
DPTI

DeepModeling

Apr 13, 2024

CONTENTS:

1	Install DPTI	3
2	Getting Started	5
3	Command line interface	7
3.1	modules	7
3.2	Sub-commands	7
4	DPTI API	25
4.1	dpti package	25
5	Phase Diagram of Water	37
6	Authors	39
7	Indices and tables	41
	Python Module Index	43
	Index	45

DPTI is a Python package to automate thermodynamic integration (TI) calculations for free energy.

INSTALL DPTI

DPTI can installed by pip:

```
pip install dpti
```


GETTING STARTED

DPTI: A Thermodynamic Integration Automization Package for Free Energy Calculation

TODO

COMMAND LINE INTERFACE

DPTI: An Automatic Workflow Software for Thermodynamic Integration Calculations

```
usage: dpti [-h] {equi,hti,hti_liq,hti_ice,hti_water,ti,ti_water,gdi,mti} ...
```

3.1 modules

the subcommands of dpti

module

Possible choices: equi, hti, hti_liq, hti_ice, hti_water, ti, ti_water, gdi, mti

module-level help

3.2 Sub-commands

3.2.1 equi

equilibration simulations

```
dpti equi [-h] {gen,extract,stat-bond,compute,run} ...
```

Positional Arguments

command

Possible choices: gen, extract, stat-bond, compute, run

commands for equilibration simulations

Sub-commands

gen

generate a job

```
dpti equi gen [-h] [-e ENSEMBLE] [-t TEMPERATURE] [-p PRESSURE] [-a]
              [-c CONF_NPT] [-o OUTPUT]
              PARAM
```

Positional Arguments

PARAM	json parameter file
--------------	---------------------

Named Arguments

-e, --ensemble	the ensemble of the simulation
-t, --temperature	the temperature of the system
-p, --pressure	the pressure of the system
-a, --avg-posi	dump the average position of atoms Default: False
-c, --conf-npt	use conf computed from NPT simulation
-o, --output	the output folder for the job Default: “new_job”

extract

extract the conf

```
dpti equi extract [-h] [-o OUTPUT] JOB
```

Positional Arguments

JOB	folder of the job
------------	-------------------

Named Arguments

-o, --output	output conf file name Default: “conf.lmp”
---------------------	--

stat-bond

Statistic of the bonds

```
dpti equi stat-bond [-h] [-s SKIP] JOB
```

Positional Arguments

JOB	folder of the job
------------	-------------------

Named Arguments

-s, --skip	skip this number of frames
	Default: 1

compute

Compute thermodynamics

```
dpti equi compute [-h] JOB
```

Positional Arguments

JOB	folder of the job
------------	-------------------

run

run the job

```
dpti equi run [-h] JOB machine
```

Positional Arguments

JOB	folder of the job
machine	machine.json file for the job

3.2.2 hti

Hamiltonian thermodynamic integration for atomic solid

```
dpti hti [-h] {gen,compute,run} ...
```

Positional Arguments

command	Possible choices: gen, compute, run commands of Hamiltonian thermodynamic integration for atomic solid
----------------	---

Sub-commands

gen

generate a job

```
dpti hti gen [-h] [-o OUTPUT] [-s {one-step,two-step,three-step}] [-z] PARAM
```

Positional Arguments

PARAM	json parameter file
--------------	---------------------

Named Arguments

-o, --output	the output folder for the job Default: “new_job”
-s, --switch	Possible choices: one-step, two-step, three-step one-step: switching on DP and switching off spring simultaneously. two-step: 1 switching on DP, 2 switching off spring. three-step: 1 switching on soft LJ, 2 switching on DP, 3 switching off spring and soft LJ. Default: “one-step”
-z, --meam	whether use meam instead of dp Default: False

compute

Compute the result of a job

```
dpti hti compute [-h] [-t {helmholtz,gibbs}] [-m {inte,mbar}] [-s SCHEME]
                  [-g PV] [-G PV_ERR]
                  JOB
```

Positional Arguments

JOB folder of the job

Named Arguments

-t, --type Possible choices: helmholtz, gibbs
the type of free energy
Default: “helmholtz”

-m, --inte-method Possible choices: inte, mbar
the method of thermodynamic integration
Default: “inte”

-s, --scheme the numeric integration scheme
Default: “simpson”

-g, --pv press*vol value override to calculate Gibbs free energy

-G, --pv-err press*vol error

run

run the job

```
dpti hti run [-h] [--no-dp] JOB machine task_name
```

Positional Arguments

JOB folder of the job

machine machine.json file for the job

task_name task name, can be 00, 01, or 02

Named Arguments

--no-dp whether to use Deep Potential or not
Default: False

3.2.3 hti_liq

Hamiltonian thermodynamic integration for atomic liquid

```
dpti hti_liq [-h] {gen,compute,run} ...
```

Positional Arguments

command	Possible choices: gen, compute, run commands of Hamiltonian thermodynamic integration for atomic liquid
----------------	--

Sub-commands

gen

Generate a job

```
dpti hti_liq gen [-h] [-o OUTPUT] [-z] PARAM
```

Positional Arguments

PARAM	json parameter file
--------------	---------------------

Named Arguments

-o, --output	the output folder for the job Default: “new_job”
-z, --meam	whether use meam instead of dp Default: False

compute

Compute the result of a job

```
dpti hti_liq compute [-h] [-t {helmholtz,gibbs}] [-g PV] [-G PV_ERR] JOB
```

Positional Arguments

JOB	folder of the job
------------	-------------------

Named Arguments

-t, --type	Possible choices: helmholtz, gibbs the type of free energy Default: “helmholtz”
-g, --pv	press*vol value override to calculate Gibbs free energy
-G, --pv-err	press*vol error

run

run the job

```
dpti hti_liq run [-h] [--no-dp] JOB machine task_name
```

Positional Arguments

JOB	folder of the job
machine	machine.json file for the job
task_name	task name, can be 00, 01, or 02

Named Arguments

--no-dp	whether to use Deep Potential or not Default: False
----------------	--

3.2.4 hti_ice

Hamiltonian thermodynamic integration for ice

```
dpti hti_ice [-h] {gen,compute,refine,run} ...
```

