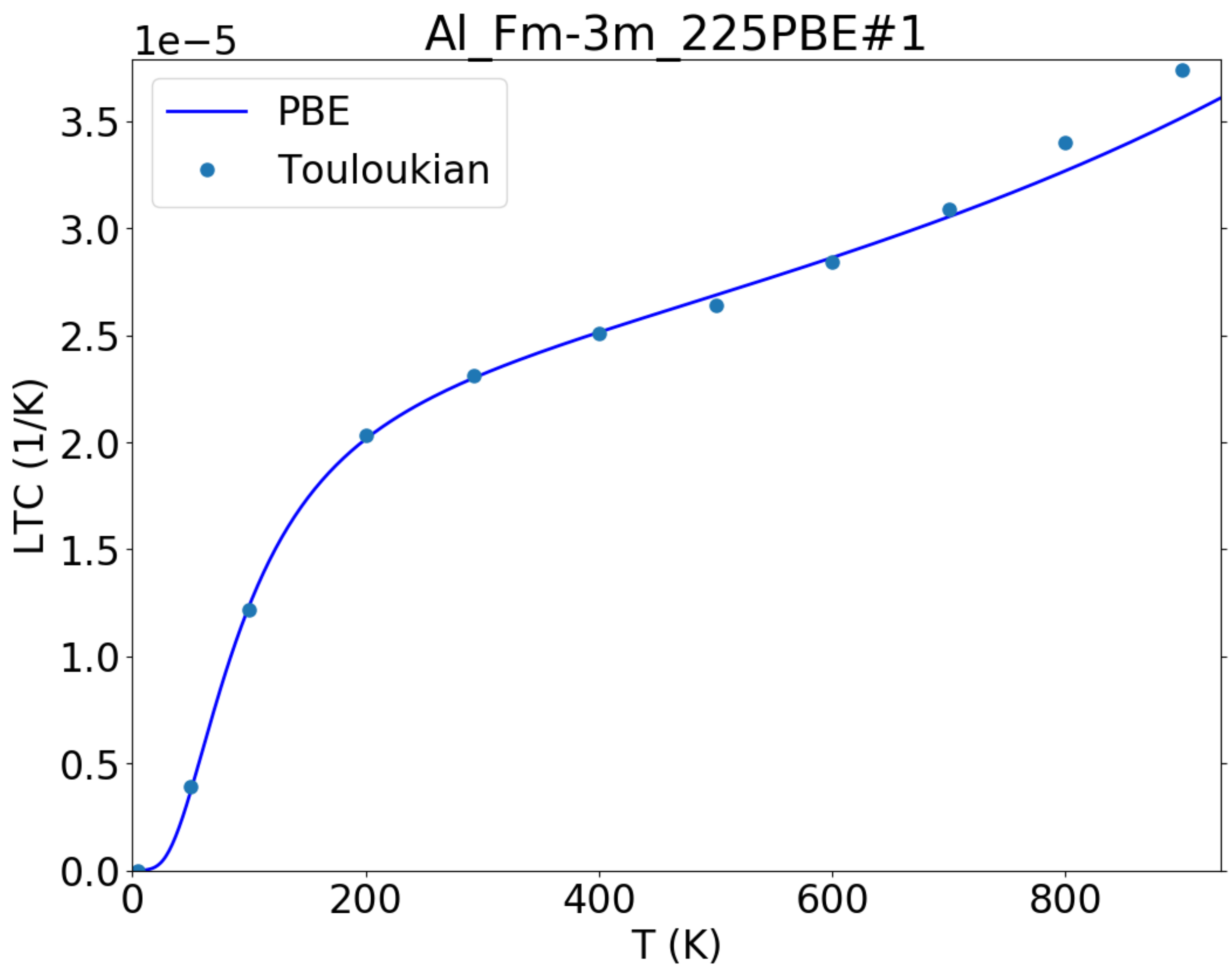
```
cd Al/input
dfttk run -wf robust -f POSCAR.Al -l -m 1
```

