
DP-GEN

DeepModeling

Sep 04, 2022

OVERVIEW

1	Overview	1
1.1	About DP-GEN	1
1.2	Download and install	1
1.3	Use DP-GEN	2
1.4	Case Studies	2
1.5	License	2
2	Command line interface	3
2.1	Sub-commands:	3
3	Code Structure	9
4	Run	11
4.1	Overview of the Run process	11
4.2	Example-of-param.json	13
4.3	Example of machine.json	16
4.4	dpngen run param parameters	18
4.5	dpngen run machine parameters	35
5	Init	69
5.1	Init_bulk	69
5.2	dpngen init_bulk machine parameters	70
5.3	Init_surf	81
5.4	dpngen init_surf machine parameters	84
5.5	init_reaction	95
5.6	dpngen init_reaction parameters	96
5.7	dpngen init_reaction machine parameters	98
6	Simplify	133
6.1	Simplify	133
6.2	dpngen simplify parameters	135
6.3	dpngen simplify machine parameters	142
7	Auto test	177
7.1	Autotest Overview: Autotest for Deep Generator	177
7.2	Make run and post	179
7.3	Relaxation	180
7.4	Property	188
7.5	Refine	205
7.6	Reproduce	206

8	User Guide	211
8.1	Discussions	211
8.2	Issue	211
8.3	Tutorials	211
8.4	Example for parameters	211
8.5	Pull requests - How to contribute	212
8.6	Troubleshooting	212
8.7	Common Errors	212
9	Contributing Guide	215
9.1	Contributing Guide	215
10	DP-GEN API	219
10.1	dpngen package	219
11	Authors	287
	Bibliography	291
	Python Module Index	293
	Index	295

OVERVIEW

1.1 About DP-GEN

DP-GEN (Deep Generator) is a software written in Python, delicately designed to generate a deep learning based model of interatomic potential energy and force field. DP-GEN is dependent on [DeepMD-kit](#). With highly scalable interface with common softwares for molecular simulation, DP-GEN is capable to automatically prepare scripts and maintain job queues on HPC machines (High Performance Cluster) and analyze results.

If you use this software in any publication, please cite:

Yuzhi Zhang, Haidi Wang, Weijie Chen, Jinzhe Zeng, Linfeng Zhang, Han Wang, and Weinan E, DP-GEN: A concurrent learning platform for the generation of reliable deep learning based potential energy models, *Computer Physics Communications*, 2020, 107206.

1.1.1 Highlighted features

- **Accurate and efficient:** DP-GEN is capable to sample more than tens of million structures and select only a few for first principles calculation. DP-GEN will finally obtain a uniformly accurate model.
- **User-friendly and automatic:** Users may install and run DP-GEN easily. Once succusefully running, DP-GEN can dispatch and handle all jobs on HPCs, and thus there's no need for any personal effort.
- **Highly scalable:** With modularized code structures, users and developers can easily extend DP-GEN for their most relevant needs. DP-GEN currently supports for HPC systems (Slurm, PBS, LSF and cloud machines), Deep Potential interface with DeePMD-kit, MD interface with [LAMMPS](#), [Gromacs](#) and *ab-initio* calculation interface with VASP, PWSCF, CP2K, SIESTA and Gaussian, Abacus, PWMAT, etc . We're sincerely welcome and embraced to users' contributions, with more possibilities and cases to use DP-GEN.

1.2 Download and install

Please follow our [GitHub](#) webpage to download the [latest released version](#) and [development version](#). One can download the source code of dpngen by

```
git clone https://github.com/deepmodeling/dpngen.git
```

DP-GEN offers multiple installation methods. It is recommend using easily methods like:

- offline packages: find them in [releases](#),
- pip: use `pip install dpngen`, see [dpngen-PyPI](#)

- [conda](#): use `conda install -c deepmodeling dpgen`, see [dpgen-conda](#)

To test if the installation is successful, you may execute

```
dpgen -h
```

or just

```
dpgen
```

1.3 Use DP-GEN

A quick-start on using DPGEN can be found [here](#). You can follow the [Handson tutorial](#), it is friendly to new users.

1.4 Case Studies

- [Practical-Guidelines-for-DP](#)

Before starting a new Deep Potential (DP) project, we suggest people (especially those who are newbies) read the following context first to get some insights into what tools we can use, what kinds of risks and difficulties we may meet, and how we can advance a new DP project smoothly.

- [Convergence-Test](#)

to ensure the data quality, the reliability of the final model, as well as the feasibility of the project, a convergence test should be done first.

- [Gas-phase](#)

In this tutorial, we will take the simulation of methane combustion as an example and introduce the procedure of DP-based MD simulation.

- [Mg-Y_alloy](#)

We will briefly analyze the candidate configurational space of a metallic system by taking Mg-based Mg-Y binary alloy as an example. The task is divided into steps during the DP-GEN process.

- [Transfer-learning](#)

This tutorial will introduce how to implement potential energy surface (PES) transfer-learning by using the DP-GEN software. In DP-GEN (version > 0.8.0), the “simplify” module is designed for this purpose.

1.5 License

The project dpgen is licensed under [GNU LGPLv3.0](#)

COMMAND LINE INTERFACE

dpngen is a convenient script that uses DeepGenerator to prepare initial data, drive DeepMDkit and analyze results. This script works based on several sub-commands with their own options. To see the options for the sub-commands, type “dpngen sub-command -h”.

```
usage: dpngen [-h]
              {init_surf,init_bulk,auto_gen_param,init_reaction,run,run/report,collect,
              ↪simplify,autotest,db}
              ...
```

2.1 Sub-commands:

2.1.1 init_surf

Generating initial data for surface systems.

```
dpngen init_surf [-h] PARAM [MACHINE]
```

Positional Arguments

PARAM	parameter file, json/yaml format
MACHINE	machine file, json/yaml format

2.1.2 init_bulk

Generating initial data for bulk systems.

```
dpngen init_bulk [-h] PARAM [MACHINE]
```

Positional Arguments

PARAM	parameter file, json/yaml format
MACHINE	machine file, json/yaml format

2.1.3 auto_gen_param

auto gen param.json

```
dpngen auto_gen_param [-h] PARAM
```

Positional Arguments

PARAM	parameter file, json/yaml format
--------------	----------------------------------

2.1.4 init_reaction

Generating initial data for reactive systems.

```
dpngen init_reaction [-h] PARAM [MACHINE]
```

Positional Arguments

PARAM	parameter file, json/yaml format
MACHINE	machine file, json/yaml format

2.1.5 run

Main process of Deep Potential Generator.

```
dpngen run [-h] [-d] PARAM MACHINE
```

Positional Arguments

PARAM	parameter file, json/yaml format
MACHINE	machine file, json/yaml format

Named Arguments

-d, --debug log debug info
Default: False

2.1.6 run/report

Report the systems and the thermodynamic conditions of the labeled frames.

```
dpgen run/report [-h] [-s] [-i] [-t] [-p PARAM] [-v] JOB_DIR
```

Positional Arguments

JOB_DIR the directory of the DP-GEN job,

Named Arguments

-s, --stat-sys count the labeled frames for each system
Default: False

-i, --stat-iter print the iteration candidate,failed,accurate count and fp calculation,success and fail count
Default: False

-t, --stat-time print the iteration time, warning!! assume model_devi parallel cores == 1
Default: False

-p, --param the json file provides DP-GEN paramters, should be located in JOB_DIR
Default: "param.json"

-v, --verbose being loud
Default: False

2.1.7 collect

Collect data.

```
dpgen collect [-h] [-p PARAMETER] [-v] [-m] [-s] JOB_DIR OUTPUT
```

Positional Arguments

JOB_DIR the directory of the DP-GEN job
OUTPUT the output directory of data

Named Arguments

-p, --parameter	the json file provides DP-GEN paramters, should be located in JOB_DIR Default: "param.json"
-v, --verbose	print number of data in each system Default: False
-m, --merge	merge the systems with the same chemical formula Default: False
-s, --shuffle	shuffle the data systems Default: False

2.1.8 simplify

Simplify data.

```
dpngen simplify [-h] [-d] PARAM MACHINE
```

Positional Arguments

PARAM	parameter file, json/yaml format
MACHINE	machine file, json/yaml format

Named Arguments

-d, --debug	log debug info Default: False
--------------------	----------------------------------

2.1.9 autotest

Auto-test for Deep Potential.

```
dpngen autotest [-h] [-d] TASK PARAM [MACHINE]
```

Positional Arguments

TASK	task can be make, run or post
PARAM	parameter file, json/yaml format
MACHINE	machine file, json/yaml format

Named Arguments

-d, --debug log debug info
Default: False

2.1.10 db

Collecting data from DP-GEN.

`dpgen db [-h] PARAM`

Positional Arguments

PARAM parameter file, json format

CODE STRUCTURE

Let's look at the home page of DP-GEN. <https://github.com/deepmodeling/dpgen>

```
— build
— CITATION.cff
— conda
— dist
— doc
— dpgen
— dpgen.egg-info
— examples
— LICENSE
— README.md
— requirements.txt
— setup.py
— tests
```

- `tests` : unittest tools for developers.
- `examples`: templates for PARAM and MACHINE files for different software, versions and tasks. For details of the parameters in PARAM, you can refer to `TASK parameters` chapters in this document. If you are confused about how to set up a JSON file, you can also use `dpgui`

Most of the code related to DP-GEN functions is in the `dpgen` directory. Open the `dpgen` directory, and we can see

```
— arginfo.py
— auto_test
— collect
— data
— database
— _date.py
— dispatcher
— generator
— __init__.py
— main.py
— __pycache__
— remote
— simplify
— tools
— util.py
— _version.py
```

- `auto_test` corresponds to `dpgen autotest`, for undertaking materials property analysis.

- `collect` corresponds to `dpgen collect`.
- `data` corresponds to `dpgen init_bulk`, `dpgen init_surf` and `dpgen init_reaction`, for preparing initial data of bulk and surf systems.
- `database` is the source code for collecting data generated by DP-GEN and interface with database.
- `simplify` corresponds to `dpgen simplify`.
- `remote` and `dispatcher` : source code for automatically submitting scripts, maintaining job queues and collecting results. **Notice this part has been integrated into `dpdispatcher`** generator is the core part of DP-GEN. It's for main process of deep generator. Let's open this folder.

```
|— arginfo.py
|— ch4
|— __init__.py
|— lib
|— run.py
```

`run.py` is the core of DP-GEN, corresponding to `dpgen run`. We can find `make_train`, `run_train`, ... `post_fp`, and other steps related functions here.

4.1 Overview of the Run process

The run process contains a series of successive iterations, undertaken in order such as heating the system to certain temperatures. Each iteration is composed of three steps: exploration, labeling, and training. Accordingly, there are three sub-folders: 00.train, 01.model_devi, and 02.fp in each iteration.

00.train: DP-GEN will train several (default 4) models based on initial and generated data. The only difference between these models is the random seed for neural network initialization.

01.model_devi : represent for model-deviation. DP-GEN will use models obtained from 00.train to run Molecular Dynamics(default LAMMPS). Larger deviation for structure properties (default is the force of atoms) means less accuracy of the models. Using this criterion, a few structures will be selected and put into the next stage 02.fp for more accurate calculation based on First Principles.

02.fp : Selected structures will be calculated by first-principles methods(default VASP). DP-GEN will obtain some new data and put them together with initial data and data generated in previous iterations. After that, new training will be set up and DP-GEN will enter the next iteration!

In the run process of the DP-GEN, we need to specify the basic information about the system, the initial data, and details of the training, exploration, and labeling tasks. In addition, we need to specify the software, machine environment, and computing resource and enable the process of job generation, submission, query, and collection automatically. We can perform the run process as we expect by specifying the keywords in param.json and machine.json, and they will be introduced in detail in the following sections.

Here, we give a general description of the run process. We can execute the run process of DP-GEN easily by:

```
dpgen run param.json machine.json
```

The following files or folders will be created and upgraded by codes

- iter.00000x contains the main results that DP-GEN generates in the first iteration.
- record.dpgen records the current stage of the run process.
- dpgen.log includes time and iteration information.

When the first iteration is completed, the folder structure of iter.000000 is like this:

```
$ ls iter.000000
00.train 01.model_devi 02.fp
```

In folder iter.000000/ 00.train:

- Folder 00x contains the input and output files of the DeePMD-kit, in which a model is trained.

- graph.00x.pb is the model DeePMD-kit generates. The only difference between these models is the random seed for neural network initialization.

In folder iter.000000/ 01.model_devi

- Folder confs contains the initial configurations for LAMMPS MD converted from POSCAR you set in "sys_configs" of param.json.
- Folder task.000.00000x contains the input and output files of the LAMMPS. In folder task.000.00000x, file model_devi.out records the model deviation of concerned labels, energy and force in MD. It serves as the criterion for selecting which structures and doing first-principle calculations.

In folder iter.000000/ 02.fp

- candidate.shuffle.000.out records which structures will be selected from last step 01.model_devi. There are always far more candidates than the maximum you expect to calculate at one time. In this condition, DP-GEN will randomly choose up to "fp_task_max" structures and form the folder task.*.
- rest_accurate.shuffle.000.out records the other structures where our model is accurate ("max_devi_f" is less than "model_devi_f_trust_lo", no need to calculate any more),
- rest_failed.shuffled.000.out records the other structures where our model is too inaccurate (larger than "model_devi_f_trust_hi", there may be some error).
- data.000: After first-principle calculations, DP-GEN will collect these data and change them into the format DeePMD-kit needs. In the next iteration's 00.train, these data will be trained together as well as the initial data.

DP-GEN identifies the stage of the run process by a record file, record.dpgen, which will be created and upgraded by codes. Each line contains two numbers: the first is the index of iteration, and the second, ranging from 0 to 9, records which stage in each iteration is currently running.

Index of iterations	Stage in each iteration	Process
0	0	make_train
0	1	run_train
0	2	post_train
0	3	make_model_devi
0	4	run_model_devi
0	5	post_model_devi
0	6	make_fp
0	7	run_fp
0	8	post_fp

0,1,2 correspond to make_train, run_train, post_train. DP-GEN will write scripts in make_train, run the task by specific machine in run_train and collect result in post_train. The records for model_devi and fp stage follow similar rules.

If the process of DP-GEN stops for some reasons, DP-GEN will automatically recover the main process by record.dpgen. You may also change it manually for your purpose, such as removing the last iterations and recovering from one check-point.

4.2 Example-of-param.json

We have provided different examples of param.json in dpgen/examples/run/. In this section, we give a description of the param.json, taking dpgen/examples/run/dp2.x-lammps-vasp/param_CH4_deepmd-kit-2.0.1.json as an example. This is a param.json for a gas-phase methane molecule. Here, DeePMD-kit (v2.x), LAMMPS and VASP codes are used for training, exploration and labeling respectively.

4.2.1 basics

The basics related keys in param.json are given as follows

```
"type_map": [
  "H",
  "C"
],
"mass_map": [
  1,
  12
],
```

The basics related keys specify the basic information about the system. “type_map” gives the atom types, i.e. “H” and “C”. “mass_map” gives the standard atom weights, i.e. “1” and “12”.

4.2.2 data

The data related keys in param.json are given as follows

```
"init_data_prefix": "...../init/",
"init_data_sys": [
  "CH4.POSCAR.01x01x01/02.md/sys-0004-0001/deepmd"
],

"sys_configs_prefix": "...../init/",
"sys_configs": [
  [
    "CH4.POSCAR.01x01x01/01.scale_pert/sys-0004-0001/scale*/00000*/POSCAR"
  ],
  [
    "CH4.POSCAR.01x01x01/01.scale_pert/sys-0004-0001/scale*/00001*/POSCAR"
  ]
],
```

The data related keys specify the init data for training initial DP models and structures used for model_devi calculations. “init_data_prefix” and “init_data_sys” specify the location of the init data. “sys_configs_prefix” and “sys_configs” specify the location of the structures.

Here, the init data is provided at “..... /init/CH4.POSCAR.01x01x01/02.md/sys-0004-0001/deepmd”. These structures are divided into two groups and provided at “...../init/CH4.POSCAR.01x01x01/01.scale_pert/sys-0004-0001/scale*/00000*/POSCAR” and “...../init/CH4.POSCAR.01x01x01/01.scale_pert/sys-0004-0001/scale*/00001*/POSCAR”.

4.2.3 training

The training related keys in param.json are given as follows

```
"numb_models": 4,
"default_training_param": {
},
```

The training related keys specify the details of training tasks. “numb_models” specifies the number of models to be trained. “default_training_param” specifies the training parameters for deepmd-kit.

Here, 4 DP models will be trained in `00.train`. A detailed explanation of training parameters can be found in DeePMD-kit’s documentation (<https://docs.deepmodeling.com/projects/deepmd/en/master/>).

4.2.4 exploration

The exploration related keys in param.json are given as follows

```
"model_devi_dt": 0.002,
"model_devi_skip": 0,
"model_devi_f_trust_lo": 0.05,
"model_devi_f_trust_hi": 0.15,
"model_devi_clean_traj": true,
"model_devi_jobs": [
  {
    "sys_idx": [
      0
    ],
    "temps": [
      100
    ],
    "press": [
      1.0
    ],
    "trj_freq": 10,
    "nsteps": 300,
    "ensemble": "nvt",
    "_idx": "00"
  },
  {
    "sys_idx": [
      1
    ],
    "temps": [
      100
    ],
    "press": [
      1.0
    ],
    "trj_freq": 10,
    "nsteps": 3000,
    "ensemble": "nvt",
    "_idx": "01"
  }
],
```

(continues on next page)

(continued from previous page)

```
}
],
```

The exploration related keys specify the details of exploration tasks. “model_devi_dt” specifies timestep for MD simulation. “model_devi_skip” specifies the number of structures skipped for saving in each MD. “model_devi_f_trust_lo” and “model_devi_f_trust_hi” specify the lower and upper bound of model_devi of forces for the selection. “model_devi_clean_traj” specifies whether to clean traj folders in MD. If type of model_devi_clean_traj is boolean type then it denote whether to clean traj folders in MD since they are too large. In “model_devi_jobs”, “sys_idx” specifies the group of structures used for model_devi calculations, “temps” specifies the temperature (K) in MD, “press” specifies the pressure (Bar) in MD, “trj_freq” specifies the frequency of trajectory saved in MD, “nsteps” specifies the running steps of MD, “ensemble” specifies the ensemble used in MD, and “_idx” specifies the index of iteration.

Here, MD simulations are performed at the temperature of 100 K and the pressure of 1.0 Bar with an integrator time of 2 fs under the nvt ensemble. Two iterations are set in “model_devi_jobs”. MD simulations are run for 300 and 3000 time steps with the first and second groups of structures in “sys_configs” in 00 and 01 iterations. We choose to save all structures generated in MD simulations and have set “trj_freq” as 10, so 30 and 300 structures are saved in 00 and 01 iterations. If the “max_devi_f” of saved structure falls between 0.05 and 0.15, DP-GEN will treat the structure as a candidate. We choose to clean traj folders in MD since they are too large. If you want to save the most recent n iterations of traj folders, you can set “model_devi_clean_traj” to be an integer.

4.2.5 labeling

The labeling related keys in param.json are given as follows

```
"fp_style": "vasp",
"shuffle_poscar": false,
"fp_task_max": 20,
"fp_task_min": 1,
"fp_pp_path": "...../methane/",
"fp_pp_files": [
    "POTCAR"
],
"fp_incar": "...../INCAR_methane"
```

The labeling related keys specify the details of labeling tasks. “fp_style” specifies software for First Principles. “fp_task_max” and “fp_task_min” specify the minimum and maximum of structures to be calculated in 02.fp of each iteration. “fp_pp_path” and “fp_pp_files” specify the location of the pseudo-potential file to be used for 02.fp. “fp_incar” specifies input file for VASP. INCAR must specify KSPACING and KGAMMA.

Here, a minimum of 1 and a maximum of 20 structures will be labeled using the VASP code with the INCAR provided at “...../INCAR_methane” and POTCAR provided at “...../methane/POTCAR” in each iteration. Note that the order of elements in POTCAR should correspond to the order in type_map.

All the keys of the DP-GEN are explained in detail in the section Parameters.

4.3 Example of machine.json

4.3.1 DPDispatcher Update Note

DPDispatcher has updated and the api of machine.json is changed. DP-GEN will use the new DPDispatcher if the value of key “api_version” in machine.json is equal to or large than 1.0. And for now, DPDispatcher is maintained on a separate repo (<https://github.com/deepmodeling/dpdispatcher>). Please check the documents (<https://deepmd.readthedocs.io/projects/dpdispatcher/en/latest/>) for more information about the new DPDispatcher.

DP-GEN will use the old DPDispatcher if the key “api_version” is not specified in machine.json or the “api_version” is smaller than 1.0. This gurantees that the old machine.json still works.

4.3.2 New DPDispatcher

Each iteration in the run process of DP-GEN is composed of three steps: exploration, labeling, and training. Accordingly, machine.json is composed of three parts: train, model_devi, and fp. Each part is a list of dicts. Each dict can be considered as an independent environment for calculation.

In this section, we will show you how to perform train task at a local workstation, model_devi task at a local Slurm cluster, and fp task at a remote PBS cluster using the new DPDispatcher. For each task, three types of keys are needed:

- Command: provides the command used to execute each step.
- Machine: specifies the machine environment (local workstation, local or remote cluster, or cloud server).
- Resources: specify the number of groups, nodes, CPU, and GPU; enable the virtual environment.

Performing train task at a local workstation

In this example, we perform the train task on a local workstation.

```
"train":
{
  "command": "dp",
  "machine": {
    "batch_type": "Shell",
    "context_type": "local",
    "local_root": "./",
    "remote_root": "/home/user1234/work_path"
  },
  "resources": {
    "number_node": 1,
    "cpu_per_node": 4,
    "gpu_per_node": 1,
    "group_size": 1,
    "source_list": ["/home/user1234/deepmd.env"]
  }
},
```

The “command” for the train task in the DeePMD-kit is “dp”.

In machine parameters, “batch_type” specifies the type of job scheduling system. If there is no job scheduling system, we can use the “Shell” to perform the task. “context_type” specifies the method of data transfer, and “local” means copying and moving data via local file storage systems (e.g. cp, mv, etc.). In DP-GEN, the paths of all tasks are

automatically located and set by the software, and therefore “local_root” is always set to “.”. The input file for each task will be sent to the “remote_root” and the task will be performed there, so we need to make sure that the path exists.

In the resources parameter, “number_node”, “cpu_per_node”, and “gpu_per_node” specify the number of nodes, the number of CPUs, and the number of GPUs required for a task respectively. “group_size”, which needs to be highlighted, specifies how many tasks will be packed into a group. In the training tasks, we need to train 4 models. If we only have one GPU, we can set the “group_size” to 4. If “group_size” is set to 1, 4 models will be trained on one GPU at the same time, as there is no job scheduling system. Finally, the environment variables can be activated by “source_list”. In this example, “source /home/user1234/deepmd.env” is executed before “dp” to load the environment variables necessary to perform the training task.

Perform model_devi task at a local Slurm cluster

In this example, we perform the model_devi task at a local Slurm workstation.

```
"model_devi":
{
  "command": "lmp",
  "machine": {
    "context_type": "local",
    "batch_type": "Slurm",
    "local_root": "./",
    "remote_root": "/home/user1234/work_path"
  },
  "resources": {
    "number_node": 1,
    "cpu_per_node": 4,
    "gpu_per_node": 1,
    "queue_name": "QueueGPU",
    "custom_flags": ["#SBATCH --mem=32G"],
    "group_size": 10,
    "source_list": ["/home/user1234/lammps.env"]
  }
}
```

The “command” for the model_devi task in the LAMMPS is “lmp”.

In the machine parameter, we specify the type of job scheduling system by changing the “batch_type” to “Slurm”.

In the resources parameter, we specify the name of the queue to which the task is submitted by adding “queue_name”. We can add additional lines to the calculation script via the “custom_flags”. In the model_devi steps, there are frequently many short tasks, so we usually pack multiple tasks (e.g. 10) into a group for submission. Other parameters are similar to that of the local workstation.

Perform fp task in a remote PBS cluster

In this example, we perform the fp task at a remote PBS cluster that can be accessed via SSH.

```
"fp":
{
  "command": "mpirun -n 32 vasp_std",
  "machine": {
    "context_type": "SSHContext",
    "batch_type": "PBS",
```

(continues on next page)

(continued from previous page)

```

    "local_root": "./",
    "remote_root": "/home/user1234/work_path",
    "remote_profile": {
        "hostname": "39.xxx.xx.xx",
        "username": "user1234"
    }
},
"resources": {
    "number_node": 1,
    "cpu_per_node": 32,
    "gpu_per_node": 0,
    "queue_name": "QueueCPU",
    "group_size": 5,
    "source_list": ["/home/user1234/vasp.env"]
}
}

```

VASP code is used for fp task and mpi is used for parallel computing, so “mpirun -n 32” is added to specify the number of parallel threads.

In the machine parameter, “context_type” is modified to “SSHContext” and “batch_type” is modified to “PBS”. It is worth noting that “remote_root” should be set to an accessible path on the remote PBS cluster. “remote_profile” is added to specify the information used to connect the remote cluster, including hostname, username, port, etc.

In the resources parameter, we set “gpu_per_node” to 0 since it is cost-effective to use the CPU for VASP calculations.

Explicit descriptions of keys in machine.json will be given in the following section.

4.4 dpgen run param parameters

run_jdata:

type: dict
 argument path: run_jdata
 param.json file

type_map:

type: list
 argument path: run_jdata/type_map
 Atom types.

mass_map:

type: list | str, optional, default: auto
 argument path: run_jdata/mass_map

Standard atomic weights (default: “auto”). if one want to use isotopes, or non-standard element names, chemical symbols, or atomic number in the type_map list, please customize the mass_map list instead of using “auto”. Tips: at present the default value will not be applied automatically, so you need to set “mass_map” manually in param.json.

use_ele_temp:

type: int, optional, default: 0

argument path: `run_jdata/use_ele_temp`

Currently only support `fp_style vasp`.

- 0: no electron temperature.
- 1: electron temperature as frame parameter.
- 2: electron temperature as atom parameter.

init_data_prefix:

type: `str`, optional

argument path: `run_jdata/init_data_prefix`

Prefix of initial data directories.

init_data_sys:

type: `list`

argument path: `run_jdata/init_data_sys`

Directories of initial data. You may use either absolute or relative path here. Systems will be detected recursively in the directories.

sys_format:

type: `str`, optional, default: `vasp/poscar`

argument path: `run_jdata/sys_format`

Format of initial data.

init_batch_size:

type: `list | str`, optional

argument path: `run_jdata/init_batch_size`

Each number is the `batch_size` of corresponding system for training in `init_data_sys`. One recommended rule for setting the `sys_batch_size` and `init_batch_size` is that `batch_size` multiply number of atoms of the structure should be larger than 32. If set to `auto`, batch size will be 32 divided by number of atoms.

sys_configs_prefix:

type: `str`, optional

argument path: `run_jdata/sys_configs_prefix`

Prefix of `sys_configs`.

sys_configs:

type: `list`

argument path: `run_jdata/sys_configs`

Containing directories of structures to be explored in iterations. Wildcard characters are supported here.

sys_batch_size:

type: `list`, optional

argument path: `run_jdata/sys_batch_size`

Each number is the `batch_size` for training of corresponding system in `sys_configs`. If set to `auto`, batch size will be 32 divided by number of atoms.

numb_models:

type: `int`

argument path: `run_jdata/numb_models`

Number of models to be trained in `00.train`. 4 is recommend.

training_iter0_model_path:

type: list, optional

argument path: run_jdata/training_iter0_model_path

The model used to init the first iter training. Number of element should be equal to numb_models.

training_init_model:

type: bool, optional

argument path: run_jdata/training_init_model

Iteration > 0, the model parameters will be initilized from the model trained at the previous iteration. Iteration == 0, the model parameters will be initialized from training_iter0_model_path.

default_training_param:

type: dict

argument path: run_jdata/default_training_param

Training parameters for deepmd-kit in 00.train. You can find instructions from here: (<https://github.com/deepmodeling/deepmd-kit>).

dp_compress:

type: bool, optional, default: False

argument path: run_jdata/dp_compress

Use dp compress to compress the model.

training_reuse_iter:

type: int | NoneType, optional

argument path: run_jdata/training_reuse_iter

The minimal index of iteration that continues training models from old models of last iteration.

training_reuse_old_ratio:

type: NoneType | float, optional

argument path: run_jdata/training_reuse_old_ratio

The probability proportion of old data during training. This option is only adopted when continuing training models from old models. This option will override default parameters.

training_reuse_numb_steps:

type: int | NoneType, optional, default: 400000, alias: *training_reuse_stop_batch*

argument path: run_jdata/training_reuse_numb_steps

Number of training batch. This option is only adopted when continuing training models from old models. This option will override default parameters.

training_reuse_start_lr:

type: NoneType | float, optional, default: 0.0001

argument path: run_jdata/training_reuse_start_lr

The learning rate the start of the training. This option is only adopted when continuing training models from old models. This option will override default parameters.

training_reuse_start_pref_e:

type: int | NoneType | float, optional, default: 0.1

argument path: run_jdata/training_reuse_start_pref_e

The prefactor of energy loss at the start of the training. This option is only adopted when continuing training models from old models. This option will override default parameters.

training_reuse_start_pref_f:

type: int | NoneType | float, optional, default: 100

argument path: run_jdata/training_reuse_start_pref_f

The prefactor of force loss at the start of the training. This option is only adopted when continuing training models from old models. This option will override default parameters.

model_devi_activation_func:

type: list | NoneType, optional

argument path: run_jdata/model_devi_activation_func

The activation function in the model. The shape of list should be (N_models, 2), where 2 represents the embedding and fitting network. This option will override default parameters.

fp_task_max:

type: int

argument path: run_jdata/fp_task_max

Maximum of structures to be calculated in 02.fp of each iteration.

fp_task_min:

type: int

argument path: run_jdata/fp_task_min

Minimum of structures to be calculated in 02.fp of each iteration.

fp_accurate_threshold:

type: float, optional

argument path: run_jdata/fp_accurate_threshold

If the accurate ratio is larger than this number, no fp calculation will be performed, i.e. fp_task_max = 0.

fp_accurate_soft_threshold:

type: float, optional

argument path: run_jdata/fp_accurate_soft_threshold

If the accurate ratio is between this number and fp_accurate_threshold, the fp_task_max linearly decays to zero.

fp_cluster_vacuum:

type: float, optional

argument path: run_jdata/fp_cluster_vacuum

If the vacuum size is smaller than this value, this cluster will not be chosen for labeling.

detailed_report_make_fp:

type: bool, optional, default: True

argument path: `run_jdata/detailed_report_make_fp`

If set to true, detailed report will be generated for each iteration.

Depending on the value of `model_devi_engine`, different sub args are accepted.

model_devi_engine:

type: str (flag key), default: `lammps`

argument path: `run_jdata/model_devi_engine`

possible choices: `lammps`, `amber`

Engine for the model deviation task.

When `model_devi_engine` is set to `lammps`:

LAMMPS

model_devi_jobs:

type: list

argument path: `run_jdata[model_devi_engine=lammps]/model_devi_jobs`

Settings for exploration in 01.model_devi. Each dict in the list corresponds to one iteration. The index of `model_devi_jobs` exactly accord with index of iterations

This argument takes a list with each element containing the following:

sys_idx:

type: list

argument path:

`run_jdata[model_devi_engine=lammps]/model_devi_jobs/sys_idx`

Systems to be selected as the initial structure of MD and be explored. The index corresponds exactly to the `sys_configs`.

temps:

type: list

argument path:

`run_jdata[model_devi_engine=lammps]/model_devi_jobs/temps`

Temperature (K) in MD.

press:

type: list, optional

argument path:

`run_jdata[model_devi_engine=lammps]/model_devi_jobs/press`

Pressure (Bar) in MD. Required when ensemble is npt.

trj_freq:

type: int

argument path: `run_jdata[model_devi_engine=lammps]/model_devi_jobs/trj_freq`

Frequency of trajectory saved in MD.

nsteps:

type: int

argument path:

`run_jdata[model_devi_engine=lammps]/model_devi_jobs/nsteps`

Running steps of MD.

ensemble:

type: str

argument path: `run_jdata[model_devi_engine=lammps]/model_devi_jobs/ensemble`

Determining which ensemble used in MD, options include “npt” and “nvt”.

neidelay:

type: int, optional

argument path: `run_jdata[model_devi_engine=lammps]/model_devi_jobs/neidelay`

delay building until this many steps since last build.

taut:

type: float, optional

argument path:

`run_jdata[model_devi_engine=lammps]/model_devi_jobs/taut`

Coupling time of thermostat (ps).

taup:

type: float, optional

argument path:

`run_jdata[model_devi_engine=lammps]/model_devi_jobs/taup`

Coupling time of barostat (ps).

model_devi_f_trust_lo:

type: dict | float, optional

argument path: `run_jdata[model_devi_engine=lammps]/model_devi_jobs/model_devi_f_trust_lo`

Lower bound of forces for the selection. If dict, should be set for each index in `sys_idx`, respectively.

model_devi_f_trust_hi:

type: dict | float, optional

argument path: `run_jdata[model_devi_engine=lammps]/model_devi_jobs/model_devi_f_trust_hi`

Upper bound of forces for the selection. If dict, should be set for each index in `sys_idx`, respectively.

model_devi_v_trust_lo:

type: dict | float, optional

argument path: `run_jdata[model_devi_engine=lammps]/model_devi_jobs/model_devi_v_trust_lo`

Lower bound of virial for the selection. If dict, should be set for each index in `sys_idx`, respectively. Should be used with DeePMD-kit v2.x.

model_devi_v_trust_hi:

type: dict | float, optional

argument path: `run_jdata[model_devi_engine=lammps]/model_devi_jobs/model_devi_v_trust_hi`

Upper bound of virial for the selection. If dict, should be set for each index in `sys_idx`, respectively. Should be used with DeePMD-kit v2.x.

model_devi_dt:

type: float

argument path: `run_jdata[model_devi_engine=lammps]/model_devi_dt`

Timestep for MD. 0.002 is recommend.

model_devi_skip:

type: int

argument path: `run_jdata[model_devi_engine=lammps]/model_devi_skip`

Number of structures skipped for fp in each MD.

model_devi_f_trust_lo:

type: list | dict | float

argument path:

`run_jdata[model_devi_engine=lammps]/model_devi_f_trust_lo`

Lower bound of forces for the selection. If list or dict, should be set for each index in `sys_configs`, respectively.

model_devi_f_trust_hi:

type: list | dict | float

argument path:

`run_jdata[model_devi_engine=lammps]/model_devi_f_trust_hi`

Upper bound of forces for the selection. If list or dict, should be set for each index in `sys_configs`, respectively.

model_devi_v_trust_lo:

type: list | dict | float, optional, default: 10000000000.0

argument path:

`run_jdata[model_devi_engine=lammps]/model_devi_v_trust_lo`

Lower bound of virial for the selection. If list or dict, should be set for each index in `sys_configs`, respectively. Should be used with DeePMD-kit v2.x.

model_devi_v_trust_hi:

type: list | dict | float, optional, default: 10000000000.0

argument path:

`run_jdata[model_devi_engine=lammps]/model_devi_v_trust_hi`

Upper bound of virial for the selection. If list or dict, should be set for each index in `sys_configs`, respectively. Should be used with DeePMD-kit v2.x.

model_devi_adapt_trust_lo:

type: bool, optional

argument path:

`run_jdata[model_devi_engine=lammps]/model_devi_adapt_trust_lo`

Adaptively determines the lower trust levels of force and virial. This option should be used together with `model_devi_numb_candi_f`, `model_devi_numb_candi_v` and optionally with `model_devi_perc_candi_f` and `model_devi_perc_candi_v`. dpngen will make two sets:

- 1. From the frames with force model deviation lower than `model_devi_f_trust_hi`, select `max(model_devi_numb_candi_f, model_devi_perc_candi_f*n_frames)` frames with largest force model deviation.
- 2. From the frames with virial model deviation lower than `model_devi_v_trust_hi`, select `max(model_devi_numb_candi_v, model_devi_perc_candi_v*n_frames)` frames with largest virial model deviation.

The union of the two sets is made as candidate dataset.

model_devi_numb_candi_f:

type: int, optional

argument path:

`run_jdata[model_devi_engine=lammps]/model_devi_numb_candi_f`

See `model_devi_adapt_trust_lo`.

model_devi_numb_candi_v:

type: int, optional

argument path:

`run_jdata[model_devi_engine=lammps]/model_devi_numb_candi_v`

See `model_devi_adapt_trust_lo`.

model_devi_perc_candi_f:

type: float, optional

argument path:

`run_jdata[model_devi_engine=lammps]/model_devi_perc_candi_f`

See `model_devi_adapt_trust_lo`.

model_devi_perc_candi_v:

type: float, optional

argument path:

`run_jdata[model_devi_engine=lammps]/model_devi_perc_candi_v`

See `model_devi_adapt_trust_lo`.

model_devi_f_avg_relative:

type: bool, optional

argument path:

`run_jdata[model_devi_engine=lammps]/model_devi_f_avg_relative`

Normalized the force model deviations by the RMS force magnitude along the trajectory. This key should not be used with `use_relative`.

model_devi_clean_traj:

type: int | bool, optional, default: True

argument path:

`run_jdata[model_devi_engine=lammps]/model_devi_clean_traj`

If type of `model_devi_clean_traj` is bool type then it denote whether to clean traj folders in MD since they are too large. If it is Int type, then the most recent n iterations of traj folders will be retained, others will be removed.

model_devi_merge_traj:

type: bool, optional, default: False

argument path:

run_jdata[model_devi_engine=lammps]/model_devi_merge_traj

If model_devi_merge_traj is set as True, only all.lammpstrj will be generated, instead of lots of small traj files.

model_devi_nopbc:

type: bool, optional, default: False

argument path: run_jdata[model_devi_engine=lammps]/model_devi_nopbc

Assume open boundary condition in MD simulations.

shuffle_poscar:

type: bool, optional, default: False

argument path: run_jdata[model_devi_engine=lammps]/shuffle_poscar

Shuffle atoms of each frame before running simulations. The purpose is to sample the element occupation of alloys.

use_relative:

type: bool, optional, default: False

argument path: run_jdata[model_devi_engine=lammps]/use_relative

Calculate relative force model deviation.

epsilon:

type: float, optional

argument path: run_jdata[model_devi_engine=lammps]/epsilon

The level parameter for computing the relative force model deviation.

use_relative_v:

type: bool, optional, default: False

argument path: run_jdata[model_devi_engine=lammps]/use_relative_v

Calculate relative virial model deviation.

epsilon_v:

type: float, optional

argument path: run_jdata[model_devi_engine=lammps]/epsilon_v

The level parameter for computing the relative virial model deviation.

When `model_devi_engine` is set to `amber`:

Amber DPRc engine. The command argument in the machine file should be path to sander.

model_devi_jobs:

type: list

argument path: run_jdata[model_devi_engine=amber]/model_devi_jobs

List of dicts. The list including the dict for information of each cycle.

This argument takes a list with each element containing the following:

sys_idx:

type: list

argument path: run_jdata[model_devi_engine=amber]/
model_devi_jobs/sys_idx

List of ints. List of systems to run.

trj_freq:

type: int

argument path: run_jdata[model_devi_engine=amber]/
model_devi_jobs/trj_freq

Frequency to dump trajectory.

low_level:

type: str

argument path: run_jdata[model_devi_engine=amber]/low_level

Low level method. The value will be filled into mdin file as @qm_theory@.

cutoff:

type: float

argument path: run_jdata[model_devi_engine=amber]/cutoff

Cutoff radius for the DPRc model.

parm7_prefix:

type: str, optional

argument path: run_jdata[model_devi_engine=amber]/parm7_prefix

The path prefix to AMBER PARM7 files.

parm7:

type: list

argument path: run_jdata[model_devi_engine=amber]/parm7

List of paths to AMBER PARM7 files. Each file maps to a system.

mdin_prefix:

type: str, optional

argument path: run_jdata[model_devi_engine=amber]/mdin_prefix

The path prefix to AMBER mdin template files.

mdin:

type: list

argument path: run_jdata[model_devi_engine=amber]/mdin

List of paths to AMBER mdin template files. Each files maps to a system. In the template, the following keywords will be replaced by the actual value: @freq@: freq to dump trajectory; @nstlim@: total time step to run; @qm_region@: AMBER mask of the QM region; @qm_theory@: The low level QM theory, such as DFTB2; @qm_charge@: The total charge of the QM theory, such as -2; @rcut@: cutoff radius of the DPRc model; @GRAPH_FILE0@, @GRAPH_FILE1@, ... : graph files.

qm_region:

type: list

argument path: run_jdata[model_devi_engine=amber]/qm_region

List of strings. AMBER mask of the QM region. Each mask maps to a system.

qm_charge:

type: list

argument path: run_jdata[model_devi_engine=amber]/qm_charge

List of ints. Charge of the QM region. Each charge maps to a system.

nsteps:

type: list

argument path: run_jdata[model_devi_engine=amber]/nsteps

List of ints. The number of steps to run. Each number maps to a system.

r:

type: list

argument path: run_jdata[model_devi_engine=amber]/r

3D or 4D list of floats. Constrict values for the enhanced sampling. The first dimension maps to systems. The second dimension maps to confs in each system. The third dimension is the constrict value. It can be a single float for 1D or list of floats for nD.

disang_prefix:

type: str, optional

argument path: run_jdata[model_devi_engine=amber]/disang_prefix

The path prefix to disang prefix.

disang:

type: list

argument path: run_jdata[model_devi_engine=amber]/disang

List of paths to AMBER disang files. Each file maps to a system. The keyword RVAL will be replaced by the constrict values, or RVAL1, RVAL2, ... for an nD system.

model_devi_f_trust_lo:

type: list | dict | float

argument path:

run_jdata[model_devi_engine=amber]/model_devi_f_trust_lo

Lower bound of forces for the selection. If dict, should be set for each index in sys_idx, respectively.

model_devi_f_trust_hi:

type: list | dict | float

argument path:

run_jdata[model_devi_engine=amber]/model_devi_f_trust_hi

Upper bound of forces for the selection. If dict, should be set for each index in sys_idx, respectively.

Depending on the value of *fp_style*, different sub args are accepted.

fp_style:

type: str (flag key)
 argument path: run_jdata/fp_style
 possible choices: *vasp*, *gaussian*, *siesta*, *cp2k*, *abacus*, *amber/diff*
 Software for First Principles.

When *fp_style* is set to *vasp*:

fp_pp_path:

type: str
 argument path: run_jdata[fp_style=vasp]/fp_pp_path
 Directory of psuedo-potential file to be used for 02.fp exists.

fp_pp_files:

type: list
 argument path: run_jdata[fp_style=vasp]/fp_pp_files
 Psuedo-potential file to be used for 02.fp. Note that the order of elements should correspond to the order in type_map.

fp_incar:

type: str
 argument path: run_jdata[fp_style=vasp]/fp_incar
 Input file for VASP. INCAR must specify KSPACING and KGAMMA.

fp_aniso_kspacing:

type: list, optional
 argument path: run_jdata[fp_style=vasp]/fp_aniso_kspacing
 Set anisotropic kspacing. Usually useful for 1-D or 2-D materials. Only support VASP. If it is setting the KSPACING key in INCAR will be ignored.

cvasp:

type: bool, optional
 argument path: run_jdata[fp_style=vasp]/cvasp
 If cvasp is true, DP-GEN will use Custodian to help control VASP calculation.

ratio_failed:

type: float, optional
 argument path: run_jdata[fp_style=vasp]/ratio_failed
 Check the ratio of unsuccessfully terminated jobs. If too many FP tasks are not converged, RuntimeError will be raised.

fp_skip_bad_box:

type: str, optional
 argument path: run_jdata[fp_style=vasp]/fp_skip_bad_box
 Skip the configurations that are obviously unreasonable before 02.fp

When `fp_style` is set to `gaussian`:

use_clusters:

type: bool, optional, default: False

argument path: `run_jdata[fp_style=gaussian]/use_clusters`

If set to true, clusters will be taken instead of the whole system.

cluster_cutoff:

type: float, optional

argument path: `run_jdata[fp_style=gaussian]/cluster_cutoff`

The soft cutoff radius of clusters if `use_clusters` is set to true. Molecules will be taken as whole even if part of atoms is out of the cluster. Use `cluster_cutoff_hard` to only take atoms within the hard cutoff radius.

cluster_cutoff_hard:

type: float, optional

argument path: `run_jdata[fp_style=gaussian]/cluster_cutoff_hard`

The hard cutoff radius of clusters if `use_clusters` is set to true. Outside the hard cutoff radius, atoms will not be taken even if they are in a molecule where some atoms are within the cutoff radius.

cluster_minify:

type: bool, optional, default: False

argument path: `run_jdata[fp_style=gaussian]/cluster_minify`

If enabled, when an atom within the soft cutoff radius connects a single bond with a non-hydrogen atom out of the soft cutoff radius, the outer atom will be replaced by a hydrogen atom. When the outer atom is a hydrogen atom, the outer atom will be kept. In this case, other atoms out of the soft cutoff radius will be removed.

fp_params:

type: dict

argument path: `run_jdata[fp_style=gaussian]/fp_params`

Parameters for Gaussian calculation.

keywords:

type: list | str

argument path:

`run_jdata[fp_style=gaussian]/fp_params/keywords`

Keywords for Gaussian input, e.g. `force b3lyp/6-31g**`. If a list, run multiple steps.

multiplicity:

type: int | str, optional, default: auto

argument path:

`run_jdata[fp_style=gaussian]/fp_params/multiplicity`

Spin multiplicity for Gaussian input. If *auto*, multiplicity will be detected automatically, with the following rules: when `fragment_guesses=True`, multiplicity will +1 for each radical, and +2 for each oxygen molecule; when

fragment_guesses=False, multiplicity will be 1 or 2, but +2 for each oxygen molecule.

nproc:

type: int

argument path: run_jdata[fp_style=gaussian]/fp_params/nproc

The number of processors for Gaussian input.

charge:

type: int, optional, default: 0

argument path: run_jdata[fp_style=gaussian]/fp_params/charge

Molecule charge. Only used when charge is not provided by the system.

fragment_guesses:

type: bool, optional, default: False

argument path:

run_jdata[fp_style=gaussian]/fp_params/fragment_guesses

Initial guess generated from fragment guesses. If True, *multiplicity* should be *auto*.

basis_set:

type: str, optional

argument path:

run_jdata[fp_style=gaussian]/fp_params/basis_set

Custom basis set.

keywords_high_multiplicity:

type: str, optional

argument path: run_jdata[fp_style=gaussian]/fp_params/
keywords_high_multiplicity

Keywords for points with multiple raicals. *multiplicity* should be *auto*. If not set, fallback to normal keywords.

ratio_failed:

type: float, optional

argument path: run_jdata[fp_style=gaussian]/ratio_failed

Check the ratio of unsuccessfully terminated jobs. If too many FP tasks are not converged, RuntimeError will be raised.

When `fp_style` is set to `siesta`:

use_clusters:

type: bool, optional

argument path: run_jdata[fp_style=siesta]/use_clusters

If set to true, clusters will be taken instead of the whole system. This option does not work with DeePMD-kit 0.x.

cluster_cutoff:

type: float, optional

argument path: `run_jdata[fp_style=siesta]/cluster_cutoff`

The cutoff radius of clusters if `use_clusters` is set to true.

fp_params:

type: dict

argument path: `run_jdata[fp_style=siesta]/fp_params`

Parameters for siesta calculation.

ecut:

type: int

argument path: `run_jdata[fp_style=siesta]/fp_params/ecut`

Define the plane wave cutoff for grid.

ediff:

type: float

argument path: `run_jdata[fp_style=siesta]/fp_params/ediff`

Tolerance of Density Matrix.

kspacing:

type: float

argument path: `run_jdata[fp_style=siesta]/fp_params/kspacing`

Sample factor in Brillouin zones.

mixingWeight:

type: float

argument path:

`run_jdata[fp_style=siesta]/fp_params/mixingWeight`

Proportion a of output Density Matrix to be used for the input Density Matrix of next SCF cycle (linear mixing).

NumberPulay:

type: int

argument path:

`run_jdata[fp_style=siesta]/fp_params/NumberPulay`

Controls the Pulay convergence accelerator.

fp_pp_path:

type: str

argument path: `run_jdata[fp_style=siesta]/fp_pp_path`

Directory of psuedo-potential or numerical orbital files to be used for 02.fp exists.

fp_pp_files:

type: list

argument path: `run_jdata[fp_style=siesta]/fp_pp_files`

Psuedo-potential file to be used for 02.fp. Note that the order of elements should correspond to the order in `type_map`.

When `fp_style` is set to `cp2k`:

user_fp_params:

type: dict, optional, alias: *fp_params*

argument path: `run_jdata[fp_style=cp2k]/user_fp_params`

Parameters for cp2k calculation. find detail in manual.cp2k.org. only the kind section must be set before use. we assume that you have basic knowledge for cp2k input.

external_input_path:

type: str, optional

argument path: `run_jdata[fp_style=cp2k]/external_input_path`

Conflict with key: `user_fp_params`, use the template input provided by user, some rules should be followed, read the following text in detail.

ratio_failed:

type: float, optional

argument path: `run_jdata[fp_style=cp2k]/ratio_failed`

Check the ratio of unsuccessfully terminated jobs. If too many FP tasks are not converged, `RuntimeError` will be raised.

When `fp_style` is set to `abacus`:

fp_pp_path:

type: str

argument path: `run_jdata[fp_style=abacus]/fp_pp_path`

Directory of psuedo-potential or numerical orbital files to be used for 02.fp exists.

fp_pp_files:

type: list

argument path: `run_jdata[fp_style=abacus]/fp_pp_files`

Psuedo-potential file to be used for 02.fp. Note that the order of elements should correspond to the order in `type_map`.

fp_orb_files:

type: list, optional

argument path: `run_jdata[fp_style=abacus]/fp_orb_files`

numerical orbital file to be used for 02.fp when using LCAO basis. Note that the order of elements should correspond to the order in `type_map`.

fp_incar:

type: str, optional

argument path: `run_jdata[fp_style=abacus]/fp_incar`

Input file for ABACUS. This is optinal but priority over `user_fp_params`, one can also setting the key and value of `INPUT` in `user_fp_params`.

fp_kpt_file:

type: str, optional

argument path: `run_jdata[fp_style=abacus]/fp_kpt_file`

KPT file for ABACUS.

fp_dpks_descriptor:

type: str, optional
argument path: run_jdata[fp_style=abacus]/fp_dpks_descriptor
DeePKS descriptor file name. The file should be in pseudopotential directory.

user_fp_params:

type: dict, optional
argument path: run_jdata[fp_style=abacus]/user_fp_params
Set the key and value of INPUT.

k_points:

type: list, optional
argument path: run_jdata[fp_style=abacus]/k_points
Monkhorst-Pack k-grids setting for generating KPT file of ABACUS

When `fp_style` is set to `amber/diff`:

Amber/diff style for DPRc models. Note: this fp style only supports to be used with model_devi_engine *amber*, where some arguments are reused. The command argument in the machine file should be path to sander. One should also install dpamber and make it visible in the PATH.

high_level:

type: str
argument path: run_jdata[fp_style=amber/diff]/high_level
High level method. The value will be filled into mdin template as @qm_theory@.

fp_params:

type: dict
argument path: run_jdata[fp_style=amber/diff]/fp_params
Parameters for FP calculation.

high_level_mdin:

type: str
argument path:
run_jdata[fp_style=amber/diff]/fp_params/high_level_mdin
Path to high-level AMBER mdin template file. %qm_theory%, %qm_region%, and %qm_charge% will be replaced.

low_level_mdin:

type: str
argument path:
run_jdata[fp_style=amber/diff]/fp_params/low_level_mdin
Path to low-level AMBER mdin template file. %qm_theory%, %qm_region%, and %qm_charge% will be replaced.

4.5 dpngen run machine parameters

Note: One can load, modify, and export the input file by using our effective web-based tool [DP-GUI](#). All parameters below can be set in DP-GUI. By clicking “SAVE JSON”, one can download the input file.

run_mdata:

type: dict

argument path: run_mdata

machine.json file

api_version:

type: str

argument path: run_mdata/api_version

Please set to 1.0

deepmd_version:

type: str, optional, default: 2

argument path: run_mdata/deepmd_version

DeePMD-kit version, e.g. 2.1.3

train:

type: dict

argument path: run_mdata/train

Parameters of command, machine, and resources for train

command:

type: str

argument path: run_mdata/train/command

Command of a program.

machine:

type: dict

argument path: run_mdata/train/machine

batch_type:

type: str

argument path: run_mdata/train/machine/batch_type

The batch job system type. Option: PBS, Shell, LSF, Lebesgue, Slurm, Torque, SlurmJobArray, DistributedShell, DpCloudServer

local_root:

type: NoneType | str

argument path: run_mdata/train/machine/local_root

The dir where the tasks and relating files locate. Typically the project dir.

remote_root:

type: `NoneType` | `str`, optional

argument path: `run_mdata/train/machine/remote_root`

The dir where the tasks are executed on the remote machine. Only needed when context is not lazy-local.

clean_asynchronously:

type: `bool`, optional, default: `False`

argument path:

`run_mdata/train/machine/clean_asynchronously`

Clean the remote directory asynchronously after the job finishes.

Depending on the value of `context_type`, different sub args are accepted.

context_type:

type: `str` (flag key)

argument path: `run_mdata/train/machine/context_type`

possible choices: [*LebesgueContext*](#), [*SSHContext*](#),
[*LocalContext*](#), [*DpCloudServerContext*](#),
[*LazyLocalContext*](#), [*HDFSContext*](#)

The connection used to remote machine. Option: `SSHContext`, `LazyLocalContext`, `LebesgueContext`, `LocalContext`, `DpCloudServerContext`, `HDFSContext`

When `context_type` is set to `LebesgueContext` (or its aliases `lebesguecontext`, `Lebesgue`, `lebesgue`):

remote_profile:

type: `dict`

argument path: `run_mdata/train/machine[LebesgueContext]/remote_profile`

The information used to maintain the connection with remote machine.

email:

type: `str`

argument path:

`run_mdata/train/machine[LebesgueContext]/remote_profile/email`

Email

password:

type: `str`

argument path:

`run_mdata/train/machine[LebesgueContext]/remote_profile/password`

Password

program_id:

type: `int`, alias: *project_id*

argument path:
 run_mdata/train/machine[LebesgueContext]/
 remote_profile/program_id

Program ID

keep_backup:

type: bool, optional
 argument path:
 run_mdata/train/machine[LebesgueContext]/
 remote_profile/keep_backup

keep download and upload zip

input_data:

type: dict
 argument path:
 run_mdata/train/machine[LebesgueContext]/
 remote_profile/input_data

Configuration of job

When `context_type` is set to `SSHContext` (or its aliases `sshcontext`, `SSH`, `ssh`):

remote_profile:

type: dict
 argument path: run_mdata/train/
 machine[SSHContext]/remote_profile

The information used to maintain the connection with remote machine.

hostname:

type: str
 argument path:
 run_mdata/train/machine[SSHContext]/
 remote_profile/hostname

hostname or ip of ssh connection.

username:

type: str
 argument path:
 run_mdata/train/machine[SSHContext]/
 remote_profile/username

username of target linux system

password:

type: str, optional
 argument path:
 run_mdata/train/machine[SSHContext]/
 remote_profile/password

(deprecated) password of linux system. Please use [SSH keys](#) instead to improve security.

port:

type: int, optional, default: 22

argument path:

run_mdata/train/machine[SSHContext]/
remote_profile/port

ssh connection port.

key_filename:

type: NoneType | str, optional, default: None

argument path:

run_mdata/train/machine[SSHContext]/
remote_profile/key_filename

key filename used by ssh connection. If left None,
find key in ~/.ssh or use password for login

passphrase:

type: NoneType | str, optional, default: None

argument path:

run_mdata/train/machine[SSHContext]/
remote_profile/passphrase

passphrase of key used by ssh connection

timeout:

type: int, optional, default: 10

argument path:

run_mdata/train/machine[SSHContext]/
remote_profile/timeout

timeout of ssh connection

totp_secret:

type: NoneType | str, optional, default: None

argument path:

run_mdata/train/machine[SSHContext]/
remote_profile/totp_secret

Time-based one time password secret. It should
be a base32-encoded string extracted from the 2D
code.

tar_compress:

type: bool, optional, default: True

argument path:

run_mdata/train/machine[SSHContext]/
remote_profile/tar_compress

The archive will be compressed in upload and
download if it is True. If not, compression will be
skipped.

When `context_type` is set to `LocalContext` (or its aliases `localcontext`,
`Local`, `local`):

remote_profile:

type: dict, optional
 argument path: run_mdata/train/
 machine[LocalContext]/remote_profile

The information used to maintain the connection with remote machine. This field is empty for this context.

When `context_type` is set to `DpCloudServerContext` (or its aliases `dpcloudservercontext`, `DpCloudServer`, `dpcloudserver`):

remote_profile:

type: dict
 argument path: run_mdata/train/
 machine[DpCloudServerContext]/remote_profile

The information used to maintain the connection with remote machine.

email:

type: str
 argument path: run_mdata/train/
 machine[DpCloudServerContext]/
 remote_profile/email

Email

password:

type: str
 argument path: run_mdata/train/
 machine[DpCloudServerContext]/
 remote_profile/password

Password

program_id:

type: int, alias: *project_id*
 argument path: run_mdata/train/
 machine[DpCloudServerContext]/
 remote_profile/program_id

Program ID

keep_backup:

type: bool, optional
 argument path: run_mdata/train/
 machine[DpCloudServerContext]/
 remote_profile/keep_backup

keep download and upload zip

input_data:

type: dict
 argument path: run_mdata/train/
 machine[DpCloudServerContext]/
 remote_profile/input_data

Configuration of job

When `context_type` is set to `LazyLocalContext` (or its aliases `lazylocalcontext`, `LazyLocal`, `lazylocal`):

remote_profile:

type: dict, optional

argument path: `run_mdata/train/machine[LazyLocalContext]/remote_profile`

The information used to maintain the connection with remote machine. This field is empty for this context.

When `context_type` is set to `HDFSContext` (or its aliases `hdfscontext`, `HDFS`, `hdfs`):

remote_profile:

type: dict, optional

argument path: `run_mdata/train/machine[HDFSContext]/remote_profile`

The information used to maintain the connection with remote machine. This field is empty for this context.

resources:

type: dict

argument path: `run_mdata/train/resources`

number_node:

type: int, optional, default: 1

argument path:
`run_mdata/train/resources/number_node`

The number of node need for each *job*

cpu_per_node:

type: int, optional, default: 1

argument path:
`run_mdata/train/resources/cpu_per_node`

cpu numbers of each node assigned to each job.

gpu_per_node:

type: int, optional, default: 0

argument path:
`run_mdata/train/resources/gpu_per_node`

gpu numbers of each node assigned to each job.

queue_name:

type: str, optional, default: `****`

argument path:
`run_mdata/train/resources/queue_name`

The queue name of batch job scheduler system.

group_size:

type: int

argument path:

run_mdata/train/resources/group_size

The number of *tasks* in a *job*. 0 means infinity.**custom_flags:**

type: list, optional

argument path:

run_mdata/train/resources/custom_flags

The extra lines pass to job submitting script header

strategy:

type: dict, optional

argument path:

run_mdata/train/resources/strategy

strategies we use to generation job submitting scripts.

if_cuda_multi_devices:

type: bool, optional, default: False

argument path:

run_mdata/train/resources/
strategy/if_cuda_multi_devices

If there are multiple nvidia GPUS on the node, and we want to assign the tasks to different GPUS. If true, dpdispatcher will manually export environment variable CUDA_VISIBLE_DEVICES to different task. Usually, this option will be used with Task.task_need_resources variable simultaneously.

ratio_unfinished:

type: float, optional, default: 0.0

argument path:

run_mdata/train/resources/
strategy/ratio_unfinishedThe ratio of *jobs* that can be unfinished.**para_deg:**

type: int, optional, default: 1

argument path:

run_mdata/train/resources/para_deg

Decide how many tasks will be run in parallel.

source_list:

type: list, optional, default: []

argument path:
`run_mdata/train/resources/source_list`

The env file to be sourced before the command execution.

module_purge:

type: bool, optional, default: False

argument path:
`run_mdata/train/resources/module_purge`

Remove all modules on HPC system before module load (`module_list`)

module_unload_list:

type: list, optional, default: []

argument path: `run_mdata/train/resources/module_unload_list`

The modules to be unloaded on HPC system before submitting jobs

module_list:

type: list, optional, default: []

argument path:
`run_mdata/train/resources/module_list`

The modules to be loaded on HPC system before submitting jobs

envs:

type: dict, optional, default: {}

argument path:
`run_mdata/train/resources/envs`

The environment variables to be exported on before submitting jobs

wait_time:

type: int | float, optional, default: 0

argument path:
`run_mdata/train/resources/wait_time`

The waiting time in second after a single *task* submitted

Depending on the value of *batch_type*, different sub args are accepted.

batch_type:

type: str (flag key)

argument path:
`run_mdata/train/resources/batch_type`

possible choices: *Shell*, *Slurm*, *PBS*, *Torque*,
SlurmJobArray, *DpCloudServer*, *LSF*,
Lebesgue, *DistributedShell*

The batch job system type loaded from machine/batch_type.

When `batch_type` is set to `Shell` (or its alias `shell`):

kwargs:

type: dict, optional
 argument path:
 run_mdata/train/resources[Shell]/kwargs

This field is empty for this batch.

When `batch_type` is set to `Slurm` (or its alias `slurm`):

kwargs:

type: dict, optional
 argument path:
 run_mdata/train/resources[Slurm]/kwargs

Extra arguments.

custom_gpu_line:

type: `NoneType` | `str`, optional, default:
 None

argument path: run_mdata/train/
 resources[Slurm]/kwargs/
 custom_gpu_line

Custom GPU configuration, starting with
 #SBATCH

When `batch_type` is set to `PBS` (or its alias `pbs`):

kwargs:

type: dict, optional
 argument path:
 run_mdata/train/resources[PBS]/kwargs

This field is empty for this batch.

When `batch_type` is set to `Torque` (or its alias `torque`):

kwargs:

type: dict, optional
 argument path: run_mdata/train/
 resources[Torque]/kwargs

This field is empty for this batch.

When `batch_type` is set to `SlurmJobArray` (or its alias
`slurmjobarray`):

kwargs:

type: dict, optional
argument path: `run_mdata/train/
resources[SlurmJobArray]/kwargs`

Extra arguments.

custom_gpu_line:

type: `NoneType` | str, optional, default:
None
argument path: `run_mdata/train/
resources[SlurmJobArray]/kwargs/
custom_gpu_line`
Custom GPU configuration, starting with
#SBATCH

When `batch_type` is set to `DpCloudServer` (or its alias `dpcloudserver`):

kwargs:

type: dict, optional
argument path: `run_mdata/train/
resources[DpCloudServer]/kwargs`

This field is empty for this batch.

When `batch_type` is set to `LSF` (or its alias `lsf`):

kwargs:

type: dict
argument path:
`run_mdata/train/resources[LSF]/kwargs`

Extra arguments.

gpu_usage:

type: bool, optional, default: False
argument path: `run_mdata/train/
resources[LSF]/kwargs/gpu_usage`
Choosing if GPU is used in the calculation
step.

gpu_new_syntax:

type: bool, optional, default: False
argument path:
`run_mdata/train/resources[LSF]/
kwargs/gpu_new_syntax`
For LFS \geq 10.1.0.3, new option `-gpu` for
#BSUB could be used. If False, and old
syntax would be used.

gpu_exclusive:

type: bool, optional, default: True

argument path:
`run_mdata/train/resources[LSF]/
 kwargs/gpu_exclusive`

Only take effect when new syntax enabled.
 Control whether submit tasks in exclusive
 way for GPU.

custom_gpu_line:

type: `NoneType | str`, optional, default:
`None`

argument path:
`run_mdata/train/resources[LSF]/
 kwargs/custom_gpu_line`

Custom GPU configuration, starting with
`#BSUB`

When `batch_type` is set to Lebesgue (or its alias `lebesgue`):

kwargs:

type: `dict`, optional
 argument path: `run_mdata/train/
 resources[Lebesgue]/kwargs`

This field is empty for this batch.

When `batch_type` is set to `DistributedShell` (or its alias
`distributedshell`):

kwargs:

type: `dict`, optional
 argument path: `run_mdata/train/
 resources[DistributedShell]/kwargs`

This field is empty for this batch.

user_forward_files:

type: `list`, optional
 argument path: `run_mdata/train/user_forward_files`

Files to be forwarded to the remote machine.

user_backward_files:

type: `list`, optional
 argument path: `run_mdata/train/user_backward_files`

Files to be backwarded from the remote machine.

model_devi:

type: `dict`
 argument path: `run_mdata/model_devi`
 Parameters of command, machine, and resources for `model_devi`

command:

type: str

argument path: run_mdata/model_devi/command

Command of a program.

machine:

type: dict

argument path: run_mdata/model_devi/machine

batch_type:

type: str

argument path: run_mdata/model_devi/
machine/batch_type

The batch job system type. Option: PBS, Shell, LSF, Lebesgue, Slurm, Torque, SlurmJobArray, DistributedShell, DpCloudServer

local_root:

type: NoneType | str

argument path: run_mdata/model_devi/
machine/local_root

The dir where the tasks and relating files locate.
Typically the project dir.

remote_root:

type: NoneType | str, optional

argument path: run_mdata/model_devi/
machine/remote_root

The dir where the tasks are executed on the remote machine. Only needed when context is not lazy-local.

clean_asynchronously:

type: bool, optional, default: False

argument path: run_mdata/model_devi/
machine/clean_asynchronously

Clean the remote directory asynchronously after the job finishes.

Depending on the value of *context_type*, different sub args are accepted.

context_type:

type: str (flag key)

argument path: run_mdata/model_devi/
machine/context_type

possible choices: *LebesgueContext*,
SSHContext, *LocalContext*,

DpCloudServerContext,
LazyLocalContext, HDFSContext

The connection used to remote machine.
 Option: SSHContext, LazyLocalContext,
 LebesgueContext, LocalContext, DpCloud-
 ServerContext, HDFSContext

When `context_type` is set to `LebesgueContext` (or its aliases
`lebesguecontext`, `Lebesgue`, `lebesgue`):

remote_profile:

type: dict
 argument path: `run_mdata/model_devi/
 machine[LebesgueContext]/
 remote_profile`

The information used to maintain the connec-
 tion with remote machine.

email:

type: str
 argument path:
`run_mdata/model_devi/
 machine[LebesgueContext]/
 remote_profile/email`

Email

password:

type: str
 argument path:
`run_mdata/model_devi/
 machine[LebesgueContext]/
 remote_profile/password`

Password

program_id:

type: int, alias: *project_id*
 argument path:
`run_mdata/model_devi/
 machine[LebesgueContext]/
 remote_profile/program_id`

Program ID

keep_backup:

type: bool, optional
 argument path:
`run_mdata/model_devi/
 machine[LebesgueContext]/
 remote_profile/keep_backup`

keep download and upload zip

input_data:

type: dict

argument path:
run_mdata/model_devi/
machine[LebesgueContext]/
remote_profile/input_data

Configuration of job

When `context_type` is set to `SSHContext` (or its aliases `sshcontext`, `SSH`, `ssh`):

remote_profile:

type: dict
argument path: run_mdata/model_devi/
machine[SSHContext]/remote_profile
The information used to maintain the connection with remote machine.

hostname:

type: str
argument path:
run_mdata/model_devi/
machine[SSHContext]/
remote_profile/hostname
hostname or ip of ssh connection.

username:

type: str
argument path:
run_mdata/model_devi/
machine[SSHContext]/
remote_profile/username
username of target linux system

password:

type: str, optional
argument path:
run_mdata/model_devi/
machine[SSHContext]/
remote_profile/password
(deprecated) password of linux system. Please use [SSH keys](#) instead to improve security.

port:

type: int, optional, default: 22
argument path:
run_mdata/model_devi/
machine[SSHContext]/
remote_profile/port
ssh connection port.

key_filename:

type: `NoneType` | `str`, optional,

default: `None`

argument path:

`run_mdata/model_devi/`

`machine[SSHContext]/`

`remote_profile/key_filename`

key filename used by ssh connection.

If left `None`, find key in `~/.ssh` or use password for login

passphrase:

type: `NoneType` | `str`, optional,

default: `None`

argument path:

`run_mdata/model_devi/`

`machine[SSHContext]/`

`remote_profile/passphrase`

passphrase of key used by ssh connection

timeout:

type: `int`, optional, default: `10`

argument path:

`run_mdata/model_devi/`

`machine[SSHContext]/`

`remote_profile/timeout`

timeout of ssh connection

totp_secret:

type: `NoneType` | `str`, optional,

default: `None`

argument path:

`run_mdata/model_devi/`

`machine[SSHContext]/`

`remote_profile/totp_secret`

Time-based one time password secret. It should be a base32-encoded string extracted from the 2D code.

tar_compress:

type: `bool`, optional, default: `True`

argument path:

`run_mdata/model_devi/`

`machine[SSHContext]/`

`remote_profile/tar_compress`

The archive will be compressed in upload and download if it is `True`. If not, compression will be skipped.

When `context_type` is set to `LocalContext` (or its aliases `localcontext`, `Local`, `local`):

remote_profile:

type: dict, optional
argument path: run_mdata/model_devi/
machine[LocalContext]/
remote_profile

The information used to maintain the connection with remote machine. This field is empty for this context.

When `context_type` is set to `DpCloudServerContext` (or its aliases `dpcloudservercontext`, `DpCloudServer`, `dpcloudserver`):

remote_profile:

type: dict
argument path: run_mdata/model_devi/
machine[DpCloudServerContext]/
remote_profile

The information used to maintain the connection with remote machine.

email:

type: str
argument path:
run_mdata/model_devi/
machine[DpCloudServerContext]/
remote_profile/email

Email

password:

type: str
argument path:
run_mdata/model_devi/
machine[DpCloudServerContext]/
remote_profile/password

Password

program_id:

type: int, alias: *project_id*
argument path:
run_mdata/model_devi/
machine[DpCloudServerContext]/
remote_profile/program_id

Program ID

keep_backup:

type: bool, optional
argument path:
run_mdata/model_devi/
machine[DpCloudServerContext]/
remote_profile/keep_backup

keep download and upload zip

input_data:

type: dict

argument path:

run_mdata/model_devi/

machine[DpCloudServerContext]/

remote_profile/input_data

Configuration of job

When `context_type` is set to `LazyLocalContext` (or its aliases `lazylocalcontext`, `LazyLocal`, `lazylocal`):

remote_profile:

type: dict, optional

argument path: run_mdata/model_devi/

machine[LazyLocalContext]/

remote_profile

The information used to maintain the connection with remote machine. This field is empty for this context.

When `context_type` is set to `HDFSContext` (or its aliases `hdfscontext`, `HDFS`, `hdfs`):

remote_profile:

type: dict, optional

argument path: run_mdata/model_devi/

machine[HDFSContext]/remote_profile

The information used to maintain the connection with remote machine. This field is empty for this context.

resources:

type: dict

argument path: run_mdata/model_devi/resources

number_node:

type: int, optional, default: 1

argument path: run_mdata/model_devi/

resources/number_node

The number of node need for each *job*

cpu_per_node:

type: int, optional, default: 1

argument path: run_mdata/model_devi/

resources/cpu_per_node

cpu numbers of each node assigned to each job.

gpu_per_node:

type: int, optional, default: 0

argument path: run_mdata/model_devi/
resources/gpu_per_node

gpu numbers of each node assigned to each job.

queue_name:

type: str, optional, default: ""

argument path: run_mdata/model_devi/
resources/queue_name

The queue name of batch job scheduler system.

group_size:

type: int

argument path: run_mdata/model_devi/
resources/group_size

The number of *tasks* in a *job*. 0 means infinity.

custom_flags:

type: list, optional

argument path: run_mdata/model_devi/
resources/custom_flags

The extra lines pass to job submitting script
header

strategy:

type: dict, optional

argument path: run_mdata/model_devi/
resources/strategy

strategies we use to generation job submitting
scripts.

if_cuda_multi_devices:

type: bool, optional, default: False

argument path:
run_mdata/model_devi/
resources/strategy/
if_cuda_multi_devices

If there are multiple nvidia GPUS
on the node, and we want to as-
sign the tasks to different GPUS.If
true, dpdispatcher will manu-
ally export environment variable
CUDA_VISIBLE_DEVICES to dif-
ferent task.Usually, this option will be
used with Task.task_need_resources
variable simultaneously.

ratio_unfinished:

type: float, optional, default: 0.0

argument path: run_mdata/
model_devi/resources/
strategy/ratio_unfinished

The ratio of *jobs* that can be unfinished.

para_deg:

type: int, optional, default: 1

argument path: run_mdata/model_devi/
resources/para_deg

Decide how many tasks will be run in parallel.

source_list:

type: list, optional, default: []

argument path: run_mdata/model_devi/
resources/source_list

The env file to be sourced before the command execution.

module_purge:

type: bool, optional, default: False

argument path: run_mdata/model_devi/
resources/module_purge

Remove all modules on HPC system before module load (module_list)

module_unload_list:

type: list, optional, default: []

argument path: run_mdata/model_devi/
resources/module_unload_list

The modules to be unloaded on HPC system before submitting jobs

module_list:

type: list, optional, default: []

argument path: run_mdata/model_devi/
resources/module_list

The modules to be loaded on HPC system before submitting jobs

envs:

type: dict, optional, default: {}

argument path:
run_mdata/model_devi/resources/envs

The environment variables to be exported on before submitting jobs

wait_time:

type: int | float, optional, default: 0
argument path: run_mdata/model_devi/
resources/wait_time

The waiting time in second after a single *task* submitted

Depending on the value of *batch_type*, different sub args are accepted.

batch_type:

type: str (flag key)
argument path: run_mdata/model_devi/
resources/batch_type
possible choices: *Shell*, *Slurm*, *PBS*,
Torque, *SlurmJobArray*, *DpCloudServer*,
LSF, *Lebesgue*, *DistributedShell*

The batch job system type loaded from machine/batch_type.

When *batch_type* is set to *Shell* (or its alias *shell*):

kwargs:

type: dict, optional
argument path: run_mdata/model_devi/
resources[Shell]/kwargs

This field is empty for this batch.

When *batch_type* is set to *Slurm* (or its alias *slurm*):

kwargs:

type: dict, optional
argument path: run_mdata/model_devi/
resources[Slurm]/kwargs

Extra arguments.

custom_gpu_line:

type: NoneType | str, optional,
default: None
argument path: run_mdata/
model_devi/resources[Slurm]/
kwargs/custom_gpu_line

Custom GPU configuration, starting
with #SBATCH

When *batch_type* is set to *PBS* (or its alias *pbs*):

kwargs:

type: dict, optional

argument path: `run_mdata/model_devi/
resources[PBS]/kwargs`

This field is empty for this batch.

When `batch_type` is set to Torque (or its alias `torque`):

kwargs:

type: dict, optional
argument path: `run_mdata/model_devi/
resources[Torque]/kwargs`

This field is empty for this batch.

When `batch_type` is set to SlurmJobArray (or its alias `slurmjobarray`):

kwargs:

type: dict, optional
argument path: `run_mdata/model_devi/
resources[SlurmJobArray]/kwargs`

Extra arguments.

custom_gpu_line:

type: `NoneType | str`, optional,
default: `None`

argument path:
`run_mdata/model_devi/
resources[SlurmJobArray]/
kwargs/custom_gpu_line`

Custom GPU configuration, starting
with `#SBATCH`

When `batch_type` is set to DpCloudServer (or its alias `dpcloudserver`):

kwargs:

type: dict, optional
argument path: `run_mdata/model_devi/
resources[DpCloudServer]/kwargs`

This field is empty for this batch.

When `batch_type` is set to LSF (or its alias `lsf`):

kwargs:

type: dict
argument path: `run_mdata/model_devi/
resources[LSF]/kwargs`

Extra arguments.

gpu_usage:

type: bool, optional, default: `False`

argument path: `run_mdata/
model_devi/resources[LSF]/
kwargs/gpu_usage`

Choosing if GPU is used in the calculation step.

gpu_new_syntax:

type: bool, optional, default: False
argument path: `run_mdata/
model_devi/resources[LSF]/
kwargs/gpu_new_syntax`

For LFS \geq 10.1.0.3, new option -gpu for #BSUB could be used. If False, and old syntax would be used.

gpu_exclusive:

type: bool, optional, default: True
argument path: `run_mdata/
model_devi/resources[LSF]/
kwargs/gpu_exclusive`

Only take effect when new syntax enabled. Control whether submit tasks in exclusive way for GPU.

custom_gpu_line:

type: NoneType | str, optional,
default: None
argument path: `run_mdata/
model_devi/resources[LSF]/
kwargs/custom_gpu_line`

Custom GPU configuration, starting with #BSUB

When `batch_type` is set to Lebesgue (or its alias `lebesgue`):

kwargs:

type: dict, optional
argument path: `run_mdata/model_devi/
resources[Lebesgue]/kwargs`

This field is empty for this batch.

When `batch_type` is set to DistributedShell (or its alias `distributedshell`):

kwargs:

type: dict, optional
argument path: `run_mdata/model_devi/
resources[DistributedShell]/kwargs`

This field is empty for this batch.

user_forward_files:

type: list, optional
 argument path:
 run_mdata/model_devi/user_forward_files
 Files to be forwarded to the remote machine.

user_backward_files:

type: list, optional
 argument path:
 run_mdata/model_devi/user_backward_files
 Files to be backwarded from the remote machine.

fp:

type: dict
 argument path: run_mdata/fp
 Parameters of command, machine, and resources for fp

command:

type: str
 argument path: run_mdata/fp/command
 Command of a program.

machine:

type: dict
 argument path: run_mdata/fp/machine

batch_type:

type: str
 argument path:
 run_mdata/fp/machine/batch_type
 The batch job system type. Option: PBS, Shell, LSF, Lebesgue, Slurm, Torque, SlurmJobArray, DistributedShell, DpCloudServer

local_root:

type: NoneType | str
 argument path:
 run_mdata/fp/machine/local_root
 The dir where the tasks and relating files locate.
 Typically the project dir.

remote_root:

type: NoneType | str, optional
 argument path:
 run_mdata/fp/machine/remote_root
 The dir where the tasks are executed on the remote machine. Only needed when context is not lazy-local.

clean_asynchronously:

type: bool, optional, default: False

argument path: run_mdata/fp/machine/
clean_asynchronously

Clean the remote directory asynchronously after the job finishes.

Depending on the value of *context_type*, different sub args are accepted.

context_type:

type: str (flag key)

argument path:

run_mdata/fp/machine/context_type

possible choices: *LebesgueContext*,
SSHContext, *LocalContext*,
DpCloudServerContext,
LazyLocalContext, *HDFSContext*

The connection used to remote machine.

Option: *SSHContext*, *LazyLocalContext*,
LebesgueContext, *LocalContext*, *DpCloud-*
ServerContext, *HDFSContext*

When *context_type* is set to *LebesgueContext* (or its aliases *lebesguecontext*, *Lebesgue*, *lebesgue*):

remote_profile:

type: dict

argument path: run_mdata/fp/
machine[*LebesgueContext*]/
remote_profile

The information used to maintain the connection with remote machine.

email:

type: str

argument path: run_mdata/fp/
machine[*LebesgueContext*]/
remote_profile/email

Email

password:

type: str

argument path: run_mdata/fp/
machine[*LebesgueContext*]/
remote_profile/password

Password

program_id:

type: int, alias: *project_id*

argument path: `run_mdata/fp/
machine[LebesgueContext]/
remote_profile/program_id`

Program ID

keep_backup:

type: bool, optional

argument path: `run_mdata/fp/
machine[LebesgueContext]/
remote_profile/keep_backup`

keep download and upload zip

input_data:

type: dict

argument path: `run_mdata/fp/
machine[LebesgueContext]/
remote_profile/input_data`

Configuration of job

When `context_type` is set to `SSHContext` (or its aliases `sshcontext`, `SSH`, `ssh`):

remote_profile:

type: dict

argument path: `run_mdata/fp/
machine[SSHContext]/remote_profile`

The information used to maintain the connection with remote machine.

hostname:

type: str

argument path: `run_mdata/fp/
machine[SSHContext]/
remote_profile/hostname`

hostname or ip of ssh connection.

username:

type: str

argument path: `run_mdata/fp/
machine[SSHContext]/
remote_profile/username`

username of target linux system

password:

type: str, optional

argument path: `run_mdata/fp/
machine[SSHContext]/
remote_profile/password`

(deprecated) password of linux system. Please use [SSH keys](#) instead to improve security.

port:

type: int, optional, default: 22
argument path: run_mdata/fp/
machine[SSHContext]/
remote_profile/port

ssh connection port.

key_filename:

type: NoneType | str, optional,
default: None
argument path: run_mdata/fp/
machine[SSHContext]/
remote_profile/key_filename

key filename used by ssh connection.
If left None, find key in ~/.ssh or use
password for login

passphrase:

type: NoneType | str, optional,
default: None
argument path: run_mdata/fp/
machine[SSHContext]/
remote_profile/passphrase

passphrase of key used by ssh connection

timeout:

type: int, optional, default: 10
argument path: run_mdata/fp/
machine[SSHContext]/
remote_profile/timeout

timeout of ssh connection

totp_secret:

type: NoneType | str, optional,
default: None
argument path: run_mdata/fp/
machine[SSHContext]/
remote_profile/totp_secret

Time-based one time password secret. It should be a base32-encoded string extracted from the 2D code.

tar_compress:

type: bool, optional, default: True
argument path: run_mdata/fp/
machine[SSHContext]/
remote_profile/tar_compress

The archive will be compressed in upload and download if it is True. If not, compression will be skipped.

