Active_Learning_Driver Documentation

Release 0.1.0

Rebecca K. Lindsey

Nov 16, 2022

CONTENTS

Note: This documentation is under still construction.

The Active Learning Driver (ALD) is an extensible multifunction workflow tool for generating ChIMES [1] models. At its simplest, the ALD can be used for model generation via iterative refinement [2], at at its most complex, via active learning [3].

Before proceeding, the user is **strongly** encouraged to familiarize themselves with the ChIMES literature (See references below) and ChIMES LSQ user manual. **UPDATE LINK** Note that the ALD itself *only* contains the tools necessary to orchestrate model generation and active learning, and must be used in conjunction with the ChIMES design matrix generator, a supported MD code, and a supported quantum code. Note also that the ALD is *only* intended for use on high performance computing platforms and currently only supports runs via slurm (SBATCH) schedulers. For additional details, see the *Quick Start* page.

[1] (link) R.K. Lindsey, L.E. Fried, N. Goldman, *JCTC*, **13**, 6222 (2017)
[2] (link) R.K. Lindsey, N. Goldman, L.E. Fried, S. Bastea, *JCP*, **153** 054103 (2020)
[3] (link) R.K. Lindsey, L.E. Fried, N. Goldman, S. Bastea, *JCP*, **153** 134117 (2020)

The ChIMES Calculator is developed at Lawrence Livermore National Laboratory with funding from the US Department of Energy (DOE), and is open source, distributed freely under the terms of the (specify) License.

For additional information, see:

CHAPTER

ONE

OVERVIEW: CHIMES ACTIVE LEARNING DRIVER

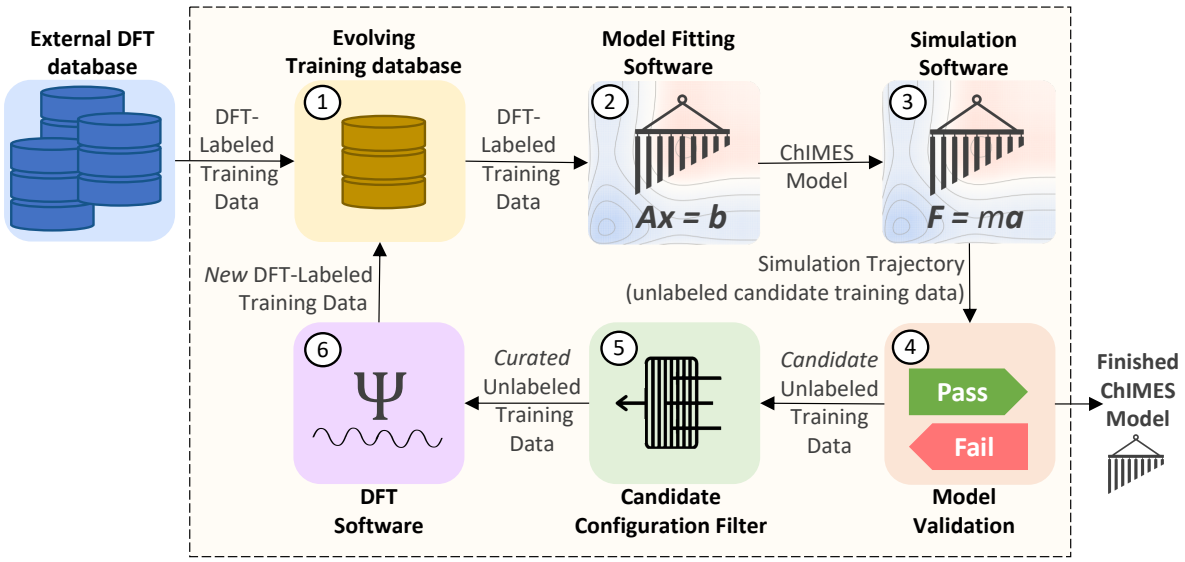

Fig. 1: **Fig. 1:** The ChIMES Active Learning Driver Workflow.

In most cases, generating robust and accurate ChIMES models necesitates an interative fitting strategy. As shown in Fig. 1, this strategy begins with selecting a number of seed configurations from a larger quantum-based data set (e.g., DFT, which will be used henceforth), which are added to an evolving training database (i.e., Fig. 1 step 1). In Fig. 1 step 2, a ChIMES model of user-specified hyperparamters generated based on the training database constructured in step 1. Generated parameters are then used to launch one or more ChIMES simulations of user-specified nature (step 3).

Fig. 1 step 4 comprises user inspection for model validation. This could entail inspecting the root-mean-squared-error in the fit from step 2, the conserved quantity from simulations in step 3, or comparing physical properties predicted via the simulations in step 3 (e.g., radial pair distribution function) to those predicted by DFT. If the user is satisfied with model performance at this step, they can terminate the ALD and proceed to production simulations with their model; otherwise, the ALD proceeds to step 5.

The 5th step, i.e. "candiate configuration filter" comprises selection of candidate unlabeled training data for assignment of DFT forces, energies, and or stresses (step 6) and subsequent addition to the evolving training database (step 1). This iterative fitting process continues until either the user-specified number of cycles is complete, or until user-terminated.

Additional information on different execution modes as well as example scripts can be found below:

1.1 Basic Fitting Mode

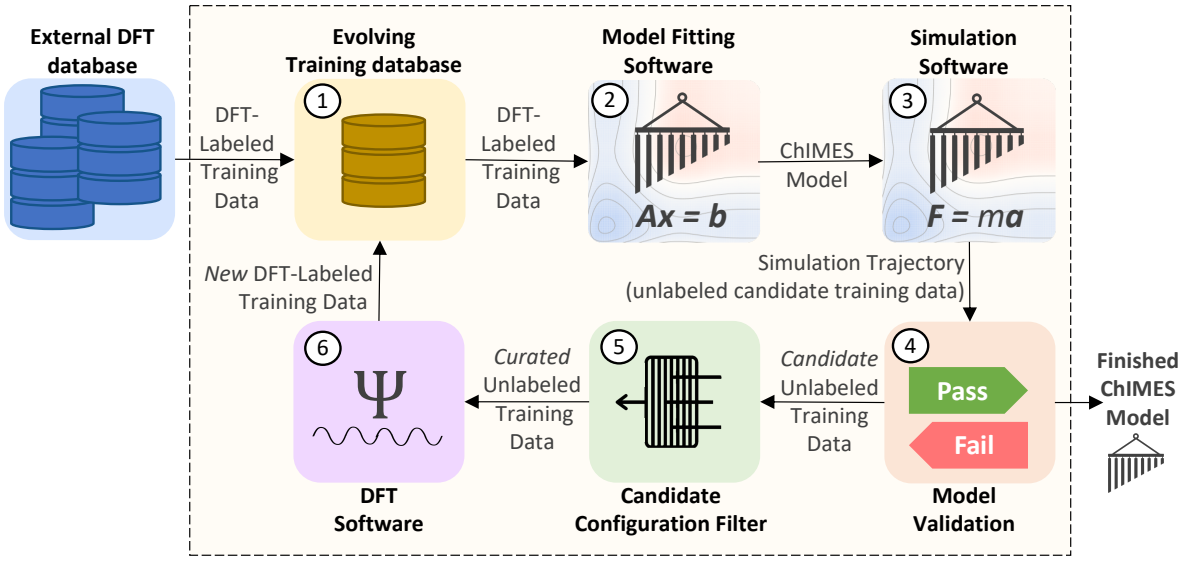

Fig. 2: **Fig. 1:** The ChIMES Active Learning Driver Workflow.