Positional Arguments

command	Possible choices: gen, compute, refine, run commands of Hamiltonian thermodynamic integration for ice
----------------	--

Sub-commands

gen

Generate a job

```
dpti hti_ice gen [-h] [-o OUTPUT] [-s {one-step,two-step,three-step}] PARAM
```

Positional Arguments

PARAM	json parameter file
--------------	---------------------

Named Arguments

-o, --output	the output folder for the job Default: “new_job”
-s, --switch	Possible choices: one-step, two-step, three-step one-step: switching on DP and switching off spring simultanenously. two-step: 1 switching on DP, 2 switching off spring. three-step: 1 switching on soft LJ, 2 switching on DP, 3 switching off spring and soft LJ. Default: “one-step”

compute

Compute the result of a job

```
dpti hti_ice compute [-h] [-t {helmholtz,gibbs}] [-m {inte,mbar}] [-d]  
[-p {3,5}] [-s SCHEME] [-S SHIFT] [-g PV] [-G PV_ERR]  
[--npt NPT]  
JOB
```

Positional Arguments

JOB	folder of the job
------------	-------------------

Named Arguments

-t, --type	Possible choices: helmholtz, gibbs the type of free energy Default: “helmholtz”
-m, --inte-method	Possible choices: inte, mbar the method of thermodynamic integration Default: “inte”

-d, --disorder-corr	apply disorder correction for ice Default: True
-p, --partial-disorder	Possible choices: 3, 5 apply partial disorder correction for ice
-s, --scheme	the numeric integration scheme Default: "simpson"
-S, --shift	a constant shift in the energy/mole computation, will be removed from FE Default: 0.0
-g, --pv	press*vol value override to calculate Gibbs free energy
-G, --pv-err	press*vol error
--npt	directory of the npt task; will use PV from npt result, where P is the control variable and V varies.

refine

Refine the grid of a job

```
dpti hti_ice refine [-h] -i INPUT -o OUTPUT -e ERROR [-p]
```

Named Arguments

-i, --input	input job
-o, --output	output job
-e, --error	the error required
-p, --print	print the refinement and exit Default: False

run

run the job

```
dpti hti_ice run [-h] [--no-dp] JOB machine task_name
```

Positional Arguments

JOB	folder of the job
machine	machine.json file for the job
task_name	task name, can be 00, 01, or 02

Named Arguments

--no-dp whether to use Deep Potential or not
Default: False

3.2.5 hti_water

Hamiltonian thermodynamic integration for liquid water

```
dpti hti_water [-h] {gen,compute,refine,run} ...
```

Positional Arguments

command Possible choices: gen, compute, refine, run
commands of Hamiltonian thermodynamic integration for liquid water

Sub-commands

gen

Generate a job

```
dpti hti_water gen [-h] [-o OUTPUT] PARAM
```

Positional Arguments

PARAM json parameter file

Named Arguments

-o, --output the output folder for the job
Default: “new_job”

compute

Compute the result of a job

```
dpti hti_water compute [-h] [-t {helmholtz,gibbs}] [-m {inte,mbar}]  
                        [-s SCHEME] [-g PV] [-G PV_ERR] [--npt NPT]  
                        JOB
```

Positional Arguments

JOB folder of the job

Named Arguments

-t, --type Possible choices: helmholtz, gibbs
the type of free energy
Default: “helmholtz”

-m, --inte-method Possible choices: inte, mbar
the method of thermodynamic integration
Default: “inte”

-s, --scheme the numeric integration scheme
Default: “simpson”

-g, --pv press*vol value override to calculate Gibbs free energy

-G, --pv-err press*vol error

--npt directory of the npt task; will use PV from npt result, where P is the control variable and V varies.

refine

Refine the grid of a job

```
dpti hti_water refine [-h] -i INPUT -o OUTPUT -e ERROR
```

Named Arguments

-i, --input input job

-o, --output output job

-e, --error the error required

run

run the job

```
dpti hti_water run [-h] [--no-dp] JOB machine task_name
```

Positional Arguments

JOB	folder of the job
machine	machine.json file for the job
task_name	task name, can be 00, 01, or 02

Named Arguments

--no-dp	whether to use Deep Potential or not Default: False
----------------	--

3.2.6 ti

thermodynamic integration along isothermal or isobaric paths

```
dpti ti [-h] {gen,compute,refine,run} ...
```

Positional Arguments

command	Possible choices: gen, compute, refine, run commands of thermodynamic integration along isothermal or isobaric paths
----------------	---

Sub-commands

gen

Generate a job

```
dpti ti gen [-h] [-o OUTPUT] [-z] PARAM
```

Positional Arguments

PARAM	json parameter file
--------------	---------------------

Named Arguments

-o, --output	the output folder for the job Default: “new_job”
-z, --meam	whether use meam instead of dp Default: False

compute

Compute the result of a job

```
dpti ti compute [-h] [-m {inte,mbar}] [-e EO] [-E EO_ERR] [-t TO] [-s SCHEME]
                [-H HTI]
                JOB
```

Positional Arguments

JOB folder of the job

Named Arguments

-m, --inte-method	Possible choices: inte, mbar the method of thermodynamic integration Default: “inte”
-e, --Eo	free energy of starting point Default: 0
-E, --Eo-err	The statistical error of the starting free energy Default: 0
-t, --To	the starting thermodynamic position
-s, --scheme	the numerical integration scheme Default: “simpson”
-H, --hti	the HTI job folder; will extract the free energy of the starting point as from the result.json file in this folder

refine

Refine the grid of a job

```
dpti ti refine [-h] -i INPUT -o OUTPUT -e ERROR
```

Named Arguments

-i, --input	input job
-o, --output	output job
-e, --error	the error required

run

run the job

```
dpti ti run [-h] JOB machine
```

Positional Arguments

JOB	folder of the job
machine	machine.json file for the job

3.2.7 ti_water

thermodynamic integration along isothermal or isobaric paths for water

```
dpti ti_water [-h] {gen,compute,refine,run} ...
```