When `context_type` is set to `LocalContext` (or its aliases `localcontext`, `Local`, `local`):

remote_profile:

type: dict, optional
 argument path: `run_mdata/fp/machine[LocalContext]/remote_profile`

The information used to maintain the connection with remote machine. This field is empty for this context.

When `context_type` is set to `DpCloudServerContext` (or its aliases `dpcloudservercontext`, `DpCloudServer`, `dpcloudserver`):

remote_profile:

type: dict
 argument path: `run_mdata/fp/machine[DpCloudServerContext]/remote_profile`

The information used to maintain the connection with remote machine.

email:

type: str
 argument path: `run_mdata/fp/machine[DpCloudServerContext]/remote_profile/email`

Email

password:

type: str
 argument path: `run_mdata/fp/machine[DpCloudServerContext]/remote_profile/password`

Password

program_id:

type: int, alias: *project_id*
 argument path: `run_mdata/fp/machine[DpCloudServerContext]/remote_profile/program_id`

Program ID

keep_backup:

type: bool, optional
 argument path: `run_mdata/fp/machine[DpCloudServerContext]/remote_profile/keep_backup`

keep download and upload zip

input_data:

type: dict
argument path: run_mdata/fp/
machine[DpCloudServerContext]/
remote_profile/input_data

Configuration of job

When `context_type` is set to `LazyLocalContext` (or its aliases `lazylocalcontext`, `LazyLocal`, `lazylocal`):

remote_profile:

type: dict, optional
argument path: run_mdata/fp/
machine[LazyLocalContext]/
remote_profile

The information used to maintain the connection with remote machine. This field is empty for this context.

When `context_type` is set to `HDFSContext` (or its aliases `hdfscontext`, `HDFS`, `hdfs`):

remote_profile:

type: dict, optional
argument path: run_mdata/fp/
machine[HDFSContext]/remote_profile

The information used to maintain the connection with remote machine. This field is empty for this context.

resources:

type: dict
argument path: run_mdata/fp/resources

number_node:

type: int, optional, default: 1
argument path:
run_mdata/fp/resources/number_node

The number of node need for each *job*

cpu_per_node:

type: int, optional, default: 1
argument path:
run_mdata/fp/resources/cpu_per_node

cpu numbers of each node assigned to each job.

gpu_per_node:

type: int, optional, default: 0

argument path:
`run_mdata/fp/resources/gpu_per_node`
 gpu numbers of each node assigned to each job.

queue_name:

type: str, optional, default: ""
 argument path:
`run_mdata/fp/resources/queue_name`
 The queue name of batch job scheduler system.

group_size:

type: int
 argument path:
`run_mdata/fp/resources/group_size`
 The number of *tasks* in a *job*. 0 means infinity.

custom_flags:

type: list, optional
 argument path:
`run_mdata/fp/resources/custom_flags`
 The extra lines pass to job submitting script header

strategy:

type: dict, optional
 argument path:
`run_mdata/fp/resources/strategy`
 strategies we use to generation job submitting scripts.

if_cuda_multi_devices:

type: bool, optional, default: False
 argument path: `run_mdata/fp/resources/strategy/if_cuda_multi_devices`

If there are multiple nvidia GPUS on the node, and we want to assign the tasks to different GPUS. If true, dpdispatcher will manually export environment variable `CUDA_VISIBLE_DEVICES` to different task. Usually, this option will be used with `Task.task_need_resources` variable simultaneously.

ratio_unfinished:

type: float, optional, default: 0.0
 argument path:
`run_mdata/fp/resources/strategy/ratio_unfinished`

The ratio of *jobs* that can be unfinished.

para_deg:

type: int, optional, default: 1

argument path:

run_mdata/fp/resources/para_deg

Decide how many tasks will be run in parallel.

source_list:

type: list, optional, default: []

argument path:

run_mdata/fp/resources/source_list

The env file to be sourced before the command execution.

module_purge:

type: bool, optional, default: False

argument path:

run_mdata/fp/resources/module_purge

Remove all modules on HPC system before module load (module_list)

module_unload_list:

type: list, optional, default: []

argument path: run_mdata/fp/resources/
module_unload_list

The modules to be unloaded on HPC system before submitting jobs

module_list:

type: list, optional, default: []

argument path:

run_mdata/fp/resources/module_list

The modules to be loaded on HPC system before submitting jobs

envs:

type: dict, optional, default: {}

argument path:

run_mdata/fp/resources/envs

The environment variables to be exported on before submitting jobs

wait_time:

type: int | float, optional, default: 0

argument path:
`run_mdata/fp/resources/wait_time`

The waiting time in second after a single *task* submitted

Depending on the value of *batch_type*, different sub args are accepted.

batch_type:

type: str (flag key)
 argument path:
`run_mdata/fp/resources/batch_type`
 possible choices: *Shell*, *Slurm*, *PBS*,
Torque, *SlurmJobArray*, *DpCloudServer*,
LSF, *Lebesgue*, *DistributedShell*
 The batch job system type loaded from machine/*batch_type*.

When *batch_type* is set to *Shell* (or its alias *shell*):

kwargs:

type: dict, optional
 argument path: `run_mdata/fp/resources[Shell]/kwargs`
 This field is empty for this batch.

When *batch_type* is set to *Slurm* (or its alias *slurm*):

kwargs:

type: dict, optional
 argument path: `run_mdata/fp/resources[Slurm]/kwargs`
 Extra arguments.

custom_gpu_line:

type: `NoneType` | str, optional,
 default: `None`
 argument path: `run_mdata/fp/resources[Slurm]/kwargs/custom_gpu_line`
 Custom GPU configuration, starting with `#SBATCH`

When *batch_type* is set to *PBS* (or its alias *pbs*):

kwargs:

type: dict, optional
 argument path:
`run_mdata/fp/resources[PBS]/kwargs`
 This field is empty for this batch.

When *batch_type* is set to *Torque* (or its alias *torque*):

kwargs:

type: dict, optional
argument path: run_mdata/fp/
resources[Torque]/kwargs

This field is empty for this batch.

When `batch_type` is set to `SlurmJobArray` (or its alias `slurmjobarray`):

kwargs:

type: dict, optional
argument path: run_mdata/fp/
resources[SlurmJobArray]/kwargs

Extra arguments.

custom_gpu_line:

type: NoneType | str, optional,
default: None
argument path: run_mdata/fp/
resources[SlurmJobArray]/
kwargs/custom_gpu_line
Custom GPU configuration, starting
with #SBATCH

When `batch_type` is set to `DpCloudServer` (or its alias `dpcloudserver`):

kwargs:

type: dict, optional
argument path: run_mdata/fp/
resources[DpCloudServer]/kwargs

This field is empty for this batch.

When `batch_type` is set to `LSF` (or its alias `lsf`):

kwargs:

type: dict
argument path:
run_mdata/fp/resources[LSF]/kwargs

Extra arguments.

gpu_usage:

type: bool, optional, default: False
argument path:
run_mdata/fp/resources[LSF]/
kwargs/gpu_usage

Choosing if GPU is used in the calculation step.

gpu_new_syntax:

type: bool, optional, default: False
 argument path:
 run_mdata/fp/resources[LSF]/
 kwargs/gpu_new_syntax

For LFS >= 10.1.0.3, new option -gpu
 for #BSUB could be used. If False,
 and old syntax would be used.

gpu_exclusive:

type: bool, optional, default: True
 argument path:
 run_mdata/fp/resources[LSF]/
 kwargs/gpu_exclusive

Only take effect when new syntax en-
 abled. Control whether submit tasks
 in exclusive way for GPU.

custom_gpu_line:

type: NoneType | str, optional,
 default: None
 argument path:
 run_mdata/fp/resources[LSF]/
 kwargs/custom_gpu_line

Custom GPU configuration, starting
 with #BSUB

When `batch_type` is set to Lebesgue (or its alias lebesgue):

kwargs:

type: dict, optional
 argument path: run_mdata/fp/
 resources[Lebesgue]/kwargs

This field is empty for this batch.

When `batch_type` is set to DistributedShell (or its alias distributedshell):

kwargs:

type: dict, optional
 argument path: run_mdata/fp/
 resources[DistributedShell]/kwargs

This field is empty for this batch.

user_forward_files:

type: list, optional
 argument path: run_mdata/fp/user_forward_files

Files to be forwarded to the remote machine.

user_backward_files:

type: list, optional

argument path: run_mdata/fp/user_backward_files

Files to be backwarded from the remote machine.

5.1 Init_bulk

You may prepare initial data for bulk systems with VASP by:

```
dpngen init_bulk PARAM [MACHINE]
```

The MACHINE configure file is optional. If this parameter exists, then the optimization tasks or MD tasks will be submitted automatically according to MACHINE.json.

Basically init_bulk can be divided into four parts , denoted as stages in PARAM:

1. Relax in folder 00.place_ele
2. Perturb and scale in folder 01.scale_pert
3. Run a short AIMD in folder 02.md
4. Collect data in folder 02.md.

All stages must be **in order**. One doesn't need to run all stages. For example, you may run stage 1 and 2, generating supercells as starting point of exploration in dpngen run.

If MACHINE is None, there should be only one stage in stages. Corresponding tasks will be generated, but user's intervention should be involved in, to manually run the scripts.

Following is an example for PARAM, which generates data from a typical structure hcp.

```
{
  "stages" : [1,2,3,4],
  "cell_type": "hcp",
  "latt": 4.479,
  "super_cell": [2, 2, 2],
  "elements": ["Mg"],
  "potcars": ["...../POTCAR"],
  "relax_incar": "...../INCAR_metal_rlx",
  "md_incar" : "...../INCAR_metal_md",
  "scale": [1.00],
  "skip_relax": false,
  "pert_numb": 2,
  "md_nstep" : 5,
  "pert_box": 0.03,
  "pert_atom": 0.01,
  "coll_ndata": 5000,
  "type_map" : [ "Mg", "Al"],
```

(continues on next page)

(continued from previous page)

```

    "_comment":      "that's all"
}

```

If you want to specify a structure as starting point for `init_bulk`, you may set in `PARAM` as follows.

```

"from_poscar":      true,
"from_poscar_path": "...../C_mp-47_conventional.POSCAR",

```

`init_bulk` supports both VASP and ABACUS for first-principle calculation. You can choose the software by specifying the key `init_fp_style`. If `init_fp_style` is not specified, the default software will be VASP.

When using ABACUS for `init_fp_style`, the keys of the paths of INPUT files for relaxation and MD simulations are the same as INCAR for VASP, which are `relax_incar` and `md_incar` respectively. Use `relax_kpt` and `md_kpt` for the relative path for KPT files of relaxation and MD simulations. They two can be omitted if `kspacing` (in unit of 1/Bohr) or `gamma_only` has been set in corresponding INPUT files. If `from_poscar` is set to `false`, you have to specify `atom_masses` in the same order as `elements`.

5.2 dpngen init_bulk machine parameters

`init_bulk_mdata`:

type: dict

argument path: `init_bulk_mdata`

machine.json file

`api_version`:

type: str

argument path: `init_bulk_mdata/api_version`

Please set to 1.0

`deepmd_version`:

type: str, optional, default: 2

argument path: `init_bulk_mdata/deepmd_version`

DeePMD-kit version, e.g. 2.1.3

`fp`:

type: dict

argument path: `init_bulk_mdata/fp`

Parameters of command, machine, and resources for fp

`command`:

type: str

argument path: `init_bulk_mdata/fp/command`

Command of a program.

`machine`:

type: dict

argument path: `init_bulk_mdata/fp/machine`

batch_type:

type: str

argument path: init_bulk_mdata/fp/
machine/batch_type

The batch job system type. Option: PBS, Shell, LSF, Lebesgue, Slurm, Torque, SlurmJobArray, DistributedShell, DpCloudServer

local_root:

type: NoneType | str

argument path: init_bulk_mdata/fp/
machine/local_root

The dir where the tasks and relating files locate. Typically the project dir.

remote_root:

type: NoneType | str, optional

argument path: init_bulk_mdata/fp/
machine/remote_root

The dir where the tasks are executed on the remote machine. Only needed when context is not lazy-local.

clean_asynchronously:

type: bool, optional, default: False

argument path: init_bulk_mdata/fp/
machine/clean_asynchronously

Clean the remote directory asynchronously after the job finishes.

Depending on the value of *context_type*, different sub args are accepted.

context_type:

type: str (flag key)

argument path: init_bulk_mdata/fp/
machine/context_type

possible choices: *LebesgueContext*,
SSHContext, *LocalContext*,
DpCloudServerContext,
LazyLocalContext, *HDFSContext*

The connection used to remote machine. Option: SSHContext, LazyLocalContext, LebesgueContext, LocalContext, DpCloudServerContext, HDFSContext

When *context_type* is set to *LebesgueContext* (or its aliases *lebesguecontext*, *Lebesgue*, *lebesgue*):

remote_profile:

type: dict
argument path: init_bulk_mdata/fp/
machine[LebesgueContext]/
remote_profile

The information used to maintain the connection with remote machine.

email:

type: str
argument path: init_bulk_mdata/
fp/machine[LebesgueContext]/
remote_profile/email

Email

password:

type: str
argument path: init_bulk_mdata/
fp/machine[LebesgueContext]/
remote_profile/password

Password

program_id:

type: int, alias: *project_id*
argument path: init_bulk_mdata/
fp/machine[LebesgueContext]/
remote_profile/program_id

Program ID

keep_backup:

type: bool, optional
argument path: init_bulk_mdata/
fp/machine[LebesgueContext]/
remote_profile/keep_backup

keep download and upload zip

input_data:

type: dict
argument path: init_bulk_mdata/
fp/machine[LebesgueContext]/
remote_profile/input_data

Configuration of job

When `context_type` is set to `SSHContext` (or its aliases `sshcontext`, `SSH`, `ssh`):

remote_profile:

type: dict
argument path: init_bulk_mdata/fp/
machine[SSHContext]/remote_profile

The information used to maintain the connection with remote machine.

hostname:

type: str
 argument path: `init_bulk_mdata/
 fp/machine[SSHContext]/
 remote_profile/hostname`
 hostname or ip of ssh connection.

username:

type: str
 argument path: `init_bulk_mdata/
 fp/machine[SSHContext]/
 remote_profile/username`
 username of target linux system

password:

type: str, optional
 argument path: `init_bulk_mdata/
 fp/machine[SSHContext]/
 remote_profile/password`
 (deprecated) password of linux system. Please use [SSH keys](#) instead to improve security.

port:

type: int, optional, default: 22
 argument path: `init_bulk_mdata/
 fp/machine[SSHContext]/
 remote_profile/port`
 ssh connection port.

key_filename:

type: `NoneType` | str, optional,
 default: None
 argument path: `init_bulk_mdata/
 fp/machine[SSHContext]/
 remote_profile/key_filename`
 key filename used by ssh connection.
 If left None, find key in `~/.ssh` or use
 password for login

passphrase:

type: `NoneType` | str, optional,
 default: None
 argument path: `init_bulk_mdata/
 fp/machine[SSHContext]/
 remote_profile/passphrase`
 passphrase of key used by ssh connection

timeout:

type: int, optional, default: 10
argument path: `init_bulk_mdata/
fp/machine[SSHContext]/
remote_profile/timeout`
timeout of ssh connection

totp_secret:

type: `NoneType` | str, optional,
default: None
argument path: `init_bulk_mdata/
fp/machine[SSHContext]/
remote_profile/totp_secret`
Time-based one time password secret. It should be a base32-encoded string extracted from the 2D code.

tar_compress:

type: bool, optional, default: True
argument path: `init_bulk_mdata/
fp/machine[SSHContext]/
remote_profile/tar_compress`
The archive will be compressed in upload and download if it is True. If not, compression will be skipped.

When `context_type` is set to `LocalContext` (or its aliases `localcontext`, `Local`, `local`):

remote_profile:

type: dict, optional
argument path: `init_bulk_mdata/fp/
machine[LocalContext]/
remote_profile`

The information used to maintain the connection with remote machine. This field is empty for this context.

When `context_type` is set to `DpCloudServerContext` (or its aliases `dpcloudservercontext`, `DpCloudServer`, `dpcloudserver`):

remote_profile:

type: dict
argument path: `init_bulk_mdata/fp/
machine[DpCloudServerContext]/
remote_profile`

The information used to maintain the connection with remote machine.

email:

type: str

argument path:
 init_bulk_mdata/fp/
 machine[DpCloudServerContext]/
 remote_profile/email

Email

password:

type: str
 argument path:
 init_bulk_mdata/fp/
 machine[DpCloudServerContext]/
 remote_profile/password

Password

program_id:

type: int, alias: *project_id*
 argument path:
 init_bulk_mdata/fp/
 machine[DpCloudServerContext]/
 remote_profile/program_id

Program ID

keep_backup:

type: bool, optional
 argument path:
 init_bulk_mdata/fp/
 machine[DpCloudServerContext]/
 remote_profile/keep_backup

keep download and upload zip

input_data:

type: dict
 argument path:
 init_bulk_mdata/fp/
 machine[DpCloudServerContext]/
 remote_profile/input_data

Configuration of job

When `context_type` is set to `LazyLocalContext` (or its aliases `lazylocalcontext`, `LazyLocal`, `lazylocal`):

remote_profile:

type: dict, optional
 argument path: init_bulk_mdata/fp/
 machine[LazyLocalContext]/
 remote_profile

The information used to maintain the connection with remote machine. This field is empty for this context.

When `context_type` is set to `HDFSContext` (or its aliases `hdfscontext`, `HDFS`, `hdfs`):

remote_profile:

type: dict, optional
argument path: `init_bulk_mdata/fp/
machine[HDFSContext]/remote_profile`

The information used to maintain the connection with remote machine. This field is empty for this context.

resources:

type: dict
argument path: `init_bulk_mdata/fp/resources`

number_node:

type: int, optional, default: 1
argument path: `init_bulk_mdata/fp/
resources/number_node`

The number of node need for each *job*

cpu_per_node:

type: int, optional, default: 1
argument path: `init_bulk_mdata/fp/
resources/cpu_per_node`

cpu numbers of each node assigned to each job.

gpu_per_node:

type: int, optional, default: 0
argument path: `init_bulk_mdata/fp/
resources/gpu_per_node`

gpu numbers of each node assigned to each job.

queue_name:

type: str, optional, default: `----`
argument path: `init_bulk_mdata/fp/
resources/queue_name`

The queue name of batch job scheduler system.

group_size:

type: int
argument path: `init_bulk_mdata/fp/
resources/group_size`

The number of *tasks* in a *job*. 0 means infinity.

custom_flags:

type: list, optional
argument path: `init_bulk_mdata/fp/
resources/custom_flags`

The extra lines pass to job submitting script header

strategy:

type: dict, optional

argument path: `init_bulk_mdata/fp/resources/strategy`

strategies we use to generation job submitting scripts.

if_cuda_multi_devices:

type: bool, optional, default: False

argument path: `init_bulk_mdata/fp/resources/strategy/if_cuda_multi_devices`

If there are multiple nvidia GPUS on the node, and we want to assign the tasks to different GPUS. If true, `dpdispatcher` will manually export environment variable `CUDA_VISIBLE_DEVICES` to different task. Usually, this option will be used with `Task.task_need_resources` variable simultaneously.

ratio_unfinished:

type: float, optional, default: 0.0

argument path: `init_bulk_mdata/fp/resources/strategy/ratio_unfinished`

The ratio of *jobs* that can be unfinished.

para_deg:

type: int, optional, default: 1

argument path: `init_bulk_mdata/fp/resources/para_deg`

Decide how many tasks will be run in parallel.

source_list:

type: list, optional, default: []

argument path: `init_bulk_mdata/fp/resources/source_list`

The env file to be sourced before the command execution.

module_purge:

type: bool, optional, default: False

argument path: `init_bulk_mdata/fp/resources/module_purge`

Remove all modules on HPC system before module load (module_list)

module_unload_list:

type: list, optional, default: []
argument path: init_bulk_mdata/fp/
resources/module_unload_list

The modules to be unloaded on HPC system before submitting jobs

module_list:

type: list, optional, default: []
argument path: init_bulk_mdata/fp/
resources/module_list

The modules to be loaded on HPC system before submitting jobs

envs:

type: dict, optional, default: {}
argument path:
init_bulk_mdata/fp/resources/envs

The environment variables to be exported on before submitting jobs

wait_time:

type: int | float, optional, default: 0
argument path: init_bulk_mdata/fp/
resources/wait_time

The waiting time in second after a single *task* submitted

Depending on the value of *batch_type*, different sub args are accepted.

batch_type:

type: str (flag key)
argument path: init_bulk_mdata/fp/
resources/batch_type
possible choices: *Shell*, *Slurm*, *PBS*,
Torque, *SlurmJobArray*, *DpCloudServer*,
LSF, *Lebesgue*, *DistributedShell*

The batch job system type loaded from machine/batch_type.

When *batch_type* is set to *Shell* (or its alias *shell*):

kwargs:

type: dict, optional

argument path: `init_bulk_mdata/fp/resources[Shell]/kwargs`

This field is empty for this batch.

When `batch_type` is set to Slurm (or its alias `slurm`):

kwargs:

type: dict, optional

argument path: `init_bulk_mdata/fp/resources[Slurm]/kwargs`

Extra arguments.

custom_gpu_line:

type: `NoneType` | str, optional,
default: None

argument path: `init_bulk_mdata/fp/resources[Slurm]/kwargs/custom_gpu_line`

Custom GPU configuration, starting with `#SBATCH`

When `batch_type` is set to PBS (or its alias `pbs`):

kwargs:

type: dict, optional

argument path: `init_bulk_mdata/fp/resources[PBS]/kwargs`

This field is empty for this batch.

When `batch_type` is set to Torque (or its alias `torque`):

kwargs:

type: dict, optional

argument path: `init_bulk_mdata/fp/resources[Torque]/kwargs`

This field is empty for this batch.

When `batch_type` is set to SlurmJobArray (or its alias `slurmjobarray`):

kwargs:

type: dict, optional

argument path: `init_bulk_mdata/fp/resources[SlurmJobArray]/kwargs`

Extra arguments.

custom_gpu_line:

type: `NoneType` | str, optional,
default: None

argument path: `init_bulk_mdata/fp/resources[SlurmJobArray]/kwargs/custom_gpu_line`

Custom GPU configuration, starting
with #SBATCH

When `batch_type` is set to `DpCloudServer` (or its alias `dpcLOUDserver`):

kwargs:

type: dict, optional
argument path: `init_bulk_mdata/fp/
resources[DpCloudServer]/kwargs`

This field is empty for this batch.

When `batch_type` is set to `LSF` (or its alias `lsf`):

kwargs:

type: dict
argument path: `init_bulk_mdata/fp/
resources[LSF]/kwargs`

Extra arguments.

gpu_usage:

type: bool, optional, default: False
argument path: `init_bulk_mdata/
fp/resources[LSF]/kwargs/
gpu_usage`

Choosing if GPU is used in the calculation step.

gpu_new_syntax:

type: bool, optional, default: False
argument path: `init_bulk_mdata/
fp/resources[LSF]/kwargs/
gpu_new_syntax`

For LFS >= 10.1.0.3, new option `-gpu` for `#BSUB` could be used. If False, and old syntax would be used.

gpu_exclusive:

type: bool, optional, default: True
argument path: `init_bulk_mdata/
fp/resources[LSF]/kwargs/
gpu_exclusive`

Only take effect when new syntax enabled. Control whether submit tasks in exclusive way for GPU.

custom_gpu_line:

type: NoneType | str, optional,
default: None
argument path: `init_bulk_mdata/
fp/resources[LSF]/kwargs/
custom_gpu_line`

Custom GPU configuration, starting
with #BSUB

When `batch_type` is set to Lebesgue (or its alias `lebesgue`):

kwargs:

type: dict, optional
argument path: `init_bulk_mdata/fp/
resources[Lebesgue]/kwargs`

This field is empty for this batch.

When `batch_type` is set to DistributedShell (or its alias
`distributedshell`):

kwargs:

type: dict, optional
argument path: `init_bulk_mdata/fp/
resources[DistributedShell]/kwargs`

This field is empty for this batch.

user_forward_files:

type: list, optional
argument path:
`init_bulk_mdata/fp/user_forward_files`

Files to be forwarded to the remote machine.

user_backward_files:

type: list, optional
argument path:
`init_bulk_mdata/fp/user_backward_files`

Files to be backwarded from the remote machine.

5.3 Init_surf

You may prepare initial data for surface systems with VASP by:

```
dpngen init_surf PARAM [MACHINE]
```

The MACHINE configure file is optional. If this parameter exists, then the optimization tasks or MD tasks will be submitted automatically according to MACHINE.json. That is to say, if one only wants to prepare `surf-xxx/sys-xxx` folders for the second stage but wants to skip relaxation, `dpngen init_surf PARAM` should be used (without MACHINE). “stages” and “skip_relax” in PARAM should be set as:

```
"stages": [1,2],  
"skip_relax": true,
```

Basically `init_surf` can be divided into two parts , denoted as `stages` in PARAM:

1. Build specific surface in folder `00.place_ele`
2. Pertub and scale in folder `01.scale_pert`

All stages must be **in order**.

Following is an example for **PARAM**, which generates data from a typical structure **fcc**.

```
{
  "stages": [
    1,
    2
  ],
  "cell_type": "fcc",
  "latt": 4.034,
  "super_cell": [
    2,
    2,
    2
  ],
  "layer_numb": 3,
  "vacuum_max": 9,
  "vacuum_resol": [
    0.5,
    1
  ],
  "mid_point": 4.0,
  "millers": [
    [
      1,
      0,
      0
    ],
    [
      1,
      1,
      0
    ],
    [
      1,
      1,
      1
    ]
  ],
  "elements": [
    "Al"
  ],
  "potcars": [
    "...../POTCAR"
  ],
  "relax_incar": "...../INCAR_metal_rlx_low",
  "scale": [
    1.0
  ],
  "skip_relax": true,
  "pert_numb": 2,
  "pert_box": 0.03,
  "pert_atom": 0.01,
}
```

(continues on next page)

(continued from previous page)

```

    "_comment": "that's all"
}

```

Another example is `from_poscar` method. Here you need to specify the POSCAR file.

```

{
  "stages": [
    1,
    2
  ],
  "cell_type": "fcc",
  "from_poscar": true,
  "from_poscar_path": "POSCAR",
  "super_cell": [
    1,
    1,
    1
  ],
  "layer_numb": 3,
  "vacuum_max": 5,
  "vacuum_resol": [0.5, 2],
  "mid_point": 2.0,
  "millers": [
    [
      1,
      0,
      0
    ]
  ],
  "elements": [
    "Al"
  ],
  "potcars": [
    "./POTCAR"
  ],
  "relax_incar" : "INCAR_metal_rlx_low",
  "scale": [
    1.0
  ],
  "skip_relax": true,
  "pert_numb": 5,
  "pert_box": 0.03,
  "pert_atom": 0.01,
  "coll_ndata": 5000,
  "_comment": "that's all"
}

```

5.4 dpngen init_surf machine parameters

init_surf_mdata:

type: dict

argument path: init_surf_mdata

machine.json file

api_version:

type: str

argument path: init_surf_mdata/api_version

Please set to 1.0

deepmd_version:

type: str, optional, default: 2

argument path: init_surf_mdata/deepmd_version

DeePMD-kit version, e.g. 2.1.3

fp:

type: dict

argument path: init_surf_mdata/fp

Parameters of command, machine, and resources for fp

command:

type: str

argument path: init_surf_mdata/fp/command

Command of a program.

machine:

type: dict

argument path: init_surf_mdata/fp/machine

batch_type:

type: str

argument path: init_surf_mdata/fp/
machine/batch_type

The batch job system type. Option: PBS, Shell,
LSF, Lebesgue, Slurm, Torque, SlurmJobAr-
ray, DistributedShell, DpCloudServer

local_root:

type: NoneType | str

argument path: init_surf_mdata/fp/
machine/local_root

The dir where the tasks and relating files locate.
Typically the project dir.

remote_root:

type: `NoneType | str`, optional
 argument path: `init_surf_mdata/fp/machine/remote_root`

The dir where the tasks are executed on the remote machine. Only needed when context is not lazy-local.

clean_asynchronously:

type: `bool`, optional, default: `False`
 argument path: `init_surf_mdata/fp/machine/clean_asynchronously`

Clean the remote directory asynchronously after the job finishes.

Depending on the value of `context_type`, different sub args are accepted.

context_type:

type: `str` (flag key)
 argument path: `init_surf_mdata/fp/machine/context_type`
 possible choices: `LebesgueContext`, `SSHContext`, `LocalContext`, `DpCloudServerContext`, `LazyLocalContext`, `HDFSContext`

The connection used to remote machine.
 Option: `SSHContext`, `LazyLocalContext`, `LebesgueContext`, `LocalContext`, `DpCloudServerContext`, `HDFSContext`

When `context_type` is set to `LebesgueContext` (or its aliases `lebesguecontext`, `Lebesgue`, `lebesgue`):

remote_profile:

type: `dict`
 argument path: `init_surf_mdata/fp/machine[LebesgueContext]/remote_profile`

The information used to maintain the connection with remote machine.

email:

type: `str`
 argument path: `init_surf_mdata/fp/machine[LebesgueContext]/remote_profile/email`

Email

password:

type: str
argument path: init_surf_mdata/
fp/machine[LebesgueContext]/
remote_profile/password
Password

program_id:

type: int, alias: *project_id*
argument path: init_surf_mdata/
fp/machine[LebesgueContext]/
remote_profile/program_id
Program ID

keep_backup:

type: bool, optional
argument path: init_surf_mdata/
fp/machine[LebesgueContext]/
remote_profile/keep_backup
keep download and upload zip

input_data:

type: dict
argument path: init_surf_mdata/
fp/machine[LebesgueContext]/
remote_profile/input_data
Configuration of job

When `context_type` is set to `SSHContext` (or its aliases `sshcontext`, `SSH`, `ssh`):

remote_profile:

type: dict
argument path: init_surf_mdata/fp/
machine[SSHContext]/remote_profile

The information used to maintain the connection with remote machine.

hostname:

type: str
argument path: init_surf_mdata/
fp/machine[SSHContext]/
remote_profile/hostname
hostname or ip of ssh connection.

username:

type: str
argument path: init_surf_mdata/
fp/machine[SSHContext]/
remote_profile/username
username of target linux system

password:

type: str, optional
 argument path: `init_surf_mdata/
 fp/machine[SSHContext]/
 remote_profile/password`

(deprecated) password of linux system. Please use [SSH keys](#) instead to improve security.

port:

type: int, optional, default: 22
 argument path: `init_surf_mdata/
 fp/machine[SSHContext]/
 remote_profile/port`

ssh connection port.

key_filename:

type: `NoneType` | str, optional,
 default: None
 argument path: `init_surf_mdata/
 fp/machine[SSHContext]/
 remote_profile/key_filename`

key filename used by ssh connection.
 If left None, find key in `~/.ssh` or use
 password for login

passphrase:

type: `NoneType` | str, optional,
 default: None
 argument path: `init_surf_mdata/
 fp/machine[SSHContext]/
 remote_profile/passphrase`

passphrase of key used by ssh connection

timeout:

type: int, optional, default: 10
 argument path: `init_surf_mdata/
 fp/machine[SSHContext]/
 remote_profile/timeout`

timeout of ssh connection

totp_secret:

type: `NoneType` | str, optional,
 default: None
 argument path: `init_surf_mdata/
 fp/machine[SSHContext]/
 remote_profile/totp_secret`

Time-based one time password secret. It should be a base32-encoded string extracted from the 2D code.

tar_compress:

type: bool, optional, default: True
argument path: `init_surf_mdata/
fp/machine[SSHContext]/
remote_profile/tar_compress`

The archive will be compressed in upload and download if it is True. If not, compression will be skipped.

When `context_type` is set to `LocalContext` (or its aliases `localcontext`, `Local`, `local`):

remote_profile:

type: dict, optional
argument path: `init_surf_mdata/fp/
machine[LocalContext]/
remote_profile`

The information used to maintain the connection with remote machine. This field is empty for this context.

When `context_type` is set to `DpCloudServerContext` (or its aliases `dpcloudservercontext`, `DpCloudServer`, `dpcloudserver`):

remote_profile:

type: dict
argument path: `init_surf_mdata/fp/
machine[DpCloudServerContext]/
remote_profile`

The information used to maintain the connection with remote machine.

email:

type: str
argument path:
`init_surf_mdata/fp/
machine[DpCloudServerContext]/
remote_profile/email`

Email

password:

type: str
argument path:
`init_surf_mdata/fp/
machine[DpCloudServerContext]/
remote_profile/password`

Password

program_id:

type: int, alias: *project_id*

argument path:
 init_surf_mdata/fp/
 machine[DpCloudServerContext]/
 remote_profile/program_id

Program ID

keep_backup:

type: bool, optional
 argument path:
 init_surf_mdata/fp/
 machine[DpCloudServerContext]/
 remote_profile/keep_backup

keep download and upload zip

input_data:

type: dict
 argument path:
 init_surf_mdata/fp/
 machine[DpCloudServerContext]/
 remote_profile/input_data

Configuration of job

When `context_type` is set to `LazyLocalContext` (or its aliases `lazylocalcontext`, `LazyLocal`, `lazylocal`):

remote_profile:

type: dict, optional
 argument path: init_surf_mdata/fp/
 machine[LazyLocalContext]/
 remote_profile

The information used to maintain the connection with remote machine. This field is empty for this context.

When `context_type` is set to `HDFSContext` (or its aliases `hdfscontext`, `HDFS`, `hdfs`):

remote_profile:

type: dict, optional
 argument path: init_surf_mdata/fp/
 machine[HDFSContext]/remote_profile

The information used to maintain the connection with remote machine. This field is empty for this context.

resources:

type: dict
 argument path: init_surf_mdata/fp/resources

number_node:

type: int, optional, default: 1

argument path: `init_surf_mdata/fp/resources/number_node`

The number of node need for each *job*

cpu_per_node:

type: int, optional, default: 1

argument path: `init_surf_mdata/fp/resources/cpu_per_node`

cpu numbers of each node assigned to each job.

gpu_per_node:

type: int, optional, default: 0

argument path: `init_surf_mdata/fp/resources/gpu_per_node`

gpu numbers of each node assigned to each job.

queue_name:

type: str, optional, default: `****`

argument path: `init_surf_mdata/fp/resources/queue_name`

The queue name of batch job scheduler system.

group_size:

type: int

argument path: `init_surf_mdata/fp/resources/group_size`

The number of *tasks* in a *job*. 0 means infinity.

custom_flags:

type: list, optional

argument path: `init_surf_mdata/fp/resources/custom_flags`

The extra lines pass to job submitting script header

strategy:

type: dict, optional

argument path: `init_surf_mdata/fp/resources/strategy`

strategies we use to generation job submitting scripts.

if_cuda_multi_devices:

type: bool, optional, default: False

argument path: `init_surf_mdata/fp/resources/strategy/if_cuda_multi_devices`

If there are multiple nvidia GPUS on the node, and we want to assign the tasks to different GPUS. If true, dpdispatcher will manually export environment variable CUDA_VISIBLE_DEVICES to different task. Usually, this option will be used with Task.task_need_resources variable simultaneously.

ratio_unfinished:

type: float, optional, default: 0.0
 argument path: init_surf_mdata/
 fp/resources/strategy/
 ratio_unfinished

The ratio of *jobs* that can be unfinished.

para_deg:

type: int, optional, default: 1
 argument path: init_surf_mdata/fp/
 resources/para_deg

Decide how many tasks will be run in parallel.

source_list:

type: list, optional, default: []
 argument path: init_surf_mdata/fp/
 resources/source_list

The env file to be sourced before the command execution.

module_purge:

type: bool, optional, default: False
 argument path: init_surf_mdata/fp/
 resources/module_purge

Remove all modules on HPC system before module load (module_list)

module_unload_list:

type: list, optional, default: []
 argument path: init_surf_mdata/fp/
 resources/module_unload_list

The modules to be unloaded on HPC system before submitting jobs

module_list:

type: list, optional, default: []
 argument path: init_surf_mdata/fp/
 resources/module_list

The modules to be loaded on HPC system before submitting jobs

envs:

type: dict, optional, default: {}

argument path:

init_surf_mdata/fp/resources/envs

The environment variables to be exported on before submitting jobs

wait_time:

type: int | float, optional, default: 0

argument path: init_surf_mdata/fp/
resources/wait_time

The waiting time in second after a single *task* submitted

Depending on the value of *batch_type*, different sub args are accepted.

batch_type:

type: str (flag key)

argument path: init_surf_mdata/fp/
resources/batch_type

possible choices: *Shell*, *Slurm*, *PBS*,
Torque, *SlurmJobArray*, *DpCloudServer*,
LSF, *Lebesgue*, *DistributedShell*

The batch job system type loaded from machine/batch_type.

When *batch_type* is set to *Shell* (or its alias *shell*):

kwargs:

type: dict, optional

argument path: init_surf_mdata/fp/
resources[Shell]/kwargs

This field is empty for this batch.

When *batch_type* is set to *Slurm* (or its alias *slurm*):

kwargs:

type: dict, optional

argument path: init_surf_mdata/fp/
resources[Slurm]/kwargs

Extra arguments.

custom_gpu_line:

type: NoneType | str, optional,
default: None

argument path: `init_surf_mdata/
fp/resources[Slurm]/kwargs/
custom_gpu_line`

Custom GPU configuration, starting
with `#SBATCH`

When `batch_type` is set to PBS (or its alias `pbs`):

kwargs:

type: dict, optional
argument path: `init_surf_mdata/fp/
resources[PBS]/kwargs`

This field is empty for this batch.

When `batch_type` is set to Torque (or its alias `torque`):

kwargs:

type: dict, optional
argument path: `init_surf_mdata/fp/
resources[Torque]/kwargs`

This field is empty for this batch.

When `batch_type` is set to SlurmJobArray (or its alias
`slurmjobarray`):

kwargs:

type: dict, optional
argument path: `init_surf_mdata/fp/
resources[SlurmJobArray]/kwargs`

Extra arguments.

custom_gpu_line:

type: `NoneType` | str, optional,
default: None
argument path: `init_surf_mdata/
fp/resources[SlurmJobArray]/
kwargs/custom_gpu_line`

Custom GPU configuration, starting
with `#SBATCH`

When `batch_type` is set to DpCloudServer (or its alias
`dpcloudserver`):

kwargs:

type: dict, optional
argument path: `init_surf_mdata/fp/
resources[DpCloudServer]/kwargs`

This field is empty for this batch.

When `batch_type` is set to LSF (or its alias `lsf`):

kwargs:

type: dict
argument path: `init_surf_mdata/fp/resources[LSF]/kwargs`

Extra arguments.

gpu_usage:

type: bool, optional, default: False
argument path: `init_surf_mdata/fp/resources[LSF]/kwargs/gpu_usage`

Choosing if GPU is used in the calculation step.

gpu_new_syntax:

type: bool, optional, default: False
argument path: `init_surf_mdata/fp/resources[LSF]/kwargs/gpu_new_syntax`

For LFS \geq 10.1.0.3, new option -gpu for #BSUB could be used. If False, and old syntax would be used.

gpu_exclusive:

type: bool, optional, default: True
argument path: `init_surf_mdata/fp/resources[LSF]/kwargs/gpu_exclusive`

Only take effect when new syntax enabled. Control whether submit tasks in exclusive way for GPU.

custom_gpu_line:

type: NoneType | str, optional,
default: None
argument path: `init_surf_mdata/fp/resources[LSF]/kwargs/custom_gpu_line`

Custom GPU configuration, starting with #BSUB

When `batch_type` is set to Lebesgue (or its alias `lebesgue`):

kwargs:

type: dict, optional
argument path: `init_surf_mdata/fp/resources[Lebesgue]/kwargs`

This field is empty for this batch.

When `batch_type` is set to DistributedShell (or its alias `distributedshell`):

kwargs:

type: dict, optional
 argument path: `init_surf_mdata/fp/resources[DistributedShell]/kwargs`

This field is empty for this batch.

user_forward_files:

type: list, optional
 argument path:
`init_surf_mdata/fp/user_forward_files`

Files to be forwarded to the remote machine.

user_backward_files:

type: list, optional
 argument path:
`init_surf_mdata/fp/user_backward_files`

Files to be backwarded from the remote machine.

5.5 init_reaction

`dpngen init_reaction` is a workflow to initialize data for reactive systems of small gas-phase molecules. The workflow was introduced in the “Initialization” section of *Energy & Fuels*, 2021, 35 (1), 762–769.

To start the workflow, one needs a box containing reactive systems. The following packages are required for each of the step:

- Exploring: [LAMMPS](#)
- Sampling: [MDDatasetBuilder](#)
- Labeling: [Gaussian](#)

The Exploring step uses LAMMPS `pair_style reaxff` to run a short ReaxMD NVT MD simulation. In the Sampling step, molecular clusters are taken and k-means clustering algorithm is applied to remove the redundancy, which is described in *Nature Communications*, 11, 5713 (2020). The Labeling step calculates energies and forces using the Gaussian package.

An example of `reaction.json` is given below:

```

1 {
2   "type_map": [
3     "H",
4     "O"
5   ],
6   "reaxff": {
7     "data": "data.hydrogen",
8     "ff": "ffield.reax.cho",
9     "control": "lmp_control",
10    "temp": 3000,
11    "tau_t": 100,
12    "dt": 0.1,

```

(continues on next page)

(continued from previous page)

```

13     "nstep": 10000,
14     "dump_freq": 100
15 },
16 "cutoff": 3.5,
17 "dataset_size": 100,
18 "qmkeywords": "b3lyp/6-31g** force Geom=PrintInputOrient"
19 }

```

For detailed parameters, see *parameters* and *machine parameters*.

The generated data can be used to continue DP-GEN concurrent learning workflow. Read *Energy & Fuels*, 2021, 35 (1), 762–769 for details.

5.6 dpngen init_reaction parameters

init_reaction_jdata:

type: dict

argument path: init_reaction_jdata

Generate initial data for reactive systems for small gas-phase molecules, from a ReaxFF NVT MD trajectory.

type_map:

type: list

argument path: init_reaction_jdata/type_map

Type map, which should match types in the initial data. e.g. ["C", "H", "O"]

reaxff:

type: dict

argument path: init_reaction_jdata/reaxff

Parameters for ReaxFF NVT MD.

data:

type: str

argument path: init_reaction_jdata/reaxff/data

Path to initial LAMMPS data file. The atom_style should be charge.

ff:

type: str

argument path: init_reaction_jdata/reaxff/ff

Path to ReaxFF force field file. Available in the lammps/potentials directory.

control:

type: str

argument path: init_reaction_jdata/reaxff/control

Path to ReaxFF control file.

temp:

type: int | float

argument path: init_reaction_jdata/reaxff/temp

Target Temperature for the NVT MD simulation. Unit: K.

dt:

type: int | float

argument path: init_reaction_jdata/reaxff/dt

Real time for every time step. Unit: fs.

tau_t:

type: int | float

argument path: init_reaction_jdata/reaxff/tau_t

Time to determine how rapidly the temperature. Unit: fs.

dump_freq:

type: int

argument path:

init_reaction_jdata/reaxff/dump_freq

Frequency of time steps to collect trajectory.

nstep:

type: int

argument path: init_reaction_jdata/reaxff/nstep

Total steps to run the ReaxFF MD simulation.

cutoff:

type: float

argument path: init_reaction_jdata/cutoff

Cutoff radius to take clusters from the trajectory. Note that only a complete molecule or free radical will be taken.

dataset_size:

type: int

argument path: init_reaction_jdata/dataset_size

Collected dataset size for each bond type.

qmkeywords:

type: str

argument path: init_reaction_jdata/qmkeywords

Gaussian keywords for first-principle calculations. e.g. force mn15/6-31g**
Geom=PrintInputOrient. Note that “force” job is necessary to collect data.
Geom=PrintInputOrient should be used when there are more than 50 atoms in
a cluster.

5.7 dpngen init_reaction machine parameters

init_reaction_mdata:

type: dict

argument path: init_reaction_mdata

machine.json file

api_version:

type: str

argument path: init_reaction_mdata/api_version

Please set to 1.0

deepmd_version:

type: str, optional, default: 2

argument path: init_reaction_mdata/deepmd_version

DeePMD-kit version, e.g. 2.1.3

reaxff:

type: dict

argument path: init_reaction_mdata/reaxff

Parameters of command, machine, and resources for reaxff

command:

type: str

argument path: init_reaction_mdata/reaxff/command

Command of a program.

machine:

type: dict

argument path: init_reaction_mdata/reaxff/machine

batch_type:

type: str

argument path: init_reaction_mdata/
reaxff/machine/batch_type

The batch job system type. Option: PBS, Shell,
LSF, Lebesgue, Slurm, Torque, SlurmJobAr-
ray, DistributedShell, DpCloudServer

local_root:

type: NoneType | str

argument path: init_reaction_mdata/
reaxff/machine/local_root

The dir where the tasks and relating files locate.
Typically the project dir.

remote_root:

type: `NoneType | str`, optional
 argument path: `init_reaction_mdata/reaxff/machine/remote_root`

The dir where the tasks are executed on the remote machine. Only needed when context is not lazy-local.

clean_asynchronously:

type: `bool`, optional, default: `False`
 argument path: `init_reaction_mdata/reaxff/machine/clean_asynchronously`

Clean the remote directory asynchronously after the job finishes.

Depending on the value of *context_type*, different sub args are accepted.

context_type:

type: `str` (flag key)
 argument path: `init_reaction_mdata/reaxff/machine/context_type`
 possible choices: *LebesgueContext*, *SSHContext*, *LocalContext*, *DpCloudServerContext*, *LazyLocalContext*, *HDFSContext*

The connection used to remote machine.
 Option: *SSHContext*, *LazyLocalContext*, *LebesgueContext*, *LocalContext*, *DpCloudServerContext*, *HDFSContext*

When *context_type* is set to *LebesgueContext* (or its aliases *lebesguecontext*, *Lebesgue*, *lebesgue*):

remote_profile:

type: `dict`
 argument path: `init_reaction_mdata/reaxff/machine[LebesgueContext]/remote_profile`

The information used to maintain the connection with remote machine.

email:

type: `str`
 argument path:
`init_reaction_mdata/reaxff/machine[LebesgueContext]/remote_profile/email`

Email

password:

type: str
argument path:
init_reaction_mdata/reaxff/
machine[LebesgueContext]/
remote_profile/password
Password

program_id:

type: int, alias: *project_id*
argument path:
init_reaction_mdata/reaxff/
machine[LebesgueContext]/
remote_profile/program_id
Program ID

keep_backup:

type: bool, optional
argument path:
init_reaction_mdata/reaxff/
machine[LebesgueContext]/
remote_profile/keep_backup
keep download and upload zip

input_data:

type: dict
argument path:
init_reaction_mdata/reaxff/
machine[LebesgueContext]/
remote_profile/input_data
Configuration of job

When `context_type` is set to `SSHContext` (or its aliases `sshcontext`, `SSH`, `ssh`):

remote_profile:

type: dict
argument path:
init_reaction_mdata/reaxff/
machine[SSHContext]/remote_profile

The information used to maintain the connection with remote machine.

hostname:

type: str
argument path:
init_reaction_mdata/reaxff/
machine[SSHContext]/
remote_profile/hostname
hostname or ip of ssh connection.

username:

type: str
 argument path:
 init_reaction_mdata/reaxff/
 machine[SSHContext]/
 remote_profile/username
 username of target linux system

password:

type: str, optional
 argument path:
 init_reaction_mdata/reaxff/
 machine[SSHContext]/
 remote_profile/password
 (deprecated) password of linux system. Please use [SSH keys](#) instead to improve security.

port:

type: int, optional, default: 22
 argument path:
 init_reaction_mdata/reaxff/
 machine[SSHContext]/
 remote_profile/port
 ssh connection port.

key_filename:

type: NoneType | str, optional,
 default: None
 argument path:
 init_reaction_mdata/reaxff/
 machine[SSHContext]/
 remote_profile/key_filename
 key filename used by ssh connection.
 If left None, find key in ~/.ssh or use
 password for login

passphrase:

type: NoneType | str, optional,
 default: None
 argument path:
 init_reaction_mdata/reaxff/
 machine[SSHContext]/
 remote_profile/passphrase
 passphrase of key used by ssh connection

timeout:

type: int, optional, default: 10
 argument path:
 init_reaction_mdata/reaxff/

machine[SSHContext]/
remote_profile/timeout

timeout of ssh connection

totp_secret:

type: NoneType | str, optional,
default: None

argument path:

init_reaction_mdata/reaxff/
machine[SSHContext]/
remote_profile/totp_secret

Time-based one time password secret. It should be a base32-encoded string extracted from the 2D code.

tar_compress:

type: bool, optional, default: True

argument path:

init_reaction_mdata/reaxff/
machine[SSHContext]/
remote_profile/tar_compress

The archive will be compressed in upload and download if it is True. If not, compression will be skipped.

When `context_type` is set to `LocalContext` (or its aliases `localcontext`, `Local`, `local`):

remote_profile:

type: dict, optional

argument path: init_reaction_mdata/
reaxff/machine[LocalContext]/
remote_profile

The information used to maintain the connection with remote machine. This field is empty for this context.

When `context_type` is set to `DpCloudServerContext` (or its aliases `dpcloudservercontext`, `DpCloudServer`, `dpcloudserver`):

remote_profile:

type: dict

argument path:

init_reaction_mdata/reaxff/
machine[DpCloudServerContext]/
remote_profile

The information used to maintain the connection with remote machine.

email:

type: str

argument path:
 init_reaction_mdata/reaxff/
 machine[DpCloudServerContext]/
 remote_profile/email

Email

password:

type: str
 argument path:
 init_reaction_mdata/reaxff/
 machine[DpCloudServerContext]/
 remote_profile/password

Password

program_id:

type: int, alias: *project_id*
 argument path:
 init_reaction_mdata/reaxff/
 machine[DpCloudServerContext]/
 remote_profile/program_id

Program ID

keep_backup:

type: bool, optional
 argument path:
 init_reaction_mdata/reaxff/
 machine[DpCloudServerContext]/
 remote_profile/keep_backup

keep download and upload zip

input_data:

type: dict
 argument path:
 init_reaction_mdata/reaxff/
 machine[DpCloudServerContext]/
 remote_profile/input_data

Configuration of job

When `context_type` is set to `LazyLocalContext` (or its aliases `lazylocalcontext`, `LazyLocal`, `lazylocal`):

remote_profile:

type: dict, optional
 argument path: init_reaction_mdata/
 reaxff/machine[LazyLocalContext]/
 remote_profile

The information used to maintain the connection with remote machine. This field is empty for this context.

When `context_type` is set to `HDFSContext` (or its aliases `hdfscontext`, `HDFS`, `hdfs`):

remote_profile:

type: dict, optional

argument path:

init_reaction_mdata/reaxff/
machine[HDFSText]/remote_profile

The information used to maintain the connection with remote machine. This field is empty for this context.

resources:

type: dict

argument path:

init_reaction_mdata/reaxff/resources

number_node:

type: int, optional, default: 1

argument path: init_reaction_mdata/
reaxff/resources/number_node

The number of node need for each *job*

cpu_per_node:

type: int, optional, default: 1

argument path: init_reaction_mdata/
reaxff/resources/cpu_per_node

cpu numbers of each node assigned to each job.

gpu_per_node:

type: int, optional, default: 0

argument path: init_reaction_mdata/
reaxff/resources/gpu_per_node

gpu numbers of each node assigned to each job.

queue_name:

type: str, optional, default: ""

argument path: init_reaction_mdata/
reaxff/resources/queue_name

The queue name of batch job scheduler system.

group_size:

type: int

argument path: init_reaction_mdata/
reaxff/resources/group_size

The number of *tasks* in a *job*. 0 means infinity.

custom_flags:

type: list, optional

argument path: `init_reaction_mdata/
reaxff/resources/custom_flags`

The extra lines pass to job submitting script header

strategy:

type: dict, optional

argument path: `init_reaction_mdata/
reaxff/resources/strategy`

strategies we use to generation job submitting scripts.

if_cuda_multi_devices:

type: bool, optional, default: False

argument path:

`init_reaction_mdata/reaxff/
resources/strategy/
if_cuda_multi_devices`

If there are multiple nvidia GPUS on the node, and we want to assign the tasks to different GPUS. If true, dpdispatcher will manually export environment variable `CUDA_VISIBLE_DEVICES` to different task. Usually, this option will be used with `Task.task_need_resources` variable simultaneously.

ratio_unfinished:

type: float, optional, default: 0.0

argument path:

`init_reaction_mdata/reaxff/
resources/strategy/
ratio_unfinished`

The ratio of *jobs* that can be unfinished.

para_deg:

type: int, optional, default: 1

argument path: `init_reaction_mdata/
reaxff/resources/para_deg`

Decide how many tasks will be run in parallel.

source_list:

type: list, optional, default: []

argument path: `init_reaction_mdata/
reaxff/resources/source_list`

The env file to be sourced before the command execution.

module_purge:

type: bool, optional, default: False
argument path: `init_reaction_mdata/
reaxff/resources/module_purge`
Remove all modules on HPC system before
module load (`module_list`)

module_unload_list:

type: list, optional, default: []
argument path: `init_reaction_mdata/
reaxff/resources/module_unload_list`
The modules to be unloaded on HPC system
before submitting jobs

module_list:

type: list, optional, default: []
argument path: `init_reaction_mdata/
reaxff/resources/module_list`
The modules to be loaded on HPC system be-
fore submitting jobs

envs:

type: dict, optional, default: {}
argument path: `init_reaction_mdata/
reaxff/resources/envs`
The environment variables to be exported on
before submitting jobs

wait_time:

type: int | float, optional, default: 0
argument path: `init_reaction_mdata/
reaxff/resources/wait_time`
The waiting time in second after a single *task*
submitted

Depending on the value of *batch_type*, different sub args are
accepted.

batch_type:

type: str (flag key)
argument path: `init_reaction_mdata/
reaxff/resources/batch_type`
possible choices: *Shell*, *Slurm*, *PBS*,
Torque, *SlurmJobArray*, *DpCloudServer*,
LSF, *Lebesgue*, *DistributedShell*
The batch job system type loaded from ma-
chine/`batch_type`.

When *batch_type* is set to *Shell* (or its alias *shell*):

kwargs:

type: dict, optional
 argument path: `init_reaction_mdata/`
`reaxff/resources[Shell]/kwargs`

This field is empty for this batch.

When `batch_type` is set to Slurm (or its alias `slurm`):

kwargs:

type: dict, optional
 argument path: `init_reaction_mdata/`
`reaxff/resources[Slurm]/kwargs`

Extra arguments.

custom_gpu_line:

type: `NoneType` | str, optional,
 default: `None`
 argument path:
`init_reaction_mdata/reaxff/`
`resources[Slurm]/kwargs/`
`custom_gpu_line`

Custom GPU configuration, starting
 with `#SBATCH`

When `batch_type` is set to PBS (or its alias `pbs`):

kwargs:

type: dict, optional
 argument path: `init_reaction_mdata/`
`reaxff/resources[PBS]/kwargs`

This field is empty for this batch.

When `batch_type` is set to Torque (or its alias `torque`):

kwargs:

type: dict, optional
 argument path: `init_reaction_mdata/`
`reaxff/resources[Torque]/kwargs`

This field is empty for this batch.

When `batch_type` is set to SlurmJobArray (or its alias `slurmjobarray`):

kwargs:

type: dict, optional
 argument path:
`init_reaction_mdata/reaxff/`
`resources[SlurmJobArray]/kwargs`

Extra arguments.

custom_gpu_line:

type: NoneType | str, optional,
default: None

argument path:

init_reaction_mdata/reaxff/
resources[SlurmJobArray]/
kwargs/custom_gpu_line

Custom GPU configuration, starting
with #SBATCH

When `batch_type` is set to `DpCloudServer` (or its alias `dpcloudserver`):

kwargs:

type: dict, optional

argument path:

init_reaction_mdata/reaxff/
resources[DpCloudServer]/kwargs

This field is empty for this batch.

When `batch_type` is set to `LSF` (or its alias `lsf`):

kwargs:

type: dict

argument path: init_reaction_mdata/
reaxff/resources[LSF]/kwargs

Extra arguments.

gpu_usage:

type: bool, optional, default: False

argument path:

init_reaction_mdata/reaxff/
resources[LSF]/kwargs/
gpu_usage

Choosing if GPU is used in the calcu-
lation step.

gpu_new_syntax:

type: bool, optional, default: False

argument path:

init_reaction_mdata/reaxff/
resources[LSF]/kwargs/
gpu_new_syntax

For LFS >= 10.1.0.3, new option -gpu
for #BSUB could be used. If False,
and old syntax would be used.

gpu_exclusive:

type: bool, optional, default: True

argument path:
 init_reaction_mdata/reaxff/
 resources[LSF]/kwargs/
 gpu_exclusive

Only take effect when new syntax enabled. Control whether submit tasks in exclusive way for GPU.

custom_gpu_line:

type: NoneType | str, optional,
 default: None
 argument path:
 init_reaction_mdata/reaxff/
 resources[LSF]/kwargs/
 custom_gpu_line

Custom GPU configuration, starting with #BSUB

When `batch_type` is set to Lebesgue (or its alias lebesgue):

kwargs:

type: dict, optional
 argument path: init_reaction_mdata/
 reaxff/resources[Lebesgue]/kwargs

This field is empty for this batch.

When `batch_type` is set to DistributedShell (or its alias distributedshell):

kwargs:

type: dict, optional
 argument path:
 init_reaction_mdata/reaxff/
 resources[DistributedShell]/kwargs

This field is empty for this batch.

user_forward_files:

type: list, optional
 argument path:
 init_reaction_mdata/reaxff/user_forward_files

Files to be forwarded to the remote machine.

user_backward_files:

type: list, optional
 argument path:
 init_reaction_mdata/reaxff/user_backward_files

Files to be backwarded from the remote machine.

build:

type: dict

argument path: `init_reaction_mdata/build`

Parameters of command, machine, and resources for build

command:

type: `str`

argument path: `init_reaction_mdata/build/command`

Command of a program.

machine:

type: `dict`

argument path: `init_reaction_mdata/build/machine`

batch_type:

type: `str`

argument path: `init_reaction_mdata/build/machine/batch_type`

The batch job system type. Option: PBS, Shell, LSF, Lebesgue, Slurm, Torque, SlurmJobArray, DistributedShell, DpCloudServer

local_root:

type: `NoneType | str`

argument path: `init_reaction_mdata/build/machine/local_root`

The dir where the tasks and relating files locate. Typically the project dir.

remote_root:

type: `NoneType | str`, optional

argument path: `init_reaction_mdata/build/machine/remote_root`

The dir where the tasks are executed on the remote machine. Only needed when context is not lazy-local.

clean_asynchronously:

type: `bool`, optional, default: `False`

argument path: `init_reaction_mdata/build/machine/clean_asynchronously`

Clean the remote directory asynchronously after the job finishes.

Depending on the value of *context_type*, different sub args are accepted.

context_type:

type: `str` (flag key)

argument path: `init_reaction_mdata/build/machine/context_type`

possible choices: *LebesgueContext*,
SSHContext, *LocalContext*,
DpCloudServerContext,
LazyLocalContext, *HDFSContext*

The connection used to remote machine.
 Option: *SSHContext*, *LazyLocalContext*,
LebesgueContext, *LocalContext*, *DpCloud-*
ServerContext, *HDFSContext*

When *context_type* is set to *LebesgueContext* (or its aliases
lebesguecontext, *Lebesgue*, *lebesgue*):

remote_profile:

type: dict
 argument path: *init_reaction_mdata/*
build/machine[LebesgueContext]/
remote_profile

The information used to maintain the connec-
 tion with remote machine.

email:

type: str
 argument path:
init_reaction_mdata/build/
machine[LebesgueContext]/
remote_profile/email
 Email

password:

type: str
 argument path:
init_reaction_mdata/build/
machine[LebesgueContext]/
remote_profile/password
 Password

program_id:

type: int, alias: *project_id*
 argument path:
init_reaction_mdata/build/
machine[LebesgueContext]/
remote_profile/program_id
 Program ID

keep_backup:

type: bool, optional
 argument path:
init_reaction_mdata/build/
machine[LebesgueContext]/
remote_profile/keep_backup
 keep download and upload zip

input_data:

type: dict
argument path:
init_reaction_mdata/build/
machine[LebesgueContext]/
remote_profile/input_data

Configuration of job

When `context_type` is set to `SSHContext` (or its aliases `sshcontext`, `SSH`, `ssh`):

remote_profile:

type: dict
argument path:
init_reaction_mdata/build/
machine[SSHContext]/remote_profile

The information used to maintain the connection with remote machine.

hostname:

type: str
argument path:
init_reaction_mdata/build/
machine[SSHContext]/
remote_profile/hostname

hostname or ip of ssh connection.

username:

type: str
argument path:
init_reaction_mdata/build/
machine[SSHContext]/
remote_profile/username

username of target linux system

password:

type: str, optional
argument path:
init_reaction_mdata/build/
machine[SSHContext]/
remote_profile/password

(deprecated) password of linux system. Please use [SSH keys](#) instead to improve security.

port:

type: int, optional, default: 22
argument path:
init_reaction_mdata/build/
machine[SSHContext]/
remote_profile/port

ssh connection port.

key_filename:

type: `NoneType` | `str`, optional,

default: `None`

argument path:

`init_reaction_mdata/build/
machine[SSHContext]/
remote_profile/key_filename`

key filename used by ssh connection.

If left `None`, find key in `~/.ssh` or use password for login

passphrase:

type: `NoneType` | `str`, optional,

default: `None`

argument path:

`init_reaction_mdata/build/
machine[SSHContext]/
remote_profile/passphrase`

passphrase of key used by ssh connection

timeout:

type: `int`, optional, default: `10`

argument path:

`init_reaction_mdata/build/
machine[SSHContext]/
remote_profile/timeout`

timeout of ssh connection

totp_secret:

type: `NoneType` | `str`, optional,

default: `None`

argument path:

`init_reaction_mdata/build/
machine[SSHContext]/
remote_profile/totp_secret`

Time-based one time password secret. It should be a base32-encoded string extracted from the 2D code.

tar_compress:

type: `bool`, optional, default: `True`

argument path:

`init_reaction_mdata/build/
machine[SSHContext]/
remote_profile/tar_compress`

The archive will be compressed in upload and download if it is `True`. If not, compression will be skipped.

When `context_type` is set to `LocalContext` (or its aliases `localcontext`, `Local`, `local`):

remote_profile:

type: dict, optional
argument path: `init_reaction_mdata/build/machine[LocalContext]/remote_profile`

The information used to maintain the connection with remote machine. This field is empty for this context.

When `context_type` is set to `DpCloudServerContext` (or its aliases `dpcloudservercontext`, `DpCloudServer`, `dpcloudserver`):

remote_profile:

type: dict
argument path: `init_reaction_mdata/build/machine[DpCloudServerContext]/remote_profile`

The information used to maintain the connection with remote machine.

email:

type: str
argument path: `init_reaction_mdata/build/machine[DpCloudServerContext]/remote_profile/email`

Email

password:

type: str
argument path: `init_reaction_mdata/build/machine[DpCloudServerContext]/remote_profile/password`

Password

program_id:

type: int, alias: *project_id*
argument path: `init_reaction_mdata/build/machine[DpCloudServerContext]/remote_profile/program_id`

Program ID

keep_backup:

type: bool, optional

argument path:
 init_reaction_mdata/build/
 machine[DpCloudServerContext]/
 remote_profile/keep_backup

keep download and upload zip

input_data:

type: dict
 argument path:
 init_reaction_mdata/build/
 machine[DpCloudServerContext]/
 remote_profile/input_data

Configuration of job

When `context_type` is set to `LazyLocalContext` (or its aliases `lazylocalcontext`, `LazyLocal`, `lazylocal`):

remote_profile:

type: dict, optional
 argument path: init_reaction_mdata/
 build/machine[LazyLocalContext]/
 remote_profile

The information used to maintain the connection with remote machine. This field is empty for this context.

When `context_type` is set to `HDFSContext` (or its aliases `hdfscontext`, `HDFS`, `hdfs`):

remote_profile:

type: dict, optional
 argument path:
 init_reaction_mdata/build/
 machine[HDFSContext]/remote_profile

The information used to maintain the connection with remote machine. This field is empty for this context.

resources:

type: dict
 argument path:
 init_reaction_mdata/build/resources

number_node:

type: int, optional, default: 1
 argument path: init_reaction_mdata/
 build/resources/number_node

The number of node need for each *job*

cpu_per_node:

type: int, optional, default: 1

argument path: `init_reaction_mdata/
build/resources/cpu_per_node`
cpu numbers of each node assigned to each job.

gpu_per_node:

type: int, optional, default: 0
argument path: `init_reaction_mdata/
build/resources/gpu_per_node`
gpu numbers of each node assigned to each job.

queue_name:

type: str, optional, default: ""
argument path: `init_reaction_mdata/
build/resources/queue_name`
The queue name of batch job scheduler system.

group_size:

type: int
argument path: `init_reaction_mdata/
build/resources/group_size`
The number of *tasks* in a *job*. 0 means infinity.

custom_flags:

type: list, optional
argument path: `init_reaction_mdata/
build/resources/custom_flags`
The extra lines pass to job submitting script header

strategy:

type: dict, optional
argument path: `init_reaction_mdata/
build/resources/strategy`
strategies we use to generation job submitting scripts.

if_cuda_multi_devices:

type: bool, optional, default: False
argument path:
`init_reaction_mdata/build/
resources/strategy/
if_cuda_multi_devices`

If there are multiple nvidia GPUS on the node, and we want to assign the tasks to different GPUS. If true, dpdispatcher will manually export environment variable `CUDA_VISIBLE_DEVICES` to different task. Usually, this option will be

used with `Task.task_need_resources` variable simultaneously.

ratio_unfinished:

type: float, optional, default: 0.0
 argument path:
 init_reaction_mdata/build/
 resources/strategy/
 ratio_unfinished

The ratio of *jobs* that can be unfinished.

para_deg:

type: int, optional, default: 1
 argument path: init_reaction_mdata/
 build/resources/para_deg

Decide how many tasks will be run in parallel.

source_list:

type: list, optional, default: []
 argument path: init_reaction_mdata/
 build/resources/source_list

The env file to be sourced before the command execution.

module_purge:

type: bool, optional, default: False
 argument path: init_reaction_mdata/
 build/resources/module_purge

Remove all modules on HPC system before module load (module_list)

module_unload_list:

type: list, optional, default: []
 argument path: init_reaction_mdata/
 build/resources/module_unload_list

The modules to be unloaded on HPC system before submitting jobs

module_list:

type: list, optional, default: []
 argument path: init_reaction_mdata/
 build/resources/module_list

The modules to be loaded on HPC system before submitting jobs

envs:

type: dict, optional, default: {}

argument path: `init_reaction_mdata/
build/resources/envs`

The environment variables to be exported on
before submitting jobs

wait_time:

type: `int | float`, optional, default: `0`
argument path: `init_reaction_mdata/
build/resources/wait_time`

The waiting time in second after a single *task*
submitted

Depending on the value of *batch_type*, different sub args are
accepted.

batch_type:

type: `str` (flag key)
argument path: `init_reaction_mdata/
build/resources/batch_type`
possible choices: *Shell*, *Slurm*, *PBS*,
Torque, *SlurmJobArray*, *DpCloudServer*,
LSF, *Lebesgue*, *DistributedShell*

The batch job system type loaded from ma-
chine/*batch_type*.

When *batch_type* is set to *Shell* (or its alias *shell*):

kwargs:

type: `dict`, optional
argument path: `init_reaction_mdata/
build/resources[Shell]/kwargs`

This field is empty for this batch.

When *batch_type* is set to *Slurm* (or its alias *slurm*):

kwargs:

type: `dict`, optional
argument path: `init_reaction_mdata/
build/resources[Slurm]/kwargs`

Extra arguments.

custom_gpu_line:

type: `NoneType | str`, optional,
default: `None`
argument path:
`init_reaction_mdata/build/
resources[Slurm]/kwargs/
custom_gpu_line`

Custom GPU configuration, starting
with `#SBATCH`

When `batch_type` is set to PBS (or its alias pbs):

kwargs:

type: dict, optional
 argument path: `init_reaction_mdata/
 build/resources[PBS]/kwargs`

This field is empty for this batch.

When `batch_type` is set to Torque (or its alias torque):

kwargs:

type: dict, optional
 argument path: `init_reaction_mdata/
 build/resources[Torque]/kwargs`

This field is empty for this batch.

When `batch_type` is set to SlurmJobArray (or its alias slurmjobarray):

kwargs:

type: dict, optional
 argument path:
`init_reaction_mdata/build/
 resources[SlurmJobArray]/kwargs`

Extra arguments.

custom_gpu_line:

type: `NoneType | str`, optional,
 default: `None`
 argument path:
`init_reaction_mdata/build/
 resources[SlurmJobArray]/
 kwargs/custom_gpu_line`

Custom GPU configuration, starting
 with `#SBATCH`

When `batch_type` is set to DpCloudServer (or its alias dpcloudserver):

kwargs:

type: dict, optional
 argument path:
`init_reaction_mdata/build/
 resources[DpCloudServer]/kwargs`

This field is empty for this batch.

When `batch_type` is set to LSF (or its alias lsf):

kwargs:

type: dict

argument path: `init_reaction_mdata/
build/resources[LSF]/kwargs`

Extra arguments.

gpu_usage:

type: bool, optional, default: False

argument path:

`init_reaction_mdata/build/
resources[LSF]/kwargs/
gpu_usage`

Choosing if GPU is used in the calculation step.

gpu_new_syntax:

type: bool, optional, default: False

argument path:

`init_reaction_mdata/build/
resources[LSF]/kwargs/
gpu_new_syntax`

For LFS \geq 10.1.0.3, new option `-gpu` for `#BSUB` could be used. If False, and old syntax would be used.

gpu_exclusive:

type: bool, optional, default: True

argument path:

`init_reaction_mdata/build/
resources[LSF]/kwargs/
gpu_exclusive`

Only take effect when new syntax enabled. Control whether submit tasks in exclusive way for GPU.

custom_gpu_line:

type: `NoneType` | `str`, optional,

default: `None`

argument path:

`init_reaction_mdata/build/
resources[LSF]/kwargs/
custom_gpu_line`

Custom GPU configuration, starting with `#BSUB`

When `batch_type` is set to Lebesgue (or its alias `lebesgue`):

kwargs:

type: dict, optional

argument path: `init_reaction_mdata/
build/resources[Lebesgue]/kwargs`

This field is empty for this batch.

When `batch_type` is set to `DistributedShell` (or its alias `distributedshell`):

kwargs:

type: dict, optional
 argument path:
`init_reaction_mdata/build/resources[DistributedShell]/kwargs`

This field is empty for this batch.

user_forward_files:

type: list, optional
 argument path:
`init_reaction_mdata/build/user_forward_files`

Files to be forwarded to the remote machine.

user_backward_files:

type: list, optional
 argument path:
`init_reaction_mdata/build/user_backward_files`

Files to be backwarded from the remote machine.

fp:

type: dict
 argument path: `init_reaction_mdata/fp`
 Parameters of command, machine, and resources for fp

command:

type: str
 argument path: `init_reaction_mdata/fp/command`
 Command of a program.

machine:

type: dict
 argument path: `init_reaction_mdata/fp/machine`

batch_type:

type: str
 argument path: `init_reaction_mdata/fp/machine/batch_type`

The batch job system type. Option: PBS, Shell, LSF, Lebesgue, Slurm, Torque, SlurmJobArray, DistributedShell, DpCloudServer

local_root:

type: NoneType | str

argument path: `init_reaction_mdata/fp/machine/local_root`

The dir where the tasks and relating files locate. Typically the project dir.

remote_root:

type: `NoneType | str`, optional

argument path: `init_reaction_mdata/fp/machine/remote_root`

The dir where the tasks are executed on the remote machine. Only needed when context is not lazy-local.

clean_asynchronously:

type: `bool`, optional, default: `False`

argument path: `init_reaction_mdata/fp/machine/clean_asynchronously`

Clean the remote directory asynchronously after the job finishes.

Depending on the value of `context_type`, different sub args are accepted.

context_type:

type: `str` (flag key)

argument path: `init_reaction_mdata/fp/machine/context_type`

possible choices: `LebesgueContext`, `SSHContext`, `LocalContext`, `DpCloudServerContext`, `LazyLocalContext`, `HDFSContext`

The connection used to remote machine. Option: `SSHContext`, `LazyLocalContext`, `LebesgueContext`, `LocalContext`, `DpCloudServerContext`, `HDFSContext`

When `context_type` is set to `LebesgueContext` (or its aliases `lebesguecontext`, `Lebesgue`, `lebesgue`):

remote_profile:

type: `dict`

argument path: `init_reaction_mdata/fp/machine[LebesgueContext]/remote_profile`

The information used to maintain the connection with remote machine.

email:

type: `str`

argument path:
 init_reaction_mdata/fp/
 machine[LebesgueContext]/
 remote_profile/email

Email

password:

type: str
 argument path:
 init_reaction_mdata/fp/
 machine[LebesgueContext]/
 remote_profile/password

Password

program_id:

type: int, alias: *project_id*
 argument path:
 init_reaction_mdata/fp/
 machine[LebesgueContext]/
 remote_profile/program_id

Program ID

keep_backup:

type: bool, optional
 argument path:
 init_reaction_mdata/fp/
 machine[LebesgueContext]/
 remote_profile/keep_backup

keep download and upload zip

input_data:

type: dict
 argument path:
 init_reaction_mdata/fp/
 machine[LebesgueContext]/
 remote_profile/input_data

Configuration of job

When `context_type` is set to `SSHContext` (or its aliases `sshcontext`, `SSH`, `ssh`):

remote_profile:

type: dict
 argument path: init_reaction_mdata/fp/
 machine[SSHContext]/remote_profile

The information used to maintain the connection with remote machine.

hostname:

type: str

argument path:
init_reaction_mdata/fp/
machine[SSHContext]/
remote_profile/hostname

hostname or ip of ssh connection.

username:

type: str
argument path:
init_reaction_mdata/fp/
machine[SSHContext]/
remote_profile/username

username of target linux system

password:

type: str, optional
argument path:
init_reaction_mdata/fp/
machine[SSHContext]/
remote_profile/password

(deprecated) password of linux system. Please use [SSH keys](#) instead to improve security.

port:

type: int, optional, default: 22
argument path:
init_reaction_mdata/fp/
machine[SSHContext]/
remote_profile/port

ssh connection port.

key_filename:

type: NoneType | str, optional,
default: None
argument path:
init_reaction_mdata/fp/
machine[SSHContext]/
remote_profile/key_filename

key filename used by ssh connection.
If left None, find key in ~/.ssh or use
password for login

passphrase:

type: NoneType | str, optional,
default: None
argument path:
init_reaction_mdata/fp/
machine[SSHContext]/
remote_profile/passphrase

passphrase of key used by ssh connection

timeout:

type: int, optional, default: 10

argument path:

init_reaction_mdata/fp/

machine[SSHContext]/

remote_profile/timeout

timeout of ssh connection

totp_secret:

type: NoneType | str, optional,

default: None

argument path:

init_reaction_mdata/fp/

machine[SSHContext]/

remote_profile/totp_secret

Time-based one time password secret. It should be a base32-encoded string extracted from the 2D code.

tar_compress:

type: bool, optional, default: True

argument path:

init_reaction_mdata/fp/

machine[SSHContext]/

remote_profile/tar_compress

The archive will be compressed in upload and download if it is True. If not, compression will be skipped.

When `context_type` is set to `LocalContext` (or its aliases `localcontext`, `Local`, `local`):

remote_profile:

type: dict, optional

argument path: init_reaction_mdata/fp/

machine[LocalContext]/

remote_profile

The information used to maintain the connection with remote machine. This field is empty for this context.

When `context_type` is set to `DpCloudServerContext` (or its aliases `dpcloudservercontext`, `DpCloudServer`, `dpcloudserver`):

remote_profile:

type: dict

argument path: init_reaction_mdata/fp/

machine[DpCloudServerContext]/

remote_profile

The information used to maintain the connection with remote machine.

email:

type: str
argument path:
init_reaction_mdata/fp/
machine[DpCloudServerContext]/
remote_profile/email

Email

password:

type: str
argument path:
init_reaction_mdata/fp/
machine[DpCloudServerContext]/
remote_profile/password

Password

program_id:

type: int, alias: *project_id*
argument path:
init_reaction_mdata/fp/
machine[DpCloudServerContext]/
remote_profile/program_id

Program ID

keep_backup:

type: bool, optional
argument path:
init_reaction_mdata/fp/
machine[DpCloudServerContext]/
remote_profile/keep_backup

keep download and upload zip

input_data:

type: dict
argument path:
init_reaction_mdata/fp/
machine[DpCloudServerContext]/
remote_profile/input_data

Configuration of job

When `context_type` is set to `LazyLocalContext` (or its aliases `lazylocalcontext`, `LazyLocal`, `lazylocal`):

remote_profile:

type: dict, optional
argument path: init_reaction_mdata/fp/
machine[LazyLocalContext]/
remote_profile

The information used to maintain the connection with remote machine. This field is empty for this context.

When `context_type` is set to `HDFSContext` (or its aliases `hdfscontext`, `HDFS`, `hdfs`):

remote_profile:

type: dict, optional
 argument path: `init_reaction_mdata/fp/machine[HDFSContext]/remote_profile`
 The information used to maintain the connection with remote machine. This field is empty for this context.

resources:

type: dict
 argument path: `init_reaction_mdata/fp/resources`

number_node:

type: int, optional, default: 1
 argument path: `init_reaction_mdata/fp/resources/number_node`
 The number of node need for each *job*

cpu_per_node:

type: int, optional, default: 1
 argument path: `init_reaction_mdata/fp/resources/cpu_per_node`
 cpu numbers of each node assigned to each job.

gpu_per_node:

type: int, optional, default: 0
 argument path: `init_reaction_mdata/fp/resources/gpu_per_node`
 gpu numbers of each node assigned to each job.

queue_name:

type: str, optional, default: `****`
 argument path: `init_reaction_mdata/fp/resources/queue_name`
 The queue name of batch job scheduler system.

group_size:

type: int
 argument path: `init_reaction_mdata/fp/resources/group_size`
 The number of *tasks* in a *job*. 0 means infinity.

custom_flags:

type: list, optional

argument path: `init_reaction_mdata/fp/
resources/custom_flags`

The extra lines pass to job submitting script header

strategy:

type: dict, optional

argument path: `init_reaction_mdata/fp/
resources/strategy`

strategies we use to generation job submitting scripts.

if_cuda_multi_devices:

type: bool, optional, default: False

argument path:

`init_reaction_mdata/fp/
resources/strategy/
if_cuda_multi_devices`

If there are multiple nvidia GPUS on the node, and we want to assign the tasks to different GPUS. If true, dpdispatcher will manually export environment variable `CUDA_VISIBLE_DEVICES` to different task. Usually, this option will be used with `Task.task_need_resources` variable simultaneously.

ratio_unfinished:

type: float, optional, default: 0.0

argument path:

`init_reaction_mdata/fp/
resources/strategy/
ratio_unfinished`

The ratio of *jobs* that can be unfinished.

para_deg:

type: int, optional, default: 1

argument path: `init_reaction_mdata/fp/
resources/para_deg`

Decide how many tasks will be run in parallel.

source_list:

type: list, optional, default: []

argument path: `init_reaction_mdata/fp/
resources/source_list`

The env file to be sourced before the command execution.

module_purge:

type: bool, optional, default: False
 argument path: `init_reaction_mdata/fp/resources/module_purge`

Remove all modules on HPC system before module load (`module_list`)

module_unload_list:

type: list, optional, default: []
 argument path: `init_reaction_mdata/fp/resources/module_unload_list`

The modules to be unloaded on HPC system before submitting jobs

module_list:

type: list, optional, default: []
 argument path: `init_reaction_mdata/fp/resources/module_list`

The modules to be loaded on HPC system before submitting jobs

envs:

type: dict, optional, default: {}
 argument path: `init_reaction_mdata/fp/resources/envs`

The environment variables to be exported on before submitting jobs

wait_time:

type: int | float, optional, default: 0
 argument path: `init_reaction_mdata/fp/resources/wait_time`

The waiting time in second after a single *task* submitted

Depending on the value of *batch_type*, different sub args are accepted.

batch_type:

type: str (flag key)
 argument path: `init_reaction_mdata/fp/resources/batch_type`
 possible choices: *Shell*, *Slurm*, *PBS*, *Torque*, *SlurmJobArray*, *DpCloudServer*, *LSF*, *Lebesgue*, *DistributedShell*

The batch job system type loaded from machine/`batch_type`.