The "active learning" portion of the ALD largely entails intelligent strategies for selecting candidate unlabeled training data (step 5 in the schematic above). However, the ALD can also be run in a simpler iterative refinement scheme, which is quite efficient for low complexity non-reactive problems. In this page, a single-state-point fit is demonstrated using VASP, and all additional options for fitting in this model are overviewed.

1.1.1 Example Fit: Molten Carbon

Note: Files for this example are located in `./<al_driver base folder>/examples/simple_iter_single_statepoint`

In this section, an example 3-iteration fit for molten carbon at 6000 K and 2.0 g/cc is overviewed. The model will include up-to-three body interactions with the following hyperparameters. For more information on ChIMES hyperparameters and selection strategies, see:

- The ChIMES LSQ code manual (link)
- R.K. Lindsey, L.E. Fried, N. Goldman, *JCTC*, **13**, 6222 (2017) (link)
- R.K. Lindsey, L.E. Fried, N. Goldman, *JCTC* **15** 436 (2019) (link)

Hyperparameter	Value
2-body order	12
2-body outer cutoff	3.15
3-body order	4
3-body outer cutoff	3.15
inner cutoff	0.98
Morse lambda	1.25
Tersoff parameter	0.75

Input Files

The neccesary input files and directory tree structure are provided in the example folder, i.e.:

```
$: tree
.
├── ALL_BASE_FILES
│   ├── ALC-0_BASEFILES
│   │   ├── fm_setup.in
│   │   ├── liquid_6000K_2.0gcc.xyzf
│   │   └── traj_list.dat
│   ├── CHIMESMD_BASEFILES
│   │   ├── bonds.dat
│   │   ├── case-0.indep-0.input.xyz
│   │   ├── case-0.indep-0.run_md.in
│   │   └── run_molanal.sh
│   └── QM_BASEFILES
│       ├── 6000.INCAR
│       ├── C.POTCAR
│       └── KPOINTS
└── config.py
```

Briefly:

- `ALL_BASE_FILES/ALC-0_BASEFILES` contains files specifying how step 2 of figure 1 should be run, i.e., model hyperparameters (`fm_setup.in`), a list of training configuration files (`traj_list.dat`), and in this case, a single initial training configuration file (`liquid_6000K_2.0gcc.xyzf`).
- The `ALL_BASE_FILES/CHIMESMD_BASEFILES` directory contains files specifying how step 3 of figure 1 should be run, i.e., simulation parameters (`case-0.indep-0.run_md.in`), initial system configurations for simulation (`case-0.indep-0.input.xyz`), and hyperparameters for simulation output post-processing (`bonds.dat` VERIFY RUN MOLANAL IS NEEDED … CAN MOVE IT INTO AL DRIVER FILES).
- The `ALL_BASE_FILES/QM_BASEFILES` directory contains files specifying how step 6 of figure 1 should be run, i.e., quantum calculation instructions (`6000.INCAR`), psuedopotential files (`C.POTCAR`), and a K-point file (`KPOINTS`) file.
- the `config.py` provides high-level instructions on how *all* steps in fig. 1 should be run.

A detailed description of the files in `ALL_BASE_FILES/ALC-0_BASEFILES` and `ALL_BASE_FILES/CHIMESMD_BASEFILES` can be found in the (ChIMES LSQ manual).

Tip: In `fm_setup.in`, 3-and-greater polnomial orders are given as n+1. In the following example, a 3-body order of 4 is desired, hence a value of n+1 = 5 is given in the example `fm_setup.in`.

Contents of the `config.py` file must be modified to reflect your e-mail address and absolute paths prior to running this example, i.e. on the lines highlighed below:

```
##################################
##### General options
##################################

EMAIL_ADD = "lindsey11@llnl.gov"

ATOM_TYPES     = ["C"]
NO_CASES       = 1

DRIVER_DIR     = "/p/lustre2/rlindsey/al_driver/"
WORKING_DIR    = "/p/lustre2/rlindsey/al_driver/examples/simple_iter_single_statepoint"
CHIMES_SRCDIR  = "/p/lustre2/rlindsey/chimes_lsq/src/"

##################################
##### ChIMES LSQ
##################################

ALC0_FILES    = WORKING_DIR + "ALL_BASE_FILES/ALC-0_BASEFILES/"
CHIMES_LSQ    = CHIMES_SRCDIR + "../build/chimes_lsq"
CHIMES_SOLVER = CHIMES_SRCDIR + "../build/chimes_lsq.py"
CHIMES_POSTPRC= CHIMES_SRCDIR + "../build/post_proc_chimes_lsq.py"

# Generic weight settings

WEIGHTS_FORCE =   1.0

REGRESS_ALG   = "dlasso"
REGRESS_VAR   = "1.0E-5"
REGRESS_NRM   = True

# Job submitting settings (avoid defaults because they will lead to long queue times)

CHIMES_BUILD_NODES = 2
CHIMES_BUILD_QUEUE = "pdebug"
CHIMES_BUILD_TIME  = "01:00:00"

CHIMES_SOLVE_NODES = 2
CHIMES_SOLVE_QUEUE = "pdebug"
CHIMES_SOLVE_TIME  = "01:00:00"

##################################
##### Molecular Dynamics
##################################

MD_STYLE        = "CHIMES"
CHIMES_MD_MPI   = CHIMES_SRCDIR + "../build/chimes_md"

MOLANAL         = CHIMES_SRCDIR + "../contrib/molanal/src/"
MOLANAL_SPECIES = ["C1"]

```

(continues on next page)

(continued from previous page)

```
################################
##### Single-Point QM
################################

QM_FILES = WORKING_DIR + "ALL_BASE_FILES/QM_BASEFILES"
VASP_EXE = "/usr/gapps/emc-vasp/vasp.5.4.4/build/gam/vasp"
```

Running

Depending on standard queuing times for your system, the ALD could take quite some time (e.g., hours) finish. For this reason it is generally, it is recommended to run the ALD from within a screen session on your HPC system. To do so, log into your HPC system and execute the following commands:

```
$: cd /path/to/my/example/files
$: screen
$: unbuffer python3 /path/to/your/ald/installation/main.py 0 1 2 3 | tee driver-0.log
```

Note that in the final line above, the sequence of numbers indicates 3 active learning cycles will be run (i.e., the `0` is ignored but required when simple iterative refinement mode is selected), and `| tee driver.log` sends all output to both the screen and a file named driver.log.

Tip: To detach from the screen session, execute `ctrl a` followed by `ctrl d`. You can now log out of the HPC system without dirupting the ALD. Be sure to take note of which node you were logged into. You can reattach to the session later by logging into the same node and executing `screen -r`

Inspecting the output

Once the ALD has finished running, execute the following commands:

```
$: cd /path/to/examples/simple_iter_single_statepoint/
$: for i in {1..3}; do cd ALC-${i}/GEN_FF; paste b_comb.txt force.txt > compare.txt; cd -
↪; done
```

Then, plot `ALC-{3,2,1}/GEN_FF/compare.txt` with your favorite plotting software. The resulting figure should look like the following:

This force parity plot provides DFT-assigned per-atom forces on the x-axis, and corresponding ChIMES predicted forces on the y-axis, in kcal/mol/Angstrom. The ALC-1 data corresponds to data generated by DFT (i.e., the forces contained in `liquid_6000K_2.0gcc.xyzf`); the ALC-2 data contain everything from ALC-1, as well as forces for the ChIMES-generated configurations selected in step 5 of figure 1, which were assigned DFT forces in step 6 of figure 1. The ALC-3 data is structured similarly.

