dpdata Documentation

Release 0.0.0-rc

Han Wang

Apr 03, 2024

CONTENTS:

CHAPTER

ONE

TRY DPDATA ONLINE

CHAPTER

TWO

COMMAND LINE INTERFACE

dpdata: Manipulating multiple atomic simulation data formats

```
usage: dpdata [-h] [--to_file TO_FILE] [--from_format FROM_FORMAT]
              [--to_format TO_FORMAT] [--no-labeled] [--multi]
              [--type-map TYPE_MAP [TYPE_MAP ...]] [--version]
              from_file
```

2.1 Positional Arguments

from_file	read data from a file

2.2 Named Arguments

--to_file, -O	dump data to a file
--from_format, -i	the format of from_file
	Default: "auto"
--to_format, -o	the format of to_file
--no-labeled, -n	labels aren't provided
	Default: False
--multi, -m	the system contains multiple directories
	Default: False
--type-map, -t	type map
--version	show program's version number and exit

CHAPTER

THREE

SUPPORTED FORMATS

dpdata supports the following formats:

Table 1: Supported Formats

Format	Alias	Support
psi4/out format	psi4/out	*LabeledS*
psi4/inp format	psi4/inp	*System.to*
cp2k/aimd_output format	cp2k/aimd_output	*LabeledS*
cp2k/output format	cp2k/output	*LabeledS*
orca/spout format	orca/spout	*LabeledS*
stru format	stru abacus/stru	*System()*
abacus/lcao/scf format	abacus/lcao/scf abacus/pw/scf abacus/scf	*LabeledS*
abacus/lcao/md format	abacus/lcao/md abacus/pw/md abacus/md	*LabeledS*
abacus/lcao/relax format	abacus/lcao/relax abacus/pw/relax abacus/relax	*LabeledS*
vasp/contcar format	vasp/contcar vasp/poscar contcar poscar	*System()*
vasp/string format	vasp/string	*System.to*
vasp/outcar format	vasp/outcar outcar	*LabeledS*
vasp/xml format	vasp/xml xml	*LabeledS*
deepmd/raw format	deepmd/raw deepmd	*System()*
deepmd/comp format	deepmd/comp deepmd/npy	*System()*
deepmd/npy/mixed format	deepmd/npy/mixed	*System.to*
deepmd/hdf5 format	deepmd/hdf5	*System()*
siesta/output format	siesta/output	*System()*
siesta/aimd_output format	siesta/aimd_output	*System()*
pymatgen/structure format	pymatgen/structure	*System.to*
pymatgen/molecule format	pymatgen/molecule	*System()*
pymatgen/computedstructureentry format	pymatgen/computedstructureentry	*LabeledS*
list format	list	*System.to*
mol_file format	mol_file mol	*BondOrd*
sdf_file format	sdf_file sdf	*BondOrd*
lammps/lmp format	lammps/lmp lmp	*System()*
lammps/dump format	lammps/dump dump	*System()*
pwmat/output format	pwmat/output pwmat/mlmd pwmat/movement mlmd movement	*LabeledS*
pwmat/final.config format	pwmat/final.config pwmat/atom.config final.config atom.config	*System()*
ase/structure format	ase/structure	*System()*
ase/traj format	ase/traj	*System()*
gaussian/log format	gaussian/log	*LabeledS*
gaussian/md format	gaussian/md	*LabeledS*
gaussian/gjf format	gaussian/gjf	*System()*
fhi_aims/output format	fhi_aims/output fhi_aims/md	*LabeledS*

Table 1 – continued from previou

Format	Alias	Support
fhi_aims/scf format	`fhi_aims/scf`	*LabeledS*
qe/cp/traj format	`qe/cp/traj`	*System()*
qe/pw/scf format	`qe/pw/scf`	*LabeledS*
3dmol format	`3dmol`	*System.to*
xyz format	`xyz`	*System()*
quip/gap/xyz_file format	`quip/gap/xyz_file` `quip/gap/xyz`	*LabeledS*
gromacs/gro format	`gromacs/gro` `gro`	*System()*
n2p2 format	`n2p2`	*LabeledS*
dftbplus format	`dftbplus`	*LabeledS*
amber/md format	`amber/md`	*System()*
sqm/out format	`sqm/out`	*System()*
sqm/in format	`sqm/in`	*System.to*
openmx/md format	`openmx/md`	*System()*

3.1 ase/structure format

Class: *ASEStructureFormat*

Format for the Atomic Simulation Environment (ase).