When `batch_type` is set to *Shell* (or its alias *shell*):

kwargs:

type: dict, optional
argument path: `init_reaction_mdata/fp/resources[Shell]/kwargs`

This field is empty for this batch.

When `batch_type` is set to Slurm (or its alias `slurm`):

kwargs:

type: dict, optional
argument path: `init_reaction_mdata/fp/resources[Slurm]/kwargs`

Extra arguments.

custom_gpu_line:

type: `NoneType` | str, optional,
default: `None`
argument path:
`init_reaction_mdata/fp/resources[Slurm]/kwargs/custom_gpu_line`

Custom GPU configuration, starting
with `#SBATCH`

When `batch_type` is set to PBS (or its alias `pbs`):

kwargs:

type: dict, optional
argument path: `init_reaction_mdata/fp/resources[PBS]/kwargs`

This field is empty for this batch.

When `batch_type` is set to Torque (or its alias `torque`):

kwargs:

type: dict, optional
argument path: `init_reaction_mdata/fp/resources[Torque]/kwargs`

This field is empty for this batch.

When `batch_type` is set to `SlurmJobArray` (or its alias `slurmjobarray`):

kwargs:

type: dict, optional
argument path: `init_reaction_mdata/fp/resources[SlurmJobArray]/kwargs`

Extra arguments.

custom_gpu_line:

type: `NoneType` | `str`, optional,
default: `None`

argument path:

`init_reaction_mdata/fp/
resources[SlurmJobArray]/
kwargs/custom_gpu_line`

Custom GPU configuration, starting
with `#SBATCH`

When `batch_type` is set to `DpCloudServer` (or its alias `dpcLOUDserver`):

kwargs:

type: `dict`, optional

argument path: `init_reaction_mdata/fp/
resources[DpCloudServer]/kwargs`

This field is empty for this batch.

When `batch_type` is set to `LSF` (or its alias `lsf`):

kwargs:

type: `dict`

argument path: `init_reaction_mdata/fp/
resources[LSF]/kwargs`

Extra arguments.

gpu_usage:

type: `bool`, optional, default: `False`

argument path:

`init_reaction_mdata/fp/
resources[LSF]/kwargs/
gpu_usage`

Choosing if GPU is used in the calcu-
lation step.

gpu_new_syntax:

type: `bool`, optional, default: `False`

argument path:

`init_reaction_mdata/fp/
resources[LSF]/kwargs/
gpu_new_syntax`

For LFS \geq 10.1.0.3, new option `-gpu`
for `#BSUB` could be used. If `False`,
and old syntax would be used.

gpu_exclusive:

type: `bool`, optional, default: `True`

argument path:

`init_reaction_mdata/fp/
resources[LSF]/kwargs/
gpu_exclusive`

Only take effect when new syntax enabled. Control whether submit tasks in exclusive way for GPU.

custom_gpu_line:

type: `NoneType` | `str`, optional,
default: `None`

argument path:
`init_reaction_mdata/fp/
resources[LSF]/kwargs/
custom_gpu_line`

Custom GPU configuration, starting
with `#BSUB`

When `batch_type` is set to Lebesgue (or its alias `lebesgue`):

kwargs:

type: `dict`, optional
argument path: `init_reaction_mdata/fp/
resources[Lebesgue]/kwargs`

This field is empty for this batch.

When `batch_type` is set to `DistributedShell` (or its alias `distributedshell`):

kwargs:

type: `dict`, optional
argument path: `init_reaction_mdata/fp/
resources[DistributedShell]/kwargs`

This field is empty for this batch.

user_forward_files:

type: `list`, optional
argument path:
`init_reaction_mdata/fp/user_forward_files`

Files to be forwarded to the remote machine.

user_backward_files:

type: `list`, optional
argument path:
`init_reaction_mdata/fp/user_backward_files`

Files to be backwarded from the remote machine.

SIMPLIFY

6.1 Simplify

When you have a dataset containing lots of repeated data, this step will help you simplify your dataset. The workflow contains three stages: train, model_devi, and fp. The train stage and the fp stage are as the same as the run step, and the model_devi stage will calculate model deviations of the rest data that has not been confirmed accurate. Data with small model deviations will be confirmed accurate, while the program will pick data from those with large model deviations to the new dataset.

Use the following script to start the workflow:

```
dpngen simplify param.json machine.json
```

Here is an example of param.json for QM7 dataset:

```
{
  "type_map": [
    "C",
    "H",
    "N",
    "O",
    "S"
  ],
  "mass_map": [
    12.011,
    1.008,
    14.007,
    15.999,
    32.065
  ],
  "pick_data": "/scratch/jz748/simplify/qm7",
  "init_data_prefix": "",
  "init_data_sys": [],
  "sys_batch_size": [
    "auto"
  ],
  "numb_models": 4,
  "default_training_param": {
    "model": {
      "type_map": [
        "C",
```

(continues on next page)

(continued from previous page)

```

        "H",
        "N",
        "O",
        "S"
    ],
    "descriptor": {
        "type": "se_a",
        "sel": [
            7,
            16,
            3,
            3,
            1
        ],
        "rcut_smth": 1.00,
        "rcut": 6.00,
        "neuron": [
            25,
            50,
            100
        ],
        "resnet_dt": false,
        "axis_neuron": 12
    },
    "fitting_net": {
        "neuron": [
            240,
            240,
            240
        ],
        "resnet_dt": true
    }
},
"learning_rate": {
    "type": "exp",
    "start_lr": 0.001,
    "decay_steps": 10,
    "decay_rate": 0.99
},
"loss": {
    "start_pref_e": 0.02,
    "limit_pref_e": 1,
    "start_pref_f": 1000,
    "limit_pref_f": 1,
    "start_pref_v": 0,
    "limit_pref_v": 0,
    "start_pref_pf": 0,
    "limit_pref_pf": 0
},
"training": {
    "set_prefix": "set",
    "stop_batch": 10000,

```

(continues on next page)

(continued from previous page)

```

        "disp_file": "lcurve.out",
        "disp_freq": 1000,
        "numb_test": 1,
        "save_freq": 1000,
        "save_ckpt": "model.ckpt",
        "disp_training": true,
        "time_training": true,
        "profiling": false,
        "profiling_file": "timeline.json"
    },
    "_comment": "that's all"
},
"fp_style": "gaussian",
"shuffle_poscar": false,
"fp_task_max": 1000,
"fp_task_min": 10,
"fp_pp_path": "/home/jzzeng/",
"fp_pp_files": [],
"fp_params": {
    "keywords": "mn15/6-31g** force nosymm scf(maxcyc=512)",
    "nproc": 28,
    "multiplicity": 1,
    "_comment": " that's all "
},
"init_pick_number": 100,
"iter_pick_number": 100,
"f_trust_lo": 0.25,
"f_trust_hi": 0.45,
"_comment": " that's all "
}

```

Here pick_data is the directory to data to simplify where the program recursively detects systems System with deepmd/npz format. init_pick_number and iter_pick_number are the numbers of picked frames. e_trust_lo, e_trust_hi mean the range of the deviation of the frame energy, and f_trust_lo and f_trust_hi mean the range of the max deviation of atomic forces in a frame. fp_style can only be gaussian currently. Other parameters are as the same as those of generator.

6.2 dpngen simplify parameters

simplify_jdata:

type: dict

argument path: simplify_jdata

Parameters for simplify.json, the first argument of *dpngen simplify*.

type_map:

type: list

argument path: simplify_jdata/type_map

Atom types.

mass_map:

type: list | str, optional, default: auto
argument path: simplify_jdata/mass_map

Standard atomic weights (default: “auto”). if one want to use isotopes, or non-standard element names, chemical symbols, or atomic number in the type_map list, please customize the mass_map list instead of using “auto”. Tips: at present the default value will not be applied automatically, so you need to set “mass_map” manually in param.json.

use_ele_temp:

type: int, optional, default: 0
argument path: simplify_jdata/use_ele_temp

Currently only support fp_style vasp.

- 0: no electron temperature.
- 1: eletron temperature as frame parameter.
- 2: electron temperature as atom parameter.

init_data_prefix:

type: str, optional
argument path: simplify_jdata/init_data_prefix

Prefix of initial data directories.

init_data_sys:

type: list
argument path: simplify_jdata/init_data_sys

Directories of initial data. You may use either absolute or relative path here. Systems will be detected recursively in the directories.

sys_format:

type: str, optional, default: vasp/poscar
argument path: simplify_jdata/sys_format

Format of initial data.

init_batch_size:

type: list | str, optional
argument path: simplify_jdata/init_batch_size

Each number is the batch_size of corresponding system for training in init_data_sys. One recommended rule for setting the sys_batch_size and init_batch_size is that batch_size mutiply number of atoms ot the stucture should be larger than 32. If set to auto, batch size will be 32 divided by number of atoms.

sys_configs_prefix:

type: str, optional
argument path: simplify_jdata/sys_configs_prefix

Prefix of sys_configs.

sys_configs:

type: list
argument path: simplify_jdata/sys_configs

Containing directories of structures to be explored in iterations. Wildcard characters are supported here.

sys_batch_size:

type: list, optional

argument path: simplify_jdata/sys_batch_size

Each number is the batch_size for training of corresponding system in sys_configs. If set to auto, batch size will be 32 divided by number of atoms.

labeled:

type: bool, optional, default: False

argument path: simplify_jdata/labeled

If true, the initial data is labeled.

pick_data:

type: str

argument path: simplify_jdata/pick_data

Path to the directory with the pick data with the deepmd/npz format. Systems are detected recursively.

init_pick_number:

type: int

argument path: simplify_jdata/init_pick_number

The number of initial pick data.

iter_pick_number:

type: int

argument path: simplify_jdata/iter_pick_number

The number of pick data in each iteration.

model_devi_f_trust_lo:

type: float

argument path: simplify_jdata/model_devi_f_trust_lo

The lower bound of forces for the selection for the model deviation.

model_devi_f_trust_hi:

type: float

argument path: simplify_jdata/model_devi_f_trust_hi

The higher bound of forces for the selection for the model deviation.

numb_models:

type: int

argument path: simplify_jdata/numb_models

Number of models to be trained in 00.train. 4 is recommend.

training_iter0_model_path:

type: list, optional

argument path: simplify_jdata/training_iter0_model_path

The model used to init the first iter training. Number of element should be equal to numb_models.

training_init_model:

type: bool, optional

argument path: `simplify_jdata/training_init_model`

Iteration > 0, the model parameters will be initialized from the model trained at the previous iteration. Iteration == 0, the model parameters will be initialized from `training_iter0_model_path`.

default_training_param:

type: dict

argument path: `simplify_jdata/default_training_param`

Training parameters for deepmd-kit in `00.train`. You can find instructions from here: (<https://github.com/deepmodeling/deepmd-kit>).

dp_compress:

type: bool, optional, default: False

argument path: `simplify_jdata/dp_compress`

Use dp compress to compress the model.

training_reuse_iter:

type: int | NoneType, optional

argument path: `simplify_jdata/training_reuse_iter`

The minimal index of iteration that continues training models from old models of last iteration.

training_reuse_old_ratio:

type: NoneType | float, optional

argument path: `simplify_jdata/training_reuse_old_ratio`

The probability proportion of old data during training. This option is only adopted when continuing training models from old models. This option will override default parameters.

training_reuse_num_steps:

type: int | NoneType, optional, default: 400000, alias:

`training_reuse_stop_batch`

argument path: `simplify_jdata/training_reuse_num_steps`

Number of training batch. This option is only adopted when continuing training models from old models. This option will override default parameters.

training_reuse_start_lr:

type: NoneType | float, optional, default: 0.0001

argument path: `simplify_jdata/training_reuse_start_lr`

The learning rate the start of the training. This option is only adopted when continuing training models from old models. This option will override default parameters.

training_reuse_start_pref_e:

type: int | NoneType | float, optional, default: 0.1

argument path: `simplify_jdata/training_reuse_start_pref_e`

The prefactor of energy loss at the start of the training. This option is only adopted when continuing training models from old models. This option will override default parameters.

training_reuse_start_pref_f:

type: int | NoneType | float, optional, default: 100

argument path: `simplify_jdata/training_reuse_start_pref_f`

The prefactor of force loss at the start of the training. This option is only adopted when continuing training models from old models. This option will override default parameters.

model_devi_activation_func:

type: list | NoneType, optional

argument path: `simplify_jdata/model_devi_activation_func`

The activation function in the model. The shape of list should be (N_models, 2), where 2 represents the embedding and fitting network. This option will override default parameters.

fp_task_max:

type: int, optional

argument path: `simplify_jdata/fp_task_max`

Maximum of structures to be calculated in 02.fp of each iteration.

fp_task_min:

type: int, optional

argument path: `simplify_jdata/fp_task_min`

Minimum of structures to be calculated in 02.fp of each iteration.

fp_accurate_threshold:

type: float, optional

argument path: `simplify_jdata/fp_accurate_threshold`

If the accurate ratio is larger than this number, no fp calculation will be performed, i.e. `fp_task_max = 0`.

fp_accurate_soft_threshold:

type: float, optional

argument path: `simplify_jdata/fp_accurate_soft_threshold`

If the accurate ratio is between this number and `fp_accurate_threshold`, the `fp_task_max` linearly decays to zero.

Depending on the value of `fp_style`, different sub args are accepted.

fp_style:

type: str (flag key), default: none

argument path: `simplify_jdata/fp_style`

possible choices: *none*, *vasp*, *gaussian*

Software for First Principles, if `labeled` is false. Options include “vasp”, “gaussian” up to now.

When `fp_style` is set to *none*:

No fp.

When `fp_style` is set to *vasp*:

VASP.

fp_pp_path:

type: str

argument path: `simplify_jdata[vasp]/fp_pp_path`

Directory of psuedo-potential file to be used for 02.fp exists.

fp_pp_files:

type: list

argument path: `simplify_jdata[vasp]/fp_pp_files`Psuedo-potential file to be used for 02.fp. Note that the order of elements should correspond to the order in `type_map`.**fp_incar:**

type: str

argument path: `simplify_jdata[vasp]/fp_incar`

Input file for VASP. INCAR must specify KSPACING and KGAMMA.

fp_aniso_kspacing:

type: list, optional

argument path: `simplify_jdata[vasp]/fp_aniso_kspacing`

Set anisotropic kspacing. Usually useful for 1-D or 2-D materials. Only support VASP. If it is setting the KSPACING key in INCAR will be ignored.

cvasp:

type: bool, optional

argument path: `simplify_jdata[vasp]/cvasp`If `cvasp` is true, DP-GEN will use Custodian to help control VASP calculation.**ratio_failed:**

type: float, optional

argument path: `simplify_jdata[vasp]/ratio_failed`Check the ratio of unsuccessfully terminated jobs. If too many FP tasks are not converged, `RuntimeError` will be raised.**fp_skip_bad_box:**

type: str, optional

argument path: `simplify_jdata[vasp]/fp_skip_bad_box`

Skip the configurations that are obviously unreasonable before 02.fp

When `fp_style` is set to gaussian:Gaussian. The command should be set as `g16 < input`.**use_clusters:**

type: bool, optional, default: False

argument path: `simplify_jdata[gaussian]/use_clusters`

If set to true, clusters will be taken instead of the whole system.

cluster_cutoff:

type: float, optional

argument path: `simplify_jdata[gaussian]/cluster_cutoff`

The soft cutoff radius of clusters if *use_clusters* is set to true. Molecules will be taken as whole even if part of atoms is out of the cluster. Use *cluster_cutoff_hard* to only take atoms within the hard cutoff radius.

cluster_cutoff_hard:

type: float, optional

argument path: `simplify_jdata[gaussian]/cluster_cutoff_hard`

The hard cutoff radius of clusters if *use_clusters* is set to true. Outside the hard cutoff radius, atoms will not be taken even if they are in a molecule where some atoms are within the cutoff radius.

cluster_minify:

type: bool, optional, default: False

argument path: `simplify_jdata[gaussian]/cluster_minify`

If enabled, when an atom within the soft cutoff radius connects a single bond with a non-hydrogen atom out of the soft cutoff radius, the outer atom will be replaced by a hydrogen atom. When the outer atom is a hydrogen atom, the outer atom will be kept. In this case, other atoms out of the soft cutoff radius will be removed.

fp_params:

type: dict

argument path: `simplify_jdata[gaussian]/fp_params`

Parameters for Gaussian calculation.

keywords:

type: list | str

argument path:

`simplify_jdata[gaussian]/fp_params/keywords`

Keywords for Gaussian input, e.g. `force b3lyp/6-31g**`. If a list, run multiple steps.

multiplicity:

type: int | str, optional, default: auto

argument path: `simplify_jdata[gaussian]/fp_params/multiplicity`

Spin multiplicity for Gaussian input. If *auto*, multiplicity will be detected automatically, with the following rules: when *fragment_guesses*=True, multiplicity will +1 for each radical, and +2 for each oxygen molecule; when *fragment_guesses*=False, multiplicity will be 1 or 2, but +2 for each oxygen molecule.

nproc:

type: int

argument path:

`simplify_jdata[gaussian]/fp_params/nproc`

The number of processors for Gaussian input.

charge:

type: int, optional, default: 0

argument path:

simplify_jdata[gaussian]/fp_params/charge

Molecule charge. Only used when charge is not provided by the system.

fragment_guesses:

type: bool, optional, default: False

argument path: simplify_jdata[gaussian]/

fp_params/fragment_guesses

Initial guess generated from fragment guesses. If True, *multiplicity* should be *auto*.

basis_set:

type: str, optional

argument path:

simplify_jdata[gaussian]/fp_params/basis_set

Custom basis set.

keywords_high_multiplicity:

type: str, optional

argument path: simplify_jdata[gaussian]/

fp_params/keywords_high_multiplicity

Keywords for points with multiple raicals. *multiplicity* should be *auto*. If not set, fallback to normal keywords.

ratio_failed:

type: float, optional

argument path: simplify_jdata[gaussian]/ratio_failed

Check the ratio of unsuccessfully terminated jobs. If too many FP tasks are not converged, RuntimeError will be raised.

6.3 dpngen simplify machine parameters

simplify_mdata:

type: dict

argument path: simplify_mdata

machine.json file

api_version:

type: str

argument path: simplify_mdata/api_version

Please set to 1.0

deepmd_version:

type: str, optional, default: 2
 argument path: simplify_mdata/deepmd_version
 DeePMD-kit version, e.g. 2.1.3

train:

type: dict
 argument path: simplify_mdata/train
 Parameters of command, machine, and resources for train

command:

type: str
 argument path: simplify_mdata/train/command
 Command of a program.

machine:

type: dict
 argument path: simplify_mdata/train/machine

batch_type:

type: str
 argument path: simplify_mdata/train/machine/batch_type
 The batch job system type. Option: PBS, Shell, LSF, Lebesgue, Slurm, Torque, SlurmJobArray, DistributedShell, DpCloudServer

local_root:

type: NoneType | str
 argument path: simplify_mdata/train/machine/local_root
 The dir where the tasks and relating files locate. Typically the project dir.

remote_root:

type: NoneType | str, optional
 argument path: simplify_mdata/train/machine/remote_root
 The dir where the tasks are executed on the remote machine. Only needed when context is not lazy-local.

clean_asynchronously:

type: bool, optional, default: False
 argument path: simplify_mdata/train/machine/clean_asynchronously
 Clean the remote directory asynchronously after the job finishes.

Depending on the value of *context_type*, different sub args are accepted.

context_type:

type: str (flag key)
argument path: `simplify_mdata/train/
machine/context_type`
possible choices: *LebesgueContext*,
SSHContext, *LocalContext*,
DpCloudServerContext,
LazyLocalContext, *HDFSContext*

The connection used to remote machine.
Option: *SSHContext*, *LazyLocalContext*,
LebesgueContext, *LocalContext*, *DpCloud-
ServerContext*, *HDFSContext*

When *context_type* is set to *LebesgueContext* (or its aliases *lebesguecontext*, *Lebesgue*, *lebesgue*):

remote_profile:

type: dict
argument path: `simplify_mdata/train/
machine[LebesgueContext]/
remote_profile`

The information used to maintain the connection with remote machine.

email:

type: str
argument path:
`simplify_mdata/train/
machine[LebesgueContext]/
remote_profile/email`
Email

password:

type: str
argument path:
`simplify_mdata/train/
machine[LebesgueContext]/
remote_profile/password`
Password

program_id:

type: int, alias: *project_id*
argument path:
`simplify_mdata/train/
machine[LebesgueContext]/
remote_profile/program_id`
Program ID

keep_backup:

type: bool, optional
 argument path:
 simplify_mdata/train/
 machine[LebesgueContext]/
 remote_profile/keep_backup
 keep download and upload zip

input_data:

type: dict
 argument path:
 simplify_mdata/train/
 machine[LebesgueContext]/
 remote_profile/input_data
 Configuration of job

When `context_type` is set to `SSHContext` (or its aliases `sshcontext`, `SSH`, `ssh`):

remote_profile:

type: dict
 argument path: simplify_mdata/train/
 machine[SSHContext]/remote_profile

The information used to maintain the connection with remote machine.

hostname:

type: str
 argument path: simplify_mdata/
 train/machine[SSHContext]/
 remote_profile/hostname
 hostname or ip of ssh connection.

username:

type: str
 argument path: simplify_mdata/
 train/machine[SSHContext]/
 remote_profile/username
 username of target linux system

password:

type: str, optional
 argument path: simplify_mdata/
 train/machine[SSHContext]/
 remote_profile/password
 (deprecated) password of linux system. Please use [SSH keys](#) instead to improve security.

port:

type: int, optional, default: 22

argument path: `simplify_mdata/
train/machine[SSHContext]/
remote_profile/port`

ssh connection port.

key_filename:

type: `NoneType | str`, optional,
default: `None`

argument path: `simplify_mdata/
train/machine[SSHContext]/
remote_profile/key_filename`

key filename used by ssh connection.
If left `None`, find key in `~/.ssh` or use
password for login

passphrase:

type: `NoneType | str`, optional,
default: `None`

argument path: `simplify_mdata/
train/machine[SSHContext]/
remote_profile/passphrase`

passphrase of key used by ssh connection

timeout:

type: `int`, optional, default: `10`

argument path: `simplify_mdata/
train/machine[SSHContext]/
remote_profile/timeout`

timeout of ssh connection

totp_secret:

type: `NoneType | str`, optional,
default: `None`

argument path: `simplify_mdata/
train/machine[SSHContext]/
remote_profile/totp_secret`

Time-based one time password secret. It should be a base32-encoded string extracted from the 2D code.

tar_compress:

type: `bool`, optional, default: `True`

argument path: `simplify_mdata/
train/machine[SSHContext]/
remote_profile/tar_compress`

The archive will be compressed in upload and download if it is `True`. If not, compression will be skipped.

When `context_type` is set to `LocalContext` (or its aliases `localcontext`, `Local`, `local`):

remote_profile:

type: dict, optional
 argument path: simplify_mdata/train/
 machine[LocalContext]/
 remote_profile

The information used to maintain the connection with remote machine. This field is empty for this context.

When `context_type` is set to `DpCloudServerContext` (or its aliases `dpcloudservercontext`, `DpCloudServer`, `dpcloudserver`):

remote_profile:

type: dict
 argument path: simplify_mdata/train/
 machine[DpCloudServerContext]/
 remote_profile

The information used to maintain the connection with remote machine.

email:

type: str
 argument path:
 simplify_mdata/train/
 machine[DpCloudServerContext]/
 remote_profile/email

Email

password:

type: str
 argument path:
 simplify_mdata/train/
 machine[DpCloudServerContext]/
 remote_profile/password

Password

program_id:

type: int, alias: *project_id*
 argument path:
 simplify_mdata/train/
 machine[DpCloudServerContext]/
 remote_profile/program_id

Program ID

keep_backup:

type: bool, optional
 argument path:
 simplify_mdata/train/
 machine[DpCloudServerContext]/
 remote_profile/keep_backup

keep download and upload zip

input_data:

type: dict

argument path:

simplify_mdata/train/

machine[DpCloudServerContext]/

remote_profile/input_data

Configuration of job

When `context_type` is set to `LazyLocalContext` (or its aliases `lazylocalcontext`, `LazyLocal`, `lazylocal`):

remote_profile:

type: dict, optional

argument path: simplify_mdata/train/

machine[LazyLocalContext]/

remote_profile

The information used to maintain the connection with remote machine. This field is empty for this context.

When `context_type` is set to `HDFSContext` (or its aliases `hdfscontext`, `HDFS`, `hdfs`):

remote_profile:

type: dict, optional

argument path: simplify_mdata/train/

machine[HDFSContext]/remote_profile

The information used to maintain the connection with remote machine. This field is empty for this context.

resources:

type: dict

argument path: simplify_mdata/train/resources

number_node:

type: int, optional, default: 1

argument path: simplify_mdata/train/

resources/number_node

The number of node need for each *job*

cpu_per_node:

type: int, optional, default: 1

argument path: simplify_mdata/train/

resources/cpu_per_node

cpu numbers of each node assigned to each job.

gpu_per_node:

type: int, optional, default: 0

argument path: simplify_mdata/train/
resources/gpu_per_node

gpu numbers of each node assigned to each job.

queue_name:

type: str, optional, default: ""

argument path: simplify_mdata/train/
resources/queue_name

The queue name of batch job scheduler system.

group_size:

type: int

argument path: simplify_mdata/train/
resources/group_size

The number of *tasks* in a *job*. 0 means infinity.

custom_flags:

type: list, optional

argument path: simplify_mdata/train/
resources/custom_flags

The extra lines pass to job submitting script
header

strategy:

type: dict, optional

argument path: simplify_mdata/train/
resources/strategy

strategies we use to generation job submitting
scripts.

if_cuda_multi_devices:

type: bool, optional, default: False

argument path: simplify_mdata/
train/resources/strategy/
if_cuda_multi_devices

If there are multiple nvidia GPUS
on the node, and we want to as-
sign the tasks to different GPUS.If
true, dpdispatcher will manu-
ally export environment variable
CUDA_VISIBLE_DEVICES to dif-
ferent task.Usually, this option will be
used with Task.task_need_resources
variable simultaneously.

ratio_unfinished:

type: float, optional, default: 0.0
argument path: simplify_mdata/
train/resources/strategy/
ratio_unfinished

The ratio of *jobs* that can be unfinished.

para_deg:

type: int, optional, default: 1
argument path: simplify_mdata/train/
resources/para_deg

Decide how many tasks will be run in parallel.

source_list:

type: list, optional, default: []
argument path: simplify_mdata/train/
resources/source_list

The env file to be sourced before the command execution.

module_purge:

type: bool, optional, default: False
argument path: simplify_mdata/train/
resources/module_purge

Remove all modules on HPC system before module load (module_list)

module_unload_list:

type: list, optional, default: []
argument path: simplify_mdata/train/
resources/module_unload_list

The modules to be unloaded on HPC system before submitting jobs

module_list:

type: list, optional, default: []
argument path: simplify_mdata/train/
resources/module_list

The modules to be loaded on HPC system before submitting jobs

envs:

type: dict, optional, default: {}
argument path:
simplify_mdata/train/resources/envs

The environment variables to be exported on before submitting jobs

wait_time:

type: int | float, optional, default: 0
 argument path: `simplify_mdata/train/resources/wait_time`

The waiting time in second after a single *task* submitted

Depending on the value of *batch_type*, different sub args are accepted.

batch_type:

type: str (flag key)
 argument path: `simplify_mdata/train/resources/batch_type`
 possible choices: *Shell*, *Slurm*, *PBS*, *Torque*, *SlurmJobArray*, *DpCloudServer*, *LSF*, *Lebesgue*, *DistributedShell*

The batch job system type loaded from machine/*batch_type*.

When *batch_type* is set to *Shell* (or its alias *shell*):

kwargs:

type: dict, optional
 argument path: `simplify_mdata/train/resources[Shell]/kwargs`

This field is empty for this batch.

When *batch_type* is set to *Slurm* (or its alias *slurm*):

kwargs:

type: dict, optional
 argument path: `simplify_mdata/train/resources[Slurm]/kwargs`

Extra arguments.

custom_gpu_line:

type: NoneType | str, optional,
 default: None
 argument path: `simplify_mdata/train/resources[Slurm]/kwargs/custom_gpu_line`
 Custom GPU configuration, starting with `#SBATCH`

When *batch_type* is set to *PBS* (or its alias *pbs*):

kwargs:

type: dict, optional

argument path: `simplify_mdata/train/resources[PBS]/kwargs`

This field is empty for this batch.

When `batch_type` is set to Torque (or its alias `torque`):

kwargs:

type: dict, optional

argument path: `simplify_mdata/train/resources[Torque]/kwargs`

This field is empty for this batch.

When `batch_type` is set to SlurmJobArray (or its alias `slurmjobarray`):

kwargs:

type: dict, optional

argument path: `simplify_mdata/train/resources[SlurmJobArray]/kwargs`

Extra arguments.

custom_gpu_line:

type: `NoneType` | str, optional,
default: None

argument path:
`simplify_mdata/train/resources[SlurmJobArray]/kwargs/custom_gpu_line`

Custom GPU configuration, starting
with `#SBATCH`

When `batch_type` is set to DpCloudServer (or its alias `dpcLOUDserver`):

kwargs:

type: dict, optional

argument path: `simplify_mdata/train/resources[DpCloudServer]/kwargs`

This field is empty for this batch.

When `batch_type` is set to LSF (or its alias `lsf`):

kwargs:

type: dict

argument path: `simplify_mdata/train/resources[LSF]/kwargs`

Extra arguments.

gpu_usage:

type: bool, optional, default: False

argument path: `simplify_mdata/
train/resources[LSF]/kwargs/
gpu_usage`

Choosing if GPU is used in the calculation step.

gpu_new_syntax:

type: bool, optional, default: False
argument path: `simplify_mdata/
train/resources[LSF]/kwargs/
gpu_new_syntax`

For LFS \geq 10.1.0.3, new option -gpu for #BSUB could be used. If False, and old syntax would be used.

gpu_exclusive:

type: bool, optional, default: True
argument path: `simplify_mdata/
train/resources[LSF]/kwargs/
gpu_exclusive`

Only take effect when new syntax enabled. Control whether submit tasks in exclusive way for GPU.

custom_gpu_line:

type: NoneType | str, optional,
default: None
argument path: `simplify_mdata/
train/resources[LSF]/kwargs/
custom_gpu_line`

Custom GPU configuration, starting with #BSUB

When `batch_type` is set to Lebesgue (or its alias `lebesgue`):

kwargs:

type: dict, optional
argument path: `simplify_mdata/train/
resources[Lebesgue]/kwargs`

This field is empty for this batch.

When `batch_type` is set to DistributedShell (or its alias `distributedshell`):

kwargs:

type: dict, optional
argument path: `simplify_mdata/train/
resources[DistributedShell]/kwargs`

This field is empty for this batch.

user_forward_files:

type: list, optional
argument path:
simplify_mdata/train/user_forward_files
Files to be forwarded to the remote machine.

user_backward_files:

type: list, optional
argument path:
simplify_mdata/train/user_backward_files
Files to be backwarded from the remote machine.

model_devi:

type: dict
argument path: simplify_mdata/model_devi
Parameters of command, machine, and resources for model_devi

command:

type: str
argument path: simplify_mdata/model_devi/command
Command of a program.

machine:

type: dict
argument path: simplify_mdata/model_devi/machine

batch_type:

type: str
argument path: simplify_mdata/
model_devi/machine/batch_type
The batch job system type. Option: PBS, Shell,
LSF, Lebesgue, Slurm, Torque, SlurmJobAr-
ray, DistributedShell, DpCloudServer

local_root:

type: NoneType | str
argument path: simplify_mdata/
model_devi/machine/local_root
The dir where the tasks and relating files locate.
Typically the project dir.

remote_root:

type: NoneType | str, optional
argument path: simplify_mdata/
model_devi/machine/remote_root
The dir where the tasks are executed on the re-
mote machine. Only needed when context is
not lazy-local.

clean_asynchronously:

type: bool, optional, default: False

argument path:

simplify_mdata/model_devi/machine/
clean_asynchronously

Clean the remote directory asynchronously after the job finishes.

Depending on the value of *context_type*, different sub args are accepted.

context_type:

type: str (flag key)

argument path: simplify_mdata/
model_devi/machine/context_type

possible choices: *LebesgueContext*,
SSHContext, *LocalContext*,
DpCloudServerContext,
LazyLocalContext, *HDFSContext*

The connection used to remote machine.

Option: *SSHContext*, *LazyLocalContext*,
LebesgueContext, *LocalContext*, *DpCloud-*
ServerContext, *HDFSContext*

When *context_type* is set to *LebesgueContext* (or its aliases *lebesguecontext*, *Lebesgue*, *lebesgue*):

remote_profile:

type: dict

argument path:

simplify_mdata/model_devi/
machine[*LebesgueContext*]/
remote_profile

The information used to maintain the connection with remote machine.

email:

type: str

argument path:

simplify_mdata/model_devi/
machine[*LebesgueContext*]/
remote_profile/email

Email

password:

type: str

argument path:

simplify_mdata/model_devi/
machine[*LebesgueContext*]/
remote_profile/password

Password

program_id:

type: int, alias: *project_id*
argument path:
simplify_mdata/model_devi/
machine[LebesgueContext]/
remote_profile/program_id
Program ID

keep_backup:

type: bool, optional
argument path:
simplify_mdata/model_devi/
machine[LebesgueContext]/
remote_profile/keep_backup
keep download and upload zip

input_data:

type: dict
argument path:
simplify_mdata/model_devi/
machine[LebesgueContext]/
remote_profile/input_data
Configuration of job

When `context_type` is set to `SSHContext` (or its aliases `sshcontext`, `SSH`, `ssh`):

remote_profile:

type: dict
argument path:
simplify_mdata/model_devi/
machine[SSHContext]/remote_profile
The information used to maintain the connection with remote machine.

hostname:

type: str
argument path:
simplify_mdata/model_devi/
machine[SSHContext]/
remote_profile/hostname
hostname or ip of ssh connection.

username:

type: str
argument path:
simplify_mdata/model_devi/
machine[SSHContext]/
remote_profile/username
username of target linux system

password:

type: str, optional
 argument path:
 simplify_mdata/model_devi/
 machine[SSHContext]/
 remote_profile/password
 (deprecated) password of linux system. Please use [SSH keys](#) instead to improve security.

port:

type: int, optional, default: 22
 argument path:
 simplify_mdata/model_devi/
 machine[SSHContext]/
 remote_profile/port
 ssh connection port.

key_filename:

type: NoneType | str, optional,
 default: None
 argument path:
 simplify_mdata/model_devi/
 machine[SSHContext]/
 remote_profile/key_filename
 key filename used by ssh connection.
 If left None, find key in ~/.ssh or use
 password for login

passphrase:

type: NoneType | str, optional,
 default: None
 argument path:
 simplify_mdata/model_devi/
 machine[SSHContext]/
 remote_profile/passphrase
 passphrase of key used by ssh connection

timeout:

type: int, optional, default: 10
 argument path:
 simplify_mdata/model_devi/
 machine[SSHContext]/
 remote_profile/timeout
 timeout of ssh connection

totp_secret:

type: NoneType | str, optional,
 default: None

argument path:
simplify_mdata/model_devi/
machine[SSHContext]/
remote_profile/totp_secret

Time-based one time password secret. It should be a base32-encoded string extracted from the 2D code.

tar_compress:

type: bool, optional, default: True
argument path:
simplify_mdata/model_devi/
machine[SSHContext]/
remote_profile/tar_compress

The archive will be compressed in upload and download if it is True. If not, compression will be skipped.

When `context_type` is set to `LocalContext` (or its aliases `localcontext`, `Local`, `local`):

remote_profile:

type: dict, optional
argument path: simplify_mdata/
model_devi/machine[LocalContext]/
remote_profile

The information used to maintain the connection with remote machine. This field is empty for this context.

When `context_type` is set to `DpCloudServerContext` (or its aliases `dpcloudservercontext`, `DpCloudServer`, `dpcloudserver`):

remote_profile:

type: dict
argument path:
simplify_mdata/model_devi/
machine[DpCloudServerContext]/
remote_profile

The information used to maintain the connection with remote machine.

email:

type: str
argument path:
simplify_mdata/model_devi/
machine[DpCloudServerContext]/
remote_profile/email

Email

password:

type: str
 argument path:
 simplify_mdata/model_devi/
 machine[DpCloudServerContext]/
 remote_profile/password
 Password

program_id:

type: int, alias: *project_id*
 argument path:
 simplify_mdata/model_devi/
 machine[DpCloudServerContext]/
 remote_profile/program_id
 Program ID

keep_backup:

type: bool, optional
 argument path:
 simplify_mdata/model_devi/
 machine[DpCloudServerContext]/
 remote_profile/keep_backup
 keep download and upload zip

input_data:

type: dict
 argument path:
 simplify_mdata/model_devi/
 machine[DpCloudServerContext]/
 remote_profile/input_data
 Configuration of job

When `context_type` is set to `LazyLocalContext` (or its aliases `lazylocalcontext`, `LazyLocal`, `lazylocal`):

remote_profile:

type: dict, optional
 argument path:
 simplify_mdata/model_devi/
 machine[LazyLocalContext]/
 remote_profile

The information used to maintain the connection with remote machine. This field is empty for this context.

When `context_type` is set to `HDFSContext` (or its aliases `hdfscontext`, `HDFS`, `hdfs`):

remote_profile:

type: dict, optional

argument path:
simplify_mdata/model_devi/
machine[HDFSContext]/remote_profile

The information used to maintain the connection with remote machine. This field is empty for this context.

resources:

type: dict
argument path:
simplify_mdata/model_devi/resources

number_node:

type: int, optional, default: 1
argument path: simplify_mdata/
model_devi/resources/number_node

The number of node need for each *job*

cpu_per_node:

type: int, optional, default: 1
argument path: simplify_mdata/
model_devi/resources/cpu_per_node

cpu numbers of each node assigned to each job.

gpu_per_node:

type: int, optional, default: 0
argument path: simplify_mdata/
model_devi/resources/gpu_per_node

gpu numbers of each node assigned to each job.

queue_name:

type: str, optional, default: ""
argument path: simplify_mdata/
model_devi/resources/queue_name

The queue name of batch job scheduler system.

group_size:

type: int
argument path: simplify_mdata/
model_devi/resources/group_size

The number of *tasks* in a *job*. 0 means infinity.

custom_flags:

type: list, optional
argument path: simplify_mdata/
model_devi/resources/custom_flags

The extra lines pass to job submitting script header

strategy:

type: dict, optional

argument path: simplify_mdata/
model_devi/resources/strategy

strategies we use to generation job submitting scripts.

if_cuda_multi_devices:

type: bool, optional, default: False

argument path:
simplify_mdata/model_devi/
resources/strategy/
if_cuda_multi_devices

If there are multiple nvidia GPUS on the node, and we want to assign the tasks to different GPUS. If true, dpdispatcher will manually export environment variable CUDA_VISIBLE_DEVICES to different task. Usually, this option will be used with Task.task_need_resources variable simultaneously.

ratio_unfinished:

type: float, optional, default: 0.0

argument path: simplify_mdata/
model_devi/resources/
strategy/ratio_unfinished

The ratio of *jobs* that can be unfinished.

para_deg:

type: int, optional, default: 1

argument path: simplify_mdata/
model_devi/resources/para_deg

Decide how many tasks will be run in parallel.

source_list:

type: list, optional, default: []

argument path: simplify_mdata/
model_devi/resources/source_list

The env file to be sourced before the command execution.

module_purge:

type: bool, optional, default: False

argument path: simplify_mdata/
model_devi/resources/module_purge

Remove all modules on HPC system before
module load (module_list)

module_unload_list:

type: list, optional, default: []
argument path:
simplify_mdata/model_devi/
resources/module_unload_list

The modules to be unloaded on HPC system
before submitting jobs

module_list:

type: list, optional, default: []
argument path: simplify_mdata/
model_devi/resources/module_list

The modules to be loaded on HPC system be-
fore submitting jobs

envs:

type: dict, optional, default: {}
argument path: simplify_mdata/
model_devi/resources/envs

The environment variables to be exported on
before submitting jobs

wait_time:

type: int | float, optional, default: 0
argument path: simplify_mdata/
model_devi/resources/wait_time

The waiting time in second after a single *task*
submitted

Depending on the value of *batch_type*, different sub args are
accepted.

batch_type:

type: str (flag key)
argument path: simplify_mdata/
model_devi/resources/batch_type
possible choices: *Shell*, *Slurm*, *PBS*,
Torque, *SlurmJobArray*, *DpCloudServer*,
LSF, *Lebesgue*, *DistributedShell*

The batch job system type loaded from ma-
chine/batch_type.

When *batch_type* is set to *Shell* (or its alias *shell*):

kwargs:

type: dict, optional

argument path: `simplify_mdata/
model_devi/resources[Shell]/kwargs`

This field is empty for this batch.

When `batch_type` is set to Slurm (or its alias `slurm`):

kwargs:

type: dict, optional
argument path: `simplify_mdata/
model_devi/resources[Slurm]/kwargs`

Extra arguments.

custom_gpu_line:

type: `NoneType` | str, optional,
default: None
argument path: `simplify_mdata/
model_devi/resources[Slurm]/
kwargs/custom_gpu_line`

Custom GPU configuration, starting
with `#SBATCH`

When `batch_type` is set to PBS (or its alias `pbs`):

kwargs:

type: dict, optional
argument path: `simplify_mdata/
model_devi/resources[PBS]/kwargs`

This field is empty for this batch.

When `batch_type` is set to Torque (or its alias `torque`):

kwargs:

type: dict, optional
argument path: `simplify_mdata/
model_devi/resources[Torque]/kwargs`

This field is empty for this batch.

When `batch_type` is set to `SlurmJobArray` (or its alias `slurmjobarray`):

kwargs:

type: dict, optional
argument path:
`simplify_mdata/model_devi/
resources[SlurmJobArray]/kwargs`

Extra arguments.

custom_gpu_line:

type: `NoneType` | str, optional,
default: None

argument path:
simplify_mdata/model_devi/
resources[SlurmJobArray]/
kwargs/custom_gpu_line

Custom GPU configuration, starting
with #SBATCH

When `batch_type` is set to `DpCloudServer` (or its alias `dpcloudserver`):

kwargs:

type: dict, optional
argument path:
simplify_mdata/model_devi/
resources[DpCloudServer]/kwargs

This field is empty for this batch.

When `batch_type` is set to `LSF` (or its alias `lsf`):

kwargs:

type: dict
argument path: simplify_mdata/
model_devi/resources[LSF]/kwargs

Extra arguments.

gpu_usage:

type: bool, optional, default: False
argument path: simplify_mdata/
model_devi/resources[LSF]/
kwargs/gpu_usage

Choosing if GPU is used in the calcu-
lation step.

gpu_new_syntax:

type: bool, optional, default: False
argument path: simplify_mdata/
model_devi/resources[LSF]/
kwargs/gpu_new_syntax

For LFS >= 10.1.0.3, new option -gpu
for #BSUB could be used. If False,
and old syntax would be used.

gpu_exclusive:

type: bool, optional, default: True
argument path: simplify_mdata/
model_devi/resources[LSF]/
kwargs/gpu_exclusive

Only take effect when new syntax en-
abled. Control whether submit tasks
in exclusive way for GPU.

custom_gpu_line:

type: `NoneType` | `str`, optional,
 default: `None`
 argument path: `simplify_mdata/
 model_devi/resources[LSF]/
 kwargs/custom_gpu_line`
 Custom GPU configuration, starting
 with `#BSUB`

When `batch_type` is set to `Lebesgue` (or its alias `lebesgue`):

kwargs:

type: `dict`, optional
 argument path:
`simplify_mdata/model_devi/
 resources[Lebesgue]/kwargs`

This field is empty for this batch.

When `batch_type` is set to `DistributedShell` (or its alias `distributedshell`):

kwargs:

type: `dict`, optional
 argument path:
`simplify_mdata/model_devi/
 resources[DistributedShell]/kwargs`

This field is empty for this batch.

user_forward_files:

type: `list`, optional
 argument path:
`simplify_mdata/model_devi/user_forward_files`

Files to be forwarded to the remote machine.

user_backward_files:

type: `list`, optional
 argument path:
`simplify_mdata/model_devi/user_backward_files`

Files to be backwarded from the remote machine.

fp:

type: `dict`
 argument path: `simplify_mdata/fp`
 Parameters of command, machine, and resources for `fp`

command:

type: `str`
 argument path: `simplify_mdata/fp/command`
 Command of a program.

machine:

type: dict

argument path: `simplify_mdata/fp/machine`

batch_type:

type: str

argument path: `simplify_mdata/fp/machine/batch_type`

The batch job system type. Option: PBS, Shell, LSF, Lebesgue, Slurm, Torque, SlurmJobArray, DistributedShell, DpCloudServer

local_root:

type: `NoneType` | str

argument path: `simplify_mdata/fp/machine/local_root`

The dir where the tasks and relating files locate. Typically the project dir.

remote_root:

type: `NoneType` | str, optional

argument path: `simplify_mdata/fp/machine/remote_root`

The dir where the tasks are executed on the remote machine. Only needed when context is not lazy-local.

clean_asynchronously:

type: bool, optional, default: False

argument path: `simplify_mdata/fp/machine/clean_asynchronously`

Clean the remote directory asynchronously after the job finishes.

Depending on the value of *context_type*, different sub args are accepted.

context_type:

type: str (flag key)

argument path: `simplify_mdata/fp/machine/context_type`

possible choices: *LebesgueContext*, *SSHContext*, *LocalContext*, *DpCloudServerContext*, *LazyLocalContext*, *HDFSContext*

The connection used to remote machine. Option: *SSHContext*, *LazyLocalContext*, *LebesgueContext*, *LocalContext*, *DpCloudServerContext*, *HDFSContext*

When `context_type` is set to `LebesgueContext` (or its aliases `lebesguecontext`, `Lebesgue`, `lebesgue`):

remote_profile:

type: dict
 argument path: `simplify_mdata/fp/machine[LebesgueContext]/remote_profile`

The information used to maintain the connection with remote machine.

email:

type: str
 argument path: `simplify_mdata/fp/machine[LebesgueContext]/remote_profile/email`
 Email

password:

type: str
 argument path: `simplify_mdata/fp/machine[LebesgueContext]/remote_profile/password`
 Password

program_id:

type: int, alias: *project_id*
 argument path: `simplify_mdata/fp/machine[LebesgueContext]/remote_profile/program_id`
 Program ID

keep_backup:

type: bool, optional
 argument path: `simplify_mdata/fp/machine[LebesgueContext]/remote_profile/keep_backup`
 keep download and upload zip

input_data:

type: dict
 argument path: `simplify_mdata/fp/machine[LebesgueContext]/remote_profile/input_data`
 Configuration of job

When `context_type` is set to `SSHContext` (or its aliases `sshcontext`, `SSH`, `ssh`):

remote_profile:

type: dict

argument path: `simplify_mdata/fp/machine[SSHContext]/remote_profile`

The information used to maintain the connection with remote machine.

hostname:

type: `str`
argument path: `simplify_mdata/fp/machine[SSHContext]/remote_profile/hostname`
hostname or ip of ssh connection.

username:

type: `str`
argument path: `simplify_mdata/fp/machine[SSHContext]/remote_profile/username`
username of target linux system

password:

type: `str`, optional
argument path: `simplify_mdata/fp/machine[SSHContext]/remote_profile/password`
(deprecated) password of linux system. Please use [SSH keys](#) instead to improve security.

port:

type: `int`, optional, default: 22
argument path: `simplify_mdata/fp/machine[SSHContext]/remote_profile/port`
ssh connection port.

key_filename:

type: `NoneType` | `str`, optional,
default: `None`
argument path: `simplify_mdata/fp/machine[SSHContext]/remote_profile/key_filename`
key filename used by ssh connection.
If left `None`, find key in `~/.ssh` or use password for login

passphrase:

type: `NoneType` | `str`, optional,
default: `None`
argument path: `simplify_mdata/fp/machine[SSHContext]/remote_profile/passphrase`

passphrase of key used by ssh connection

timeout:

type: int, optional, default: 10
 argument path: `simplify_mdata/
 fp/machine[SSHContext]/
 remote_profile/timeout`
 timeout of ssh connection

totp_secret:

type: `NoneType` | str, optional,
 default: None
 argument path: `simplify_mdata/
 fp/machine[SSHContext]/
 remote_profile/totp_secret`
 Time-based one time password secret. It should be a base32-encoded string extracted from the 2D code.

tar_compress:

type: bool, optional, default: True
 argument path: `simplify_mdata/
 fp/machine[SSHContext]/
 remote_profile/tar_compress`
 The archive will be compressed in upload and download if it is True. If not, compression will be skipped.

When `context_type` is set to `LocalContext` (or its aliases `localcontext`, `Local`, `local`):

remote_profile:

type: dict, optional
 argument path: `simplify_mdata/fp/
 machine[LocalContext]/
 remote_profile`
 The information used to maintain the connection with remote machine. This field is empty for this context.

When `context_type` is set to `DpCloudServerContext` (or its aliases `dpcloudservercontext`, `DpCloudServer`, `dpcloudserver`):

remote_profile:

type: dict
 argument path: `simplify_mdata/fp/
 machine[DpCloudServerContext]/
 remote_profile`
 The information used to maintain the connection with remote machine.

email:

type: str
argument path:
simplify_mdata/fp/
machine[DpCloudServerContext]/
remote_profile/email
Email

password:

type: str
argument path:
simplify_mdata/fp/
machine[DpCloudServerContext]/
remote_profile/password
Password

program_id:

type: int, alias: *project_id*
argument path:
simplify_mdata/fp/
machine[DpCloudServerContext]/
remote_profile/program_id
Program ID

keep_backup:

type: bool, optional
argument path:
simplify_mdata/fp/
machine[DpCloudServerContext]/
remote_profile/keep_backup
keep download and upload zip

input_data:

type: dict
argument path:
simplify_mdata/fp/
machine[DpCloudServerContext]/
remote_profile/input_data
Configuration of job

When `context_type` is set to `LazyLocalContext` (or its aliases `lazylocalcontext`, `LazyLocal`, `lazylocal`):

remote_profile:

type: dict, optional
argument path: simplify_mdata/fp/
machine[LazyLocalContext]/
remote_profile

The information used to maintain the connection with remote machine. This field is empty for this context.

When `context_type` is set to `HDFSContext` (or its aliases `hdfscontext`, `HDFS`, `hdfs`):

remote_profile:

type: dict, optional
 argument path: `simplify_mdata/fp/machine[HDFSContext]/remote_profile`
 The information used to maintain the connection with remote machine. This field is empty for this context.

resources:

type: dict
 argument path: `simplify_mdata/fp/resources`

number_node:

type: int, optional, default: 1
 argument path: `simplify_mdata/fp/resources/number_node`
 The number of node need for each *job*

cpu_per_node:

type: int, optional, default: 1
 argument path: `simplify_mdata/fp/resources/cpu_per_node`
 cpu numbers of each node assigned to each job.

gpu_per_node:

type: int, optional, default: 0
 argument path: `simplify_mdata/fp/resources/gpu_per_node`
 gpu numbers of each node assigned to each job.

queue_name:

type: str, optional, default: `****`
 argument path: `simplify_mdata/fp/resources/queue_name`
 The queue name of batch job scheduler system.

group_size:

type: int
 argument path: `simplify_mdata/fp/resources/group_size`
 The number of *tasks* in a *job*. 0 means infinity.

custom_flags:

type: list, optional

argument path: `simplify_mdata/fp/
resources/custom_flags`

The extra lines pass to job submitting script header

strategy:

type: dict, optional

argument path: `simplify_mdata/fp/
resources/strategy`

strategies we use to generation job submitting scripts.

if_cuda_multi_devices:

type: bool, optional, default: False

argument path: `simplify_mdata/
fp/resources/strategy/
if_cuda_multi_devices`

If there are multiple nvidia GPUS on the node, and we want to assign the tasks to different GPUS. If true, dpdispatcher will manually export environment variable `CUDA_VISIBLE_DEVICES` to different task. Usually, this option will be used with `Task.task_need_resources` variable simultaneously.

ratio_unfinished:

type: float, optional, default: 0.0

argument path: `simplify_mdata/fp/resources/
strategy/ratio_unfinished`

The ratio of *jobs* that can be unfinished.

para_deg:

type: int, optional, default: 1

argument path: `simplify_mdata/fp/
resources/para_deg`

Decide how many tasks will be run in parallel.

source_list:

type: list, optional, default: []

argument path: `simplify_mdata/fp/
resources/source_list`

The env file to be sourced before the command execution.

module_purge:

type: bool, optional, default: False

argument path: `simplify_mdata/fp/resources/module_purge`

Remove all modules on HPC system before module load (`module_list`)

module_unload_list:

type: list, optional, default: []

argument path: `simplify_mdata/fp/resources/module_unload_list`

The modules to be unloaded on HPC system before submitting jobs

module_list:

type: list, optional, default: []

argument path: `simplify_mdata/fp/resources/module_list`

The modules to be loaded on HPC system before submitting jobs

envs:

type: dict, optional, default: {}

argument path: `simplify_mdata/fp/resources/envs`

The environment variables to be exported on before submitting jobs

wait_time:

type: int | float, optional, default: 0

argument path: `simplify_mdata/fp/resources/wait_time`

The waiting time in second after a single *task* submitted

Depending on the value of *batch_type*, different sub args are accepted.

batch_type:

type: str (flag key)

argument path: `simplify_mdata/fp/resources/batch_type`

possible choices: *Shell*, *Slurm*, *PBS*, *Torque*, *SlurmJobArray*, *DpCloudServer*, *LSF*, *Lebesgue*, *DistributedShell*

The batch job system type loaded from machine/`batch_type`.

When `batch_type` is set to *Shell* (or its alias `shell`):

kwargs:

type: dict, optional
argument path: `simplify_mdata/fp/
resources[Shell]/kwargs`

This field is empty for this batch.

When `batch_type` is set to Slurm (or its alias `slurm`):

kwargs:

type: dict, optional
argument path: `simplify_mdata/fp/
resources[Slurm]/kwargs`

Extra arguments.

custom_gpu_line:

type: `NoneType` | `str`, optional,
default: `None`
argument path: `simplify_mdata/
fp/resources[Slurm]/kwargs/
custom_gpu_line`
Custom GPU configuration, starting
with `#SBATCH`

When `batch_type` is set to PBS (or its alias `pbs`):

kwargs:

type: dict, optional
argument path: `simplify_mdata/fp/
resources[PBS]/kwargs`

This field is empty for this batch.

When `batch_type` is set to Torque (or its alias `torque`):

kwargs:

type: dict, optional
argument path: `simplify_mdata/fp/
resources[Torque]/kwargs`

This field is empty for this batch.

When `batch_type` is set to `SlurmJobArray` (or its alias `slurmjobarray`):

kwargs:

type: dict, optional
argument path: `simplify_mdata/fp/
resources[SlurmJobArray]/kwargs`

Extra arguments.

custom_gpu_line:

type: `NoneType` | `str`, optional,
default: `None`

argument path: `simplify_mdata/
fp/resources[SlurmJobArray]/
kwargs/custom_gpu_line`

Custom GPU configuration, starting
with `#SBATCH`

When `batch_type` is set to `DpCloudServer` (or its alias `dpcloudserver`):

kwargs:

type: dict, optional
argument path: `simplify_mdata/fp/
resources[DpCloudServer]/kwargs`

This field is empty for this batch.

When `batch_type` is set to `LSF` (or its alias `lsf`):

kwargs:

type: dict
argument path: `simplify_mdata/fp/
resources[LSF]/kwargs`

Extra arguments.

gpu_usage:

type: bool, optional, default: False
argument path: `simplify_mdata/
fp/resources[LSF]/kwargs/
gpu_usage`

Choosing if GPU is used in the calcu-
lation step.

gpu_new_syntax:

type: bool, optional, default: False
argument path: `simplify_mdata/
fp/resources[LSF]/kwargs/
gpu_new_syntax`

For LFS $\geq 10.1.0.3$, new option `-gpu`
for `#BSUB` could be used. If False,
and old syntax would be used.

gpu_exclusive:

type: bool, optional, default: True
argument path: `simplify_mdata/
fp/resources[LSF]/kwargs/
gpu_exclusive`

Only take effect when new syntax en-
abled. Control whether submit tasks
in exclusive way for GPU.

custom_gpu_line:

type: NoneType | str, optional,
default: None

argument path: `simplify_mdata/
fp/resources[LSF]/kwargs/
custom_gpu_line`

Custom GPU configuration, starting
with `#BSUB`

When `batch_type` is set to Lebesgue (or its alias `lebesgue`):

kwargs:

type: dict, optional

argument path: `simplify_mdata/fp/
resources[Lebesgue]/kwargs`

This field is empty for this batch.

When `batch_type` is set to `DistributedShell` (or its alias `distributedshell`):

kwargs:

type: dict, optional

argument path: `simplify_mdata/fp/
resources[DistributedShell]/kwargs`

This field is empty for this batch.

user_forward_files:

type: list, optional

argument path:
`simplify_mdata/fp/user_forward_files`

Files to be forwarded to the remote machine.

user_backward_files:

type: list, optional

argument path:
`simplify_mdata/fp/user_backward_files`

Files to be backwarded from the remote machine.

AUTO TEST

7.1 Autotest Overview: Autotest for Deep Generator

Suppose that we have a potential (can be DFT, DP, MEAM...), `autotest` helps us automatically calculate M properties on N configurations. The folder where the `autotest` runs is called the working directory of `autotest`. Different potentials should be tested in different working directories.

A property is tested in three steps: `make`, `run` and `post`. `make` prepares all computational tasks that are needed to calculate the property. For example to calculate EOS, `make` prepares a series of tasks, each of which has a scaled configuration with certain volume, and all necessary input files necessary for starting a VASP, ABACUS, or LAMMPS calculations. `run` sends all the computational tasks to remote computational resources defined in a machine configuration file like `machine.json`, and automatically collects the results when remote calculations finish. `post` calculates the desired property from the collected results.

7.1.1 Relaxation

The relaxation of a structure should be carried out before calculating all other properties:

```
dpngen autotest make relax.json
dpngen autotest run relax.json machine.json
dpngen autotest post relax.json
```

If, for some reasons, the main program terminated at stage `run`, one can easily restart with the same command. `relax.json` is the parameter file. An example for `deepmd` relaxation is given as:

```
{
  "structures": "confs/mp-*",
  "interaction": {
    "type": "deepmd",
    "model": "frozen_model.pb",
    "type_map": {"Al": 0, "Mg": 1}
  },
  "relaxation": {}
}
```

where the key `structures` provides the structures to relax. `interaction` is provided with `deepmd`, and other options are `vasp`, `abacus`, `meam`...

7.1.2 Task type

There are now six task types implemented in the package: `vasp`, `abacus`, `deepmd`, `meam`, `eam_fs`, and `eam_alloy`. An `inter.json` file in json format containing the interaction parameters will be written in the directory of each task after make. We give input examples of the `interaction` part for each type below:

VASP:

The default of `potcar_prefix` is `""`.

```
"interaction": {
  "type": "vasp",
  "incar": "vasp_input/INCAR",
  "potcar_prefix": "vasp_input",
  "potcars": {"Al": "POTCAR.al", "Mg": "POTCAR.mg"}}
}
```

ABACUS:

The default of `potcar_prefix` is `""`. The path of `potcars/orb_files/deepks_desc` is `potcar_prefix + potcars/orb_files/deepks_desc`.

```
"interaction": {
  "type": "abacus",
  "incar": "abacus_input/INPUT",
  "potcar_prefix": "abacus_input",
  "potcars": {"Al": "pseudo_potential.al", "Mg": "pseudo_potential.
↪mg"},
  "orb_files": {"Al": "numerical_orb.al", "Mg": "numerical_orb.mg"},
  "atom_masses": {"Al": 26.9815, "Mg": 24.305},
  "deepks_desc": "jle.orb"
}
```

deepmd:

Only 1 model can be used in autotest in one working directory.

```
"interaction": {
  "type": "deepmd",
  "model": "frozen_model.pb",
  "type_map": {"Al": 0, "Mg": 1}
}
```

meam:

Please make sure the [USER-MEAMC package](#) has already been installed in LAMMPS.

```
"interaction": {
  "type": "meam",
  "model": ["meam.lib", "AlMg.meam"],
  "type_map": {"Al": 1, "Mg": 2}
}
```

eam_fs & eam_alloy:

Please make sure the [MANYBODY package](#) has already been installed in LAMMPS

```

    "interaction": {
      "type":          "eam_fs (eam_alloy)",
      "model":         "AlMg.eam.fs (AlMg.eam.alloy)",
      "type_map":      {"Al": 1, "Mg": 2}
    }

```

7.1.3 Property type

Now the supported property types are `eos`, `elastic`, `vacancy`, `interstitial`, `surface`, and `gamma`. Before property tests, `relaxation` should be done first or the relaxation results should be present in the corresponding directory `confs/mp-*/relaxation/relax_task`. A file named `task.json` in json format containing the property parameter will be written in the directory of each task after make step. Multiple property tests can be performed simultaneously.

7.2 Make run and post

There are three operations in auto test package, namely `make`, `run`, and `post`. Here we take `eos` property as an example for property type.

7.2.1 Make

The INCAR, POSCAR, POTCAR input files for VASP or `in.lammps`, `conf.lmp`, and the interatomic potential files for LAMMPS will be generated in the directory `confs/mp-*/relaxation/relax_task` for relaxation or `confs/mp-*/eos_00/task.[0-9]*[0-9]` for EOS. The `machine.json` file is not needed for make. Example:

```
dpngen autotest make relaxation.json
```

7.2.2 Run

The jobs would be dispatched according to the parameter in `machine.json` file and the calculation results would be sent back. Example:

```
dpngen autotest run relaxation.json machine.json
```

7.2.3 Post

The post process of calculation results would be performed. `result.json` in json format will be generated in `confs/mp-*/relaxation/relax_task` for relaxation and `result.json` in json format and `result.out` in txt format in `confs/mp-*/eos_00` for EOS. The `machine.json` file is also not needed for post. Example:

```
dpngen autotest post relaxation.json
```

7.3 Relaxation

7.3.1 Relaxation make

The list of the directories storing structures are ["confs/std-*"] in the previous example. For single element system, if POSCAR doesn't exist in the directories: `std-fcc`, `std-hcp`, `std-dhcp`, `std-bcc`, `std-diamond`, and `std-sc`, the package will automatically generate the standard crystal structures **fcc**, **hcp**, **dhcp**, **bcc**, **diamond**, and **sc** in the corresponding directories, respectively. In other conditions and for multi-component system (more than 1), if POSCAR doesn't exist, the package will terminate and print the error **"no configuration for autotest"**.

VASP relaxation

Take the input example of Al in the previous section, when we do make as follows:

```
dpngen autotest make relaxation.json
```

the following files would be generated:

```
tree confs/std-fcc/relaxation/
```

```
confs/std-fcc/relaxation/
|-- INCAR
|-- POTCAR
`-- relax_task
    |-- INCAR -> ../INCAR
    |-- inter.json
    |-- KPOINTS
    |-- POSCAR -> ../../POSCAR
    |-- POTCAR -> ../POTCAR
    `-- task.json
```

`inter.json` records the information in the interaction dictionary and `task.json` records the information in the relaxation dictionary.

LAMMPS relaxation

```
dpngen autotest make relaxation.json
tree confs/std-fcc/
```

the output would be:

```
confs/std-fcc/
|-- POSCAR
`-- relaxation
    |-- frozen_model.pb -> ../../../../frozen_model.pb
    |-- in.lammps
    `-- relax_task
        |-- conf.lmp
        |-- frozen_model.pb -> ../frozen_model.pb
        |-- in.lammps -> ../in.lammps
        |-- inter.json
```

(continues on next page)

(continued from previous page)

```
|-- POSCAR -> ../../POSCAR
|-- task.json
```

the `conf.lmp` is the input configuration and `in.lammps` is the input command file for lammps.

in.lammps: the package would generate the file `confs/mp-*/relaxation/in.lammps` as follows and we refer the user to the further information of `fix box/relax` function in lammps:

```
clear
units                metal
dimension            3
boundary             p p p
atom_style           atomic
box                  tilt large
read_data            conf.lmp
mass                 1 26.982
neigh_modify         every 1 delay 0 check no
pair_style deepmd    frozen_model.pb
pair_coeff
compute              mype all pe
thermo               100
thermo_style         custom step pe pxx pyy pzz pxy pxz pyz lx ly lz vol c_mype
dump                 1 all custom 100 dump.relax id type xs ys zs fx fy fz
min_style            cg
fix                  1 all box/relax iso 0.0
minimize             0 1.000000e-10 5000 500000
fix                  1 all box/relax aniso 0.0
minimize             0 1.000000e-10 5000 500000
variable             N equal count(all)
variable             V equal vol
variable             E equal "c_mype"
variable             tmlpx equal lx
variable             tmlpy equal ly
variable             Pxx equal pxx
variable             Pyy equal pyy
variable             Pzz equal pzz
variable             Pxy equal pxy
variable             Pxz equal pxz
variable             Pyz equal pyz
variable             Epa equal ${E}/${N}
variable             Vpa equal ${V}/${N}
variable             AA equal (${tmlpx}*${tmlpy})
print "All done"
print "Total number of atoms = ${N}"
print "Final energy per atoms = ${Epa}"
print "Final volume per atoms = ${Vpa}"
print "Final Base area = ${AA}"
print "Final Stress (xx yy zz xy xz yz) = ${Pxx} ${Pyy} ${Pzz} ${Pxy} ${Pxz} ${Pyz}"
```

If user provides lammps input command file `in.lammps`, the `thermo_style` and `dump` commands should be the same as the above file.

interatomic potential model: the `frozen_model.pb` in `confs/mp-*/relaxation` would link to the `frozen_model.pb` file given in the input.

7.3.2 Relaxation run

The work path of each task should be in the form like `confs/mp-*/relaxation` and all task is in the form like `confs/mp-*/relaxation/relax_task`.

The `machine.json` file should be applied in this process and the machine parameters (eg. GPU or CPU) are determined according to the task type (VASP or LAMMPS). Then in each work path, the corresponding tasks would be submitted and the results would be sent back through `make_dispatcher`.

Take `deepmd` run for example:

```
nohup dpngen autotest run relaxation.json machine-ali.json > run.result 2>&1 &
tree confs/std-fcc/relaxation/
```

the output would be:

```
confs/std-fcc/relaxation/
|-- frozen_model.pb -> ../../../../frozen_model.pb
|-- in.lammps
|-- jr.json
`-- relax_task
    |-- conf.lmp
    |-- dump.relax
    |-- frozen_model.pb -> ../frozen_model.pb
    |-- in.lammps -> ../in.lammps
    |-- inter.json
    |-- log.lammps
    |-- outlog
    |-- POSCAR -> ../../POSCAR
    `-- task.json
```

`dump.relax` is the file storing configurations and `log.lammps` is the output file for lammps.

7.3.3 Relaxation post

Take `deepmd` post for example:

```
dpngen autotest post relaxation.json
tree confs/std-fcc/relaxation/
```

the output will be:

```
confs/std-fcc/relaxation/
|-- frozen_model.pb -> ../../../../frozen_model.pb
|-- in.lammps
|-- jr.json
`-- relax_task
    |-- conf.lmp
    |-- CONTCAR
    |-- dump.relax
    |-- frozen_model.pb -> ../frozen_model.pb
    |-- in.lammps -> ../in.lammps
    |-- inter.json
    |-- log.lammps
```

(continues on next page)

(continued from previous page)

```

|-- outlog
|-- POSCAR -> ../../POSCAR
|-- result.json
`-- task.json

```

result.json stores the box cell, coordinates, energy, force, virial,... information of each frame in the relaxation trajectory and CONTCAR is the final equilibrium configuration.

result.json:

```

{
  "@module": "dpdata.system",
  "@class": "LabeledSystem",
  "data": {
    "atom_nums": [
      1
    ],
    "atom_names": [
      "Al"
    ],
    "atom_types": {
      "@module": "numpy",
      "@class": "array",
      "dtype": "int64",
      "data": [
        0
      ]
    },
    "orig": {
      "@module": "numpy",
      "@class": "array",
      "dtype": "int64",
      "data": [
        0,
        0,
        0
      ]
    },
    "cells": {
      "@module": "numpy",
      "@class": "array",
      "dtype": "float64",
      "data": [
        [
          2.8637824638,
          0.0,
          0.0
        ],
        [
          1.4318912319,
          2.4801083646,
          0.0
        ]
      ]
    }
  }
}

```

(continues on next page)

(continued from previous page)

```

        ],
        [
            1.4318912319,
            0.8267027882,
            2.3382685902
        ]
    ],
    [
        [
            2.8549207998018438,
            0.0,
            0.0
        ],
        [
            1.4274603999009239,
            2.472433938457684,
            0.0
        ],
        [
            1.4274603999009212,
            0.8241446461525599,
            2.331033071844216
        ]
    ],
    [
        [
            2.854920788303194,
            0.0,
            0.0
        ],
        [
            1.427460394144466,
            2.472433928487206,
            0.0
        ],
        [
            1.427460394154763,
            0.8241446428350139,
            2.331033062460779
        ]
    ]
]
},
"coords": {
    "@module": "numpy",
    "@class": "array",
    "dtype": "float64",
    "data": [
        [
            0.0,
            0.0,

```

(continues on next page)

(continued from previous page)

```

        0.0
    ],
    [
        [
            5.709841595683707e-25,
            -4.3367974740910857e-19,
            0.0
        ]
    ],
    [
        [
            -8.673606219968035e-19,
            8.673619637565944e-19,
            8.673610853102186e-19
        ]
    ]
],
{
    "energies": {
        "@module": "numpy",
        "@class": "array",
        "dtype": "float64",
        "data": [
            -3.745029,
            -3.7453815,
            -3.7453815
        ]
    },
    "forces": {
        "@module": "numpy",
        "@class": "array",
        "dtype": "float64",
        "data": [
            [
                [
                    0.0,
                    -6.93889e-18,
                    -3.46945e-18
                ]
            ],
            [
                [
                    1.38778e-17,
                    6.93889e-18,
                    -1.73472e-17
                ]
            ],
            [
                [
                    1.38778e-17,
                    1.73472e-17,

```

(continues on next page)

(continued from previous page)

```

        -4.51028e-17
    ]
]
},
"virials": {
    "@module": "numpy",
    "@class": "array",
    "dtype": "float64",
    "data": [
        [
            [
                -0.07534992071654338,
                1.2156615579052586e-17,
                1.3904892126132796e-17
            ],
            [
                1.2156615579052586e-17,
                -0.07534992071654338,
                4.61571024026576e-12
            ],
            [
                1.3904892126132796e-17,
                4.61571024026576e-12,
                -0.07534992071654338
            ]
        ],
        [
            [
                -9.978994290457664e-08,
                -3.396452753975288e-15,
                8.785831629151552e-16
            ],
            [
                -3.396452753975288e-15,
                -9.991375413666671e-08,
                5.4790751628409565e-12
            ],
            [
                8.785831629151552e-16,
                5.4790751628409565e-12,
                -9.973497959053003e-08
            ]
        ],
        [
            [
                1.506940521266962e-11,
                1.1152016233536118e-11,
                -8.231900529157644e-12
            ],
            [
                1.1152016233536118e-11,

```

(continues on next page)

(continued from previous page)

```

        -6.517665029355618e-11,
        -6.33706710415926e-12
    ],
    [
        -8.231900529157644e-12,
        -6.33706710415926e-12,
        5.0011471096530724e-11
    ]
]
],
"stress": {
    "@module": "numpy",
    "@class": "array",
    "dtype": "float64",
    "data": [
        [
            [
                -7.26922500000000005,
                1.1727839e-15,
                1.3414452e-15
            ],
            [
                1.1727839e-15,
                -7.26922500000000005,
                4.45290930000000003e-10
            ],
            [
                1.3414452e-15,
                4.45290930000000003e-10,
                -7.26922500000000005
            ]
        ],
        [
            [
                -9.71695e-06,
                -3.3072633e-13,
                8.5551193e-14
            ],
            [
                -3.3072633e-13,
                -9.72900600000000001e-06,
                5.3351969e-10
            ],
            [
                8.5551193e-14,
                5.3351969e-10,
                -9.711598e-06
            ]
        ]
    ],
    [
        [

```

(continues on next page)

(continued from previous page)

```

    1.4673689e-09,
    1.0859169e-09,
    -8.0157343e-10
  ],
  [
    1.0859169e-09,
    -6.3465139e-09,
    -6.1706584e-10
  ],
  [
    -8.0157343e-10,
    -6.1706584e-10,
    4.8698191e-09
  ]
]
}
}
}
}
}
}
```

7.4 Property

7.4.1 Property get started and input examples

Here we take deepmd for example and the input file for other task types is similar.

```
{
  "structures": ["confs/std-*"],
  "interaction": {
    "type": "deepmd",
    "model": "frozen_model.pb",
    "type_map": {"Al": 0}
  },
  "properties": [
    {
      "type": "eos",
      "vol_start": 0.9,
      "vol_end": 1.1,
      "vol_step": 0.01
    },
    {
      "type": "elastic",
      "norm_deform": 1e-2,
      "shear_deform": 1e-2
    },
    {
      "type": "vacancy",
      "supercell": [3, 3, 3],
      "start_confs_path": "../vasp/confs"
    }
  ],
}
```

(continues on next page)

(continued from previous page)

```

{
  "type": "interstitial",
  "supercell": [3, 3, 3],
  "insert_ele": ["Al"],
  "conf_filters": {"min_dist": 1.5},
  "cal_setting": {"input_prop": "lammps_input/lammps_high"}
},
{
  "type": "surface",
  "min_slab_size": 10,
  "min_vacuum_size": 11,
  "max_miller": 2,
  "cal_type": "static"
},
{
  "type": "gamma",
  "lattice_type": "fcc",
  "miller_index": [1, 1, 1],
  "displace_direction": [1, 1, 0],
  "supercell_size": [1, 1, 10],
  "min_vacuum_size": 10,
  "add_fix": ["true", "true", "false"],
  "n_steps": 20
}
]
}

```

Universal key words for properties

Key words	data structure	example	description
type	String	"eos"	property type
skip	Boolean	true	whether to skip current property or not
start_confs_path	String	"../vasp/confs"	start from the equilibrium configuration in other path only for the current property type
cal_setting["input_prop"]	String	"lammps_input/lammps_high"	input commands file
cal_setting["overwrite_interaction"]	Boolean		overwrite the interaction in the interaction part only for the current property type

other parameters in cal_setting and cal_type in relaxation also apply in property.

Key words for EOS

Key words	data structure	example	description
vol_start	Float	0.9	the starting volume related to the equilibrium structure
vol_end	Float	1.1	the biggest volume related to the equilibrium structure
vol_step	Float	0.01	the volume increment related to the equilibrium structure
vol_abs	Boolean	false	whether to treat vol_start, vol_end and vol_step as absolute volume or not (as relative volume), default = false

Key words for **Elastic**

Key words	data structure	example	description
norm_deform	Float	1e-2	deformation in xx, yy, zz, default = 1e-2
shear_deform	Float	1e-2	deformation in other directions, default = 1e-2

Key words for **Vacancy**

Key words	data structure	example	description
supercell	List of Int	[3,3,3]	the supercell to be constructed, default = [1,1,1]

Key words for **Interstitial**

Key words	data structure	example	description
in- sert_ele	List of String	["Al"]	the element to be inserted
supercell	List of Int	[3,3,3]	the supercell to be constructed, default = [1,1,1]
conf_filters	Dict	"min_dist": 1.5	filter out the undesirable configuration
bcc_self	Boolean	false	whether to do the self-interstitial calculations for bcc structures, default = false

Key words for **Surface**

Key words	data structure	example	description
min_slab_size	Int	10	minimum size of slab thickness
min_vacuum_size	Int	11	minimum size of vacuum width
pert_xz	Float	0.01	perturbation through xz direction used to compute surface energy, default = 0.01
max_miller	Int	2	the maximum miller index, default = 2

Key words for **Gamma**

Key words	data structure	example	description
lattice_type	String	"fcc"	"bcc" or "fcc" at this stage
miller_index	List of Int	[1,1,1]	slip plane for gamma-line calculation
dis- place_direction	List of Int	[1,1,0]	slip direction for gamma-line calculation
supercell_size	List of Int	[1,1,10]	the supercell to be constructed, default = [1,1,5]
min_vacuum_size	Int or Float	10	minimum size of vacuum width, default = 20
add_fix	List of String	['true','true','false']	whether to fix atoms in the direction, default = ['true','true','false'] (standard method)
n_steps	Int	20	Number of points for gamma-line calculation, default = 10

7.4.2 Property make

```
dpngen autotest make property.json
```

EOS output:

```
confs/std-fcc/eos_00/
|-- frozen_model.pb -> ../../../../frozen_model.pb
|-- task.000000
|   |-- conf.lmp
|   |-- eos.json
|   |-- frozen_model.pb -> ../frozen_model.pb
|   |-- in.lammps
|   |-- inter.json
|   |-- POSCAR
|   |-- POSCAR.orig -> ../../relaxation/relax_task/CONTCAR
|   `-- task.json
|-- task.000001
|   |-- conf.lmp
|   |-- eos.json
|   |-- frozen_model.pb -> ../frozen_model.pb
|   |-- in.lammps
|   |-- inter.json
|   |-- POSCAR
|   |-- POSCAR.orig -> ../../relaxation/relax_task/CONTCAR
|   `-- task.json
...
`-- task.000019
    |-- conf.lmp
    |-- eos.json
    |-- frozen_model.pb -> ../frozen_model.pb
    |-- in.lammps
    |-- inter.json
    |-- POSCAR
    |-- POSCAR.orig -> ../../relaxation/relax_task/CONTCAR
    `-- task.json
```

eos.json records the volume and scale of the corresponding task.

Elastic output:

```
confs/std-fcc/elastic_00/
|-- equi.stress.json
|-- frozen_model.pb -> ../../../../frozen_model.pb
|-- in.lammps
|-- POSCAR -> ../relaxation/relax_task/CONTCAR
|-- task.000000
|   |-- conf.lmp
|   |-- frozen_model.pb -> ../frozen_model.pb
|   |-- in.lammps -> ../in.lammps
|   |-- inter.json
|   |-- POSCAR
|   |-- strain.json
|   `-- task.json
```

(continues on next page)

(continued from previous page)

```

|-- task.000001
|   |-- conf.lmp
|   |-- frozen_model.pb -> ../frozen_model.pb
|   |-- in.lammps -> ../in.lammps
|   |-- inter.json
|   |-- POSCAR
|   |-- strain.json
|   `-- task.json
...
`-- task.000023
    |-- conf.lmp
    |-- frozen_model.pb -> ../frozen_model.pb
    |-- in.lammps -> ../in.lammps
    |-- inter.json
    |-- POSCAR
    |-- strain.json
    `-- task.json

```

`equi.stress.json` records the stress information of the equilibrium task and `strain.json` records the deformation information of the corresponding task.