Positional Arguments

command	Possible choices: gen, compute, refine, run commands of thermodynamic integration along isothermal or isobaric paths for water
----------------	---

Sub-commands

gen

Generate a job

```
dpti ti_water gen [-h] [-o OUTPUT] PARAM
```

Positional Arguments

PARAM	json parameter file
--------------	---------------------

Named Arguments

-o, --output	the output folder for the job Default: “new_job”
---------------------	---

compute

Compute the result of a job

```
dpti ti_water compute [-h] [-m {inte,mbar}] [-e EO] [-E EO_ERR] [-t TO]
                        [-s SCHEME] [-S SHIFT] [-H HTI]
                        JOB
```

Positional Arguments

JOB folder of the job

Named Arguments

-m, --inte-method	Possible choices: inte, mbar the method of thermodynamic integration Default: “inte”
-e, --Eo	free energy of starting point
-E, --Eo-err	the statistical error of the starting free energy
-t, --To	the starting thermodynamic position
-s, --scheme	the numerical integration scheme Default: “simpson”
-S, --shift	a constant shift in the energy/mole computation, will be removed from FE Default: 0.0
-H, --hti	the HTI job folder; will extract the free energy of the starting point as from the result.json file in this folder

refine

Refine the grid of a job

```
dpti ti_water refine [-h] -i INPUT -o OUTPUT -e ERROR
```

Named Arguments

-i, --input	input job
-o, --output	output job
-e, --error	the error required

run

run the job

```
dpti ti_water run [-h] JOB machine
```

Positional Arguments

JOB	folder of the job
machine	machine.json file for the job

3.2.8 gdi

compute the phase boundary via Gibbs-Duhem integration

```
dpti gdi [-h] [-g GDIDATA_JSON] [-b BEGIN] [-e END] [-d {t,p}]  
        [-i INITIAL_VALUE] [-s STEP_VALUE [STEP_VALUE ...]] [-a ABS_TOL]  
        [-r REL_TOL] [-w] [-o OUTPUT] [-f FIRST_STEP] [-S SHIFT SHIFT] [-v]  
        [-z]  
        PARAM MACHINE
```

Positional Arguments

PARAM	json parameter file
MACHINE	json machine file

Named Arguments

-g, --gdidata-json	json gdi integration file
-b, --begin	start of the integration
-e, --end	end of the integration
-d, --direction	Possible choices: t, p direction of the integration, along T or P
-i, --initial-value	the initial value of T (direction=p) or P (direction=t)
-s, --step-value	the T (direction=t) or P (direction=p) values must be evaluated
-a, --abs-tol	the absolute tolerance of the integration Default: 10
-r, --rel-tol	the relative tolerance of the integration Default: 0.01
-w, --if-water	assumes water molecules: nmols = natoms//3 Default: False

-o, --output	the output folder for the job Default: “new_job”
-f, --first-step	the first step size of the integrator
-S, --shift	the output folder for the job Default: [0.0, 0.0]
-v, --verbose	print detailed information Default: False
-z, --if-meam	whether use meam instead of dp Default: False

3.2.9 mti

mass thermodynamic integration: quantum free energy calculation using PIMD

```
dpti mti [-h] {gen,run,compute} ...
```

Positional Arguments

command	Possible choices: gen, run, compute commands of mass thermodynamic integration
----------------	---

Sub-commands

gen

Generate a job

```
dpti mti gen [-h] [-o OUTPUT] PARAM
```

Positional Arguments

PARAM	json parameter file
--------------	---------------------

Named Arguments

-o, --output	the output folder for the job Default: “new_job”
---------------------	---

run

run the job

```
dpti mti run [-h] JOB PARAM machine
```

Positional Arguments

JOB	folder of the job
PARAM	json parameter file
machine	machine.json file for the job

compute

Compute the result of a job

```
dpti mti compute [-h] [--natom_mol NATOM_MOL] JOB
```

Positional Arguments

JOB	folder of the job
------------	-------------------

Named Arguments

--natom_mol	the number of atoms in the molecule
--------------------	-------------------------------------

4.1 dpti package

4.1.1 Subpackages

dpti.dags package

Submodules

dpti.dags.dp_ti_gdi module

```
class dpti.dags.dp_ti_gdi.GDIDAGFactory(gdi_name, dag_work_base)
```

Bases: `object`

Methods

<code>create_loop_dag</code>
<code>create_main_dag</code>

```
create_loop_dag()
```

```
create_main_dag()
```

```
dagargs: ClassVar[Dict[str, object]] = {'default_args': {'owner': 'airflow',  
'start_date': datetime.datetime(2018, 1, 1, 0, 0)}, 'schedule_interval': None}
```

```
default_args: ClassVar[Dict[str, object]] = {'owner': 'airflow', 'start_date':  
datetime.datetime(2018, 1, 1, 0, 0)}
```

```
class dpti.dags.dp_ti_gdi.GDIWorkflow(dag_name, var_name)
```

Bases: `object`

Methods

get_dag_run_state
trigger_loop
wait_until_end

get_dag_run_state()

trigger_loop(*submission, task_list, mdata*)

wait_until_end()

dpti.dags.utils module

dpti.dags.utils.get_empty_submission(*job_work_dir, context*)

dpti.lib package

Submodules

dpti.lib.RemoteJob module

class **dpti.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 **dpti.lib.RemoteJob.JobStatus**(*value*)
Bases: *Enum*
An enumeration.
finished = 5
running = 3

```

terminated = 4

unknown = 100

unsubmitted = 1

waiting = 2

```

```

class dpti.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 dpti.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 dpti.lib.RemoteJob.SSHSession(jdata)
    Bases: object

```

Methods

close
get_session_root
get_ssh_client

close()

get_session_root()

get_ssh_client()

class `dpti.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*)

`dpti.lib.dump` module

`dpti.lib.dump`.**box2dumpbox**(*orig, box*)

`dpti.lib.dump`.**dumpbox2box**(*bounds, tilt*)

`dpti.lib.dump`.**get_atype**(*lines*)