ASE supports parsing a few dozen of data formats. As described in i the documentation, many of these formats can be determined automatically. Use the *ase_fmt* keyword argument to supply the format if automatic detection fails.

3.1.1 Conversions

Convert from this format to System

`dpdata.System`(*atoms: 'ase.Atoms'*, *fmt: Literal['ase/structure'] = None*) → *dpdata.system.System*

`dpdata.System.from_ase_structure`(*atoms: 'ase.Atoms'*) → *dpdata.system.System*

Convert ase.Atoms to a System.

Parameters

atoms
[ase.Atoms] an ASE Atoms, containing a structure

Returns

System
converted system

Convert from System to this format

dpdata.System.**to**(*fmt: Literal['ase/structure']*)

dpdata.System.**to_ase_structure**()

Convert System to ASE Atom obj.

Convert from LabeledSystem to this format

dpdata.LabeledSystem.**to**(*fmt: Literal['ase/structure']*)

dpdata.LabeledSystem.**to_ase_structure**()

Convert System to ASE Atoms object.

Convert from this format to LabeledSystem

dpdata.**LabeledSystem**(*atoms: 'ase.Atoms'*, *fmt: Literal['ase/structure'] = None*) → *dpdata.system.LabeledSystem*

dpdata.LabeledSystem.**from_ase_structure**(*atoms: 'ase.Atoms'*) → *dpdata.system.LabeledSystem*

Convert ase.Atoms to a LabeledSystem. Energies and forces are calculated by the calculator.

Parameters

atoms
[ase.Atoms] an ASE Atoms, containing a structure

Returns

LabeledSystem
converted system

Raises

RuntimeError
ASE will raise RuntimeError if the atoms does not have a calculator

Convert from this format to MultiSystems

dpdata.MultiSystems.**from_ase_structure**(*file_name: str*, *begin: int | None = None*, *end: int | None = None*, *step: int | None = None*, *ase_fmt: str | None = None*) → *dpdata.system.MultiSystems*

Convert a ASE supported file to ASE Atoms.

It will finally be converted to MultiSystems.

Parameters

file_name
[str] path to file

begin
[int, optional] begin frame index

end
[int, optional] end frame index

step
[int, optional] frame index step

ase_fmt
[str, optional] ASE format. See the ASE documentation about supported formats

Returns

MultiSystems
converted system

3.2 ase/traj format

Class: *ASETrajFormat*

Format for the ASE's trajectory format <https://wiki.fysik.dtu.dk/ase/ase/io/trajectory.html#module-ase.io.trajectory>`_ (ase).' a `traj' contains a sequence of frames, each of which is an `Atoms' object.

3.2.1 Conversions

Convert from this format to System

dpdata.**System**(*file_name: str*, *begin: int | None = 0*, *end: int | None = None*, *step: int | None = 1*, *fmt: Literal['ase/traj'] = None*) → *dpdata.system.System*

dpdata.System.**from_ase_traj**(*file_name: str*, *begin: int | None = 0*, *end: int | None = None*, *step: int | None = 1*) → *dpdata.system.System*

Read ASE's trajectory file to *System* of multiple frames.

Parameters

file_name
[str] ASE's trajectory file

begin
[int, optional] begin frame index

end
[int, optional] end frame index

step
[int, optional] frame index step

Returns

System
converted system

Convert from this format to LabeledSystem

dpdata.**LabeledSystem**(*file_name: str*, *begin: int | None = 0*, *end: int | None = None*, *step: int | None = 1*, *fmt: Literal['ase/traj'] = None*) → *dpdata.system.LabeledSystem*

dpdata.LabeledSystem.**from_ase_traj**(*file_name: str*, *begin: int | None = 0*, *end: int | None = None*, *step: int | None = 1*) → *dpdata.system.LabeledSystem*

Read ASE's trajectory file to *System* of multiple frames.