Vacancy output:

```

confs/std-fcc/vacancy_00/
|-- frozen_model.pb -> ../../../frozen_model.pb
|-- in.lammps
|-- POSCAR -> ../relaxation/relax_task/CONTCAR
`-- task.000000
    |-- conf.lmp
    |-- frozen_model.pb -> ../frozen_model.pb
    |-- in.lammps -> ../in.lammps
    |-- inter.json
    |-- POSCAR
    |-- supercell.json
    `-- task.json

```

`supercell.json` records the supercell size information of the corresponding task.

Interstitial output:

```

confs/std-fcc/interstitial_00/
|-- element.out
|-- frozen_model.pb -> ../../../frozen_model.pb
|-- in.lammps
|-- POSCAR -> ../relaxation/relax_task/CONTCAR
|-- task.000000
|   |-- conf.lmp
|   |-- frozen_model.pb -> ../frozen_model.pb
|   |-- in.lammps -> ../in.lammps
|   |-- inter.json
|   |-- POSCAR
|   |-- supercell.json
|   `-- task.json

```

(continues on next page)

(continued from previous page)

```

`-- task.000001
  |-- conf.lmp
  |-- frozen_model.pb -> ../frozen_model.pb
  |-- in.lammps -> ../in.lammps
  |-- inter.json
  |-- POSCAR
  |-- supercell.json
  `-- task.json

```

element.out records the inserted element type of each task and supercell.json records the supercell size information of the corresponding task.

Surface output:

```

confs/std-fcc/surface_00/
|-- frozen_model.pb -> ../../../frozen_model.pb
|-- in.lammps
|-- POSCAR -> ../relaxation/relax_task/CONTCAR
|-- task.000000
|   |-- conf.lmp
|   |-- frozen_model.pb -> ../frozen_model.pb
|   |-- in.lammps -> ../in.lammps
|   |-- inter.json
|   |-- miller.json
|   |-- POSCAR
|   |-- POSCAR.tmp
|   `-- task.json
|-- task.000001
|   |-- conf.lmp
|   |-- frozen_model.pb -> ../frozen_model.pb
|   |-- in.lammps -> ../in.lammps
|   |-- inter.json
|   |-- miller.json
|   |-- POSCAR
|   |-- POSCAR.tmp
|   `-- task.json
...
`-- task.000008
    |-- conf.lmp
    |-- frozen_model.pb -> ../frozen_model.pb
    |-- in.lammps -> ../in.lammps
    |-- inter.json
    |-- miller.json
    |-- POSCAR
    |-- POSCAR.tmp
    `-- task.json

```

miller.json records the miller index of the corresponding task.

7.4.3 Property run

```
nohup dpngen autotest run property.json machine-ali.json > run.result 2>&1 &
```

the result file `log.lammps`, `dump.relax`, and `outlog` would be sent back.

7.4.4 Property-post

Use command

```
dpngen autotest post property.json
```

to post results as `result.json` and `result.out` in each property's path.

7.4.5 Properties

EOS get started and input examples

Equation of State (EOS) here calculates the energies of the most stable structures as a function of volume. Users can refer to Figure 4 of the [dpngen CPC paper](#) for more information of EOS.

An example of the input file for EOS by VASP:

```
{
  "structures":      ["confs/mp-*", "confs/std-*", "confs/test-*"],
  "interaction": {
    "type":          "vasp",
    "incar":         "vasp_input/INCAR",
    "potcar_prefix": "vasp_input",
    "potcars":       {"Al": "POTCAR.al", "Mg": "POTCAR.mg"}
  },
  "properties": [
    {
      "type":         "eos",
      "vol_start":    0.9,
      "vol_end":      1.1,
      "vol_step":     0.01
    }
  ]
}
```

`vol_start` is the starting volume relative to the equilibrium structure, `vol_step` is the volume increment step relative to the equilibrium structure, and the biggest relative volume is smaller than `vol_end`.

EOS make

Step 1. Before make in EOS, the equilibrium configuration CONTCAR must be present in `confs/mp-*/relaxation`.

Step 2. For the input example in the previous section, when we do make, 40 tasks would be generated as `confs/mp-*/eos_00/task.000000`, `confs/mp-*/eos_00/task.000001`, ..., `confs/mp-*/eos_00/task.000039`. The suffix 00 is used for possible refine later.

Step 3. If the task directory, for example `confs/mp-*/eos_00/task.000000` is not empty, the old input files in it including INCAR, POSCAR, POTCAR, `conf.lmp`, `in.lammps` would be deleted.

Step 4. In each task directory, `POSCAR.orig` would link to `confs/mp-*/relaxation/CONTCAR`. Then the scale parameter can be calculated as:

```
scale = (vol_current / vol_equi) ** (1. / 3.)
```

`vol_current` is the corresponding volume per atom of the current task and `vol_equi` is the volume per atom of the equilibrium configuration. Then the `poscar_scale` function in `dpgen.auto_test.lib.vasp` module would help to generate POSCAR file with `vol_current` in `confs/mp-*/eos_00/task.[0-9]*[0-9]`.

Step 5. According to the task type, the input file including INCAR, POTCAR or `conf.lmp`, `in.lammps` would be written in every `confs/mp-*/eos_00/task.[0-9]*[0-9]`.

EOS run

The work path of each task should be in the form like `confs/mp-*/eos_00` and all task is in the form like `confs/mp-*/eos_00/task.[0-9]*[0-9]`.

When we dispatch tasks, we would go through every individual work path in the list `confs/mp-*/eos_00`, and then submit `task.[0-9]*[0-9]` in each work path.

EOS post

The post processing of EOS would go to every directory in `confs/mp-*/eos_00` and do the post processing. Let's suppose we are now in `confs/mp-100/eos_00` and there are `task.000000`, `task.000001`, ..., `task.000039` in this directory. By reading `inter.json` file in every task directory, the task type can be determined and the energy and force information of every task can further be obtained. By appending the dict of energy and force into a list, an example of the list with 1 atom is given as:

```
[
  {"energy": E1, "force": [fx1, fy1, fz1]},
  {"energy": E2, "force": [fx2, fy2, fz2]},
  ...
  {"energy": E40, "force": [fx40, fy40, fz40]}
]
```

Then the volume can be calculated from the task id and the corresponding energy can be obtained from the list above. Finally, there would be `result.json` in json format and `result.out` in txt format in `confs/mp-100/eos_00` containing the EOS results.

An example of `result.json` is give as:

```
{
  "14.808453313267595": -3.7194474,
  "14.972991683415014": -3.7242038,
```

(continues on next page)

(continued from previous page)

```

...
"17.934682346068534": -3.7087655
}

```

An example of `result.out` is given below:

```

onf_dir: /root/auto_test_example/deepmd/confs/std-fcc/eos_00
VpA(A^3)  EpA(eV)
14.808    -3.7194
14.973    -3.7242
...
17.935    -3.7088

```

Elastic get started and input examples

Here we calculate the mechanical properties which include elastic constants (C11 to C66), bulk modulus Bv, shear modulus Gv, Youngs modulus Ev, and Poission ratio Uv of a certain crystal structure.

An example of the input file for Elastic by deepmd:

```

{
  "structures": ["confs/mp-*", "confs/std-*", "confs/test-*"],
  "interaction": {
    "type": "deepmd",
    "model": "frozen_model.pb",
    "type_map": {"Al": 0, "Mg": 1}
  },
  "properties": [
    {
      "type": "elastic",
      "norm_deform": 1e-2,
      "shear_deform": 1e-2
    }
  ]
}

```

Here the default values of `norm_deform` and `shear_deform` are **1e-2** and **1e-2**, respectively. A list of `norm_strains` and `shear_strains` would be generated as below:

```

[-norm_def, -0.5 * norm_def, 0.5 * norm_def, norm_def]
[-shear_def, -0.5 * shear_def, 0.5 * shear_def, shear_def]

```


Elastic make

Step 1. The `DeformedStructureSet` module in `pymatgen.analysis.elasticity.strain` is used to generate a set of independently deformed structures. `equi.stress.out` file is written to record the equilibrium stress in the Elastic directory. For the example in the previous section, `equi.stress.out` should be in `confs/mp-*/elastic_00`.

Step 2. If there are `init_from_suffix` and `output_suffix` parameter in the `properties` part, the *refine process* follows. Else, the deformed structure (POSCAR) and strain information (`strain.out`) are written in the task directory, for example, in `confs/mp-*/elastic_00/task.000000`.

Step 3. When doing elastic by VASP, `ISIF=2`. When doing by LAMMPS, the following `in.lammps` would be written.

```
units          metal
dimension      3
boundary       p      p      p
atom_style     atomic
box            tilt large
read_data      conf.lmp
mass           1 1
mass           2 1
neigh_modify   every 1 delay 0 check no
pair_style     deepmd frozen_model.pb
pair_coeff
compute        mype all pe
thermo         100
thermo_style   custom step pe pxx pyy pzz pxy pxz pyz lx ly lz vol c_mype
dump           1 all custom 100 dump.relax id type xs ys zs fx fy fz
min_style      cg
minimize       0 1.0000000e-10 5000 500000
variable       N equal count(all)
variable       V equal vol
variable       E equal "c_mype"
variable       Pxx equal pxx
variable       Pyy equal pyy
variable       Pzz equal pzz
variable       Pxy equal pxy
variable       Pxz equal pxz
variable       Pyz equal pyz
variable       Epa equal ${E}/${N}
variable       Vpa equal ${V}/${N}
print "All done"
print "Total number of atoms = ${N}"
print "Final energy per atoms = ${Epa}"
print "Final volume per atoms = ${Vpa}"
print "Final Stress (xx yy zz xy xz yz) = ${Pxx} ${Pyy} ${Pzz} ${Pxy} ${Pxz} ${Pyz}"
```

Elastic run

Very similar to the `run` operation of EOS except for in different directories. Now the work path of each task should be in the form like `confs/mp-*/elastic_00` and all task is in the form like `confs/mp-*/elastic_00/task.[0-9]*[0-9]`.

Elastic post

The `ElasticTensor` module in `pymatgen.analysis.elasticity.elastic` is used to get the elastic tensor, `Bv`, and `Gv`. The mechanical properties of a crystal structure would be written in `result.json` in json format and `result.out` in txt format. The example of the output file is give below.

result.json

```
{
  "elastic_tensor": [
    134.90955999999997,
    54.329958699999985,
    51.802386099999985,
    3.5745279599999993,
    -1.3886325999999648e-05,
    -1.9638233999999486e-05,
    54.55840299999999,
    134.59654699999996,
    51.7972336,
    -3.53972684,
    1.839568799999963e-05,
    8.756799399999951e-05,
    51.91324859999999,
    51.913292199999994,
    137.01763799999998,
    -5.090339399999969e-05,
    6.99251629999996e-05,
    3.736478699999946e-05,
    3.8780564440000007,
    -3.770445632,
    -1.2766205999999956,
    35.41343199999999,
    2.24795908000000023e-05,
    1.38376920000000172e-06,
    -4.959999999495933e-06,
    2.5800000003918792e-06,
    1.48000000030874965e-06,
    2.90000000008417968e-06,
    35.375960199999994,
    3.8608356,
    0.0,
    0.0,
    0.0,
    0.0,
    4.02554856,
```

(continues on next page)

(continued from previous page)

```

    38.375018399999995
  ],
  "BV": 80.3153630222222,
  "GV": 38.40582656,
  "EV": 99.37716395728943,
  "uV": 0.2937771799031088
}

```

The order of `elastic_tensor` is C11, C12, ..., C16, C21, C22, ..., C26, ..., C66 and the unit of Bv, Gv, Ev, and uv is GPa.

result.out

```

/root/auto_test_example/deepmd/confs/std-fcc/elastic_00
134.91  54.33  51.80   3.57  -0.00  -0.00
 54.56 134.60  51.80  -3.54   0.00   0.00
 51.91  51.91 137.02  -0.00   0.00   0.00
  3.88  -3.77  -1.28  35.41   0.00   0.00
-0.00   0.00   0.00   0.00  35.38   3.86
 0.00   0.00   0.00   0.00   4.03  38.38
# Bulk   Modulus BV = 80.32 GPa
# Shear  Modulus GV = 38.41 GPa
# Youngs Modulus EV = 99.38 GPa
# Poission Ratio uV = 0.29

```

Vacancy get started and input examples

Vacancy calculates the energy difference when removing an atom from the crystal structure. We only need to give the information of `supercell` to help calculate the vacancy energy and the default value of `supercell` is [1, 1, 1].

An example of the input file for Vacancy by deepmd:

```

{
  "structures":      "confs/mp-*",
  "interaction": {
    "type":          "deepmd",
    "model":          "frozen_model.pb",
    "type_map":       {"Al": 0, "Mg": 1}
  },
  "properties": [
    {
      "type":         "vacancy",
      "supercell":     [1, 1, 1]
    }
  ]
}

```

Vacancy make

Step 1. The `VacancyGenerator` module in `pymatgen.analysis.defects.generators` is used to generate a set of structures with vacancy.

Step 2. If there are `init_from_suffix` and `output_suffix` parameter in the properties part, the *refine process* follows. If `reproduce` is evoked, the *reproduce process* follows. Otherwise, the vacancy structure (POSCAR) and supercell information (`supercell.out`) are written in the task directory, for example, in `confs/mp-*/vacancy_00/task.000000` with the check and possible removing of the old input files like before.

Step 3. When doing vacancy by VASP, `ISIF = 3`. When doing vacancy by LAMMPS, the same `in.lammps` as that in *EOS* (*change_box is True*) would be generated with `scale` set to one.

Vacancy run

Very similar to the run operation of EOS except for in different directories. Now the work path of each task should be in the form like `confs/mp-*/vacancy_00` and all task is in the form like `confs/mp-*/vacancy_00/task.[0-9]*[0-9]`.

Vacancy post

For Vacancy, we need to calculate the energy difference between a crystal structure with and without a vacancy. The examples of the output files `result.json` in json format and `result.out` in txt format are given below.

result.json

```
{
  "[3, 3, 3]-task.000000": [
    0.7352769999999964,
    -96.644642,
    -97.379919
  ]
}
```

result.out

```
/root/auto_test_example/deepmd/confs/std-fcc/vacancy_00
Structure:          Vac_E(eV)  E(eV)  equi_E(eV)
[3, 3, 3]-task.000000:  0.735  -96.645 -97.380
```

Interstitial get started and input examples

Interstitial calculates the energy difference when adding an atom into the crystal structure. We need to give the information of `supercell` (default value is [1, 1, 1]) and `insert_ele` list for the element types of the atoms added in.

An example of the input file for Interstitial by deepmd:

```
{
  "structures":      "confs/mp-*",
  "interaction": {
    "type":          "deepmd",
    "model":          "frozen_model.pb",
    "type_map":       {"Al": 0, "Mg": 1}
  },
  "properties": [
    {
      "type":         "interstitial",
      "supercell":     [3, 3, 3],
      "insert_ele":    ["Al"],
      "conf_filters": {"min_dist": 1.5},
      "cal_setting":   {"input_prop": "lammps_input/lammps_high"}
    }
  ]
}
```

We add a `conf_filters` parameter in `properties` part and this parameter can help to eliminate undesirable structure which can render rather difficult convergence in calculations. In the example above, “**min_dist**”: 1.5 means if the smallest atomic distance in the structure is less than 1.5 angstrom, the configuration would be eliminated and not used in calculations.

Interstitial make

Step 1. For each element in `insert_ele` list, `InterstitialGenerator` module in `pymatgen.analysis.defects.generators` would help to generate interstitial structure. The structure would be appended into a list if it can meet the requirements in `conf_filters`.

Step 2. If `refine` is True, we do *refine process*. If `reprod-opt` is True (the default is **False**), we do *reproduce process*. Else, the vacancy structure (POSCAR) and supercell information (`supercell.out`) are written in the task directory, for example, in `confs/mp-*/interstitial_00/task.000000` with the check and possible removing of the old input files like before.

Step 3. In interstitial by VASP, `ISIF = 3`. In interstitial by LAMMPS, the same `in.lammps` as that in *EOS* (*change_box is True*) would be generated with `scale` set to one.

Interstitial run

Very similar to the run operation of EOS except for in different directories. Now the work path of each task should be in the form like `confs/mp-*/interstitial_00` and all task is in the form like `confs/mp-*/interstitial_00/task.[0-9]*[0-9]`.

Interstitial post

For `Interstitial`, we need to calculate the energy difference between a crystal structure with and without atom added in. The examples of the output files `result.json` in json format and `result.out` in txt format are given below.

result.json

```
{
  "Al-[3, 3, 3]-task.000000": [
    4.0229520000000004,
    -100.84773,
    -104.870682
  ],
  "Al-[3, 3, 3]-task.000001": [
    2.78295200000000088,
    -102.08773,
    -104.870682
  ]
}
```

result.out

```
/root/auto_test_example/deepmd/confs/std-fcc/interstitial_00
Insert_ele-Struct: Inter_E(eV)  E(eV)  equi_E(eV)
Al-[3, 3, 3]-task.000000:   4.023   -100.848  -104.871
Al-[3, 3, 3]-task.000001:   2.783   -102.088  -104.871
```

Surface get started and input examples

Surface calculates the surface energy. We need to give the information of `min_slab_size`, `min_vacuum_size`, `max_miller` (default value is 2), and `pert_xz` which means perturbations in xz and will help work around vasp bug.

An example of the input file for Surface by deepmd:

```
{
  "structures":      "confs/mp-*",
  "interaction": {
    "type":          "deepmd",
    "model":          "frozen_model.pb",
    "type_map":      {"Al": 0, "Mg": 1}
  },
}
```

(continues on next page)

(continued from previous page)

```

    "properties": [
      {
        "type": "surface",
        "min_slab_size": 10,
        "min_vacuum_size": 11,
        "max_miller": 2,
        "cal_type": "static"
      }
    ]
  }
}

```

Surface make

Step 1. Based on the equilibrium configuration, `generate_all_slabs` module in `pymatgen.core.surface` would help to generate surface structure list with using `max_miller`, `min_slab_size`, and `min_vacuum_size` parameters.

Step 2. If `refine` is True, we do *refine process*. If `reprod-opt` is True (the default is False), we do *reproduce process*. Otherwise, the surface structure (POSCAR) with perturbations in xz and miller index information (`miller.out`) are written in the task directory, for example, in `confs/mp-*/interstitial_00/task.000000` with the check and possible removing of the old input files like before.

Surface run

Very similar to the `run` operation of EOS except for in different directories. Now the work path of each task should be in the form like `confs/mp-*/surface_00` and all task is in the form like `confs/mp-*/surface_00/task.[0-9]*[0-9]`.

Surface post

For Surface, we need to calculate the energy difference between a crystal structure with and without a surface with a certain miller index divided by the surface area.

The examples of the output files `result.json` in json format and `result.out` in txt format are given below.

result.json

```

{
  "[1, 1, 1]-task.000000": [
    0.8051037974207992,
    -3.6035018,
    -3.7453815
  ],
  "[2, 2, 1]-task.000001": [
    0.9913881928811771,
    -3.5781115999999997,
    -3.7453815
  ],
  "[1, 1, 0]-task.000002": [
    0.9457333586026173,

```

(continues on next page)

(continued from previous page)

```

        -3.5529366000000002,
        -3.7453815
    ],
    "[2, 2, -1]-task.000003": [
        0.9868013100872397,
        -3.5590607142857142,
        -3.7453815
    ],
    "[2, 1, 1]-task.000004": [
        1.0138239046484236,
        -3.563035875,
        -3.7453815
    ],
    "[2, 1, -1]-task.000005": [
        1.0661817319108005,
        -3.5432459166666668,
        -3.7453815
    ],
    "[2, 1, -2]-task.000006": [
        1.034003253044026,
        -3.550884125,
        -3.7453815
    ],
    "[2, 0, -1]-task.000007": [
        0.9569958287615818,
        -3.5685403333333334,
        -3.7453815
    ],
    "[2, -1, -1]-task.000008": [
        0.9432935501134583,
        -3.5774615714285716,
        -3.7453815
    ]
}

```

result.out

```

/root/auto_test_example/deepmd/confs/std-fcc/surface_00
Miller_Indices:      Surf_E(J/m^2)  EpA(eV)  equi_EpA(eV)
[1, 1, 1]-task.000000:      0.805      -3.604      -3.745
[2, 2, 1]-task.000001:      0.991      -3.578      -3.745
[1, 1, 0]-task.000002:      0.946      -3.553      -3.745
[2, 2, -1]-task.000003:      0.987      -3.559      -3.745
[2, 1, 1]-task.000004:      1.014      -3.563      -3.745
[2, 1, -1]-task.000005:      1.066      -3.543      -3.745
[2, 1, -2]-task.000006:      1.034      -3.551      -3.745
[2, 0, -1]-task.000007:      0.957      -3.569      -3.745
[2, -1, -1]-task.000008:      0.943      -3.577      -3.745

```


7.5 Refine

7.5.1 Refine get started and input examples

Sometimes we want to refine the calculation of a property from previous results. For example, when higher convergence criteria EDIFF and EDIFFG are necessary in VASP, the new VASP calculation is desired to start from the previous output configuration, rather than starting from scratch.

An example of the input file `refine.json` is given below:

```
{
  "structures":      ["confs/std-*"],
  "interaction": {
    "type":          "deepmd",
    "model":          "frozen_model.pb",
    "type_map":       {"Al": 0}
  },
  "properties": [
    {
      "type":          "vacancy",
      "init_from_suffix": "00",
      "output_suffix":   "01",
      "cal_setting":     {"input_prop": "lammps_input/lammps_high"}
    }
  ]
}
```

In this example, `refine` would output the results to `vacancy_01` based on the previous results in `vacancy_00` by using a different input commands file for lammps.

7.5.2 Refine make

```
dpngen autotest make refine.json
tree confs/std-fcc/vacancy_01/
```

the output will be:

```
confs/std-fcc/vacancy_01/
|-- frozen_model.pb -> ../../../../frozen_model.pb
|-- in.lammps
`-- task.000000
    |-- conf.lmp
    |-- frozen_model.pb -> ../frozen_model.pb
    |-- in.lammps -> ../in.lammps
    |-- inter.json
    |-- POSCAR -> ../../vacancy_00/task.000000/CONTCAR
    |-- supercell.json -> ../../vacancy_00/task.000000/supercell.json
    `-- task.json
```

an new directory `vacancy_01` would be established and the starting configuration links to previous results.

7.5.3 Refine run

```
nohup dpngen autotest run refine.json machine-ali.json > run.result 2>&1 &
```

the run process of `refine` is similar to before.

7.5.4 Refine post

```
dpngen autotest post refine.json
```

the post process of `refine` is similar to the corresponding property.

7.6 Reproduce

7.6.1 Reproduce get started and input examples

Sometimes we want to reproduce the initial results with the same configurations for cross validation. This version of autotest package can accomplish this successfully in all property types except for `Elastic`. An input example for using `deepmd` to reproduce the VASP Interstitial results is given below:

```
{
  "structures":      ["confs/std-*"],
  "interaction": {
    "type":          "deepmd",
    "model":          "frozen_model.pb",
    "type_map":       {"Al": 0}
  },
  "properties": [
    {
      "type":          "interstitial",
      "reproduce":       true,
      "init_from_suffix": "00",
      "init_data_path": "../vasp/confs",
      "reprod_last_frame": false
    }
  ]
}
```

`reproduce` denotes whether to do reproduce or not and the default value is `False`.

`init_data_path` is the path of VASP or LAMMPS initial data to be reproduced. `init_from_suffix` is the suffix of the initial data and the default value is `"00"`. In this case, the VASP Interstitial results are stored in `../vasp/confs/std-*/interstitial_00` and the reproduced Interstitial results would be in `deepmd/confs/std-*/interstitial_reprod`.

`reprod_last_frame` denotes if only the last frame is used in reproduce. The default value is `True` for `eos` and `surface`, but is `False` for `vacancy` and `interstitial`.

7.6.2 Reproduce make

```
dpngen autotest make reproduce.json
tree confs/std-fcc/interstitial_reprod/
```

the output will be:

```
confs/std-fcc/interstitial_reprod/
|-- frozen_model.pb -> ../../../../frozen_model.pb
|-- in.lammps
|-- task.000000
|   |-- conf.lmp
|   |-- frozen_model.pb -> ../frozen_model.pb
|   |-- in.lammps -> ../in.lammps
|   |-- inter.json
|   |-- POSCAR
|   `-- task.json
|-- task.000001
|   |-- conf.lmp
|   |-- frozen_model.pb -> ../frozen_model.pb
|   |-- in.lammps -> ../in.lammps
|   |-- inter.json
|   |-- POSCAR
|   `-- task.json
...
`-- task.000038
    |-- conf.lmp
    |-- frozen_model.pb -> ../frozen_model.pb
    |-- in.lammps -> ../in.lammps
    |-- inter.json
    |-- POSCAR
    `-- task.json
```

every single frame in the initial data is split into each task and the following `in.lammps` would help to do the static calculation:

```
clear
units                metal
dimension            3
boundary             p p p
atom_style           atomic
box                  tilt large
read_data            conf.lmp
mass                 1 26.982
neigh_modify         every 1 delay 0 check no
pair_style deepmd    frozen_model.pb
pair_coeff
compute              mype all pe
thermo               100
thermo_style         custom step pe pxx pyy pzz pxy pxz pyz lx ly lz vol c_mype
dump                 1 all custom 100 dump.relax id type xs ys zs fx fy fz
run                  0
variable             N equal count(all)
```

(continues on next page)

(continued from previous page)

```

variable      V equal vol
variable      E equal "c_mype"
variable      tmplx equal lx
variable      tmply equal ly
variable      Pxx equal pxx
variable      Pyy equal pyy
variable      Pzz equal pzz
variable      Pxy equal pxy
variable      Pxz equal pxz
variable      Pyz equal pyz
variable      Epa equal ${E}/${N}
variable      Vpa equal ${V}/${N}
variable      AA equal (${tmplx}*${tmply})
print "All done"
print "Total number of atoms = ${N}"
print "Final energy per atoms = ${Epa}"
print "Final volume per atoms = ${Vpa}"
print "Final Base area = ${AA}"
print "Final Stress (xx yy zz xy xz yz) = ${Pxx} ${Pyy} ${Pzz} ${Pxy} ${Pxz} ${Pyz}"

```

7.6.3 Reproduce run

```
nohup dpngen autotest run reproduce.json machine-ali.json > run.result 2>&1 &
```

the run process of reproduce is similar to before.

7.6.4 Reproduce post

```
dpngen autotest post reproduce.json
```

the output will be:

result.out:

```

/root/auto_test_example/deepmd/confs/std-fcc/interstitial_reprod
Reproduce: Initial_path Init_E(eV/atom)  Reprod_E(eV/atom)  Difference(eV/atom)
.../vasp/confs/std-fcc/interstitial_00/task.000000 -3.020 -3.240 -0.220
.../vasp/confs/std-fcc/interstitial_00/task.000000 -3.539 -3.541 -0.002
.../vasp/confs/std-fcc/interstitial_00/task.000000 -3.582 -3.582 -0.001
.../vasp/confs/std-fcc/interstitial_00/task.000000 -3.582 -3.581 0.001
.../vasp/confs/std-fcc/interstitial_00/task.000000 -3.594 -3.593 0.001
.../vasp/confs/std-fcc/interstitial_00/task.000000 -3.594 -3.594 0.001
.../vasp/confs/std-fcc/interstitial_00/task.000000 -3.598 -3.597 0.001
.../vasp/confs/std-fcc/interstitial_00/task.000000 -3.600 -3.600 0.001
.../vasp/confs/std-fcc/interstitial_00/task.000000 -3.600 -3.600 0.001
.../vasp/confs/std-fcc/interstitial_00/task.000000 -3.601 -3.600 0.001
.../vasp/confs/std-fcc/interstitial_00/task.000000 -3.602 -3.601 0.001
.../vasp/confs/std-fcc/interstitial_00/task.000000 -3.603 -3.602 0.001
.../vasp/confs/std-fcc/interstitial_00/task.000000 -3.603 -3.602 0.001
.../vasp/confs/std-fcc/interstitial_00/task.000000 -3.603 -3.602 0.001

```

(continues on next page)

(continued from previous page)

```

.../vasp/confs/std-fcc/interstitial_00/task.000000 -3.603 -3.602 0.001
.../vasp/confs/std-fcc/interstitial_00/task.000000 -3.603 -3.602 0.001
.../vasp/confs/std-fcc/interstitial_00/task.000000 -3.603 -3.602 0.001
.../vasp/confs/std-fcc/interstitial_00/task.000000 -3.603 -3.602 0.001
.../vasp/confs/std-fcc/interstitial_00/task.000001 -3.345 -3.372 -0.027
.../vasp/confs/std-fcc/interstitial_00/task.000001 -3.546 -3.556 -0.009
.../vasp/confs/std-fcc/interstitial_00/task.000001 -3.587 -3.593 -0.007
.../vasp/confs/std-fcc/interstitial_00/task.000001 -3.593 -3.599 -0.006
.../vasp/confs/std-fcc/interstitial_00/task.000001 -3.600 -3.606 -0.006
.../vasp/confs/std-fcc/interstitial_00/task.000001 -3.600 -3.606 -0.006
.../vasp/confs/std-fcc/interstitial_00/task.000001 -3.624 -3.631 -0.006
.../vasp/confs/std-fcc/interstitial_00/task.000001 -3.634 -3.640 -0.007
.../vasp/confs/std-fcc/interstitial_00/task.000001 -3.637 -3.644 -0.007
.../vasp/confs/std-fcc/interstitial_00/task.000001 -3.637 -3.644 -0.007
.../vasp/confs/std-fcc/interstitial_00/task.000001 -3.638 -3.645 -0.007
.../vasp/confs/std-fcc/interstitial_00/task.000001 -3.638 -3.645 -0.007
.../vasp/confs/std-fcc/interstitial_00/task.000001 -3.639 -3.646 -0.007
.../vasp/confs/std-fcc/interstitial_00/task.000001 -3.639 -3.646 -0.007
.../vasp/confs/std-fcc/interstitial_00/task.000001 -3.639 -3.646 -0.007
.../vasp/confs/std-fcc/interstitial_00/task.000001 -3.639 -3.646 -0.007
.../vasp/confs/std-fcc/interstitial_00/task.000001 -3.639 -3.646 -0.007
.../vasp/confs/std-fcc/interstitial_00/task.000001 -3.639 -3.646 -0.007
.../vasp/confs/std-fcc/interstitial_00/task.000001 -3.639 -3.646 -0.007
.../vasp/confs/std-fcc/interstitial_00/task.000001 -3.639 -3.646 -0.007
.../vasp/confs/std-fcc/interstitial_00/task.000001 -3.639 -3.646 -0.007
.../vasp/confs/std-fcc/interstitial_00/task.000001 -3.639 -3.646 -0.007

```

the comparison of the initial and reproduced results as well as the absolute path of the initial data is recorded.

result.json:

```

{
  "/root/auto_test_example/vasp/confs/std-fcc/interstitial_00/task.000000": {
    "nframes": 18,
    "error": 0.0009738182472213228
  },
  "/root/auto_test_example/vasp/confs/std-fcc/interstitial_00/task.000001": {
    "nframes": 21,
    "error": 0.0006417039154057605
  }
}

```

the error analysis corresponding to the initial data is recorded and the error of the first frame is disregarded when all the frames are considered in reproduce.

USER GUIDE

This part aims to show you how to get the community's help. Some frequently asked questions are listed in troubleshooting, and the explanation of errors that often occur is listed in common errors. If other unexpected problems occur, you're welcome to contact us for help.

8.1 Discussions

Welcome everyone to participate in the discussion about DP-GEN in the [discussion](#) module. You can ask for help, share an idea or anything to discuss here. Note: before you raise a question, please check TUTORIAL/FAQs and search history discussions to find solutions.

8.2 Issue

If you want to make a bug report or a request for new features, you can make an issue in the issue module.

Here are the types you can choose. A proper type can help developer figure out what you need. Also, you can assign yourself to solve the issue. Your contribution is welcome!

Note: before you raise a question, please check TUTORIAL/FAQs and search history issues to find solutions.

8.3 Tutorials

Tutorials can be found [here](#).

8.4 Example for parameters

If you have no idea how to prepare a PARAM for your task, you can find examples of PARAM for different tasks in [examples](#).

For example, if you want to set specific template for LAMMPS, you can find an example [here](#)

If you want to learn more about Machine parameters, please check [docs for dpdispatcher](#)

8.5 Pull requests - How to contribute

8.6 Troubleshooting

1. The most common problem is whether two settings correspond with each other, including:
 - The order of elements in `type_map` and `mass_map` and **`fp_pp_files`**.
 - Size of `init_data_sys` and `init_batch_size`.
 - Size of `sys_configs` and `sys_batch_size`.
 - Size of `sel_a` and actual types of atoms in your system.
 - Index of `sys_configs` and `sys_idx`.
2. Please verify the directories of `sys_configs`. If there isn't any POSCAR for `01.model_devi` in one iteration, it may happen that you write the false path of `sys_configs`.
3. Correct format of JSON file.
4. The frames of one system should be larger than `batch_size` and `numb_test` in `default_training_param`. It happens that one iteration adds only a few structures and causes error in next iteration's training. In this condition, you may let `fp_task_min` be larger than `numb_test`.

8.7 Common Errors

(Errors are sorted alphabetically)

8.7.1 `dargs.dargs.ArgumentTypeError: [at root location] key xxx gets wrong value type, requires but gets`

Please check your parameters with [DPGEN's Document](#). Maybe you have superfluous parentheses in your parameter file.

8.7.2 Dargs: `xxx` is not allowed in strict mode.

Strict format check has been applied since version 0.10.7. To avoid misleading users, some older-version keys that are already ignored or absorbed into default settings are not allowed to be present. And the expected structure of the dictionary in the `param.json` also differs from those before version 0.10.7. This error will occur when format check finds older-fashion keys in the json file. Please try deleting or annotating these keys, or correspondingly modulate the json file. Example files in the newest format could be found in [examples](#).

8.7.3 FileNotFoundError: [Errno 2] No such file or directory: '.../01.model_devi/graph.xxx.pb'

If you find this error occurs, please check your initial data. Your model will not be generated if the initial data is incorrect.

8.7.4 json.decoder.JSONDecodeError

Your .json file is incorrect. It may be a mistake in syntax or a missing comma.

8.7.5 RuntimeError: job:xxxxxxx failed 3 times

```
RuntimeError: job:xxxxxxx failed 3 times

.....

RuntimeError: Meet errors will handle unexpected submission state.
Debug information: remote_root==xxxxxxx
Debug information: submission_hash==xxxxxxx
Please check the dirs and scripts in remote_root. The job information mentioned above,
↪may help.
```

If a user finds an error like this, he or she is advised to check the files on the remote server. It shows that your job has failed 3 times, but has not shown the reason.

To find the reason, you can check the log on the remote root. For example, you can check train.log, which is generated by DeepMD-kit. It can tell you more details. If it doesn't help, you can manually run the .sub script, whose path is shown in Debug information: remote_root==xxxxxxx

Some common reasons are as follows:

1. Two or more jobs are submitted manually or automatically at the same time, and their hash value collide. This bug will be fixed in dpdispatcher.
2. You may have something wrong in your input files, which causes the process to fail.

8.7.6 RuntimeError: find too many unsuccessfully terminated jobs.

The ratio of failed jobs is larger than ratio_failure. You can set a high value for ratio_failure or check if there is something wrong with your input files.

8.7.7 ValueError: Cannot load file containing picked data when allow_picked=False

Please ensure that you write the correct path of the dataset with no excess files.

CONTRIBUTING GUIDE

9.1 Contributing Guide

The way to make contributions is through making pull requests(PR for short). After your PR is merged, the changes you make can be applied by other users.

Firstly, fork in DP-GEN repository. Then you can clone the repository, build a new branch, make changes and then make a pull request.

9.1.1 How to contribute to DP-GEN

Welcome to the repository of [DP-GEN](#)

DP-GEN adopts the same convention as other software in DeepModeling Community.

You can first refer to DeePMD-kit's [Contributing guide](#) and [Developer guide](#).

You can also read relative chapters on [Github Docs](#).

If you have no idea how to fix your problem or where to find the relative source code, please check [Code Structure](#) of the DP-GEN repository on this website.

Use command line

You can use git with the command line, or open the repository on Github Desktop. Here is a video as a demo of making changes to DP-GEN and publishing it with command line.

If you have never used Github before, remember to generate your ssh key and configure the public key in Github Settings. If you can't configure your username and password, please use token. The explanation from Github see [Github Blog: token authentication requirements for git operations](#). A discussion on [StackOverflow](#) can solve this problem.

Use Github Desktop

Also, you can use Github Desktop to make PR. The following shows the steps to clone the repository and add your doc to tutorials. If it is your first time using Github, Open with Github Desktop is recommended. Github Desktop is a software, which can make your operations on branches visually.

After you clone it to your PC, you can open it with Github Desktop.

Firstly, create your new branch based on devel branch.

Secondly, add your doc to the certain directory in your local repository, and add its name into index.

Here is an [example](#). Remember to add the filename of your doc into index!

Thirdly, select the changes that you want to push, and commit to it. Press “Publish branch” to push your origin repository to the remote branch.

Finally, you can check it on github and make a pull request. Press “Compare & pull request” to make a PR.

(Note: please commit pr to the devel branch)

9.1.2 How to contribute to DP-GEN tutorials and documents

Welcome to [the documents of DP-GEN](#)

- If you want to add the documentation of a toy model, simply put your file in the directory doc/toymodels/ and push;
- If you want to add a new directory for a new category of instructions, make a new directory and add it in doc/index.rst.

Also welcome to [Tutorials repository](#) You can find the structure of tutorials and preparations before writing a document in [Writing Tips](#).

The latest page of DP-GEN Docs

Examples of contributions

- [Example 1](#)
- [Example 2](#) (a simple one for beginner)

1. Push your doc

2. Add the directory in index.rst

3. Build and check it

As mentioned in “How to build the website to check if the modification works”.

4. Make pull request to dpngen

9.1.3 Find how a parameter is used in the code

It is strongly recommended that you use the `find in files` function of Visual Studio software, `Search` function of Visual Studio Code, or similar functions of other software. Type in the name of the parameter you are looking for, and you will see where it is read in and used in the procedure. Of course, you can also search for the relevant code according to the above guide.

9.1.4 Want to modify a function?

If you have special requirements, you can make personalized modifications in the code corresponding to the function. If you think your modification can benefit the public, and it does not conflict with the current DP-GEN function; or if you fix a bug, please make a pull request to contribute the optimization to the DP-GEN repository.

9.1.5 DP-GEN dependencies

`dpdispatcher` and `dpdata` are dependencies of DP-GEN. `dpdispatcher` is related to task submission, monitoring and recovery, and `dpdata` is related to data processing. If you encounter an error and want to find the reason, please judge whether the problem comes from DP-GEN, `dpdispatcher` or `dpdata` according to the last line of `Traceback`.

9.1.6 About the update of the parameter file

You may have noticed that there are `arginfo.py` files in many folders. This is a file used to generate parameter documentation. If you add or modify a parameter in DP-GEN and intend to export it to the main repository, please sync your changes in `arginfo`.

9.1.7 Tips

1. Please try to submit a PR after finishing all the changes
2. Please briefly describe what you do with `git commit -m "<conclude-the-change-you-make>"`! “No description provided.” will make the maintainer feel confused.
3. It is not recommended to make changes directly in the `devel` branch. It is recommended to pull a branch from `devel`: `git checkout -b <new-branch-name>`
4. When switching branches, remember to check if you want to bring the changes to the next branch!
5. Please fix the errors reported by the unit test. You can firstly test on your local machine before pushing commits. Hint: The way to test the code is to go from the main directory to the tests directory, and use the command `python3 -m unittest`. You can watch the demo video for review. Sometimes you may fail unit tests due to your local circumstance. You can check whether the error reported is related to the part you modified to eliminate this problem. After submitting, as long as there is a green check mark after the PR title on the webpage, it means that the test has been passed.
6. Pay attention to whether there are comments under your PR. If there is a change request, you need to check and modify the code. If there are conflicts, you need to solve them manually.

After successfully making a PR, developers will check it and give comments. It will be merged after everything done. Then CONGRATULATIONS! You become a first-time contributor to DP-GEN!

How to get help from the community

DP-GEN API

10.1 dpgen package

`dpgen.info()`

10.1.1 Subpackages

`dpgen.auto_test` package

Subpackages

`dpgen.auto_test.lib` package

Submodules

`dpgen.auto_test.lib.BatchJob` module

```
class dpgen.auto_test.lib.BatchJob.BatchJob(job_dir="",job_script="",job_finish_tag='tag_finished',  
                                           job_id_file='tag_jobid')
```

Bases: object

Abstract class of a batch job It submit a job (leave the id in file tag_jobid) It check the status of the job (return JobStatus) NOTICE: I assume that when a job finishes, a tag file named tag_finished should be touched by the user. TYPICAL USAGE: job = DERIVED_BatchJob (dir, script) job.submit () stat = job.check_status ()

Methods

<code>submit_command()</code>	submission is \$ [command] [script]
-------------------------------	-------------------------------------

check_status	
get_job_id	
submit	

check_status()

```
get_job_id()
```

```
submit()
```

```
submit_command()
```

```
    submission is $ [command] [script]
```

```
class dpgen.auto_test.lib.BatchJob.JobStatus(value)
```

```
    Bases: Enum
```

```
    An enumeration.
```

```
    finished = 5
```

```
    running = 3
```

```
    terminated = 4
```

```
    unknow = 100
```

```
    unsubmitted = 1
```

```
    waiting = 2
```

dpgen.auto_test.lib.RemoteJob module

```
class dpgen.auto_test.lib.RemoteJob.CloudMachineJob(ssh_session, local_root)
```

```
    Bases: RemoteJob
```

Methods

block_call	
block_checkcall	
check_status	
clean	
download	
get_job_root	
submit	
upload	

```
check_status()
```

```
submit(job_dirs, cmd, args=None, resources=None)
```

```
class dpgen.auto_test.lib.RemoteJob.JobStatus(value)
```

```
    Bases: Enum
```

```
    An enumeration.
```

```
    finished = 5
```

```
    running = 3
```

```
    terminated = 4
```



```
unknown = 100
```

```
unsubmitted = 1
```

```
waiting = 2
```

```
class dpngen.auto_test.lib.RemoteJob.PBSJob(ssh_session, local_root)
```

```
Bases: RemoteJob
```

Methods

block_call	
block_checkcall	
check_status	
clean	
download	
get_job_root	
submit	
upload	

```
check_status()
```

```
submit(job_dirs, cmd, args=None, resources=None)
```

```
class dpngen.auto_test.lib.RemoteJob.RemoteJob(ssh_session, local_root)
```

```
Bases: object
```

Methods

block_call	
block_checkcall	
clean	
download	
get_job_root	
upload	

```
block_call(cmd)
```

```
block_checkcall(cmd)
```

```
clean()
```

```
download(job_dirs, remote_down_files)
```

```
get_job_root()
```

```
upload(job_dirs, local_up_files, dereference=True)
```

```
class dpngen.auto_test.lib.RemoteJob.SSHSession(jdata)
```

```
Bases: object
```

Methods

close	
get_session_root	
get_ssh_client	

close()

get_session_root()

get_ssh_client()

class dpgen.auto_test.lib.RemoteJob.**SlurmJob**(*ssh_session, local_root*)

Bases: *RemoteJob*

Methods

block_call	
block_checkcall	
check_status	
clean	
download	
get_job_root	
submit	
upload	

check_status()

submit(*job_dirs, cmd, args=None, resources=None*)

dpgen.auto_test.lib.SlurmJob module

class dpgen.auto_test.lib.SlurmJob.**SlurmJob**(*job_dir="", job_script="", job_finish_tag='tag_finished',
job_id_file='tag_jobid'*)

Bases: *BatchJob*

Methods

<i>submit_command()</i>	submission is \$ [command] [script]
-------------------------	-------------------------------------

check_status	
get_job_id	
submit	

check_status()

submit_command()

submission is \$ [command] [script]

dpgen.auto_test.lib.abacus module

```

dpngen.auto_test.lib.abacus.check_finished(fname)
dpngen.auto_test.lib.abacus.check_stru_fixed(struf, fixed)
dpngen.auto_test.lib.abacus.final_stru(abacus_path)
dpngen.auto_test.lib.abacus.make_kspacing_kpt(struf, kspacing)
dpngen.auto_test.lib.abacus.modify_stru_path(struf, tpath)
dpngen.auto_test.lib.abacus.poscar2stru(poscar, inter_param, stru)
    • poscar: POSCAR for input
    • inter_param: dictionary of 'interaction' from param.json
      some key words for ABACUS are:
        - atom_masses: a dictionary of atoms' masses
        - orb_files: a dictionary of orbital files
        - deepks_desc: a string of deepks descriptor file
    • stru: output filename, usually is 'STRU'
dpngen.auto_test.lib.abacus.stru2Structure(struf)
dpngen.auto_test.lib.abacus.stru_fix_atom(struf, fix_atom=[True, True, True])
    ... ATOMIC_POSITIONS Cartesian #Cartesian(Unit is LATTICE_CONSTANT) Si #Name of element 0.0
    #Magnetic for this element. 2 #Number of atoms 0.00 0.00 0.00 0 0 0 #x,y,z, move_x, move_y, move_z 0.25
    0.25 0.25 0 0 0
dpngen.auto_test.lib.abacus.stru_scale(stru_in, stru_out, scale)
dpngen.auto_test.lib.abacus.write_input(inputf, inputdict)
dpngen.auto_test.lib.abacus.write_kpt(kptf, kptlist)

```

dpngen.auto_test.lib.crys module

```

dpngen.auto_test.lib.crys.bcc(ele_name='ele', a=3.2144871302356037)
dpngen.auto_test.lib.crys.dhcp(ele_name='ele', a=2.863782463805517, c=9.353074360871936)
dpngen.auto_test.lib.crys.diamond(ele_name='ele', a=2.551340126037118)
dpngen.auto_test.lib.crys.fcc(ele_name='ele', a=4.05)
dpngen.auto_test.lib.crys.fcc1(ele_name='ele', a=4.05)
dpngen.auto_test.lib.crys.hcp(ele_name='ele', a=2.863782463805517, c=4.676537180435968)
dpngen.auto_test.lib.crys.sc(ele_name='ele', a=2.551340126037118)

```

dpngen.auto_test.lib.lammps module

dpngen.auto_test.lib.lammps.apply_type_map(*conf_file*, *deepmd_type_map*, *ptypes*)
apply type map. *conf_file*: conf file converted from POSCAR *deepmd_type_map*: deepmd atom type map
ptypes: atom types defined in POSCAR

dpngen.auto_test.lib.lammps.check_finished(*fname*)

dpngen.auto_test.lib.lammps.check_finished_new(*fname*, *keyword*)

dpngen.auto_test.lib.lammps.cvt_lammps_conf(*fin*, *fout*, *type_map*, *ofmt*='lammps/data')
Format convert from *fin* to *fout*, specify the output format by *ofmt* Incomplete situation

dpngen.auto_test.lib.lammps.element_list(*type_map*)

dpngen.auto_test.lib.lammps.get_base_area(*log*)
get base area

dpngen.auto_test.lib.lammps.get_nev(*log*)
get natoms, energy_per_atom and volume_per_atom from lammps log

dpngen.auto_test.lib.lammps.get_stress(*log*)
get stress from lammps log

dpngen.auto_test.lib.lammps.inter_deepmd(*param*)

dpngen.auto_test.lib.lammps.inter_eam_alloy(*param*)

dpngen.auto_test.lib.lammps.inter_eam_fs(*param*)

dpngen.auto_test.lib.lammps.inter_meam(*param*)

dpngen.auto_test.lib.lammps.make_lammps_elastic(*conf*, *type_map*, *interaction*, *param*, *etol*=0, *ftol*=1e-10,
maxiter=5000, *maxeval*=500000)

dpngen.auto_test.lib.lammps.make_lammps_equi(*conf*, *type_map*, *interaction*, *param*, *etol*=0, *ftol*=1e-10,
maxiter=5000, *maxeval*=500000, *change_box*=True)

dpngen.auto_test.lib.lammps.make_lammps_eval(*conf*, *type_map*, *interaction*, *param*)

dpngen.auto_test.lib.lammps.make_lammps_phonon(*conf*, *masses*, *interaction*, *param*, *etol*=0, *ftol*=1e-10,
maxiter=5000, *maxeval*=500000)
make lammps input for elastic calculation

dpngen.auto_test.lib.lammps.make_lammps_press_relax(*conf*, *type_map*, *scale2equi*, *interaction*, *param*,
B0=70, *bp*=0, *etol*=0, *ftol*=1e-10, *maxiter*=5000,
maxeval=500000)

dpngen.auto_test.lib.lammps.poscar_from_last_dump(*dump*, *poscar_out*, *deepmd_type_map*)
get poscar from the last frame of a lammps MD traj (dump format)

dpgen.auto_test.lib.lmp module

```

dpngen.auto_test.lib.lmp.box2lmpbox(orig, box)
dpngen.auto_test.lib.lmp.from_system_data(system)
dpngen.auto_test.lib.lmp.get_atoms(lines)
dpngen.auto_test.lib.lmp.get_atype(lines)
dpngen.auto_test.lib.lmp.get_lmpbox(lines)
dpngen.auto_test.lib.lmp.get_natoms(lines)
dpngen.auto_test.lib.lmp.get_natoms_vec(lines)
dpngen.auto_test.lib.lmp.get_natomtypes(lines)
dpngen.auto_test.lib.lmp.get_posi(lines)
dpngen.auto_test.lib.lmp.lmpbox2box(lohi, tilt)
dpngen.auto_test.lib.lmp.system_data(lines)
dpngen.auto_test.lib.lmp.to_system_data(lines)

```

dpngen.auto_test.lib.mfp_eosfit module

```

dpngen.auto_test.lib.mfp_eosfit.BM4(vol, pars)
    Birch-Murnaghan 4 pars equation from PRB 70, 224107, 3-order
dpngen.auto_test.lib.mfp_eosfit.BM5(vol, pars)
    Birch-Murnaghan 5 pars equation from PRB 70, 224107, 4-Order
dpngen.auto_test.lib.mfp_eosfit.LOG4(vol, pars)
    Natrual strain (Poirier-Tarantola)EOS with 4 paramters Seems only work in near-equilibrium range.
dpngen.auto_test.lib.mfp_eosfit.LOG5(vol, parameters)
    Natrual strain (Poirier-Tarantola)EOS with 5 paramters
dpngen.auto_test.lib.mfp_eosfit.Li4p(V, parameters)
    Li JH, APL, 87, 194111 (2005)
dpngen.auto_test.lib.mfp_eosfit.SJX_5p(vol, par)
    SJX_5p's five parameters EOS, Physica B: Condens Mater, 2011, 406: 1276-1282
dpngen.auto_test.lib.mfp_eosfit.SJX_v2(vol, par)
    Sun Jiuxun, et al. J phys Chem Solids, 2005, 66: 773-782. They said it is satisfied for the limiting condition at high pressure.
dpngen.auto_test.lib.mfp_eosfit.TEOS(v, par)
    Holland, et al, Journal of Metamorphic Geology, 2011, 29(3): 333-383 Modified Tait equation of Huang & Chow
dpngen.auto_test.lib.mfp_eosfit.birch(v, parameters)
    From Intermetallic compounds: Principles and Practice, Vol. I: Princples Chapter 9 pages 195-210 by M. Mehl.
    B. Klein, D. Papaconstantopoulos paper downloaded from Web
    case where n=0

```

```
dpngen.auto_test.lib.mfp_eosfit.calc_props_BM4(pars)
dpngen.auto_test.lib.mfp_eosfit.calc_props_LOG4(pars)
dpngen.auto_test.lib.mfp_eosfit.calc_props_SJX_5p(par)
dpngen.auto_test.lib.mfp_eosfit.calc_props_mBM4(pars)
dpngen.auto_test.lib.mfp_eosfit.calc_props_mBM4poly(pars)
dpngen.auto_test.lib.mfp_eosfit.calc_props_mBM5poly(pars)
dpngen.auto_test.lib.mfp_eosfit.calc_props_morse(pars)
dpngen.auto_test.lib.mfp_eosfit.calc_props_morse_6p(par)
dpngen.auto_test.lib.mfp_eosfit.calc_props_vinet(pars)
dpngen.auto_test.lib.mfp_eosfit.calc_v0_mBM4poly(x, pars)
dpngen.auto_test.lib.mfp_eosfit.calc_v0_mBM5poly(x, pars)
dpngen.auto_test.lib.mfp_eosfit.ext_splint(xp, yp, order=3, method='unispl')
dpngen.auto_test.lib.mfp_eosfit.ext_vec(func, fin, p0, fs, fe, vols=None, vole=None, ndata=101, refit=0,
                                         show_fig=False)
    extrapolate the data points for E-V based on the fitted parameters in small or very large volume range.
dpngen.auto_test.lib.mfp_eosfit.ext_velp(fin, fstart, fend, vols, vole, ndata, order=3, method='unispl',
                                         fout='ext_velop.dat', show_fig=False)
    extrapolate the lattice parameters based on input data
dpngen.auto_test.lib.mfp_eosfit.get_eos_list()
dpngen.auto_test.lib.mfp_eosfit.get_eos_list_3p()
dpngen.auto_test.lib.mfp_eosfit.get_eos_list_4p()
dpngen.auto_test.lib.mfp_eosfit.get_eos_list_5p()
dpngen.auto_test.lib.mfp_eosfit.get_eos_list_6p()
dpngen.auto_test.lib.mfp_eosfit.init_guess(fin)
dpngen.auto_test.lib.mfp_eosfit.lsqfit_eos(func, fin, par, fstart, fend, show_fig=False, fout='EoSfit.out',
                                           refit=-1)

dpngen.auto_test.lib.mfp_eosfit.mBM4(vol, pars)
    Birch-Murnaghan 4 pars equation from PRB 70, 224107, 3-order BM
dpngen.auto_test.lib.mfp_eosfit.mBM4poly(vol, parameters)
    modified BM5 EOS, Shang SL comput mater sci, 2010: 1040-1048, original expressions.
dpngen.auto_test.lib.mfp_eosfit.mBM5(vol, pars)
    modified BM5 EOS, Shang SL comput mater sci, 2010: 1040-1048
dpngen.auto_test.lib.mfp_eosfit.mBM5poly(vol, pars)
    modified BM5 EOS, Shang SL comput mater sci, 2010: 1040-1048, original expressions.
```

`dpgen.auto_test.lib.mfp_eosfit.mie(v, p)`
 Mie model for song's FVT

`dpgen.auto_test.lib.mfp_eosfit.mie_simple(v, p)`
 Mie_simple model for song's FVT

`dpgen.auto_test.lib.mfp_eosfit.morse(v, pars)`
 Reproduce from ShunliShang's matlab script.

`dpgen.auto_test.lib.mfp_eosfit.morse_3p(volume, p)`
 morse_AB EOS formula from Song's FVT sources $A = 0.5 * B$

`dpgen.auto_test.lib.mfp_eosfit.morse_6p(vol, par)`
 Generalized Morse EOS proposed by Qin, see: Qin et al. Phys Rev B, 2008, 78, 214108. Qin et al. Phys Rev B, 2008, 77, 220103(R).

`dpgen.auto_test.lib.mfp_eosfit.morse_AB(volume, p)`
 morse_AB EOS formula from Song's FVT sources

`dpgen.auto_test.lib.mfp_eosfit.murnaghan(vol, pars)`
 Four-parameters murnaghan EOS. From PRB 28,5480 (1983)

`dpgen.auto_test.lib.mfp_eosfit.parse_argument()`

`dpgen.auto_test.lib.mfp_eosfit.rBM4(vol, pars)`
 Implementions as Alberto Otero-de-la-Roza, i.e. rBM4 is used here Comput Physics Comm, 2011, 182: 1708-1720

`dpgen.auto_test.lib.mfp_eosfit.rBM4_pv(vol, pars)`
 Implementions as Alberto Otero-de-la-Roza, i.e. rBM4 is used here Comput Physics Comm, 2011, 182: 1708-1720 Fit for V-P relations

`dpgen.auto_test.lib.mfp_eosfit.rBM5(vol, pars)`
 Implementions as Alberto Otero-de-la-Roza, i.e. rBM5 is used here Comput Physics Comm, 2011, 182: 1708-1720

`dpgen.auto_test.lib.mfp_eosfit.rBM5_pv(vol, pars)`
 Implementions as Alberto Otero-de-la-Roza, i.e. rBM5 is used here Comput Physics Comm, 2011, 182: 1708-1720 Fit for V-P relations

`dpgen.auto_test.lib.mfp_eosfit.rPT4(vol, pars)`
 Natrual strain EOS with 4 paramters Seems only work in near-equilibrium range. Implementions as Alberto Otero-de-la-Roza, i.e. rPT4 is used here Comput Physics Comm, 2011, 182: 1708-1720, in their article, labeled as PT3 (3-order), however, we mention it as rPT4 for 4-parameters EOS.

`dpgen.auto_test.lib.mfp_eosfit.rPT4_pv(vol, pars)`
 Natrual strain (Poirier-Tarantola)EOS with 4 paramters Seems only work in near-equilibrium range. Implementions as Alberto Otero-de-la-Roza, i.e. rPT4 is used here Comput Physics Comm, 2011, 182: 1708-1720, in their article, labeled as PT3 (3-order), however, we mention it as rPT4 for 4-parameters EOS.

`dpgen.auto_test.lib.mfp_eosfit.rPT5(vol, pars)`
 Natrual strain EOS with 4 paramters Seems only work in near-equilibrium range. Implementions as Alberto Otero-de-la-Roza, i.e. rPT5 is used here Comput Physics Comm, 2011, 182: 1708-1720, in their article, labeled as PT3 (3-order), however, we mention it as rPT5 for 4-parameters EOS.

`dpngen.auto_test.lib.mfp_eosfit.rPT5_pv(vol, pars)`

Natrual strain (Poirier-Tarantola)EOS with 5 paramters Implementions as Alberto Otero-de-la-Roza, i.e. rPT5 is used here Comput Physics Comm, 2011, 182: 1708-1720, in their article, labeled as PT3 (3-order), however, we mention it as rPT5 for 4-parameters EOS.

`dpngen.auto_test.lib.mfp_eosfit.read_ve(fin)`

`dpngen.auto_test.lib.mfp_eosfit.read_velp(fin, fstart, fend)`

`dpngen.auto_test.lib.mfp_eosfit.read_vlp(fin, fstart, fend)`

`dpngen.auto_test.lib.mfp_eosfit.repro_ve(func, vol_i, p)`

`dpngen.auto_test.lib.mfp_eosfit.repro_vp(func, vol_i, pars)`

`dpngen.auto_test.lib.mfp_eosfit.res_BM4(pars, y, x)`

`dpngen.auto_test.lib.mfp_eosfit.res_BM5(pars, y, x)`

`dpngen.auto_test.lib.mfp_eosfit.res_LOG4(pars, y, x)`

`dpngen.auto_test.lib.mfp_eosfit.res_LOG5(pars, y, x)`

`dpngen.auto_test.lib.mfp_eosfit.res_Li4p(p, y, x)`

`dpngen.auto_test.lib.mfp_eosfit.res_SJX_5p(p, e, v)`

`dpngen.auto_test.lib.mfp_eosfit.res_SJX_v2(p, e, v)`

`dpngen.auto_test.lib.mfp_eosfit.res_TEOS(p, e, v)`

`dpngen.auto_test.lib.mfp_eosfit.res_birch(pars, y, x)`

`dpngen.auto_test.lib.mfp_eosfit.res_mBM4(pars, y, x)`

`dpngen.auto_test.lib.mfp_eosfit.res_mBM4poly(pars, y, x)`

`dpngen.auto_test.lib.mfp_eosfit.res_mBM5(pars, y, x)`

`dpngen.auto_test.lib.mfp_eosfit.res_mBM5poly(pars, y, x)`

`dpngen.auto_test.lib.mfp_eosfit.res_mie(p, e, v)`

`dpngen.auto_test.lib.mfp_eosfit.res_mie_simple(p, e, v)`

`dpngen.auto_test.lib.mfp_eosfit.res_morse(p, en, volume)`

`dpngen.auto_test.lib.mfp_eosfit.res_morse_3p(p, en, volume)`

`dpngen.auto_test.lib.mfp_eosfit.res_morse_6p(p, en, volume)`

`dpngen.auto_test.lib.mfp_eosfit.res_morse_AB(p, en, volume)`

`dpngen.auto_test.lib.mfp_eosfit.res_murnaghan(pars, y, x)`

`dpngen.auto_test.lib.mfp_eosfit.res_rBM4(pars, y, x)`

`dpngen.auto_test.lib.mfp_eosfit.res_rBM4_pv(par, y, x)`

`dpngen.auto_test.lib.mfp_eosfit.res_rBM5(pars, y, x)`


```

dpgen.auto_test.lib.mfp_eosfit.res_rBM5_pv(par, y, x)
dpgen.auto_test.lib.mfp_eosfit.res_rPT4(pars, y, x)
dpgen.auto_test.lib.mfp_eosfit.res_rPT4_pv(pars, y, x)
dpgen.auto_test.lib.mfp_eosfit.res_rPT5(pars, y, x)
dpgen.auto_test.lib.mfp_eosfit.res_rPT5_pv(pars, y, x)
dpgen.auto_test.lib.mfp_eosfit.res_universal(pars, y, x)
dpgen.auto_test.lib.mfp_eosfit.res_vinet(pars, y, x)
dpgen.auto_test.lib.mfp_eosfit.res_vinet_pv(par, y, x)
dpgen.auto_test.lib.mfp_eosfit.universal(vol, parameters)
    Universal equation of state(Vinet P et al., J. Phys.: Condens. Matter 1, p1941 (1989))
dpgen.auto_test.lib.mfp_eosfit.vinet(vol, pars)
    Vinet equation from PRB 70, 224107 Following, Shang Shunli et al., comput mater sci, 2010: 1040-1048,
    original expressions.
dpgen.auto_test.lib.mfp_eosfit.vinet_pv(vol, pars)

```

dpgen.auto_test.lib.pwscf module

```
dpgen.auto_test.lib.pwscf.make_pwscf_input(sys_data, fp_pp_files, fp_params)
```

dpgen.auto_test.lib.siesta module

```
dpgen.auto_test.lib.siesta.make_siesta_input(sys_data, fp_pp_files, fp_params)
```

dpgen.auto_test.lib.util module

```

dpgen.auto_test.lib.util.collect_task(all_task, task_type)
dpgen.auto_test.lib.util.get_machine_info(mdata, task_type)
dpgen.auto_test.lib.util.insert_data(task, task_type, username, file_name)
dpgen.auto_test.lib.util.make_work_path(jdata, task, reprod_opt, static, user)
dpgen.auto_test.lib.util.voigt_to_stress(inpt)

```

dpngen.auto_test.lib.utils module

```
dpngen.auto_test.lib.utils.cmd_append_log(cmd, log_file)
dpngen.auto_test.lib.utils.copy_file_list(file_list, from_path, to_path)
dpngen.auto_test.lib.utils.create_path(path)
dpngen.auto_test.lib.utils.log_iter(task, ii, jj)
dpngen.auto_test.lib.utils.log_task(message)
dpngen.auto_test.lib.utils.make_iter_name(iter_index)
dpngen.auto_test.lib.utils.record_iter(record, confs, ii, jj)
dpngen.auto_test.lib.utils.repeat_to_length(string_to_expand, length)
dpngen.auto_test.lib.utils.replace(file_name, pattern, subst)
```

dpngen.auto_test.lib.vasp module

```
exception dpngen.auto_test.lib.vasp.OutcarItemError
```

```
    Bases: Exception
```

```
dpngen.auto_test.lib.vasp.check_finished(fname)
dpngen.auto_test.lib.vasp.get_boxes(fname)
dpngen.auto_test.lib.vasp.get_energies(fname)
dpngen.auto_test.lib.vasp.get_nev(fname)
dpngen.auto_test.lib.vasp.get_poscar_natoms(fname)
dpngen.auto_test.lib.vasp.get_poscar_types(fname)
dpngen.auto_test.lib.vasp.get_stress(fname)
dpngen.auto_test.lib.vasp.make_kspacing_kpoints(poscar, kspacing, kgamma)
dpngen.auto_test.lib.vasp.make_vasp_kpoints(kpoints, kgamma=False)
dpngen.auto_test.lib.vasp.make_vasp_kpoints_from_incar(work_dir, jdata)
dpngen.auto_test.lib.vasp.make_vasp_phonon_incar(ecut, ediff, npar, kpar, kspacing=0.5, kgamma=True,
                                                ismear=1, sigma=0.2)
dpngen.auto_test.lib.vasp.make_vasp_relax_incar(ecut, ediff, relax_ion, relax_shape, relax_volume, npar,
                                                kpar, kspacing=0.5, kgamma=True, ismear=1,
                                                sigma=0.22)
dpngen.auto_test.lib.vasp.make_vasp_static_incar(ecut, ediff, npar, kpar, kspacing=0.5, kgamma=True,
                                                ismear=1, sigma=0.2)
dpngen.auto_test.lib.vasp.perturb_xz(poscar_in, poscar_out, pert=0.01)
```

```

dpgen.auto_test.lib.vasp.poscar_natoms(poscar_in)
dpgen.auto_test.lib.vasp.poscar_scale(poscar_in, poscar_out, scale)
dpgen.auto_test.lib.vasp.poscar_vol(poscar_in)
dpgen.auto_test.lib.vasp.reciprocal_box(box)
dpgen.auto_test.lib.vasp.regulate_poscar(poscar_in, poscar_out)
dpgen.auto_test.lib.vasp.sort_poscar(poscar_in, poscar_out, new_names)

```

Submodules

dpgen.auto_test.ABACUS module

```
class dpgen.auto_test.ABACUS.ABACUS(inter_parameter, path_to_poscar)
```

Bases: [Task](#)

Methods

<i>backward_files</i> ([property_type])	staticmethod(function) -> method
<i>compute</i> (output_dir)	Compute output of the task.
<i>forward_common_files</i> ([property_type])	staticmethod(function) -> method
<i>forward_files</i> ([property_type])	staticmethod(function) -> method
<i>make_input_file</i> (output_dir, task_type, ...)	Prepare input files for a computational task For example, the VASP prepares INCAR.
<i>make_potential_files</i> (output_dir)	Prepare potential files for a computational task.

modify_input	
---------------------	--

```
backward_files(property_type='relaxation')
```

staticmethod(function) -> method

Convert a function to be a static method.

A static method does not receive an implicit first argument. To declare a static method, use this idiom:

```

class C:
    @staticmethod def f(arg1, arg2, ...):
        ...

```

It can be called either on the class (e.g. C.f()) or on an instance (e.g. C().f()). The instance is ignored except for its class.

Static methods in Python are similar to those found in Java or C++. For a more advanced concept, see the classmethod builtin.

```
compute(output_dir)
```

Compute output of the task. IMPORTANT: The output configuration should be converted and stored in a CONTCAR file.

Parameters

output_dir

[str] The directory storing the input and output files.

Returns**result_dict: dict**

A dict that storing the result. For example: { "energy": xxx, "force": [xxx] }

forward_common_files(*property_type='relaxation'*)

staticmethod(function) -> method

Convert a function to be a static method.

A static method does not receive an implicit first argument. To declare a static method, use this idiom:

class C:

```
@staticmethod def f(arg1, arg2, ...):
```

```
...
```

It can be called either on the class (e.g. C.f()) or on an instance (e.g. C().f()). The instance is ignored except for its class.

Static methods in Python are similar to those found in Java or C++. For a more advanced concept, see the classmethod builtin.

forward_files(*property_type='relaxation'*)

staticmethod(function) -> method

Convert a function to be a static method.

A static method does not receive an implicit first argument. To declare a static method, use this idiom:

class C:

```
@staticmethod def f(arg1, arg2, ...):
```

```
...
```

It can be called either on the class (e.g. C.f()) or on an instance (e.g. C().f()). The instance is ignored except for its class.

Static methods in Python are similar to those found in Java or C++. For a more advanced concept, see the classmethod builtin.

make_input_file(*output_dir, task_type, task_param*)

Prepare input files for a computational task For example, the VASP prepares INCAR. LAMMPS (including DeePMD, MEAM...) prepares in.lammps.

Parameters**output_dir**

[str] The directory storing the input files.

task_type

[str] Can be - "relaxation": structure relaxation - "static": static computation calculates the energy, force... of a strcture

task_parame: dict

The parameters of the task. For example the VASP interaction can be provided with { "ediff": 1e-6, "ediffg": 1e-5 }

make_potential_files(*output_dir*)

Prepare potential files for a computational task. For example, the VASP prepares POTCAR. DeePMD prepares frozen model(s). IMPORTANT: Interaction should be stored in output_dir/inter.json

Parameters**output_dir**

[str] The directory storing the potential files.

Outputs**inter.json: output file**

The task information is stored in *output_dir/inter.json*

`modify_input(incar, x, y)`

dpngen.auto_test.EOS module

class dpngen.auto_test.EOS.EOS(*parameter*, *inter_param*=None)

Bases: *Property*

Methods

<code>compute(output_file, print_file, path_to_work)</code>	Postprocess the finished tasks to compute the property.
<code><i>make_confs</i>(path_to_work, path_to_equi[, refine])</code>	Make configurations needed to compute the property.
<code><i>post_process</i>(task_list)</code>	post_process the KPOINTS file in elastic.
<code><i>task_param</i>()</code>	Return the parameter of each computational task, for example, {'ediffg': 1e-4}
<code><i>task_type</i>()</code>	Return the type of each computational task, for example, 'relaxation', 'static'....

`make_confs`(*path_to_work*, *path_to_equi*, *refine*=False)

Make configurations needed to compute the property. The tasks directory will be named as *path_to_work/task.xxxxxx* IMPORTANT: handel the case when the directory exists.

Parameters

`path_to_work`

[str] The path where the tasks for the property are located

`path_to_equi`

[str] -refine == False: The path to the directory that equilibrated the configuration.
-refine == True: The path to the directory that has property confs.

`refine`: str

To refine existing property confs or generate property confs from a equilibrated conf

Returns

`task_list`: list of str

The list of task directories.

`post_process`(*task_list*)

post_process the KPOINTS file in elastic.

`task_param`()

Return the parameter of each computational task, for example, {'ediffg': 1e-4}

`task_type`()

Return the type of each computational task, for example, 'relaxation', 'static'....

dpgen.auto_test.Elastic module

class dpgen.auto_test.Elastic.**Elastic**(parameter, inter_param=None)

Bases: *Property*

Methods

<code>compute(output_file, print_file, path_to_work)</code>	Postprocess the finished tasks to compute the property.
<code>make_confs(path_to_work, path_to_equi[, refine])</code>	Make configurations needed to compute the property.
<code>post_process(task_list)</code>	post_process the KPOINTS file in elastic.
<code>task_param()</code>	Return the parameter of each computational task, for example, {'ediffg': 1e-4}
<code>task_type()</code>	Return the type of each computational task, for example, 'relaxation', 'static'....

make_confs(path_to_work, path_to_equi, refine=False)

Make configurations needed to compute the property. The tasks directory will be named as path_to_work/task.xxxxxx IMPORTANT: handel the case when the directory exists.

Parameters**path_to_work**

[str] The path where the tasks for the property are located

path_to_equi

[str] -refine == False: The path to the directory that equilibrated the configuration.
-refine == True: The path to the directory that has property confs.

refine: str

To refine existing property confs or generate property confs from a equilibrated conf

Returns**task_list: list of str**

The list of task directories.

post_process(task_list)

post_process the KPOINTS file in elastic.

task_param()

Return the parameter of each computational task, for example, {'ediffg': 1e-4}

task_type()

Return the type of each computational task, for example, 'relaxation', 'static'....

dpgen.auto_test.Gamma module

class dpgen.auto_test.Gamma.**Gamma**(parameter, inter_param=None)

Bases: *Property*

Calculation of common gamma lines for bcc and fcc

Methods

<code>compute(output_file, print_file, path_to_work)</code>	Postprocess the finished tasks to compute the property.
<code>make_confs(path_to_work, path_to_equi[, refine])</code>	Make configurations needed to compute the property.
<code>post_process(task_list)</code>	post_process the KPOINTS file in elastic.
<code>task_param()</code>	Return the parameter of each computational task, for example, {'ediffg': 1e-4}
<code>task_type()</code>	Return the type of each computational task, for example, 'relaxation', 'static'....

centralize_slab	
return_direction	

static `centralize_slab(slab)` → None

make_confs(*path_to_work*, *path_to_equi*, *refine=False*)

Make configurations needed to compute the property. The tasks directory will be named as *path_to_work/task.xxxxxx* IMPORTANT: handel the case when the directory exists.

Parameters

path_to_work

[str] The path where the tasks for the property are located

path_to_equi

[str] -refine == False: The path to the directory that equilibrated the configuration.
-refine == True: The path to the directory that has property confs.

refine: str

To refine existing property confs or generate property confs from a equilibrated conf

Returns

task_list: list of str

The list of task directories.

post_process(*task_list*)

post_process the KPOINTS file in elastic.

return_direction()

task_param()

Return the parameter of each computational task, for example, {'ediffg': 1e-4}

task_type()

Return the type of each computational task, for example, 'relaxation', 'static'....

dpgen.auto_test.Interstitial module

class dpgen.auto_test.Interstitial.**Interstitial**(*parameter*, *inter_param=None*)

Bases: *Property*

Methods

<code>compute(output_file, print_file, path_to_work)</code>	Postprocess the finished tasks to compute the property.
<code>make_confs(path_to_work, path_to_equi[, refine])</code>	Make configurations needed to compute the property.
<code>post_process(task_list)</code>	post_process the KPOINTS file in elastic.
<code>task_param()</code>	Return the parameter of each computational task, for example, {'ediffg': 1e-4}
<code>task_type()</code>	Return the type of each computational task, for example, 'relaxation', 'static'....

make_confs(*path_to_work*, *path_to_equi*, *refine=False*)

Make configurations needed to compute the property. The tasks directory will be named as `path_to_work/task.xxxxxx` IMPORTANT: handel the case when the directory exists.

Parameters**path_to_work**

[str] The path where the tasks for the property are located

path_to_equi

[str] -refine == False: The path to the directory that equilibrated the configuration.
-refine == True: The path to the directory that has property confs.

refine: str

To refine existing property confs or generate property confs from a equilibrated conf

Returns**task_list: list of str**

The list of task directories.

post_process(*task_list*)

post_process the KPOINTS file in elastic.

task_param()

Return the parameter of each computational task, for example, {'ediffg': 1e-4}

task_type()

Return the type of each computational task, for example, 'relaxation', 'static'....

dpgen.auto_test.Lammps module

class dpgen.auto_test.Lammps.**Lammps**(*inter_parameter*, *path_to_poscar*)

Bases: *Task*

Methods

<code>backward_files([property_type])</code>	staticmethod(function) -> method
<code>compute(output_dir)</code>	Compute output of the task.
<code>forward_common_files([property_type])</code>	staticmethod(function) -> method
<code>forward_files([property_type])</code>	staticmethod(function) -> method
<code>make_input_file(output_dir, task_type, ...)</code>	Prepare input files for a computational task For example, the VASP prepares INCAR.
<code>make_potential_files(output_dir)</code>	Prepare potential files for a computational task.

<code>set_inter_type_func</code>	
<code>set_model_param</code>	

backward_files(*property_type*='relaxation')

staticmethod(function) -> method

Convert a function to be a static method.

A static method does not receive an implicit first argument. To declare a static method, use this idiom:

```
class C:
    @staticmethod def f(arg1, arg2, ...):
        ...
```

It can be called either on the class (e.g. C.f()) or on an instance (e.g. C().f()). The instance is ignored except for its class.

Static methods in Python are similar to those found in Java or C++. For a more advanced concept, see the classmethod builtin.

compute(*output_dir*)

Compute output of the task. IMPORTANT: The output configuration should be converted and stored in a CONTCAR file.

Parameters

output_dir

[str] The directory storing the input and output files.

Returns

result_dict: dict

A dict that storing the result. For example: { "energy": xxx, "force": [xxx] }

forward_common_files(*property_type*='relaxation')

staticmethod(function) -> method

Convert a function to be a static method.

A static method does not receive an implicit first argument. To declare a static method, use this idiom:

```
class C:
    @staticmethod def f(arg1, arg2, ...):
        ...
```

It can be called either on the class (e.g. C.f()) or on an instance (e.g. C().f()). The instance is ignored except for its class.

Static methods in Python are similar to those found in Java or C++. For a more advanced concept, see the classmethod builtin.

forward_files(*property_type*='relaxation')

staticmethod(function) -> method

Convert a function to be a static method.

A static method does not receive an implicit first argument. To declare a static method, use this idiom:

```
class C:
    @staticmethod def f(arg1, arg2, ...):
        ...
```

It can be called either on the class (e.g. `C.f()`) or on an instance (e.g. `C().f()`). The instance is ignored except for its class.

Static methods in Python are similar to those found in Java or C++. For a more advanced concept, see the `classmethod` builtin.

make_input_file(*output_dir*, *task_type*, *task_param*)

Prepare input files for a computational task. For example, the VASP prepares INCAR. LAMMPS (including DeePMD, MEAM...) prepares `in.lammps`.

Parameters

output_dir

[str] The directory storing the input files.

task_type

[str] Can be - “relaxation”: structure relaxation - “static”: static computation calculates the energy, force... of a structure

task_param: dict

The parameters of the task. For example the VASP interaction can be provided with { “ediff”: 1e-6, “ediffg”: 1e-5 }

make_potential_files(*output_dir*)

Prepare potential files for a computational task. For example, the VASP prepares POTCAR. DeePMD prepares frozen model(s). IMPORTANT: Interaction should be stored in `output_dir/inter.json`

Parameters

output_dir

[str] The directory storing the potential files.

Outputs

inter.json: output file

The task information is stored in `output_dir/inter.json`

set_inter_type_func()

set_model_param()

dpgen.auto_test.Property module

class dpgen.auto_test.Property.**Property**(*parameter*)

Bases: ABC

Attributes

task_param

Return the parameter of each computational task, for example, { ‘ediffg’: 1e-4 }

task_type

Return the type of each computational task, for example, ‘relaxation’, ‘static’....

Methods

<code>compute(output_file, print_file, path_to_work)</code>	Postprocess the finished tasks to compute the property.
<code>make_confs(path_to_work, path_to_equi[, refine])</code>	Make configurations needed to compute the property.
<code>post_process(task_list)</code>	post_process the KPOINTS file in elastic.

compute(*output_file*, *print_file*, *path_to_work*)

Postprocess the finished tasks to compute the property. Output the result to a json database

Parameters

output_file:

The file to output the property in json format

print_file:

The file to output the property in txt format

path_to_work:

The working directory where the computational tasks locate.

abstract make_confs(*path_to_work*, *path_to_equi*, *refine=False*)

Make configurations needed to compute the property. The tasks directory will be named as *path_to_work/task.xxxxxx* IMPORTANT: handel the case when the directory exists.

Parameters

path_to_work

[str] The path where the tasks for the property are located

path_to_equi

[str] -refine == False: The path to the directory that equilibrated the configuration.
-refine == True: The path to the directory that has property confs.

refine: str

To refine existing property confs or generate property confs from a equilibrated conf

Returns

task_list: list of str

The list of task directories.

abstract post_process(*task_list*)

post_process the KPOINTS file in elastic.

abstract property task_param

Return the parameter of each computational task, for example, {'ediffg': 1e-4}

abstract property task_type

Return the type of each computational task, for example, 'relaxation', 'static'...

dpngen.auto_test.Surface module

class dpngen.auto_test.Surface.**Surface**(*parameter*, *inter_param=None*)

Bases: *Property*

Methods

<code>compute(output_file, print_file, path_to_work)</code>	Postprocess the finished tasks to compute the property.
<code>make_confs(path_to_work, path_to_equi[, refine])</code>	Make configurations needed to compute the property.
<code>post_process(task_list)</code>	post_process the KPOINTS file in elastic.
<code>task_param()</code>	Return the parameter of each computational task, for example, {'ediffg': 1e-4}
<code>task_type()</code>	Return the type of each computational task, for example, 'relaxation', 'static'....

make_confs(*path_to_work*, *path_to_equi*, *refine=False*)

Make configurations needed to compute the property. The tasks directory will be named as *path_to_work/task.xxxxxx* IMPORTANT: handel the case when the directory exists.