`dpti.lib.dump`.**get_dumpbox**(*lines*)

`dpti.lib.dump`.**get_natoms**(*lines*)

`dpti.lib.dump`.**get_natoms_vec**(*lines*)

`dpti.lib.dump`.**get_natomtypes**(*lines*)

`dpti.lib.dump`.**get_posi**(*lines*)

`dpti.lib.dump`.**split_traj**(*dump_lines*)

`dpti.lib.dump`.**system_data**(*lines*)

dpti.lib.lammps module

```
dpti.lib.lammps.get_last_dump(dump)
dpti.lib.lammps.get_natoms(filename)
dpti.lib.lammps.get_thermo(filename)
dpti.lib.lammps.get_thermo_old(filename)
```

dpti.lib.lmp module

```
dpti.lib.lmp.box2lmpbox(orig, box)
dpti.lib.lmp.from_system_data(system)
dpti.lib.lmp.get_atoms(lines)
dpti.lib.lmp.get_atype(lines)
dpti.lib.lmp.get_id(lines)
dpti.lib.lmp.get_lmpbox(lines)
dpti.lib.lmp.get_natoms(lines)
dpti.lib.lmp.get_natoms_vec(lines)
dpti.lib.lmp.get_natomtypes(lines)
dpti.lib.lmp.get_posi(lines)
dpti.lib.lmp.lmpbox2box(lohi, tilt)
dpti.lib.lmp.system_data(lines)
dpti.lib.lmp.to_system_data(lines)
```

dpti.lib.ovito_file_convert module**dpti.lib.utils module**

```
dpti.lib.utils.block_avg(inp, skip=0, block_size=10)
dpti.lib.utils.compute_nrefine(all_t, integrand, err, error_scale=None)
dpti.lib.utils.copy_file_list(file_list, from_path, to_path)
dpti.lib.utils.create_dict_not_empty_key(**kwargs)
dpti.lib.utils.create_path(path)
dpti.lib.utils.cvt_conf(fin, fout, ofmt='vasp')
```

Format convert from fin to fout, specify the output format by ofmt.

```
dpti.lib.utils.get_file_md5(file_path)
dpti.lib.utils.get_first_matched_key_from_dict(dct, lst)
dpti.lib.utils.get_task_file_abspath(task_name, file_name)
dpti.lib.utils.integrate(xx, yy, ye, scheme_='s')
dpti.lib.utils.integrate_range(xx, yy, ye, scheme='s')
dpti.lib.utils.integrate_range_hti(all_lambda, de, de_err, scheme='s')
dpti.lib.utils.integrate_range_simpson(xx, yy, ye)
dpti.lib.utils.integrate_range_trapezoidal(xx, yy, ye)
dpti.lib.utils.integrate_simpson(xx, yy, ye)
dpti.lib.utils.integrate_simpson_nonuniform(x, f, fe)
dpti.lib.utils.integrate_sys_err(xx, yy, scheme_='s')
dpti.lib.utils.integrate_sys_err_simpson(xx, yy)
dpti.lib.utils.integrate_sys_err_trapezoidal(xx, yy)
dpti.lib.utils.integrate_trapezoidal(xx, yy, ye)
dpti.lib.utils.interval_sys_err_trapezoidal(xx, yy, mode)
dpti.lib.utils.link_file_in_dict(dct, key_list, target_dir)
dpti.lib.utils.make_iter_name(iter_index)
dpti.lib.utils.parse_seq(in_s, *, protect_eps=None)
dpti.lib.utils.relative_link_file(file_path, target_dir)
```

dpti.lib.vasp module

```
dpti.lib.vasp.perturb_xz(poscar_in, poscar_out, pert=0.01)
dpti.lib.vasp.poscar_natoms(poscar_in)
dpti.lib.vasp.poscar_scale(poscar_in, poscar_out, scale)
dpti.lib.vasp.poscar_vol(poscar_in)
dpti.lib.vasp.reciprocal_box(box)
dpti.lib.vasp.regulate_poscar(poscar_in, poscar_out)
dpti.lib.vasp.sort_poscar(poscar_in, poscar_out, new_names)
```

dpti.lib.water module

```
dpti.lib.water.add_bonds(lines_, max_roh=1.3)
dpti.lib.water.compute_bonds(box, atype, posis, max_roh=1.3, uniq_hbond=True)
dpti.lib.water.dist_via_oh_list(box, posis, list_oh)
dpti.lib.water.min_ho(box, atype, posis)
dpti.lib.water.min_oh_list(box, atype, posis)
dpti.lib.water.min_oho(box, atype, posis)
dpti.lib.water.min_oo(box, atype, posis)
dpti.lib.water.posi_diff(box, r0, r1)
dpti.lib.water.posi_shift(box, r0, r1)
```

4.1.2 Submodules

4.1.3 dpti.einstein module

```
dpti.einstein.compute_lambda(temp, mass)
dpti.einstein.compute_spring(temp, spring_k)
dpti.einstein.free_energy(job)
dpti.einstein.frenkel(job)
dpti.einstein.ideal_gas_fe(job)
```

4.1.4 dpti.equi module

```
dpti.equi.add_module_subparsers(main_subparsers)
dpti.equi.add_subparsers(module_subparsers)
dpti.equi.exec_args(args, parser)
dpti.equi.extract(job_dir, output)
dpti.equi.gen_equi_dump_settings(if_dump_avg_posi)
dpti.equi.gen_equi_ensemble_settings(ens)
dpti.equi.gen_equi_force_field(model, if_meam=False, meam_model=None)
dpti.equi.gen_equi_header(nsteps, thermo_freq, dump_freq, mass_map, temp, tau_t, tau_p, equi_conf,  
                        pres=None)
```

```
dpti.equi.gen_equi_lammps_input(nsteps, thermo_freq, dump_freq, mass_map, temp, tau_t, tau_p, equi_conf,
                               model, timestep, if_dump_avg_posi, ens, pres=None, if_meam=False,
                               meam_model=None)

dpti.equi.gen_equi_thermo_settings(timestep)

dpti.equi.handle_compute(args)

dpti.equi.handle_extract(args)

dpti.equi.handle_gen(args)

dpti.equi.handle_run(args)

dpti.equi.handle_stat_bond(args)

dpti.equi.make_task(iter_name, jdata, ens=None, temp=None, pres=None, if_dump_avg_posi=None,
                   npt_dir=None)

dpti.equi.npt_equi_conf(npt_dir)

dpti.equi.post_task(iter_name, natoms=None, is_water=None)

dpti.equi.run_task(task_name, machine_file)

dpti.equi.water_bond(iter_name, skip=1)
```