To postprocess calculations after the VASP calculation done, run

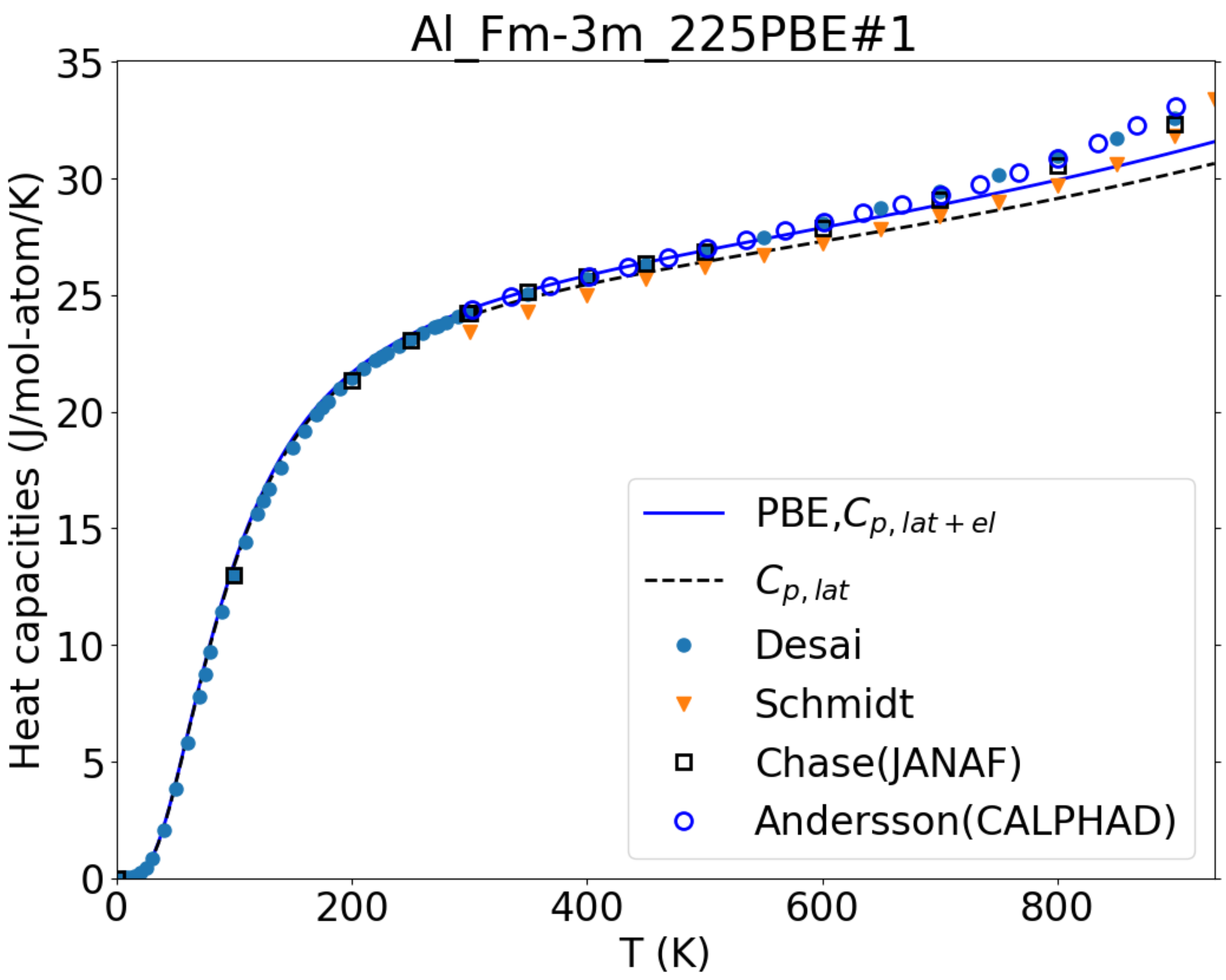
```
cd Al
dfttk thfind -py -td -50 -plot find_or_DFT -eq 4 -smooth -el 1 -renew -get -metatag␣
↪0c1887fa-0cb4-4ce2-9559-7f7909ffa11a -expt ../ExptData/ExptData.json
#note that the key ``0c1887fa-0cb4-4ce2-9559-7f7909ffa11a`` is obtained from the file␣
↪``input/METADATAS.yaml`` automatically produced by the VASP calculation step.
```

The above will produce more thatn 20 figures stored in the folder “Al_Fm-3m_225PBE/figures/” and they can be view them using the linux command `display` to show the figure, for example

```
display Al_Fm-3m_225PBE/figures/LTC.png #to see the linear thermal expansion␣
↪coefficient
```



```
display Al_Fm-3m_225PBE/figures/Heat_capacities.png #to see the heat capaticity, and␣
↪so on
```


Al_Fm-3m_225PBE#1
Heat capacities (J/mol-atom/K)
T (K)
0
200
400
600
800
0
5
10
15
20
25
30
35
PBE,$C_{p,lat+el}$
$C_{p,lat}$
Desai
Schmidt
Chase(JANAF)
Andersson(CALPHAD)

CHAPTER

FIVE

SETTINGS FILE

The user can change the settings for dfttk calculations by providing a settings file.

5.1 File name instruction

5.1.1 Globale settings

By default, the global settings file is named as `SETTINGS.yaml` or `SETTINGS.json`.

The user can change the name by `-s` parameter in `dfttk run`. e.g. if the user run dfttk by `dfttk run -s SET`, then the files named with `SET.yaml` or `SET.json` is the global settings file.

5.1.2 Individual settings

The user can provide individual settings for some structures whoes settings is different with others. The individual settings file should be named with `SETTINGS-FILENAME_WITH_EXT.yaml(json)` or `FILENAME_WITH_EXT-SETTINGS.yaml(json)`

Note:

- `FILENAME_WITH_EXT` is the file name of structure with extension.
- `SETTINGS` is case insensitive, both `settings` and `SETTINGS` are OK.

e.g. In the working folder, there are some structure files as follows:

```
Fe3Ni.cif
POSCAR
POSCAR-2
```

Then the following files will be recognized as settings files.

```
SETTINGS-Fe3Ni.cif.yaml
POSCAR-SETTINGS.yaml
settings-POSCAR-2.yaml
```

5.2 Keywords in the file

5.2.1 Default settings

The settings file is optional, if the user does not provide the settings file, the default value will be used.