Parameters

path_to_work

[str] The path where the tasks for the property are located

path_to_equi

[str] -refine == False: The path to the directory that equilibrated the configuration.
-refine == True: The path to the directory that has property confs.

refine: str

To refine existing property confs or generate property confs from a equilibrated conf

Returns

task_list: list of str

The list of task directories.

post_process(*task_list*)

post_process the KPOINTS file in elastic.

task_param()

Return the parameter of each computational task, for example, {'ediffg': 1e-4}

task_type()

Return the type of each computational task, for example, 'relaxation', 'static'....

dpgen.auto_test.Task module

class dpgen.auto_test.Task.**Task**(*inter_parameter*, *path_to_poscar*)

Bases: ABC

Attributes

backward_files

staticmethod(function) -> method

forward_common_files

staticmethod(function) -> method

forward_files

staticmethod(function) -> method

Methods

<code>compute(output_dir)</code>	Compute output of the task.
<code>make_input_file(output_dir, task_type, ...)</code>	Prepare input files for a computational task For example, the VASP prepares INCAR.
<code>make_potential_files(output_dir)</code>	Prepare potential files for a computational task.

abstract property backward_files

staticmethod(function) -> method

Convert a function to be a static method.

A static method does not receive an implicit first argument. To declare a static method, use this idiom:

```
class C:
    @staticmethod def f(arg1, arg2, ...):
        ...
```

It can be called either on the class (e.g. C.f()) or on an instance (e.g. C().f()). The instance is ignored except for its class.

Static methods in Python are similar to those found in Java or C++. For a more advanced concept, see the classmethod builtin.

abstract compute(output_dir)

Compute output of the task. IMPORTANT: The output configuration should be converted and stored in a CONTCAR file.

Parameters

output_dir

[str] The directory storing the input and output files.

Returns

result_dict: dict

A dict that storing the result. For example: { "energy": xxx, "force": [xxx] }

abstract property forward_common_files

staticmethod(function) -> method

Convert a function to be a static method.

A static method does not receive an implicit first argument. To declare a static method, use this idiom:

```
class C:
    @staticmethod def f(arg1, arg2, ...):
        ...
```

It can be called either on the class (e.g. C.f()) or on an instance (e.g. C().f()). The instance is ignored except for its class.

Static methods in Python are similar to those found in Java or C++. For a more advanced concept, see the classmethod builtin.

abstract property forward_files

staticmethod(function) -> method

Convert a function to be a static method.

A static method does not receive an implicit first argument. To declare a static method, use this idiom:

```
class C:
    @staticmethod def f(arg1, arg2, ...):
        ...
```

It can be called either on the class (e.g. `C.f()`) or on an instance (e.g. `C().f()`). The instance is ignored except for its class.

Static methods in Python are similar to those found in Java or C++. For a more advanced concept, see the `classmethod` builtin.

abstract `make_input_file(output_dir, task_type, task_param)`

Prepare input files for a computational task For example, the VASP prepares INCAR. LAMMPS (including DeePMD, MEAM...) prepares `in.lammps`.

Parameters

output_dir

[str] The directory storing the input files.

task_type

[str] Can be - “relaxation”: structure relaxation - “static”: static computation calculates the energy, force... of a strcture

task_param: dict

The parameters of the task. For example the VASP interaction can be provided with { “ediff”: 1e-6, “ediffg”: 1e-5 }

abstract `make_potential_files(output_dir)`

Prepare potential files for a computational task. For example, the VASP prepares POTCAR. DeePMD prepares frozen model(s). IMPORTANT: Interaction should be stored in `output_dir/inter.json`

Parameters

output_dir

[str] The directory storing the potential files.

Outputs

inter.json: output file

The task information is stored in `output_dir/inter.json`

dpgen.auto_test.VASP module

class `dpgen.auto_test.VASP.VASP(inter_parameter, path_to_poscar)`

Bases: *Task*

Methods

<code>backward_files([property_type])</code>	staticmethod(function) -> method
<code>compute(output_dir)</code>	Compute output of the task.
<code>forward_common_files([property_type])</code>	staticmethod(function) -> method
<code>forward_files([property_type])</code>	staticmethod(function) -> method
<code>make_input_file(output_dir, task_type, ...)</code>	Prepare input files for a computational task For example, the VASP prepares INCAR.
<code>make_potential_files(output_dir)</code>	Prepare potential files for a computational task.

backward_files(property_type='relaxation')

staticmethod(function) -> method

Convert a function to be a static method.

A static method does not receive an implicit first argument. To declare a static method, use this idiom:

class C:

 @staticmethod def f(arg1, arg2, ...):

...

It can be called either on the class (e.g. `C.f()`) or on an instance (e.g. `C().f()`). The instance is ignored except for its class.

Static methods in Python are similar to those found in Java or C++. For a more advanced concept, see the `classmethod` builtin.

compute(*output_dir*)

Compute output of the task. IMPORTANT: The output configuration should be converted and stored in a `CONTCAR` file.

Parameters

output_dir

[str] The directory storing the input and output files.

Returns

result_dict: dict

A dict that storing the result. For example: { "energy": xxx, "force": [xxx] }

forward_common_files(*property_type='relaxation'*)

staticmethod(function) -> method

Convert a function to be a static method.

A static method does not receive an implicit first argument. To declare a static method, use this idiom:

class C:

 @staticmethod def f(arg1, arg2, ...):

...

It can be called either on the class (e.g. `C.f()`) or on an instance (e.g. `C().f()`). The instance is ignored except for its class.

Static methods in Python are similar to those found in Java or C++. For a more advanced concept, see the `classmethod` builtin.

forward_files(*property_type='relaxation'*)

staticmethod(function) -> method

Convert a function to be a static method.

A static method does not receive an implicit first argument. To declare a static method, use this idiom:

class C:

 @staticmethod def f(arg1, arg2, ...):

...

It can be called either on the class (e.g. `C.f()`) or on an instance (e.g. `C().f()`). The instance is ignored except for its class.

Static methods in Python are similar to those found in Java or C++. For a more advanced concept, see the `classmethod` builtin.

make_input_file(*output_dir, task_type, task_param*)

Prepare input files for a computational task For example, the VASP prepares `INCAR`. LAMMPS (including DeepMD, MEAM...) prepares `in.lammps`.

Parameters

output_dir

[str] The directory storing the input files.

task_type

[str] Can be - "relaxation": structure relaxation - "static": static computation calculates the energy, force... of a structure

task_param: dict

The parameters of the task. For example the VASP interaction can be provided with { "ediff": 1e-6, "ediffg": 1e-5 }

make_potential_files(*output_dir*)

Prepare potential files for a computational task. For example, the VASP prepares POTCAR. DeePMD prepares frozen model(s). IMPORTANT: Interaction should be stored in *output_dir/inter.json*

Parameters**output_dir**

[str] The directory storing the potential files.

Outputs**inter.json: output file**

The task information is stored in *output_dir/inter.json*

dpgen.auto_test.Vacancy module

class dpgen.auto_test.Vacancy.**Vacancy**(*parameter*, *inter_param=None*)

Bases: *Property*

Methods

<code>compute(output_file, print_file, path_to_work)</code>	Postprocess the finished tasks to compute the property.
<code>make_confs(path_to_work, path_to_equi[, refine])</code>	Make configurations needed to compute the property.
<code>post_process(task_list)</code>	post_process the KPOINTS file in elastic.
<code>task_param()</code>	Return the parameter of each computational task, for example, {'ediffg': 1e-4}
<code>task_type()</code>	Return the type of each computational task, for example, 'relaxation', 'static'....

make_confs(*path_to_work*, *path_to_equi*, *refine=False*)

Make configurations needed to compute the property. The tasks directory will be named as *path_to_work/task.xxxxxx* IMPORTANT: handel the case when the directory exists.

Parameters**path_to_work**

[str] The path where the tasks for the property are located

path_to_equi

[str] -refine == False: The path to the directory that equilibrated the configuration.
-refine == True: The path to the directory that has property confs.

refine: str

To refine existing property confs or generate property confs from a equilibrated conf

Returns**task_list: list of str**

The list of task directories.

post_process(*task_list*)

post_process the KPOINTS file in elastic.

task_param()

Return the parameter of each computational task, for example, {'ediffg': 1e-4}

task_type()

Return the type of each computational task, for example, 'relaxation', 'static'....

dpngen.auto_test.calculator module

`dpngen.auto_test.calculator.make_calculator(inter_parameter, path_to_poscar)`

Make an instance of Task

dpngen.auto_test.common_equi module

`dpngen.auto_test.common_equi.make_equi(confs, inter_param, relax_param)`

`dpngen.auto_test.common_equi.post_equi(confs, inter_param)`

`dpngen.auto_test.common_equi.run_equi(confs, inter_param, mdata)`

dpngen.auto_test.common_prop module

`dpngen.auto_test.common_prop.make_property(confs, inter_param, property_list)`

`dpngen.auto_test.common_prop.make_property_instance(parameters, inter_param)`

Make an instance of Property

`dpngen.auto_test.common_prop.post_property(confs, inter_param, property_list)`

`dpngen.auto_test.common_prop.run_property(confs, inter_param, property_list, mdata)`

`dpngen.auto_test.common_prop.worker(work_path, all_task, forward_common_files, forward_files, backward_files, mdata, inter_type)`

dpngen.auto_test.gen_confs module

`dpngen.auto_test.gen_confs.gen_alloy(eles, key)`

`dpngen.auto_test.gen_confs.gen_ele_std(ele_name, ctype)`

`dpngen.auto_test.gen_confs.gen_element(ele_name, key)`

`dpngen.auto_test.gen_confs.gen_element_std(ele_name)`

`dpngen.auto_test.gen_confs.make_path_mp(ii)`

`dpngen.auto_test.gen_confs.test_fit(struct, data)`

dpngen.auto_test.mpdb module

`dpngen.auto_test.mpdb.check_apikey()`

`dpngen.auto_test.mpdb.get_structure(mp_id)`

dpngen.auto_test.refine module

`dpngen.auto_test.refine.make_refine(init_from_suffix, output_suffix, path_to_work)`

dpngen.auto_test.reproduce module

`dpngen.auto_test.reproduce.make_repro(inter_param, init_data_path, init_from_suffix, path_to_work, reprod_last_frame=True)`

`dpngen.auto_test.reproduce.post_repro(init_data_path, init_from_suffix, all_tasks, ptr_data, reprod_last_frame=True)`

dpngen.auto_test.run module

`dpngen.auto_test.run.gen_test(args)`

`dpngen.auto_test.run.run_task(step, param_file, machine_file=None)`

dpngen.collect package**Submodules****dpngen.collect.collect module**

`dpngen.collect.collect.collect_data(target_folder, param_file, output, verbose=True, shuffle=True, merge=True)`

`dpngen.collect.collect.gen_collect(args)`

dpngen.data package**Subpackages****dpngen.data.tools package****Submodules****dpngen.data.tools.bcc module**

`dpngen.data.tools.bcc.gen_box()`

`dpngen.data.tools.bcc.numb_atoms()`

`dpngen.data.tools.bcc.poscar_unit(latt)`

dpngen.data.tools.cessp2force_lin module

```

dpngen.data.tools.cessp2force_lin.Parser()
dpngen.data.tools.cessp2force_lin.get_outcar_files(directory, recursive)
dpngen.data.tools.cessp2force_lin.process_outcar_file_v5_dev(outcars, data, numbers, types,
                                                             max_types, elements=None,
                                                             windex=None, fout='potfit.configs')
dpngen.data.tools.cessp2force_lin.scan_outcar_file(file_handle)
dpngen.data.tools.cessp2force_lin.uniq(seq)

```

dpngen.data.tools.create_random_disturb module

```

dpngen.data.tools.create_random_disturb.RandomDisturbParser()
dpngen.data.tools.create_random_disturb.create_disturbs_abacus_dev(fin, nfile, dmax=1.0,
                                                                    etmax=0.1, ofmt='abacus',
                                                                    dstyle='uniform',
                                                                    write_d=False, diag=0)
dpngen.data.tools.create_random_disturb.create_disturbs_ase(fin, nfile, dmax=1.0, ofmt='lmp',
                                                             dstyle='uniform', write_d=False)
dpngen.data.tools.create_random_disturb.create_disturbs_ase_dev(fin, nfile, dmax=1.0, etmax=0.1,
                                                                ofmt='lmp', dstyle='uniform',
                                                                write_d=False, diag=0)
dpngen.data.tools.create_random_disturb.create_disturbs_atomsk(fin, nfile, dmax=1.0, ofmt='lmp')
dpngen.data.tools.create_random_disturb.create_random_alloys(fin, alloy_dist, ifmt='vasp',
                                                             ofmt='vasp')

```

In fact, atomsk also gives us the convinient tool to do this

```

dpngen.data.tools.create_random_disturb.gen_random_disturb(dmax, a, b, dstyle='uniform')
dpngen.data.tools.create_random_disturb.gen_random_emat(etmax, diag=0)
dpngen.data.tools.create_random_disturb.random_range(a, b, ndata=1)

```

dpngen.data.tools.diamond module

```

dpngen.data.tools.diamond.gen_box()
dpngen.data.tools.diamond.numb_atoms()
dpngen.data.tools.diamond.poscar_unit(latt)

```

dpgen.data.tools.fcc module

```
dpgen.data.tools.fcc.gen_box()
dpgen.data.tools.fcc.numb_atoms()
dpgen.data.tools.fcc.poscar_unit(latt)
```

dpgen.data.tools.hcp module

```
dpgen.data.tools.hcp.gen_box()
dpgen.data.tools.hcp.numb_atoms()
dpgen.data.tools.hcp.poscar_unit(latt)
```

dpgen.data.tools.io_lammps module

ASE Atoms convert to LAMMPS configuration Some functions are adapted from ASE lammpsrun.py

```
dpgen.data.tools.io_lammps.ase2lammpsdata(atoms, typeids=None, fout='out.lmp')
dpgen.data.tools.io_lammps.car2dir(v, Ainv)
    Cartesian to direct coordinates
dpgen.data.tools.io_lammps.convert_cell(ase_cell)
    Convert a parallel piped (forming right hand basis) to lower triangular matrix LAMMPS can accept. This function
    transposes cell matrix so the bases are column vectors
dpgen.data.tools.io_lammps.convert_forces(forces0, cell0, cell_new)
dpgen.data.tools.io_lammps.convert_positions(pos0, cell0, cell_new, direct=False)
dpgen.data.tools.io_lammps.convert_stress(s6_0, cell0, cell_new)
dpgen.data.tools.io_lammps.dir2car(v, A)
    Direct to cartesian coordinates
dpgen.data.tools.io_lammps.get_atoms_ntypes(atoms)
dpgen.data.tools.io_lammps.get_typeid(typeids, csymbol)
dpgen.data.tools.io_lammps.is_upper_triangular(mat)
    test if 3x3 matrix is upper triangular LAMMPS has a rule for cell matrix definition
dpgen.data.tools.io_lammps.set_atoms_typeids(atoms)
dpgen.data.tools.io_lammps.set_atoms_typeids_with_atomic_numbers(atoms)
dpgen.data.tools.io_lammps.stress6_to_stress9(s6)
dpgen.data.tools.io_lammps.stress9_to_stress6(s9)
```

dpgen.data.tools.ovito_file_convert module

dpgen.data.tools.poscar_copy module

dpgen.data.tools.sc module

`dpgen.data.tools.sc.gen_box()`

`dpgen.data.tools.sc.numb_atoms()`

`dpgen.data.tools.sc.poscar_unit(latt)`

Submodules

dpgen.data.arginfo module

`dpgen.data.arginfo.init_bulk_mdata_arginfo()` → Argument

Generate arginfo for dpgen init_bulk mdata.

Returns

Argument

arginfo

`dpgen.data.arginfo.init_reaction_jdata_arginfo()` → Argument

Generate arginfo for dpgen init_reaction jdata.

Returns

Argument

dpgen init_reaction jdata arginfo

`dpgen.data.arginfo.init_reaction_mdata_arginfo()` → Argument

Generate arginfo for dpgen init_reaction mdata.

Returns

Argument

arginfo

`dpgen.data.arginfo.init_surf_mdata_arginfo()` → Argument

Generate arginfo for dpgen init_surf mdata.

Returns

Argument

arginfo

dpgen.data.gen module

`dpgen.data.gen.class_cell_type(jdata)`

`dpgen.data.gen.coll_abacus_md(jdata)`

`dpgen.data.gen.coll_vasp_md(jdata)`

`dpgen.data.gen.create_path(path, back=False)`

`dpgen.data.gen.gen_init_bulk(args)`

```
dpgen.data.gen.make_abacus_md(jdata, mdata)
dpgen.data.gen.make_abacus_relax(jdata, mdata)
dpgen.data.gen.make_combines(dim, natoms)
dpgen.data.gen.make_scale(jdata)
dpgen.data.gen.make_scale_ABACUS(jdata)
dpgen.data.gen.make_super_cell(jdata)
dpgen.data.gen.make_super_cell_ABACUS(jdata, stru_data)
dpgen.data.gen.make_super_cell_STRU(jdata)
dpgen.data.gen.make_super_cell_poscar(jdata)
dpgen.data.gen.make_unit_cell(jdata)
dpgen.data.gen.make_unit_cell_ABACUS(jdata)
dpgen.data.gen.make_vasp_md(jdata, mdata)
dpgen.data.gen.make_vasp_relax(jdata, mdata)
dpgen.data.gen.out_dir_name(jdata)
dpgen.data.gen.pert_scaled(jdata)
dpgen.data.gen.place_element(jdata)
dpgen.data.gen.place_element_ABACUS(jdata, supercell_stru)
dpgen.data.gen.poscar_ele(poscar_in, poscar_out, eles, natoms)
dpgen.data.gen.poscar_natoms(lines)
dpgen.data.gen.poscar_scale(poscar_in, poscar_out, scale)
dpgen.data.gen.poscar_scale_abacus(poscar_in, poscar_out, scale, jdata)
dpgen.data.gen.poscar_scale_cartesian(str_in, scale)
dpgen.data.gen.poscar_scale_direct(str_in, scale)
dpgen.data.gen.poscar_shuffle(poscar_in, poscar_out)
dpgen.data.gen.replace(file_name, pattern, subst)
dpgen.data.gen.run_abacus_md(jdata, mdata)
dpgen.data.gen.run_abacus_relax(jdata, mdata)
dpgen.data.gen.run_vasp_md(jdata, mdata)
dpgen.data.gen.run_vasp_relax(jdata, mdata)
dpgen.data.gen.shuffle_stru_data(supercell_stru)
dpgen.data.gen.stru_ele(supercell_stru, stru_out, eles, natoms, jdata, path_work)
```

dpngen.data.reaction module

input: trajectory 00: ReaxFF MD (lammps) 01: build dataset (mddatasetbuilder) 02: fp (gaussian) 03: convert to deepmd data output: data

```
dpngen.data.reaction.convert_data(jdata)

dpngen.data.reaction.gen_init_reaction(args)

dpngen.data.reaction.link_fp_input()

dpngen.data.reaction.link_reaxff(jdata)

dpngen.data.reaction.link_trj(jdata)
    link lammppstrj

dpngen.data.reaction.make_lmp(jdata)

dpngen.data.reaction.run_build_dataset(jdata, mdata, log_file='build_log')

dpngen.data.reaction.run_fp(jdata, mdata, log_file='output', forward_common_files=[])

dpngen.data.reaction.run_reaxff(jdata, mdata, log_file='reaxff_log')
```

dpngen.data.surf module

```
dpngen.data.surf.class_cell_type(jdata)

dpngen.data.surf.create_path(path)

dpngen.data.surf.gen_init_surf(args)

dpngen.data.surf.make_combines(dim, natoms)

dpngen.data.surf.make_scale(jdata)

dpngen.data.surf.make_super_cell_pymatgen(jdata)

dpngen.data.surf.make_unit_cell(jdata)

dpngen.data.surf.make_vasp_relax(jdata)

dpngen.data.surf.out_dir_name(jdata)

dpngen.data.surf.pert_scaled(jdata)

dpngen.data.surf.place_element(jdata)

dpngen.data.surf.poscar_ele(poscar_in, poscar_out, eles, natoms)

dpngen.data.surf.poscar_elong(poscar_in, poscar_out, elong, shift_center=True)

dpngen.data.surf.poscar_natoms(poscar_in)

dpngen.data.surf.poscar_scale(poscar_in, poscar_out, scale)

dpngen.data.surf.poscar_scale_cartesian(str_in, scale)
```

```
dpgen.data.surf.poscar_scale_direct(str_in, scale)
```

```
dpgen.data.surf.poscar_shuffle(poscar_in, poscar_out)
```

```
dpgen.data.surf.replace(file_name, pattern, subst)
```

```
dpgen.data.surf.run_vasp_relax(jdata, mdata)
```

dpgen.database package

Submodules

dpgen.database.entry module

```
class dpgen.database.entry.Entry(composition, calculator, inputs, data, entry_id=None, attribute=None, tag=None)
```

Bases: MSONable

An lightweight Entry object containing key computed data for storing purpose.

Attributes

number_element

Methods

<code>as_dict()</code>	A JSON serializable dict representation of an object.
<code>from_dict(d)</code>	
param d	
Dict representation.	
<code>to_json()</code>	Returns a json string representation of the MSONable object.
<code>unsafe_hash()</code>	Returns an hash of the current object.
<code>validate_monty(v)</code>	pydantic Validator for MSONable pattern

`as_dict()`

A JSON serializable dict representation of an object.

classmethod `from_dict(d)`

Parameters

d – Dict representation.

Returns

MSONable class.

property number_element

dpngen.database.run module

`dpngen.database.run.db_run(args)`

`dpngen.database.run.parsing_gaussian(path, output='dpngen_db.json')`

`dpngen.database.run.parsing_pwscf(path, output='dpngen_db.json')`

`dpngen.database.run.parsing_vasp(path, config_info_dict, skip_init, output='dpngen_db.json', id_prefix=None)`

dpngen.database.vasp module

class `dpngen.database.vasp.DPPotcar(symbols=None, functional='PBE', pp_file=None, pp_lists=None)`

Bases: `MSONable`

Methods

<code>as_dict()</code>	A JSON serializable dict representation of an object.
<code>from_dict(d)</code>	
param d	
Dict representation.	
<code>to_json()</code>	Returns a json string representation of the <code>MSONable</code> object.
<code>unsafe_hash()</code>	Returns an hash of the current object.
<code>validate_monty(v)</code>	pydantic Validator for <code>MSONable</code> pattern

from_file	
write_file	

as_dict()

A JSON serializable dict representation of an object.

classmethod from_dict(d)

Parameters

d – Dict representation.

Returns

`MSONable` class.

classmethod from_file(filename)

write_file(filename)

class `dpngen.database.vasp.VaspInput(incar, poscar, potcar, kpoints=None, optional_files=None, **kwargs)`

Bases: `dict`, `MSONable`

Class to contain a set of vasp input objects corresponding to a run.

Args:

`incar`: `Incar` object. `kpoints`: `Kpoints` object. `poscar`: `Poscar` object. `potcar`: `Potcar` object. `optional_files`:

Other input files supplied as a dict of {

`filename`: object}. The object should follow standard `pymatgen` conventions in implementing a `as_dict()` and `from_dict` method.

Methods

<code>as_dict()</code>	A JSON serializable dict representation of an object.
<code>clear()</code>	
<code>copy()</code>	
<code>from_dict(d)</code>	param d Dict representation.
<code>from_directory(input_dir[, optional_files])</code>	Read in a set of VASP input from a directory.
<code>fromkeys(iterable[, value])</code>	Create a new dictionary with keys from iterable and values set to value.
<code>get(key[, default])</code>	Return the value for key if key is in the dictionary, else default.
<code>items()</code>	
<code>keys()</code>	
<code>pop(k[,d])</code>	If key is not found, d is returned if given, otherwise KeyError is raised
<code>popitem()</code>	2-tuple; but raise KeyError if D is empty.
<code>setdefault(key[, default])</code>	Insert key with a value of default if key is not in the dictionary.
<code>to_json()</code>	Returns a json string representation of the MSONable object.
<code>unsafe_hash()</code>	Returns an hash of the current object.
<code>update([E,]**F)</code>	If E is present and has a .keys() method, then does: for k in E: D[k] = E[k] If E is present and lacks a .keys() method, then does: for k, v in E: D[k] = v In either case, this is followed by: for k in F: D[k] = F[k]
<code>validate_monty(v)</code>	pydantic Validator for MSONable pattern
<code>values()</code>	
<code>write_input([output_dir, ...])</code>	Write VASP input to a directory.

as_dict()

A JSON serializable dict representation of an object.

classmethod from_dict(d)**Parameters**

d – Dict representation.

Returns

MSONable class.

static from_directory(input_dir, optional_files=None)

Read in a set of VASP input from a directory. Note that only the standard INCAR, POSCAR, POTCAR and KPOINTS files are read unless optional_filenames is specified.

Args:

input_dir (str): Directory to read VASP input from. optional_files (dict): Optional files to read in as well as a

dict of {filename: Object type}. Object type must have a static method from_file.

write_input(*output_dir='.', make_dir_if_not_present=True*)

Write VASP input to a directory.

Args:

output_dir (str): Directory to write to. Defaults to current directory (“.”).

make_dir_if_not_present (bool): Create the directory if not present. Defaults to True.

dpngen.dispatcher package

Submodules

dpngen.dispatcher.ALI module

dpngen.dispatcher.AWS module

class dpngen.dispatcher.AWS.**AWS**(*context, uuid_names=True*)

Bases: *Batch*

Attributes

job_id

Methods

<i>AWS_check_status</i> (<i>[job_id]</i>)	to avoid query jobStatus too often, set a time interval query_dict example: {job_id: JobStatus}
<i>do_submit</i> (<i>job_dirs, cmd[, args, res, ...]</i>)	submit a single job, assuming that no job is running there.
<i>sub_script</i> (<i>job_dirs, cmd, args, res, outlog, ...</i>)	make submit script

check_finish_tag	
check_status	
default_resources	
map_aws_status_to_dpngen_status	
sub_script_cmd	
sub_script_head	
submit	

classmethod **AWS_check_status**(*job_id=""*)

to avoid query jobStatus too often, set a time interval query_dict example: {job_id: JobStatus}

```
{'40fb24b2-d0ca-4443-8e3a-c0906ea03622': <JobStatus.running: 3>,
 '41bda50c-0a23-4372-806c-87d16a680d85': <JobStatus.waiting: 2>}
```

check_status()

default_resources(*res*)

do_submit(*job_dirs, cmd, args=None, res=None, outlog='log', errlog='err'*)

submit a single job, assuming that no job is running there.

property `job_id`

static `map_aws_status_to_dpgen_status(aws_status)`

sub_script(*job_dirs, cmd, args, res, outlog, errlog*)

make submit script

job_dirs(list): directories of jobs. size: *n_job* *cmd*(list): commands to be executed. size: *n_cmd* *args*(list of list): args of commands. size of *n_cmd* x *n_job*
can be None

res(dict): resources available *outlog*(str): file name for output *errlog*(str): file name for error

dpgen.dispatcher.Batch module

class `dpgen.dispatcher.Batch.Batch(context, uuid_names=True)`

Bases: object

Methods

<code>do_submit(job_dirs, cmd[, args, res, ...])</code>	submit a single job, assuming that no job is running there.
<code>sub_script(job_dirs, cmd[, args, res, ...])</code>	make submit script

check_finish_tag	
check_status	
default_resources	
sub_script_cmd	
sub_script_head	
submit	

check_finish_tag()

check_status()

default_resources(*res*)

do_submit(*job_dirs, cmd, args=None, res=None, outlog='log', errlog='err'*)

submit a single job, assuming that no job is running there.

sub_script(*job_dirs, cmd, args=None, res=None, outlog='log', errlog='err'*)

make submit script

job_dirs(list): directories of jobs. size: *n_job* *cmd*(list): commands to be executed. size: *n_cmd* *args*(list of list): args of commands. size of *n_cmd* x *n_job*
can be None

res(dict): resources available *outlog*(str): file name for output *errlog*(str): file name for error

sub_script_cmd(*cmd, res*)

sub_script_head(*res*)

submit(*job_dirs, cmd, args=None, res=None, restart=False, outlog='log', errlog='err'*)

dpgen.dispatcher.Dispatcher module

```
class dpgen.dispatcher.Dispatcher.Dispatcher(remote_profile, context_type='local', batch_type='slurm',
                                             job_record='jr.json')
```

Bases: object

Methods

all_finished	
run_jobs	
submit_jobs	

```
all_finished(job_handler, mark_failure, clean=True)
```

```
run_jobs(resources, command, work_path, tasks, group_size, forward_common_files, forward_task_files,
          backward_task_files, forward_task_deference=True, mark_failure=False, outlog='log',
          errlog='err')
```

```
submit_jobs(resources, command, work_path, tasks, group_size, forward_common_files, forward_task_files,
             backward_task_files, forward_task_deference=True, outlog='log', errlog='err')
```

```
class dpgen.dispatcher.Dispatcher.JobRecord(path, task_chunks, fname='job_record.json', ip=None)
```

Bases: object

Methods

check_all_finished	
check_finished	
check_nfail	
check_submitted	
dump	
get_uuid	
increase_nfail	
load	
record_finish	
record_remote_context	
valid_hash	

```
check_all_finished()
```

```
check_finished(chunk_hash)
```

```
check_nfail(chunk_hash)
```

```
check_submitted(chunk_hash)
```

```
dump()
```

```
get_uuid(chunk_hash)
```

```
increase_nfail(chunk_hash)
```

load()

record_finish(*chunk_hash*)

record_remote_context(*chunk_hash, local_root, remote_root, job_uuid, ip=None, instance_id=None*)

valid_hash(*chunk_hash*)

dpgen.dispatcher.Dispatcher.make_dispatcher(*mdata, mdata_resource=None, work_path=None, run_tasks=None, group_size=None*)

dpgen.dispatcher.Dispatcher.make_submission(*mdata_machine, mdata_resources, commands, work_path, run_tasks, group_size, forward_common_files, forward_files, backward_files, outlog, errlog*)

dpgen.dispatcher.Dispatcher.make_submission_compat(*machine: dict, resources: dict, commands: List[str], work_path: str, run_tasks: List[str], group_size: int, forward_common_files: List[str], forward_files: List[str], backward_files: List[str], outlog: str = 'log', errlog: str = 'err', api_version: str = '0.9') → None*

Make submission with compatibility of both dispatcher API v0 and v1.

If *api_version* is less than 1.0, use *make_dispatcher*. If *api_version* is large than 1.0, use *make_submission*.

Parameters

machine

[dict] machine dict

resources

[dict] resource dict

commands

[list[str]] list of commands

work_path

[str] working directory

run_tasks

[list[str]] list of paths to running tasks

group_size

[int] group size

forward_common_files

[list[str]] forwarded common files shared for all tasks

forward_files

[list[str]] forwarded files for each task

backward_files

[list[str]] backwarded files for each task

outlog

[str, default=log] path to log from stdout

errlog

[str, default=err] path to log from stderr

api_version

[str, default=0.9] API version. 1.0 is recommended

dpgen.dispatcher.Dispatcher.mdata_arginfo() → List[Argument]

This method generates arginfo for a single mdata.

A submission requires the following keys: command, machine, and resources.

Returns

list[Argument]

arginfo

dpgen.dispatcher.DispatcherList module

class dpgen.dispatcher.DispatcherList.**DispatcherList**(*mdata_machine, mdata_resources, work_path, run_tasks, group_size, cloud_resources=None*)

Bases: object

Methods

<i>catch_dispatcher_exception(ii)</i>	everything is okay: return 0 ssh not active : return 1 machine callback : return 2
<i>check_all_dispatchers_finished(ratio_failure)</i>	
<i>check_dispatcher_status(ii[, allow_failure])</i>	catch running dispatcher exception if no exception occured, check finished
<i>clean()</i>	
<i>create(ii)</i>	case1: use existed machine(finished) to make_dispatcher case2: create one machine, then make_dispatcher, change status from unallocated to unsubmitted
<i>delete(ii)</i>	delete one machine if entity is none, means this ma- chine is used by another dispatcher, shouldn't be deleted
<i>exception_handling(ratio_failure)</i>	
<i>init()</i>	
<i>make_dispatcher(ii)</i>	
<i>run_jobs(resources, command, work_path, ...)</i>	
<i>update()</i>	

catch_dispatcher_exception(ii)

everything is okay: return 0 ssh not active : return 1 machine callback : return 2

check_all_dispatchers_finished(ratio_failure)

check_dispatcher_status(ii, allow_failure=False)

catch running dispatcher exception if no exception occured, check finished

clean()

create(ii)

case1: use existed machine(finished) to make_dispatcher case2: create one machine, then
make_dispatcher, change status from unallocated to unsubmitted

delete(ii)

delete one machine if entity is none, means this machine is used by another dispatcher, shouldn't be deleted

exception_handling(ratio_failure)

init()

make_dispatcher(ii)

run_jobs(*resources, command, work_path, tasks, group_size, forward_common_files, forward_task_files, backward_task_files, forward_task_deference=True, mark_failure=False, outlog='log', errlog='err'*)

update()

class dpngen.dispatcher.DispatcherList.**Entity**(*ip, instance_id, job_record=None, job_handler=None*)

Bases: object

dpngen.dispatcher.JobStatus module

class dpngen.dispatcher.JobStatus.**JobStatus**(*value*)

Bases: Enum

An enumeration.

completing = 6

finished = 5

running = 3

terminated = 4

unknown = 100

unsubmitted = 1

waiting = 2

dpngen.dispatcher.LSF module

class dpngen.dispatcher.LSF.**LSF**(*context, uuid_names=True*)

Bases: *Batch*

Methods

<i>default_resources</i> (res_)	set default value if a key in res_ is not found
<i>do_submit</i> (job_dirs, cmd[, args, res, ...])	submit a single job, assuming that no job is running there.
<i>sub_script</i> (job_dirs, cmd[, args, res, ...])	make submit script

check_finish_tag	
check_status	
sub_script_cmd	
sub_script_head	
submit	

check_status()

default_resources(*res_*)

set default value if a key in **res_** is not found

do_submit(*job_dirs*, *cmd*, *args=None*, *res=None*, *outlog='log'*, *errlog='err'*)

submit a single job, assuming that no job is running there.

sub_script_cmd(*cmd*, *arg*, *res*)

sub_script_head(*res*)

dpngen.dispatcher.LazyLocalContext module

class dpngen.dispatcher.LazyLocalContext.LazyLocalContext(*local_root*, *work_profile=None*,
job_uuid=None)

Bases: object

Methods

block_call	
block_checkcall	
call	
check_file_exists	
check_finish	
clean	
download	
get_job_root	
get_return	
kill	
read_file	
upload	
write_file	

block_call(*cmd*)

block_checkcall(*cmd*)

call(*cmd*)

check_file_exists(*fname*)

check_finish(*proc*)

clean()

download(*job_dirs*, *remote_down_files*, *check_exists=False*, *mark_failure=True*, *back_error=False*)

get_job_root()

get_return(*proc*)

kill(*proc*)

```
read_file(fname)
```

```
upload(job_dirs, local_up_files, dereference=True)
```

```
write_file(fname, write_str)
```

```
class dpngen.dispatcher.LazyLocalContext.SPRetObj(ret)
```

```
    Bases: object
```

Methods

read	
readlines	

```
read()
```

```
readlines()
```

dpngen.dispatcher.LocalContext module

```
class dpngen.dispatcher.LocalContext.LocalContext(local_root, work_profile, job_uuid=None)
```

```
    Bases: object
```

Methods

block_call	
block_checkcall	
call	
check_file_exists	
check_finish	
clean	
download	
get_job_root	
get_return	
kill	
read_file	
upload	
write_file	

```
block_call(cmd)
```

```
block_checkcall(cmd)
```

```
call(cmd)
```

```
check_file_exists(fname)
```

```
check_finish(proc)
```

```
clean()
```

```
download(job_dirs, remote_down_files, check_exists=False, mark_failure=True, back_error=False)
```

```
get_job_root()
```

```
get_return(proc)
```

```
kill(proc)
```

```
read_file(fname)
```

```
upload(job_dirs, local_up_files, dereference=True)
```

```
write_file(fname, write_str)
```

```
class dpgen.dispatcher.LocalContext.LocalSession(jdata)
```

```
Bases: object
```

Methods

get_work_root	
----------------------	--

```
get_work_root()
```

```
class dpgen.dispatcher.LocalContext.SPRetObj(ret)
```

```
Bases: object
```

Methods

read	
readlines	

```
read()
```

```
readlines()
```

dpgen.dispatcher.PBS module

```
class dpgen.dispatcher.PBS.PBS(context, uuid_names=True)
```

```
Bases: Batch
```

Methods

<i>default_resources</i> (res_)	set default value if a key in res_ is not found
<i>do_submit</i> (job_dirs, cmd[, args, res, ...])	submit a single job, assuming that no job is running there.
<i>sub_script</i> (job_dirs, cmd[, args, res, ...])	make submit script

check_finish_tag	
check_status	
sub_script_cmd	
sub_script_head	
submit	

check_status()

default_resources(*res_*)

set default value if a key in **res_** is not found

do_submit(*job_dirs*, *cmd*, *args=None*, *res=None*, *outlog='log'*, *errlog='err'*)

submit a single job, assuming that no job is running there.

sub_script_cmd(*cmd*, *arg*, *res*)

sub_script_head(*res*)

dpgen.dispatcher.SSHContext module

class dpgen.dispatcher.SSHContext.**SSHContext**(*local_root*, *ssh_session*, *job_uuid=None*)

Bases: object

Attributes

 sftp

 ssh

Methods

block_call	
block_checkcall	
call	
check_file_exists	
check_finish	
clean	
close	
download	
get_job_root	
get_return	
kill	
read_file	
upload	
write_file	

block_call(*cmd*)

block_checkcall(*cmd*, *retry=0*)

call(*cmd*)

check_file_exists(*fname*)

```
check_finish(cmd_pipes)

clean()

close()

download(job_dirs, remote_down_files, check_exists=False, mark_failure=True, back_error=False)

get_job_root()

get_return(cmd_pipes)

kill(cmd_pipes)

read_file(fname)

property sftp

property ssh

upload(job_dirs, local_up_files, dereference=True)

write_file(fname, write_str)

class dpngen.dispatcher.SSHContext.SSHSession(jdata)
    Bases: object
    Attributes
        sftp
            Returns sftp.
```

Methods

<i>exec_command</i> (cmd[, retry])	Calling self.ssh.exec_command but has an exception check.
------------------------------------	---

close	
ensure_alive	
get_session_root	
get_ssh_client	

```
close()

ensure_alive(max_check=10, sleep_time=10)

exec_command(cmd, retry=0)
    Calling self.ssh.exec_command but has an exception check.

get_session_root()

get_ssh_client()

property sftp
    Returns sftp. Open a new one if not existing.
```

dpgen.dispatcher.Shell module

class dpgen.dispatcher.Shell.**Shell**(context, uuid_names=True)

Bases: *Batch*

Methods

<i>do_submit</i> (job_dirs, cmd[, args, res, ...])	submit a single job, assuming that no job is running there.
<i>sub_script</i> (job_dirs, cmd[, args, res, ...])	make submit script

check_finish_tag	
check_running	
check_status	
default_resources	
sub_script_cmd	
sub_script_head	
submit	

check_running()

check_status()

default_resources(res_)

do_submit(job_dirs, cmd, args=None, res=None, outlog='log', errlog='err')
submit a single job, assuming that no job is running there.

sub_script_cmd(cmd, arg, res)

sub_script_head(resources)

dpgen.dispatcher.Slurm module

class dpgen.dispatcher.Slurm.**Slurm**(context, uuid_names=True)

Bases: *Batch*

Methods

<i>check_status</i> ()	check the status of a job
<i>default_resources</i> (res_)	set default value if a key in res_ is not found
<i>do_submit</i> (job_dirs, cmd[, args, res, ...])	submit a single job, assuming that no job is running there.
<i>sub_script</i> (job_dirs, cmd[, args, res, ...])	make submit script

check_finish_tag	
sub_script_cmd	
sub_script_head	
submit	

check_status()

check the status of a job

default_resources(res_)

set default value if a key in **res_** is not found

do_submit(job_dirs, cmd, args=None, res=None, outlog='log', errlog='err')

submit a single job, assuming that no job is running there.

sub_script_cmd(cmd, arg, res)**sub_script_head(res)****dpngen.generator package****Subpackages****dpngen.generator.lib package****Submodules****dpngen.generator.lib.abacus_scf module**

`dpngen.generator.lib.abacus_scf.get_abacus_STRU(STRU, INPUT=None, n_ele=None)`

`dpngen.generator.lib.abacus_scf.get_abacus_input_parameters(INPUT)`

`dpngen.generator.lib.abacus_scf.get_additional_from_STRU(geometry_inlines, nele)`

`dpngen.generator.lib.abacus_scf.get_mass_from_STRU(geometry_inlines, inlines, atom_names)`

`dpngen.generator.lib.abacus_scf.get_natoms_from_stru(geometry_inlines)`

`dpngen.generator.lib.abacus_scf.make_abacus_scf_input(fp_params)`

`dpngen.generator.lib.abacus_scf.make_abacus_scf_kpt(fp_params)`

`dpngen.generator.lib.abacus_scf.make_abacus_scf_stru(sys_data, fp_pp_files, fp_orb_files=None,
fp_dpks_descriptor=None, fp_params=None)`

`dpngen.generator.lib.abacus_scf.make_kspacing_kpoints_stru(stru, kspacing)`

`dpngen.generator.lib.abacus_scf.make_supercell_abacus(from_struct, super_cell)`

dpngen.generator.lib.calypso_check_outcar module**dpngen.generator.lib.calypso_run_model_devi module****dpngen.generator.lib.calypso_run_opt module****dpngen.generator.lib.cp2k module**

`dpgen.generator.lib.cp2k.iterdict(d, out_list, flag=None)`

Doc

a recursive expansion of dictionary into cp2k input

K

current key

V

current value

D

current dictionary under expansion

Flag

used to record dictionary state. if flag is None,
it means we are in top level dict. flag is a string.

`dpgen.generator.lib.cp2k.make_cp2k_input(sys_data, fp_params)`

`dpgen.generator.lib.cp2k.make_cp2k_input_from_external(sys_data, exinput_path)`

`dpgen.generator.lib.cp2k.make_cp2k_xyz(sys_data)`

`dpgen.generator.lib.cp2k.update_dict(old_d, update_d)`

a method to recursive update dict :old_d: old dictionary :update_d: some update value written in dictionary form

dpgen.generator.lib.cvasp module

`dpgen.generator.lib.cvasp.runvasp(cmd, opt=False, max_errors=3, backup=False, auto_gamma=False,
auto_npar=False, ediffg=-0.05)`

cmd example: cmd=['mpirun', '-np', '32', '-machinefile', 'hosts', 'vasp_std']

dpgen.generator.lib.ele_temp module

`class dpgen.generator.lib.ele_temp.NBandsEsti(test_list)`

Bases: object

Methods

predict	
save	

`predict(target_dir, tolerance=0.5)`

`save(fname)`

dpngen.generator.lib.gaussian module

```
dpngen.generator.lib.gaussian.detect_multiplicity(symbols)
dpngen.generator.lib.gaussian.make_gaussian_input(sys_data, fp_params)
dpngen.generator.lib.gaussian.take_cluster(old_conf_name, type_map, idx, jdata)
```

dpngen.generator.lib.lammps module

```
dpngen.generator.lib.lammps.get_all_dumped_forces(file_name)
dpngen.generator.lib.lammps.get_dumped_forces(file_name)
dpngen.generator.lib.lammps.make_lammps_input(ensemble, conf_file, graphs, nsteps, dt, neidelay, trj_freq,
                                              mass_map, temp, jdata, tau_t=0.1, pres=None,
                                              tau_p=0.5, pka_e=None, ele_temp_f=None,
                                              ele_temp_a=None, max_seed=1000000, nopbc=False,
                                              deepmd_version='0.1')
```

dpngen.generator.lib.make_calypso module

```
dpngen.generator.lib.make_calypso.make_calypso_input(nameofatoms, numberofatoms, numberofformula,
                                                    volume, distanceofion, psoratio, popsize,
                                                    maxstep, icode, split, vsc, maxnumatom,
                                                    ctrlrange, pstress, fmax)
dpngen.generator.lib.make_calypso.write_model_devi_out(devi, fname)
```

dpngen.generator.lib.parse_calypso module**dpngen.generator.lib.pwmat module**

```
dpngen.generator.lib.pwmat.input_upper(dinput)
dpngen.generator.lib.pwmat.make_pwmat_input_dict(node1, node2, atom_config, ecut, e_error, rho_error,
                                                  icmix=None, smearing=None, sigma=None,
                                                  kspacing=0.5, flag_symm=None)
dpngen.generator.lib.pwmat.make_pwmat_input_user_dict(fp_params)
dpngen.generator.lib.pwmat.write_input_dict(input_dict)
```

dpgen.generator.lib.pwscf module

```
dpgen.generator.lib.pwscf.cvt_1frame(fin, fout)
dpgen.generator.lib.pwscf.get_atom_types(lines)
dpgen.generator.lib.pwscf.get_block(lines, keyword, skip=0)
dpgen.generator.lib.pwscf.get_cell(lines)
dpgen.generator.lib.pwscf.get_coords(lines)
dpgen.generator.lib.pwscf.get_energy(lines)
dpgen.generator.lib.pwscf.get_force(lines)
dpgen.generator.lib.pwscf.get_natoms(lines)
dpgen.generator.lib.pwscf.get_stress(lines, cells)
dpgen.generator.lib.pwscf.get_types(lines)
dpgen.generator.lib.pwscf.make_pwscf_01_runctrl_dict(sys_data, idict)
dpgen.generator.lib.pwscf.make_pwscf_input(sys_data, fp_pp_files, fp_params, user_input=True)
```

dpgen.generator.lib.run_calypso module

calypso as model devi engine:

1. gen_structures
2. analysis
3. model devi

```
dpgen.generator.lib.run_calypso.analysis(iter_index, jdata, calypso_model_devi_path)
dpgen.generator.lib.run_calypso.gen_main(iter_index, jdata, mdata, caly_run_opt_list, gen_idx)
dpgen.generator.lib.run_calypso.gen_structures(iter_index, jdata, mdata, caly_run_path, current_idx,
length_of_caly_runopt_list)
dpgen.generator.lib.run_calypso.run_calypso_model_devi(iter_index, jdata, mdata)
```

dpgen.generator.lib.siesta module

```
dpgen.generator.lib.siesta.make_siesta_input(sys_data, fp_pp_files, fp_params)
```

dpgen.generator.lib.utils module

```

dpgen.generator.lib.utils.cmd_append_log(cmd, log_file)
dpgen.generator.lib.utils.copy_file_list(file_list, from_path, to_path)
dpgen.generator.lib.utils.create_path(path)
dpgen.generator.lib.utils.log_iter(task, ii, jj)
dpgen.generator.lib.utils.log_task(message)
dpgen.generator.lib.utils.make_iter_name(iter_index)
dpgen.generator.lib.utils.record_iter(record, ii, jj)
dpgen.generator.lib.utils.repeat_to_length(string_to_expand, length)
dpgen.generator.lib.utils.replace(file_name, pattern, subst)
dpgen.generator.lib.utils.symlink_user_forward_files(mdata, task_type, work_path,
                                                    task_format=None)
    Symlink user-defined forward_common_files Current path should be work_path, such as 00.train
mdata
    [dict] machine parameters
task_type: str
    task_type, such as “train”
work_path
    [str] work_path, such as “iter.000001/00.train”
    None

```

dpgen.generator.lib.vasp module

```

dpgen.generator.lib.vasp.incar_upper(dincar)
dpgen.generator.lib.vasp.make_vasp_incar_user_dict(fp_params)
dpgen.generator.lib.vasp.write_incar_dict(incar_dict)

```

Submodules

dpgen.generator.arginfo module

```

dpgen.generator.arginfo.basic_args() → List[Argument]
dpgen.generator.arginfo.data_args() → List[Argument]
dpgen.generator.arginfo.fp_args() → List[Argument]
dpgen.generator.arginfo.fp_style_abacus_args() → List[Argument]

```

`dpgen.generator.arginfo.fp_style_amber_diff_args()` → List[Argument]

Arguments for FP style amber/diff.

Returns

list[dargs.Argument]

list of Gaussian fp style arguments

`dpgen.generator.arginfo.fp_style_cp2k_args()` → List[Argument]

`dpgen.generator.arginfo.fp_style_gaussian_args()` → List[Argument]

Gaussian fp style arguments.

Returns

list[dargs.Argument]

list of Gaussian fp style arguments

`dpgen.generator.arginfo.fp_style_siesta_args()` → List[Argument]

`dpgen.generator.arginfo.fp_style_variant_type_args()` → Variant

`dpgen.generator.arginfo.fp_style_vasp_args()` → List[Argument]

`dpgen.generator.arginfo.model_devi_amber_args()` → List[Argument]

Amber engine arguments.

`dpgen.generator.arginfo.model_devi_args()` → List[Variant]

`dpgen.generator.arginfo.model_devi_jobs_args()` → List[Argument]

`dpgen.generator.arginfo.model_devi_lmp_args()` → List[Argument]

`dpgen.generator.arginfo.run_jdata_arginfo()` → Argument

Argument information for dpngen run mdata.

Returns

Argument

argument information

`dpgen.generator.arginfo.run_mdata_arginfo()` → Argument

Generate arginfo for dpngen run mdata.

Returns

Argument

arginfo

`dpgen.generator.arginfo.training_args()` → List[Argument]

Traning arguments.

Returns

list[dargs.Argument]

List of training arguments.

dpngen.generator.run module

init: data iter:

00.train 01.model_devi 02.vasp 03.data

`dpngen.generator.run.check_bad_box(conf_name, criteria, fmt='lammps/dump')`

`dpngen.generator.run.check_cluster(conf_name, fp_cluster_vacuum, fmt='lammps/dump')`

```

dpgen.generator.run.copy_model(numb_model, prv_iter_index, cur_iter_index)
dpgen.generator.run.detect_batch_size(batch_size, system=None)
dpgen.generator.run.dump_to_deepmd_raw(dump, deepmd_raw, type_map, fmt='gromacs/gro',
                                       charge=None)
dpgen.generator.run.expand_idx(in_list)
dpgen.generator.run.expand_matrix_values(target_list, cur_idx=0)
dpgen.generator.run.find_only_one_key(lmp_lines, key)
dpgen.generator.run.gen_run(args)
dpgen.generator.run.get_atomic_masses(atom)
dpgen.generator.run.get_job_names(jdata)
dpgen.generator.run.get_sys_index(task)
dpgen.generator.run.make_fp(iter_index, jdata, mdata)
dpgen.generator.run.make_fp_abacus_scf(iter_index, jdata)
dpgen.generator.run.make_fp_amber_diff(iter_index: int, jdata: dict)

```

Run amber twice to calculate high-level and low-level potential, and then generate difference between them.

Besides AMBER, one needs to install *dpamber* package, which is available at <https://github.com/njzjz/dpamber>

Currently, it should be used with the AMBER model_devi driver.

Parameters

iter_index

[int] iter index

jdata

[dict]

Run parameters. The following parameters are used in this method:

mdin_prefix

[str] The path prefix to AMBER mdin files

qm_region

[list[str]] AMBER mask of the QM region. Each mask maps to a system.

qm_charge

[list[int]] Charge of the QM region. Each charge maps to a system.

high_level

[str] high level method

low_level

[str] low level method

fp_params

[dict]

This parameters includes:

high_level_mdin

[str] High-level AMBER mdin file. %qm_theory%, %qm_region%, and %qm_charge% will be replace.

low_level_mdin

[str] Low-level AMBER mdin file. %qm_theory%, %qm_region%, and %qm_charge% will be replace.

parm7_prefix

[str] The path prefix to AMBER PARM7 files

parm7

[list[str]] List of paths to AMBER PARM7 files. Each file maps to a system.

References

[1]

```
dpgen.generator.run.make_fp_cp2k(iter_index, jdata)
dpgen.generator.run.make_fp_gaussian(iter_index, jdata)
dpgen.generator.run.make_fp_pwmat(iter_index, jdata)
dpgen.generator.run.make_fp_pwscf(iter_index, jdata)
dpgen.generator.run.make_fp_siesta(iter_index, jdata)
dpgen.generator.run.make_fp_task_name(sys_idx, counter)
dpgen.generator.run.make_fp_vasp(iter_index, jdata)
dpgen.generator.run.make_fp_vasp_cp_cvasp(iter_index, jdata)
dpgen.generator.run.make_fp_vasp_incar(iter_index, jdata, nbands_esti=None)
dpgen.generator.run.make_fp_vasp_kp(iter_index, jdata)
dpgen.generator.run.make_model_devi(iter_index, jdata, mdata)
dpgen.generator.run.make_model_devi_conf_name(sys_idx, conf_idx)
dpgen.generator.run.make_model_devi_task_name(sys_idx, task_idx)
dpgen.generator.run.make_pwmat_input(jdata, filename)
dpgen.generator.run.make_train(iter_index, jdata, mdata)
dpgen.generator.run.make_vasp_incar(jdata, filename)
dpgen.generator.run.make_vasp_incar_ele_temp(jdata, filename, ele_temp, nbands_esti=None)
dpgen.generator.run.parse_cur_job(cur_job)
dpgen.generator.run.parse_cur_job_revmat(cur_job, use_plm=False)
dpgen.generator.run.parse_cur_job_sys_revmat(cur_job, sys_idx, use_plm=False)
dpgen.generator.run.poscar_natoms(lines)
dpgen.generator.run.poscar_shuffle(poscar_in, poscar_out)
dpgen.generator.run.poscar_to_conf(poscar, conf)
```

```

dpgen.generator.run.post_fp(iter_index, jdata)
dpgen.generator.run.post_fp_abacus_scf(iter_index, jdata)
dpgen.generator.run.post_fp_amber_diff(iter_index, jdata)
dpgen.generator.run.post_fp_check_fail(iter_index, jdata, rfailed=None)
dpgen.generator.run.post_fp_cp2k(iter_index, jdata, rfailed=None)
dpgen.generator.run.post_fp_gaussian(iter_index, jdata)
dpgen.generator.run.post_fp_pwmat(iter_index, jdata, rfailed=None)
dpgen.generator.run.post_fp_pwscf(iter_index, jdata)
dpgen.generator.run.post_fp_siesta(iter_index, jdata)
dpgen.generator.run.post_fp_vasp(iter_index, jdata, rfailed=None)
dpgen.generator.run.post_model_devi(iter_index, jdata, mdata)
dpgen.generator.run.post_train(iter_index, jdata, mdata)
dpgen.generator.run.revise_by_keys(lmp_lines, keys, values)
dpgen.generator.run.revise_lmp_input_dump(lmp_lines, trj_freq)
dpgen.generator.run.revise_lmp_input_model(lmp_lines, task_model_list, trj_freq, deepmd_version='1')
dpgen.generator.run.revise_lmp_input_plm(lmp_lines, in_plm, out_plm='output.plumed')
dpgen.generator.run.run_fp(iter_index, jdata, mdata)
dpgen.generator.run.run_fp_inner(iter_index, jdata, mdata, forward_files, backward_files, check_fin,
                                log_file='fp.log', forward_common_files=[])
dpgen.generator.run.run_iter(param_file, machine_file)
dpgen.generator.run.run_md_model_devi(iter_index, jdata, mdata)
dpgen.generator.run.run_model_devi(iter_index, jdata, mdata)
dpgen.generator.run.run_train(iter_index, jdata, mdata)
dpgen.generator.run.set_version(mdata)
dpgen.generator.run.sys_link_fp_vasp_pp(iter_index, jdata)
dpgen.generator.run.update_mass_map(jdata)

```

dpgen.remote package

Submodules

dpgen.remote.RemoteJob module

class dpgen.remote.RemoteJob.CloudMachineJob(*ssh_session, local_root, job_uuid=None*)

Bases: [RemoteJob](#)

Methods

block_call	
block_checkcall	
check_status	
clean	
download	
get_job_root	
submit	
upload	

check_status()

submit(*job_dirs, cmd, args=None, resources=None*)

class dpgen.remote.RemoteJob.JobStatus(*value*)

Bases: Enum

An enumeration.

finished = 5

running = 3

terminated = 4

unknown = 100

unsubmitted = 1

waiting = 2

class dpgen.remote.RemoteJob.LSFJob(*ssh_session, local_root, job_uuid=None*)

Bases: [RemoteJob](#)

Methods

block_call	
block_checkcall	
check_limit	
check_status	
clean	
download	
get_job_root	
submit	
upload	

check_limit(*task_max*)

check_status()

submit(*job_dirs*, *cmd*, *args=None*, *resources=None*, *restart=False*)

class dpgen.remote.RemoteJob.**PBSJob**(*ssh_session*, *local_root*, *job_uuid=None*)

Bases: [RemoteJob](#)

Methods

block_call	
block_checkcall	
check_status	
clean	
download	
get_job_root	
submit	
upload	

check_status()

submit(*job_dirs*, *cmd*, *args=None*, *resources=None*)

class dpgen.remote.RemoteJob.**RemoteJob**(*ssh_session*, *local_root*, *job_uuid=None*)

Bases: object

Methods

block_call	
block_checkcall	
clean	
download	
get_job_root	
upload	

block_call(*cmd*)

```
block_checkcall(cmd)  
clean()  
download(job_dirs, remote_down_files, back_error=False)  
get_job_root()  
upload(job_dirs, local_up_files, dereference=True)
```

```
class dpgen.remote.RemoteJob.SSHSession(jdata)  
    Bases: object
```

Methods

close	
get_session_root	
get_ssh_client	

```
close()  
get_session_root()  
get_ssh_client()
```

```
class dpgen.remote.RemoteJob.SlurmJob(ssh_session, local_root, job_uuid=None)  
    Bases: RemoteJob
```

Methods

block_call	
block_checkcall	
check_status	
clean	
download	
get_job_root	
submit	
upload	

```
check_status()  
submit(job_dirs, cmd, args=None, resources=None, restart=False)
```

```
class dpgen.remote.RemoteJob.awsMachineJob(remote_root, work_path, job_uuid=None)  
    Bases: object
```

Methods

download	
upload	

download(*job_dir*, *remote_down_files*, *dereference=True*)

upload(*job_dir*, *local_up_files*, *dereference=True*)

dpgen.remote.decide_machine module

dpgen.remote.decide_machine.**convert_mdata**(*mdata*, *task_types*=['train', 'model_devi', 'fp'])

Convert mdata for DP-GEN main process. New conversion is like mdata["fp"]["machine"], DP-GEN needs mdata["fp_machine"]

Notice that we deprecate the function which can automatically select one most available machine, since this function was only used by Angus, and only supports for Slurm. In the future this can be implemented.

Parameters

mdata

[dict] Machine parameters to be converted.

task_types

[list of string] Type of tasks, default is ["train", "model_devi", "fp"]

Returns

dict

mdata converted

dpgen.remote.group_jobs module

class dpgen.remote.group_jobs.**PMap**(*path*, *fname*='pmap.json')

Bases: object

Path map class to operate {read,write,delte} the pmap.json file

Methods

delete	
dump	
load	

delete()

dump(*pmap*, *indent=4*)

load()

dpgen.remote.group_jobs.**aws_submit_jobs**(*machine*, *resources*, *command*, *work_path*, *tasks*, *group_size*, *forward_common_files*, *forward_task_files*, *backward_task_files*, *forward_task_deference=True*)

```
dpgen.remote.group_jobs.group_local_jobs(ssh_sess, resources, command, work_path, tasks, group_size,  
                                           forward_common_files, forward_task_files,  
                                           backward_task_files, forward_task_deference=True)
```

```
dpgen.remote.group_jobs.group_slurm_jobs(ssh_sess, resources, command, work_path, tasks, group_size,  
                                           forward_common_files, forward_task_files,  
                                           backward_task_files, remote_job=<class  
                                           'dpgen.remote.RemoteJob.SlurmJob'>,  
                                           forward_task_deference=True)
```

```
dpgen.remote.group_jobs.ucloud_submit_jobs(machine, resources, command, work_path, tasks, group_size,  
                                           forward_common_files, forward_task_files,  
                                           backward_task_files, forward_task_deference=True)
```

dpgen.simplify package

Submodules

dpgen.simplify.arginfo module

```
dpgen.simplify.arginfo.fp_args() → List[Argument]
```

Generate arginfo for fp.

Returns

List[Argument]

arginfo

```
dpgen.simplify.arginfo.fp_style_variant_type_args() → Variant
```

Generate variant for fp style variant type.

Returns

Variant

variant for fp style

```
dpgen.simplify.arginfo.general_simplify_arginfo() → Argument
```

General simplify arginfo.

Returns

Argument

arginfo

```
dpgen.simplify.arginfo.simplify_jdata_arginfo() → Argument
```

Generate arginfo for dpgen simplify jdata.

Returns

Argument

arginfo

```
dpgen.simplify.arginfo.simplify_mdata_arginfo() → Argument
```

Generate arginfo for dpgen simplify mdata.

Returns

Argument

arginfo

dpgen.simplify.simplify module

Simplify dataset (minimize the dataset size).

Init: pick up init data from dataset randomly

Iter: 00: train models (same as generator) 01: calculate model deviations of the rest dataset, pick up data with proper model deviation 02: fp (optional, if the original dataset do not have fp data, same as generator)

`dpgen.simplify.simplify.gen_simplify(args)`

`dpgen.simplify.simplify.get_multi_system(path, jdata)`

`dpgen.simplify.simplify.get_system_cls(jdata)`

`dpgen.simplify.simplify.init_model(iter_index, jdata, mdata)`

`dpgen.simplify.simplify.init_pick(iter_index, jdata, mdata)`

pick up init data from dataset randomly

`dpgen.simplify.simplify.make_fp(iter_index, jdata, mdata)`

`dpgen.simplify.simplify.make_fp_calculation(iter_index, jdata)`

`dpgen.simplify.simplify.make_fp_configs(iter_index, jdata)`

`dpgen.simplify.simplify.make_fp_gaussian(iter_index, jdata)`

`dpgen.simplify.simplify.make_fp_labeled(iter_index, jdata)`

`dpgen.simplify.simplify.make_fp_vasp(iter_index, jdata)`

`dpgen.simplify.simplify.make_model_devi(iter_index, jdata, mdata)`

calculate the model deviation of the rest idx

`dpgen.simplify.simplify.post_model_devi(iter_index, jdata, mdata)`

calculate the model deviation

`dpgen.simplify.simplify.run_iter(param_file, machine_file)`

init (iter 0): init_pick

tasks (iter > 0): 00 make_train (same as generator) 01 run_train (same as generator) 02 post_train (same as generator) 03 make_model_devi 04 run_model_devi 05 post_model_devi 06 make_fp 07 run_fp (same as generator) 08 post_fp (same as generator)

`dpgen.simplify.simplify.run_model_devi(iter_index, jdata, mdata)`

submit dp test tasks

dpgen.tools package

Submodules

dpgen.tools.auto_gen_param module

```
class dpngen.tools.auto_gen_param.Iteration(temps, nsteps_list=[500, 500, 1000, 1000, 3000, 3000, 6000,
6000], sub_iteration_num=8, ensemble='npt', press=[1.0,
10.0, 100.0, 1000.0, 5000.0, 10000.0, 20000.0, 50000.0],
trj_freq=10)
```

Bases: object

Attributes

`index_iteration`

Methods

<code>gen_sub_iter</code>	
<code>register_iteration</code>	
<code>register_sub_iteartion</code>	

```
current_num_of_itearation = 0
```

```
current_num_of_sub_itearation = 0
```

```
gen_sub_iter(system_list)
```

```
property index_iteration
```

```
classmethod register_iteration()
```

```
classmethod register_sub_iteartion()
```

```
class dpngen.tools.auto_gen_param.System(system_prefix="")
```

Bases: object

Attributes

`index_system`

Methods

<code>add_sub_system</code>	
<code>get_sub_system</code>	
<code>register_sub_system</code>	
<code>register_system</code>	

```
add_sub_system(idx2, files_list)
```

```
current_num_of_sub_systems = 0
```

```
current_num_of_system = 0
```

```
get_sub_system()
```

```
property index_system
```

```
classmethod register_sub_system()
```

```
classmethod register_system()
```

```

dpngen.tools.auto_gen_param.auto_gen_param(args)

dpngen.tools.auto_gen_param.default_map_generator(map_list=[1, 1, 2, 2, 2, 4, 4, 4], data_list=None)

dpngen.tools.auto_gen_param.default_temps_generator(melt_point, temps_interval=0.1, num_temps=5)

dpngen.tools.auto_gen_param.get_basic_param_json(melt_point, out_param_filename='param_basic.json',
                                                  scan_dir='.', file_name='POSCAR',
                                                  init_file_name='type.raw', min_allow_files_num=16,
                                                  map_list=[1, 1, 2, 2, 2, 4, 4, 4], meta_iter_num=4,
                                                  sub_iteration_num=8, map_iterator=None,
                                                  nsteps_list=[500, 500, 1000, 1000, 3000, 3000, 6000,
                                                  6000], press=[1.0, 10.0, 100.0, 1000.0, 5000.0,
                                                  10000.0, 20000.0, 50000.0], temps_iterator=None,
                                                  ensemble='npt', trj_freq=10, temps_interval=0.1,
                                                  num_temps=5)

dpngen.tools.auto_gen_param.get_init_data_sys(scan_dir='.', init_file_name='type.raw')

dpngen.tools.auto_gen_param.get_model_devi_jobs(melt_point, system_list, nsteps_list=[500, 500, 1000,
1000, 3000, 3000, 6000, 6000], press=[1.0, 10.0,
100.0, 1000.0, 5000.0, 10000.0, 20000.0, 50000.0],
meta_iter_num=4, sub_iteration_num=8,
temps_iterator=None, ensemble='npt', trj_freq=10,
temps_interval=0.1, num_temps=5)

dpngen.tools.auto_gen_param.get_sys_configs(system_list)

dpngen.tools.auto_gen_param.get_system_list(system_dict, map_list=[1, 1, 2, 2, 2, 4, 4, 4],
meta_iter_num=4, sub_iteration_num=8,
map_iterator=None, file_name='POSCAR')

:Exmaple [['000000', '000001'], ['00000[2-9]',], ['00001?', '000020'],]

dpngen.tools.auto_gen_param.scan_files(scan_dir='.', file_name='POSCAR', min_allow_files_num=20)

```

dpngen.tools.collect_data module

```

dpngen.tools.collect_data.collect_data(target_folder, param_file, output, verbose=True)

dpngen.tools.collect_data.file_len(fname)

```

dpngen.tools.relabel module

```

dpngen.tools.relabel.copy_pp_files(tdir, fp_pp_path, fp_pp_files)

dpngen.tools.relabel.create_init_tasks(target_folder, param_file, output, fp_json, verbose=True)

dpngen.tools.relabel.create_tasks(target_folder, param_file, output, fp_json, verbose=True, numb_iter=-1)

dpngen.tools.relabel.get_lmp_info(input_file)

dpngen.tools.relabel.link_pp_files(tdir, fp_pp_path, fp_pp_files)

```

`dpgen.tools.relabel.make_pwscf(tdir, fp_params, mass_map, fp_pp_path, fp_pp_files, user_input)`

`dpgen.tools.relabel.make_siesta(tdir, fp_params, fp_pp_path, fp_pp_files)`

`dpgen.tools.relabel.make_vasp(tdir, fp_params)`

`dpgen.tools.relabel.make_vasp_incar(tdir, fp_incar)`

dpgen.tools.run_report module

`dpgen.tools.run_report.run_report(args)`

dpgen.tools.stat_iter module

`dpgen.tools.stat_iter.stat_iter(target_folder, param_file='param.json', verbose=True, mute=False)`

dpgen.tools.stat_sys module

`dpgen.tools.stat_sys.ascii_hist(count)`

`dpgen.tools.stat_sys.run_report(args)`

`dpgen.tools.stat_sys.stat_sys(target_folder, param_file='param.json', verbose=True, mute=False)`

dpgen.tools.stat_time module

`dpgen.tools.stat_time.stat_time(target_folder, param_file='param.json', verbose=True, mute=False)`

10.1.2 Submodules

10.1.3 dpgen.arginfo module

`dpgen.arginfo.general_mdata_arginfo(name: str, tasks: Tuple[str]) → Argument`

Generate arginfo for general mdata.

Parameters

name

[str] mdata name

tasks

[tuple[str]] tuple of task keys, e.g. (“train”, “model_devi”, “fp”)

Returns

Argument

arginfo

10.1.4 dpgen.main module

`dpgen.main.main()`

`dpgen.main.main_parser()` → `ArgumentParser`

Returns parser for *dpgen* command.

Returns

`argparse.ArgumentParser`

parser for *dpgen* command

10.1.5 dpgen.util module

`dpgen.util.box_center(ch="", fill=' ', sp='|')`

put the string at the center of | |

`dpgen.util.expand_sys_str(root_dir: Union[str, Path])` → `List[str]`

Recursively iterate over directories taking those that contain *type.raw* file.

Parameters

`root_dir`

[`Union[str, Path]`] starting directory

Returns

`List[str]`

list of string pointing to system directories

`dpgen.util.normalize(arginfo: Argument, data: dict, strict_check: bool = True)` → `dict`

Normalize and check input data.