Next, plot the `ALC-{1..3}/CASE-0_INDEP_0/md_statistics.out` files. The resulting figure should look like the following:

This figure shows how the conserved quantity varies during ChIMES-MD NVT simulations using the models generated at each ALC. As expected due to the minimal initial training set, dynamics with the ALC-1 model are very unstable (i.e., varying by 55 kcal/mol/atom over 60 ps). Stability is signficantly improved by ALC-2, with the conserved quantity varying by only ~2 kcal/mol/atom. By ALC-3, the model is fully stable, varying by less than .01 kca/mol/atom over the 60 ps trajectory).

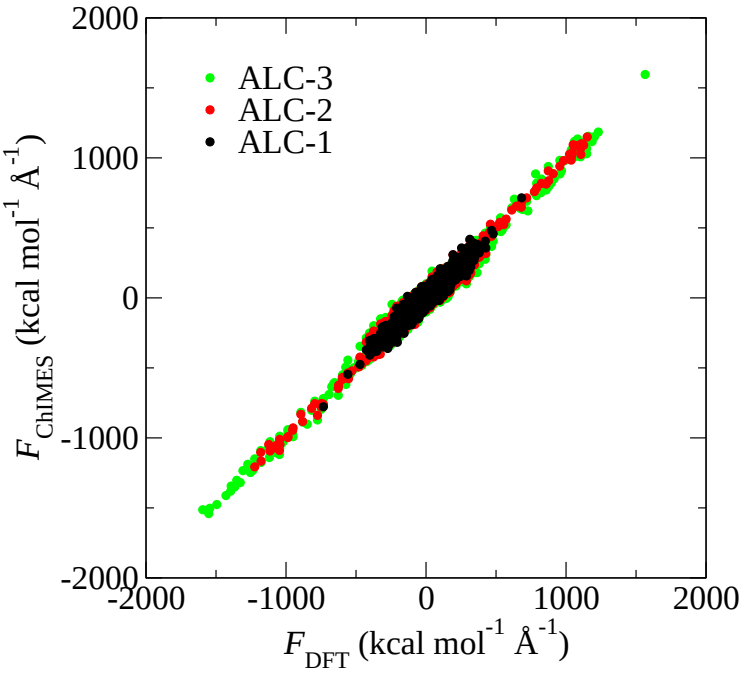

Fig. 3: **Fig. 2:** ALD fitting force pairty plot.

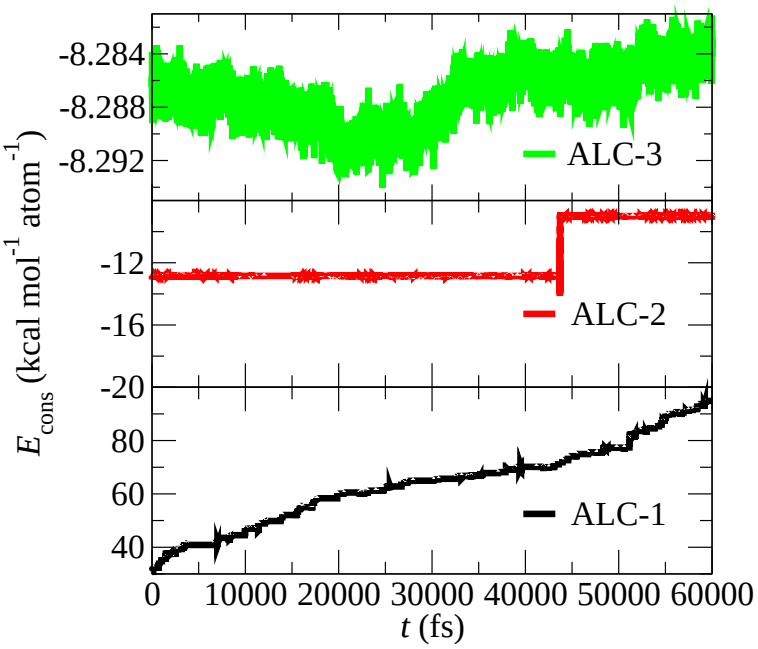

Fig. 4: **Fig. 2:** Conserved quantity for ChIMES moleuclar dynamics during ALD iterations.

1.1.2 In-depth Setup and Options Overview

Setting up Steps 1 & 2

As with a standard ChIMES fit (see e.g, the (ChIMES LSQ manual)), model generation must begin with selecting an intial training set and specifying fitting hyperparameters. In the ALD, this involves the following files, at a minimum:

```
<my_fit>/ALL_BASE_FILES/ALC-0_BASEFILES/fm_setup.in
<my_fit>/ALL_BASE_FILES/ALC-0_BASEFILES/traj_list.dat
<my_fit>/ALL_BASE_FILES/ALC-0_BASEFILES/*xyzf
<my_fit>/config.py
```

The `fm_setup.in` file is created as usual, except:

- The `# TRJFILE #` option must be set to `MULTI traj_list.dat`
- The `# SPLITFI #` option must be set to `false`.

See the `(ChIMES LSQ manual) <https://chimes-lsq.readthedocs.io/en/latest/index.html>`_for more information on what these settings control.

Warning: Arbitrary specification of fit hyperparameters (i.e., set in `fm_setup.in`) **will** result in inaccurate and/or unstable models. For more information on ChIMES hyperparameters and selection strategies, see:

- The ChIMES LSQ code manual (link)
- R.K. Lindsey, L.E. Fried, N. Goldman, *JCTC*, **13**, 6222 (2017) (link)
- R.K. Lindsey, L.E. Fried, N. Goldman, *JCTC* **15** 436 (2019) (link)

The `traj_list.dat` file should be structured as usual for ChIMES LSQ, but lines containing the first n-cases entries should have a temperature in Kelvin specified at the very end, where n-cases is the number of statepoints the user would like to simultaneously conduct iterative training to:

```
3
10 1000K_1.0gcc.xyzf 1000
10 2000K_2.0gcc.xyzf 2000
10 3000K_3.5gcc.xyzf 3000
```

Note that the above .xyzf files correspond to `<my_fit>/ALL_BASE_FILES/ALC-0_BASEFILES/*xyzf`.

Finally, options for this first phase of fitting `config.py` must be specified. <PAGE> provides a complete set of options and details default values. Note that for this basic overview we will assume:

- The user is running on a SLURM/SBATCH based HPC system (**set by default**)
- The HPC system has 36 processors per compute node (**set by default**)
- We want to generate hydrogen parameters by iteratively fitting at 3 statepoints, simultaneously (**indicated by line 6**).

The minimal config.py lines necessary for steps 1 & 2 are provided in the code block below. Recalling that ALD functions primarily as a workflow tool, it must be linked with external software. Here, we tell the ALD:

- Where the ALD source code is located (line 8),
- Where the ALD will be run (line 9), and
- Where to find our ChIMES_LSQ installation (line 10).

Lines 16-19 tell the ALD where all the files needed to run chimes_lsq are, specifically:

- The ChIMES LSQ input files, fm_setup.in and traj_list.dat (line 16),
- The ChIMES LSQ design matrix generation executable, chimes_lsq (line 17),
- The ChIMES LSQ matrix solution script, chimes_lsq.py (line 18), and
- The ChIMES LSQ parameter file scrubber, post_proc_chimes_lsq.py (line 19).