- Common settings

Keywords	Default Value	Comments
magmom	No	The MAGMOM for each atom, e.g. [4, 4, -4, 4]. *Note*, the length must agree with the number of atoms
metadata	No	The metadata of the calculation. If the user provides it, DFTTK will find the existing calculations in the databasse. If not, it will generate by uuid4.
isif4	False	If run ISIF=4 following ISIF=7 in RobustOptmizeFW
level	1	Optimize level. If run ISIF=2 after last ISIF=4 in RobustOptimizeFW 1 for run and 2 for not
override_symmetry_tolerances	None	Override the default symmetry tolerance, if None, {'tol_strain':0.05,'tol_energy':0.025, 'tol_bond':0.10}
override_default_vasp_params	{}	Override the default vasp settings The optional keys is 'user_incar_settings', 'user_kpoints_settings', 'user_potcar_functional'. For more details, ref. https://pymatgen.org/pymatgen.io.vasp.sets.html
modify_incar_params	{}	Modify the incar settings in the fireworks level. e.g. modify_incar_params = { 'Full relax': {'incar_update': {"LCHARG":False}}, 'static': {'incar_update': "LAECHG":True}}
modify_kpoints_params	{}	Modify the kpoints settings in the fireworks level.
store_volumetric_data	False	Store the volumetric data (True) or not (False)
verbose	False	print(True) or not(False) some informations, for debug
passinitrun	False	Pass init vasp result. **It will be dropped in the future**
run_isif2	False	If run ISIF=2 before ISIF=4 (True) or not (False). **It will be dropped in the future**
pass_isif4	False	Whether pass isif=4 calculation. **It will be dropped in the future**
symmetry_tolerance	0.05	The tolerannce for symmetry. **It will be dropped in the future**
relax_path	''	The path of relaxiation. **It will be dropped in the future**

- Phonon settings

Keywords	Default Value	Comments
phonon	False	Run phonon (True) or not(False, Debye model)
phonon_supercell_matrix	atoms	The supercell matrix for phonon calculations. It can take the following values: **Matrix**, e.g. [[2, 0, 0], [0, 2, 0], [0, 0, 2]] **String**: atom/lattice/volume(the first letter works) Determining the supercell matrix automatically by atoms/lattice/volume ranges
phonon_supercell_matrix_min	60	The lower boundary for phonon_supercell_matrix(String)
phonon_supercell_matrix_max	130	The upper boundary for phonon_supercell_matrix(String)
force_phonon	False	Force run phonon (True) or not(False), No matter ISIF=4/stable_tor pass or not
stable_tor	0.01	Stable torlerance (The percentage of negative dos), If the negative part of DOS is larger than this value, DFTTK won't run phonon for this structure.

- QHA settings

Keywords	Default Value	Comments
num_deformations	7	The number of deformations/structures
deformation_fraction	[-0.1, 0.1]	The range of deformation, 0.1 means 10%
eos_tolerance	0.01	The tolerance for eos fitting. If larger than this value, DFTTK will append volumes automatically
t_min	5	The mimimum of temperature in QHA process
t_max	2000	The maximum of temperature in QHA process
t_step	5	The step of temperature in QHA process
volume_spacing_min	0.03	Minimum ratio of Volumes spacing. This keyword will be dropped in the future

- Elastic settings

Keywords	Default Value	Comments
strain_states	None	Strain modes, if it is None, it will generated by atomate
stencils	None	The amplitude of the strain modes/states
analysis	True	Analysis (True) or not (False)
sym_reduce	False	Reduce the strain according to the symmetry or not
order	2	The order of the elastic constants
conventional	False	Convert the structure into conventional format or not

CHAPTER

SIX

DFT JOB MONITORING

- Check run results

To get a brief summary for data in the database, use

```
dfttk thfind -v 1
```

this will report the metadata, number of finished phonon calculations, number of finished static calculations, supercell size used for the phonon calculations, brief phase name; and the following command reports the location where the DFT calculations were submitted

```
dfttk thfind -v 1 -jobpath parentfolder
```

where parentfolder is the parent folder whose subfolders hold the individual DFT calculation job submission data.

- Batch postprocessing results

```
dfttk thfind -get -py -td -50 -plot find_or_DFT -eq 4 -renew -ss 30 -w Pb-Ti-O
```

where `-get` instruct thfind to call `thelec` module to postprocess the data, `-py` to use Yphon to recalculate the phonon density of states based on the force constants saved in the `phonon` collection, `-td -50` to use the self-adapted temperature mesh, `-eq 4` to use Birch-Murnaghan fitting to get equilibrium volume, `-ss 30` means only handle results with supercell size larger thatn 30, -w elist means only extract for compounds with all elements in the list of elist. Other enhanced screening conditions can be combinations of the follows

```
-all elist # compounds must contain all elements in the list of elist;
-any elist # compounds contain any elements in the list of elist;
-xall elist # compounds must not contain all elements in the list of elist;
-xany elist # compounds must not contain an y elements in the list of elist.
```

Batch postprocess may take longer time to finish. If it is the case, it is recommended to submit batch job. When submit batch, make sure compatibilities of non-ascii character by including the following in the job script:

```
export LC_ALL='en_US.utf8' #for bsh;
setenv LC_ALL en_US.utf8 #for csh
```

- Monitor workflows

To check running status of all submitted dfttk jobs, use

```
lpad get_wflows
```

To find the running calculations

```
lpad get_fws -s RUNNING
```

To find the jobdir for a job with specific fw_id (The number portion of job id reported by `lpad get_wflows`), use lpad get_launchdir fw_id

- FIZZLED jobs

`FIZZLED` means failed. The following command can summarize all FIZZLED jobs:

```
lpad get_fws -s FIZZLED; or
lpad get_wflows -s FIZZLED
```

One can rerun `FIZZLED` by

```
lpad rerun_fws -s FIZZLED
```

For more details, see ref. FIZZLED.