Parameters

file_name
[str] ASE's trajectory file

begin
[int, optional] begin frame index

end
[int, optional] end frame index

step
[int, optional] frame index step

Returns

LabeledSystem
converted system

3.3 abacus/lcao/md format

3.4 abacus/pw/md format

3.5 abacus/md format

Class: *AbacusMDFormat*

3.5.1 Conversions

Convert from this format to LabeledSystem

dpdata.**LabeledSystem**(*file_name*, *fmt: Literal['abacus/lcao/md'] = None*) → *dpdata.system.LabeledSystem*

dpdata.**LabeledSystem**(*file_name*, *fmt: Literal['abacus/pw/md'] = None*) → *dpdata.system.LabeledSystem*

dpdata.**LabeledSystem**(*file_name*, *fmt: Literal['abacus/md'] = None*) → *dpdata.system.LabeledSystem*

dpdata.LabeledSystem.**from_abacus_lcao_md**(*file_name*) → *dpdata.system.LabeledSystem*

dpdata.LabeledSystem.**from_abacus_pw_md**(*file_name*) → *dpdata.system.LabeledSystem*

dpdata.LabeledSystem.**from_abacus_md**(*file_name*) → *dpdata.system.LabeledSystem*

Convert this format to LabeledSystem.

Returns

LabeledSystem
converted system

3.6 abacus/lcao/relax format

3.7 abacus/pw/relax format

3.8 abacus/relax format

Class: *AbacusRelaxFormat*

3.8.1 Conversions

Convert from this format to LabeledSystem

dpdata.**LabeledSystem**(*file_name*, *fmt: Literal['abacus/lcao/relax'] = None*) → *dpdata.system.LabeledSystem*

dpdata.**LabeledSystem**(*file_name*, *fmt: Literal['abacus/pw/relax'] = None*) → *dpdata.system.LabeledSystem*

dpdata.**LabeledSystem**(*file_name*, *fmt: Literal['abacus/relax'] = None*) → *dpdata.system.LabeledSystem*

dpdata.LabeledSystem.**from_abacus_lcao_relax**(*file_name*) → *dpdata.system.LabeledSystem*

dpdata.LabeledSystem.**from_abacus_pw_relax**(*file_name*) → *dpdata.system.LabeledSystem*

dpdata.LabeledSystem.**from_abacus_relax**(*file_name*) → *dpdata.system.LabeledSystem*

Convert this format to LabeledSystem.

Returns

LabeledSystem
converted system

3.9 abacus/lcao/scf format

3.10 abacus/pw/scf format

3.11 abacus/scf format

Class: *AbacusSCFFormat*

3.11.1 Conversions

Convert from this format to LabeledSystem

dpdata.**LabeledSystem**(*file_name*, *fmt: Literal['abacus/lcao/scf'] = None*) → *dpdata.system.LabeledSystem*

dpdata.**LabeledSystem**(*file_name*, *fmt: Literal['abacus/pw/scf'] = None*) → *dpdata.system.LabeledSystem*

dpdata.**LabeledSystem**(*file_name*, *fmt: Literal['abacus/scf'] = None*) → *dpdata.system.LabeledSystem*

dpdata.LabeledSystem.**from_abacus_lcao_scf**(*file_name*) → *dpdata.system.LabeledSystem*

dpdata.LabeledSystem.**from_abacus_pw_scf**(*file_name*) → *dpdata.system.LabeledSystem*

dpdata.LabeledSystem.**from_abacus_scf**(*file_name*) → *dpdata.system.LabeledSystem*

Convert this format to `LabeledSystem`.

Returns

LabeledSystem
converted system

3.12 stru format

3.13 abacus/stru format

Class: *AbacusSTRUFormat*

3.13.1 Conversions

Convert from this format to System

dpdata.**System**(*file_name*, *fmt: Literal['stru'] = None*) → *dpdata.system.System*

dpdata.**System**(*file_name*, *fmt: Literal['abacus/stru'] = None*) → *dpdata.system.System*

dpdata.System.**from_stru**(*file_name*) → *dpdata.system.System*

dpdata.System.**from_abacus_stru**(*file_name*) → *dpdata.system.System*

Convert this format to `System`.

Returns

System
converted system

Convert from System to this format

dpdata.System.**to**(*fmt: Literal['stru']*, *file_name*, *frame_idx=0*)

dpdata.System.**to**(*fmt: Literal['stru']*, *file_name*, *frame_idx=0*)

dpdata.System.**to_stru**(*file_name*, *frame_idx=0*)

dpdata.System.**to_abacus_stru**(*file_name*, *frame_idx=0*)

Dump the system into ABACUS STRU format file.