Finally, lines 23-25 specify how forces, energies, and stresses should be weighted, while lines 27-29 specify how the matrix solution problem should be executed, i.e., using distributed lasso (line 27) with a regularization variable of 1e-8 (line 28), and with a normalized design matrix (line 29). Note that there are *many* options for these lines, described in detail in <PAGE>.

```
1  ################################
2  ##### General options
3  ################################
4
5  ATOM_TYPES     = ["H"]
6  NO_CASES       = 3
7
8  DRIVER_DIR     = "/path/to/active_learning_driver/src"
9  WORKING_DIR    = "/path/to/directory/where/learning/will/occur"
10 CHIMES_SRCDIR  = "/path/to/chimes_lsq/installation/src"
11
12 ################################
13 ##### ChIMES LSQ
14 ################################
15
16 ALC0_FILES    = WORKING_DIR + "ALL_BASE_FILES/ALC-0_BASEFILES/"
17 CHIMES_LSQ    = CHIMES_SRCDIR + "chimes_lsq"
18 CHIMES_SOLVER = CHIMES_SRCDIR + "lsq2.py"
19 CHIMES_POSTPRC= CHIMES_SRCDIR + "post_proc_chimes_lsq.py"
20
21 # Generic weight settings
22
23 WEIGHTS_FORCE =   1.0
24 WEIGHTS_ENER  =   0.1
25 WEIGHTS_STRES = 100.0
26
27 REGRESS_ALG   = "dlasso"
28 REGRESS_VAR   = "1.0E-8"
29 REGRESS_NRM   = True
```

Setting up Step 3

Step 3 comprises molecular dynamics (MD) simulation with the parameters generated in step 2. Beyond the parameter file, this requires the following at a minimum:

- An initial coordinate file,
- A MD input file specifying the simulation style,
- A MD code executable, and
- Instructions on how to post-process resultant trajectories

Recalling that the current example concerns concurrent iterative fitting for three cases (training state points), this is specified by the following in `/path/to/ALL_BASE_FILES/CHIMESMD_BASEFILES/` and `config.py`, i.e.:

```
$: ls /path/to//ALL_BASE_FILES/CHIMESMD_BASEFILES/
<my_fit>/ALL_BASE_FILES/CHIMESMD_BASEFILES/case-0.indep-0.input.xyz
<my_fit>/ALL_BASE_FILES/CHIMESMD_BASEFILES/case-0.indep-0.run_md.in
<my_fit>/ALL_BASE_FILES/CHIMESMD_BASEFILES/case-1.indep-0.input.xyz
<my_fit>/ALL_BASE_FILES/CHIMESMD_BASEFILES/case-1.indep-0.run_md.in
<my_fit>/ALL_BASE_FILES/CHIMESMD_BASEFILES/case-2.indep-0.input.xyz
<my_fit>/ALL_BASE_FILES/CHIMESMD_BASEFILES/case-2.indep-0.run_md.in
<my_fit>/ALL_BASE_FILES/CHIMESMD_BASEFILES/bonds.dat
```

and

```
1 ################################
2 ##### Molecular Dynamics
3 ################################
4
5 MD_STYLE        = "CHIMES"
6 CHIMES_MD_MPI   = CHIMES_SRCDIR + "chimes_md-mpi"
7
8 MOLANAL         = "/path/to/molanlal/folder/"
9 MOLANAL_SPECIES = ["H1", "H2 1(H-H)", "H3 2(H-H)"]
```

Each `case-*.indep-0.input.xyz` is a ChIMES `.xyz` file containing initial coordinates for the system of interest for the corresponding case, while each `case-*.indep-0.run_md.in` is the corresponding ChIMES MD input file. Note that `case-*.indep-0.run_md.in` options `# PRMFILE #` and `# CRDFILE #` should be set to `WILL_AUTO_UPDATE`. For more information on these files, see the (ChIMES LSQ manual). The bonds.dat file will be described below.

In the config.py file snipped above, lines 5 and 6 tell the ALD to use ChIMES MD for MD simulation runs, and provides a path to the MPI-enabled and serial compilations. Lines 9 and 10 provide information on how to post-process the trajectory. Specifically, the ALD will use the a molecular analyzer ("molanal") <https://pubs.acs.org/doi/pdf/10.1021/ja808196e>`_ to determine speciation for the generated MD trajectories. Once speciation is determined, the ALD will provide a summary of lifetimes and molefractions for species listed in ``MOLANAL_SPECIES`. Note that the species names must match the "Molecule type" fields produced by molanal *exactly*. These strings are usually determined by running molanal on DFT-MD trajectories, prior to any ALD. Finally, the `bonds.dat` file specifies bond length and lifetime criteria for molanal. See the molanal `readme.txt` file for additional information. Be sure to verify specified bonds.dat lifetime criteria are consistent with the timestep and output frequency specified in `case-*.indep-0.run_md.in`

Setting up Step 4

Model validation is purposefully left to the user, as optimal strategies are still an active area of research and are most efficient when application-specific. The user is encouraged to investigate fit performance and physical property recovery on their own.

Setting up Step 5

Candidate configuration filtering is conducted in step 5. For basic fitting mode, this simply comprises selecting a subset of configurations generated during the previous MD step for single point evaluation using, e.g., DFT. This is handled entirely automatically by the ALD.

For basic iterative refinement mode, this entails selecting up to 20 evenly spaced configurations from ChIMES-MD simulations at each case, for which all atoms are:

1. Outside the penalty function kick-in region
2. Within the penalty function kick-in region but outside the inner cutoffs

The latter configurations are included to inform the short-ranged region of the interaction potential, which is generally poorly sampled by DFT-MD.

Setting up Step 6

Step 6 comprises single point evaluation of configurations selected in step 5 via the user's requested quantum-based reference method. In this overview, we will assume the user is employing VASP but additional options are described in **`options`_**. To do so, the following must be provided, at a minimum:

```
<my_fit>/ALL_BASE_FILES/QM_BASEFILES/*.INCAR
<my_fit>/ALL_BASE_FILES/QM_BASEFILES/KPOINTS
<my_fit>/ALL_BASE_FILES/QM_BASEFILES/*.POTCAR
```

and

```
################################
##### Single-Point QM
################################

QM_FILES = WORKING_DIR + "ALL_BASE_FILES/QM_BASEFILES"
VASP_EXE = "/path/to/vasp/executable"
```

There should be one `*.INCAR` file for each case temperature, i.e. `{1000,2000,3000}.INCAR` for the present example, with all options set to user desired values for single point evaluation. Note that `IALGO = 48` should be used to specifiy the electronic minimisation algorithm, and any variable related to restart should be set to the corresponding "new" value. There should also be one `.*POTCAR` file for each atom type considered, i.e. H.POTCAR for the present example.

Note: Support for additional data labeling schemes (i.e., both quantum- and moleuclar mechanics-based) are incoming.

Warning: QM codes can fail to converge in unexpected cases, in manners that are challenging to detect. If you notice your force pairity plots indicate generally good model performance but show a few unexpected outliers, verify your QM code is providing the correct answer. This can be done by evaluating the offending configurationw with a different code version or a different code altogether.

1.2 Hierarchical Fitting Mode

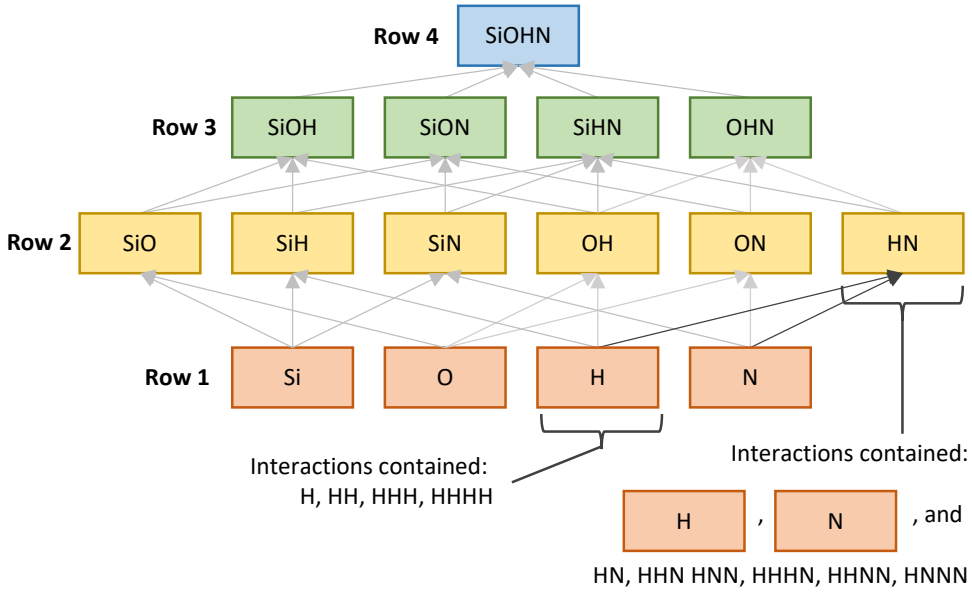

Fig. 5: **Fig. 1:** The ChIMES parameter hierarchy for a Si, O, H, and N-containing system.