To get help for the subcommands, try `lpad get_fws -h` or `lpad get_wflows -s FIZZLED -h`

CHAPTER SEVEN

TROUBLESHOOTING

This document covers how to handle jobs that have fizzled. There are three main sections: common troubleshooting, general troubleshooting workflow to find the root cause of issue and specific issues and their fixes.

7.1 Common troubleshooting

1. In the config stages, one may need properly set up VASP environments, such as

```
module load intel impi vasp
```

2. Make the latest automate installed;
3. In .bashrc/.cshrc, make sure not messed up with previous atomate FW config;
4. Make sure the setup in the .bashrc file or (equivalently in the .cshrc file)

```
export FW_CONFIG_FILE=~/dfttk/config/FW_config.yaml
```

5. git push issue for contributors, see https://docs.github.com/en/github/authenticating-to-github/adding-a-new-ssh-key-to-your-github-account
6. for batch run of the postprocessing modules, make sure compatibilities of non-ascii character by:

```
export LC_ALL='en_US.utf8' #for bsh;
setenv LC_ALL en_US.utf8 #for csh
```

7. For phonon calculations, due to certain reasons (such as temperature range too high), one may not see results in the ‘qha_phonon’ or “qha” MongoDB collections. In this case, the subcommand ‘dfttk thfind’ will try to find results from the “phonon” collection and process the data by calling “Yphon’
8. When you are interesting in revising the code, if have job running in the system before your changes, the codes in the batch system might not be updated and the results might be not as you assumed. It takes me two days to figure out this problem. The solution is to kill all the dfttk running job and resubmit tem.

7.2 Troubleshooting Workflow

My job has fizzled! The following steps can help you get information about how your job. You can imagine it as a decision tree. Check one thing before moving on to the next one.

1. Check that the job ran and has raised an exception with a traceback.

   Run `lpad get_fws -i <ID> -d more`, replacing `<ID>` with the integer id of the Firework. Search the output for `_exception`. What you see is the Python exception that was raised when running the Firework.

   TIP:: Searching works well when you pipe the output to `less` with `lpad get_fws -i <ID> -d more | less` and search using `/`.

2. Check the traceback is not a common error.

   See the Common Errors section

7.3 Common Errors

7.3.1 Custodian VasprunXMLValidator failed

In this error, you get a traceback that looks something like:

```
Traceback (most recent call last):
  File "/storage/home/bjb54/.conda/envs/wfs/lib/python3.7/site-packages/custodian/
↪custodian.py", line 320, in run
    self._run_job(job_n, job)
  File "/storage/home/bjb54/.conda/envs/wfs/lib/python3.7/site-packages/custodian/
↪custodian.py", line 428, in _run_job
    raise CustodianError(s, True, v)
custodian.custodian.CustodianError: (CustodianError(...), 'Validation failed:
↪<custodian.vasp.validators.VasprunXMLValidator object at 0x2af45b1d3908>')

During handling of the above exception, another exception occurred:

Traceback (most recent call last):
  File "/storage/home/bjb54/.conda/envs/wfs/lib/python3.7/site-packages/fireworks/
↪core/rocket.py", line 262, in run
    m_action = t.run_task(my_spec)
  File "/storage/home/bjb54/.conda/envs/wfs/lib/python3.7/site-packages/atomate/vasp/
↪firetasks/run_calc.py", line 204, in run_task
    c.run()
  File "/storage/home/bjb54/.conda/envs/wfs/lib/python3.7/site-packages/custodian/
↪custodian.py", line 330, in run
    .format(self.total_errors, ex))
RuntimeError: 0 errors reached: (CustodianError(...), 'Validation failed: <custodian.
↪vasp.validators.VasprunXMLValidator object at 0x2af45b1d3908>'). Exited...
```

With the key being that Custodian fails to validate the `vasprun.xml`. After running VASP, Custodian will try to parse the `vasprun.xml` file using pymatgen.

There are usually two possible triggers for this failure:

1. VASP failed to run at all (more common) or quit in a way that custodian did not detect (less common)
2. The `vasprun.xml` file could not be parsed by pymatgen.

To investigate this, first check that VASP ran (e.g. the `OUTCAR` shows that the run completed successfully). If VASP did not run, find out why and fix that issue. If VASP did run successfully, it was probably an issue parsing the `vasprun.xml` file. Try parsing the `vasprun.xml` file using the `pymatgen.io.vasp.outputs.Vasprun` class. If it throws an error when you try to parse, that's what made Custodian fail and you should fix that.

CHAPTER

EIGHT

MONGODB VIRTUAL MACHINES (VM) HOSTING

This section is for the dfttk users who want to host their MongoDB by themselves.

MongoDB is one of the most popular document-oriented databases under the banner of NoSQL database. The schema-free implementation of MongoDB eliminates the prerequisites of defining a fixed structure required by the SQL database.

In computing, a virtual machine (VM) is an emulation of a computer system. Virtual machines are based on computer architectures and provide functionality of a physical computer. Their implementations may involve specialized hardware, software, or a combination. See https://en.wikipedia.org/wiki/Virtual_machine

Our MongoDB databases are currently by Penn State's VM hosting service that provides cost-effective, reliable VM for departments, colleges, and research units at Penn State University.