4.1.5 dpti.gdi module

```
class dpti.gdi.GibbsDuhemFunc(jdata, mdata, task_path, inte_dir, pref=1.0, natoms=None, shift=[0, 0],
                              verbose=False, if_meam=False, meam_model=None, workflow=None)
```

Bases: `object`

Methods

<code>__call__(x, y)</code>	Call self as a function.
-----------------------------	--------------------------

```
dpti.gdi.add_module_subparsers(main_subparsers)

dpti.gdi.gdi_main_loop(jdata, mdata, gdidata_dict={}, gdidata_cli={}, workflow=None)

dpti.gdi.handle_gdi(args)

dpti.gdi.make_dpdt(temp, pres, inte_dir, task_path, mdata, natoms=None, shift=[0, 0], verbose=False,
                  if_meam=False, meam_model=None, workflow=None)
```

4.1.6 dpti.hti module

```

dpti.hti.add_module_subparsers(main_subparsers)

dpti.hti.compute_task(job, free_energy_type='helmholtz', method='inte', scheme='simpson',
                      manual_pv=None, manual_pv_err=None)

dpti.hti.handle_compute(args)

dpti.hti.handle_gen(args)

dpti.hti.handle_run(args)

dpti.hti.hti_phase_trans_analyze(job, jdata=None)

dpti.hti.make_iter_name(iter_index)

dpti.hti.make_tasks(iter_name, jdata, ref='einstein', switch='one-step', if_meam=None)

dpti.hti.post_tasks(iter_name, jdata, natoms=None, method='inte', scheme='s')

dpti.hti.print_thermo_info(info)

dpti.hti.refine_task(from_task, to_task, err, print_ref=False, if_meam=None, meam_model=None)

dpti.hti.run_task(task_dir, machine_file, task_name, no_dp=False)

```

4.1.7 dpti.hti_ice module

```

dpti.hti_ice.add_module_subparsers(main_subparsers)

dpti.hti_ice.add_subparsers(module_subparsers)

dpti.hti_ice.exec_args(args, parser)

dpti.hti_ice.handle_compute(args)

dpti.hti_ice.handle_gen(args)

dpti.hti_ice.handle_refine(args)

dpti.hti_ice.handle_run(args)

```

4.1.8 dpti.hti_liq module

```

dpti.hti_liq.add_module_subparsers(main_subparsers)

dpti.hti_liq.compute_task(job, free_energy_type='helmholtz', scheme='simpson', manual_pv=None,
                           manual_pv_err=None)

dpti.hti_liq.handle_compute(args)

dpti.hti_liq.handle_gen(args)

dpti.hti_liq.handle_run(args)

```

```
dpti.hti_liq.make_iter_name(iter_index)
dpti.hti_liq.make_tasks(iter_name, jdata, if_meam=None)
dpti.hti_liq.post_tasks(iter_name, natoms)
```

4.1.9 dpti.hti_water module

```
dpti.hti_water.add_module_subparsers(main_subparsers)
dpti.hti_water.add_subparsers(module_subparsers)
dpti.hti_water.compute_ideal_mol(iter_name)
dpti.hti_water.exec_args(args, parser)
dpti.hti_water.handle_compute(args)
dpti.hti_water.handle_gen(args)
dpti.hti_water.handle_refine(args)
dpti.hti_water.handle_run(args)
dpti.hti_water.make_tasks(iter_name, jdata)
dpti.hti_water.post_tasks(iter_name, natoms, method='inte', scheme='s')
dpti.hti_water.refine_tasks(from_task, to_task, err)
dpti.hti_water.spring_inte(temp, kk, r0)
```

4.1.10 dpti.main module

```
dpti.main.create_parser()
dpti.main.main()
```

4.1.11 dpti.mti module

```
dpti.mti.add_module_subparsers(main_subparsers)
dpti.mti.add_subparsers(module_subparsers)
dpti.mti.exec_args(args, parser)
dpti.mti.handle_compute(args)
dpti.mti.handle_gen(args)
dpti.mti.handle_run(args)
dpti.mti.make_tasks(iter_name, jdata)
dpti.mti.post_tasks(iter_name, jdata, natoms_mol=None)
dpti.mti.run_task(task_name, jdata, machine_file)
```

4.1.12 dpti.old_equi module

```
dpti.old_equi.extract(job_dir, output)

dpti.old_equi.make_task(iter_name, jdata, ens=None, temp=None, pres=None, avg_posi=None,
                        npt_conf=None, if_meam=None)

dpti.old_equi.npt_equi_conf(npt_name)

dpti.old_equi.post_task(iter_name, natoms=None, is_water=False)

dpti.old_equi.water_bond(iter_name, skip=1)
```

4.1.13 dpti.relax module

4.1.14 dpti.ti module

```
dpti.ti.add_module_subparsers(main_subparsers)

dpti.ti.compute_task(job, inte_method, Eo, Eo_err, To, scheme='simpson')

dpti.ti.handle_compute(args)

dpti.ti.handle_gen(args)

dpti.ti.handle_refine(args)

dpti.ti.handle_run(args)

dpti.ti.make_iter_name(iter_index)

dpti.ti.make_tasks(iter_name, jdata, if_meam=None)

dpti.ti.parse_seq_ginv(seq)

dpti.ti.post_tasks(iter_name, jdata, Eo, Eo_err=0, To=None, natoms=None, scheme='simpson', shift=0.0)

dpti.ti.post_tasks_mbar(iter_name, jdata, Eo, natoms=None)

dpti.ti.refine_task(from_task, to_task, err)

dpti.ti.run_task(task_name, machine_file)
```

4.1.15 dpti.ti_water module

```
dpti.ti_water.add_module_subparsers(main_subparsers)

dpti.ti_water.add_subparsers(module_subparsers)

dpti.ti_water.exec_args(args, parser)

dpti.ti_water.handle_compute(args)

dpti.ti_water.handle_gen(args)

dpti.ti_water.handle_refine(args)

dpti.ti_water.handle_run(args)
```


PHASE DIAGRAM OF WATER

This example describes how to calculate the phase diagram of water.