The most common strategy for generating machine-learned interatomic models involves fitting all parameters at once but this can (1) prove challenging for high complexity problems and (2) limit transferability. However, the inhently hierarchical nature of ChIMES parameters allows for an alternative strategy in which relatively low-complexity "families" of parameters can be generated independently from one another. These families can then be combined and built upon via transfer learning to describe higher complexity systems. For example, Fig. 1 shows the CHIMES parameter hierarchy for an up-to-4-body model describing interactions in a Si, O, H, and N-containing system. Each "tile" represents a family of parametrs, e.g., the H tile contains the 1-through 4-body parameters for H, H-H, H-H-H, and H-H-H-H interactions. Tiles on the same row (e.g. H and N) can be fit indepent of one another; tiles containing two or more atoms describe *only* simultaneous cross iteractions between the indicated atom types, e.g., the HN tile *only* contains parameters for HN, HHN, HNN, HHHN, HHNN, and HNNN interactions. Practically, this means simulating an H- and N- containing system requires all the parameters contained in the H, N, and HN tiles.

Fitting row-1 tiles requires no special treatment. However, fitting tiles on row-2 and above requires pre-processing training data during each learning iteration to remove contributions from the relavant lower row tiles. For example, an HN tile fit would require H and N tile contributions to be removed from the training data. Additionally, parameter sets must be combined into a cohesive file before running dynamics. *The ALD can perform these tasks automatically.*

This section provides an overview of how to configure the ALD for a hierarchical fitting strategy, within the context of a liquid C/N system. Before proceding, ensure you have read through and fully understand the *ChIMES Active Learning Driver Configuration File Options*.

For additional information on strategies and benefits of hierarchical fitting, see:

- R.K. Lindsey, B. Steele, S. Bastea, I.-F. Kuo, L.E. Fried, and N. Goldman *In Prep*

1.2.1 Example Fit: Solid C/N

Note: Files for this example are located in `./<al_driver base folder>/examples/hierarch_fit`

In this section, an example 3-iteration fit for a solid C- and N-containing system at ~75% C, 6000K, and 3.5 g/cc is overviewed **<DOUBLE CHECK THESE NUMBERS>**. The model will include up-to-4 body interactions. Given the substantial increase in number of fitting parameters and system complexity relative to pure carbon case the basic fitting example, this case will take substantially longer to run.

The neccesary input files and directory tree structure are provided in the example folder, i.e.:

```
$: tree
.
├── ALC-0_BASEFILES
│   ├── 20.3percN_3.5gcc.temps
│   ├── 20.3percN_3.5gcc.xyzf
│   ├── fm_setup.in
│   └── traj_list.dat
├── CHIMESMD_BASEFILES
│   ├── base.run_md.in
│   ├── bonds.dat
│   ├── case-0.indep-0.input.xyz
│   ├── case-0.indep-0.run_md.in
│   └── run_molanal.sh
├── HIERARCH_PARAMS
│   ├── C.params.txt.reduced
│   └── N.params.txt.reduced
└── QM_BASEFILES
    ├── 6000.INCAR
    ├── C.POTCAR
    ├── KPOINTS
    ├── N.POTCAR
    └── POTCAR
```

Comparing with the `ALC-0_BASEFILES` folder provided in the *ChIMES Active Learning Driver Configuration File Options*, the primary difference is the `HIERARCH_PARAMS` directory, i.e., which contains parameters for the C and N tiles, and the `.temps` file, which provides a single temperature for each frame in the corresponding `.xyzf` file, are highlighted.

Input Files

The ALC-0_BASEFILES Files

Warning: The `ALC-0_BASEFILES/fm_setup.in` requires a few special edits for hierarchical learning mode:

- `fm_setup.in` should have `# HIERARC #` set `true`
- All 1- through *n*-body interactions described in in the reference (`HIERARCH_PARAM_FILES`) files must be explicitly excluded
- Orders in the `ALC-0_BASEFILES/fm_setup.in` file should be greater or equal to those in the reference (`HIERARCH_PARAM_FILES`) files
- `TYPEIDX` and `PAIRIDX` entries in the base fm_setup.in file must be consistent with respect to the `HIERARCH_PARAM_FILES` files
- `SPECIAL XB` cutoffs must be set to `SPECIFIC N`, where *N* is the number of **NON**-excluded *X*B interaction types

For additional information on how to configure these options, see the ChIMES LSQ manual **`(link <UPDATE LINK>)`_**.

The config.py File

The *config.py* file is given below:

```
################################
##### General variables
################################

EMAIL_ADD     = "lindsey11@llnl.gov" # driver will send updates on the status of the current run ... If blank (""), no emails are sent

ATOM_TYPES = ['C', 'N']
NO_CASES = 1

DRIVER_DIR     = "/p/lustre2/rlindsey/al_driver/src/"
WORKING_DIR    = "/p/lustre2/rlindsey/al_driver/examples/hierarch_fit"
CHIMES_SRCDIR  = "/p/lustre2/rlindsey/chimes_lsq/src/"

################################
##### ChIMES LSQ
################################

ALC0_FILES    = WORKING_DIR + "ALL_BASE_FILES/ALC-0_BASEFILES/"
CHIMES_LSQ    = CHIMES_SRCDIR + "../build/chimes_lsq"
CHIMES_SOLVER = CHIMES_SRCDIR + "../build/chimes_lsq.py"
CHIMES_POSTPRC= CHIMES_SRCDIR + "../build/post_proc_chimes_lsq.py"

# Generic weight settings

WEIGHTS_FORCE =   1.0
```

(continues on next page)

(continued from previous page)

```
26
27 REGRESS_ALG   = "dlasso"
28 REGRESS_VAR   = "1.0E-5"
29 REGRESS_NRM   = True
30
31 # Job submitting settings (avoid defaults because they will lead to long queue times)
32
33 CHIMES_BUILD_NODES = 2
34 CHIMES_BUILD_QUEUE = "pdebug"
35 CHIMES_BUILD_TIME  = "01:00:00"
36
37 CHIMES_SOLVE_NODES = 2
38 CHIMES_SOLVE_QUEUE = "pdebug"
39 CHIMES_SOLVE_TIME  = "01:00:00"
40
41 ################################
42 ##### Molecular Dynamics
43 ################################
44
45 MD_STYLE        = "CHIMES"
46 CHIMES_MD_MPI   = CHIMES_SRCDIR + "../build/chimes_md"
47
48 MOLANAL         = CHIMES_SRCDIR + "../contrib/molanal/src/"
49 MOLANAL_SPECIES = ["C1", "N1"]
50
51 ################################
52 ##### Hierarchical fitting block
53 ################################
54
55 DO_HIERARCH = True
56 HIERARCH_PARAM_FILES = ['C.params.txt.reduced', 'N.params.txt.reduced']
57 HIERARCH_EXE = CHIMES_MD_SER
58
59 ################################
60 ##### Single-Point QM
61 ################################
62
63 QM_FILES = WORKING_DIR + "ALL_BASE_FILES/QM_BASEFILES"
64 VASP_EXE = "/usr/gapps/emc-vasp/vasp.5.4.4/build/gam/vasp"
```

The primary difference between the present `config.py` and that provided in the file *ChIMES Active Learning Driver Configuration File Options* documentation are the highlighted lines 55–57, which specify hierarchical fitting should be performed (line 55), the name of all parameter files that the present model should be built upon (line 56), and the executable to use when evaluating contributions from the parameter files specified on line 56 (line 57); for this example, we're using ChIMES_MD. Note that this executable should be compiled for serial runs to prevent issues with the queueing system. As in the example provided in *ChIMES Active Learning Driver Configuration File Options* documentation, contents of the `config.py` file must be modified to reflect your e-mail address and absolute paths prior to running this example.