Parameters

`arginfo`

[`dargs.Argument`] argument information

`data`

[`dict`] input data

`strict_check`

[`bool`, `default=True`] strict check data or not

Returns

`dict`

normalized data

`dpgen.util.sepline(ch='-', sp='-', screen=False)`

seperate the output by ‘-’

AUTHORS

- AnguseZhang
- Anopaul
- BaozCWJ
- Cloudac7
- EC2 Default User
- Ericwang6
- Futaki Haduki
- Futaki Hatuki
- Han Wang
- HuangJiameng
- Jinzh Zeng
- Jinzhe Zeng
- Kick-H
- LiangWenshuo1118
- Liu Renxi
- Liu-RX
- LiuGroupHNU
- Manyi Yang
- Pan Xiang
- Pinchen Xie
- Silvia-liu
- TaipingHu
- Tongqi Wen
- TongqiWen
- Waikit Chan
- Wanrun Jiang
- Yingze Wang

- Yixiao Chen
- Yongbin Zhuang
- Yuan Fengbo
- Yuan Fengbo ()
- Yunfan Xu
- Yunpei Liu
- Yuzhi Zhang
- Zhiwei Zhang
- baihuyu12
- cherushui
- cyFortneu
- deepmodeling
- dingzhaohan
- dinngzhaohan
- felix5572
- fqgong
- haidi
- hongriTianqi
- jameswind
- pee8379
- pxxingliang
- robinzhuang
- robinzyb
- root
- shazj99
- tianhongzhen
- tuoping
- unknown
- yuzhi
- zhang yuzhi
- zhangbei07
- zhaohan
- zhengming-HIT
- zhenyu
- ziqi-hu
-

- `genindex`
- `modindex`
- `search`

BIBLIOGRAPHY

- [1] Development of Range-Corrected Deep Learning Potentials for Fast, Accurate Quantum Mechanical/Molecular Mechanical Simulations of Chemical Reactions in Solution, Jinzhe Zeng, Timothy J. Giese, Şölen Ekesan, and Darrin M. York, *Journal of Chemical Theory and Computation* 2021 17 (11), 6993-7009

PYTHON MODULE INDEX

d

dpngen, 219
dpngen.arginfo, 284
dpngen.auto_test, 219
dpngen.auto_test.ABACUS, 231
dpngen.auto_test.calculator, 245
dpngen.auto_test.common_equi, 245
dpngen.auto_test.common_prop, 245
dpngen.auto_test.Elastic, 234
dpngen.auto_test.EOS, 233
dpngen.auto_test.Gamma, 234
dpngen.auto_test.gen_confs, 245
dpngen.auto_test.Interstitial, 236
dpngen.auto_test.Lammps, 236
dpngen.auto_test.lib, 219
dpngen.auto_test.lib.abacus, 223
dpngen.auto_test.lib.BatchJob, 219
dpngen.auto_test.lib.crys, 223
dpngen.auto_test.lib.lammps, 224
dpngen.auto_test.lib.lmp, 225
dpngen.auto_test.lib.mfp_eosfit, 225
dpngen.auto_test.lib.pwscf, 229
dpngen.auto_test.lib.RemoteJob, 220
dpngen.auto_test.lib.siesta, 229
dpngen.auto_test.lib.SlurmJob, 222
dpngen.auto_test.lib.util, 229
dpngen.auto_test.lib.utils, 230
dpngen.auto_test.lib.vasp, 230
dpngen.auto_test.mpdb, 245
dpngen.auto_test.Property, 238
dpngen.auto_test.refine, 246
dpngen.auto_test.reproduce, 246
dpngen.auto_test.run, 246
dpngen.auto_test.Surface, 239
dpngen.auto_test.Task, 240
dpngen.auto_test.Vacancy, 244
dpngen.auto_test.VASP, 242
dpngen.collect, 246
dpngen.collect.collect, 246
dpngen.data, 246
dpngen.data.arginfo, 249
dpngen.data.gen, 249
dpngen.data.reaction, 251
dpngen.data.surf, 251
dpngen.data.tools, 246
dpngen.data.tools.bcc, 246
dpngen.data.tools.cessp2force_lin, 247
dpngen.data.tools.create_random_disturb, 247
dpngen.data.tools.diamond, 247
dpngen.data.tools.fcc, 248
dpngen.data.tools.hcp, 248
dpngen.data.tools.io_lammps, 248
dpngen.data.tools.sc, 249
dpngen.database, 252
dpngen.database.entry, 252
dpngen.database.run, 253
dpngen.database.vasp, 253
dpngen.dispatcher, 255
dpngen.dispatcher.AWS, 255
dpngen.dispatcher.Batch, 256
dpngen.dispatcher.Dispatcher, 257
dpngen.dispatcher.DispatcherList, 259
dpngen.dispatcher.JobStatus, 260
dpngen.dispatcher.LazyLocalContext, 261
dpngen.dispatcher.LocalContext, 262
dpngen.dispatcher.LSF, 260
dpngen.dispatcher.PBS, 263
dpngen.dispatcher.Shell, 266
dpngen.dispatcher.Slurm, 266
dpngen.dispatcher.SSHContext, 264
dpngen.generator, 267
dpngen.generator.arginfo, 271
dpngen.generator.lib, 267
dpngen.generator.lib.abacus_scf, 267
dpngen.generator.lib.cp2k, 267
dpngen.generator.lib.cvasp, 268
dpngen.generator.lib.ele_temp, 268
dpngen.generator.lib.gaussian, 269
dpngen.generator.lib.lammps, 269
dpngen.generator.lib.make_calypso, 269
dpngen.generator.lib.parse_calypso, 269
dpngen.generator.lib.pwmat, 269
dpngen.generator.lib.pwscf, 270
dpngen.generator.lib.run_calypso, 270

- `dpgen.generator.lib.siesta`, 270
- `dpgen.generator.lib.utils`, 271
- `dpgen.generator.lib.vasp`, 271
- `dpgen.generator.run`, 272
- `dpgen.main`, 285
- `dpgen.remote`, 276
 - `dpgen.remote.decide_machine`, 279
 - `dpgen.remote.group_jobs`, 279
 - `dpgen.remote.RemoteJob`, 276
- `dpgen.simplify`, 280
 - `dpgen.simplify.arginfo`, 280
 - `dpgen.simplify.simplify`, 281
- `dpgen.tools`, 281
 - `dpgen.tools.auto_gen_param`, 281
 - `dpgen.tools.collect_data`, 283
 - `dpgen.tools.relabel`, 283
 - `dpgen.tools.run_report`, 284
 - `dpgen.tools.stat_iter`, 284
 - `dpgen.tools.stat_sys`, 284
 - `dpgen.tools.stat_time`, 284
- `dpgen.util`, 285

INDEX

A

ABACUS (class in *dpngen.auto_test.ABACUS*), 231
add_sub_system() (*dp-gen.tools.auto_gen_param.System* method), 282
all_finished() (*dpngen.dispatcher.Dispatcher.Dispatcher* method), 257
analysis() (in module *dp-gen.generator.lib.run_calypso*), 270
api_version:
 init_bulk_mdata/api_version (Argument), 70
 init_reaction_mdata/api_version (Argument), 98
 init_surf_mdata/api_version (Argument), 84
 run_mdata/api_version (Argument), 35
 simplify_mdata/api_version (Argument), 142
apply_type_map() (in module *dp-gen.auto_test.lib.lammps*), 224
as_dict() (*dpngen.database.entry.Entry* method), 252
as_dict() (*dpngen.database.vasp.DPPotcar* method), 253
as_dict() (*dpngen.database.vasp.VaspInput* method), 254
ascii_hist() (in module *dpngen.tools.stat_sys*), 284
ase2lammpsdata() (in module *dp-gen.data.tools.io_lammps*), 248
auto_gen_param() (in module *dp-gen.tools.auto_gen_param*), 282
AWS (class in *dpngen.dispatcher.AWS*), 255
AWS_check_status() (*dpngen.dispatcher.AWS.AWS* class method), 255
aws_submit_jobs() (in module *dp-gen.remote.group_jobs*), 279
awsMachineJob (class in *dpngen.remote.RemoteJob*), 278

B

backward_files (*dpngen.auto_test.Task.Task* property), 241
backward_files() (*dpngen.auto_test.ABACUS.ABACUS* method), 231
backward_files() (*dpngen.auto_test.Lammps.Lammps* method), 237

backward_files() (*dpngen.auto_test.VASP.VASP* method), 242
basic_args() (in module *dpngen.generator.arginfo*), 271
basis_set:
 run_jdata[fp_style=gaussian]/fp_params/basis_set (Argument), 31
 simplify_jdata[gaussian]/fp_params/basis_set (Argument), 142
Batch (class in *dpngen.dispatcher.Batch*), 256
batch_type:
 init_bulk_mdata/fp/machine/batch_type (Argument), 70
 init_bulk_mdata/fp/resources/batch_type (Argument), 78
 init_reaction_mdata/build/machine/batch_type (Argument), 110
 init_reaction_mdata/build/resources/batch_type (Argument), 118
 init_reaction_mdata/fp/machine/batch_type (Argument), 121
 init_reaction_mdata/fp/resources/batch_type (Argument), 129
 init_reaction_mdata/reaxff/machine/batch_type (Argument), 98
 init_reaction_mdata/reaxff/resources/batch_type (Argument), 106
 init_surf_mdata/fp/machine/batch_type (Argument), 84
 init_surf_mdata/fp/resources/batch_type (Argument), 92
 run_mdata/fp/machine/batch_type (Argument), 57
 run_mdata/fp/resources/batch_type (Argument), 65
 run_mdata/model_devi/machine/batch_type (Argument), 46
 run_mdata/model_devi/resources/batch_type (Argument), 54
 run_mdata/train/machine/batch_type (Argument), 35
 run_mdata/train/resources/batch_type (Argument), 42

simplify_mdata/fp/machine/batch_type (Argument), 166
 simplify_mdata/fp/resources/batch_type (Argument), 173
 simplify_mdata/model_devi/machine/batch_type (Argument), 154
 simplify_mdata/model_devi/resources/batch_type (Argument), 162
 simplify_mdata/train/machine/batch_type (Argument), 143
 simplify_mdata/train/resources/batch_type (Argument), 151
 BatchJob (class in *dpgen.auto_test.lib.BatchJob*), 219
 bcc() (in module *dpgen.auto_test.lib.crys*), 223
 birch() (in module *dpgen.auto_test.lib.mfp_eosfit*), 225
 block_call() (*dpgen.auto_test.lib.RemoteJob.RemoteJob* method), 221
 block_call() (*dpgen.dispatcher.LazyLocalContext.LazyLocalContext* method), 261
 block_call() (*dpgen.dispatcher.LocalContext.LocalContext* method), 262
 block_call() (*dpgen.dispatcher.SSHContext.SSHContext* method), 264
 block_call() (*dpgen.remote.RemoteJob.RemoteJob* method), 277
 block_checkcall() (*dp-gen.auto_test.lib.RemoteJob.RemoteJob* method), 221
 block_checkcall() (*dp-gen.dispatcher.LazyLocalContext.LazyLocalContext* method), 261
 block_checkcall() (*dp-gen.dispatcher.LocalContext.LocalContext* method), 262
 block_checkcall() (*dp-gen.dispatcher.SSHContext.SSHContext* method), 264
 block_checkcall() (*dp-gen.remote.RemoteJob.RemoteJob* method), 277
 BM4() (in module *dpgen.auto_test.lib.mfp_eosfit*), 225
 BM5() (in module *dpgen.auto_test.lib.mfp_eosfit*), 225
 box2lmpbox() (in module *dpgen.auto_test.lib.lmp*), 225
 box_center() (in module *dpgen.util*), 285
 build:
 init_reaction_mdata/build (Argument), 109
C
 calc_props_BM4() (in module *dp-gen.auto_test.lib.mfp_eosfit*), 225
 calc_props_LOG4() (in module *dp-gen.auto_test.lib.mfp_eosfit*), 226
 calc_props_mBM4() (in module *dp-gen.auto_test.lib.mfp_eosfit*), 226
 calc_props_mBM4poly() (in module *dp-gen.auto_test.lib.mfp_eosfit*), 226
 calc_props_mBM5poly() (in module *dp-gen.auto_test.lib.mfp_eosfit*), 226
 calc_props_morse() (in module *dp-gen.auto_test.lib.mfp_eosfit*), 226
 calc_props_morse_6p() (in module *dp-gen.auto_test.lib.mfp_eosfit*), 226
 calc_props_SJX_5p() (in module *dp-gen.auto_test.lib.mfp_eosfit*), 226
 calc_props_vinet() (in module *dp-gen.auto_test.lib.mfp_eosfit*), 226
 calc_v0_mBM4poly() (in module *dp-gen.auto_test.lib.mfp_eosfit*), 226
 calc_v0_mBM5poly() (in module *dp-gen.auto_test.lib.mfp_eosfit*), 226
 call() (*dpgen.dispatcher.LazyLocalContext.LazyLocalContext* method), 261
 call() (*dpgen.dispatcher.LocalContext.LocalContext* method), 262
 call() (*dpgen.dispatcher.SSHContext.SSHContext* method), 264
 car2dir() (in module *dpgen.data.tools.io_lammps*), 248
 catch_dispatcher_exception() (*dp-gen.dispatcher.DispatcherList.DispatcherList* method), 259
 centralize_slab() (*dpgen.auto_test.Gamma.Gamma* static method), 235
 charge:
 run_jdata[fp_style=gaussian]/fp_params/charge (Argument), 31
 simplify_jdata[gaussian]/fp_params/charge (Argument), 141
 check_all_dispatchers_finished() (*dp-gen.dispatcher.DispatcherList.DispatcherList* method), 259
 check_all_finished() (*dp-gen.dispatcher.Dispatcher.JobRecord* method), 257
 check_apikey() (in module *dpgen.auto_test.mpdb*), 245
 check_bad_box() (in module *dpgen.generator.run*), 272
 check_cluster() (in module *dpgen.generator.run*), 272
 check_dispatcher_status() (*dp-gen.dispatcher.DispatcherList.DispatcherList* method), 259
 check_file_exists() (*dp-gen.dispatcher.LazyLocalContext.LazyLocalContext* method), 261
 check_file_exists() (*dp-gen.dispatcher.LocalContext.LocalContext* method), 262
 check_file_exists() (*dp-gen.dispatcher.SSHContext.SSHContext* method), 264

`check_finish()` (`dpgen.dispatcher.LazyLocalContext.LazyLocalContext` method), 278
`check_finish()` (`dpgen.dispatcher.LocalContext.LocalContext` method), 262
`check_finish()` (`dpgen.dispatcher.SSHContext.SSHContext` method), 264
`check_finish_tag()` (`dpgen.dispatcher.Batch.Batch` method), 256
`check_finished()` (`dp-gen.dispatcher.Dispatcher.JobRecord` method), 257
`check_finished()` (in module `dp-gen.auto_test.lib.abacus`), 223
`check_finished()` (in module `dp-gen.auto_test.lib.lammps`), 224
`check_finished()` (in module `dp-gen.auto_test.lib.vasp`), 230
`check_finished_new()` (in module `dp-gen.auto_test.lib.lammps`), 224
`check_limit()` (`dpgen.remote.RemoteJob.LSFJob` method), 277
`check_nfail()` (`dpgen.dispatcher.Dispatcher.JobRecord` method), 257
`check_running()` (`dpgen.dispatcher.Shell.Shell` method), 266
`check_status()` (`dpgen.auto_test.lib.BatchJob.BatchJob` method), 219
`check_status()` (`dpgen.auto_test.lib.RemoteJob.CloudMachineJob` method), 220
`check_status()` (`dpgen.auto_test.lib.RemoteJob.PBSJob` method), 221
`check_status()` (`dpgen.auto_test.lib.RemoteJob.SlurmJob` method), 222
`check_status()` (`dpgen.auto_test.lib.SlurmJob.SlurmJob` method), 222
`check_status()` (`dpgen.dispatcher.AWS.AWS` method), 255
`check_status()` (`dpgen.dispatcher.Batch.Batch` method), 256
`check_status()` (`dpgen.dispatcher.LSF.LSF` method), 260
`check_status()` (`dpgen.dispatcher.PBS.PBS` method), 264
`check_status()` (`dpgen.dispatcher.Shell.Shell` method), 266
`check_status()` (`dpgen.dispatcher.Slurm.Slurm` method), 267
`check_status()` (`dpgen.remote.RemoteJob.CloudMachineJob` method), 276
`check_status()` (`dpgen.remote.RemoteJob.LSFJob` method), 277
`check_status()` (`dpgen.remote.RemoteJob.PBSJob` method), 277
`check_status()` (`dpgen.remote.RemoteJob.SlurmJob` method), 277
`check_stru_fixed()` (in module `dp-gen.auto_test.lib.abacus`), 223
`check_submitted()` (`dp-gen.dispatcher.Dispatcher.JobRecord` method), 257
`class_cell_type()` (in module `dpgen.data.gen`), 249
`class_cell_type()` (in module `dpgen.data.surf`), 251
`clean()` (`dpgen.auto_test.lib.RemoteJob.RemoteJob` method), 221
`clean()` (`dpgen.dispatcher.DispatcherList.DispatcherList` method), 259
`clean()` (`dpgen.dispatcher.LazyLocalContext.LazyLocalContext` method), 261
`clean()` (`dpgen.dispatcher.LocalContext.LocalContext` method), 262
`clean()` (`dpgen.dispatcher.SSHContext.SSHContext` method), 265
`clean()` (`dpgen.remote.RemoteJob.RemoteJob` method), 278
`clean_asynchronously:`
`init_bulk_mdata/fp/machine/clean_asynchronously` (Argument), 71
`init_reaction_mdata/build/machine/clean_asynchronously` (Argument), 110
`init_reaction_mdata/fp/machine/clean_asynchronously` (Argument), 122
`init_reaction_mdata/reaxff/machine/clean_asynchronously` (Argument), 99
`init_surf_mdata/fp/machine/clean_asynchronously` (Argument), 85
`run_mdata/fp/machine/clean_asynchronously` (Argument), 58
`run_mdata/model_devi/machine/clean_asynchronously` (Argument), 46
`run_mdata/train/machine/clean_asynchronously` (Argument), 36
`simplify_mdata/fp/machine/clean_asynchronously` (Argument), 166
`simplify_mdata/model_devi/machine/clean_asynchronously` (Argument), 155
`simplify_mdata/train/machine/clean_asynchronously` (Argument), 143
`close()` (`dpgen.auto_test.lib.RemoteJob.SSHSession` method), 222
`close()` (`dpgen.dispatcher.SSHContext.SSHContext` method), 265
`close()` (`dpgen.dispatcher.SSHContext.SSHSession` method), 265
`close()` (`dpgen.remote.RemoteJob.SSHSession` method), 278
`CloudMachineJob` (class in `dp-gen.auto_test.lib.RemoteJob`), 220
`CloudMachineJob` (class in `dpgen.remote.RemoteJob`),

276

`cluster_cutoff:`
 `run_jdata[fp_style=gaussian]/cluster_cutoff` (Argument), 30
 `run_jdata[fp_style=siesta]/cluster_cutoff` (Argument), 31
 `simplify_jdata[gaussian]/cluster_cutoff` (Argument), 140

`cluster_cutoff_hard:`
 `run_jdata[fp_style=gaussian]/cluster_cutoff_hard` (Argument), 30
 `simplify_jdata[gaussian]/cluster_cutoff_hard` (Argument), 141

`cluster_minify:`
 `run_jdata[fp_style=gaussian]/cluster_minify` (Argument), 30
 `simplify_jdata[gaussian]/cluster_minify` (Argument), 141

`cmd_append_log()` (in module `dp-gen.auto_test.lib.utils`), 230

`cmd_append_log()` (in module `dp-gen.generator.lib.utils`), 271

`coll_abacus_md()` (in module `dpgen.data.gen`), 249

`coll_vasp_md()` (in module `dpgen.data.gen`), 249

`collect_data()` (in module `dpgen.collect.collect`), 246

`collect_data()` (in module `dpgen.tools.collect_data`), 283

`collect_task()` (in module `dpgen.auto_test.lib.util`), 229

`command:`
 `init_bulk_mdata/fp/command` (Argument), 70
 `init_reaction_mdata/build/command` (Argument), 110
 `init_reaction_mdata/fp/command` (Argument), 121
 `init_reaction_mdata/reaxff/command` (Argument), 98
 `init_surf_mdata/fp/command` (Argument), 84
 `run_mdata/fp/command` (Argument), 57
 `run_mdata/model_devi/command` (Argument), 45
 `run_mdata/train/command` (Argument), 35
 `simplify_mdata/fp/command` (Argument), 165
 `simplify_mdata/model_devi/command` (Argument), 154
 `simplify_mdata/train/command` (Argument), 143

`completing` (`dpgen.dispatcher.JobStatus.JobStatus` attribute), 260

`compute()` (`dpgen.auto_test.ABACUS.ABACUS` method), 231

`compute()` (`dpgen.auto_test.Lammps.Lammps` method), 237

`compute()` (`dpgen.auto_test.Property.Property` method), 239

`compute()` (`dpgen.auto_test.Task.Task` method), 241

`compute()` (`dpgen.auto_test.VASP.VASP` method), 243

`context_type:`
 `init_bulk_mdata/fp/machine/context_type` (Argument), 71
 `init_reaction_mdata/build/machine/context_type` (Argument), 110
 `init_reaction_mdata/fp/machine/context_type` (Argument), 122
 `init_reaction_mdata/reaxff/machine/context_type` (Argument), 99
 `init_surf_mdata/fp/machine/context_type` (Argument), 85
 `run_mdata/fp/machine/context_type` (Argument), 58
 `run_mdata/model_devi/machine/context_type` (Argument), 46
 `run_mdata/train/machine/context_type` (Argument), 36
 `simplify_mdata/fp/machine/context_type` (Argument), 166
 `simplify_mdata/model_devi/machine/context_type` (Argument), 155
 `simplify_mdata/train/machine/context_type` (Argument), 144

`control:`
 `init_reaction_jdata/reaxff/control` (Argument), 96

`convert_cell()` (in module `dp-gen.data.tools.io_lammps`), 248

`convert_data()` (in module `dpgen.data.reaction`), 251

`convert_forces()` (in module `dp-gen.data.tools.io_lammps`), 248

`convert_mdata()` (in module `dp-gen.remote.decide_machine`), 279

`convert_positions()` (in module `dp-gen.data.tools.io_lammps`), 248

`convert_stress()` (in module `dp-gen.data.tools.io_lammps`), 248

`copy_file_list()` (in module `dp-gen.auto_test.lib.utils`), 230

`copy_file_list()` (in module `dp-gen.generator.lib.utils`), 271

`copy_model()` (in module `dpgen.generator.run`), 272

`copy_pp_files()` (in module `dpgen.tools.relabel`), 283

`cpu_per_node:`
 `init_bulk_mdata/fp/resources/cpu_per_node` (Argument), 76
 `init_reaction_mdata/build/resources/cpu_per_node` (Argument), 115
 `init_reaction_mdata/fp/resources/cpu_per_node` (Argument), 127
 `init_reaction_mdata/reaxff/resources/cpu_per_node` (Argument), 104

`init_surf_mdata/fp/resources/cpu_per_node`
 (*Argument*), **90**
`run_mdata/fp/resources/cpu_per_node`
 (*Argument*), **62**
`run_mdata/model_devi/resources/cpu_per_node`
 (*Argument*), **51**
`run_mdata/train/resources/cpu_per_node`
 (*Argument*), **40**
`simplify_mdata/fp/resources/cpu_per_node`
 (*Argument*), **171**
`simplify_mdata/model_devi/resources/cpu_per_node`
 (*Argument*), **160**
`simplify_mdata/train/resources/cpu_per_node`
 (*Argument*), **148**
`create()` (*dpngen.dispatcher.DispatcherList.DispatcherList*
method), **259**
`create_disturbs_abacus_dev()` (*in module dp-*
gen.data.tools.create_random_disturb), **247**
`create_disturbs_ase()` (*in module dp-*
gen.data.tools.create_random_disturb), **247**
`create_disturbs_ase_dev()` (*in module dp-*
gen.data.tools.create_random_disturb), **247**
`create_disturbs_atomsk()` (*in module dp-*
gen.data.tools.create_random_disturb), **247**
`create_init_tasks()` (*in module dpngen.tools.relabel*),
283
`create_path()` (*in module dpngen.auto_test.lib.utils*),
230
`create_path()` (*in module dpngen.data.gen*), **249**
`create_path()` (*in module dpngen.data.surf*), **251**
`create_path()` (*in module dpngen.generator.lib.utils*),
271
`create_random_alloys()` (*in module dp-*
gen.data.tools.create_random_disturb), **247**
`create_tasks()` (*in module dpngen.tools.relabel*), **283**
`current_num_of_itearation` (*dp-*
gen.tools.auto_gen_param.Iteration attribute),
282
`current_num_of_sub_itearation` (*dp-*
gen.tools.auto_gen_param.Iteration attribute),
282
`current_num_of_sub_systems` (*dp-*
gen.tools.auto_gen_param.System attribute),
282
`current_num_of_system` (*dp-*
gen.tools.auto_gen_param.System attribute),
282
`custom_flags:`
`init_bulk_mdata/fp/resources/custom_flags`
 (*Argument*), **76**
`init_reaction_mdata/build/resources/custom_flags`
 (*Argument*), **116**
`init_reaction_mdata/fp/resources/custom_flags`
 (*Argument*), **127**
`init_reaction_mdata/reaxff/resources/custom_flags`
 (*Argument*), **104**
`init_surf_mdata/fp/resources/custom_flags`
 (*Argument*), **90**
`run_mdata/fp/resources/custom_flags`
 (*Argument*), **63**
`run_mdata/model_devi/resources/custom_flags`
 (*Argument*), **52**
`run_mdata/train/resources/custom_flags`
 (*Argument*), **41**
`simplify_mdata/fp/resources/custom_flags`
 (*Argument*), **171**
`simplify_mdata/model_devi/resources/custom_flags`
 (*Argument*), **160**
`simplify_mdata/train/resources/custom_flags`
 (*Argument*), **149**
`custom_gpu_line:`
`init_bulk_mdata/fp/resources[LSF]/kwargs/custom_gpu_li`
 (*Argument*), **80**
`init_bulk_mdata/fp/resources[SlurmJobArray]/kwargs/cus`
 (*Argument*), **79**
`init_bulk_mdata/fp/resources[Slurm]/kwargs/custom_gpu_`
 (*Argument*), **79**
`init_reaction_mdata/build/resources[LSF]/kwargs/custom`
 (*Argument*), **120**
`init_reaction_mdata/build/resources[SlurmJobArray]/kwa`
 (*Argument*), **119**
`init_reaction_mdata/build/resources[Slurm]/kwargs/cust`
 (*Argument*), **118**
`init_reaction_mdata/fp/resources[LSF]/kwargs/custom_gp`
 (*Argument*), **132**
`init_reaction_mdata/fp/resources[SlurmJobArray]/kwargs`
 (*Argument*), **130**
`init_reaction_mdata/fp/resources[Slurm]/kwargs/custom_`
 (*Argument*), **130**
`init_reaction_mdata/reaxff/resources[LSF]/kwargs/custo`
 (*Argument*), **109**
`init_reaction_mdata/reaxff/resources[SlurmJobArray]/kw`
 (*Argument*), **107**
`init_reaction_mdata/reaxff/resources[Slurm]/kwargs/cus`
 (*Argument*), **107**
`init_surf_mdata/fp/resources[LSF]/kwargs/custom_gpu_li`
 (*Argument*), **94**
`init_surf_mdata/fp/resources[SlurmJobArray]/kwargs/cus`
 (*Argument*), **93**
`init_surf_mdata/fp/resources[Slurm]/kwargs/custom_gpu_`
 (*Argument*), **92**
`run_mdata/fp/resources[LSF]/kwargs/custom_gpu_line`
 (*Argument*), **67**
`run_mdata/fp/resources[SlurmJobArray]/kwargs/custom_gp`
 (*Argument*), **66**
`run_mdata/fp/resources[Slurm]/kwargs/custom_gpu_line`
 (*Argument*), **65**
`run_mdata/model_devi/resources[LSF]/kwargs/custom_gpu_`

(Argument), 56
 run_mdata/model_devi/resources[SlurmJobArray]/kwargs/custom_gpu_line
 (Argument), 55
 run_mdata/model_devi/resources[Slurm]/kwargs/custom_gpu_line
 (Argument), 54
 run_mdata/deepmd_version (Argument), 35
 run_mdata/train/resources[LSF]/kwargs/custom_gpu_line
 (Argument), 45
 run_mdata/train/resources[SlurmJobArray]/kwargs/custom_gpu_line
 (Argument), 44
 run_mdata/train/resources[Slurm]/kwargs/cuda_bulk_purification
 (Argument), 43
 simplify_mdata/fp/resources[LSF]/kwargs/cuda_bulk_purification
 (Argument), 175
 simplify_mdata/fp/resources[SlurmJobArray]/kwargs/cuda_bulk_purification
 (Argument), 174
 simplify_mdata/fp/resources[Slurm]/kwargs/default_gpu_line
 (Argument), 174
 simplify_mdata/model_devi/resources[LSF]/kwargs/cuda_bulk_purification
 (Argument), 164
 simplify_mdata/model_devi/resources[SlurmJobArray]/kwargs/cuda_bulk_purification
 (Argument), 163
 simplify_mdata/model_devi/resources[Slurm]/kwargs/cuda_bulk_purification
 (Argument), 163
 simplify_mdata/train/resources[LSF]/kwargs/default_training_param
 (Argument), 153
 simplify_mdata/train/resources[SlurmJobArray]/kwargs/custom_gpu_line
 (Argument), 152
 simplify_mdata/train/resources[Slurm]/kwargs/custom_gpu_line
 (Argument), 151
 cutoff:
 init_reaction_jdata/cutoff (Argument), 97
 run_jdata[model_devi_engine=amber]/cutoff
 (Argument), 27
 cvasp:
 run_jdata[fp_style=vasp]/cvasp (Argument), 29
 simplify_jdata[vasp]/cvasp (Argument), 140
 cvt_1frame() (in module dpgen.generator.lib.pwscf), 270
 cvt_lammps_conf() (in module dp-gen.auto_test.lib.lammps), 224
D
 data:
 init_reaction_jdata/reaxff/data (Argument), 96
 data_args() (in module dpgen.generator.arginfo), 271
 dataset_size:
 init_reaction_jdata/dataset_size (Argument), 97
 db_run() (in module dpgen.database.run), 253
 deepmd_version:
 init_bulk_mdata/deepmd_version (Argument), 70
 init_reaction_mdata/deepmd_version (Argument), 98
 init_surf_mdata/deepmd_version (Argument), 81
 run_mdata/deepmd_version (Argument), 35
 simplify_mdata/deepmd_version (Argument), 142
 default_training_param() (in module dp-gen.tools.auto_gen_param), 283
 cuda_bulk_purification() (dpgen.dispatcher.AWS.AWS method), 255
 cuda_bulk_purification() (dpgen.dispatcher.Batch.Batch method), 256
 cuda_bulk_purification() (dpgen.dispatcher.LSF.LSF method), 261
 default_gpu_line() (dpgen.dispatcher.PBS.PBS method), 264
 cuda_bulk_purification() (dpgen.dispatcher.Shell.Shell method), 266
 cuda_bulk_purification() (dpgen.dispatcher.Slurm.Slurm method), 267
 default_training_param() (in module dp-gen.tools.auto_gen_param), 283
 default_training_param:
 run_jdata/default_training_param (Argument), 138
 simplify_jdata/default_training_param
 (Argument), 138
 delete() (dpgen.dispatcher.DispatcherList.DispatcherList method), 259
 delete() (dpgen.remote.group_jobs.PMap method), 279
 detailed_report_make_fp:
 run_jdata/detailed_report_make_fp (Argument), 21
 detect_batch_size() (in module dp-gen.generator.run), 273
 detect_multiplicity() (in module dp-gen.generator.lib.gaussian), 269
 dhcp() (in module dpgen.auto_test.lib.crys), 223
 diamond() (in module dpgen.auto_test.lib.crys), 223
 dir2car() (in module dpgen.data.tools.io_lammps), 248
 disang:
 run_jdata[model_devi_engine=amber]/disang
 (Argument), 28
 disang_prefix:
 run_jdata[model_devi_engine=amber]/disang_prefix
 (Argument), 28
 Dispatcher (class in dpgen.dispatcher.Dispatcher), 257
 DispatcherList (class in dp-gen.dispatcher.DispatcherList), 259
 do_submit() (dpgen.dispatcher.AWS.AWS method), 255
 do_submit() (dpgen.dispatcher.Batch.Batch method), 256
 do_submit() (dpgen.dispatcher.LSF.LSF method), 261

`do_submit()` (*dpgen.dispatcher.PBS.PBS method*), 264
`do_submit()` (*dpgen.dispatcher.Shell.Shell method*), 266
`do_submit()` (*dpgen.dispatcher.Slurm.Slurm method*), 267
`download()` (*dpgen.auto_test.lib.RemoteJob.RemoteJob method*), 221
`download()` (*dpgen.dispatcher.LazyLocalContext.LazyLocalContext method*), 261
`download()` (*dpgen.dispatcher.LocalContext.LocalContext method*), 262
`download()` (*dpgen.dispatcher.SSHContext.SSHContext method*), 265
`download()` (*dpgen.remote.RemoteJob.awsMachineJob method*), 279
`download()` (*dpgen.remote.RemoteJob.RemoteJob method*), 278
`dp_compress:`
 `run_jdata/dp_compress (Argument)`, 20
 `simplify_jdata/dp_compress (Argument)`, 138
`dpngen`
 module, 219
`dpngen.arginfo`
 module, 284
`dpngen.auto_test`
 module, 219
`dpngen.auto_test.ABACUS`
 module, 231
`dpngen.auto_test.calculator`
 module, 245
`dpngen.auto_test.common_equi`
 module, 245
`dpngen.auto_test.common_prop`
 module, 245
`dpngen.auto_test.Elastic`
 module, 234
`dpngen.auto_test.EOS`
 module, 233
`dpngen.auto_test.Gamma`
 module, 234
`dpngen.auto_test.gen_confs`
 module, 245
`dpngen.auto_test.Interstitial`
 module, 236
`dpngen.auto_test.Lammps`
 module, 236
`dpngen.auto_test.lib`
 module, 219
`dpngen.auto_test.lib.abacus`
 module, 223
`dpngen.auto_test.lib.BatchJob`
 module, 219
`dpngen.auto_test.lib.crys`
 module, 223
`dpngen.auto_test.lib.lammps`
 module, 224
`dpngen.auto_test.lib.lmp`
 module, 225
`dpngen.auto_test.lib.mfp_eosfit`
 module, 225
`dpngen.auto_test.lib.pwscf`
 module, 229
`dpngen.auto_test.lib.RemoteJob`
 module, 220
`dpngen.auto_test.lib.siesta`
 module, 229
`dpngen.auto_test.lib.SlurmJob`
 module, 222
`dpngen.auto_test.lib.util`
 module, 229
`dpngen.auto_test.lib.utils`
 module, 230
`dpngen.auto_test.lib.vasp`
 module, 230
`dpngen.auto_test.mpdb`
 module, 245
`dpngen.auto_test.Property`
 module, 238
`dpngen.auto_test.refine`
 module, 246
`dpngen.auto_test.reproduce`
 module, 246
`dpngen.auto_test.run`
 module, 246
`dpngen.auto_test.Surface`
 module, 239
`dpngen.auto_test.Task`
 module, 240
`dpngen.auto_test.Vacancy`
 module, 244
`dpngen.auto_test.VASP`
 module, 242
`dpngen.collect`
 module, 246
`dpngen.collect.collect`
 module, 246
`dpngen.data`
 module, 246
`dpngen.data.arginfo`
 module, 249
`dpngen.data.gen`
 module, 249
`dpngen.data.reaction`
 module, 251
`dpngen.data.surf`
 module, 251
`dpngen.data.tools`
 module, 246
`dpngen.data.tools.bcc`

- module, 246
- dpngen.data.tools.cessp2force_lin
 - module, 247
- dpngen.data.tools.create_random_disturb
 - module, 247
- dpngen.data.tools.diamond
 - module, 247
- dpngen.data.tools.fcc
 - module, 248
- dpngen.data.tools.hcp
 - module, 248
- dpngen.data.tools.io_lammps
 - module, 248
- dpngen.data.tools.sc
 - module, 249
- dpngen.database
 - module, 252
- dpngen.database.entry
 - module, 252
- dpngen.database.run
 - module, 253
- dpngen.database.vasp
 - module, 253
- dpngen.dispatcher
 - module, 255
- dpngen.dispatcher.AWS
 - module, 255
- dpngen.dispatcher.Batch
 - module, 256
- dpngen.dispatcher.Dispatcher
 - module, 257
- dpngen.dispatcher.DispatcherList
 - module, 259
- dpngen.dispatcher.JobStatus
 - module, 260
- dpngen.dispatcher.LazyLocalContext
 - module, 261
- dpngen.dispatcher.LocalContext
 - module, 262
- dpngen.dispatcher.LSF
 - module, 260
- dpngen.dispatcher.PBS
 - module, 263
- dpngen.dispatcher.Shell
 - module, 266
- dpngen.dispatcher.Slurm
 - module, 266
- dpngen.dispatcher.SSHContext
 - module, 264
- dpngen.generator
 - module, 267
- dpngen.generator.arginfo
 - module, 271
- dpngen.generator.lib

- module, 267
- dpngen.generator.lib.abacus_scf
 - module, 267
- dpngen.generator.lib.cp2k
 - module, 267
- dpngen.generator.lib.cvasp
 - module, 268
- dpngen.generator.lib.ele_temp
 - module, 268
- dpngen.generator.lib.gaussian
 - module, 269
- dpngen.generator.lib.lammps
 - module, 269
- dpngen.generator.lib.make_calypso
 - module, 269
- dpngen.generator.lib.parse_calypso
 - module, 269
- dpngen.generator.lib.pwmat
 - module, 269
- dpngen.generator.lib.pwscf
 - module, 270
- dpngen.generator.lib.run_calypso
 - module, 270
- dpngen.generator.lib.siesta
 - module, 270
- dpngen.generator.lib.utils
 - module, 271
- dpngen.generator.lib.vasp
 - module, 271
- dpngen.generator.run
 - module, 272
- dpngen.main
 - module, 285
- dpngen.remote
 - module, 276
- dpngen.remote.decide_machine
 - module, 279
- dpngen.remote.group_jobs
 - module, 279
- dpngen.remote.RemoteJob
 - module, 276
- dpngen.simplify
 - module, 280
- dpngen.simplify.arginfo
 - module, 280
- dpngen.simplify.simplify
 - module, 281
- dpngen.tools
 - module, 281
- dpngen.tools.auto_gen_param
 - module, 281
- dpngen.tools.collect_data
 - module, 283
- dpngen.tools.relabel

module, 283
 dpngen.tools.run_report
 module, 284
 dpngen.tools.stat_iter
 module, 284
 dpngen.tools.stat_sys
 module, 284
 dpngen.tools.stat_time
 module, 284
 dpngen.util
 module, 285
 DPPotcar (class in dpngen.database.vasp), 253
 dt:
 init_reaction_jdata/reaxff/dt (Argument), 97
 dump() (dpngen.dispatcher.Dispatcher.JobRecord method), 257
 dump() (dpngen.remote.group_jobs.PMap method), 279
 dump_freq:
 init_reaction_jdata/reaxff/dump_freq (Argument), 97
 dump_to_deepmd_raw() (in module dpngen.generator.run), 273
E
 ecut:
 run_jdata[fp_style=siesta]/fp_params/ecut ensemble: (Argument), 32
 ediff:
 run_jdata[fp_style=siesta]/fp_params/ediff ensure_alive() (dpngen.dispatcher.SSHContext.SSHSession method), 265
 Elastic (class in dpngen.auto_test.Elastic), 234
 element_list() (in module dpngen.auto_test.lib.lammps), 224
 email:
 init_bulk_mdata/fp/machine[DpCloudServerContext]/remote_profile/email (Argument), 74
 init_bulk_mdata/fp/machine[LebesgueContext]/remote_profile/email (Argument), 72
 init_reaction_mdata/build/machine[DpCloudServerContext]/remote_profile/email (Argument), 114
 init_reaction_mdata/build/machine[LebesgueContext]/remote_profile/email (Argument), 111
 init_reaction_mdata/fp/machine[DpCloudServerContext]/remote_profile/email (Argument), 125
 init_reaction_mdata/fp/machine[LebesgueContext]/remote_profile/email (Argument), 122
 init_reaction_mdata/reaxff/machine[DpCloudServerContext]/remote_profile/email (Argument), 102
 init_reaction_mdata/reaxff/machine[LebesgueContext]/remote_profile/email (Argument), 99
 init_surf_mdata/fp/machine[DpCloudServerContext]/remote_profile/email (Argument), 88
 init_surf_mdata/fp/machine[LebesgueContext]/remote_profile/email (Argument), 85
 run_mdata/fp/machine[DpCloudServerContext]/remote_profile/email (Argument), 61
 run_mdata/fp/machine[LebesgueContext]/remote_profile/email (Argument), 58
 run_mdata/model_devi/machine[DpCloudServerContext]/remote_profile/email (Argument), 50
 run_mdata/model_devi/machine[LebesgueContext]/remote_profile/email (Argument), 47
 run_mdata/train/machine[DpCloudServerContext]/remote_profile/email (Argument), 39
 run_mdata/train/machine[LebesgueContext]/remote_profile/email (Argument), 36
 simplify_mdata/fp/machine[DpCloudServerContext]/remote_profile/email (Argument), 169
 simplify_mdata/fp/machine[LebesgueContext]/remote_profile/email (Argument), 167
 simplify_mdata/model_devi/machine[DpCloudServerContext]/remote_profile/email (Argument), 158
 simplify_mdata/model_devi/machine[LebesgueContext]/remote_profile/email (Argument), 155
 simplify_mdata/train/machine[DpCloudServerContext]/remote_profile/email (Argument), 147
 simplify_mdata/train/machine[LebesgueContext]/remote_profile/email (Argument), 144
 run_jdata[model_devi_engine=lammps]/model_devi_jobs/ensure_alive() (Argument), 23
 Entity (class in dpngen.dispatcher.DispatcherList), 260
 Entry (class in dpngen.database.entry), 252
 envs:
 init_bulk_mdata/fp/resources/envs (Argument), 78
 init_reaction_mdata/build/resources/envs (Argument), 117
 init_reaction_mdata/fp/resources/envs (Argument), 129
 init_reaction_mdata/reaxff/resources/envs (Argument), 106
 init_surf_mdata/fp/resources/envs (Argument), 92
 run_mdata/fp/resources/envs (Argument), 64
 run_mdata/model_devi/resources/envs (Argument), 53
 run_mdata/train/resources/envs (Argument), 42
 simplify_mdata/fp/resources/envs (Argument), 173
 simplify_mdata/model_devi/resources/envs (Argument), 162
 simplify_mdata/train/resources/envs (Argument), 144

(Argument), 150
 EOS (class in *dpgen.auto_test.EOS*), 233
 epsilon:
 run_jdata[model_devi_engine=lammps]/epsilon
 (Argument), 26
 epsilon_v:
 run_jdata[model_devi_engine=lammps]/epsilon_v
 (Argument), 26
 exception_handling() (dp-
 gen.dispatcher.DispatcherList.DispatcherList
 method), 259
 exec_command() (dpgen.dispatcher.SSHContext.SSHSession
 method), 265
 expand_idx() (in module *dpgen.generator.run*), 273
 expand_matrix_values() (in module *dp-
 gen.generator.run*), 273
 expand_sys_str() (in module *dpgen.util*), 285
 ext_splint() (in module *dp-
 gen.auto_test.lib.mfp_eosfit*), 226
 ext_vec() (in module *dpgen.auto_test.lib.mfp_eosfit*),
 226
 ext_velp() (in module *dpgen.auto_test.lib.mfp_eosfit*),
 226
 external_input_path:
 run_jdata[fp_style=cp2k]/external_input_path
 (Argument), 33

F

fcc() (in module *dpgen.auto_test.lib.crys*), 223
 fcc1() (in module *dpgen.auto_test.lib.crys*), 223
 ff:
 init_reaction_jdata/reaxff/ff (Argument),
 96
 file_len() (in module *dpgen.tools.collect_data*), 283
 final_stru() (in module *dpgen.auto_test.lib.abacus*),
 223
 find_only_one_key() (in module *dp-
 gen.generator.run*), 273
 finished (dpgen.auto_test.lib.BatchJob.JobStatus
 attribute), 220
 finished (dpgen.auto_test.lib.RemoteJob.JobStatus
 attribute), 220
 finished (dpgen.dispatcher.JobStatus.JobStatus
 attribute), 260
 finished (dpgen.remote.RemoteJob.JobStatus
 attribute), 276
 forward_common_files (dpgen.auto_test.Task.Task
 property), 241
 forward_common_files() (dp-
 gen.auto_test.ABACUS.ABACUS method),
 232
 forward_common_files() (dp-
 gen.auto_test.Lammps.Lammps method),
 237
 forward_common_files() (dp-
 gen.auto_test.VASP.VASP method), 243
 forward_files (dpgen.auto_test.Task.Task
 property), 241
 forward_files() (dpgen.auto_test.ABACUS.ABACUS
 method), 232
 forward_files() (dpgen.auto_test.Lammps.Lammps
 method), 237
 forward_files() (dpgen.auto_test.VASP.VASP
 method), 243
 fp:
 init_bulk_mdata/fp (Argument), 70
 init_reaction_mdata/fp (Argument), 121
 init_surf_mdata/fp (Argument), 84
 run_mdata/fp (Argument), 57
 simplify_mdata/fp (Argument), 165
 fp_accurate_soft_threshold:
 run_jdata/fp_accurate_soft_threshold (Ar-
 gument), 21
 simplify_jdata/fp_accurate_soft_threshold
 (Argument), 139
 fp_accurate_threshold:
 run_jdata/fp_accurate_threshold (Argu-
 ment), 21
 simplify_jdata/fp_accurate_threshold (Ar-
 gument), 139
 fp_aniso_kspacing:
 run_jdata[fp_style=vasp]/fp_aniso_kspacing
 (Argument), 29
 simplify_jdata[vasp]/fp_aniso_kspacing
 (Argument), 140
 fp_args() (in module *dpgen.generator.arginfo*), 271
 fp_args() (in module *dpgen.simplify.arginfo*), 280
 fp_cluster_vacuum:
 run_jdata/fp_cluster_vacuum (Argument), 21
 fp_dpks_descriptor:
 run_jdata[fp_style=abacus]/fp_dpks_descriptor
 (Argument), 33
 fp_incar:
 run_jdata[fp_style=abacus]/fp_incar
 (Argument), 33
 run_jdata[fp_style=vasp]/fp_incar (Argu-
 ment), 29
 simplify_jdata[vasp]/fp_incar (Argument),
 140
 fp_kpt_file:
 run_jdata[fp_style=abacus]/fp_kpt_file
 (Argument), 33
 fp_orb_files:
 run_jdata[fp_style=abacus]/fp_orb_files
 (Argument), 33
 fp_params:
 run_jdata[fp_style=amber/diff]/fp_params
 (Argument), 34

run_jdata[fp_style=gaussian]/fp_params
 (Argument), 30
 run_jdata[fp_style=siesta]/fp_params (Argument), 32
 simplify_jdata[gaussian]/fp_params (Argument), 141
 fp_pp_files:
 run_jdata[fp_style=abacus]/fp_pp_files
 (Argument), 33
 run_jdata[fp_style=siesta]/fp_pp_files
 (Argument), 32
 run_jdata[fp_style=vasp]/fp_pp_files (Argument), 29
 simplify_jdata[vasp]/fp_pp_files (Argument), 140
 fp_pp_path:
 run_jdata[fp_style=abacus]/fp_pp_path
 (Argument), 33
 run_jdata[fp_style=siesta]/fp_pp_path
 (Argument), 32
 run_jdata[fp_style=vasp]/fp_pp_path
 (Argument), 29
 simplify_jdata[vasp]/fp_pp_path (Argument), 139
 fp_skip_bad_box:
 run_jdata[fp_style=vasp]/fp_skip_bad_box
 (Argument), 29
 simplify_jdata[vasp]/fp_skip_bad_box (Argument), 140
 fp_style:
 run_jdata/fp_style (Argument), 29
 simplify_jdata/fp_style (Argument), 139
 fp_style_abacus_args() (in module dp-gen.generator.arginfo), 271
 fp_style_amber_diff_args() (in module dp-gen.generator.arginfo), 271
 fp_style_cp2k_args() (in module dp-gen.generator.arginfo), 272
 fp_style_gaussian_args() (in module dp-gen.generator.arginfo), 272
 fp_style_siesta_args() (in module dp-gen.generator.arginfo), 272
 fp_style_variant_type_args() (in module dp-gen.generator.arginfo), 272
 fp_style_variant_type_args() (in module dp-gen.simplify.arginfo), 280
 fp_style_vasp_args() (in module dp-gen.generator.arginfo), 272
 fp_task_max:
 run_jdata/fp_task_max (Argument), 21
 simplify_jdata/fp_task_max (Argument), 139
 fp_task_min:
 run_jdata/fp_task_min (Argument), 21
 simplify_jdata/fp_task_min (Argument), 139

fragment_guesses:
 run_jdata[fp_style=gaussian]/fp_params/fragment_guesses
 (Argument), 31
 simplify_jdata[gaussian]/fp_params/fragment_guesses
 (Argument), 142
 from_dict() (dpgen.database.entry.Entry class method), 252
 from_dict() (dpgen.database.vasp.DPPotcar class method), 253
 from_dict() (dpgen.database.vasp.VaspInput class method), 254
 from_directory() (dpgen.database.vasp.VaspInput static method), 254
 from_file() (dpgen.database.vasp.DPPotcar class method), 253
 from_system_data() (in module dp-gen.auto_test.lib.lmp), 225

G

Gamma (class in dpgen.auto_test.Gamma), 234
 gen_alloy() (in module dpgen.auto_test.gen_confs), 245
 gen_box() (in module dpgen.data.tools.bcc), 246
 gen_box() (in module dpgen.data.tools.diamond), 247
 gen_box() (in module dpgen.data.tools.fcc), 248
 gen_box() (in module dpgen.data.tools.hcp), 248
 gen_box() (in module dpgen.data.tools.sc), 249
 gen_collect() (in module dpgen.collect.collect), 246
 gen_ele_std() (in module dpgen.auto_test.gen_confs), 245
 gen_element() (in module dpgen.auto_test.gen_confs), 245
 gen_element_std() (in module dp-gen.auto_test.gen_confs), 245
 gen_init_bulk() (in module dpgen.data.gen), 249
 gen_init_reaction() (in module dp-gen.data.reaction), 251
 gen_init_surf() (in module dpgen.data.surf), 251
 gen_main() (in module dp-gen.generator.lib.run_calypso), 270
 gen_random_disturb() (in module dp-gen.data.tools.create_random_disturb), 247
 gen_random_emat() (in module dp-gen.data.tools.create_random_disturb), 247
 gen_run() (in module dpgen.generator.run), 273
 gen_simplify() (in module dpgen.simplify.simplify), 281
 gen_structures() (in module dp-gen.generator.lib.run_calypso), 270
 gen_sub_iter() (dpgen.tools.auto_gen_param.Iteration method), 282
 gen_test() (in module dpgen.auto_test.run), 246
 general_mdata_arginfo() (in module dpgen.arginfo), 284

`general_simplify_arginfo()` (in module `dp-gen.simplify.arginfo`), 280
`get_abacus_input_parameters()` (in module `dp-gen.generator.lib.abacus_scf`), 267
`get_abacus_STRU()` (in module `dp-gen.generator.lib.abacus_scf`), 267
`get_additional_from_STRU()` (in module `dp-gen.generator.lib.abacus_scf`), 267
`get_all_dumped_forces()` (in module `dp-gen.generator.lib.lammps`), 269
`get_atom_types()` (in module `dp-gen.generator.lib.pwscf`), 270
`get_atomic_masses()` (in module `dp-gen.generator.run`), 273
`get_atoms()` (in module `dp-gen.auto_test.lib.lmp`), 225
`get_atoms_ntypes()` (in module `dp-gen.data.tools.io_lammps`), 248
`get_atype()` (in module `dp-gen.auto_test.lib.lmp`), 225
`get_base_area()` (in module `dp-gen.auto_test.lib.lammps`), 224
`get_basic_param_json()` (in module `dp-gen.tools.auto_gen_param`), 283
`get_block()` (in module `dp-gen.generator.lib.pwscf`), 270
`get_boxes()` (in module `dp-gen.auto_test.lib.vasp`), 230
`get_cell()` (in module `dp-gen.generator.lib.pwscf`), 270
`get_coords()` (in module `dp-gen.generator.lib.pwscf`), 270
`get_dumped_forces()` (in module `dp-gen.generator.lib.lammps`), 269
`get_energies()` (in module `dp-gen.auto_test.lib.vasp`), 230
`get_energy()` (in module `dp-gen.generator.lib.pwscf`), 270
`get_eos_list()` (in module `dp-gen.auto_test.lib.mfp_eosfit`), 226
`get_eos_list_3p()` (in module `dp-gen.auto_test.lib.mfp_eosfit`), 226
`get_eos_list_4p()` (in module `dp-gen.auto_test.lib.mfp_eosfit`), 226
`get_eos_list_5p()` (in module `dp-gen.auto_test.lib.mfp_eosfit`), 226
`get_eos_list_6p()` (in module `dp-gen.auto_test.lib.mfp_eosfit`), 226
`get_force()` (in module `dp-gen.generator.lib.pwscf`), 270
`get_init_data_sys()` (in module `dp-gen.tools.auto_gen_param`), 283
`get_job_id()` (`dp-gen.auto_test.lib.BatchJob.BatchJob` method), 219
`get_job_names()` (in module `dp-gen.generator.run`), 273
`get_job_root()` (`dp-gen.auto_test.lib.RemoteJob.RemoteJob` method), 221
`get_job_root()` (`dp-gen.dispatcher.LazyLocalContext.LazyLocalContext` method), 261
`get_job_root()` (`dp-gen.dispatcher.LocalContext.LocalContext` method), 263
`get_job_root()` (`dp-gen.dispatcher.SSHContext.SSHContext` method), 265
`get_job_root()` (`dp-gen.remote.RemoteJob.RemoteJob` method), 278
`get_lmp_info()` (in module `dp-gen.tools.relabel`), 283
`get_lmpbox()` (in module `dp-gen.auto_test.lib.lmp`), 225
`get_machine_info()` (in module `dp-gen.auto_test.lib.util`), 229
`get_mass_from_STRU()` (in module `dp-gen.generator.lib.abacus_scf`), 267
`get_model_devi_jobs()` (in module `dp-gen.tools.auto_gen_param`), 283
`get_multi_system()` (in module `dp-gen.simplify.simplify`), 281
`get_natoms()` (in module `dp-gen.auto_test.lib.lmp`), 225
`get_natoms()` (in module `dp-gen.generator.lib.pwscf`), 270
`get_natoms_from_stru()` (in module `dp-gen.generator.lib.abacus_scf`), 267
`get_natoms_vec()` (in module `dp-gen.auto_test.lib.lmp`), 225
`get_natomtypes()` (in module `dp-gen.auto_test.lib.lmp`), 225
`get_nev()` (in module `dp-gen.auto_test.lib.lammps`), 224
`get_nev()` (in module `dp-gen.auto_test.lib.vasp`), 230
`get_outcar_files()` (in module `dp-gen.data.tools.cessp2force_lin`), 247
`get_poscar_natoms()` (in module `dp-gen.auto_test.lib.vasp`), 230
`get_poscar_types()` (in module `dp-gen.auto_test.lib.vasp`), 230
`get_posi()` (in module `dp-gen.auto_test.lib.lmp`), 225
`get_return()` (`dp-gen.dispatcher.LazyLocalContext.LazyLocalContext` method), 261
`get_return()` (`dp-gen.dispatcher.LocalContext.LocalContext` method), 263
`get_return()` (`dp-gen.dispatcher.SSHContext.SSHContext` method), 265
`get_session_root()` (`dp-gen.auto_test.lib.RemoteJob.SSHSession` method), 222
`get_session_root()` (`dp-gen.dispatcher.SSHContext.SSHSession` method), 265
`get_session_root()` (`dp-gen.remote.RemoteJob.SSHSession` method), 278
`get_ssh_client()` (`dp-gen.auto_test.lib.RemoteJob.SSHSession` method), 222
`get_ssh_client()` (`dp-gen.dispatcher.LazyLocalContext.LazyLocalContext` method), 261

gen.dispatcher.SSHContext.SSHSession
method), 265
get_ssh_client() (*dp-gen.remote.RemoteJob.SSHSession* *method*), 278
get_stress() (*in module dpgen.auto_test.lib.lammps*), 224
get_stress() (*in module dpgen.auto_test.lib.vasp*), 230
get_stress() (*in module dpgen.generator.lib.pwscf*), 270
get_structure() (*in module dpgen.auto_test.mpdb*), 245
get_sub_system() (*dp-gen.tools.auto_gen_param.System* *method*), 282
get_sys_configs() (*in module dp-gen.tools.auto_gen_param*), 283
get_sys_index() (*in module dpgen.generator.run*), 273
get_system_cls() (*in module dpgen.simplify.simplify*), 281
get_system_list() (*in module dp-gen.tools.auto_gen_param*), 283
get_typeid() (*in module dpgen.data.tools.io_lammps*), 248
get_types() (*in module dpgen.generator.lib.pwscf*), 270
get_uuid() (*dpgen.dispatcher.Dispatcher.JobRecord* *method*), 257
get_work_root() (*dp-gen.dispatcher.LocalContext.LocalSession* *method*), 263
gpu_exclusive:
init_bulk_mdata/fp/resources[LSF]/kwargs/gpu_exclusive (*Argument*), 80
init_reaction_mdata/build/resources[LSF]/kwargs/gpu_exclusive (*Argument*), 120
init_reaction_mdata/fp/resources[LSF]/kwargs/gpu_exclusive (*Argument*), 131
init_reaction_mdata/reaxff/resources[LSF]/kwargs/gpu_exclusive (*Argument*), 108
init_surf_mdata/fp/resources[LSF]/kwargs/gpu_exclusive (*Argument*), 94
run_mdata/fp/resources[LSF]/kwargs/gpu_exclusive (*Argument*), 67
run_mdata/model_devi/resources[LSF]/kwargs/gpu_exclusive (*Argument*), 56
run_mdata/train/resources[LSF]/kwargs/gpu_exclusive (*Argument*), 44
simplify_mdata/fp/resources[LSF]/kwargs/gpu_exclusive (*Argument*), 175
simplify_mdata/model_devi/resources[LSF]/kwargs/gpu_exclusive (*Argument*), 164
simplify_mdata/train/resources[LSF]/kwargs/gpu_exclusive (*Argument*), 153
gpu_new_syntax:
init_bulk_mdata/fp/resources[LSF]/kwargs/gpu_new_syntax (*Argument*), 80
init_reaction_mdata/build/resources[LSF]/kwargs/gpu_new_syntax (*Argument*), 120
init_reaction_mdata/fp/resources[LSF]/kwargs/gpu_new_syntax (*Argument*), 131
init_reaction_mdata/reaxff/resources[LSF]/kwargs/gpu_new_syntax (*Argument*), 108
init_surf_mdata/fp/resources[LSF]/kwargs/gpu_new_syntax (*Argument*), 94
run_mdata/fp/resources[LSF]/kwargs/gpu_new_syntax (*Argument*), 66
run_mdata/model_devi/resources[LSF]/kwargs/gpu_new_syntax (*Argument*), 56
run_mdata/train/resources[LSF]/kwargs/gpu_new_syntax (*Argument*), 44
simplify_mdata/fp/resources[LSF]/kwargs/gpu_new_syntax (*Argument*), 175
simplify_mdata/model_devi/resources[LSF]/kwargs/gpu_new_syntax (*Argument*), 164
simplify_mdata/train/resources[LSF]/kwargs/gpu_new_syntax (*Argument*), 153
gpu_per_node:
init_bulk_mdata/fp/resources/gpu_per_node (*Argument*), 76
init_reaction_mdata/build/resources/gpu_per_node (*Argument*), 116
init_reaction_mdata/fp/resources/gpu_per_node (*Argument*), 127
init_reaction_mdata/reaxff/resources/gpu_per_node (*Argument*), 104
init_bulk_mdata/build/resources/gpu_per_node (*Argument*), 90
init_bulk_mdata/fp/resources/gpu_per_node (*Argument*), 62
init_reaction_mdata/build/resources/gpu_per_node (*Argument*), 51
init_reaction_mdata/fp/resources/gpu_per_node (*Argument*), 40
init_reaction_mdata/reaxff/resources/gpu_per_node (*Argument*), 171
simplify_mdata/fp/resources/gpu_per_node (*Argument*), 160
simplify_mdata/model_devi/resources/gpu_per_node (*Argument*), 148
gpu_usage:
init_bulk_mdata/fp/resources[LSF]/kwargs/gpu_usage (*Argument*), 80
init_reaction_mdata/build/resources[LSF]/kwargs/gpu_usage (*Argument*), 120
init_reaction_mdata/fp/resources[LSF]/kwargs/gpu_usage (*Argument*), 131
init_reaction_mdata/reaxff/resources[LSF]/kwargs/gpu_usage (*Argument*), 108

(Argument), 108
 init_surf_mdata/fp/resources[LSF]/kwargs/gpu_usage (Argument), 112
 (Argument), 94
 run_mdata/fp/resources[LSF]/kwargs/gpu_usage (Argument), 66
 (Argument), 66
 run_mdata/model_devi/resources[LSF]/kwargs/gpu_usage (Argument), 100
 (Argument), 55
 run_mdata/train/resources[LSF]/kwargs/gpu_usage (Argument), 86
 (Argument), 44
 simplify_mdata/fp/resources[LSF]/kwargs/gpu_usage (Argument), 59
 (Argument), 175
 simplify_mdata/model_devi/resources[LSF]/kwargs/gpu_usage (Argument), 48
 (Argument), 164
 simplify_mdata/train/resources[LSF]/kwargs/gpu_usage (Argument), 37
 (Argument), 152
 group_local_jobs() (in module dp-gen.remote.group_jobs), 279
 group_size:
 init_bulk_mdata/fp/resources/group_size (Argument), 76
 init_reaction_mdata/build/resources/group_size (Argument), 116
 init_reaction_mdata/fp/resources/group_size (Argument), 127
 init_reaction_mdata/reaxff/resources/group_size (Argument), 104
 init_surf_mdata/fp/resources/group_size (Argument), 90
 run_mdata/fp/resources/group_size (Argument), 63
 run_mdata/model_devi/resources/group_size (Argument), 52
 run_mdata/train/resources/group_size (Argument), 40
 simplify_mdata/fp/resources/group_size (Argument), 171
 simplify_mdata/model_devi/resources/group_size (Argument), 160
 simplify_mdata/train/resources/group_size (Argument), 149
 group_slurm_jobs() (in module dp-gen.remote.group_jobs), 280

H

hcp() (in module dpgen.auto_test.lib.crys), 223
 high_level:
 run_jdata[fp_style=amber/diff]/high_level (Argument), 34
 high_level_mdin:
 run_jdata[fp_style=amber/diff]/fp_params/high_level_mdin (Argument), 34
 hostname:
 init_bulk_mdata/fp/machine[SSHContext]/remote_profile/hostname (Argument), 73
 init_reaction_mdata/build/machine[SSHContext]/remote_profile/hostname (Argument), 112
 init_reaction_mdata/fp/machine[SSHContext]/remote_profile/hostname (Argument), 123
 init_reaction_mdata/reaxff/machine[SSHContext]/remote_profile/hostname (Argument), 100
 init_surf_mdata/fp/machine[SSHContext]/remote_profile/hostname (Argument), 86
 run_mdata/fp/machine[SSHContext]/remote_profile/hostname (Argument), 59
 run_mdata/model_devi/machine[SSHContext]/remote_profile/hostname (Argument), 48
 run_mdata/train/machine[SSHContext]/remote_profile/hostname (Argument), 37
 simplify_mdata/fp/machine[SSHContext]/remote_profile/hostname (Argument), 168
 simplify_mdata/model_devi/machine[SSHContext]/remote_profile/hostname (Argument), 156
 simplify_mdata/train/machine[SSHContext]/remote_profile/hostname (Argument), 145

if_cuda_multi_devices:
 init_bulk_mdata/fp/resources/strategy/if_cuda_multi_devices (Argument), 77
 init_reaction_mdata/build/resources/strategy/if_cuda_multi_devices (Argument), 116
 init_reaction_mdata/fp/resources/strategy/if_cuda_multi_devices (Argument), 128
 init_reaction_mdata/reaxff/resources/strategy/if_cuda_multi_devices (Argument), 105
 init_surf_mdata/fp/resources/strategy/if_cuda_multi_devices (Argument), 90
 run_mdata/fp/resources/strategy/if_cuda_multi_devices (Argument), 63
 run_mdata/model_devi/resources/strategy/if_cuda_multi_devices (Argument), 52
 run_mdata/train/resources/strategy/if_cuda_multi_devices (Argument), 41
 simplify_mdata/fp/resources/strategy/if_cuda_multi_devices (Argument), 172
 simplify_mdata/model_devi/resources/strategy/if_cuda_multi_devices (Argument), 161
 simplify_mdata/train/resources/strategy/if_cuda_multi_devices (Argument), 149

incar_upper() (in module dpgen.generator.lib.vasp), 271
 increase_nfail() (dp-gen.dispatcher.Dispatcher.JobRecord method), 271
 index_iteration (dp-gen.tools.auto_gen_param.Iteration property), 271

`index_system` (`dpngen.tools.auto_gen_param.System` property), 282
`info()` (in module `dpngen`), 219
`init()` (`dpngen.dispatcher.DispatcherList.DispatcherList` method), 259
`init_batch_size:`
 `run_jdata/init_batch_size` (Argument), 19
 `simplify_jdata/init_batch_size` (Argument), 136
`init_bulk_mdata` (Argument)
 `init_bulk_mdata:`, 70
`init_bulk_mdata/api_version` (Argument)
 `api_version:`, 70
`init_bulk_mdata/deepmd_version` (Argument)
 `deepmd_version:`, 70
`init_bulk_mdata/fp` (Argument)
 `fp:`, 70
`init_bulk_mdata/fp/command` (Argument)
 `command:`, 70
`init_bulk_mdata/fp/machine` (Argument)
 `machine:`, 70
`init_bulk_mdata/fp/machine/batch_type` (Argument)
 `batch_type:`, 70
`init_bulk_mdata/fp/machine/clean_asynchronously` (Argument)
 `clean_asynchronously:`, 71
`init_bulk_mdata/fp/machine/context_type` (Argument)
 `context_type:`, 71
`init_bulk_mdata/fp/machine/local_root` (Argument)
 `local_root:`, 71
`init_bulk_mdata/fp/machine/remote_root` (Argument)
 `remote_root:`, 71
`init_bulk_mdata/fp/machine[DpCloudServerContext]` (Argument)
 `remote_profile:`, 74
`init_bulk_mdata/fp/machine[DpCloudServerContext]` (Argument)
 `email:`, 74
`init_bulk_mdata/fp/machine[DpCloudServerContext]` (Argument)
 `input_data:`, 75
`init_bulk_mdata/fp/machine[DpCloudServerContext]` (Argument)
 `keep_backup:`, 75
`init_bulk_mdata/fp/machine[DpCloudServerContext]` (Argument)
 `password:`, 75
`init_bulk_mdata/fp/machine[DpCloudServerContext]` (Argument)
 `program_id:`, 75
`init_bulk_mdata/fp/machine[HDFSContext]/remote_profile` (Argument)
 `remote_profile:`, 75
`init_bulk_mdata/fp/machine[LazyLocalContext]/remote_profile` (Argument)
 `remote_profile:`, 75
`init_bulk_mdata/fp/machine[LebesgueContext]/remote_profile` (Argument)
 `remote_profile:`, 71
`init_bulk_mdata/fp/machine[LebesgueContext]/remote_profile` (Argument)
 `email:`, 72
`init_bulk_mdata/fp/machine[LebesgueContext]/remote_profile` (Argument)
 `input_data:`, 72
`init_bulk_mdata/fp/machine[LebesgueContext]/remote_profile` (Argument)
 `keep_backup:`, 72
`init_bulk_mdata/fp/machine[LebesgueContext]/remote_profile` (Argument)
 `password:`, 72
`init_bulk_mdata/fp/machine[LebesgueContext]/remote_profile` (Argument)
 `program_id:`, 72
`init_bulk_mdata/fp/machine[LocalContext]/remote_profile` (Argument)
 `remote_profile:`, 74
`init_bulk_mdata/fp/machine[SSHContext]/remote_profile` (Argument)
 `remote_profile:`, 72
`init_bulk_mdata/fp/machine[SSHContext]/remote_profile/hostname` (Argument)
 `hostname:`, 73
`init_bulk_mdata/fp/machine[SSHContext]/remote_profile/key_filename` (Argument)
 `key_filename:`, 73
`init_bulk_mdata/fp/machine[SSHContext]/remote_profile/passphrase` (Argument)
 `passphrase:`, 73
`init_bulk_mdata/fp/machine[SSHContext]/remote_profile/password` (Argument)
 `password:`, 73
`init_bulk_mdata/fp/machine[SSHContext]/remote_profile/port` (Argument)
 `port:`, 73
`init_bulk_mdata/fp/machine[SSHContext]/remote_profile/tar_compress` (Argument)
 `tar_compress:`, 74
`init_bulk_mdata/fp/machine[SSHContext]/remote_profile/timeout` (Argument)
 `timeout:`, 73
`init_bulk_mdata/fp/machine[SSHContext]/remote_profile/total_secret` (Argument)
 `total_secret:`, 74

```

init_bulk_mdata/fp/machine[SSHContext]/remote_profile/username (Argument) 78
    username:, 73
init_bulk_mdata/fp/resources (Argument)
    resources:, 76
init_bulk_mdata/fp/resources/batch_type (Argument)
    batch_type:, 78
init_bulk_mdata/fp/resources/cpu_per_node (Argument)
    cpu_per_node:, 76
init_bulk_mdata/fp/resources/custom_flags (Argument)
    custom_flags:, 76
init_bulk_mdata/fp/resources/envs (Argument)
    envs:, 78
init_bulk_mdata/fp/resources/gpu_per_node (Argument)
    gpu_per_node:, 76
init_bulk_mdata/fp/resources/group_size (Argument)
    group_size:, 76
init_bulk_mdata/fp/resources/module_list (Argument)
    module_list:, 78
init_bulk_mdata/fp/resources/module_purge (Argument)
    module_purge:, 77
init_bulk_mdata/fp/resources/module_unload_list (Argument)
    module_unload_list:, 78
init_bulk_mdata/fp/resources/number_node (Argument)
    number_node:, 76
init_bulk_mdata/fp/resources/para_deg (Argument)
    para_deg:, 77
init_bulk_mdata/fp/resources/queue_name (Argument)
    queue_name:, 76
init_bulk_mdata/fp/resources/source_list (Argument)
    source_list:, 77
init_bulk_mdata/fp/resources/strategy (Argument)
    strategy:, 77
init_bulk_mdata/fp/resources/strategy/if_cuda_multi_devices (Argument)
    if_cuda_multi_devices:, 77
init_bulk_mdata/fp/resources/strategy/ratio_unfinished (Argument)
    ratio_unfinished:, 77
init_bulk_mdata/fp/resources/wait_time (Argument)
init_bulk_mdata/fp/resources[SSHContext]/remote_profile/username (Argument) 78
init_bulk_mdata/fp/resources[DistributedShell]/kwargs (Argument)
    kwargs:, 81
init_bulk_mdata/fp/resources[DpCloudServer]/kwargs (Argument)
    kwargs:, 80
init_bulk_mdata/fp/resources[Lebesgue]/kwargs (Argument)
    kwargs:, 81
init_bulk_mdata/fp/resources[LSF]/kwargs (Argument)
    kwargs:, 80
init_bulk_mdata/fp/resources[LSF]/kwargs/custom_gpu_line (Argument)
    custom_gpu_line:, 80
init_bulk_mdata/fp/resources[LSF]/kwargs/gpu_exclusive (Argument)
    gpu_exclusive:, 80
init_bulk_mdata/fp/resources[LSF]/kwargs/gpu_new_syntax (Argument)
    gpu_new_syntax:, 80
init_bulk_mdata/fp/resources[LSF]/kwargs/gpu_usage (Argument)
    gpu_usage:, 80
init_bulk_mdata/fp/resources[PBS]/kwargs (Argument)
    kwargs:, 79
init_bulk_mdata/fp/resources[Shell]/kwargs (Argument)
    kwargs:, 78
init_bulk_mdata/fp/resources[SlurmJobArray]/kwargs (Argument)
    kwargs:, 79
init_bulk_mdata/fp/resources[SlurmJobArray]/kwargs/custom_gpu_line (Argument)
    custom_gpu_line:, 79
init_bulk_mdata/fp/resources[Slurm]/kwargs (Argument)
    kwargs:, 79
init_bulk_mdata/fp/resources[Slurm]/kwargs/custom_gpu_line (Argument)
    custom_gpu_line:, 79
init_bulk_mdata/fp/resources[Torque]/kwargs (Argument)
    kwargs:, 79
init_bulk_mdata/fp/user_backward_files (Argument)
    user_backward_files:, 81
init_bulk_mdata/fp/user_forward_files (Argument)
    user_forward_files:, 81
init_bulk_mdata:
    init_bulk_mdata (Argument), 70

```

`init_bulk_mdata_arginfo()` (in module `dp-gen.data_arginfo`), 249
`init_data_prefix:`
 `run_jdata/init_data_prefix (Argument)`, 19
 `simplify_jdata/init_data_prefix (Argument)`, 136
`init_data_sys:`
 `run_jdata/init_data_sys (Argument)`, 19
 `simplify_jdata/init_data_sys (Argument)`, 136
`init_guess()` (in module `dp-gen.auto_test.lib.mfp_eosfit`), 226
`init_model()` (in module `dp-gen.simplify.simplify`), 281
`init_pick()` (in module `dp-gen.simplify.simplify`), 281
`init_pick_number:`
 `simplify_jdata/init_pick_number (Argument)`, 137
`init_reaction_jdata (Argument)`
 `init_reaction_jdata:`, 96
`init_reaction_jdata/cutoff (Argument)`
 `cutoff:`, 97
`init_reaction_jdata/dataset_size (Argument)`
 `dataset_size:`, 97
`init_reaction_jdata/qmkeywords (Argument)`
 `qmkeywords:`, 97
`init_reaction_jdata/reaxff (Argument)`
 `reaxff:`, 96
`init_reaction_jdata/reaxff/control (Argument)`
 `control:`, 96
`init_reaction_jdata/reaxff/data (Argument)`
 `data:`, 96
`init_reaction_jdata/reaxff/dt (Argument)`
 `dt:`, 97
`init_reaction_jdata/reaxff/dump_freq (Argument)`
 `dump_freq:`, 97
`init_reaction_jdata/reaxff/ff (Argument)`
 `ff:`, 96
`init_reaction_jdata/reaxff/nstep (Argument)`
 `nstep:`, 97
`init_reaction_jdata/reaxff/tau_t (Argument)`
 `tau_t:`, 97
`init_reaction_jdata/reaxff/temp (Argument)`
 `temp:`, 96
`init_reaction_jdata/type_map (Argument)`
 `type_map:`, 96
`init_reaction_jdata:`
 `init_reaction_jdata (Argument)`, 96
`init_reaction_jdata_arginfo()` (in module `dp-gen.data_arginfo`), 249
`init_reaction_mdata (Argument)`
 `init_reaction_mdata:`, 98
`init_reaction_mdata/api_version (Argument)`
 `api_version:`, 98
`init_reaction_mdata/build (Argument)`
 `build:`, 109
`init_reaction_mdata/build/command (Argument)`
 `command:`, 110
`init_reaction_mdata/build/machine (Argument)`
 `machine:`, 110
`init_reaction_mdata/build/machine/batch_type (Argument)`
 `batch_type:`, 110
`init_reaction_mdata/build/machine/clean_asynchronously (Argument)`
 `clean_asynchronously:`, 110
`init_reaction_mdata/build/machine/context_type (Argument)`
 `context_type:`, 110
`init_reaction_mdata/build/machine/local_root (Argument)`
 `local_root:`, 110
`init_reaction_mdata/build/machine/remote_root (Argument)`
 `remote_root:`, 110
`init_reaction_mdata/build/machine[DpCloudServerContext]/remote_profile (Argument)`
 `remote_profile:`, 114
`init_reaction_mdata/build/machine[DpCloudServerContext]/remote_email (Argument)`
 `email:`, 114
`init_reaction_mdata/build/machine[DpCloudServerContext]/remote_input_data (Argument)`
 `input_data:`, 115
`init_reaction_mdata/build/machine[DpCloudServerContext]/remote_keep_backup (Argument)`
 `keep_backup:`, 114
`init_reaction_mdata/build/machine[DpCloudServerContext]/remote_password (Argument)`
 `password:`, 114
`init_reaction_mdata/build/machine[DpCloudServerContext]/remote_program_id (Argument)`
 `program_id:`, 114
`init_reaction_mdata/build/machine[HDFSContext]/remote_profile (Argument)`
 `remote_profile:`, 115
`init_reaction_mdata/build/machine[LazyLocalContext]/remote_profile (Argument)`
 `remote_profile:`, 115
`init_reaction_mdata/build/machine[LebesgueContext]/remote_profile (Argument)`
 `remote_profile:`, 111
`init_reaction_mdata/build/machine[LebesgueContext]/remote_email (Argument)`
 `email:`, 111
`init_reaction_mdata/build/machine[LebesgueContext]/remote_input_data (Argument)`
 `input_data:`, 111

```

init_reaction_mdata/build/machine[LebesgueContext]/reaction_mdata/build/backup
  (Argument)                                argument)
  keep_backup:, 111                          envs:, 117
init_reaction_mdata/build/machine[LebesgueContext]/reaction_mdata/build/resources/gpu_per_node
  (Argument)                                (Argument)
  password:, 111                             gpu_per_node:, 116
init_reaction_mdata/build/machine[LebesgueContext]/reaction_mdata/build/resources/group_size
  (Argument)                                (Argument)
  program_id:, 111                           group_size:, 116
init_reaction_mdata/build/machine[LocalContext]/remote_profile/build/resources/module_list
  (Argument)                                (Argument)
  remote_profile:, 114                       module_list:, 117
init_reaction_mdata/build/machine[SSHContext]/remote_profile/build/resources/module_purge
  (Argument)                                (Argument)
  remote_profile:, 112                       module_purge:, 117
init_reaction_mdata/build/machine[SSHContext]/remote_profile/build/resources/module_unload_list
  (Argument)                                (Argument)
  hostname:, 112                             module_unload_list:, 117
init_reaction_mdata/build/machine[SSHContext]/remote_profile/build/resources/number_node
  (Argument)                                (Argument)
  key_filename:, 113                         number_node:, 115
init_reaction_mdata/build/machine[SSHContext]/remote_profile/build/resources/para_deg
  (Argument)                                (Argument)
  passphrase:, 113                           para_deg:, 117
init_reaction_mdata/build/machine[SSHContext]/remote_profile/build/resources/queue_name
  (Argument)                                (Argument)
  password:, 112                             queue_name:, 116
init_reaction_mdata/build/machine[SSHContext]/remote_profile/build/resources/source_list
  (Argument)                                (Argument)
  port:, 112                                 source_list:, 117
init_reaction_mdata/build/machine[SSHContext]/remote_profile/build/resources/strategy
  (Argument)                                (Argument)
  tar_compress:, 113                         strategy:, 116
init_reaction_mdata/build/machine[SSHContext]/remote_profile/build/resources/strategy/if_cuda_multi
  (Argument)                                (Argument)
  timeout:, 113                              if_cuda_multi_devices:, 116
init_reaction_mdata/build/machine[SSHContext]/remote_profile/build/resources/strategy/ratio_unfinis
  (Argument)                                (Argument)
  totp_secret:, 113                          ratio_unfinished:, 117
init_reaction_mdata/build/machine[SSHContext]/remote_profile/build/resources/wait_time
  (Argument)                                (Argument)
  username:, 112                             wait_time:, 118
init_reaction_mdata/build/resources (Argu- init_reaction_mdata/build/resources[DistributedShell]/kwar
  ment)                                (Argument)
  resources:, 115                           kwargs:, 121
init_reaction_mdata/build/resources/batch_typeinit_reaction_mdata/build/resources[DpCloudServer]/kwargs
  (Argument)                                (Argument)
  batch_type:, 118                           kwargs:, 119
init_reaction_mdata/build/resources/cpu_per_nodeinit_reaction_mdata/build/resources[Lebesgue]/kwargs
  (Argument)                                (Argument)
  cpu_per_node:, 115                         kwargs:, 120
init_reaction_mdata/build/resources/custom_flagsinit_reaction_mdata/build/resources[LSF]/kwargs
  (Argument)                                (Argument)
  custom_flags:, 116                         kwargs:, 119

```

```

init_reaction_mdata/build/resources[LSF]/kwargs/custom_gpu_line
    (Argument)                                context_type:, 122
    custom_gpu_line:, 120                     init_reaction_mdata/fp/machine/local_root
init_reaction_mdata/build/resources[LSF]/kwargs/gpu_exclusive
    (Argument)                                local_root:, 121
    gpu_exclusive:, 120                       init_reaction_mdata/fp/machine/remote_root
init_reaction_mdata/build/resources[LSF]/kwargs/gpu_new_syntax
    (Argument)                                remote_root:, 122
    gpu_new_syntax:, 120                     init_reaction_mdata/fp/machine[DpCloudServerContext]/remote_root
init_reaction_mdata/build/resources[LSF]/kwargs/gpu_usage
    (Argument)                                remote_profile:, 125
    gpu_usage:, 120                          init_reaction_mdata/fp/machine[DpCloudServerContext]/remote_profile
init_reaction_mdata/build/resources[PBS]/kwargs
    (Argument)                                email:, 125
    kwargs:, 119                             init_reaction_mdata/fp/machine[DpCloudServerContext]/remote_profile
init_reaction_mdata/build/resources[Shell]/kwargs
    (Argument)                                input_data:, 126
    kwargs:, 118                             init_reaction_mdata/fp/machine[DpCloudServerContext]/remote_profile
init_reaction_mdata/build/resources[SlurmJobArray]/kwargs
    (Argument)                                keep_backup:, 126
    kwargs:, 119                             init_reaction_mdata/fp/machine[DpCloudServerContext]/remote_profile
init_reaction_mdata/build/resources[SlurmJobArray]/kwargs/custom_gpu_line
    (Argument)                                password:, 126
    custom_gpu_line:, 119                   init_reaction_mdata/fp/machine[DpCloudServerContext]/remote_profile
init_reaction_mdata/build/resources[Slurm]/kwargs
    (Argument)                                program_id:, 126
    kwargs:, 118                             init_reaction_mdata/fp/machine[HDFSContext]/remote_profile
init_reaction_mdata/build/resources[Slurm]/kwargs/custom_gpu_line
    (Argument)                                remote_profile:, 127
    custom_gpu_line:, 118                   init_reaction_mdata/fp/machine[LazyLocalContext]/remote_profile
init_reaction_mdata/build/resources[Torque]/kwargs
    (Argument)                                remote_profile:, 126
    kwargs:, 119                             init_reaction_mdata/fp/machine[LebesgueContext]/remote_profile
init_reaction_mdata/build/user_backward_files
    (Argument)                                remote_profile:, 122
    user_backward_files:, 121               init_reaction_mdata/fp/machine[LebesgueContext]/remote_profile
init_reaction_mdata/build/user_forward_files
    (Argument)                                email:, 122
    user_forward_files:, 121               init_reaction_mdata/fp/machine[LebesgueContext]/remote_profile
init_reaction_mdata/deepmd_version (Argument)
    deepmd_version:, 98                    (Argument)
init_reaction_mdata/fp (Argument)
    fp:, 121                               input_data:, 123
init_reaction_mdata/fp/command (Argument)
    command:, 121                           init_reaction_mdata/fp/machine[LebesgueContext]/remote_profile
init_reaction_mdata/fp/machine (Argument)
    machine:, 121                           (Argument)
init_reaction_mdata/fp/machine/batch_type
    (Argument)                             password:, 123
    batch_type:, 121                       init_reaction_mdata/fp/machine[LebesgueContext]/remote_profile
init_reaction_mdata/fp/machine/clean_asynchronously
    (Argument)                             (Argument)
    clean_asynchronously:, 122             program_id:, 123
init_reaction_mdata/fp/machine/context_type
    init_reaction_mdata/fp/machine[LocalContext]/remote_profile
    (Argument)
    remote_profile:, 125
    init_reaction_mdata/fp/machine[SSHContext]/remote_profile