Parameters

file_name
[str] The output file name

frame_idx
[int] The index of the frame to dump

pp_file
[list of string, optional] List of pseudo potential files

numerical_orbital
[list of string, optional] List of orbital files

mass
[list of float, optional] List of atomic masses

numerical_descriptor
[str, optional] numerical descriptor file

Convert from LabeledSystem to this format

dpdata.LabeledSystem.**to**(*fmt: Literal['stru']*, *file_name*, *frame_idx=0*)

dpdata.LabeledSystem.**to**(*fmt: Literal['stru']*, *file_name*, *frame_idx=0*)

dpdata.LabeledSystem.**to_stru**(*file_name*, *frame_idx=0*)

dpdata.LabeledSystem.**to_abacus_stru**(*file_name*, *frame_idx=0*)

Dump the system into ABACUS STRU format file.

Parameters

file_name
[str] The output file name

frame_idx
[int] The index of the frame to dump

pp_file
[list of string, optional] List of pseudo potential files

numerical_orbital
[list of string, optional] List of orbital files

mass
[list of float, optional] List of atomic masses

numerical_descriptor
[str, optional] numerical descriptor file

3.14 amber/md format

Class: *AmberMDFormat*

3.14.1 Conversions

Convert from this format to System

dpdata.**System**(*file_name=None*, *parm7_file=None*, *nc_file=None*, *use_element_symbols=None*, *fmt: Literal['amber/md'] = None*) → *dpdata.system.System*

dpdata.System.**from_amber_md**(*file_name=None*, *parm7_file=None*, *nc_file=None*, *use_element_symbols=None*) → *dpdata.system.System*

Convert this format to `System`.

Returns

System
converted system

Convert from this format to LabeledSystem

dpdata.**LabeledSystem**(*file_name=None*, *parm7_file=None*, *nc_file=None*, *mdfrc_file=None*, *mden_file=None*, *mdout_file=None*, *use_element_symbols=None*, *fmt: Literal['amber/md'] = None*) → *dpdata.system.LabeledSystem*

dpdata.LabeledSystem.**from_amber_md**(*file_name=None*, *parm7_file=None*, *nc_file=None*, *mdfrc_file=None*, *mden_file=None*, *mdout_file=None*, *use_element_symbols=None*) → *dpdata.system.LabeledSystem*

Convert this format to `LabeledSystem`.

Returns

LabeledSystem
converted system

3.15 cp2k/aimd_output format

Class: *CP2KAIMDOutputFormat*

3.15.1 Conversions

Convert from this format to LabeledSystem

dpdata.**LabeledSystem**(*file_name*, *restart=False*, *fmt: Literal['cp2k/aimd_output'] = None*) → *dpdata.system.LabeledSystem*

dpdata.LabeledSystem.**from_cp2k_aimd_output**(*file_name*, *restart=False*) → *dpdata.system.LabeledSystem*

Convert this format to `LabeledSystem`.

Returns

LabeledSystem
converted system

3.16 cp2k/output format

Class: *CP2KOutputFormat*

3.16.1 Conversions

Convert from this format to LabeledSystem

dpdata.**LabeledSystem**(*file_name*, *restart=False*, *fmt: Literal['cp2k/output'] = None*) → *dpdata.system.LabeledSystem*

dpdata.LabeledSystem.**from_cp2k_output**(*file_name*, *restart=False*) → *dpdata.system.LabeledSystem*

Convert this format to `LabeledSystem`.

Returns

LabeledSystem
converted system

3.17 dftbplus format

Class: *DFTBplusFormat*

The DFTBplusFormat class handles files in the DFTB+ format.

This class provides a method to read DFTB+ files from a labeled system and returns a dictionary containing various properties of the system.For more information, please refer to the official documentation at the following URL: https://dftbplus.org/documentation

Attributes

None

Methods

from_labeled_system(file_paths, **kwargs): Reads system information from files.

3.17.1 Conversions

Convert from this format to LabeledSystem

dpdata.**LabeledSystem**(*file_paths*, *fmt: Literal['dftbplus'] = None*) → *dpdata.system.LabeledSystem*

dpdata.LabeledSystem.**from_dftbplus**(*file_paths*) → *dpdata.system.LabeledSystem*

Reads system information from the given DFTB+ file paths.