In our case, we have changed the default tcp port from 27017 into 27018 due to historical reason

8.1 VM operation

Connect your VM by ssh (ssh youruserid@146.186.149.69 in our case). One should use VPN if you have firewall for your system

- To add user to your VM linux system

  Run the follwing command under Linux

```
sudo adduser newuser
usermod -aG sudo newuser #add admin user to your VM
```

Note on adding other users to access the VM. In order to add other users to access the VM from the Morpheus portal you would need to add them to the VM. For the VM itself you would need to add their user ID to the /etc/security/access.conf file and if they need sudo access you would need to add their ID to the /etc/sudoers.d/sudo-users file as well.

8.2 MongoDB operation

- MongoDB installation

  Run the follwing command

```
apt update
apt install mongodb
```

More information about installing and configuring MongoDB 3.6 is available here: https://docs.mongodb.com/manual/tutorial/install-mongodb-on-ubuntu/

If you would like the most recent version of MongoDB (4.4), please reference this guide for installation and configuration instructions for Ubuntu 18.04: https://docs.mongodb.com/manual/tutorial/install-mongodb-on-ubuntu/. #Note: In step 2 of the link listed above, please reference the command for Ubuntu 18.04 (Bionic)

Additional configuration info for MongoDB 4.4 can be found here: https://www.digitalocean.com/community/tutorials/how-to-install-mongodb-on-ubuntu-18-04-source

- Start mongdb service

```
mongod --bind_ip_all -port 27018 &
```

- Shut down mongdb service

```
db.adminCommand( { shutdown: 1 } )
```

For more details on MongodB user management, see https://docs.mongodb.com/manual/tutorial/enable-authentication/

- Connect to MongoDB by port 27018 for management

```
mongod -port 27018
```

- Quit from MongoDB

  hit `Ctrl+d`

- Create admin user for mongdb

After connected to your mongoDB by `mongo -port 27018`, input the following lines

```
use admin
db.createUser(
  {
    user: "admin",
    pwd: "xxxxxxxxx", // xxxxxxxx is the admin password of your choice
    roles: [ { role: "userAdminAnyDatabase", db: "admin" }, "readWriteAnyDatabase" ]
  }
)
```

- Create general user

Connect to your mongoDB as admin user by

```
mongo --port 27018 --authenticationDatabase "admin" -u "admin" -p
```

followed by inputting the following lines

```
use userid-fws
db.createUser({user: "userid", pwd: "B5nRcUvoCZ92", roles: [{role: "dbOwner", db:
↪"userid-fws"}]})
use userid-results
db.createUser({user: "userid", pwd: "BeFihJ2mrKGm", roles: [{role: "dbOwner", db:
↪"userid-results"}]})
db.createUser({user: "userid-ro", pwd: "QIvaUT9ca6H8", roles: [{role: "read", db:
↪"userid-results"}]})
```

These lines can be produced by dfttk by run a python code named `mongodb_user.py` which can be downlonded from https://github.com/PhasesResearchLab/dfttk/tree/master/dfttk/scripts After download the code, one can run it by

```
python mongodb_user.py
```

The run will prompt the MongoDB system manager to input an userid for the user. After you input userid and hit enter, one gets the above outputs in the screen.

Meanwhile, a file named `db.json` in the JSON format containing something similiar to the following lines which should be sent to the MongoDB user.

```
{
    "database": "userid-results",
    "collection": "tasks",
    "admin_user": "userid",
    "admin_password": "BeFihJ2mrKGm",
    "readonly_user": "userid-ro",
    "readonly_password": "QIvaUT9ca6H8",
    "host": "146.186.149.69",
    "port": 27018,
    "aliases": {}
}
```

The MongoDB user should save this data in a json file named `db.json` under the path `dfttk/config` that created by `dfttk config -mp -aci` command.

- Remove user

```
db.removeUser(username)
```

- Check if mongodb is running, use

```
ps -ef | grep mongo
```

CHAPTER

NINE

RECIPES

9.1 Construct a series of Debye workflows by substituting into a template structure

DRY_RUN = True # Don't submit the workflows VERBOSE = True # Turn on printing of substitutions

# Filename of the template structure to use, usually a dilute or SQS structure TEMPLATE_STRUCTURE_FILENAME = 'structures/FCC_A1.sqs.12Ni-4Fe.POSCAR'

# sublattice configuration of the template structure. # This will be substitued exactly, this does not need to be sorted, # however the individual configurations to build should be sorted. TEMPLATE_SUBLATTICE_CONFIGURATION = [['Fe', 'Ni']] TEMPLATE_SUBLATTICE_OCCUPANCIES = [[0.25, 0.75]] SUBLATTICE_SITE_RATIOS = [1.0]

PHASE_NAME = 'FCC_A1'

configurations_to_build = [# list of sublattice configurations in DFTTK format [['Cr', 'Ni']], [['Fe', 'Ni']], # [['V', 'Ni']], # should not use this because it's out of order, make a second script with the templates flipped]

# Dictionary of densities for each pure element. # Not necessary, using the peridic_table in pymatgen instead #DENSITY_DICT = { # 'V': 6.313, # bcc # 'Cr': 7.463, # bcc # 'Ni': 9.03, # fcc # 'Ti': 4.58, # hcp # 'Fe': 8.028 # bcc #}