AUTHORS

- A bot of @njzjz
- Chenqqian Zhang
- Feiyang472
- Han Wang
- Jinzhe Zeng
- Liu Renxi
- Minghao Guo
- Shaochen Shi
- Wanrun Jiang
- Yifan Li
- Yuan Fengbo
- Yuan Fengbo ()
- felix5572

INDICES AND TABLES

- `genindex`
- `modindex`
- `search`

PYTHON MODULE INDEX

d

- `dpti`, 25
- `dpti.dags`, 25
- `dpti.dags.dp_ti_gdi`, 25
- `dpti.dags.utils`, 26
- `dpti.einstein`, 31
- `dpti.equi`, 31
- `dpti.gdi`, 32
- `dpti.hti`, 33
- `dpti.hti_ice`, 33
- `dpti.hti_liq`, 33
- `dpti.hti_water`, 34
- `dpti.lib`, 26
- `dpti.lib.dump`, 28
- `dpti.lib.lammps`, 29
- `dpti.lib.lmp`, 29
- `dpti.lib.RemoteJob`, 26
- `dpti.lib.utils`, 29
- `dpti.lib.vasp`, 30
- `dpti.lib.water`, 31
- `dpti.main`, 34
- `dpti.mti`, 34
- `dpti.old_equi`, 35
- `dpti.ti`, 35
- `dpti.ti_water`, 35

A

[add_bonds\(\)](#) (in module *dpti.lib.water*), 31
[add_module_subparsers\(\)](#) (in module *dpti.equi*), 31
[add_module_subparsers\(\)](#) (in module *dpti.gdi*), 32
[add_module_subparsers\(\)](#) (in module *dpti.hti*), 33
[add_module_subparsers\(\)](#) (in module *dpti.hti_ice*), 33
[add_module_subparsers\(\)](#) (in module *dpti.hti_liq*), 33
[add_module_subparsers\(\)](#) (in module *dpti.hti_water*), 34
[add_module_subparsers\(\)](#) (in module *dpti.mti*), 34
[add_module_subparsers\(\)](#) (in module *dpti.ti*), 35
[add_module_subparsers\(\)](#) (in module *dpti.ti_water*), 35
[add_subparsers\(\)](#) (in module *dpti.equi*), 31
[add_subparsers\(\)](#) (in module *dpti.hti_ice*), 33
[add_subparsers\(\)](#) (in module *dpti.hti_water*), 34
[add_subparsers\(\)](#) (in module *dpti.mti*), 34
[add_subparsers\(\)](#) (in module *dpti.ti_water*), 35

B

[block_avg\(\)](#) (in module *dpti.lib.utils*), 29
[block_call\(\)](#) (*dpti.lib.RemoteJob.RemoteJob* method), 27
[block_checkcall\(\)](#) (*dpti.lib.RemoteJob.RemoteJob* method), 27
[box2dumpbox\(\)](#) (in module *dpti.lib.dump*), 28
[box2lmpbox\(\)](#) (in module *dpti.lib.lmp*), 29

C

[check_status\(\)](#) (*dpti.lib.RemoteJob.CloudMachineJob* method), 26
[check_status\(\)](#) (*dpti.lib.RemoteJob.PBSJob* method), 27
[check_status\(\)](#) (*dpti.lib.RemoteJob.SlurmJob* method), 28
[clean\(\)](#) (*dpti.lib.RemoteJob.RemoteJob* method), 27
[close\(\)](#) (*dpti.lib.RemoteJob.SSHSession* method), 28
[CloudMachineJob](#) (class in *dpti.lib.RemoteJob*), 26
[compute_bonds\(\)](#) (in module *dpti.lib.water*), 31
[compute_ideal_mol\(\)](#) (in module *dpti.hti_water*), 34
[compute_lambda\(\)](#) (in module *dpti.einstein*), 31
[compute_nrefine\(\)](#) (in module *dpti.lib.utils*), 29

[compute_spring\(\)](#) (in module *dpti.einstein*), 31
[compute_task\(\)](#) (in module *dpti.hti*), 33
[compute_task\(\)](#) (in module *dpti.hti_liq*), 33
[compute_task\(\)](#) (in module *dpti.ti*), 35
[copy_file_list\(\)](#) (in module *dpti.lib.utils*), 29
[create_dict_not_empty_key\(\)](#) (in module *dpti.lib.utils*), 29
[create_loop_dag\(\)](#) (*dpti.dags.dp_ti_gdi.GDIDAGFactory* method), 25
[create_main_dag\(\)](#) (*dpti.dags.dp_ti_gdi.GDIDAGFactory* method), 25
[create_parser\(\)](#) (in module *dpti.main*), 34
[create_path\(\)](#) (in module *dpti.lib.utils*), 29
[cvt_conf\(\)](#) (in module *dpti.lib.utils*), 29