Parameters

file_paths
[tuple] A tuple containing the input and output file paths. - Input file (file_in): Contains information about symbols and coord. - Output file (file_out): Contains information about energy and force.

Returns

LabeledSystem
converted system

3.18 deepmd/comp format

3.19 deepmd/npy format

Class: *DeePMDCompFormat*

3.19.1 Conversions

Convert from this format to System

dpdata.**System**(*file_name*, *type_map=None*, *fmt: Literal['deepmd/comp'] = None*) → *dpdata.system.System*

dpdata.**System**(*file_name*, *type_map=None*, *fmt: Literal['deepmd/npy'] = None*) → *dpdata.system.System*

dpdata.System.**from_deepmd_comp**(*file_name*, *type_map=None*) → *dpdata.system.System*

dpdata.System.**from_deepmd_npy**(*file_name*, *type_map=None*) → *dpdata.system.System*

Convert this format to System.

Returns

System
converted system

Convert from System to this format

dpdata.System.**to**(*fmt: Literal['deepmd/comp'], file_name, set_size=5000, prec=<class 'numpy.float64'>*)

dpdata.System.**to**(*fmt: Literal['deepmd/comp'], file_name, set_size=5000, prec=<class 'numpy.float64'>*)

dpdata.System.**to_deepmd_comp**(*file_name*, *set_size=5000*, *prec=<class 'numpy.float64'>*)

dpdata.System.**to_deepmd_npy**(*file_name*, *set_size=5000*, *prec=<class 'numpy.float64'>*)

Dump the system in deepmd compressed format (numpy binary) to *folder*.

The frames are firstly split to sets, then dumped to seperated subfolders named as *folder/set.000*, *folder/set.001*, ...

Each set contains *set_size* frames. The last set may have less frames than *set_size*.

Parameters

file_name
[str] The output folder

set_size
[int] The size of each set.

prec
[{numpy.float32, numpy.float64}] The floating point precision of the compressed data

Convert from LabeledSystem to this format

dpdata.LabeledSystem.**to**(*fmt: Literal['deepmd/comp'], file_name, set_size=5000, prec=<class 'numpy.float64'>*)

dpdata.LabeledSystem.**to**(*fmt: Literal['deepmd/comp'], file_name, set_size=5000, prec=<class 'numpy.float64'>*)

dpdata.LabeledSystem.**to_deepmd_comp**(*file_name*, *set_size=5000*, *prec=<class 'numpy.float64'>*)

dpdata.LabeledSystem.**to_deepmd_npy**(*file_name*, *set_size=5000*, *prec=<class 'numpy.float64'>*)

Dump the system in deepmd compressed format (numpy binary) to *folder*.

The frames are firstly split to sets, then dumped to seperated subfolders named as *folder/set.000*, *folder/set.001*, ...

Each set contains *set_size* frames. The last set may have less frames than *set_size*.

Parameters

file_name
[str] The output folder

set_size
[int] The size of each set.

prec
[{numpy.float32, numpy.float64}] The floating point precision of the compressed data

Convert from this format to LabeledSystem

dpdata.**LabeledSystem**(*file_name*, *type_map=None*, *fmt: Literal['deepmd/comp'] = None*) → *dpdata.system.LabeledSystem*

dpdata.**LabeledSystem**(*file_name*, *type_map=None*, *fmt: Literal['deepmd/npy'] = None*) → *dpdata.system.LabeledSystem*

dpdata.LabeledSystem.**from_deepmd_comp**(*file_name*, *type_map=None*) → *dpdata.system.LabeledSystem*

dpdata.LabeledSystem.**from_deepmd_npy**(*file_name*, *type_map=None*) → *dpdata.system.LabeledSystem*

Convert this format to `LabeledSystem`.

Returns

LabeledSystem
converted system

Convert from this format to MultiSystems

dpdata.MultiSystems.**from_deepmd_comp**(*directory*) → *dpdata.system.MultiSystems*

dpdata.MultiSystems.**from_deepmd_npy**(*directory*) → *dpdata.system.MultiSystems*

Convert this format to `MultiSystems`.