Running

Inspecting the output

1.2.2 In-depth Setup and Options Overview

For detailed instructions on setting up and running the ALD, see the *ChIMES Active Learning Driver Configuration File Options*

CHAPTER

TWO

QUICK START

The ALD is a workflow tool that autonomously generates ChIMES models by orchestrating, running, and monitoring the various different tasks involved in iterative learning a model. By necessity, this involves generating input for, using, and post-processing output from several codes, download and installation of which are described in the following sections.

Note: System requirements for the ALD include:

- An HPC platform with job queueing - currently only SLURM/SBATCH systems are supported
- C, C++11, and Fortran 77, 90, and 08 compilers
- MPI compilers
- MKL
- Python version 3

Note that the ALD is trivially extendable to other queuing sytems for all running modes except cluster-based active learning, and can be run without cluster-based active learning support. See the "**<EXTENDING>**" page for additional details.

2.1 Installing ChIMES LSQ and ChIMES MD

The ALD requires a specific version of the ChIMES LSQ/MD code. To download and compile it, log into your HPC system, execute the following commands, and agree to all prompted questions:

```
cd /path/to/my/software/folder
mkdir chimes_lsq-forALD
git clone https://github.com/rk-lindsey/chimes_lsq.git chimes_lsq-forALD
cd chimes_lsq-forALD
./install.sh
```

Warning: If you are note running on an LLNL (Quartz) or UM (Great Lakes) system, you will need to manually configure your compilers. We recommend Intel OneAPI, which is freely available. You will need to compile dlars and molanal by hand (see install script for steps).

If the above instructions are followed properly, the following executables/scripts should be generated:

```
./build/chimes_lsq
./build/chimes_md-serial
./build/chimes_md-mpi
./build/chimes_lsq.py
./build/post_proc_chimes_lsq.py
./contrib/dlars/src/dlars
./contrib/molanal/src/molanal.new
```

2.2 Installing Reference (Data Labeling) Methods

The ALD currently supports VASP and DFTB+ for data labeling (i.e. providing forces, energies, and stresses for configurations) in periodic system and Gaussian for non-periodic systems. Current implmentations are configured for the following software versions:

- VASP 5.4.1 (link)
- Gaussian 16 (link)
- DFTB+ 17.1 (link)

Support for newer VASP and DFTB+ versions is in progress. Future efforts will also focus on supporting LAMMPS as a data labeling method, allowing, e.g., coarse-grained model development based on molecular mechanics potentials.

2.3 Note on Correction Support

The ALD currently supports generating ChIMES corrections for DFTB via DFTB+, however it requires an in-house compilation. Support via DFTB+/the ChIMES calculator is under development.

CHAPTER

THREE

CHIMES ACTIVE LEARNING DRIVER CONFIGURATION FILE OPTIONS

3.1 Optional config.py Variables:

3.1.1 Assorted General Options

Input variable	Variable type	Required	Default	Value/Options/Notes
EMAIL_ADD =	str	N	""	E-mail address for driver to sent status updates to. If blank (""), no emails are sent.
SEED =	int	N	1	Only used for active learning strategies are selected. Seed for random number generator.
ATOM_TYPES =	list of str	Y	None	List of atom types in system of interest, e.g. ["C","H","O"].
NO_CASES =	int	Y	None	Number of different state points at which to conduct iterative learning.
MOLANAL_SPECIES =	list of str	Y	[]	List of species to track in molanal output, e.g. ["C1 O1 1(O-C)", "C1 O2 2(O-C)"].
USE_AL_STRS =	int	N	0	Cycle at which to start including stress tensors from ALC generated configrations.
STRS_STYLE =	str	N	"ALL"	How stress tensors should be included in the fit. Options are: "DIAG" or "ALL".
THIS_SMEAR =	int	N	float	Thermal smearing temperature in K; if "None", different values are used for each case, set in the ALL_BASE_FILES traj_list.dat.

3.1.2 General HPC Options

Input variable	Variable type	Required	Default	Value/Options/Notes
HPC_PPN =	int	N	36	Number of processors per node on HPC platform.
HPC_ACCOUNT =	str	N	pbronze	Charge bank/account name on HPC platform.
HPC_SYSTEM =	str	N	slurm	HPC platform type (Only "slurm" supported currently).
HPC_PYTHON =	str	N	/usr/tce/bin/python	Full path to python2.X exectuable on HPC platform.
HPC_EMAIL =	bool	N	True	Controls whether driver status updates are e-mailed to user.

3.1.3 ChIMES LSQ Options

Input variable	Variable type	Required	Default	Value/Options/Notes
ALC0_FILES =	str	N	WORKING_DIR + "ALL_BASE_FILES/ALC-0_BASEFILES/"	Path to base files required by the driver (e.g. ChIMES input files, VASP, input files, etc.)
CHIMES_LSQ =	str	N	CHIMES_SRCDIR + "chimes_lsq"	Absolute path to ChIMES_lsq executable.
CHIMES_SOLVER =	str	N	CHIMES_SRCDIR + "lsq2.py"	Absolute path to ChIMES_lsq.py (formely, lsq2.py).
CHIMES_POSTPRC =	str	N	CHIMES_SRCDIR + "post_proc_lsq2.py"	Absolute path to post_proc_lsq2.py.
WEIGHTS_SET_ALC0 =	bool	N	False	Should ALC-0 (or 1 if no clustering) weights be read directly from a user specified file?
WEIGHTS_ALC_0 =	str	N	None	Set if WEIGHTS_SET_ALC_0 is true; path to user specified ALC-0 (or ALC-1) weights.
WEIGHTS_FORCE =	special	N	1.0	Weights to apply to full-frame forces - many options, see note below.
WEIGHTS_FGAS =	special	N	5.0	Weights to apply to gas phase forces - many options, see note below.
WEIGHTS_ENER =	special	N	0.1	Weights to apply to full-frame energies - many options, see note below.
WEIGHTS_EGAS =	special	N	0.1	Weights to apply to gas phase energies - many options, see note below.
WEIGHTS_STRES =	special	N	250.0	Weights to apply to full-frame stress tensor components - many options, see note below.
REGRESS_ALG =	str	N	dlasso	Regression algorithm to use for fitting; only dlasso supported for now
REGRESS_VAR =	float	N	1e-5	Regression regularization variable.
REGRESS_NRM =	bool	N	True	Controls whether A-matrix is normalized prior to solution.
CHIMES_BUILD_NODES =	int	N	4	Number of nodes to use when running chimes_lsq.
CHIMES_BUILD_QUEUE =	str	N	pbatch	Queue to submit chimes_lsq job to.
CHIMES_BUILD_TIME =	str	N	"04:00:00"	Walltime for chimes_lsq job.
CHIMES_SOLVE_NODES =	int	N	8	Number of nodes to use when running dlasso
CHIMES_SOLVE_PPN =	int	N	HPC_PPN	Number of procs per node to use when running dlasso
CHIMES_SOLVE_QUEUE =	str	N	pbatch	Queue to submit the dlasso job to
CHIMES_SOLVE_TIME =	str	N	"04:00:00"	Walltime for dlasso job

Note: There are numerous options available for weighting, and weights are applied separately to full-frame forces, gas phase forces, full-frame energies, gas phase energies, and full-frame stress.