```

(Argument)	module_purge:, 128
remote_profile:, 123	init_reaction_mdata/fp/resources/module_unload_list
init_reaction_mdata/fp/machine[SSHContext]/remote_profile/hostname	file/hostname
(Argument)	module_unload_list:, 129
hostname:, 123	init_reaction_mdata/fp/resources/number_node
init_reaction_mdata/fp/machine[SSHContext]/remote_profile/key/filename	file/key/filename
(Argument)	number_node:, 127
key_filename:, 124	init_reaction_mdata/fp/resources/para_deg
init_reaction_mdata/fp/machine[SSHContext]/remote_profile/passphrase	file/passphrase
(Argument)	para_deg:, 128
passphrase:, 124	init_reaction_mdata/fp/resources/queue_name
init_reaction_mdata/fp/machine[SSHContext]/remote_profile/password	file/password
(Argument)	queue_name:, 127
password:, 124	init_reaction_mdata/fp/resources/source_list
init_reaction_mdata/fp/machine[SSHContext]/remote_profile/port	file/port
(Argument)	source_list:, 128
port:, 124	init_reaction_mdata/fp/resources/strategy
init_reaction_mdata/fp/machine[SSHContext]/remote_profile/tar/compress	file/tar/compress
(Argument)	strategy:, 128
tar_compress:, 125	init_reaction_mdata/fp/resources/strategy/if_cuda_multi_devices
init_reaction_mdata/fp/machine[SSHContext]/remote_profile/timeout	file/timeout
(Argument)	if_cuda_multi_devices:, 128
timeout:, 124	init_reaction_mdata/fp/resources/strategy/ratio_unfinished
init_reaction_mdata/fp/machine[SSHContext]/remote_profile/totp_secret	file/totp_secret
(Argument)	ratio_unfinished:, 128
totp_secret:, 125	init_reaction_mdata/fp/resources/wait_time
init_reaction_mdata/fp/machine[SSHContext]/remote_profile/username	file/username
(Argument)	wait_time:, 129
username:, 124	init_reaction_mdata/fp/resources[DistributedShell]/kwargs
init_reaction_mdata/fp/resources (Argument)	(Argument)
resources:, 127	kwargs:, 132
init_reaction_mdata/fp/resources/batch_type (Argument)	init_reaction_mdata/fp/resources[DpCloudServer]/kwargs
batch_type:, 129	(Argument)
init_reaction_mdata/fp/resources/cpu_per_node (Argument)	kwargs:, 131
cpu_per_node:, 127	init_reaction_mdata/fp/resources[Lebesgue]/kwargs
init_reaction_mdata/fp/resources/custom_flags (Argument)	(Argument)
custom_flags:, 127	kwargs:, 132
init_reaction_mdata/fp/resources/envs (Argument)	init_reaction_mdata/fp/resources[LSF]/kwargs
envs:, 129	(Argument)
init_reaction_mdata/fp/resources/gpu_per_node (Argument)	kwargs:, 131
gpu_per_node:, 127	init_reaction_mdata/fp/resources[LSF]/kwargs/custom_gpu_line
init_reaction_mdata/fp/resources/group_size (Argument)	(Argument)
group_size:, 127	custom_gpu_line:, 132
init_reaction_mdata/fp/resources/module_list (Argument)	init_reaction_mdata/fp/resources[LSF]/kwargs/gpu_exclusive
module_list:, 129	(Argument)
init_reaction_mdata/fp/resources/module_purge (Argument)	gpu_exclusive:, 131
	init_reaction_mdata/fp/resources[LSF]/kwargs/gpu_new_syntax
	(Argument)
	gpu_new_syntax:, 131
	init_reaction_mdata/fp/resources[LSF]/kwargs/gpu_usage
	(Argument)
	gpu_usage:, 131
	init_reaction_mdata/fp/resources[PBS]/kwargs
	(Argument)

kwargs:, 130	input_data:, 103
init_reaction_mdata/fp/resources[Shell]/kwargsinit_reaction_mdata/reaxff/machine[DpCloudServerContext]/remote_profile (Argument)	(Argument)
kwargs:, 129	keep_backup:, 103
init_reaction_mdata/fp/resources[SlurmJobArray]/kwargsinit_reaction_mdata/reaxff/machine[DpCloudServerContext]/remote_profile (Argument)	(Argument)
kwargs:, 130	password:, 103
init_reaction_mdata/fp/resources[SlurmJobArray]/kwargscustom_gpu_lineinit_reaction_mdata/reaxff/machine[DpCloudServerContext]/remote_profile (Argument)	(Argument)
custom_gpu_line:, 130	program_id:, 103
init_reaction_mdata/fp/resources[Slurm]/kwargsinit_reaction_mdata/reaxff/machine[HDFSContext]/remote_profile (Argument)	(Argument)
kwargs:, 130	remote_profile:, 103
init_reaction_mdata/fp/resources[Slurm]/kwargsinit_reaction_mdata/reaxff/machine[LazyLocalContext]/remote_profile (Argument)	(Argument)
custom_gpu_line:, 130	remote_profile:, 103
init_reaction_mdata/fp/resources[Torque]/kwargsinit_reaction_mdata/reaxff/machine[LebesgueContext]/remote_profile (Argument)	(Argument)
kwargs:, 130	remote_profile:, 99
init_reaction_mdata/fp/user_backward_filesinit_reaction_mdata/reaxff/machine[LebesgueContext]/remote_profile (Argument)	(Argument)
user_backward_files:, 132	email:, 99
init_reaction_mdata/fp/user_forward_filesinit_reaction_mdata/reaxff/machine[LebesgueContext]/remote_profile (Argument)	(Argument)
user_forward_files:, 132	input_data:, 100
init_reaction_mdata/reaxff (Argument)	init_reaction_mdata/reaxff/machine[LebesgueContext]/remote_profile (Argument)
reaxff:, 98	keep_backup:, 100
init_reaction_mdata/reaxff/command (Argument)	init_reaction_mdata/reaxff/machine[LebesgueContext]/remote_profile (Argument)
command:, 98	password:, 99
init_reaction_mdata/reaxff/machine (Argument)	init_reaction_mdata/reaxff/machine[LebesgueContext]/remote_profile (Argument)
machine:, 98	program_id:, 100
init_reaction_mdata/reaxff/machine/batch_typeinit_reaction_mdata/reaxff/machine[LocalContext]/remote_profile (Argument)	(Argument)
batch_type:, 98	remote_profile:, 102
init_reaction_mdata/reaxff/machine/clean_asynchronouslyinit_reaction_mdata/reaxff/machine[SSHContext]/remote_profile (Argument)	(Argument)
clean_asynchronously:, 99	remote_profile:, 100
init_reaction_mdata/reaxff/machine/context_typeinit_reaction_mdata/reaxff/machine[SSHContext]/remote_profile (Argument)	(Argument)
context_type:, 99	hostname:, 100
init_reaction_mdata/reaxff/machine/local_rootinit_reaction_mdata/reaxff/machine[SSHContext]/remote_profile (Argument)	(Argument)
local_root:, 98	key_filename:, 101
init_reaction_mdata/reaxff/machine/remote_rootinit_reaction_mdata/reaxff/machine[SSHContext]/remote_profile (Argument)	(Argument)
remote_root:, 98	passphrase:, 101
init_reaction_mdata/reaxff/machine[DpCloudServerContext]/remote_profileinit_reaction_mdata/reaxff/machine[SSHContext]/remote_profile (Argument)	(Argument)
remote_profile:, 102	password:, 101
init_reaction_mdata/reaxff/machine[DpCloudServerContext]/remote_profileinit_reaction_mdata/reaxff/machine[SSHContext]/remote_profile (Argument)	(Argument)
email:, 102	
init_reaction_mdata/reaxff/machine[DpCloudServerContext]/remote_profileinit_reaction_mdata/reaxff/machine[SSHContext]/remote_profile (Argument)	(Argument)

port:, 101	source_list:, 105
init_reaction_mdata/reaxff/machine[SSHContext]/init_reaction_mdata/reaxff/resources/strategy	init_reaction_mdata/reaxff/resources/strategy
(Argument)	(Argument)
tar_compress:, 102	strategy:, 105
init_reaction_mdata/reaxff/machine[SSHContext]/init_reaction_mdata/reaxff/resources/strategy/if_cuda_multi_devices:	init_reaction_mdata/reaxff/resources/strategy/if_cuda_multi_devices:, 105
(Argument)	(Argument)
timeout:, 101	if_cuda_multi_devices:, 105
init_reaction_mdata/reaxff/machine[SSHContext]/init_reaction_mdata/reaxff/resources/strategy/ratio_unfinished:	init_reaction_mdata/reaxff/resources/strategy/ratio_unfinished:, 105
(Argument)	(Argument)
totp_secret:, 102	ratio_unfinished:, 105
init_reaction_mdata/reaxff/machine[SSHContext]/init_reaction_mdata/reaxff/resources/wait_time	init_reaction_mdata/reaxff/resources/wait_time
(Argument)	(Argument)
username:, 100	wait_time:, 106
init_reaction_mdata/reaxff/resources (Argument)	init_reaction_mdata/reaxff/resources[DistributedShell]/kwargs
resources:, 104	(Argument)
init_reaction_mdata/reaxff/resources/batch_type:	kwargs:, 109
(Argument)	(Argument)
batch_type:, 106	kwargs:, 108
init_reaction_mdata/reaxff/resources/cpu_per_node:	init_reaction_mdata/reaxff/resources[Lebesgue]/kwargs
(Argument)	(Argument)
cpu_per_node:, 104	kwargs:, 109
init_reaction_mdata/reaxff/resources/custom_flags:	init_reaction_mdata/reaxff/resources[LSF]/kwargs
(Argument)	(Argument)
custom_flags:, 104	kwargs:, 108
init_reaction_mdata/reaxff/resources/envs	init_reaction_mdata/reaxff/resources[LSF]/kwargs/custom_gpu_line:
(Argument)	(Argument)
envs:, 106	custom_gpu_line:, 109
init_reaction_mdata/reaxff/resources/gpu_per_node:	init_reaction_mdata/reaxff/resources[LSF]/kwargs/gpu_exclusive:
(Argument)	(Argument)
gpu_per_node:, 104	gpu_exclusive:, 108
init_reaction_mdata/reaxff/resources/group_size:	init_reaction_mdata/reaxff/resources[LSF]/kwargs/gpu_new_syntax:
(Argument)	(Argument)
group_size:, 104	gpu_new_syntax:, 108
init_reaction_mdata/reaxff/resources/module_list:	init_reaction_mdata/reaxff/resources[LSF]/kwargs/gpu_usage:
(Argument)	(Argument)
module_list:, 106	gpu_usage:, 108
init_reaction_mdata/reaxff/resources/module_purge:	init_reaction_mdata/reaxff/resources[PBS]/kwargs
(Argument)	(Argument)
module_purge:, 105	kwargs:, 107
init_reaction_mdata/reaxff/resources/module_unload_list:	init_reaction_mdata/reaxff/resources[Shell]/kwargs
(Argument)	(Argument)
module_unload_list:, 106	kwargs:, 106
init_reaction_mdata/reaxff/resources/number_node:	init_reaction_mdata/reaxff/resources[SlurmJobArray]/kwargs
(Argument)	(Argument)
number_node:, 104	kwargs:, 107
init_reaction_mdata/reaxff/resources/para_deg	init_reaction_mdata/reaxff/resources[SlurmJobArray]/kwargs
(Argument)	(Argument)
para_deg:, 105	custom_gpu_line:, 107
init_reaction_mdata/reaxff/resources/queue_name:	init_reaction_mdata/reaxff/resources[Slurm]/kwargs
(Argument)	(Argument)
queue_name:, 104	kwargs:, 107
init_reaction_mdata/reaxff/resources/source_list:	init_reaction_mdata/reaxff/resources[Slurm]/kwargs/custom_gpu_line:
(Argument)	(Argument)


```

    custom_gpu_line:, 107
init_reaction_mdata/reaxff/resources[Torque]/kwargs (Argument)
    kwargs:, 107
init_reaction_mdata/reaxff/user_backward_files (Argument)
    user_backward_files:, 109
init_reaction_mdata/reaxff/user_forward_files (Argument)
    user_forward_files:, 109
init_reaction_mdata:
    init_reaction_mdata (Argument), 98
init_reaction_mdata_arginfo() (in module dp-
    gen.data.arginfo), 249
init_surf_mdata (Argument)
    init_surf_mdata:, 84
init_surf_mdata/api_version (Argument)
    api_version:, 84
init_surf_mdata/deepmd_version (Argument)
    deepmd_version:, 84
init_surf_mdata/fp (Argument)
    fp:, 84
init_surf_mdata/fp/command (Argument)
    command:, 84
init_surf_mdata/fp/machine (Argument)
    machine:, 84
init_surf_mdata/fp/machine/batch_type (Argument)
    batch_type:, 84
init_surf_mdata/fp/machine/clean_asynchronously (Argument)
    clean_asynchronously:, 85
init_surf_mdata/fp/machine/context_type (Argument)
    context_type:, 85
init_surf_mdata/fp/machine/local_root (Argument)
    local_root:, 84
init_surf_mdata/fp/machine/remote_root (Argument)
    remote_root:, 84
init_surf_mdata/fp/machine[DpCloudServerContext]/remote_profile (Argument)
    remote_profile:, 88
init_surf_mdata/fp/machine[DpCloudServerContext]/remote_profile/email:, 88
init_surf_mdata/fp/machine[DpCloudServerContext]/remote_profile/input_data:, 89
init_surf_mdata/fp/machine[DpCloudServerContext]/remote_profile/keep_backup:, 89
init_surf_mdata/fp/machine[DpCloudServerContext]/remote_profile/password:, 88
init_surf_mdata/fp/machine[DpCloudServerContext]/remote_profile/program_id:, 88
init_surf_mdata/fp/machine[HDFSContext]/remote_profile (Argument)
    remote_profile:, 89
init_surf_mdata/fp/machine[LazyLocalContext]/remote_profile (Argument)
    remote_profile:, 89
init_surf_mdata/fp/machine[LebesgueContext]/remote_profile (Argument)
    remote_profile:, 85
init_surf_mdata/fp/machine[LebesgueContext]/remote_profile/email:, 85
init_surf_mdata/fp/machine[LebesgueContext]/remote_profile/input_data:, 86
init_surf_mdata/fp/machine[LebesgueContext]/remote_profile/keep_backup:, 86
init_surf_mdata/fp/machine[LebesgueContext]/remote_profile/password:, 85
init_surf_mdata/fp/machine[LebesgueContext]/remote_profile/program_id:, 86
init_surf_mdata/fp/machine[LocalContext]/remote_profile (Argument)
    remote_profile:, 88
init_surf_mdata/fp/machine[SSHContext]/remote_profile (Argument)
    remote_profile:, 86
init_surf_mdata/fp/machine[SSHContext]/remote_profile/hostname:, 86
init_surf_mdata/fp/machine[SSHContext]/remote_profile/key_path:, 87
init_surf_mdata/fp/machine[SSHContext]/remote_profile/passphrase:, 87
init_surf_mdata/fp/machine[SSHContext]/remote_profile/port:, 87
init_surf_mdata/fp/machine[SSHContext]/remote_profile/tar_compress:, 87

```

```

init_surf_mdata/fp/machine[SSHContext]/remote_profile/total_devices:, 90
    (Argument)
    timeout:, 87
init_surf_mdata/fp/machine[SSHContext]/remote_profile/total_finished:, 91
    (Argument)
    totp_secret:, 87
init_surf_mdata/fp/machine[SSHContext]/remote_profile/username:, 86
    (Argument)
    username:, 86
init_surf_mdata/fp/resources (Argument)
    resources:, 89
init_surf_mdata/fp/resources/batch_type
    (Argument)
    batch_type:, 92
init_surf_mdata/fp/resources/cpu_per_node
    (Argument)
    cpu_per_node:, 90
init_surf_mdata/fp/resources/custom_flags
    (Argument)
    custom_flags:, 90
init_surf_mdata/fp/resources/envs (Argument)
    envs:, 92
init_surf_mdata/fp/resources/gpu_per_node
    (Argument)
    gpu_per_node:, 90
init_surf_mdata/fp/resources/group_size
    (Argument)
    group_size:, 90
init_surf_mdata/fp/resources/module_list (Argument)
    module_list:, 91
init_surf_mdata/fp/resources/module_purge
    (Argument)
    module_purge:, 91
init_surf_mdata/fp/resources/module_unload_list
    (Argument)
    module_unload_list:, 91
init_surf_mdata/fp/resources/number_node (Argument)
    number_node:, 89
init_surf_mdata/fp/resources/para_deg (Argument)
    para_deg:, 91
init_surf_mdata/fp/resources/queue_name
    (Argument)
    queue_name:, 90
init_surf_mdata/fp/resources/source_list (Argument)
    source_list:, 91
init_surf_mdata/fp/resources/strategy (Argument)
    strategy:, 90
init_surf_mdata/fp/resources/strategy/if_cuda_is_defined
    (Argument)
init_surf_mdata/fp/resources/strategy/ratio_unfinished
    (Argument)
init_surf_mdata/fp/resources/wait_time (Argument)
init_surf_mdata/fp/resources[DistributedShell]/kwargs
    (Argument)
    kwargs:, 94
init_surf_mdata/fp/resources[DpCloudServer]/kwargs
    (Argument)
    kwargs:, 93
init_surf_mdata/fp/resources[Lebesgue]/kwargs
    (Argument)
    kwargs:, 94
init_surf_mdata/fp/resources[LSF]/kwargs (Argument)
    kwargs:, 93
init_surf_mdata/fp/resources[LSF]/kwargs/custom_gpu_line
    (Argument)
    custom_gpu_line:, 94
init_surf_mdata/fp/resources[LSF]/kwargs/gpu_exclusive
    (Argument)
    gpu_exclusive:, 94
init_surf_mdata/fp/resources[LSF]/kwargs/gpu_new_syntax
    (Argument)
    gpu_new_syntax:, 94
init_surf_mdata/fp/resources[LSF]/kwargs/gpu_usage
    (Argument)
    gpu_usage:, 94
init_surf_mdata/fp/resources[PBS]/kwargs (Argument)
    kwargs:, 93
init_surf_mdata/fp/resources[Shell]/kwargs
    (Argument)
    kwargs:, 92
init_surf_mdata/fp/resources[SlurmJobArray]/kwargs
    (Argument)
    kwargs:, 93
init_surf_mdata/fp/resources[SlurmJobArray]/kwargs/custom_gpu_line
    (Argument)
    custom_gpu_line:, 93
init_surf_mdata/fp/resources[Slurm]/kwargs
    (Argument)
    kwargs:, 92
init_surf_mdata/fp/resources[Slurm]/kwargs/custom_gpu_line
    (Argument)
    custom_gpu_line:, 92
init_surf_mdata/fp/resources[Torque]/kwargs
    (Argument)
    kwargs:, 93
init_surf_mdata/fp/user_backward_files (Argument)

```

user_backward_files:, 95
 init_surf_mdata/fp/user_forward_files (Argument), 95
 user_forward_files:, 95
 init_surf_mdata:
 init_surf_mdata (Argument), 84
 init_surf_mdata_arginfo() (in module dp-gen.data.arginfo), 249
 input_data:
 init_bulk_mdata/fp/machine[DpCloudServerContext]/remote_profile/input_data (Argument), 75
 init_bulk_mdata/fp/machine[LebesgueContext]/remote_profile/input_data (Argument), 72
 init_reaction_mdata/build/machine[DpCloudServerContext]/remote_profile/input_data (Argument), 115
 init_reaction_mdata/build/machine[LebesgueContext]/remote_profile/input_data (Argument), 111
 init_reaction_mdata/fp/machine[DpCloudServerContext]/remote_profile/input_data (Argument), 126
 init_reaction_mdata/fp/machine[LebesgueContext]/remote_profile/input_data (Argument), 123
 init_reaction_mdata/reaxff/machine[DpCloudServerContext]/remote_profile/input_data (Argument), 103
 init_reaction_mdata/reaxff/machine[LebesgueContext]/remote_profile/input_data (Argument), 100
 init_surf_mdata/fp/machine[DpCloudServerContext]/remote_profile/input_data (Argument), 89
 init_surf_mdata/fp/machine[LebesgueContext]/remote_profile/input_data (Argument), 86
 run_mdata/fp/machine[DpCloudServerContext]/remote_profile/input_data (Argument), 61
 run_mdata/fp/machine[LebesgueContext]/remote_profile/input_data (Argument), 59
 run_mdata/model_devi/machine[DpCloudServerContext]/remote_profile/input_data (Argument), 51
 run_mdata/model_devi/machine[LebesgueContext]/remote_profile/input_data (Argument), 47
 run_mdata/train/machine[DpCloudServerContext]/remote_profile/input_data (Argument), 39
 run_mdata/train/machine[LebesgueContext]/remote_profile/input_data (Argument), 37
 simplify_mdata/fp/machine[DpCloudServerContext]/remote_profile/input_data (Argument), 170
 simplify_mdata/fp/machine[LebesgueContext]/remote_profile/input_data (Argument), 167
 simplify_mdata/model_devi/machine[DpCloudServerContext]/remote_profile/input_data (Argument), 159
 simplify_mdata/model_devi/machine[LebesgueContext]/remote_profile/input_data (Argument), 156
 simplify_mdata/train/machine[DpCloudServerContext]/remote_profile/input_data (Argument), 148
 simplify_mdata/train/machine[LebesgueContext]/remote_profile/input_data (Argument), 145
 input_upper() (in module dp-gen.generator.lib.pwmat), 269
 insert_data() (in module dp-gen.auto_test.lib.util), 229
 inter_deepmd() (in module dp-gen.auto_test.lib.lammps), 224
 inter_eam_alloy() (in module dp-gen.auto_test.lib.lammps), 224
 inter_eam_fs() (in module dp-gen.auto_test.lib.lammps), 224
 inter_eam() (in module dp-gen.auto_test.lib.lammps), 224
 Interstitial (class in dp-gen.auto_test.Interstitial), 236
 io_lammps (module dp-gen.data.tools.io_lammps), 248
 simplify_jdata/iter_pick_number (Argument), 137
 Iteration (class in dp-gen.tools.auto_gen_param), 281
 LocalDpFile (class in dp-gen.tools.LocalDpFile), 267
 job_id (dp-gen.dispatcher.AWS.AWS property), 255
 JobStatus (class in dp-gen.dispatcher.JobStatus), 257
 JobStatus (class in dp-gen.dispatcher.JobStatus), 220
 JobStatus (class in dp-gen.dispatcher.JobStatus), 260
 JobStatus (class in dp-gen.dispatcher.JobStatus), 276
 k_points:
 run_jdata[fp_style=abacus]/k_points (Argument), 34
 keep_backup:
 init_bulk_mdata/fp/machine[DpCloudServerContext]/remote_profile/input_data (Argument), 75
 init_bulk_mdata/fp/machine[LebesgueContext]/remote_profile/input_data (Argument), 72
 init_reaction_mdata/build/machine[DpCloudServerContext]/remote_profile/input_data (Argument), 114
 init_reaction_mdata/build/machine[LebesgueContext]/remote_profile/input_data (Argument), 111
 init_reaction_mdata/fp/machine[DpCloudServerContext]/remote_profile/input_data (Argument), 126
 init_reaction_mdata/fp/machine[LebesgueContext]/remote_profile/input_data (Argument), 123
 init_reaction_mdata/reaxff/machine[DpCloudServerContext]/remote_profile/input_data (Argument), 103
 init_reaction_mdata/reaxff/machine[LebesgueContext]/remote_profile/input_data (Argument), 100
 init_surf_mdata/fp/machine[DpCloudServerContext]/remote_profile/input_data (Argument), 89
 init_surf_mdata/fp/machine[LebesgueContext]/remote_profile/input_data (Argument), 86
 run_mdata/fp/machine[DpCloudServerContext]/remote_profile/input_data (Argument), 61

```

run_mdata/fp/machine[LebesgueContext]/remote_profile/keep_backup
  (Argument), 59 kill() (dpgen.dispatcher.LazyLocalContext.LazyLocalContext
run_mdata/model_devi/machine[DpCloudServerContext]/remote_profile/keep_backup
  (Argument), 50 kill() (dpgen.dispatcher.LocalContext.LocalContext
run_mdata/model_devi/machine[LebesgueContext]/remote_profile/keep_backup
  (Argument), 47 kill() (dpgen.dispatcher.SSHContext.SSHContext
run_mdata/train/machine[DpCloudServerContext]/remote_profile/keep_backup
  (Argument), 39 kspacing:
run_mdata/train/machine[LebesgueContext]/remote_profile/keep_backup
  (Argument), 37 (Argument), 32
simplify_mdata/fp/machine[DpCloudServerContext]/remote_profile/keep_backup
  (Argument), 170 init_bulk_mdata/fp/resources[DistributedShell]/kwargs
simplify_mdata/fp/machine[LebesgueContext]/remote_profile/keep_backup
  (Argument), 167 init_bulk_mdata/fp/resources[DpCloudServer]/kwargs
simplify_mdata/model_devi/machine[DpCloudServerContext]/remote_profile/keep_backup
  (Argument), 159 init_bulk_mdata/fp/resources[Lebesgue]/kwargs
simplify_mdata/model_devi/machine[LebesgueContext]/remote_profile/keep_backup
  (Argument), 156 init_bulk_mdata/fp/resources[LSF]/kwargs
simplify_mdata/train/machine[DpCloudServerContext]/remote_profile/keep_backup
  (Argument), 147 init_bulk_mdata/fp/resources[PBS]/kwargs
simplify_mdata/train/machine[LebesgueContext]/remote_profile/keep_backup
  (Argument), 144 init_bulk_mdata/fp/resources[Shell]/kwargs
key_filename:
  (Argument), 78
init_bulk_mdata/fp/machine[SSHContext]/remote_profile/keep_backup
  (Argument), 73 init_bulk_mdata/fp/resources[SlurmJobArray]/kwargs
  (Argument), 79
init_reaction_mdata/build/machine[SSHContext]/remote_profile/keep_backup
  (Argument), 113 init_bulk_mdata/fp/resources[Slurm]/kwargs
  (Argument), 79
init_reaction_mdata/fp/machine[SSHContext]/remote_profile/keep_backup
  (Argument), 124 init_bulk_mdata/fp/resources[Torque]/kwargs
  (Argument), 79
init_reaction_mdata/reaxff/machine[SSHContext]/remote_profile/keep_backup
  (Argument), 101 init_bulk_mdata/fp/resources[DistributedShell]/kwargs
  (Argument), 121
init_surf_mdata/fp/machine[SSHContext]/remote_profile/keep_backup
  (Argument), 87 init_bulk_mdata/fp/resources[DpCloudServer]/kwargs
  (Argument), 119
run_mdata/fp/machine[SSHContext]/remote_profile/keep_backup
  (Argument), 60 init_bulk_mdata/build/resources[Lebesgue]/kwargs
  (Argument), 120
run_mdata/model_devi/machine[SSHContext]/remote_profile/keep_backup
  (Argument), 48 init_bulk_mdata/build/resources[LSF]/kwargs
  (Argument), 119
run_mdata/train/machine[SSHContext]/remote_profile/keep_backup
  (Argument), 38 init_bulk_mdata/build/resources[PBS]/kwargs
  (Argument), 119
simplify_mdata/fp/machine[SSHContext]/remote_profile/keep_backup
  (Argument), 168 init_bulk_mdata/build/resources[Shell]/kwargs
  (Argument), 118
simplify_mdata/model_devi/machine[SSHContext]/remote_profile/keep_backup
  (Argument), 157 init_bulk_mdata/build/resources[SlurmJobArray]/kwargs
  (Argument), 119
simplify_mdata/train/machine[SSHContext]/remote_profile/keep_backup
  (Argument), 146 init_bulk_mdata/build/resources[Slurm]/kwargs
  (Argument), 118
keywords:
  init_reaction_mdata/build/resources[Torque]/kwargs
run_jdata[fp_style=gaussian]/fp_params/keywords (Argument), 119
  (Argument), 30 init_reaction_mdata/fp/resources[DistributedShell]/kwargs
simplify_jdata[gaussian]/fp_params/keywords (Argument), 132
  (Argument), 141 init_reaction_mdata/fp/resources[DpCloudServer]/kwargs
  (Argument), 131
keywords_high_multiplicity:
  (Argument), 132
run_jdata[fp_style=gaussian]/fp_params/keywords_high_multiplicity
  (Argument), 31 init_reaction_mdata/fp/resources[Lebesgue]/kwargs
  (Argument), 132
simplify_jdata[gaussian]/fp_params/keywords_high_multiplicity
  (Argument), 132

```

(Argument), 131
 init_reaction_mdata/fp/resources[PBS]/kwargs (Argument), 130
 init_reaction_mdata/fp/resources[Shell]/kwargs (Argument), 129
 init_reaction_mdata/fp/resources[SlurmJobArray]/kwargs (Argument), 130
 init_reaction_mdata/fp/resources[Slurm]/kwargs (Argument), 130
 init_reaction_mdata/fp/resources[Torque]/kwargs (Argument), 130
 init_reaction_mdata/reaxff/resources[DistributedShell]/kwargs (Argument), 109
 init_reaction_mdata/reaxff/resources[DpCloudServer]/kwargs (Argument), 108
 init_reaction_mdata/reaxff/resources[Lebesgue]/kwargs (Argument), 109
 init_reaction_mdata/reaxff/resources[LSF]/kwargs (Argument), 108
 init_reaction_mdata/reaxff/resources[PBS]/kwargs (Argument), 107
 init_reaction_mdata/reaxff/resources[Shell]/kwargs (Argument), 106
 init_reaction_mdata/reaxff/resources[SlurmJobArray]/kwargs (Argument), 107
 init_reaction_mdata/reaxff/resources[Slurm]/kwargs (Argument), 107
 init_reaction_mdata/reaxff/resources[Torque]/kwargs (Argument), 107
 init_surf_mdata/fp/resources[DistributedShell]/kwargs (Argument), 94
 init_surf_mdata/fp/resources[DpCloudServer]/kwargs (Argument), 93
 init_surf_mdata/fp/resources[Lebesgue]/kwargs (Argument), 94
 init_surf_mdata/fp/resources[LSF]/kwargs (Argument), 93
 init_surf_mdata/fp/resources[PBS]/kwargs (Argument), 93
 init_surf_mdata/fp/resources[Shell]/kwargs (Argument), 92
 init_surf_mdata/fp/resources[SlurmJobArray]/kwargs (Argument), 93
 init_surf_mdata/fp/resources[Slurm]/kwargs (Argument), 92
 init_surf_mdata/fp/resources[Torque]/kwargs (Argument), 93
 run_mdata/fp/resources[DistributedShell]/kwargs (Argument), 67
 run_mdata/fp/resources[DpCloudServer]/kwargs (Argument), 66
 run_mdata/fp/resources[Lebesgue]/kwargs (Argument), 67
 run_mdata/fp/resources[LSF]/kwargs (Argument), 66
 run_mdata/fp/resources[PBS]/kwargs (Argument), 65
 run_mdata/fp/resources[Shell]/kwargs (Argument), 65
 run_mdata/fp/resources[SlurmJobArray]/kwargs (Argument), 66
 run_mdata/fp/resources[Slurm]/kwargs (Argument), 65
 run_mdata/fp/resources[Torque]/kwargs (Argument), 65
 run_mdata/model_devi/resources[DistributedShell]/kwargs (Argument), 56
 run_mdata/model_devi/resources[DpCloudServer]/kwargs (Argument), 55
 run_mdata/model_devi/resources[Lebesgue]/kwargs (Argument), 56
 run_mdata/model_devi/resources[LSF]/kwargs (Argument), 55
 run_mdata/model_devi/resources[PBS]/kwargs (Argument), 54
 run_mdata/model_devi/resources[Shell]/kwargs (Argument), 54
 run_mdata/model_devi/resources[SlurmJobArray]/kwargs (Argument), 55
 run_mdata/model_devi/resources[Slurm]/kwargs (Argument), 54
 run_mdata/model_devi/resources[Torque]/kwargs (Argument), 55
 run_mdata/train/resources[DistributedShell]/kwargs (Argument), 45
 run_mdata/train/resources[DpCloudServer]/kwargs (Argument), 44
 run_mdata/train/resources[Lebesgue]/kwargs (Argument), 45
 run_mdata/train/resources[LSF]/kwargs (Argument), 44
 run_mdata/train/resources[PBS]/kwargs (Argument), 43
 run_mdata/train/resources[Shell]/kwargs (Argument), 43
 run_mdata/train/resources[SlurmJobArray]/kwargs (Argument), 43
 run_mdata/train/resources[Slurm]/kwargs (Argument), 43
 run_mdata/train/resources[Torque]/kwargs (Argument), 43
 simplify_mdata/fp/resources[DistributedShell]/kwargs (Argument), 176
 simplify_mdata/fp/resources[DpCloudServer]/kwargs (Argument), 175
 simplify_mdata/fp/resources[Lebesgue]/kwargs (Argument), 176
 simplify_mdata/fp/resources[LSF]/kwargs (Argument), 176

- (Argument), 175
 simplify_mdata/fp/resources[PBS]/kwargs (Argument), 174
 simplify_mdata/fp/resources[Shell]/kwargs (Argument), 173
 simplify_mdata/fp/resources[SlurmJobArray]/kwargs (Argument), 174
 simplify_mdata/fp/resources[Slurm]/kwargs (Argument), 174
 simplify_mdata/fp/resources[Torque]/kwargs (Argument), 174
 simplify_mdata/model_devi/resources[DistributedShell]/kwargs (Argument), 165
 simplify_mdata/model_devi/resources[DpCloudServer]/kwargs (Argument), 164
 simplify_mdata/model_devi/resources[Lebesgue]/kwargs (Argument), 165
 simplify_mdata/model_devi/resources[LSF]/kwargs (Argument), 164
 simplify_mdata/model_devi/resources[PBS]/kwargs (Argument), 163
 simplify_mdata/model_devi/resources[Shell]/kwargs (Argument), 162
 simplify_mdata/model_devi/resources[SlurmJobArray]/kwargs (Argument), 163
 simplify_mdata/model_devi/resources[Slurm]/kwargs (Argument), 163
 simplify_mdata/model_devi/resources[Torque]/kwargs (Argument), 163
 simplify_mdata/train/resources[DistributedShell]/kwargs (Argument), 153
 simplify_mdata/train/resources[DpCloudServer]/kwargs (Argument), 152
 simplify_mdata/train/resources[Lebesgue]/kwargs (Argument), 153
 simplify_mdata/train/resources[LSF]/kwargs (Argument), 152
 simplify_mdata/train/resources[PBS]/kwargs (Argument), 151
 simplify_mdata/train/resources[Shell]/kwargs (Argument), 151
 simplify_mdata/train/resources[SlurmJobArray]/kwargs (Argument), 152
 simplify_mdata/train/resources[Slurm]/kwargs (Argument), 151
 simplify_mdata/train/resources[Torque]/kwargs (Argument), 152
- L**
 labeled:
 simplify_jdata/labeled (Argument), 137
 Lammmps (class in dpgen.auto_test.Lammmps), 236
 LazyLocalContext (class in dp-gen.dispatcher.LazyLocalContext), 261
 Li4p() (in module dpgen.auto_test.lib.mfp_eosfit), 225
 link_fp_input() (in module dpgen.data.reaction), 251
 link_pp_files() (in module dpgen.tools.relabel), 283
 link_reaxff() (in module dpgen.data.reaction), 251
 link_trj() (in module dpgen.data.reaction), 251
 LinkBox() (in module dpgen.auto_test.lib.lmp), 225
 load() (dpgen.dispatcher.Dispatcher.JobRecord method), 257
 load() (dpgen.remote.group_jobs.PMap method), 279
 local_root:
 init_bulk_mdata/fp/machine/local_root (Argument), 174
 init_reaction_mdata/build/machine/local_root (Argument), 165
 init_reaction_mdata/fp/machine/local_root (Argument), 164
 init_reaction_mdata/reaxff/machine/local_root (Argument), 121
 init_surf_mdata/fp/machine/local_root (Argument), 98
 run_mdata/fp/machine/local_root (Argument), 84
 run_mdata/model_devi/machine/local_root (Argument), 57
 run_mdata/train/machine/local_root (Argument), 46
 simplify_mdata/fp/machine/local_root (Argument), 166
 simplify_mdata/model_devi/machine/local_root (Argument), 154
 simplify_mdata/train/machine/local_root (Argument), 143
 LocalContext (class in dpgen.dispatcher.LocalContext), 262
 LocalSession (class in dpgen.dispatcher.LocalContext), 263
 LOG4() (in module dpgen.auto_test.lib.mfp_eosfit), 225
 LOG5() (in module dpgen.auto_test.lib.mfp_eosfit), 225
 log_iter() (in module dpgen.auto_test.lib.utils), 230
 log_iter() (in module dpgen.generator.lib.utils), 271
 log_task() (in module dpgen.auto_test.lib.utils), 230
 log_task() (in module dpgen.generator.lib.utils), 271
 low_level:
 run_jdata[model_devi_engine=amber]/low_level (Argument), 27
 run_jdata[fp_style=amber/diff]/fp_params/low_level_mdin (Argument), 34
 LSF (class in dpgen.dispatcher.LSF), 260
 LSFJob (class in dpgen.remote.RemoteJob), 276
 lsqfit_eos() (in module dp-gen.auto_test.lib.mfp_eosfit), 226

M

- machine:
- `init_bulk_mdata/fp/machine` (Argument), 70
 - `init_reaction_mdata/build/machine` (Argument), 110
 - `init_reaction_mdata/fp/machine` (Argument), 121
 - `init_reaction_mdata/reaxff/machine` (Argument), 98
 - `init_surf_mdata/fp/machine` (Argument), 84
 - `run_mdata/fp/machine` (Argument), 57
 - `run_mdata/model_devi/machine` (Argument), 46
 - `run_mdata/train/machine` (Argument), 35
 - `simplify_mdata/fp/machine` (Argument), 165
 - `simplify_mdata/model_devi/machine` (Argument), 154
 - `simplify_mdata/train/machine` (Argument), 143
- `main()` (in module `dpgen.main`), 285
- `main_parser()` (in module `dpgen.main`), 285
- `make_abacus_md()` (in module `dpgen.data.gen`), 249
- `make_abacus_relax()` (in module `dpgen.data.gen`), 250
- `make_abacus_scf_input()` (in module `dp-gen.generator.lib.abacus_scf`), 267
- `make_abacus_scf_kpt()` (in module `dp-gen.generator.lib.abacus_scf`), 267
- `make_abacus_scf_stru()` (in module `dp-gen.generator.lib.abacus_scf`), 267
- `make_calculator()` (in module `dp-gen.auto_test.calculator`), 245
- `make_calypso_input()` (in module `dp-gen.generator.lib.make_calypso`), 269
- `make_combines()` (in module `dpgen.data.gen`), 250
- `make_combines()` (in module `dpgen.data.surf`), 251
- `make_confs()` (`dpgen.auto_test.Elastic.Elastic` method), 234
- `make_confs()` (`dpgen.auto_test.EOS.EOS` method), 233
- `make_confs()` (`dpgen.auto_test.Gamma.Gamma` method), 235
- `make_confs()` (`dpgen.auto_test.Interstitial.Interstitial` method), 236
- `make_confs()` (`dpgen.auto_test.Property.Property` method), 239
- `make_confs()` (`dpgen.auto_test.Surface.Surface` method), 240
- `make_confs()` (`dpgen.auto_test.Vacancy.Vacancy` method), 244
- `make_cp2k_input()` (in module `dp-gen.generator.lib.cp2k`), 268
- `make_cp2k_input_from_external()` (in module `dp-gen.generator.lib.cp2k`), 268
- `make_cp2k_xyz()` (in module `dpgen.generator.lib.cp2k`), 268
- `make_dispatcher()` (`dp-gen.dispatcher.DispatcherList.DispatcherList` method), 260
- `make_dispatcher()` (in module `dp-gen.dispatcher.Dispatcher`), 258
- `make_equi()` (in module `dp-gen.auto_test.common_equi`), 245
- `make_fp()` (in module `dpgen.generator.run`), 273
- `make_fp()` (in module `dpgen.simplify.simplify`), 281
- `make_fp_abacus_scf()` (in module `dp-gen.generator.run`), 273
- `make_fp_amber_diff()` (in module `dp-gen.generator.run`), 273
- `make_fp_calculation()` (in module `dp-gen.simplify.simplify`), 281
- `make_fp_configs()` (in module `dp-gen.simplify.simplify`), 281
- `make_fp_cp2k()` (in module `dpgen.generator.run`), 274
- `make_fp_gaussian()` (in module `dpgen.generator.run`), 274
- `make_fp_gaussian()` (in module `dp-gen.simplify.simplify`), 281
- `make_fp_labeled()` (in module `dp-gen.simplify.simplify`), 281
- `make_fp_pwmat()` (in module `dpgen.generator.run`), 274
- `make_fp_pwscf()` (in module `dpgen.generator.run`), 274
- `make_fp_siesta()` (in module `dpgen.generator.run`), 274
- `make_fp_task_name()` (in module `dp-gen.generator.run`), 274
- `make_fp_vasp()` (in module `dpgen.generator.run`), 274
- `make_fp_vasp()` (in module `dpgen.simplify.simplify`), 281
- `make_fp_vasp_cp_cvasp()` (in module `dp-gen.generator.run`), 274
- `make_fp_vasp_incar()` (in module `dp-gen.generator.run`), 274
- `make_fp_vasp_kp()` (in module `dpgen.generator.run`), 274
- `make_gaussian_input()` (in module `dp-gen.generator.lib.gaussian`), 269
- `make_input_file()` (`dp-gen.auto_test.ABACUS.ABACUS` method), 232
- `make_input_file()` (`dpgen.auto_test.Lammps.Lammps` method), 238
- `make_input_file()` (`dpgen.auto_test.Task.Task` method), 242
- `make_input_file()` (`dpgen.auto_test.VASP.VASP` method), 243
- `make_iter_name()` (in module `dp-gen.auto_test.lib.utils`), 230
- `make_iter_name()` (in module `dp-gen.generator.lib.utils`), 271

`make_kspacing_kpoints()` (in module `dp-gen.auto_test.lib.vasp`), 230
`make_kspacing_kpoints_stru()` (in module `dp-gen.generator.lib.abacus_scf`), 267
`make_kspacing_kpt()` (in module `dp-gen.auto_test.lib.abacus`), 223
`make_lammps_elastic()` (in module `dp-gen.auto_test.lib.lammps`), 224
`make_lammps_equi()` (in module `dp-gen.auto_test.lib.lammps`), 224
`make_lammps_eval()` (in module `dp-gen.auto_test.lib.lammps`), 224
`make_lammps_input()` (in module `dp-gen.generator.lib.lammps`), 269
`make_lammps_phonon()` (in module `dp-gen.auto_test.lib.lammps`), 224
`make_lammps_press_relax()` (in module `dp-gen.auto_test.lib.lammps`), 224
`make_lmp()` (in module `dp-gen.data.reaction`), 251
`make_model_devi()` (in module `dp-gen.generator.run`), 274
`make_model_devi()` (in module `dp-gen.simplify.simplify`), 281
`make_model_devi_conf_name()` (in module `dp-gen.generator.run`), 274
`make_model_devi_task_name()` (in module `dp-gen.generator.run`), 274
`make_path_mp()` (in module `dp-gen.auto_test.gen_confs`), 245
`make_potential_files()` (`dp-gen.auto_test.ABACUS.ABACUS` method), 232
`make_potential_files()` (`dp-gen.auto_test.Lammps.Lammps` method), 238
`make_potential_files()` (`dp-gen.auto_test.Task.Task` method), 242
`make_potential_files()` (`dp-gen.auto_test.VASP.VASP` method), 244
`make_property()` (in module `dp-gen.auto_test.common_prop`), 245
`make_property_instance()` (in module `dp-gen.auto_test.common_prop`), 245
`make_pwmat_input()` (in module `dp-gen.generator.run`), 274
`make_pwmat_input_dict()` (in module `dp-gen.generator.lib.pwmat`), 269
`make_pwmat_input_user_dict()` (in module `dp-gen.generator.lib.pwmat`), 269
`make_pwscf()` (in module `dp-gen.tools.relabel`), 283
`make_pwscf_01_runctrl_dict()` (in module `dp-gen.generator.lib.pwscf`), 270
`make_pwscf_input()` (in module `dp-gen.auto_test.lib.pwscf`), 229
`make_pwscf_input()` (in module `dp-gen.generator.lib.pwscf`), 270
`make_refine()` (in module `dp-gen.auto_test.refine`), 246
`make_repro()` (in module `dp-gen.auto_test.reproduce`), 246
`make_scale()` (in module `dp-gen.data.gen`), 250
`make_scale()` (in module `dp-gen.data.surf`), 251
`make_scale_ABACUS()` (in module `dp-gen.data.gen`), 250
`make_siesta()` (in module `dp-gen.tools.relabel`), 284
`make_siesta_input()` (in module `dp-gen.auto_test.lib.siesta`), 229
`make_siesta_input()` (in module `dp-gen.generator.lib.siesta`), 270
`make_submission()` (in module `dp-gen.dispatcher.Dispatcher`), 258
`make_submission_compat()` (in module `dp-gen.dispatcher.Dispatcher`), 258
`make_super_cell()` (in module `dp-gen.data.gen`), 250
`make_super_cell_ABACUS()` (in module `dp-gen.data.gen`), 250
`make_super_cell_poscar()` (in module `dp-gen.data.gen`), 250
`make_super_cell_pymatgen()` (in module `dp-gen.data.surf`), 251
`make_super_cell_STRU()` (in module `dp-gen.data.gen`), 250
`make_supercell_abacus()` (in module `dp-gen.generator.lib.abacus_scf`), 267
`make_train()` (in module `dp-gen.generator.run`), 274
`make_unit_cell()` (in module `dp-gen.data.gen`), 250
`make_unit_cell()` (in module `dp-gen.data.surf`), 251
`make_unit_cell_ABACUS()` (in module `dp-gen.data.gen`), 250
`make_vasp()` (in module `dp-gen.tools.relabel`), 284
`make_vasp_incar()` (in module `dp-gen.generator.run`), 274
`make_vasp_incar()` (in module `dp-gen.tools.relabel`), 284
`make_vasp_incar_ele_temp()` (in module `dp-gen.generator.run`), 274
`make_vasp_incar_user_dict()` (in module `dp-gen.generator.lib.vasp`), 271
`make_vasp_kpoints()` (in module `dp-gen.auto_test.lib.vasp`), 230
`make_vasp_kpoints_from_incar()` (in module `dp-gen.auto_test.lib.vasp`), 230
`make_vasp_md()` (in module `dp-gen.data.gen`), 250
`make_vasp_phonon_incar()` (in module `dp-gen.auto_test.lib.vasp`), 230
`make_vasp_relax()` (in module `dp-gen.data.gen`), 250
`make_vasp_relax()` (in module `dp-gen.data.surf`), 251
`make_vasp_relax_incar()` (in module `dp-gen.auto_test.lib.vasp`), 230
`make_vasp_static_incar()` (in module `dp-`

gen.auto_test.lib.vasp), 230
make_work_path() (in module *dp-gen.auto_test.lib.util*), 229
map_aws_status_to_dp-gen_status() (*dp-gen.dispatcher.AWS.AWS static method*), 256
mass_map:
 run_jdata/mass_map (*Argument*), 18
 simplify_jdata/mass_map (*Argument*), 135
mBM4() (in module *dp-gen.auto_test.lib.mfp_eosfit*), 226
mBM4poly() (in module *dp-gen.auto_test.lib.mfp_eosfit*), 226
mBM5() (in module *dp-gen.auto_test.lib.mfp_eosfit*), 226
mBM5poly() (in module *dp-gen.auto_test.lib.mfp_eosfit*), 226
mdata_arginfo() (in module *dp-gen.dispatcher.Dispatcher*), 258
mdin:
 run_jdata[model_devi_engine=amber]/mdin (*Argument*), 27
mdin_prefix:
 run_jdata[model_devi_engine=amber]/mdin_prefix (*Argument*), 27
mie() (in module *dp-gen.auto_test.lib.mfp_eosfit*), 226
mie_simple() (in module *dp-gen.auto_test.lib.mfp_eosfit*), 227
mixingWeight:
 run_jdata[fp_style=siesta]/fp_params/mixingWeight (*Argument*), 32
model_devi:
 run_mdata/model_devi (*Argument*), 45
 simplify_mdata/model_devi (*Argument*), 154
model_devi_activation_func:
 run_jdata/model_devi_activation_func (*Argument*), 21
 simplify_jdata/model_devi_activation_func (*Argument*), 139
model_devi_adapt_trust_lo:
 run_jdata[model_devi_engine=lammps]/model_devi_adapt_trust_lo (*Argument*), 24
model_devi_amber_args() (in module *dp-gen.generator.arginfo*), 272
model_devi_args() (in module *dp-gen.generator.arginfo*), 272
model_devi_clean_traj:
 run_jdata[model_devi_engine=lammps]/model_devi_clean_traj (*Argument*), 25
model_devi_dt:
 run_jdata[model_devi_engine=lammps]/model_devi_dt (*Argument*), 24
model_devi_engine:
 run_jdata/model_devi_engine (*Argument*), 22
model_devi_f_avg_relative:
 run_jdata[model_devi_engine=lammps]/model_devi_f_avg_relative (*Argument*), 25
model_devi_f_trust_hi:
 run_jdata[model_devi_engine=amber]/model_devi_f_trust_hi (*Argument*), 28
 run_jdata[model_devi_engine=lammps]/model_devi_f_trust_hi (*Argument*), 24
 run_jdata[model_devi_engine=lammps]/model_devi_jobs/mdata_arginfo (*Argument*), 23
 simplify_jdata/model_devi_f_trust_hi (*Argument*), 137
model_devi_f_trust_lo:
 run_jdata[model_devi_engine=amber]/model_devi_f_trust_lo (*Argument*), 28
 run_jdata[model_devi_engine=lammps]/model_devi_f_trust_lo (*Argument*), 24
 run_jdata[model_devi_engine=lammps]/model_devi_jobs/mdata_arginfo (*Argument*), 23
 simplify_jdata/model_devi_f_trust_lo (*Argument*), 137
model_devi_jobs:
 run_jdata[model_devi_engine=amber]/model_devi_jobs (*Argument*), 26
 run_jdata[model_devi_engine=lammps]/model_devi_jobs (*Argument*), 22
model_devi_jobs_args() (in module *dp-gen.generator.arginfo*), 272
model_devi_lmp_args() (in module *dp-gen.generator.arginfo*), 272
model_devi_merge_traj:
 run_jdata[model_devi_engine=lammps]/model_devi_merge_traj (*Argument*), 25
model_devi_nopbc:
 run_jdata[model_devi_engine=lammps]/model_devi_nopbc (*Argument*), 26
model_devi_numb_candi_f:
 run_jdata[model_devi_engine=lammps]/model_devi_numb_candi_f (*Argument*), 25
model_devi_numb_candi_v:
 run_jdata[model_devi_engine=lammps]/model_devi_numb_candi_v (*Argument*), 25
model_devi_perc_candi_f:
 run_jdata[model_devi_engine=lammps]/model_devi_perc_candi_f (*Argument*), 25
model_devi_perc_candi_v:
 run_jdata[model_devi_engine=lammps]/model_devi_perc_candi_v (*Argument*), 25
model_devi_skip:
 run_jdata[model_devi_engine=lammps]/model_devi_skip (*Argument*), 24
model_devi_v_trust_hi:
 run_jdata[model_devi_engine=lammps]/model_devi_jobs/mdata_arginfo (*Argument*), 23
 run_jdata[model_devi_engine=lammps]/model_devi_v_trust_hi (*Argument*), 24

model_devi_v_trust_lo:
 run_jdata[model_devi_engine=lammps]/model_devi_v_trust_lo
 (Argument), 23
 run_jdata[model_devi_engine=lammps]/model_devi_v_trust_lo
 (Argument), 24
 modify_input() (dpgen.auto_test.ABACUS.ABACUS
 method), 232
 modify_stru_path() (in module dp-
 gen.auto_test.lib.abacus), 223
 module
 dpgen, 219
 dpgen.argo, 284
 dpgen.auto_test, 219
 dpgen.auto_test.ABACUS, 231
 dpgen.auto_test.calculator, 245
 dpgen.auto_test.common_equi, 245
 dpgen.auto_test.common_prop, 245
 dpgen.auto_test.Elastic, 234
 dpgen.auto_test.EOS, 233
 dpgen.auto_test.Gamma, 234
 dpgen.auto_test.gen_confs, 245
 dpgen.auto_test.Interstitial, 236
 dpgen.auto_test.Lammps, 236
 dpgen.auto_test.lib, 219
 dpgen.auto_test.lib.abacus, 223
 dpgen.auto_test.lib.BatchJob, 219
 dpgen.auto_test.lib.crys, 223
 dpgen.auto_test.lib.lammps, 224
 dpgen.auto_test.lib.lmp, 225
 dpgen.auto_test.lib.mfp_eosfit, 225
 dpgen.auto_test.lib.pwscf, 229
 dpgen.auto_test.lib.RemoteJob, 220
 dpgen.auto_test.lib.siesta, 229
 dpgen.auto_test.lib.SlurmJob, 222
 dpgen.auto_test.lib.util, 229
 dpgen.auto_test.lib.utils, 230
 dpgen.auto_test.lib.vasp, 230
 dpgen.auto_test.mpdb, 245
 dpgen.auto_test.Property, 238
 dpgen.auto_test.refine, 246
 dpgen.auto_test.reproduce, 246
 dpgen.auto_test.run, 246
 dpgen.auto_test.Surface, 239
 dpgen.auto_test.Task, 240
 dpgen.auto_test.Vacancy, 244
 dpgen.auto_test.VASP, 242
 dpgen.collect, 246
 dpgen.collect.collect, 246
 dpgen.data, 246
 dpgen.data.argo, 249
 dpgen.data.gen, 249
 dpgen.data.reaction, 251
 dpgen.data.surf, 251
 dpgen.data.tools, 246
 dpgen.data.tools.bcc, 246
 dpgen.data.tools.create_random_disturb, 247
 dpgen.data.tools.fcc, 248
 dpgen.data.tools.hcp, 248
 dpgen.data.tools.io_lammps, 248
 dpgen.data.tools.sc, 249
 dpgen.database, 252
 dpgen.database.entry, 252
 dpgen.database.run, 253
 dpgen.database.vasp, 253
 dpgen.dispatcher, 255
 dpgen.dispatcher.AWS, 255
 dpgen.dispatcher.Batch, 256
 dpgen.dispatcher.Dispatcher, 257
 dpgen.dispatcher.DispatcherList, 259
 dpgen.dispatcher.JobStatus, 260
 dpgen.dispatcher.LazyLocalContext, 261
 dpgen.dispatcher.LocalContext, 262
 dpgen.dispatcher.LSF, 260
 dpgen.dispatcher.PBS, 263
 dpgen.dispatcher.Shell, 266
 dpgen.dispatcher.Slurm, 266
 dpgen.dispatcher.SSHContext, 264
 dpgen.generator, 267
 dpgen.generator.argo, 271
 dpgen.generator.lib, 267
 dpgen.generator.lib.abacus_scf, 267
 dpgen.generator.lib.cp2k, 267
 dpgen.generator.lib.cvasp, 268
 dpgen.generator.lib.ele_temp, 268
 dpgen.generator.lib.gaussian, 269
 dpgen.generator.lib.lammps, 269
 dpgen.generator.lib.make_calypso, 269
 dpgen.generator.lib.parse_calypso, 269
 dpgen.generator.lib.pwmat, 269
 dpgen.generator.lib.pwscf, 270
 dpgen.generator.lib.run_calypso, 270
 dpgen.generator.lib.siesta, 270
 dpgen.generator.lib.utils, 271
 dpgen.generator.lib.vasp, 271
 dpgen.generator.run, 272
 dpgen.main, 285
 dpgen.remote, 276
 dpgen.remote.decide_machine, 279
 dpgen.remote.group_jobs, 279
 dpgen.remote.RemoteJob, 276
 dpgen.simplify, 280
 dpgen.simplify.argo, 280
 dpgen.simplify.simplify, 281
 dpgen.tools, 281
 dpgen.tools.auto_gen_param, 281

dpgen.tools.collect_data, 283
 dpgen.tools.relabel, 283
 dpgen.tools.run_report, 284
 dpgen.tools.stat_iter, 284
 dpgen.tools.stat_sys, 284
 dpgen.tools.stat_time, 284
 dpgen.util, 285
 module_list:
 init_bulk_mdata/fp/resources/module_list (Argument), 78
 init_reaction_mdata/build/resources/module_list (Argument), 117
 init_reaction_mdata/fp/resources/module_list (Argument), 129
 init_reaction_mdata/reaxff/resources/module_list (Argument), 106
 init_surf_mdata/fp/resources/module_list (Argument), 91
 run_mdata/fp/resources/module_list (Argument), 64
 run_mdata/model_devi/resources/module_list (Argument), 53
 run_mdata/train/resources/module_list (Argument), 42
 simplify_mdata/fp/resources/module_list (Argument), 173
 simplify_mdata/model_devi/resources/module_list (Argument), 162
 simplify_mdata/train/resources/module_list (Argument), 150
 module_purge:
 init_bulk_mdata/fp/resources/module_purge (Argument), 77
 init_reaction_mdata/build/resources/module_purge (Argument), 117
 init_reaction_mdata/fp/resources/module_purge (Argument), 128
 init_reaction_mdata/reaxff/resources/module_purge (Argument), 105
 init_surf_mdata/fp/resources/module_purge (Argument), 91
 run_mdata/fp/resources/module_purge (Argument), 64
 run_mdata/model_devi/resources/module_purge (Argument), 53
 run_mdata/train/resources/module_purge (Argument), 42
 simplify_mdata/fp/resources/module_purge (Argument), 172
 simplify_mdata/model_devi/resources/module_purge (Argument), 161
 simplify_mdata/train/resources/module_purge (Argument), 150
 module_unload_list:
 init_bulk_mdata/fp/resources/module_unload_list (Argument), 78
 init_reaction_mdata/build/resources/module_unload_list (Argument), 117
 init_reaction_mdata/fp/resources/module_unload_list (Argument), 129
 init_reaction_mdata/reaxff/resources/module_unload_list (Argument), 106
 init_surf_mdata/fp/resources/module_unload_list (Argument), 91
 run_mdata/fp/resources/module_unload_list (Argument), 64
 run_mdata/model_devi/resources/module_unload_list (Argument), 53
 run_mdata/train/resources/module_unload_list (Argument), 42
 simplify_mdata/fp/resources/module_unload_list (Argument), 173
 simplify_mdata/model_devi/resources/module_unload_list (Argument), 162
 simplify_mdata/train/resources/module_unload_list (Argument), 150
 morse() (in module dpgen.auto_test.lib.mfp_eosfit), 227
 morse_3p() (in module dpgen.auto_test.lib.mfp_eosfit), 227
 morse_6p() (in module dpgen.auto_test.lib.mfp_eosfit), 227
 morse_AB() (in module dpgen.auto_test.lib.mfp_eosfit), 227
 multiplicity:
 run_jdata[fp_style=gaussian]/fp_params/multiplicity (Argument), 30
 simplify_jdata[gaussian]/fp_params/multiplicity (Argument), 141
 murnaghan() (in module dpgen.auto_test.lib.mfp_eosfit), 227
N
 NPurge
 NBandsEsti (class in dpgen.generator.lib.ele_temp), 268
 neidelay:
 run_jdata[model_devi_engine=lammps]/model_devi_jobs/neidelay (Argument), 23
 normalize() (in module dpgen.util), 285
 nproc:
 run_jdata[fp_style=gaussian]/fp_params/nproc (Argument), 31
 simplify_jdata[gaussian]/fp_params/nproc (Argument), 141
 nstep:
 init_reaction_jdata/reaxff/nstep (Argument), 97
 nsteps:
 run_jdata[model_devi_engine=amber]/nsteps (Argument), 28

- 223
- poscar_ele() (in module *dpgen.data.gen*), 250
- poscar_ele() (in module *dpgen.data.surf*), 251
- poscar_elong() (in module *dpgen.data.surf*), 251
- poscar_from_last_dump() (in module *dp-gen.auto_test.lib.lammps*), 224
- poscar_natoms() (in module *dpgen.auto_test.lib.vasp*), 230
- poscar_natoms() (in module *dpgen.data.gen*), 250
- poscar_natoms() (in module *dpgen.data.surf*), 251
- poscar_natoms() (in module *dpgen.generator.run*), 274
- poscar_scale() (in module *dpgen.auto_test.lib.vasp*), 231
- poscar_scale() (in module *dpgen.data.gen*), 250
- poscar_scale() (in module *dpgen.data.surf*), 251
- poscar_scale_abacus() (in module *dpgen.data.gen*), 250
- poscar_scale_cartesian() (in module *dp-gen.data.gen*), 250
- poscar_scale_cartesian() (in module *dp-gen.data.surf*), 251
- poscar_scale_direct() (in module *dpgen.data.gen*), 250
- poscar_scale_direct() (in module *dpgen.data.surf*), 251
- poscar_shuffle() (in module *dpgen.data.gen*), 250
- poscar_shuffle() (in module *dpgen.data.surf*), 252
- poscar_shuffle() (in module *dpgen.generator.run*), 274
- poscar_to_conf() (in module *dpgen.generator.run*), 274
- poscar_unit() (in module *dpgen.data.tools.bcc*), 246
- poscar_unit() (in module *dpgen.data.tools.diamond*), 247
- poscar_unit() (in module *dpgen.data.tools.fcc*), 248
- poscar_unit() (in module *dpgen.data.tools.hcp*), 248
- poscar_unit() (in module *dpgen.data.tools.sc*), 249
- poscar_vol() (in module *dpgen.auto_test.lib.vasp*), 231
- post_equi() (in module *dp-gen.auto_test.common_equi*), 245
- post_fp() (in module *dpgen.generator.run*), 274
- post_fp_abacus_scf() (in module *dp-gen.generator.run*), 275
- post_fp_amber_diff() (in module *dp-gen.generator.run*), 275
- post_fp_check_fail() (in module *dp-gen.generator.run*), 275
- post_fp_cp2k() (in module *dpgen.generator.run*), 275
- post_fp_gaussian() (in module *dpgen.generator.run*), 275
- post_fp_pwmat() (in module *dpgen.generator.run*), 275
- post_fp_pwscf() (in module *dpgen.generator.run*), 275
- post_fp_siesta() (in module *dpgen.generator.run*), 275
- post_fp_vasp() (in module *dpgen.generator.run*), 275
- post_model_devi() (in module *dpgen.generator.run*), 275
- post_model_devi() (in module *dp-gen.simplify.simplify*), 281
- post_process() (*dpgen.auto_test.Elastic.Elastic* method), 234
- post_process() (*dpgen.auto_test.EOS.EOS* method), 233
- post_process() (*dpgen.auto_test.Gamma.Gamma* method), 235
- post_process() (*dpgen.auto_test.Interstitial.Interstitial* method), 236
- post_process() (*dpgen.auto_test.Property.Property* method), 239
- post_process() (*dpgen.auto_test.Surface.Surface* method), 240
- post_process() (*dpgen.auto_test.Vacancy.Vacancy* method), 244
- post_property() (in module *dp-gen.auto_test.common_prop*), 245
- post_repro() (in module *dpgen.auto_test.reproduce*), 246
- post_train() (in module *dpgen.generator.run*), 275
- predict() (*dpgen.generator.lib.ele_temp.NBandsEsti* method), 268
- press:
- run_jdata[model_devi_engine=lammps]/model_devi_jobs/pro (Argument), 22
- process_outcar_file_v5_dev() (in module *dp-gen.data.tools.cessp2force_lin*), 247
- program_id:
- init_bulk_mdata/fp/machine[DpCloudServerContext]/remote (Argument), 75
- init_bulk_mdata/fp/machine[LebesgueContext]/remote_pro (Argument), 72
- init_reaction_mdata/build/machine[DpCloudServerContext] (Argument), 114
- init_reaction_mdata/build/machine[LebesgueContext]/rem (Argument), 111
- init_reaction_mdata/fp/machine[DpCloudServerContext]/r (Argument), 126
- init_reaction_mdata/fp/machine[LebesgueContext]/remote (Argument), 123
- init_reaction_mdata/reaxff/machine[DpCloudServerContext] (Argument), 103
- init_reaction_mdata/reaxff/machine[LebesgueContext]/re (Argument), 100
- init_surf_mdata/fp/machine[DpCloudServerContext]/remote (Argument), 88
- init_surf_mdata/fp/machine[LebesgueContext]/remote_pro (Argument), 86
- run_mdata/fp/machine[DpCloudServerContext]/remote_prof (Argument), 61

run_mdata/fp/machine[LebesgueContext]/remote_profile/program_id
 (Argument), 58
 run_mdata/model_devi/machine[DpCloudServerContext]/remote_profile/program_id
 (Argument), 50
 run_mdata/model_devi/machine[LebesgueContext]/remote_profile/program_id
 (Argument), 47
R
 run_mdata/train/machine[DpCloudServerContext]/remote_profile/program_id
 (Argument), 39
 run_mdata/train/machine[LebesgueContext]/remote_profile/program_id
 (Argument), 36
 simplify_mdata/fp/machine[DpCloudServerContext]/remote_profile/program_id
 (Argument), 170
 simplify_mdata/fp/machine[LebesgueContext]/remote_profile/program_id
 (Argument), 167
 simplify_mdata/model_devi/machine[DpCloudServerContext]/remote_profile/program_id
 (Argument), 159
 simplify_mdata/model_devi/machine[LebesgueContext]/remote_profile/program_id
 (Argument), 155
 simplify_mdata/train/machine[DpCloudServerContext]/remote_profile/program_id
 (Argument), 147
 simplify_mdata/train/machine[LebesgueContext]/remote_profile/program_id
 (Argument), 144
 Property (class in *dpgen.auto_test.Property*), 238
Q
 qm_charge:
 run_jdata[model_devi_engine=amber]/qm_charge
 (Argument), 28
 qm_region:
 run_jdata[model_devi_engine=amber]/qm_region
 (Argument), 27
 qmkeywords:
 init_reaction_jdata/qmkeywords (Argument),
 97
 queue_name:
 init_bulk_mdata/fp/resources/queue_name
 (Argument), 76
 init_reaction_mdata/build/resources/queue_name
 (Argument), 116
 init_reaction_mdata/fp/resources/queue_name
 (Argument), 127
 init_reaction_mdata/reaxff/resources/queue_name
 (Argument), 104
 init_surf_mdata/fp/resources/queue_name
 (Argument), 90
 run_mdata/fp/resources/queue_name (Argu-
 ment), 63
 run_mdata/model_devi/resources/queue_name
 (Argument), 52
 run_mdata/train/resources/queue_name (Ar-
 gument), 40
 simplify_mdata/fp/resources/queue_name
 (Argument), 171
 init_bulk_mdata/fp/resources/strategy/ratio_unfinished
 (Argument), 77
 init_reaction_mdata/build/resources/strategy/ratio_unf
 (Argument), 117
 init_reaction_mdata/fp/resources/strategy/ratio_unfini
 (Argument), 128
 init_reaction_mdata/reaxff/resources/strategy/ratio_un
 (Argument), 105
 init_surf_mdata/fp/resources/strategy/ratio_unfinished
 (Argument), 91
 run_mdata/fp/resources/strategy/ratio_unfinished
 (Argument), 63
 run_mdata/model_devi/resources/strategy/ratio_unfinish
 (Argument), 52
 run_mdata/train/resources/strategy/ratio_unfinished
 (Argument), 41
 simplify_mdata/fp/resources/strategy/ratio_unfinished
 (Argument), 172
 simplify_mdata/model_devi/resources/strategy/ratio_unf
 (Argument), 161
 simplify_mdata/train/resources/strategy/ratio_unfinish
 (Argument), 149
 rBM4() (in module *dpgen.auto_test.lib.mfp_eosfit*), 227
 rBM4_pv() (in module *dpgen.auto_test.lib.mfp_eosfit*),
 227
 rBM5() (in module *dpgen.auto_test.lib.mfp_eosfit*), 227
 rBM5_pv() (in module *dpgen.auto_test.lib.mfp_eosfit*),
 227

`read()` (*dpgen.dispatcher.LazyLocalContext.SPRetObj* method), 262
`read()` (*dpgen.dispatcher.LocalContext.SPRetObj* method), 263
`read_file()` (*dpgen.dispatcher.LazyLocalContext.LazyLocalContext* method), 261
`read_file()` (*dpgen.dispatcher.LocalContext.LocalContext* method), 263
`read_file()` (*dpgen.dispatcher.SSHContext.SSHContext* method), 265
`read_ve()` (in module *dpgen.auto_test.lib.mfp_eosfit*), 228
`read_velp()` (in module *dpgen.auto_test.lib.mfp_eosfit*), 228
`read_vlp()` (in module *dpgen.auto_test.lib.mfp_eosfit*), 228
`readlines()` (*dpgen.dispatcher.LazyLocalContext.SPRetObj* method), 262
`readlines()` (*dpgen.dispatcher.LocalContext.SPRetObj* method), 263
`reaxff:`
 `init_reaction_jdata/reaxff` (*Argument*), 96
 `init_reaction_mdata/reaxff` (*Argument*), 98
`reciprocal_box()` (in module *dp-gen.auto_test.lib.vasp*), 231
`record_finish()` (*dp-gen.dispatcher.Dispatcher.JobRecord* method), 258
`record_iter()` (in module *dpgen.auto_test.lib.utils*), 230
`record_iter()` (in module *dpgen.generator.lib.utils*), 271
`record_remote_context()` (*dp-gen.dispatcher.Dispatcher.JobRecord* method), 258
`register_iteration()` (*dp-gen.tools.auto_gen_param.Iteration* class method), 282
`register_sub_iteration()` (*dp-gen.tools.auto_gen_param.Iteration* class method), 282
`register_sub_system()` (*dp-gen.tools.auto_gen_param.System* class method), 282
`register_system()` (*dp-gen.tools.auto_gen_param.System* class method), 282
`regulate_poscar()` (in module *dp-gen.auto_test.lib.vasp*), 231
`remote_profile:`
 `init_bulk_mdata/fp/machine[DpCloudServerContext]/remote_profile` (*Argument*), 74
 `init_bulk_mdata/fp/machine[HDFSContext]/remote_profile` (*Argument*), 75
 `init_bulk_mdata/fp/machine[LazyLocalContext]/remote_profile` (*Argument*), 75
 `init_bulk_mdata/fp/machine[LocalContext]/remote_profile` (*Argument*), 74
 `init_bulk_mdata/fp/machine[SSHContext]/remote_profile` (*Argument*), 72
 `init_reaction_mdata/build/machine[DpCloudServerContext]/remote_profile` (*Argument*), 114
 `init_reaction_mdata/build/machine[HDFSContext]/remote_profile` (*Argument*), 115
 `init_reaction_mdata/build/machine[LazyLocalContext]/remote_profile` (*Argument*), 115
 `init_reaction_mdata/build/machine[LebesgueContext]/remote_profile` (*Argument*), 111
 `init_reaction_mdata/build/machine[LocalContext]/remote_profile` (*Argument*), 114
 `init_reaction_mdata/build/machine[SSHContext]/remote_profile` (*Argument*), 112
 `init_reaction_mdata/fp/machine[DpCloudServerContext]/remote_profile` (*Argument*), 125
 `init_reaction_mdata/fp/machine[HDFSContext]/remote_profile` (*Argument*), 127
 `init_reaction_mdata/fp/machine[LazyLocalContext]/remote_profile` (*Argument*), 126
 `init_reaction_mdata/fp/machine[LebesgueContext]/remote_profile` (*Argument*), 122
 `init_reaction_mdata/fp/machine[LocalContext]/remote_profile` (*Argument*), 125
 `init_reaction_mdata/fp/machine[SSHContext]/remote_profile` (*Argument*), 123
 `init_reaction_mdata/reaxff/machine[DpCloudServerContext]/remote_profile` (*Argument*), 102
 `init_reaction_mdata/reaxff/machine[HDFSContext]/remote_profile` (*Argument*), 103
 `init_reaction_mdata/reaxff/machine[LazyLocalContext]/remote_profile` (*Argument*), 103
 `init_reaction_mdata/reaxff/machine[LebesgueContext]/remote_profile` (*Argument*), 99
 `init_reaction_mdata/reaxff/machine[LocalContext]/remote_profile` (*Argument*), 102
 `init_reaction_mdata/reaxff/machine[SSHContext]/remote_profile` (*Argument*), 100
 `init_surf_mdata/fp/machine[DpCloudServerContext]/remote_profile` (*Argument*), 88
 `init_surf_mdata/fp/machine[HDFSContext]/remote_profile` (*Argument*), 89
 `init_surf_mdata/fp/machine[LazyLocalContext]/remote_profile` (*Argument*), 89
 `init_surf_mdata/fp/machine[LebesgueContext]/remote_profile` (*Argument*), 85
 `init_surf_mdata/fp/machine[LocalContext]/remote_profile` (*Argument*), 88

init_surf_mdata/fp/machine[SSHContext]/remote_profile, 86
 init_surf_mdata/model_devi/machine[LazyLocalContext]/remote_profile, 159
 run_mdata/fp/machine[DpCloudServerContext]/remote_profile, 61
 run_mdata/fp/machine[LebesgueContext]/remote_profile, 155
 run_mdata/fp/machine[HDFSContext]/remote_profile, 62
 run_mdata/fp/machine[LocalContext]/remote_profile, 158
 run_mdata/fp/machine[LazyLocalContext]/remote_profile, 62
 run_mdata/fp/machine[SSHContext]/remote_profile, 156
 run_mdata/fp/machine[LebesgueContext]/remote_profile, 58
 run_mdata/fp/machine[DpCloudServerContext]/remote_profile, 147
 run_mdata/fp/machine[LocalContext]/remote_profile, 61
 run_mdata/fp/machine[HDFSContext]/remote_profile, 148
 run_mdata/fp/machine[SSHContext]/remote_profile, 59
 run_mdata/fp/machine[LazyLocalContext]/remote_profile, 148
 run_mdata/model_devi/machine[DpCloudServerContext]/remote_profile, 50
 run_mdata/model_devi/machine[LebesgueContext]/remote_profile, 144
 run_mdata/model_devi/machine[HDFSContext]/remote_profile, 51
 run_mdata/model_devi/machine[LocalContext]/remote_profile, 146
 run_mdata/model_devi/machine[LazyLocalContext]/remote_profile, 51
 run_mdata/model_devi/machine[SSHContext]/remote_profile, 145
 run_mdata/model_devi/machine[LebesgueContext]/remote_profile, 47
 run_mdata/model_devi/machine[LocalContext]/remote_profile, 71
 run_mdata/model_devi/machine[SSHContext]/remote_profile, 49
 run_mdata/model_devi/machine[SSHContext]/remote_profile, 110
 run_mdata/train/machine[DpCloudServerContext]/remote_profile, 39
 run_mdata/train/machine[HDFSContext]/remote_profile, 98
 run_mdata/train/machine[LocalContext]/remote_profile, 40
 run_mdata/train/machine[LazyLocalContext]/remote_profile, 84
 run_mdata/train/machine[LebesgueContext]/remote_profile, 57
 run_mdata/train/machine[LocalContext]/remote_profile, 36
 run_mdata/train/machine[SSHContext]/remote_profile, 35
 run_mdata/train/machine[SSHContext]/remote_profile, 37
 simplify_mdata/fp/machine[DpCloudServerContext]/remote_profile, 169
 simplify_mdata/fp/machine[HDFSContext]/remote_profile, 171
 simplify_mdata/fp/machine[LazyLocalContext]/remote_profile, 170
 simplify_mdata/fp/machine[LebesgueContext]/remote_profile, 167
 simplify_mdata/fp/machine[LocalContext]/remote_profile, 169
 simplify_mdata/fp/machine[SSHContext]/remote_profile, 167
 simplify_mdata/model_devi/machine[DpCloudServerContext]/remote_profile, 158
 simplify_mdata/model_devi/machine[HDFSContext]/remote_profile, 159
 init_bulk_mdata/fp/machine/remote_root, 71
 init_reaction_mdata/build/machine/remote_root, 110
 init_reaction_mdata/fp/machine/remote_root, 122
 init_reaction_mdata/reaxff/machine/remote_root, 98
 init_surf_mdata/fp/machine/remote_root, 84
 run_mdata/fp/machine/remote_root, 57
 run_mdata/model_devi/machine/remote_root, 46
 run_mdata/train/machine/remote_root, 35
 simplify_mdata/fp/machine/remote_root, 156
 simplify_mdata/model_devi/machine/remote_root, 154
 simplify_mdata/train/machine/remote_root, 143
 RemoteJob (class in *dpgen.auto_test.lib.RemoteJob*), 221
 RemoteJob (class in *dpgen.remote.RemoteJob*), 277
 repeat_to_length() (in module *dp-auto_test.lib.utils*), 230
 repeat_to_length() (in module *dp-generator.lib.utils*), 271
 replace() (in module *dpgen.auto_test.lib.utils*), 230
 replace() (in module *dpgen.data.surf*), 252
 replace() (in module *dpgen.generator.lib.utils*), 271
 repro_ve() (in module *dpgen.auto_test.lib.mfp_eosfit*),

228
 repro_vp() (in module dpgen.auto_test.lib.mfp_eosfit), 228
 res_birch() (in module dpgen.auto_test.lib.mfp_eosfit), 228
 res_BM4() (in module dpgen.auto_test.lib.mfp_eosfit), 228
 res_BM5() (in module dpgen.auto_test.lib.mfp_eosfit), 228
 res_Li4p() (in module dpgen.auto_test.lib.mfp_eosfit), 228
 res_LOG4() (in module dpgen.auto_test.lib.mfp_eosfit), 228
 res_LOG5() (in module dpgen.auto_test.lib.mfp_eosfit), 228
 res_mBM4() (in module dpgen.auto_test.lib.mfp_eosfit), 228
 res_mBM4poly() (in module dpgen.auto_test.lib.mfp_eosfit), 228
 res_mBM5() (in module dpgen.auto_test.lib.mfp_eosfit), 228
 res_mBM5poly() (in module dpgen.auto_test.lib.mfp_eosfit), 228
 res_mie() (in module dpgen.auto_test.lib.mfp_eosfit), 228
 res_mie_simple() (in module dpgen.auto_test.lib.mfp_eosfit), 228
 res_morse() (in module dpgen.auto_test.lib.mfp_eosfit), 228
 res_morse_3p() (in module dpgen.auto_test.lib.mfp_eosfit), 228
 res_morse_6p() (in module dpgen.auto_test.lib.mfp_eosfit), 228
 res_morse_AB() (in module dpgen.auto_test.lib.mfp_eosfit), 228
 res_murnaghan() (in module dpgen.auto_test.lib.mfp_eosfit), 228
 res_rBM4() (in module dpgen.auto_test.lib.mfp_eosfit), 228
 res_rBM4_pv() (in module dpgen.auto_test.lib.mfp_eosfit), 228
 res_rBM5() (in module dpgen.auto_test.lib.mfp_eosfit), 228
 res_rBM5_pv() (in module dpgen.auto_test.lib.mfp_eosfit), 228
 res_rPT4() (in module dpgen.auto_test.lib.mfp_eosfit), 229
 res_rPT4_pv() (in module dpgen.auto_test.lib.mfp_eosfit), 229
 res_rPT5() (in module dpgen.auto_test.lib.mfp_eosfit), 229
 res_rPT5_pv() (in module dpgen.auto_test.lib.mfp_eosfit), 229
 res_SJX_5p() (in module dpgen.auto_test.lib.mfp_eosfit), 228
 res_SJX_v2() (in module dpgen.auto_test.lib.mfp_eosfit), 228
 res_TEOS() (in module dpgen.auto_test.lib.mfp_eosfit), 228
 res_universal() (in module dpgen.auto_test.lib.mfp_eosfit), 229
 res_vinet() (in module dpgen.auto_test.lib.mfp_eosfit), 229
 res_vinet_pv() (in module dpgen.auto_test.lib.mfp_eosfit), 229
 resources:
 init_bulk_mdata/fp/resources (Argument), 76
 init_reaction_mdata/build/resources (Argument), 115
 init_reaction_mdata/fp/resources (Argument), 127
 init_reaction_mdata/reaxff/resources (Argument), 104
 init_surf_mdata/fp/resources (Argument), 89
 run_mdata/fp/resources (Argument), 62
 run_mdata/model_devi/resources (Argument), 51
 run_mdata/train/resources (Argument), 40
 simplify_mdata/fp/resources (Argument), 171
 simplify_mdata/model_devi/resources (Argument), 160
 simplify_mdata/train/resources (Argument), 148
 return_direction() (dpgen.auto_test.Gamma.Gamma method), 235
 revise_by_keys() (in module dpgen.generator.run), 275
 revise_lmp_input_dump() (in module dpgen.generator.run), 275
 revise_lmp_input_model() (in module dpgen.generator.run), 275
 revise_lmp_input_plm() (in module dpgen.generator.run), 275
 rPT4() (in module dpgen.auto_test.lib.mfp_eosfit), 227
 rPT4_pv() (in module dpgen.auto_test.lib.mfp_eosfit), 227
 rPT5() (in module dpgen.auto_test.lib.mfp_eosfit), 227
 rPT5_pv() (in module dpgen.auto_test.lib.mfp_eosfit), 227
 run_abacus_md() (in module dpgen.data.gen), 250
 run_abacus_relax() (in module dpgen.data.gen), 250
 run_build_dataset() (in module dpgen.data.reaction), 251
 run_calypso_model_devi() (in module dpgen.generator.lib.run_calypso), 270
 run_equi() (in module dpgen.auto_test.common_equi), 245

`run_fp()` (in module `dpgen.data.reaction`), 251
`run_fp()` (in module `dpgen.generator.run`), 275
`run_fp_inner()` (in module `dpgen.generator.run`), 275
`run_iter()` (in module `dpgen.generator.run`), 275
`run_iter()` (in module `dpgen.simplify.simplify`), 281
`run_jdata` (Argument)
 `run_jdata`:, 18
`run_jdata/default_training_param` (Argument)
 `default_training_param`:, 20
`run_jdata/detailed_report_make_fp` (Argument)
 `detailed_report_make_fp`:, 21
`run_jdata/dp_compress` (Argument)
 `dp_compress`:, 20
`run_jdata/fp_accurate_soft_threshold` (Argument)
 `fp_accurate_soft_threshold`:, 21
`run_jdata/fp_accurate_threshold` (Argument)
 `fp_accurate_threshold`:, 21
`run_jdata/fp_cluster_vacuum` (Argument)
 `fp_cluster_vacuum`:, 21
`run_jdata/fp_style` (Argument)
 `fp_style`:, 29
`run_jdata/fp_task_max` (Argument)
 `fp_task_max`:, 21
`run_jdata/fp_task_min` (Argument)
 `fp_task_min`:, 21
`run_jdata/init_batch_size` (Argument)
 `init_batch_size`:, 19
`run_jdata/init_data_prefix` (Argument)
 `init_data_prefix`:, 19
`run_jdata/init_data_sys` (Argument)
 `init_data_sys`:, 19
`run_jdata/mass_map` (Argument)
 `mass_map`:, 18
`run_jdata/model_devi_activation_func` (Argument)
 `model_devi_activation_func`:, 21
`run_jdata/model_devi_engine` (Argument)
 `model_devi_engine`:, 22
`run_jdata/numb_models` (Argument)
 `numb_models`:, 19
`run_jdata/sys_batch_size` (Argument)
 `sys_batch_size`:, 19
`run_jdata/sys_configs` (Argument)
 `sys_configs`:, 19
`run_jdata/sys_configs_prefix` (Argument)
 `sys_configs_prefix`:, 19
`run_jdata/sys_format` (Argument)
 `sys_format`:, 19
`run_jdata/training_init_model` (Argument)
 `training_init_model`:, 20
`run_jdata/training_iter0_model_path` (Argument)
 `training_iter0_model_path`:, 19
`run_jdata/training_reuse_iter` (Argument)
 `training_reuse_iter`:, 20
`run_jdata/training_reuse_num_steps` (Argument)
 `training_reuse_num_steps`:, 20
`run_jdata/training_reuse_old_ratio` (Argument)
 `training_reuse_old_ratio`:, 20
`run_jdata/training_reuse_start_lr` (Argument)
 `training_reuse_start_lr`:, 20
`run_jdata/training_reuse_start_pref_e` (Argument)
 `training_reuse_start_pref_e`:, 20
`run_jdata/training_reuse_start_pref_f` (Argument)
 `training_reuse_start_pref_f`:, 21
`run_jdata/type_map` (Argument)
 `type_map`:, 18
`run_jdata/use_ele_temp` (Argument)
 `use_ele_temp`:, 18
`run_jdata`:
 `run_jdata` (Argument), 18
`run_jdata_arginfo()` (in module `dp-gen.generator.arginfo`), 272
`run_jdata[fp_style=abacus]/fp_dpks_descriptor` (Argument)
 `fp_dpks_descriptor`:, 33
`run_jdata[fp_style=abacus]/fp_incar` (Argument)
 `fp_incar`:, 33
`run_jdata[fp_style=abacus]/fp_kpt_file` (Argument)
 `fp_kpt_file`:, 33
`run_jdata[fp_style=abacus]/fp_orb_files` (Argument)
 `fp_orb_files`:, 33
`run_jdata[fp_style=abacus]/fp_pp_files` (Argument)
 `fp_pp_files`:, 33
`run_jdata[fp_style=abacus]/fp_pp_path` (Argument)
 `fp_pp_path`:, 33
`run_jdata[fp_style=abacus]/k_points` (Argument)
 `k_points`:, 34
`run_jdata[fp_style=abacus]/user_fp_params` (Argument)
 `user_fp_params`:, 34
`run_jdata[fp_style=amber/diff]/fp_params` (Argument)
 `fp_params`:, 34
`run_jdata[fp_style=amber/diff]/fp_params/high_level_mdin` (Argument)
 `high_level_mdin`:, 34

```

run_jdata[fp_style=amber/diff]/fp_params/low_level_jdata[fp_style=siesta]/cluster_cutoff
  (Argument)                                (Argument)
  low_level_mdin:, 34                        cluster_cutoff:, 31
run_jdata[fp_style=amber/diff]/high_level    run_jdata[fp_style=siesta]/fp_params (Argu-
  (Argument)                                ment)
  high_level:, 34                          fp_params:, 32
run_jdata[fp_style=cp2k]/external_input_path run_jdata[fp_style=siesta]/fp_params/ecut
  (Argument)                                (Argument)
  external_input_path:, 33                ecut:, 32
run_jdata[fp_style=cp2k]/ratio_failed (Argu- run_jdata[fp_style=siesta]/fp_params/ediff
  (Argument)                                (Argument)
  ratio_failed:, 33                        ediff:, 32
run_jdata[fp_style=cp2k]/user_fp_params      run_jdata[fp_style=siesta]/fp_params/kspacing
  (Argument)                                (Argument)
  user_fp_params:, 32                      kspacing:, 32
run_jdata[fp_style=gaussian]/cluster_cutoff run_jdata[fp_style=siesta]/fp_params/mixingWeight
  (Argument)                                (Argument)
  cluster_cutoff:, 30                     mixingWeight:, 32
run_jdata[fp_style=gaussian]/cluster_cutoff_hard run_jdata[fp_style=siesta]/fp_params/NumberPulay
  (Argument)                                (Argument)
  cluster_cutoff_hard:, 30                 NumberPulay:, 32
run_jdata[fp_style=gaussian]/cluster_minify  run_jdata[fp_style=siesta]/fp_pp_files (Argu-
  (Argument)                                ment)
  cluster_minify:, 30                     fp_pp_files:, 32
run_jdata[fp_style=gaussian]/fp_params (Argu- run_jdata[fp_style=siesta]/fp_pp_path (Argu-
  (Argument)                                ment)
  fp_params:, 30                          fp_pp_path:, 32
run_jdata[fp_style=gaussian]/fp_params/basis_set run_jdata[fp_style=siesta]/use_clusters
  (Argument)                                (Argument)
  basis_set:, 31                          use_clusters:, 31
run_jdata[fp_style=gaussian]/fp_params/charge run_jdata[fp_style=vasp]/cvasp (Argument)
  (Argument)                                cvasp:, 29
  charge:, 31                             run_jdata[fp_style=vasp]/fp_aniso_kspacing
run_jdata[fp_style=gaussian]/fp_params/fragment_guesses (Argument)
  (Argument)                                fp_aniso_kspacing:, 29
  fragment_guesses:, 31                   run_jdata[fp_style=vasp]/fp_incar (Argument)
run_jdata[fp_style=gaussian]/fp_params/keywords  fp_incar:, 29
  (Argument)                                run_jdata[fp_style=vasp]/fp_pp_files (Argu-
  keywords:, 30                            ment)
run_jdata[fp_style=gaussian]/fp_params/keywords_high_multiplicity:, 31
  (Argument)                                run_jdata[fp_style=vasp]/fp_pp_path (Argu-
  keywords_high_multiplicity:, 31           ment)
run_jdata[fp_style=gaussian]/fp_params/multiplicity fp_pp_path:, 29
  (Argument)                                run_jdata[fp_style=vasp]/fp_skip_bad_box (Ar-
  multiplicity:, 30                        gument)
run_jdata[fp_style=gaussian]/fp_params/nproc    fp_skip_bad_box:, 29
  (Argument)                                run_jdata[fp_style=vasp]/ratio_failed (Argu-
  nproc:, 31                              ment)
run_jdata[fp_style=gaussian]/ratio_failed      ratio_failed:, 29
  (Argument)                                run_jdata[model_devi_engine=amber]/cutoff
  ratio_failed:, 31                        (Argument)
run_jdata[fp_style=gaussian]/use_clusters      cutoff:, 27
  (Argument)                                run_jdata[model_devi_engine=amber]/disang
  use_clusters:, 30                        (Argument)