# Should not need to edit below this line.

from pymatgen import Structure from fireworks import LaunchPad from dfttk import get_wf_gibbs from dfttk.structure_builders.substitutions import substitute_configuration_with_metadata

workflows = []

temp_struct = Structure.from_file(TEMPLATE_STRUCTURE_FILENAME)

for config in configurations_to_build: struct, meta = substitute_configuration_with_metadata(temp_struct, TEMPLATE_SUBLATTICE_CONFIGURATION, config, TEMPLATE_SUBLATTICE_OCCUPANCIES, PHASE_NAME, SUBLATTICE_SITE_RATIOS) if VERBOSE:

print("PHASE: {} CONFIGURATION: {} OCCUPANCIES: {} STRUCTURE: {}".format(PHASE_NAME, config, TEMPLATE_SUBLATTICE_OCCUPANCIES, struct.composition.hill_formula))

workflows.append(get_wf_gibbs(struct, deformation_fraction=(-0.05,0.10), phonon=False, num_deformations=11, t_max=2000, metadata=meta))

if VERBOSE: print("{} workflows.".format(len(workflows)))

if DRY_RUN: exit()

if VERBOSE: print('Adding workflows to LaunchPad')

lpad = LaunchPad.auto_load()

for workflow in workflows: lpad.add_wf(workflow)

9.2 ESPEI datasets from a QHA database

The following code snippet will take ESPEI datasets from a QHA database, optionally writing the (nicely named) files to dict. The QHA database requires the following metadata schema:

```
{
  'metadata': {
    'phase_name': 'FCC_A1',
    'tag': 'ed447049-ad67-4090-ba99-378188d3416b',
    'sublattice': {
      'configuration': [['Cr', 'Ni']],
      'occupancies': [[0.03125, 0.96875]]
     },
  }
}
```

```
########
# EDIT #
########

phase_name = 'BCC_A2'
configuration_to_find = [['Ni', 'V']]
sublattice_site_ratios = [1.0]
db_username = 'BrandonResultsRO'
db_password = 'piqhg38hap3'
db_uri = 'mongodb://206.189.190.225:27018'
WRITE_FILES = True

temperature_index = 59  # index of 300 K temperature (close to 298 K), found by hand

refstate_tags = {
    'Fe': '4ac77fce-0e43-4c07-8418-4843a2cd5723',
    'Ni': '0059ee69-4a8f-4e86-9895-9b40cf67dd96',
    'Cr': '8cceb186-2796-4488-ba8c-2380c5278f62',
    'V': 'fba46b6b-1699-419f-b5e1-da9533530701',
}

########################
### SCRIPT
########################

from dfttk.analysis.formation_energies import get_formation_energy, get_thermal_props
from dfttk.espei_compat import make_dataset, dfttk_config_to_espei, dfttk_occupancies_
↪to_espei
from pymatgen import Structure
import numpy as np
from dfttk.utils import recursive_flatten
import json
from pymongo import MongoClient
```

(continues on next page)

(continued from previous page)

```
# create the MongoClient
cli = MongoClient(db_uri)
db = cli.results
db.authenticate(name=db_username, password=db_password)
coll = db.qha

# construct the energies, assupmtion of same temperature grid
# energies are J/mol-atom
refstate_energies = {}
for el, tag in refstate_tags.items():
    qha_result = coll.find_one({'metadata.tag': tag})
    refstate_energies[el] = get_thermal_props(qha_result)

# calculate all the dataset values
configs     = []
occupancies = []
hm_values   = []
sm_values   = []
cpm_values  = []

# we'll change the T to the right temperatures later
fixed_conds = {'P': 101325, 'T': 0}
temp_conds = {'P': 101325, 'T': 0}


for qha_res in coll.find({'metadata.sublattice.configuration': configuration_to_find,
↪'metadata.phase_name': phase_name}):
    configs.append(qha_res['metadata']['sublattice']['configuration'])
    occupancies.append(qha_res['metadata']['sublattice']['occupancies'])

    tprops = get_thermal_props(qha_res)
    struct = Structure.from_dict(qha_res['structure'])
    hm_form = get_formation_energy(tprops, struct, refstate_energies, 'HM',␣
↪idx=temperature_index)
    sm_form = get_formation_energy(tprops, struct, refstate_energies, 'SM',␣
↪idx=temperature_index)
    cpm_form = get_formation_energy(tprops, struct, refstate_energies, 'CPM',␣
↪thin=10)[:-2]
    fixed_temp = tprops['T'][temperature_index]
    cpm_temps = tprops['T'][::10][:-2]

    hm_values.append(hm_form)
    sm_values.append(sm_form)
    cpm_values.append(cpm_form)

fixed_conds['T'] = fixed_temp.tolist()
temp_conds['T'] = cpm_temps.tolist()

# make the HM, SM, CPM values arrays of the proper shape
hm_values = np.array([[hm_values]])
sm_values = np.array([[sm_values]])
cpm_values = np.array(cpm_values).T[np.newaxis, ...]

if WRITE_FILES:
    # write JSON files
    comps = [c.upper() for c in sorted(recursive_flatten(configuration_to_find))]
    for prop, vals, conds in [('HM_FORM', hm_values, fixed_conds), ('SM_FORM', sm_
↪values, fixed_conds), ('CPM_FORM', cpm_values, temp_conds)]:
```

(continues on next page)

(continued from previous page)

```
        ds = make_dataset(phase_name, prop, sublattice_site_ratios, configs, conds,␣
↪vals, occupancies=occupancies, tag=tag)
        with open('{}-{}-{}-DFTTK.json'.format('-'.join(comps), phase_name, prop), 'w
↪') as fp:
            json.dump(ds, fp)
```

CHAPTER

TEN

FAQ

10.1 What do the references to DFTTK-style or ESPEI-style configurations or occupancies mean?

ESPEI is separate software for fitting thermodynamic databases within the CALPHAD method. The CALPHAD method requires descriptions of the formation and mixing energies of phases described by sublattices, as in the compound energy formalism.