If a WEIGHTS_* option is set to a single floating point value, that value is applied to all candidate data of that type, e.g., if WEIGHTS_FORCE = 1.0, all full-frame forces will be assigned a weight of 1.0.

Additional weighting styles can be selected by letter:

A. w = a0

B. w = a0*(this_cycle-1)^a1 # NOTE: treats this_cycle = 0 as this_cycle = 1

C. w = a0*exp(a1*|X|/a2)

D. w = a0*exp(a1[X-a2]/a3)

E. w = n_atoms^a0

where "X" is the value being weighted.

WEIGHTS_FORCE = [["B"],[1.0,-1.0]] would select weighting style B and apply a weight of 1.0 to each full-frame force component in the first ALD cycle; weighting would decrease by a factor (this_cycle)^(-1.0) each cycle.

Multiple weighting schemes can be combined as well. For example WEIGHTS_FORCE = [["A","B"], [[100.0],[1.0,-1.0]]] would add an additional multiplicative factor of 100 to the previous example.

3.1.4 Molecular Dynamics Options

Input variable	Variable type	Required	Default	Value/Options/Notes
MD_STYLE =	str	Y	None	Iterative MD method. Options are "CHIMES" (used for ChIMES model development) or "DFTB" (used when generating ChIMES corrections to DFTB).
DFTB_MD_SER =	str	N	None	Only used when MD_STYLE set to "DFTB". DFTBplus executable absolute path.
CHIMES_MD_MPI =	str	N	CHIMES_SRCDIR + "chimes_md-mpi"	Only used when MD_STYLE set to "CHIMES". MPI-compatible ChIMES_md exectuable absolute path.
CHIMES_MD_SER =	str	N	CHIMES_SRCDIR + "chimes_md-serial"	Used when MD_STYLE set to either "CHIMES" or "DFTB*". Serial ChIMES_md executable absolute path.
MD_NODES =	list of int	N	[4] * NO_CASES	Number of nodes to use for MD jobs at each case. Number can be different for each case (e.g., [2,2,4,8] for four cases).
MD_QUEUE =	list of str	N	["pbatch"] * NO_CASES	Queue type to use for MD jobs at each case. Can be different for each case.
MD_TIME =	list of str	N	["4:00:00"] * NO_CASES	Walltime to use for MD jobs at each case. Can be different for each case.
MDFILES =	str	N	WORKING_DIR + "ALL_BASE_FILES/CHIMES_MD_BASEFILES/"	Absolute path to MD input files like case-0.indep.
CHIMES_PENALTY_PREFAC =	float	N	1.0E6	ChIMES penalty function prefactor.
CHIMES_PENALTY_DIST =	float	N	0.02	ChIMES pentalty function kick-in distance
MOLANAL =	str	N	None	Absolute path to molanal executable.

- CHIMES_MD_SER is used for old i/o based ChIMES/DFTB linking - update required, but needs bad_cfg printing in DFTB+ (requires change to interface)

3.1.5 Correction Fitting Options

Input variable	Variable type	Required	Default	Value/Options/Notes
FIT_CORRECTION =	bool	N	False	Is this ChIMES model being fit as a correction to another method?
CORRECTED_TYPE =	str	N	None	Method type being corrected. Currently only "DFTB" is supported
CORRECTED_TYPE_FILES =	list of str	N	None	?!?!?!?!IS THIS A PATH OR A FILENAME? Files needed to run simulations/single points with the method to be corrected
CORRECTED_TYPE_EXE =	str	N	None	Executable to use when subtracting existing forces/energies/stresses from method to be corrected
CORRECTED_TEMPS_BY_FILE =	bool	N	None	????!?! String or path??? Should electron temperatures be set to values in traj_list.dat (false) or in specified file location, for correction calculation? Only needed if correction method is QM-based.

Note: Note: If corrections are used, `ChIMES_MD_{NODES,QUEUE,TIME}` are all used to specify DFTB runs. These should be renamed to `simulation_{...}` for the generalized MD block (which should become SIM block). If `FIT_CORRECTION` is `True`, temperaturess in `traj_list.dat` are ignored by correction FES subtraction. Instead, searches for `<filesnames>.temps` where `.temps` replaces whatever last extension was, in `CORRECTED_TYPE_FILES`.

3.1.6 Hierarchical Fitting Options

Input variable	Variable type	Default	Value/Options/Notes
DO_HIERARCH =	bool	False	Is this a hierarchical fit (i.e., building on existing parameters?")
HIERARCH_PARAM_FILES =	list of str	None	List of parameter files to build on, which should be in ALL_BASE_FILES/HIERARCH_PARAMS
HIERARCH_EXE =	str	None	Executable to use when subtracting existing parameter contributions

Note: Consider the case of fitting 2+3+4-body C/N parameters on top of existing C- and N- parameter sets.

Users must create a new folder, `HIERARCH_PARAMS` in their `ALL_BASE_FILES` directory and place in it the pure-C and pure-N parameter files, i.e.:

```
$: ls -l <my_fit>/ALL_BASE_FILES/HIERARCH_PARAMS
-rw------- 1 rlindsey rlindsey 169630 May  1 10:55 C.params.txt.reduced
-rw------- 1 rlindsey rlindsey 160015 May  1 10:55 N.params.txt.reduced
```

Hierarchical fitting also requires special options in `ALL_BASE_FILES/ALC-0_BASEFILES/fm_setup.in` to ensure base the parameter types (e.g., in {C,N}.params.txt.reduced) are properly excluded from the fit. First, one must ensure that requested polynomial orders are greater or equal to those in the reference `ALL_BASE_FILES/HIERARCH_PARAMS` parameter files. Next, add the highlighted lines to `fm_setup.in`:

```
    # Snippet from ALL_BASE_FILES/ALC-0_BASEFILES/fm_setup.in

    # PAIRTYP #
```

(continues on next page)

(continued from previous page)

```
            CHEBYSHEV 25 10 4 -1 1
    # CHBTYPE #
            MORSE
    # SPLITFI #
            false
    # HIERARC #
            true
```

Users must also specify which interactions to exclude from the fit (i.e., interactions fully described by the ALL_BASE_FILES/HIERARCH_PARAMS files. For the present C/N fitting example, those lines would look like:

```
####### TOPOLOGY VARIABLES #######

EXCLUDE 1B INTERACTION: 2
C
N

EXCLUDE 2B INTERACTION: 2
C C
N N

EXCLUDE 3B INTERACTION: 2
C C C
N N N

EXCLUDE 4B INTERACTION: 2
C C C C
N N N N
```