```

disang:, 28	model_devi_adapt_trust_lo:, 24
run_jdata[model_devi_engine=amber]/disang_prefix: (Argument)	run_jdata[model_devi_engine=lammps]/model_devi_clean_traj: (Argument)
disang_prefix:, 28	model_devi_clean_traj:, 25
run_jdata[model_devi_engine=amber]/low_level: (Argument)	run_jdata[model_devi_engine=lammps]/model_devi_dt: (Argument)
low_level:, 27	model_devi_dt:, 24
run_jdata[model_devi_engine=amber]/mdin: (Argument)	run_jdata[model_devi_engine=lammps]/model_devi_f_avg_relative: (Argument)
mdin:, 27	model_devi_f_avg_relative:, 25
run_jdata[model_devi_engine=amber]/mdin_prefix: (Argument)	run_jdata[model_devi_engine=lammps]/model_devi_f_trust_hi: (Argument)
mdin_prefix:, 27	model_devi_f_trust_hi:, 24
run_jdata[model_devi_engine=amber]/model_devi_f_trust_hi: (Argument)	run_jdata[model_devi_engine=lammps]/model_devi_f_trust_lo: (Argument)
model_devi_f_trust_hi:, 28	model_devi_f_trust_lo:, 24
run_jdata[model_devi_engine=amber]/model_devi_f_trust_lo: (Argument)	run_jdata[model_devi_engine=lammps]/model_devi_jobs: (Argument)
model_devi_f_trust_lo:, 28	model_devi_jobs:, 22
run_jdata[model_devi_engine=amber]/model_devi_jobs: (Argument)	run_jdata[model_devi_engine=lammps]/model_devi_jobs/ensemble: (Argument)
model_devi_jobs:, 26	ensemble:, 23
run_jdata[model_devi_engine=amber]/model_devi_jobs/data_index: (Argument)	run_jdata[model_devi_engine=lammps]/model_devi_jobs/model_devi_f_trust_hi: (Argument)
sys_idx:, 26	model_devi_f_trust_hi:, 23
run_jdata[model_devi_engine=amber]/model_devi_jobs/data_index: (Argument)	run_jdata[model_devi_engine=lammps]/model_devi_jobs/model_devi_f_trust_lo: (Argument)
trj_freq:, 27	model_devi_f_trust_lo:, 23
run_jdata[model_devi_engine=amber]/nsteps: (Argument)	run_jdata[model_devi_engine=lammps]/model_devi_jobs/model_devi_v_trust_hi: (Argument)
nsteps:, 28	model_devi_v_trust_hi:, 23
run_jdata[model_devi_engine=amber]/parm7: (Argument)	run_jdata[model_devi_engine=lammps]/model_devi_jobs/model_devi_v_trust_lo: (Argument)
parm7:, 27	model_devi_v_trust_lo:, 23
run_jdata[model_devi_engine=amber]/parm7_prefix: (Argument)	run_jdata[model_devi_engine=lammps]/model_devi_jobs/neidelay: (Argument)
parm7_prefix:, 27	neidelay:, 23
run_jdata[model_devi_engine=amber]/qm_charge: (Argument)	run_jdata[model_devi_engine=lammps]/model_devi_jobs/nsteps: (Argument)
qm_charge:, 28	nsteps:, 22
run_jdata[model_devi_engine=amber]/qm_region: (Argument)	run_jdata[model_devi_engine=lammps]/model_devi_jobs/press: (Argument)
qm_region:, 27	press:, 22
run_jdata[model_devi_engine=amber]/r: (Argument)	run_jdata[model_devi_engine=lammps]/model_devi_jobs/sys_idx: (Argument)
r:, 28	sys_idx:, 22
run_jdata[model_devi_engine=lammps]/epsilon: (Argument)	run_jdata[model_devi_engine=lammps]/model_devi_jobs/taup: (Argument)
epsilon:, 26	taup:, 23
run_jdata[model_devi_engine=lammps]/epsilon_v: (Argument)	run_jdata[model_devi_engine=lammps]/model_devi_jobs/taut: (Argument)
epsilon_v:, 26	taut:, 23
run_jdata[model_devi_engine=lammps]/model_devi_adapt_trust_lo: (Argument)	run_jdata[model_devi_engine=lammps]/model_devi_jobs/temps: (Argument)


```

    temps:, 22
run_jdata[model_devi_engine=lammps]/model_devi_job_command (Argument)
    trj_freq:, 22
run_jdata[model_devi_engine=lammps]/model_devi_merge_traj (Argument)
    model_devi_merge_traj:, 25
run_jdata[model_devi_engine=lammps]/model_devi_nopbc (Argument)
    model_devi_nopbc:, 26
run_jdata[model_devi_engine=lammps]/model_devi_numb_candi_f:, 25
run_jdata[model_devi_engine=lammps]/model_devi_numb_candi_v:, 25
run_jdata[model_devi_engine=lammps]/model_devi_perc_candi_f:, 25
run_jdata[model_devi_engine=lammps]/model_devi_perc_candi_v:, 25
run_jdata[model_devi_engine=lammps]/model_devi_skip (Argument)
    model_devi_skip:, 24
run_jdata[model_devi_engine=lammps]/model_devi_v_trust_hi:, 24
run_jdata[model_devi_engine=lammps]/model_devi_v_trust_lo:, 24
run_jdata[model_devi_engine=lammps]/shuffle_poscar (Argument)
    shuffle_poscar:, 26
run_jdata[model_devi_engine=lammps]/use_relative (Argument)
    use_relative:, 26
run_jdata[model_devi_engine=lammps]/use_relative_v (Argument)
    use_relative_v:, 26
run_jobs() (dpgen.dispatcher.Dispatcher.Dispatcher
    method), 257
run_jobs() (dpgen.dispatcher.DispatcherList.DispatcherList
    method), 260
run_md_model_devi() (in module dp-
    gen.generator.run), 275
run_mdata (Argument)
    run_mdata:, 35
run_mdata/api_version (Argument)
    api_version:, 35
run_mdata/deepmd_version (Argument)
    deepmd_version:, 35
run_mdata/fp (Argument)
    fp:, 57
run_mdata/fp/command (Argument)
    command:, 57
run_mdata/fp/machine (Argument)
    machine:, 57
run_mdata/fp/machine/batch_type (Argument)
    batch_type:, 57
run_mdata/fp/machine/clean_asynchronously
    clean_asynchronously:, 58
run_mdata/fp/machine/context_type (Argument)
    context_type:, 58
run_mdata/fp/machine/local_root (Argument)
    local_root:, 57
run_mdata/fp/machine/remote_root (Argument)
    remote_root:, 57
run_mdata/fp/machine[DpCloudServerContext]/remote_profile
    remote_profile:, 61
run_mdata/fp/machine[DpCloudServerContext]/remote_profile/
    email:, 61
run_mdata/fp/machine[DpCloudServerContext]/remote_profile/
    input_data:, 61
run_mdata/fp/machine[DpCloudServerContext]/remote_profile/
    keep_backup:, 61
run_mdata/fp/machine[DpCloudServerContext]/remote_profile/
    password:, 61
run_mdata/fp/machine[DpCloudServerContext]/remote_profile/
    program_id:, 61
run_mdata/fp/machine[HDFSContext]/remote_profile
    remote_profile:, 62
run_mdata/fp/machine[LazyLocalContext]/remote_profile
    remote_profile:, 62
run_mdata/fp/machine[LebesgueContext]/remote_profile
    remote_profile:, 58
run_mdata/fp/machine[LebesgueContext]/remote_profile/email
    email:, 58
run_mdata/fp/machine[LebesgueContext]/remote_profile/input
    input_data:, 59
run_mdata/fp/machine[LebesgueContext]/remote_profile/keep
    keep_backup:, 59
run_mdata/fp/machine[LebesgueContext]/remote_profile/passw
    password:, 58

```

```

run_mdata/fp/machine[LebesgueContext]/remote_profile/module_purge (Argument) 64
    program_id:, 58
run_mdata/fp/machine[LocalContext]/remote_profile/module_purge:, 64
    (Argument)
run_mdata/fp/machine[SSHContext]/remote_profile/module_unload_list (Argument) 61
    remote_profile:, 61
run_mdata/fp/machine[SSHContext]/remote_profile/module_unload_list:, 64
    (Argument)
run_mdata/fp/machine[SSHContext]/remote_profile/number_node (Argument) 62
    remote_profile:, 59
run_mdata/fp/machine[SSHContext]/remote_profile/number_node:, 62
    (Argument)
run_mdata/fp/machine[SSHContext]/remote_profile/para_deg (Argument) 64
    hostname:, 59
run_mdata/fp/machine[SSHContext]/remote_profile/para_deg:, 64
    (Argument)
run_mdata/fp/machine[SSHContext]/remote_profile/queue_name (Argument) 63
    key_filename:, 60
run_mdata/fp/machine[SSHContext]/remote_profile/queue_name:, 63
    (Argument)
run_mdata/fp/machine[SSHContext]/remote_profile/source_list (Argument) 64
    passphrase:, 60
run_mdata/fp/machine[SSHContext]/remote_profile/source_list:, 64
    (Argument)
run_mdata/fp/machine[SSHContext]/remote_profile/strategy (Argument) 63
    password:, 59
run_mdata/fp/machine[SSHContext]/remote_profile/strategy:, 63
    (Argument)
run_mdata/fp/machine[SSHContext]/remote_profile/strategy/if_cuda_multi_devices (Argument) 63
    password:, 59
run_mdata/fp/machine[SSHContext]/remote_profile/strategy/if_cuda_multi_devices:, 63
    (Argument)
run_mdata/fp/machine[SSHContext]/remote_profile/strategy/ratio_unfinished (Argument) 63
    port:, 59
run_mdata/fp/machine[SSHContext]/remote_profile/strategy/ratio_unfinished:, 63
    (Argument)
run_mdata/fp/machine[SSHContext]/remote_profile/strategy/wait_time (Argument) 64
    tar_compress:, 60
run_mdata/fp/machine[SSHContext]/remote_profile/strategy/wait_time:, 64
    (Argument)
run_mdata/fp/machine[SSHContext]/remote_profile/tar_compress (Argument) 67
    timeout:, 60
run_mdata/fp/machine[SSHContext]/remote_profile/tar_compress:, 67
    (Argument)
run_mdata/fp/machine[SSHContext]/remote_profile/tar_compress/kwargs (Argument) 66
    totp_secret:, 60
run_mdata/fp/machine[SSHContext]/remote_profile/tar_compress/kwargs:, 66
    (Argument)
run_mdata/fp/machine[SSHContext]/remote_profile/tar_compress/kwargs/kwargs (Argument) 67
    username:, 59
run_mdata/fp/machine[SSHContext]/remote_profile/tar_compress/kwargs/kwargs:, 67
    (Argument)
run_mdata/fp/resources (Argument) 62
    resources:, 62
run_mdata/fp/resources/batch_type (Argument) 65
    batch_type:, 65
run_mdata/fp/resources/cpu_per_node (Argument) 62
    cpu_per_node:, 62
run_mdata/fp/resources/custom_flags (Argument) 63
    custom_flags:, 63
run_mdata/fp/resources/envs (Argument) 64
    envs:, 64
run_mdata/fp/resources/gpu_per_node (Argument) 62
    gpu_per_node:, 62
run_mdata/fp/resources/group_size (Argument) 63
    group_size:, 63
run_mdata/fp/resources/module_list (Argument) 64
    run_mdata/fp/resources/module_purge (Argument) 64
    run_mdata/fp/resources/module_unload_list (Argument) 61
    run_mdata/fp/resources/number_node (Argument) 62
    run_mdata/fp/resources/para_deg (Argument) 64
    run_mdata/fp/resources/queue_name (Argument) 63
    run_mdata/fp/resources/source_list (Argument) 64
    run_mdata/fp/resources/strategy (Argument) 63
    run_mdata/fp/resources/strategy/if_cuda_multi_devices (Argument) 63
    run_mdata/fp/resources/strategy/ratio_unfinished (Argument) 63
    run_mdata/fp/resources/strategy/wait_time (Argument) 64
    run_mdata/fp/resources[DistributedShell]/kwargs (Argument) 66
    run_mdata/fp/resources[DpCloudServer]/kwargs (Argument) 66
    run_mdata/fp/resources[Lebesgue]/kwargs (Argument) 66
    run_mdata/fp/resources[LSF]/kwargs (Argument) 66
    run_mdata/fp/resources[LSF]/kwargs/custom_gpu_line (Argument) 67
    run_mdata/fp/resources[LSF]/kwargs/gpu_exclusive (Argument) 67
    run_mdata/fp/resources[LSF]/kwargs/gpu_new_syntax (Argument) 66
    run_mdata/fp/resources[LSF]/kwargs/gpu_usage (Argument) 66
    run_mdata/fp/resources[PBS]/kwargs (Argument) 65
    run_mdata/fp/resources[Shell]/kwargs (Argument) 65

```

```

run_mdata/fp/resources[SlurmJobArray]/kwargs
    (Argument)
    kwargs:., 66
run_mdata/fp/resources[SlurmJobArray]/kwargs/custom_program_line, 50
    (Argument)
    custom_gpu_line:., 66
run_mdata/fp/resources[Slurm]/kwargs (Argument)
    kwargs:., 65
run_mdata/fp/resources[Slurm]/kwargs/custom_gpu_line, 51
    (Argument)
    custom_gpu_line:., 65
run_mdata/fp/resources[Torque]/kwargs (Argument)
    kwargs:., 65
run_mdata/fp/user_backward_files (Argument)
    user_backward_files:., 67
run_mdata/fp/user_forward_files (Argument)
    user_forward_files:., 67
run_mdata/model_devi (Argument)
    model_devi:., 45
run_mdata/model_devi/command (Argument)
    command:., 45
run_mdata/model_devi/machine (Argument)
    machine:., 46
run_mdata/model_devi/machine/batch_type
    (Argument)
    batch_type:., 46
run_mdata/model_devi/machine/clean_asynchronously
    (Argument)
    clean_asynchronously:., 46
run_mdata/model_devi/machine/context_type
    (Argument)
    context_type:., 46
run_mdata/model_devi/machine/local_root
    (Argument)
    local_root:., 46
run_mdata/model_devi/machine/remote_root (Argument)
    remote_root:., 46
run_mdata/model_devi/machine[DpCloudServerContext]
    (Argument)
    remote_profile:., 50
run_mdata/model_devi/machine[DpCloudServerContext]
    (Argument)
    email:., 50
run_mdata/model_devi/machine[DpCloudServerContext]
    (Argument)
    input_data:., 51
run_mdata/model_devi/machine[DpCloudServerContext]
    (Argument)
    keep_backup:., 50
run_mdata/model_devi/machine[DpCloudServerContext]
    (Argument)
    password:., 50
run_mdata/model_devi/machine[DpCloudServerContext]
    (Argument)
    remote_profile:., 51
run_mdata/model_devi/machine[HDFSContext]
    (Argument)
    remote_profile:., 51
run_mdata/model_devi/machine[LazyLocalContext]
    (Argument)
    remote_profile:., 51
run_mdata/model_devi/machine[LebesgueContext]
    (Argument)
    remote_profile:., 47
run_mdata/model_devi/machine[LebesgueContext]
    (Argument)
    email:., 47
run_mdata/model_devi/machine[LebesgueContext]
    (Argument)
    input_data:., 47
run_mdata/model_devi/machine[LebesgueContext]
    (Argument)
    keep_backup:., 47
run_mdata/model_devi/machine[LebesgueContext]
    (Argument)
    password:., 47
run_mdata/model_devi/machine[LebesgueContext]
    (Argument)
    program_id:., 47
run_mdata/model_devi/machine[LocalContext]
    (Argument)
    remote_profile:., 49
run_mdata/model_devi/machine[SSHContext]
    (Argument)
    remote_profile:., 48
run_mdata/model_devi/machine[SSHContext]
    (Argument)
    hostname:., 48
run_mdata/model_devi/machine[SSHContext]
    (Argument)
    key_filename:., 48
run_mdata/model_devi/machine[SSHContext]
    (Argument)
    passphrase:., 49
run_mdata/model_devi/machine[SSHContext]
    (Argument)
    password:., 48
run_mdata/model_devi/machine[SSHContext]
    (Argument)
    port:., 48
run_mdata/model_devi/machine[SSHContext]
    (Argument)
    tar_compress:., 49
run_mdata/model_devi/machine[SSHContext]
    (Argument)
    username:., 49

```



```

    timeout:, 49
run_mdata/model_devi/machine[SSHContext]/remote_profile/totp_secret
    (Argument)
    totp_secret:, 49
run_mdata/model_devi/machine[SSHContext]/remote_profile/username
    (Argument)
    username:, 48
run_mdata/model_devi/resources (Argument)
    resources:, 51
run_mdata/model_devi/resources/batch_type
    (Argument)
    batch_type:, 54
run_mdata/model_devi/resources/cpu_per_node
    (Argument)
    cpu_per_node:, 51
run_mdata/model_devi/resources/custom_flags
    (Argument)
    custom_flags:, 52
run_mdata/model_devi/resources/envs (Argument)
    envs:, 53
run_mdata/model_devi/resources/gpu_per_node
    (Argument)
    gpu_per_node:, 51
run_mdata/model_devi/resources/group_size
    (Argument)
    group_size:, 52
run_mdata/model_devi/resources/module_list
    (Argument)
    module_list:, 53
run_mdata/model_devi/resources/module_purge
    (Argument)
    module_purge:, 53
run_mdata/model_devi/resources/module_unload_list
    (Argument)
    module_unload_list:, 53
run_mdata/model_devi/resources/number_node
    (Argument)
    number_node:, 51
run_mdata/model_devi/resources/para_deg
    (Argument)
    para_deg:, 53
run_mdata/model_devi/resources/queue_name
    (Argument)
    queue_name:, 52
run_mdata/model_devi/resources/source_list
    (Argument)
    source_list:, 53
run_mdata/model_devi/resources/strategy
    (Argument)
    strategy:, 52
run_mdata/model_devi/resources/strategy/if_cuda_multi_devices
    (Argument)
    if_cuda_multi_devices:, 52
run_mdata/model_devi/resources/strategy/ratio_unfinished
    (Argument)
    ratio_unfinished:, 52
run_mdata/model_devi/resources/wait_time (Argument)
    wait_time:, 54
run_mdata/model_devi/resources[DistributedShell]/kwargs
    (Argument)
    kwargs:, 56
run_mdata/model_devi/resources[DpCloudServer]/kwargs
    (Argument)
    kwargs:, 55
run_mdata/model_devi/resources[Lebesgue]/kwargs
    (Argument)
    kwargs:, 56
run_mdata/model_devi/resources[LSF]/kwargs
    (Argument)
    kwargs:, 55
run_mdata/model_devi/resources[LSF]/kwargs/custom_gpu_line
    (Argument)
    custom_gpu_line:, 56
run_mdata/model_devi/resources[LSF]/kwargs/gpu_exclusive
    (Argument)
    gpu_exclusive:, 56
run_mdata/model_devi/resources[LSF]/kwargs/gpu_new_syntax
    (Argument)
    gpu_new_syntax:, 56
run_mdata/model_devi/resources[LSF]/kwargs/gpu_usage
    (Argument)
    gpu_usage:, 55
run_mdata/model_devi/resources[PBS]/kwargs
    (Argument)
    kwargs:, 54
run_mdata/model_devi/resources[Shell]/kwargs
    (Argument)
    kwargs:, 54
run_mdata/model_devi/resources[SlurmJobArray]/kwargs
    (Argument)
    kwargs:, 55
run_mdata/model_devi/resources[SlurmJobArray]/kwargs/custom_gpu_line
    (Argument)
    custom_gpu_line:, 55
run_mdata/model_devi/resources[Slurm]/kwargs
    (Argument)
    kwargs:, 54
run_mdata/model_devi/resources[Slurm]/kwargs/custom_gpu_line
    (Argument)
    custom_gpu_line:, 54
run_mdata/model_devi/resources[Torque]/kwargs
    (Argument)
    kwargs:, 55
run_mdata/model_devi/user_backward_files (Argument)
    user_backward_files:, 57

```

```

run_mdata/model_devi/user_forward_files
    (Argument)
    user_forward_files:, 56
run_mdata/train (Argument)
    train:, 35
run_mdata/train/command (Argument)
    command:, 35
run_mdata/train/machine (Argument)
    machine:, 35
run_mdata/train/machine/batch_type (Argument)
    batch_type:, 35
run_mdata/train/machine/clean_asynchronously
    (Argument)
    clean_asynchronously:, 36
run_mdata/train/machine/context_type (Argument)
    context_type:, 36
run_mdata/train/machine/local_root (Argument)
    local_root:, 35
run_mdata/train/machine/remote_root (Argument)
    remote_root:, 35
run_mdata/train/machine[DpCloudServerContext]/remote_profile
    (Argument)
    remote_profile:, 39
run_mdata/train/machine[DpCloudServerContext]/remote_profile/email
    (Argument)
    email:, 39
run_mdata/train/machine[DpCloudServerContext]/remote_profile/input_data
    (Argument)
    input_data:, 39
run_mdata/train/machine[DpCloudServerContext]/remote_profile/keep_backup
    (Argument)
    keep_backup:, 39
run_mdata/train/machine[DpCloudServerContext]/remote_profile/password
    (Argument)
    password:, 39
run_mdata/train/machine[DpCloudServerContext]/remote_profile/program_id
    (Argument)
    program_id:, 39
run_mdata/train/machine[HDFSContext]/remote_profile
    (Argument)
    remote_profile:, 40
run_mdata/train/machine[LazyLocalContext]/remote_profile
    (Argument)
    remote_profile:, 40
run_mdata/train/machine[LebesgueContext]/remote_profile
    (Argument)
    remote_profile:, 36
run_mdata/train/machine[LebesgueContext]/remote_profile/email
    (Argument)
    email:, 36
run_mdata/train/machine[LebesgueContext]/remote_profile/input_data
    (Argument)
    input_data:, 37
run_mdata/train/machine[LebesgueContext]/remote_profile/keep_backup
    (Argument)
    keep_backup:, 37
run_mdata/train/machine[LebesgueContext]/remote_profile/password
    (Argument)
    password:, 36
run_mdata/train/machine[LebesgueContext]/remote_profile/program_id
    (Argument)
    program_id:, 36
run_mdata/train/machine[LocalContext]/remote_profile
    (Argument)
    remote_profile:, 38
run_mdata/train/machine[SSHContext]/remote_profile
    (Argument)
    remote_profile:, 37
run_mdata/train/machine[SSHContext]/remote_profile/hostname
    (Argument)
    hostname:, 37
run_mdata/train/machine[SSHContext]/remote_profile/key_filename
    (Argument)
    key_filename:, 38
run_mdata/train/machine[SSHContext]/remote_profile/passphrase
    (Argument)
    passphrase:, 38
run_mdata/train/machine[SSHContext]/remote_profile/password
    (Argument)
    password:, 37
run_mdata/train/machine[SSHContext]/remote_profile/port
    (Argument)
    port:, 37
run_mdata/train/machine[SSHContext]/remote_profile/tar_compress
    (Argument)
    tar_compress:, 38
run_mdata/train/machine[SSHContext]/remote_profile/timeout
    (Argument)
    timeout:, 38
run_mdata/train/machine[SSHContext]/remote_profile/totp_secret
    (Argument)
    totp_secret:, 38
run_mdata/train/machine[SSHContext]/remote_profile/username
    (Argument)
    username:, 37
run_mdata/train/resources (Argument)
    resources:, 40
run_mdata/train/resources/batch_type (Argument)
    batch_type:, 42
run_mdata/train/resources/cpu_per_node (Argument)
    cpu_per_node:, 40
run_mdata/train/resources/custom_flags (Argument)
    custom_flags:, 41

```

```

run_mdata/train/resources/envs (Argument)
    envs:, 42
run_mdata/train/resources/gpu_per_node (Argument)
    gpu_per_node:, 40
run_mdata/train/resources/group_size (Argument)
    group_size:, 40
run_mdata/train/resources/module_list (Argument)
    module_list:, 42
run_mdata/train/resources/module_purge (Argument)
    module_purge:, 42
run_mdata/train/resources/module_unload_list (Argument)
    module_unload_list:, 42
run_mdata/train/resources/number_node (Argument)
    number_node:, 40
run_mdata/train/resources/para_deg (Argument)
    para_deg:, 41
run_mdata/train/resources/queue_name (Argument)
    queue_name:, 40
run_mdata/train/resources/source_list (Argument)
    source_list:, 41
run_mdata/train/resources/strategy (Argument)
    strategy:, 41
run_mdata/train/resources/strategy/if_cuda_multi_devices (Argument)
    if_cuda_multi_devices:, 41
run_mdata/train/resources/strategy/ratio_unfinished (Argument)
    ratio_unfinished:, 41
run_mdata/train/resources/wait_time (Argument)
    wait_time:, 42
run_mdata/train/resources[DistributedShell]/kwargs (Argument)
    kwargs:, 45
run_mdata/train/resources[DpCloudServer]/kwargs (Argument)
    kwargs:, 44
run_mdata/train/resources[Lebesgue]/kwargs (Argument)
    kwargs:, 45
run_mdata/train/resources[LSF]/kwargs (Argument)
    kwargs:, 44
run_mdata/train/resources[LSF]/kwargs/custom_gpu_line (Argument)
    custom_gpu_line:, 45
run_mdata/train/resources[LSF]/kwargs/gpu_exclusive (Argument)
    gpu_exclusive:, 44
run_mdata/train/resources[LSF]/kwargs/gpu_new_syntax (Argument)
    gpu_new_syntax:, 44
run_mdata/train/resources[LSF]/kwargs/gpu_usage (Argument)
    gpu_usage:, 44
run_mdata/train/resources[PBS]/kwargs (Argument)
    kwargs:, 43
run_mdata/train/resources[Shell]/kwargs (Argument)
    kwargs:, 43
run_mdata/train/resources[SlurmJobArray]/kwargs (Argument)
    kwargs:, 43
run_mdata/train/resources[SlurmJobArray]/kwargs/custom_gpu_line (Argument)
    custom_gpu_line:, 44
run_mdata/train/resources[Slurm]/kwargs (Argument)
    kwargs:, 43
run_mdata/train/resources[Slurm]/kwargs/custom_gpu_line (Argument)
    custom_gpu_line:, 43
run_mdata/train/resources[Torque]/kwargs (Argument)
    kwargs:, 43
run_mdata/train/user_backward_files (Argument)
    user_backward_files:, 45
run_mdata/train/user_forward_files (Argument)
    user_forward_files:, 45
run_mdata:
    run_mdata (Argument), 35
    run_mdata_arginfo() (in module dp-gen.generator.arginfo), 272
    run_model_devi() (in module dp-gen.generator.run), 275
    run_model_devi() (in module dp-gen.simplify.simplify), 281
    run_property() (in module dp-gen.auto_test.common_prop), 245
    run_reaxff() (in module dp-gen.data.reaction), 251
    run_report() (in module dp-gen.tools.run_report), 284
    run_report() (in module dp-gen.tools.stat_sys), 284
    run_task() (in module dp-gen.auto_test.run), 246
    run_train() (in module dp-gen.generator.run), 275
    run_vasp_md() (in module dp-gen.data.gen), 250
    run_vasp_relax() (in module dp-gen.data.gen), 250
    run_vasp_relax() (in module dp-gen.data.surf), 252

```

running (*dpgen.auto_test.lib.BatchJob.JobStatus* attribute), 220
 running (*dpgen.auto_test.lib.RemoteJob.JobStatus* attribute), 220
 running (*dpgen.dispatcher.JobStatus.JobStatus* attribute), 260
 running (*dpgen.remote.RemoteJob.JobStatus* attribute), 276
 runvasp() (in module *dpgen.generator.lib.cvasp*), 268

S

save() (*dpgen.generator.lib.ele_temp.NBandsEsti* method), 268
 sc() (in module *dpgen.auto_test.lib.crys*), 223
 scan_files() (in module *dp-gen.tools.auto_gen_param*), 283
 scan_outcar_file() (in module *dp-gen.data.tools.cessp2force_lin*), 247
 sepline() (in module *dpgen.util*), 285
 set_atoms_typeids() (in module *dp-gen.data.tools.io_lammps*), 248
 set_atoms_typeids_with_atomic_numbers() (in module *dpgen.data.tools.io_lammps*), 248
 set_inter_type_func() (*dp-gen.auto_test.Lammps.Lammps* method), 238
 set_model_param() (*dpgen.auto_test.Lammps.Lammps* method), 238
 set_version() (in module *dpgen.generator.run*), 275
 sftp (*dpgen.dispatcher.SSHContext.SSHContext* property), 265
 sftp (*dpgen.dispatcher.SSHContext.SSHSession* property), 265
 Shell (class in *dpgen.dispatcher.Shell*), 266
 shuffle_poscar:
 run_jdata[model_devi_engine=lammps]/shuffle_poscar (Argument), 26
 shuffle_stru_data() (in module *dpgen.data.gen*), 250
 simplify_jdata (Argument)
 simplify_jdata:, 135
 simplify_jdata/default_training_param (Argument)
 default_training_param:, 138
 simplify_jdata/dp_compress (Argument)
 dp_compress:, 138
 simplify_jdata/fp_accurate_soft_threshold (Argument)
 fp_accurate_soft_threshold:, 139
 simplify_jdata/fp_accurate_threshold (Argument)
 fp_accurate_threshold:, 139
 simplify_jdata/fp_style (Argument)
 fp_style:, 139
 simplify_jdata/fp_task_max (Argument)

 fp_task_max:, 139
 simplify_jdata/fp_task_min (Argument)
 fp_task_min:, 139
 simplify_jdata/init_batch_size (Argument)
 init_batch_size:, 136
 simplify_jdata/init_data_prefix (Argument)
 init_data_prefix:, 136
 simplify_jdata/init_data_sys (Argument)
 init_data_sys:, 136
 simplify_jdata/init_pick_number (Argument)
 init_pick_number:, 137
 simplify_jdata/iter_pick_number (Argument)
 iter_pick_number:, 137
 simplify_jdata/labeled (Argument)
 labeled:, 137
 simplify_jdata/mass_map (Argument)
 mass_map:, 135
 simplify_jdata/model_devi_activation_func (Argument)
 model_devi_activation_func:, 139
 simplify_jdata/model_devi_f_trust_hi (Argument)
 model_devi_f_trust_hi:, 137
 simplify_jdata/model_devi_f_trust_lo (Argument)
 model_devi_f_trust_lo:, 137
 simplify_jdata/numb_models (Argument)
 numb_models:, 137
 simplify_jdata/pick_data (Argument)
 pick_data:, 137
 simplify_jdata/sys_batch_size (Argument)
 sys_batch_size:, 137
 simplify_jdata/sys_configs (Argument)
 sys_configs:, 136
 simplify_jdata/sys_configs_prefix (Argument)
 sys_configs_prefix:, 136
 simplify_jdata/sys_format (Argument)
 sys_format:, 136
 simplify_jdata/training_init_model (Argument)
 training_init_model:, 137
 simplify_jdata/training_iter0_model_path (Argument)
 training_iter0_model_path:, 137
 simplify_jdata/training_reuse_iter (Argument)
 training_reuse_iter:, 138
 simplify_jdata/training_reuse_numb_steps (Argument)
 training_reuse_numb_steps:, 138
 simplify_jdata/training_reuse_old_ratio (Argument)
 training_reuse_old_ratio:, 138
 simplify_jdata/training_reuse_start_lr (Argument)
 training_reuse_start_lr:, 138

`simplify_jdata/training_reuse_start_pref_e` (*Argument*)
`training_reuse_start_pref_e`: [138](#)
`simplify_jdata/training_reuse_start_pref_f` (*Argument*)
`training_reuse_start_pref_f`: [139](#)
`simplify_jdata/type_map` (*Argument*)
`type_map`: [135](#)
`simplify_jdata/use_ele_temp` (*Argument*)
`use_ele_temp`: [136](#)
`simplify_jdata`:
`simplify_jdata` (*Argument*), [135](#)
`simplify_jdata_arginfo()` (*in module dp-gen.simplify_arginfo*), [280](#)
`simplify_jdata[gaussian]/cluster_cutoff` (*Argument*)
`cluster_cutoff`: [140](#)
`simplify_jdata[gaussian]/cluster_cutoff_hard` (*Argument*)
`cluster_cutoff_hard`: [141](#)
`simplify_jdata[gaussian]/cluster_minify` (*Argument*)
`cluster_minify`: [141](#)
`simplify_jdata[gaussian]/fp_params` (*Argument*)
`fp_params`: [141](#)
`simplify_jdata[gaussian]/fp_params/basis_set` (*Argument*)
`basis_set`: [142](#)
`simplify_jdata[gaussian]/fp_params/charge` (*Argument*)
`charge`: [141](#)
`simplify_jdata[gaussian]/fp_params/fragment_guesses` (*Argument*)
`fragment_guesses`: [142](#)
`simplify_jdata[gaussian]/fp_params/keywords` (*Argument*)
`keywords`: [141](#)
`simplify_jdata[gaussian]/fp_params/keywords_high_multiplicity` (*Argument*)
`keywords_high_multiplicity`: [142](#)
`simplify_jdata[gaussian]/fp_params/multiplicity` (*Argument*)
`multiplicity`: [141](#)
`simplify_jdata[gaussian]/fp_params/nproc` (*Argument*)
`nproc`: [141](#)
`simplify_jdata[gaussian]/ratio_failed` (*Argument*)
`ratio_failed`: [142](#)
`simplify_jdata[gaussian]/use_clusters` (*Argument*)
`use_clusters`: [140](#)
`simplify_jdata[vasp]/cvasp` (*Argument*)
`cvasp`: [140](#)
`simplify_jdata[vasp]/fp_aniso_kspacing` (*Argument*)
`fp_aniso_kspacing`: [140](#)
`simplify_jdata[vasp]/fp_incar` (*Argument*)
`fp_incar`: [140](#)
`simplify_jdata[vasp]/fp_pp_files` (*Argument*)
`fp_pp_files`: [140](#)
`simplify_jdata[vasp]/fp_pp_path` (*Argument*)
`fp_pp_path`: [139](#)
`simplify_jdata[vasp]/fp_skip_bad_box` (*Argument*)
`fp_skip_bad_box`: [140](#)
`simplify_jdata[vasp]/ratio_failed` (*Argument*)
`ratio_failed`: [140](#)
`simplify_mdata` (*Argument*)
`simplify_mdata`: [142](#)
`simplify_mdata/api_version` (*Argument*)
`api_version`: [142](#)
`simplify_mdata/deepmd_version` (*Argument*)
`deepmd_version`: [142](#)
`simplify_mdata/fp` (*Argument*)
`fp`: [165](#)
`simplify_mdata/fp/command` (*Argument*)
`command`: [165](#)
`simplify_mdata/fp/machine` (*Argument*)
`machine`: [165](#)
`simplify_mdata/fp/machine/batch_type` (*Argument*)
`batch_type`: [166](#)
`simplify_mdata/fp/machine/clean_asynchronously` (*Argument*)
`clean_asynchronously`: [166](#)
`simplify_mdata/fp/machine/context_type` (*Argument*)
`context_type`: [166](#)
`simplify_mdata/fp/machine/local_root` (*Argument*)
`local_root`: [166](#)
`simplify_mdata/fp/machine/remote_root` (*Argument*)
`remote_root`: [166](#)
`simplify_mdata/fp/machine[DpCloudServerContext]/remote_profile` (*Argument*)
`remote_profile`: [169](#)
`simplify_mdata/fp/machine[DpCloudServerContext]/remote_profile_email` (*Argument*)
`email`: [169](#)
`simplify_mdata/fp/machine[DpCloudServerContext]/remote_profile_input_data` (*Argument*)
`input_data`: [170](#)
`simplify_mdata/fp/machine[DpCloudServerContext]/remote_profile_keep_backup` (*Argument*)
`keep_backup`: [170](#)


```

simplify_mdata/fp/machine[DpCloudServerContext]/remote_profile/timeout (Argument)
password:, 170 timeout:, 169
simplify_mdata/fp/machine[DpCloudServerContext]/remote_profile/totp_secret (Argument)
program_id:, 170 totp_secret:, 169
simplify_mdata/fp/machine[HDFSContext]/remote_profile/username (Argument)
remote_profile:, 171 username:, 168
simplify_mdata/fp/machine[LazyLocalContext]/remote_profile/resources (Argument)
remote_profile:, 170 resources:, 171
simplify_mdata/fp/machine[LebesgueContext]/remote_profile/batch_type (Argument)
remote_profile:, 167 batch_type:, 173
simplify_mdata/fp/machine[LebesgueContext]/remote_profile/cpu_per_node (Argument)
remote_profile:, 167 cpu_per_node:, 171
simplify_mdata/fp/machine[LebesgueContext]/remote_profile/custom_flags (Argument)
remote_profile:, 167 custom_flags:, 171
simplify_mdata/fp/machine[LebesgueContext]/remote_profile/envs (Argument)
remote_profile:, 167 envs:, 171
simplify_mdata/fp/machine[LebesgueContext]/remote_profile/keep_backup (Argument)
remote_profile:, 167 keep_backup:, 167
simplify_mdata/fp/machine[LebesgueContext]/remote_profile/group_size (Argument)
remote_profile:, 167 group_size:, 171
simplify_mdata/fp/machine[LebesgueContext]/remote_profile/module_list (Argument)
remote_profile:, 167 module_list:, 173
simplify_mdata/fp/machine[LocalContext]/remote_profile/module_purge (Argument)
remote_profile:, 169 module_purge:, 172
simplify_mdata/fp/machine[SSHContext]/remote_profile/module_unload_list (Argument)
remote_profile:, 167 module_unload_list:, 173
simplify_mdata/fp/machine[SSHContext]/remote_profile/hostname (Argument)
hostname:, 168
simplify_mdata/fp/machine[SSHContext]/remote_profile/key_filename (Argument)
key_filename:, 168
simplify_mdata/fp/machine[SSHContext]/remote_profile/passphrase (Argument)
passphrase:, 168
simplify_mdata/fp/machine[SSHContext]/remote_profile/queue_name (Argument)
password:, 168 queue_name:, 171
simplify_mdata/fp/machine[SSHContext]/remote_profile/source_list (Argument)
password:, 168 source_list:, 172
simplify_mdata/fp/machine[SSHContext]/remote_profile/strategy (Argument)
port:, 168 strategy:, 172
simplify_mdata/fp/machine[SSHContext]/remote_profile/strategy/if_cuda_multi_devices (Argument)
tar_compress:, 169

```

<code>if_cuda_multi_devices:</code> , 172	<code>user_backward_files:</code> , 176
<code>simplify_mdata/fp/resources/strategy/ratio_unfinished</code> (Argument)	<code>simplify_mdata/fp/user_forward_files</code> (Argument)
<code>ratio_unfinished:</code> , 172	<code>user_forward_files:</code> , 176
<code>simplify_mdata/fp/resources/wait_time</code> (Argument)	<code>simplify_mdata/model_devi</code> (Argument)
<code>wait_time:</code> , 173	<code>model_devi:</code> , 154
<code>simplify_mdata/fp/resources[DistributedShell]/kwargs</code> (Argument)	<code>simplify_mdata/model_devi/command</code> (Argument)
<code>kwargs:</code> , 176	<code>command:</code> , 154
<code>simplify_mdata/fp/resources[DpCloudServer]/kwargs</code> (Argument)	<code>simplify_mdata/model_devi/machine</code> (Argument)
<code>kwargs:</code> , 175	<code>machine:</code> , 154
<code>simplify_mdata/fp/resources[Lebesgue]/kwargs</code> (Argument)	<code>simplify_mdata/model_devi/machine/batch_type</code> (Argument)
<code>kwargs:</code> , 176	<code>batch_type:</code> , 154
<code>simplify_mdata/fp/resources[LSF]/kwargs</code> (Argument)	<code>simplify_mdata/model_devi/machine/clean_asynchronously</code> (Argument)
<code>kwargs:</code> , 175	<code>clean_asynchronously:</code> , 155
<code>simplify_mdata/fp/resources[LSF]/kwargs</code> (Argument)	<code>simplify_mdata/model_devi/machine/context_type</code> (Argument)
<code>kwargs:</code> , 175	<code>context_type:</code> , 155
<code>simplify_mdata/fp/resources[LSF]/kwargs/custom_gpu_line</code> (Argument)	<code>simplify_mdata/model_devi/machine/local_root</code> (Argument)
<code>custom_gpu_line:</code> , 175	<code>local_root:</code> , 154
<code>simplify_mdata/fp/resources[LSF]/kwargs/gpu_exclusive</code> (Argument)	<code>simplify_mdata/model_devi/machine/remote_root</code> (Argument)
<code>gpu_exclusive:</code> , 175	<code>remote_root:</code> , 154
<code>simplify_mdata/fp/resources[LSF]/kwargs/gpu_new_syntax</code> (Argument)	<code>simplify_mdata/model_devi/machine[DpCloudServerContext]/remote_profile</code> (Argument)
<code>gpu_new_syntax:</code> , 175	<code>remote_profile:</code> , 158
<code>simplify_mdata/fp/resources[LSF]/kwargs/gpu_usage</code> (Argument)	<code>simplify_mdata/model_devi/machine[DpCloudServerContext]/remote_email</code> (Argument)
<code>gpu_usage:</code> , 175	<code>email:</code> , 158
<code>simplify_mdata/fp/resources[PBS]/kwargs</code> (Argument)	<code>simplify_mdata/model_devi/machine[DpCloudServerContext]/remote_input_data</code> (Argument)
<code>kwargs:</code> , 174	<code>input_data:</code> , 159
<code>simplify_mdata/fp/resources[Shell]/kwargs</code> (Argument)	<code>simplify_mdata/model_devi/machine[DpCloudServerContext]/remote_keep_backup</code> (Argument)
<code>kwargs:</code> , 173	<code>keep_backup:</code> , 159
<code>simplify_mdata/fp/resources[SlurmJobArray]/kwargs</code> (Argument)	<code>simplify_mdata/model_devi/machine[DpCloudServerContext]/remote_password</code> (Argument)
<code>kwargs:</code> , 174	<code>password:</code> , 158
<code>simplify_mdata/fp/resources[SlurmJobArray]/kwargs/custom_gpu_line</code> (Argument)	<code>simplify_mdata/model_devi/machine[DpCloudServerContext]/remote_program_id</code> (Argument)
<code>custom_gpu_line:</code> , 174	<code>program_id:</code> , 159
<code>simplify_mdata/fp/resources[Slurm]/kwargs</code> (Argument)	<code>simplify_mdata/model_devi/machine[HDFSContext]/remote_profile</code> (Argument)
<code>kwargs:</code> , 174	<code>remote_profile:</code> , 159
<code>simplify_mdata/fp/resources[Slurm]/kwargs/custom_gpu_line</code> (Argument)	<code>simplify_mdata/model_devi/machine[LazyLocalContext]/remote_profile</code> (Argument)
<code>custom_gpu_line:</code> , 174	<code>remote_profile:</code> , 159
<code>simplify_mdata/fp/resources[Torque]/kwargs</code> (Argument)	<code>simplify_mdata/model_devi/machine[LebesgueContext]/remote_profile</code> (Argument)
<code>kwargs:</code> , 174	<code>remote_profile:</code> , 155
<code>simplify_mdata/fp/user_backward_files</code> (Argument)	<code>simplify_mdata/model_devi/machine[LebesgueContext]/remote_profile</code> (Argument)

```

    email:, 155
simplify_mdata/model_devi/machine[LebesgueContext]/simplify_mdata/model_devi/resources/custom_flags
    (Argument)
    input_data:, 156
simplify_mdata/model_devi/machine[LebesgueContext]/simplify_mdata/model_devi/resources/envs (Ar-
    (Argument)
    keep_backup:, 156
simplify_mdata/model_devi/machine[LebesgueContext]/simplify_mdata/model_devi/resources/gpu_per_node
    (Argument)
    password:, 155
simplify_mdata/model_devi/machine[LebesgueContext]/simplify_mdata/model_devi/resources/group_size
    (Argument)
    program_id:, 155
simplify_mdata/model_devi/machine[LocalContext]/simplify_mdata/model_devi/resources/module_list
    (Argument)
    remote_profile:, 158
simplify_mdata/model_devi/machine[SSHContext]/simplify_mdata/model_devi/resources/module_purge
    (Argument)
    remote_profile:, 156
simplify_mdata/model_devi/machine[SSHContext]/simplify_mdata/model_devi/resources/module_unload_list
    (Argument)
    hostname:, 156
simplify_mdata/model_devi/machine[SSHContext]/simplify_mdata/model_devi/resources/number_node
    (Argument)
    key_filename:, 157
simplify_mdata/model_devi/machine[SSHContext]/simplify_mdata/model_devi/resources/para_deg
    (Argument)
    passphrase:, 157
simplify_mdata/model_devi/machine[SSHContext]/simplify_mdata/model_devi/resources/queue_name
    (Argument)
    password:, 156
simplify_mdata/model_devi/machine[SSHContext]/simplify_mdata/model_devi/resources/source_list
    (Argument)
    port:, 157
simplify_mdata/model_devi/machine[SSHContext]/simplify_mdata/model_devi/resources/strategy
    (Argument)
    tar_compress:, 158
simplify_mdata/model_devi/machine[SSHContext]/simplify_mdata/model_devi/resources/strategy/if_cuda_multi
    (Argument)
    timeout:, 157
simplify_mdata/model_devi/machine[SSHContext]/simplify_mdata/model_devi/resources/strategy/ratio_unfinis
    (Argument)
    totp_secret:, 157
simplify_mdata/model_devi/machine[SSHContext]/simplify_mdata/model_devi/resources/wait_time
    (Argument)
    username:, 156
simplify_mdata/model_devi/resources (Argu- simplify_mdata/model_devi/resources[DistributedShell]/kwar
    ment)
    resources:, 160
simplify_mdata/model_devi/resources/batch_typesimplify_mdata/model_devi/resources[DpCloudServer]/kwargs
    (Argument)
    batch_type:, 162
simplify_mdata/model_devi/resources/cpu_per_node (Argument)
    (Argument)
    cpu_per_node:, 160
simplify_mdata/model_devi/resources/custom_flags
    (Argument)
    custom_flags:, 160
simplify_mdata/model_devi/resources/envs (Ar-
    (Argument)
    envs:, 162
simplify_mdata/model_devi/resources/gpu_per_node
    (Argument)
    gpu_per_node:, 160
simplify_mdata/model_devi/resources/group_size
    (Argument)
    group_size:, 160
simplify_mdata/model_devi/resources/module_list
    (Argument)
    module_list:, 162
simplify_mdata/model_devi/resources/module_purge
    (Argument)
    module_purge:, 161
simplify_mdata/model_devi/resources/module_unload_list
    (Argument)
    module_unload_list:, 162
simplify_mdata/model_devi/resources/number_node
    (Argument)
    number_node:, 160
simplify_mdata/model_devi/resources/para_deg
    (Argument)
    para_deg:, 161
simplify_mdata/model_devi/resources/queue_name
    (Argument)
    queue_name:, 160
simplify_mdata/model_devi/resources/source_list
    (Argument)
    source_list:, 161
simplify_mdata/model_devi/resources/strategy
    (Argument)
    strategy:, 160
simplify_mdata/model_devi/resources/strategy/if_cuda_multi
    (Argument)
    if_cuda_multi_devices:, 161
simplify_mdata/model_devi/resources/strategy/ratio_unfinis
    (Argument)
    ratio_unfinished:, 161
simplify_mdata/model_devi/resources/wait_time
    (Argument)
    wait_time:, 162
simplify_mdata/model_devi/resources[DistributedShell]/kwar
    (Argument)
    kwargs:, 165
simplify_mdata/model_devi/resources[DpCloudServer]/kwargs
    (Argument)
    kwargs:, 164
simplify_mdata/model_devi/resources[Lebesgue]/kwargs
    (Argument)

```


kwargs:, 165	clean_asynchronously:, 143
simplify_mdata/model_devi/resources[LSF]/kwargs (Argument)	simplify_mdata/train/machine/context_type (Argument)
kwargs:, 164	context_type:, 144
simplify_mdata/model_devi/resources[LSF]/kwargs (Argument)	simplify_mdata/train/machine/local_root (Argument)
custom_gpu_line:, 164	local_root:, 143
simplify_mdata/model_devi/resources[LSF]/kwargs (Argument)	simplify_mdata/train/machine/remote_root (Argument)
gpu_exclusive:, 164	remote_root:, 143
simplify_mdata/model_devi/resources[LSF]/kwargs (Argument)	simplify_mdata/train/machine[DpCloudServerContext]/remote_profile (Argument)
gpu_new_syntax:, 164	remote_profile:, 147
simplify_mdata/model_devi/resources[LSF]/kwargs (Argument)	simplify_mdata/train/machine[DpCloudServerContext]/remote_email (Argument)
gpu_usage:, 164	email:, 147
simplify_mdata/model_devi/resources[PBS]/kwargs (Argument)	simplify_mdata/train/machine[DpCloudServerContext]/remote_input_data (Argument)
kwargs:, 163	input_data:, 148
simplify_mdata/model_devi/resources[Shell]/kwargs (Argument)	simplify_mdata/train/machine[DpCloudServerContext]/remote_keep_backup (Argument)
kwargs:, 162	keep_backup:, 147
simplify_mdata/model_devi/resources[SlurmJobArray]/kwargs (Argument)	simplify_mdata/train/machine[DpCloudServerContext]/remote_password (Argument)
kwargs:, 163	password:, 147
simplify_mdata/model_devi/resources[SlurmJobArray]/kwargs (Argument)	simplify_mdata/train/machine[DpCloudServerContext]/remote_program_id (Argument)
custom_gpu_line:, 163	program_id:, 147
simplify_mdata/model_devi/resources[Slurm]/kwargs (Argument)	simplify_mdata/train/machine[HDFSContext]/remote_profile (Argument)
kwargs:, 163	remote_profile:, 148
simplify_mdata/model_devi/resources[Slurm]/kwargs (Argument)	simplify_mdata/train/machine[LazyLocalContext]/remote_profile (Argument)
custom_gpu_line:, 163	remote_profile:, 148
simplify_mdata/model_devi/resources[Torque]/kwargs (Argument)	simplify_mdata/train/machine[LebesgueContext]/remote_profile (Argument)
kwargs:, 163	remote_profile:, 144
simplify_mdata/model_devi/user_backward_files (Argument)	simplify_mdata/train/machine[LebesgueContext]/remote_email (Argument)
user_backward_files:, 165	email:, 144
simplify_mdata/model_devi/user_forward_files (Argument)	simplify_mdata/train/machine[LebesgueContext]/remote_input_data (Argument)
user_forward_files:, 165	input_data:, 145
simplify_mdata/train (Argument)	simplify_mdata/train/machine[LebesgueContext]/remote_keep_backup (Argument)
train:, 143	keep_backup:, 144
simplify_mdata/train/command (Argument)	simplify_mdata/train/machine[LebesgueContext]/remote_password (Argument)
command:, 143	password:, 144
simplify_mdata/train/machine (Argument)	simplify_mdata/train/machine[LebesgueContext]/remote_program_id (Argument)
machine:, 143	program_id:, 144
simplify_mdata/train/machine/batch_type (Argument)	simplify_mdata/train/machine[LocalContext]/remote_profile (Argument)
batch_type:, 143	
simplify_mdata/train/machine/clean_asynchronously (Argument)	

```

remote_profile:, 146
simplify_mdata/train/machine[SSHContext]/remote_profile(Argument)
remote_profile:, 145
simplify_mdata/train/machine[SSHContext]/remote_profile(Argument)
hostname:, 145
simplify_mdata/train/machine[SSHContext]/remote_profile(Argument)
key_filename:, 146
simplify_mdata/train/machine[SSHContext]/remote_profile(Argument)
passphrase:, 146
simplify_mdata/train/machine[SSHContext]/remote_profile(Argument)
password:, 145
simplify_mdata/train/machine[SSHContext]/remote_profile(Argument)
port:, 145
simplify_mdata/train/machine[SSHContext]/remote_profile(Argument)
tar_compress:, 146
simplify_mdata/train/machine[SSHContext]/remote_profile(Argument)
timeout:, 146
simplify_mdata/train/machine[SSHContext]/remote_profile(Argument)
totp_secret:, 146
simplify_mdata/train/machine[SSHContext]/remote_profile(Argument)
username:, 145
simplify_mdata/train/resources (Argument)
resources:, 148
simplify_mdata/train/resources/batch_type (Argument)
batch_type:, 151
simplify_mdata/train/resources/cpu_per_node (Argument)
cpu_per_node:, 148
simplify_mdata/train/resources/custom_flags (Argument)
custom_flags:, 149
simplify_mdata/train/resources/envs (Argument)
envs:, 150
simplify_mdata/train/resources/gpu_per_node (Argument)
gpu_per_node:, 148
simplify_mdata/train/resources/group_size (Argument)
group_size:, 149
simplify_mdata/train/resources/module_list (Argument)
module_list:, 150
simplify_mdata/train/resources/module_purge (Argument)
module_purge:, 150
simplify_mdata/train/resources/module_unload_list (Argument)
module_unload_list:, 150
simplify_mdata/train/resources/number_node (Argument)
number_node:, 148
simplify_mdata/train/resources/para_deg (Argument)
para_deg:, 150
simplify_mdata/train/resources/queue_name (Argument)
queue_name:, 149
simplify_mdata/train/resources/source_list (Argument)
source_list:, 150
simplify_mdata/train/resources/strategy (Argument)
strategy:, 149
simplify_mdata/train/resources/strategy/if_cuda_multi_devices (Argument)
if_cuda_multi_devices:, 149
simplify_mdata/train/resources/strategy/ratio_unfinished (Argument)
ratio_unfinished:, 149
simplify_mdata/train/resources/strategy/totp_secret (Argument)
ratio_unfinished:, 149
simplify_mdata/train/resources/wait_time (Argument)
wait_time:, 151
simplify_mdata/train/resources[DistributedShell]/kwargs (Argument)
kwargs:, 153
simplify_mdata/train/resources[DpCloudServer]/kwargs (Argument)
kwargs:, 152
simplify_mdata/train/resources[Lebesgue]/kwargs (Argument)
kwargs:, 153
simplify_mdata/train/resources[LSF]/kwargs (Argument)
kwargs:, 152
simplify_mdata/train/resources[LSF]/kwargs/custom_gpu_line (Argument)
custom_gpu_line:, 153
simplify_mdata/train/resources[LSF]/kwargs/gpu_exclusive (Argument)
gpu_exclusive:, 153
simplify_mdata/train/resources[LSF]/kwargs/gpu_new_syntax (Argument)
gpu_new_syntax:, 153
simplify_mdata/train/resources[LSF]/kwargs/gpu_usage (Argument)
gpu_usage:, 152

```

`simplify_mdata/train/resources[PBS]/kwargs`
 (*Argument*), 151
`simplify_mdata/train/resources[Shell]/kwargs`
 (*Argument*), 151
`simplify_mdata/train/resources[SlurmJobArray]/kwargs`
 (*Argument*), 152
`simplify_mdata/train/resources[SlurmJobArray]/kwargs/custom_gpu_line`
 (*Argument*), 152
`simplify_mdata/train/resources[Slurm]/kwargs`
 (*Argument*), 151
`simplify_mdata/train/resources[Slurm]/kwargs/custom_gpu_line`
 (*Argument*), 151
`simplify_mdata/train/resources[Torque]/kwargs`
 (*Argument*), 152
`simplify_mdata/train/user_backward_files` (*Argument*)
 user_backward_files:, 154
`simplify_mdata/train/user_forward_files`
 (*Argument*)
 user_forward_files:, 153
`simplify_mdata:`
 simplify_mdata (*Argument*), 142
`simplify_mdata_arginfo()` (in module *dp-gen.simplify_arginfo*), 280
`SJX_5p()` (in module *dpgen.auto_test.lib.mfp_eosfit*), 225
`SJX_v2()` (in module *dpgen.auto_test.lib.mfp_eosfit*), 225
`Slurm` (class in *dpgen.dispatcher.Slurm*), 266
`SlurmJob` (class in *dpgen.auto_test.lib.RemoteJob*), 222
`SlurmJob` (class in *dpgen.auto_test.lib.SlurmJob*), 222
`SlurmJob` (class in *dpgen.remote.RemoteJob*), 278
`sort_poscar()` (in module *dpgen.auto_test.lib.vasp*), 231
`source_list:`
 init_bulk_mdata/fp/resources/source_list
 (*Argument*), 77
 init_reaction_mdata/build/resources/source_list
 (*Argument*), 117
 init_reaction_mdata/fp/resources/source_list
 (*Argument*), 128
 init_reaction_mdata/reaxff/resources/source_list
 (*Argument*), 105
 init_surf_mdata/fp/resources/source_list
 (*Argument*), 91
 run_mdata/fp/resources/source_list (*Argument*), 64
 run_mdata/model_devi/resources/source_list
 (*Argument*), 53
 run_mdata/train/resources/source_list
 (*Argument*), 41
 simplify_mdata/fp/resources/source_list
 (*Argument*), 172
 simplify_mdata/model_devi/resources/source_list
 (*Argument*), 161
 simplify_mdata/train/resources/source_list
 (*Argument*), 150
 SPRetObj (class in *dpgen.dispatcher.LazyLocalContext*), 262
 SPRetObj (class in *dpgen.dispatcher.LocalContext*), 263
 ssh (*dpgen.dispatcher.SSHContext.SSHContext* property), 265
 SSHContext (class in *dpgen.dispatcher.SSHContext*), 264
 SSHSession (class in *dpgen.auto_test.lib.RemoteJob*), 221
 SSHSession (class in *dpgen.dispatcher.SSHContext*), 265
 SSHSession (class in *dpgen.remote.RemoteJob*), 278
 stat_iter() (in module *dpgen.tools.stat_iter*), 284
 stat_sys() (in module *dpgen.tools.stat_sys*), 284
 stat_time() (in module *dpgen.tools.stat_time*), 284
 strategy:
 init_bulk_mdata/fp/resources/strategy
 (*Argument*), 77
 init_reaction_mdata/build/resources/strategy
 (*Argument*), 116
 init_reaction_mdata/fp/resources/strategy
 (*Argument*), 128
 init_reaction_mdata/reaxff/resources/strategy
 (*Argument*), 105
 init_surf_mdata/fp/resources/strategy
 (*Argument*), 90
 run_mdata/fp/resources/strategy (*Argument*), 63
 run_mdata/model_devi/resources/strategy
 (*Argument*), 52
 run_mdata/train/resources/strategy (*Argument*), 41
 simplify_mdata/fp/resources/strategy (*Argument*), 172
 simplify_mdata/model_devi/resources/strategy
 (*Argument*), 160
 simplify_mdata/train/resources/strategy
 (*Argument*), 149
 stress6_to_stress9() (in module *dp-gen.data.tools.io_lammps*), 248
 stress9_to_stress6() (in module *dp-gen.data.tools.io_lammps*), 248
 stru2Structure() (in module *dp-gen.auto_test.lib.abacus*), 223

- `stru_ele()` (in module `dpgen.data.gen`), 250
`stru_fix_atom()` (in module `dp-gen.auto_test.lib.abacus`), 223
`stru_scale()` (in module `dpgen.auto_test.lib.abacus`), 223
`sub_script()` (`dpgen.dispatcher.AWS.AWS` method), 256
`sub_script()` (`dpgen.dispatcher.Batch.Batch` method), 256
`sub_script_cmd()` (`dpgen.dispatcher.Batch.Batch` method), 256
`sub_script_cmd()` (`dpgen.dispatcher.LSF.LSF` method), 261
`sub_script_cmd()` (`dpgen.dispatcher.PBS.PBS` method), 264
`sub_script_cmd()` (`dpgen.dispatcher.Shell.Shell` method), 266
`sub_script_cmd()` (`dpgen.dispatcher.Slurm.Slurm` method), 267
`sub_script_head()` (`dpgen.dispatcher.Batch.Batch` method), 256
`sub_script_head()` (`dpgen.dispatcher.LSF.LSF` method), 261
`sub_script_head()` (`dpgen.dispatcher.PBS.PBS` method), 264
`sub_script_head()` (`dpgen.dispatcher.Shell.Shell` method), 266
`sub_script_head()` (`dpgen.dispatcher.Slurm.Slurm` method), 267
`submit()` (`dpgen.auto_test.lib.BatchJob.BatchJob` method), 220
`submit()` (`dpgen.auto_test.lib.RemoteJob.CloudMachineJob` method), 220
`submit()` (`dpgen.auto_test.lib.RemoteJob.PBSJob` method), 221
`submit()` (`dpgen.auto_test.lib.RemoteJob.SlurmJob` method), 222
`submit()` (`dpgen.dispatcher.Batch.Batch` method), 256
`submit()` (`dpgen.remote.RemoteJob.CloudMachineJob` method), 276
`submit()` (`dpgen.remote.RemoteJob.LSFJob` method), 277
`submit()` (`dpgen.remote.RemoteJob.PBSJob` method), 277
`submit()` (`dpgen.remote.RemoteJob.SlurmJob` method), 278
`submit_command()` (`dp-gen.auto_test.lib.BatchJob.BatchJob` method), 220
`submit_command()` (`dp-gen.auto_test.lib.SlurmJob.SlurmJob` method), 222
`submit_jobs()` (`dpgen.dispatcher.Dispatcher.Dispatcher` method), 257
`Surface` (class in `dpgen.auto_test.Surface`), 239
`symlink_user_forward_files()` (in module `dp-gen.generator.lib.utils`), 271
`sys_batch_size:`
 `run_jdata/sys_batch_size` (Argument), 19
 `simplify_jdata/sys_batch_size` (Argument), 137
`sys_configs:`
 `run_jdata/sys_configs` (Argument), 19
 `simplify_jdata/sys_configs` (Argument), 136
`sys_configs_prefix:`
 `run_jdata/sys_configs_prefix` (Argument), 19
 `simplify_jdata/sys_configs_prefix` (Argument), 136
`sys_format:`
 `run_jdata/sys_format` (Argument), 19
 `simplify_jdata/sys_format` (Argument), 136
`sys_idx:`
 `run_jdata[model_devi_engine=amber]/model_devi_jobs/sys` (Argument), 26
 `run_jdata[model_devi_engine=lammps]/model_devi_jobs/sys` (Argument), 22
`sys_link_fp_vasp_pp()` (in module `dp-gen.generator.run`), 275
`System` (class in `dpgen.tools.auto_gen_param`), 282
`system_data()` (in module `dpgen.auto_test.lib.lmp`), 225
- ## T
- `take_cluster()` (in module `dp-gen.generator.lib.gaussian`), 269
`tar_compress:`
 `init_bulk_mdata/fp/machine[SSHContext]/remote_profile/` (Argument), 74
 `init_reaction_mdata/build/machine[SSHContext]/remote_p` (Argument), 113
 `init_reaction_mdata/fp/machine[SSHContext]/remote_prof` (Argument), 125
 `init_reaction_mdata/reaxff/machine[SSHContext]/remote_` (Argument), 102
 `init_surf_mdata/fp/machine[SSHContext]/remote_profile/` (Argument), 87
 `run_mdata/fp/machine[SSHContext]/remote_profile/tar_co` (Argument), 60
 `run_mdata/model_devi/machine[SSHContext]/remote_profil` (Argument), 49
 `run_mdata/train/machine[SSHContext]/remote_profile/tar` (Argument), 38
 `simplify_mdata/fp/machine[SSHContext]/remote_profile/t` (Argument), 169
 `simplify_mdata/model_devi/machine[SSHContext]/remote_p` (Argument), 158
 `simplify_mdata/train/machine[SSHContext]/remote_profil` (Argument), 146
`Task` (class in `dpgen.auto_test.Task`), 240

`task_param` (*dpgen.auto_test.Property.Property* property), 239
`task_param()` (*dpgen.auto_test.Elastic.Elastic* method), 234
`task_param()` (*dpgen.auto_test.EOS.EOS* method), 233
`task_param()` (*dpgen.auto_test.Gamma.Gamma* method), 235
`task_param()` (*dpgen.auto_test.Interstitial.Interstitial* method), 236
`task_param()` (*dpgen.auto_test.Surface.Surface* method), 240
`task_param()` (*dpgen.auto_test.Vacancy.Vacancy* method), 244
`task_type` (*dpgen.auto_test.Property.Property* property), 239
`task_type()` (*dpgen.auto_test.Elastic.Elastic* method), 234
`task_type()` (*dpgen.auto_test.EOS.EOS* method), 233
`task_type()` (*dpgen.auto_test.Gamma.Gamma* method), 235
`task_type()` (*dpgen.auto_test.Interstitial.Interstitial* method), 236
`task_type()` (*dpgen.auto_test.Surface.Surface* method), 240
`task_type()` (*dpgen.auto_test.Vacancy.Vacancy* method), 244
`tau_t:`
 `init_reaction_jdata/reaxff/tau_t` (Argument), 97
`taup:`
 `run_jdata[model_devi_engine=lammps]/model_devi_jobs/taup` (Argument), 23
`taut:`
 `run_jdata[model_devi_engine=lammps]/model_devi_jobs/taut` (Argument), 23
`temp:`
 `init_reaction_jdata/reaxff/temp` (Argument), 96
`temps:`
 `run_jdata[model_devi_engine=lammps]/model_devi_jobs/temps` (Argument), 22
`TEOS()` (in module *dpgen.auto_test.lib.mfp_eosfit*), 225
`terminated` (*dpgen.auto_test.lib.BatchJob.JobStatus* attribute), 220
`terminated` (*dpgen.auto_test.lib.RemoteJob.JobStatus* attribute), 220
`terminated` (*dpgen.dispatcher.JobStatus.JobStatus* attribute), 260
`terminated` (*dpgen.remote.RemoteJob.JobStatus* attribute), 276
`test_fit()` (in module *dpgen.auto_test.gen_confs*), 245
`timeout:`
 `init_bulk_mdata/fp/machine[SSHContext]/remote_profile/timeout` (Argument), 73
 `init_reaction_mdata/build/machine[SSHContext]/remote_profile/timeout` (Argument), 113
 `init_reaction_mdata/fp/machine[SSHContext]/remote_profile/timeout` (Argument), 124
 `init_reaction_mdata/reaxff/machine[SSHContext]/remote_profile/timeout` (Argument), 101
 `init_surf_mdata/fp/machine[SSHContext]/remote_profile/timeout` (Argument), 87
 `run_mdata/fp/machine[SSHContext]/remote_profile/timeout` (Argument), 60
 `run_mdata/model_devi/machine[SSHContext]/remote_profile/timeout` (Argument), 49
 `run_mdata/train/machine[SSHContext]/remote_profile/timeout` (Argument), 38
 `simplify_mdata/fp/machine[SSHContext]/remote_profile/timeout` (Argument), 169
 `simplify_mdata/model_devi/machine[SSHContext]/remote_profile/timeout` (Argument), 157
 `simplify_mdata/train/machine[SSHContext]/remote_profile/timeout` (Argument), 146
`to_system_data()` (in module *dpgen.auto_test.lib.lmp*), 225
`totp_secret:`
 `init_bulk_mdata/fp/machine[SSHContext]/remote_profile/totp_secret` (Argument), 74
 `init_reaction_mdata/build/machine[SSHContext]/remote_profile/totp_secret` (Argument), 113
 `init_reaction_mdata/fp/machine[SSHContext]/remote_profile/totp_secret` (Argument), 125
 `init_reaction_mdata/reaxff/machine[SSHContext]/remote_profile/totp_secret` (Argument), 102
 `init_surf_mdata/fp/machine[SSHContext]/remote_profile/totp_secret` (Argument), 87
 `run_mdata/fp/machine[SSHContext]/remote_profile/totp_secret` (Argument), 60
 `run_mdata/model_devi/machine[SSHContext]/remote_profile/totp_secret` (Argument), 49
 `run_mdata/train/machine[SSHContext]/remote_profile/totp_secret` (Argument), 38
 `simplify_mdata/fp/machine[SSHContext]/remote_profile/totp_secret` (Argument), 169
 `simplify_mdata/model_devi/machine[SSHContext]/remote_profile/totp_secret` (Argument), 157
 `simplify_mdata/train/machine[SSHContext]/remote_profile/totp_secret` (Argument), 146
`train:`
 `run_mdata/train` (Argument), 35
 `simplify_mdata/train` (Argument), 143
`training_args()` (in module *dpgen.generator.arginfo*), 272
`training_init_model:`
 `run_jdata/training_init_model` (Argument), 272
 `simplify_jdata/training_init_model` (Argument), 143

- ment), 137
- training_iter0_model_path:
 run_jdata/training_iter0_model_path (Argument), 19
 simplify_jdata/training_iter0_model_path (Argument), 137
- training_reuse_iter:
 run_jdata/training_reuse_iter (Argument), 20
 simplify_jdata/training_reuse_iter (Argument), 138
- training_reuse_num_steps:
 run_jdata/training_reuse_num_steps (Argument), 20
 simplify_jdata/training_reuse_num_steps (Argument), 138
- training_reuse_old_ratio:
 run_jdata/training_reuse_old_ratio (Argument), 20
 simplify_jdata/training_reuse_old_ratio (Argument), 138
- training_reuse_start_lr:
 run_jdata/training_reuse_start_lr (Argument), 20
 simplify_jdata/training_reuse_start_lr (Argument), 138
- training_reuse_start_pref_e:
 run_jdata/training_reuse_start_pref_e (Argument), 20
 simplify_jdata/training_reuse_start_pref_e (Argument), 138
- training_reuse_start_pref_f:
 run_jdata/training_reuse_start_pref_f (Argument), 21
 simplify_jdata/training_reuse_start_pref_fuse_clusters: (Argument), 139
- trj_freq:
 run_jdata[model_devi_engine=amber]/model_devi_jobs/data_freq (Argument), 27
 run_jdata[model_devi_engine=lammps]/model_devi_jobs/trj_freq (Argument), 22
- type_map:
 init_reaction_jdata/type_map (Argument), 96
 run_jdata/type_map (Argument), 18
 simplify_jdata/type_map (Argument), 135
- U**
- ucloud_submit_jobs() (in module dp-gen.remote.group_jobs), 280
- uniq() (in module dpgen.data.tools.cessp2force_lin), 247
- universal() (in module dpgen.auto_test.lib.mfp_eosfit), 229
- unknown (dpgen.auto_test.lib.BatchJob.JobStatus attribute), 220
- unknown (dpgen.auto_test.lib.RemoteJob.JobStatus attribute), 220
- unknown (dpgen.dispatcher.JobStatus.JobStatus attribute), 260
- unknown (dpgen.remote.RemoteJob.JobStatus attribute), 276
- unsubmitted (dpgen.auto_test.lib.BatchJob.JobStatus attribute), 220
- unsubmitted (dpgen.auto_test.lib.RemoteJob.JobStatus attribute), 221
- unsubmitted (dpgen.dispatcher.JobStatus.JobStatus attribute), 260
- unsubmitted (dpgen.remote.RemoteJob.JobStatus attribute), 276
- update() (dpgen.dispatcher.DispatcherList.DispatcherList method), 260
- update_dict() (in module dpgen.generator.lib.cp2k), 268
- update_mass_map() (in module dpgen.generator.run), 275
- upload() (dpgen.auto_test.lib.RemoteJob.RemoteJob method), 221
- upload() (dpgen.dispatcher.LazyLocalContext.LazyLocalContext method), 262
- upload() (dpgen.dispatcher.LocalContext.LocalContext method), 263
- upload() (dpgen.dispatcher.SSHContext.SSHContext method), 265
- upload() (dpgen.remote.RemoteJob.awsMachineJob method), 279
- upload() (dpgen.remote.RemoteJob.RemoteJob method), 278
- run_jdata[fp_style=gaussian]/use_clusters (Argument), 30
- run_jdata[fp_style=siesta]/use_clusters (Argument), 31
- run_jdata[gaussian]/use_clusters (Argument), 140
- use_ele_temp:
 run_jdata/use_ele_temp (Argument), 18
 simplify_jdata/use_ele_temp (Argument), 136
- use_relative:
 run_jdata[model_devi_engine=lammps]/use_relative (Argument), 26
- use_relative_v:
 run_jdata[model_devi_engine=lammps]/use_relative_v (Argument), 26
- user_backward_files:
 init_bulk_mdata/fp/user_backward_files (Argument), 81
 init_reaction_mdata/build/user_backward_files

(Argument), 121
 init_reaction_mdata/fp/user_backward_files
 (Argument), 132
 init_reaction_mdata/reaxff/user_backward_files
 (Argument), 109
 init_surf_mdata/fp/user_backward_files
 (Argument), 95
 run_mdata/fp/user_backward_files (Argument), 67
 run_mdata/model_devi/user_backward_files
 (Argument), 57
 run_mdata/train/user_backward_files
 (Argument), 45
 simplify_mdata/fp/user_backward_files
 (Argument), 176
 simplify_mdata/model_devi/user_backward_files
 (Argument), 165
 simplify_mdata/train/user_backward_files
 (Argument), 154
 user_forward_files:
 init_bulk_mdata/fp/user_forward_files
 (Argument), 81
 init_reaction_mdata/build/user_forward_files
 (Argument), 121
 init_reaction_mdata/fp/user_forward_files
 (Argument), 132
 init_reaction_mdata/reaxff/user_forward_files
 (Argument), 109
 init_surf_mdata/fp/user_forward_files
 (Argument), 95
 run_mdata/fp/user_forward_files (Argument), 67
 run_mdata/model_devi/user_forward_files
 (Argument), 56
 run_mdata/train/user_forward_files (Argument), 45
 simplify_mdata/fp/user_forward_files (Argument), 176
 simplify_mdata/model_devi/user_forward_files
 (Argument), 165
 simplify_mdata/train/user_forward_files
 (Argument), 153
 user_fp_params:
 run_jdata[fp_style=abacus]/user_fp_params
 (Argument), 34
 run_jdata[fp_style=cp2k]/user_fp_params
 (Argument), 32
 username:
 init_bulk_mdata/fp/machine[SSHContext]/remote_profile/username
 (Argument), 73
 init_reaction_mdata/build/machine[SSHContext]/remote_profile/username
 (Argument), 112
 init_reaction_mdata/fp/machine[SSHContext]/remote_profile/username
 (Argument), 124
 init_reaction_mdata/reaxff/machine[SSHContext]/remote_profile/username
 (Argument), 100
 init_surf_mdata/fp/machine[SSHContext]/remote_profile/username
 (Argument), 86
 run_mdata/fp/machine[SSHContext]/remote_profile/username
 (Argument), 59
 run_mdata/model_devi/machine[SSHContext]/remote_profile/username
 (Argument), 48
 run_mdata/train/machine[SSHContext]/remote_profile/username
 (Argument), 37
 simplify_mdata/fp/machine[SSHContext]/remote_profile/username
 (Argument), 168
 simplify_mdata/model_devi/machine[SSHContext]/remote_profile/username
 (Argument), 156
 simplify_mdata/train/machine[SSHContext]/remote_profile/username
 (Argument), 145
V
 Vacancy (class in dpgen.auto_test.Vacancy), 244
 valid_hash() (dpgen.dispatcher.Dispatcher.JobRecord method), 258
 VASP (class in dpgen.auto_test.VASP), 242
 VaspInput (class in dpgen.database.vasp), 253
 vinet() (in module dpgen.auto_test.lib.mfp_eosfit), 229
 vinet_pv() (in module dpgen.auto_test.lib.mfp_eosfit), 229
 vort_to_stress() (in module dpgen.auto_test.lib.util), 229
W
 wait_time:
 init_bulk_mdata/fp/resources/wait_time
 (Argument), 78
 init_reaction_mdata/build/resources/wait_time
 (Argument), 118
 init_reaction_mdata/fp/resources/wait_time
 (Argument), 129
 init_reaction_mdata/reaxff/resources/wait_time
 (Argument), 106
 init_surf_mdata/fp/resources/wait_time
 (Argument), 92
 run_mdata/fp/resources/wait_time (Argument), 64
 run_mdata/model_devi/resources/wait_time
 (Argument), 54
 run_mdata/train/resources/wait_time
 (Argument), 42
 simplify_mdata/fp/resources/wait_time
 (Argument), 173
 simplify_mdata/model_devi/resources/wait_time
 (Argument), 162
 simplify_mdata/train/resources/wait_time
 (Argument), 151

`waiting` (*dpgen.auto_test.lib.BatchJob.JobStatus* attribute), 220

`waiting` (*dpgen.auto_test.lib.RemoteJob.JobStatus* attribute), 221

`waiting` (*dpgen.dispatcher.JobStatus.JobStatus* attribute), 260

`waiting` (*dpgen.remote.RemoteJob.JobStatus* attribute), 276

`worker()` (in module *dpgen.auto_test.common_prop*), 245

`write_file()` (*dpgen.database.vasp.DPPotcar* method), 253

`write_file()` (*dpgen.dispatcher.LazyLocalContext.LazyLocalContext* method), 262

`write_file()` (*dpgen.dispatcher.LocalContext.LocalContext* method), 263

`write_file()` (*dpgen.dispatcher.SSHContext.SSHContext* method), 265

`write_incar_dict()` (in module *dp-gen.generator.lib.vasp*), 271

`write_input()` (*dpgen.database.vasp.VaspInput* method), 255

`write_input()` (in module *dpgen.auto_test.lib.abacus*), 223

`write_input_dict()` (in module *dp-gen.generator.lib.pwmat*), 269

`write_kpt()` (in module *dpgen.auto_test.lib.abacus*), 223

`write_model_devi_out()` (in module *dp-gen.generator.lib.make_calypso*), 269