DFTTK is an excellent tool for calculating the temperature and composition dependent thermodynamic properties of sublattice configurations within the compound energy formalism.

In both ESPEI and DFTTK, there are three key aspects to describing a phase in the CEF: sublattice site ratios, sublattice configurations, and sublattice occupancies. Sublattice site ratios are a list of integer or fraction ratios of the sublattice. There are no constraints on the order or values. Some valid ones would be:

```
[1]  # 1 sublattice
[3.0, 2.0]  # 2 sublattices with 3 occurrences of species in sublattice 1 and 2 in␣
↪sublattice 2
[1, 0.5]  # 2 sublattices with 1 occurrence of species in sublattice 1 and 1/2 in␣
↪sublattice 2
[16, 11]  # 2 sublattices with 16 occurrences of species in sublattice 1 and 11 in␣
↪sublattice 2
```

DFTTK describes a sublattice configuration as a list of sublattices, where each sublattice is also a list of the components occupying that sublattice. Some examples are:

```
[[Fe, Ni]]  # 1 sublattice with mixing of Fe and Ni
[[Mg], [Cu]]  # 2 sublattices with no mixing in each sublattice.
[[Cr, Fe], [Cr], [Cr, Fe]]  # 3 sublattices with mixing in the first and last␣
↪sublattice
```

The ordering of sublattices themselves have no distinct meaning. As a convention, DFTTK sorts the components in each sublattice by component name, e.g., Cr (starts with C) always comes before Fe (starts with F)). ESPEI requires each sublattice to be ordered in this way.

Sublattice occupancies or occupations describe how a sublattice is filled with each particular element. The shape of these corresponds exactly to the configuration that the occupancy list describes. The sorting of the occupancies therefore is also in the same order of the configuration.

DFTTK includes several tools to help build ESPEI-compatible sublattice descriptions. The only difference between DFTTK and ESPEI coniguations and occupancies is that a singly occuped sublattice, e.g. [[Fe]] in DFTTK, is not wrapped in the sublattice parenthesis for both the configuration and occupancies. Some examples of ESPEI configuration and occupancy lists are given below:

```
[[FE, NI]], [[0.5, 0.5]]  # this is the same as DFTTK, there are no single occupations
[MG, CU], [1.0, 1.0]  # each sublattice is singly-occupied and is not wrapped in the␣
↪parenthesis
[[CR, FE]], CR, [CR, FE]], [[0.5, 0.5], 1, [0.3, 0.7]] # the second sublattice is␣
↪singly occupied and is not wrapped
```

As a final note, ESPEI format typically uses all uppercase, consistent with the CALPHAD community, while DFTTK usually uses standard element capitalization. The conversion tools take this into account.

CHAPTER

ELEVEN

CHANGELOG

11.1 0.3.0 (2020-12-10)

(Contributor: @YiWang, @mxf469, @bocklund)

- Change List:
- formally release dfttk version 0.3.0
- Revised dfttk/script/run_dfttk.py
- made workflow `get_wf_gibbs` run on `robust` with `get_wf_gibbs_robust`
- changed test module from `get_wf_gibbs` into `get_wf_gibbs_robust`
- Add elasticity calculation model
- Add Born effective charge calculations

11.2 September 17, 2020

- Bug fixes:
- Replaced `curve-fit` by `ployfit` for BM fitting
- Fixed bugs on result checks
- Summarizing installation/run documents for DFTTK (This documents)

11.3 September 3, 2020

- Added a module `EVfind` per discussion with Shunli and enhanced calculation summary
- E-V, P-V, Stress-V, Strain-V
- Implemented 4 numerical schemes for evaluating LTC
- Enhanced plots
- LO-TO splitting were marked
- Gamma point phonon frequencies
- Prepared ~50 testing structures/compounds
- Made LDA pseudopotential as an option

- Speed up calculation
- Turned on `LREAL=Auto` from default
- Change into `ISPIN=1` for nonmagnetic system from default
- For better relaxation
- Turned on `EDIFF=1.e-8` (why it showed 2e-7 in the OUTCAR?)
- Turned on `EdiffG=-1.e-3` (testing, could be the reason for poor phonon data quality)
- Data report
- Phonon quality
- LTC quality
- Found phonopy was unable handle Cv below 1 K
- Added experimental data for plotting results
- Working on
- Calculate Born effective charge/dielectric tensors for insulators
- Launch by module `EVfind` when band gaps found and if not calculated

11.4 0.2.2 (2020-08-18)

(Contributor: @YiWang, @mxf469)

- Change List:
  - added codes in the dfttk/scripts directory:
    - run_dfttk_ext.py
    - handle the argumetns for the thelec and thfind modules
  - added python code in the dfttk directory:
    - pyfind.py
    - database search engine
    - pythelec.py
    - for compatibiliy with Yphon
    - generating the majority of thermodynamic properties, such as thermal expansion coefficient, Seebech coefficients, Lorenz number etc
    - pyphon.py for
    - calculate the phonon contributions to the various thermodynamic properties
  - added python code in the dfttk/analysis directory:
    - database
    - for plot phonon dispersions for all crystalline systems
    - ywutils.py
    - general utils code
    - ywplot.py