D

[dagargs](#) (*dpti.dags.dp_ti_gdi.GDIDAGFactory* attribute), 25
[default_args](#) (*dpti.dags.dp_ti_gdi.GDIDAGFactory* attribute), 25
[dist_via_oh_list\(\)](#) (in module *dpti.lib.water*), 31
[download\(\)](#) (*dpti.lib.RemoteJob.RemoteJob* method), 27
[dpti](#)
 module, 25
[dpti.dags](#)
 module, 25
[dpti.dags.dp_ti_gdi](#)
 module, 25
[dpti.dags.utils](#)
 module, 26
[dpti.einstein](#)
 module, 31
[dpti.equi](#)
 module, 31
[dpti.gdi](#)
 module, 32
[dpti.hti](#)
 module, 33
[dpti.hti_ice](#)
 module, 33
[dpti.hti_liq](#)
 module, 33

dpti.hti_water
 module, 34
 dpti.lib
 module, 26
 dpti.lib.dump
 module, 28
 dpti.lib.lammps
 module, 29
 dpti.lib.lmp
 module, 29
 dpti.lib.RemoteJob
 module, 26
 dpti.lib.utils
 module, 29
 dpti.lib.vasp
 module, 30
 dpti.lib.water
 module, 31
 dpti.main
 module, 34
 dpti.mti
 module, 34
 dpti.old_equi
 module, 35
 dpti.ti
 module, 35
 dpti.ti_water
 module, 35
 dumpbox2box() (in module dpti.lib.dump), 28

E

exec_args() (in module dpti.equi), 31
 exec_args() (in module dpti.hti_ice), 33
 exec_args() (in module dpti.hti_water), 34
 exec_args() (in module dpti.mti), 34
 exec_args() (in module dpti.ti_water), 35
 extract() (in module dpti.equi), 31
 extract() (in module dpti.old_equi), 35

F

finished (dpti.lib.RemoteJob.JobStatus attribute), 26
 free_energy() (in module dpti.einstein), 31
 frenkel() (in module dpti.einstein), 31
 from_system_data() (in module dpti.lib.lmp), 29

G

gdi_main_loop() (in module dpti.gdi), 32
 GDIDAGFactory (class in dpti.dags.dp_ti_gdi), 25
 GDIWorkflow (class in dpti.dags.dp_ti_gdi), 25
 gen_equi_dump_settings() (in module dpti.equi), 31
 gen_equi_ensemble_settings() (in module dpti.equi), 31
 gen_equi_force_field() (in module dpti.equi), 31

gen_equi_header() (in module dpti.equi), 31
 gen_equi_lammps_input() (in module dpti.equi), 31
 gen_equi_thermo_settings() (in module dpti.equi), 32
 get_atoms() (in module dpti.lib.lmp), 29
 get_atype() (in module dpti.lib.dump), 28
 get_atype() (in module dpti.lib.lmp), 29
 get_dag_run_state()
 (dpti.dags.dp_ti_gdi.GDIWorkflow method), 26
 get_dumpbox() (in module dpti.lib.dump), 28
 get_empty_submission() (in module dpti.dags.utils), 26
 get_file_md5() (in module dpti.lib.utils), 29
 get_first_matched_key_from_dict() (in module dpti.lib.utils), 30
 get_id() (in module dpti.lib.lmp), 29
 get_job_root() (dpti.lib.RemoteJob.RemoteJob method), 27
 get_last_dump() (in module dpti.lib.lammps), 29
 get_lmpbox() (in module dpti.lib.lmp), 29
 get_natoms() (in module dpti.lib.dump), 28
 get_natoms() (in module dpti.lib.lammps), 29
 get_natoms() (in module dpti.lib.lmp), 29
 get_natoms_vec() (in module dpti.lib.dump), 28
 get_natoms_vec() (in module dpti.lib.lmp), 29
 get_natomtypes() (in module dpti.lib.dump), 28
 get_natomtypes() (in module dpti.lib.lmp), 29
 get_posi() (in module dpti.lib.dump), 28
 get_posi() (in module dpti.lib.lmp), 29
 get_session_root() (dpti.lib.RemoteJob.SSHSession method), 28
 get_ssh_client() (dpti.lib.RemoteJob.SSHSession method), 28
 get_task_file_abspath() (in module dpti.lib.utils), 30
 get_thermo() (in module dpti.lib.lammps), 29
 get_thermo_old() (in module dpti.lib.lammps), 29
 GibbsDuhemFunc (class in dpti.gdi), 32

H

handle_compute() (in module dpti.equi), 32
 handle_compute() (in module dpti.hti), 33
 handle_compute() (in module dpti.hti_ice), 33
 handle_compute() (in module dpti.hti_liq), 33
 handle_compute() (in module dpti.hti_water), 34
 handle_compute() (in module dpti.mti), 34
 handle_compute() (in module dpti.ti), 35
 handle_compute() (in module dpti.ti_water), 35
 handle_extract() (in module dpti.equi), 32
 handle_gdi() (in module dpti.gdi), 32
 handle_gen() (in module dpti.equi), 32
 handle_gen() (in module dpti.hti), 33
 handle_gen() (in module dpti.hti_ice), 33
 handle_gen() (in module dpti.hti_liq), 33

handle_gen() (in module *dpti.hti_water*), 34
 handle_gen() (in module *dpti.mti*), 34
 handle_gen() (in module *dpti.ti*), 35
 handle_gen() (in module *dpti.ti_water*), 35
 handle_refine() (in module *dpti.hti_ice*), 33
 handle_refine() (in module *dpti.hti_water*), 34
 handle_refine() (in module *dpti.ti*), 35
 handle_refine() (in module *dpti.ti_water*), 35
 handle_run() (in module *dpti.equi*), 32
 handle_run() (in module *dpti.hti*), 33
 handle_run() (in module *dpti.hti_ice*), 33
 handle_run() (in module *dpti.hti_liq*), 33
 handle_run() (in module *dpti.hti_water*), 34
 handle_run() (in module *dpti.mti*), 34
 handle_run() (in module *dpti.ti*), 35
 handle_run() (in module *dpti.ti_water*), 35
 handle_stat_bond() (in module *dpti.equi*), 32
 hti_phase_trans_analyze() (in module *dpti.hti*), 33