Users must also ensure that the `fm_setup.in` topolgy contents are consistent with those in the ALL_BASE_FILES/HIERARCH_PARAMS files. For the present C/N fitting example, those would be the highlighted lines below:

```
# NATMTYP #
        2

# TYPEIDX #     # ATM_TYP #     # ATMCHRG #     # ATMMASS #
1               C               0.0             12.0107
2               N               0.0             14.0067

# PAIRIDX #     # ATM_TY1 #     # ATM_TY1 #     # S_MINIM #     # S_MAXIM #     # S_
↪DELTA #     # MORSE_LAMBDA #        # USEOVRP #     # NIJBINS #     # NIKBINS #     #␣
↪NJKBINS #
1               C               C               0.98            5.0             0.01    ␣
↪       1.4                     false           0               0               0
2               N               N               0.86            8.0             0.01    ␣
↪       1.09                    false           0               0               0
3               C               N               1.0             5.0             0.01    ␣
↪       1.34                    false           0               0               0
```

Users must explicitly define how many (and which) many-body interactions will be fit, and the corresponding outer cutoffs to use. Note that the option `ALL` cannot be used when performing hierarchical fits.

```
SPECIAL 3B S_MAXIM: SPECIFIC 2
CCCNCN   CC CN CN 5.0 5.0 5.0
CNCNNN   CN CN NN 5.0 5.0 5.0

SPECIAL 4B S_MAXIM: SPECIFIC 3
CCCCCNCCCNCN    CC CC CN CC CN CN 4.5 4.5 4.5 4.5 4.5 4.5
CCCNCNCNCNNN    CC CN CN CN CN NN 4.5 4.5 4.5 4.5 4.5 4.5
CNCNCNNNNNNN    CN CN CN NN NN NN 4.5 4.5 4.5 4.5 4.5 4.5
```

Note: Each training trajectory file in ALL_BASE_FILES/ALC-0_BASEFILES needs a corresponding .temps file that gives the temperature for each frame **WHY?!?!?**.

TO DO ADD VASP MODULES TO CODE

3.1.7 Reference QM Method Options

Input variable	Variable type	Default	Value/Options/Notes
QM_FILES =	str	WORKING_DIR + "ALL_BASE_FILES/VASP_BASEFILES"	Absolute path to QM input files generic to all QM methods. Can specify separately if multiple methods are being used (see code-specific options below)
BULK_QM_METHOD =	str	VASP	Specifies which nominal QM code to use for bulk configurations; options are "VASP" or "DFTB+"
IGAS_QM_METHOD =	str	VASP	Specifies which nominal QM code to use for gas configurations; options are "VASP", "DFTB+", and "Gaussian"

VASP-Specific Options

Input variable	Variable type	Default	Value/Options/Notes
VASP_FILES =	str	QM_FILES	Absolute path to VASP input filess.
VASP_NODES =	int	6	Number of nodes to use for VASP jobs
VASP_PPN =	int	HPC_PPN	Number of processors to use per node for VASP jobs
VASP_TIME =	str	"04:00:00"	Walltime for VASP calculations (HH:MM:SS)
VASP_QUEUE =	str	"pbatch"	Queue to submit VASP jobs to
VASP_EXE =	str	None	A path to a VASP executable **must** be specified if BULK_QM_METHOD or IGAS_QM_METHOD are set to "VASP"
VASP_MODULES =	str	"mkl"	Modules to load during VASP run

DFTB+ -Specific Options

Input variable	Variable type	Default	Value/Options/Notes
DFTB_FILES =	str	QM_FILES	Absolute path to DFTB+ input files.
DFTB_NODES =	int	1	Number of nodes to use for VASP jobs
DFTB_PPN =	int	1	Number of processors to use per node for VASP jobs
DFTB_TIME =	str	"04:00:00"	Walltime for VASP calculations (HH:MM:SS)
DFTB_QUEUE =	str	"pbatch"	ueue to submit VASP jobs to
DFTB_EXE =	str	None	A path to a VASP executable **must** be specified if BULK_QM_METHOD or IGAS_QM_METHOD are set to "DFTB+"
DFTB_MODULES =	str	"mkl"	Modules to load during VASP run

Gaussian-Specific Options

Input variable	Variable type	Default	Value/Options/Notes
GAUS_NODES =	int	4	Number of nodes to use for Gaussian jobs
GAUS_PPN =	int	HPC_PPN	Number of processors to use per node for Gaussian jobs
GAUS_TIME =	str	"04:00:00"	Walltime for Gaussian calculations (HH:MM:SS)
GAUS_QUEUE =	str	"pbatch"	ueue to submit Gaussian jobs to
GAUS_EXE =	str	None	A path to a Gaussian executable **must** be specified if IGAS_QM_METHOD is set to "Gaussian"
GAUS_SCR =	str	None	Absolute path to Gaussian scratch directory
GAUS_REF =	str	None	Name of file containing single atom energies from Gaussian and target planewave method

Note: The file specified for GAUS_REF is structured like:

```
<chemical symbol> <Gaussian energy> <planewave code energy>
<chemical symbol> <Gaussian energy> <planewave code energy>
<chemical symbol> <Gaussian energy> <planewave code energy>
...
<chemical symbol> <Gaussian energy> <planewave code energy>
```

Energies are expected in kcal/mol and there should be an entry for each atom type of interest.

CHAPTER

FOUR

CITING THE ALD

Please cite the following publications:

Fitting Strategy	Citations
Basic	R.K. Lindsey, L.E. Fried, N. Goldman, *JCTC*, **13**, 6222 (2017)
Iterative	R.K. Lindsey, L.E. Fried, N. Goldman, *JCTC*, **13**, 6222 (2017) R.K. Lindsey, N. Goldman, L.E. Fried, S. Bastea, *JCP*, **153** 054103 (2020)
Hierarchical	R.K. Lindsey, L.E. Fried, N. Goldman, *JCTC*, **13**, 6222 (2017) R.K. Lindsey, N. Goldman, L.E. Fried, S. Bastea, *JCP*, **153** 054103 (2020)
Correction	R.K. Lindsey, L.E. Fried, N. Goldman, *JCTC*, **13**, 6222 (2017) N. Goldman, B. Aradi, R.K. Lindsey, L.E. Fried, *JCTC* **14** 2652 (2018) R.K. Lindsey, N. Goldman, L.E. Fried, S. Bastea, *JCP*, **153** 054103 (2020) R.K. Lindsey, S. Bastea, N. Goldman, L.E. Fried, *JCP*, **154** 164115 (2021)
Cluster AL	R.K. Lindsey, L.E. Fried, N. Goldman, *JCTC*, **13**, 6222 (2017) R.K. Lindsey, N. Goldman, L.E. Fried, S. Bastea, *JCP*, **153** 054103 (2020) R.K. Lindsey, L.E. Fried, N. Goldman, S. Bastea, *JCP*, **153** 134117 (2020)
Committe AL	R.K. Lindsey, L.E. Fried, N. Goldman, *JCTC*, **13**, 6222 (2017) R.K. Lindsey, N. Goldman, L.E. Fried, S. Bastea, *JCP*, **153** 054103 (2020)

CHAPTER

FIVE

EXTENDING THE ALD

UNDER CONSTRUCTION

For basic extension, add a conditional statement to `helpers.create_and_launch_job`, i.e.:

```
if args["job_system"] == "slurm":

    # Do what is currently implemented

elif args["job_system"] == "your_new_scheduler_name":

    # Use current implementation as template and modify for your scheduler

else:

    print "ERROR: Unrecognized scheduler: ", args["job_system"]
    exit()
```

and make similar edits to `helpers.wait_for_job` and `helpers.wait_for_jobs`. Search for "srun" in `*.py` files to check for compatibility with your system. When running, set `HPC_SYSTEM=your_new_scheduler_name`

To add support for cluster-based active learning, additional `utilities/new*sh` files will need to be added and properly selected for in `cluster.py`

CHAPTER

SIX

CONTACT

We can be contacted via our Google group.

This work was produced under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.