        - for plots of ~20 different phonon and thermodynamic properties in the png format

- made Yphon compatibile with phonopy
    - added codes in the CRO-soc directory:
        - phonopy2yphon, phonopy2yphon.py
        - convert the phonopy force constant matrix in hdf5 format into superfij.out format used by Yphon
    - changed codes:
        - in the dfttk/scripts directory:
            - run_dfttk.py
                - added the following lines aimed to handle the argumetns for the thelec and thfind modules

            # extension by Yi Wang, finalized on August 4, 2020 # ———————————— from dfttk.scripts.run_dfttk_ext import run_ext_thelec run_ext_thelec(subparsers)
        - in the dfttk/analysis directory:
            - debye.py is renamed as debye_ext.py
            - to include the vibrational entropy (S_vib) and heat capacity (C_vib) into the "qha" MongoDB collection
            - quasiharmonic.py:
            - copy the S_vib and C_vib from the "phonon" collection into the "qha_phonon" MongoDB collection

11.5 0.2 (2020-03-30)

New features

(Contributor: @bocklund , @Peng_Gao, @hitliaomq)

- The relax scheme is optimized. (from `ISIF=3` to `ISIF=2` followed by `ISIF=4`) (@Peng_Gao)
- Change the static workflow to dynamic workflow. (`EVcheck_QHA.py` increase the data points atomately if the fitting of initial points is incorrect) (@Peng_Gao)
- Support run dfttk by command. (Add `dfttk run [options]`) (@hitliaomq)
- Support configrate dfttk automately. (Add `dfttk config [options]`) (@hitliaomq)
- Documents' enhance. (@hitliaomq)
- Bug fix. (Including #8) (@bocklund, @Peng_Gao, @hitliaomq)

11.6 0.1 (2018-08-28)

Initial release. Includes

(Contributor: @bocklund, @mxf469)

- Gibbs workflow for stable structures
- Analysis code and libraries for calculation quasiharmonic Gibbs energies with 0K, vibrational and thermal electronic contributions
- Useful utilities for interfacing with structure, calculations and the Materials Project

CHAPTER

TWELVE

CONTRIBUTION GUIDE

12.1 Project Goal

Provide robust density functional theory workflows for calculating thermodynamic properties in temperature and composition space.

12.2 How to Contribute

1. Clone the repository to your local machine
2. Create a new branch for your addition or changes (`git checkout -b mybranchname`)
3. Write code or make changes and commit them to your branch
4. Push your branch to the repository (`git push origin mybranchname`)
5. Submit a pull request

After you submit a merge request, other members of the group are able to review your changes and give feedback. Someone with a rank of Master or higher in the project can merge your commits into the master branch.

12.3 Style Guidelines

In general, code style should follow PEP8 and PEP20. Specifics are summarized below:

- Code should be indented using spaces, not tabs. One indentation = 4 spaces
- Lines longer than 100 should be manually wrapped, but prefer readability
- Minimize blank lines: 2 around top level classes functions, 1 in nested functions
- Workflows, Fireworks, and Firetasks should follow the same naming scheme as in atomate
- Include docstrings for classes and functions (see code), add comments where needed
- Function and variable names should be descriptive (not 'x' or 'xx') and all lowercase_with_underscores
- Class names should be descriptive CapitalWords

CHAPTER THIRTEEN

RELEASING DFTTK

When releasing a new version of DFTTK

1. `git pull` to make sure you haven't missed any last-minute commits. **After this point, nothing else is making it into this version.**
2. Ensure that all tests pass locally on develop.
3. `git push` and verify all tests pass on all CI services.
4. Generate a list of commits since the last version with `git --no-pager log --oneline --no-decorate 0.1^..origin/master` Replace `0.1` with the tag of the last public version.
5. Condense the change list into something user-readable. Update and commit CHANGES.rst with the release date.``
6. `git tag 0.2 master -m "0.2"` Replace `0.2` with the new version. `git show 0.2` to ensure the correct commit was tagged `git push origin master --tags`
7. The new version is tagged in the repository. Now the public package must be built and distributed.

13.1 Uploading to PyPI

1. `rm -R dist/*` on Linux/OSX or `del dist/*` on Windows
2. With the commit checked out which was tagged with the new version: `python setup.py sdist`

   Make sure that the script correctly detected the new version exactly and not a dirty / revised state of the repo.

   Assuming a correctly configured .pypirc:

   `twine upload -u bocklund dist/*`

13.2 Some useful commands are following

```
git checkout master
#
git pull
git pull upstream master
git --version
git remote set-url --all upstream git@github.com:PhasesResearchLab/dfttk.git
git remote set-url --add upstream git@github.com:PhasesResearchLab/dfttk.git
git remote set-url --delete upstream git@github.com:PhasesResearchLab/dfttk.git
```

(continues on next page)

(continued from previous page)

```
git remote remove upstream
git remote rm upstream
git remote -v
git remote add upstream git@github.com:PhasesResearchLab/dfttk.git
#
git tag -d 0.3.0
git push --delete upstream 0.3.0
git tag 0.3.0 master -m "0.3.0"
git push upstream master --tags
rm dist/*
python setup.py sdist
twine upload -u yiwang62 dist/*
```

CHAPTER FOURTEEN

INDICES AND TABLES

- genindex
- modindex
- search