I

ideal_gas_fe() (in module *dpti.einstein*), 31
 integrate() (in module *dpti.lib.utils*), 30
 integrate_range() (in module *dpti.lib.utils*), 30
 integrate_range_hti() (in module *dpti.lib.utils*), 30
 integrate_range_simpson() (in module *dpti.lib.utils*), 30
 integrate_range_trapezoidal() (in module *dpti.lib.utils*), 30
 integrate_simpson() (in module *dpti.lib.utils*), 30
 integrate_simpson_nonuniform() (in module *dpti.lib.utils*), 30
 integrate_sys_err() (in module *dpti.lib.utils*), 30
 integrate_sys_err_simpson() (in module *dpti.lib.utils*), 30
 integrate_sys_err_trapezoidal() (in module *dpti.lib.utils*), 30
 integrate_trapezoidal() (in module *dpti.lib.utils*), 30
 interval_sys_err_trapezoidal() (in module *dpti.lib.utils*), 30

J

JobStatus (class in *dpti.lib.RemoteJob*), 26

L

link_file_in_dict() (in module *dpti.lib.utils*), 30
 lmpbox2box() (in module *dpti.lib.lmp*), 29

M

main() (in module *dpti.main*), 34
 make_dpdt() (in module *dpti.gdi*), 32
 make_iter_name() (in module *dpti.hti*), 33
 make_iter_name() (in module *dpti.hti_liq*), 33

make_iter_name() (in module *dpti.lib.utils*), 30
 make_iter_name() (in module *dpti.ti*), 35
 make_task() (in module *dpti.equi*), 32
 make_task() (in module *dpti.old_equi*), 35
 make_tasks() (in module *dpti.hti*), 33
 make_tasks() (in module *dpti.hti_liq*), 34
 make_tasks() (in module *dpti.hti_water*), 34
 make_tasks() (in module *dpti.mti*), 34
 make_tasks() (in module *dpti.ti*), 35
 min_ho() (in module *dpti.lib.water*), 31
 min_oh_list() (in module *dpti.lib.water*), 31
 min_ofo() (in module *dpti.lib.water*), 31
 min_oo() (in module *dpti.lib.water*), 31

module

dpti, 25
 dpti.dags, 25
 dpti.dags.dp_ti_gdi, 25
 dpti.dags.utils, 26
 dpti.einstein, 31
 dpti.equi, 31
 dpti.gdi, 32
 dpti.hti, 33
 dpti.hti_ice, 33
 dpti.hti_liq, 33
 dpti.hti_water, 34
 dpti.lib, 26
 dpti.lib.dump, 28
 dpti.lib.lammps, 29
 dpti.lib.lmp, 29
 dpti.lib.RemoteJob, 26
 dpti.lib.utils, 29
 dpti.lib.vasp, 30
 dpti.lib.water, 31
 dpti.main, 34
 dpti.mti, 34
 dpti.old_equi, 35
 dpti.ti, 35
 dpti.ti_water, 35

N

npt_equi_conf() (in module *dpti.equi*), 32
 npt_equi_conf() (in module *dpti.old_equi*), 35

P

parse_seq() (in module *dpti.lib.utils*), 30
 parse_seq_ginv() (in module *dpti.ti*), 35
 PBSJob (class in *dpti.lib.RemoteJob*), 27
 perturb_xz() (in module *dpti.lib.vasp*), 30
 poscar_natoms() (in module *dpti.lib.vasp*), 30
 poscar_scale() (in module *dpti.lib.vasp*), 30
 poscar_vol() (in module *dpti.lib.vasp*), 30
 posi_diff() (in module *dpti.lib.water*), 31
 posi_shift() (in module *dpti.lib.water*), 31
 post_task() (in module *dpti.equi*), 32

`post_task()` (in module `dpti.old_equi`), 35
`post_tasks()` (in module `dpti.hti`), 33
`post_tasks()` (in module `dpti.hti_liq`), 34
`post_tasks()` (in module `dpti.hti_water`), 34
`post_tasks()` (in module `dpti.mti`), 34
`post_tasks()` (in module `dpti.ti`), 35
`post_tasks_mbar()` (in module `dpti.ti`), 35
`print_thermo_info()` (in module `dpti.hti`), 33

R

`reciprocal_box()` (in module `dpti.lib.vasp`), 30
`refine_task()` (in module `dpti.hti`), 33
`refine_task()` (in module `dpti.ti`), 35
`refine_tasks()` (in module `dpti.hti_water`), 34
`regulate_poscar()` (in module `dpti.lib.vasp`), 30
`relative_link_file()` (in module `dpti.lib.utils`), 30
`RemoteJob` (class in `dpti.lib.RemoteJob`), 27
`run_task()` (in module `dpti.equi`), 32
`run_task()` (in module `dpti.hti`), 33
`run_task()` (in module `dpti.mti`), 34
`run_task()` (in module `dpti.ti`), 35
`running` (`dpti.lib.RemoteJob.JobStatus` attribute), 26

S

`SlurmJob` (class in `dpti.lib.RemoteJob`), 28
`sort_poscar()` (in module `dpti.lib.vasp`), 30
`split_traj()` (in module `dpti.lib.dump`), 28
`spring_inte()` (in module `dpti.hti_water`), 34
`SSHSession` (class in `dpti.lib.RemoteJob`), 27
`submit()` (`dpti.lib.RemoteJob.CloudMachineJob` method), 26
`submit()` (`dpti.lib.RemoteJob.PBSJob` method), 27
`submit()` (`dpti.lib.RemoteJob.SlurmJob` method), 28
`system_data()` (in module `dpti.lib.dump`), 28
`system_data()` (in module `dpti.lib.lmp`), 29

T

`terminated` (`dpti.lib.RemoteJob.JobStatus` attribute), 26
`to_system_data()` (in module `dpti.lib.lmp`), 29
`trigger_loop()` (`dpti.dags.dp_ti_gdi.GDIWorkflow` method), 26

U

`unknown` (`dpti.lib.RemoteJob.JobStatus` attribute), 27
`unsubmitted` (`dpti.lib.RemoteJob.JobStatus` attribute), 27
`upload()` (`dpti.lib.RemoteJob.RemoteJob` method), 27

W

`wait_until_end()` (`dpti.dags.dp_ti_gdi.GDIWorkflow` method), 26
`waiting` (`dpti.lib.RemoteJob.JobStatus` attribute), 27
`water_bond()` (in module `dpti.equi`), 32
`water_bond()` (in module `dpti.old_equi`), 35