Parameters

directory
[str] directory of systems

Returns

MultiSystems
converted system

Convert from MultiSystems to this format

dpdata.MultiSystems.**to**(*fmt: Literal['deepmd/comp']*, *directory*) → *dpdata.system.MultiSystems*

dpdata.MultiSystems.**to**(*fmt: Literal['deepmd/comp']*, *directory*) → *dpdata.system.MultiSystems*

dpdata.MultiSystems.**to_deepmd_comp**(*directory*) → *dpdata.system.MultiSystems*

dpdata.MultiSystems.**to_deepmd_npy**(*directory*) → *dpdata.system.MultiSystems*

Convert `MultiSystems` to this format.

Parameters

directory
[str] directory to save systems

Returns

MultiSystems
this system

3.20 deepmd/hdf5 format

Class: *DeePMDHDF5Format*

HDF5 format for DeePMD-kit.

Examples

Dump a MultiSystems to a HDF5 file:

```
>>> import dpdata
>>> dpdata.MultiSystems().from_deepmd_npy("data").to_deepmd_hdf5("data.hdf5")
```

3.20.1 Conversions

Convert from this format to System

dpdata.**System**(*file_name: 'str | (h5py.Group | h5py.File)'*, *type_map: 'list[str] | None' = None*, *fmt: Literal['deepmd/hdf5'] = None*) → *dpdata.system.System*

dpdata.System.**from_deepmd_hdf5**(*file_name: 'str | (h5py.Group | h5py.File)'*, *type_map: 'list[str] | None' = None*) → *dpdata.system.System*

Convert HDF5 file to System data.

Parameters

file_name
[str or h5py.Group or h5py.File] file name of the HDF5 file or HDF5 object. If it is a string, hashtag is used to split path to the HDF5 file and the HDF5 group

type_map
[dict[str]] type map

Returns

System
converted system

Raises

TypeError
file_name is not str or h5py.Group or h5py.File

Convert from System to this format

dpdata.System.**to**(*fmt: Literal['deepmd/hdf5']*, *file_name: 'str | (h5py.Group | h5py.File)'*, *set_size: 'int' = 5000*, *comp_prec: 'np.dtype' = <class 'numpy.float64'>*)

dpdata.System.**to_deepmd_hdf5**(*file_name: 'str | (h5py.Group | h5py.File)'*, *set_size: 'int' = 5000*, *comp_prec: 'np.dtype' = <class 'numpy.float64'>*)

Convert System data to HDF5 file.

Parameters

file_name
[str or h5py.Group or h5py.File] file name of the HDF5 file or HDF5 object. If it is a string, hashtag is used to split path to the HDF5 file and the HDF5 group

set_size
[int, default=5000] set size

comp_prec
[np.dtype] data precision

Convert from LabeledSystem to this format

dpdata.LabeledSystem.**to**(*fmt: Literal['deepmd/hdf5'], file_name: 'str | (h5py.Group | h5py.File)', set_size: 'int' = 5000, comp_prec: 'np.dtype' = <class 'numpy.float64'>*)

dpdata.LabeledSystem.**to_deepmd_hdf5**(*file_name: 'str | (h5py.Group | h5py.File)', set_size: 'int' = 5000, comp_prec: 'np.dtype' = <class 'numpy.float64'>*)

Convert System data to HDF5 file.

Parameters

file_name
[str or h5py.Group or h5py.File] file name of the HDF5 file or HDF5 object. If it is a string, hashtag is used to split path to the HDF5 file and the HDF5 group

set_size
[int, default=5000] set size

comp_prec
[np.dtype] data precision

Convert from this format to LabeledSystem

dpdata.**LabeledSystem**(*file_name: 'str | (h5py.Group | h5py.File)', type_map: 'list[str] | None' = None, fmt: Literal['deepmd/hdf5'] = None*) → *dpdata.system.LabeledSystem*

dpdata.LabeledSystem.**from_deepmd_hdf5**(*file_name: 'str | (h5py.Group | h5py.File)', type_map: 'list[str] | None' = None*) → *dpdata.system.LabeledSystem*

Convert HDF5 file to LabeledSystem data.

Parameters

file_name
[str or h5py.Group or h5py.File] file name of the HDF5 file or HDF5 object. If it is a string, hashtag is used to split path to the HDF5 file and the HDF5 group

type_map
[dict[str]] type map

Returns

LabeledSystem
converted system

Raises

TypeError
file_name is not str or h5py.Group or h5py.File

Convert from this format to MultiSystems

dpdata.MultiSystems.**from_deepmd_hdf5**(*directory: 'str'*) → *dpdata.system.MultiSystems*

Generate HDF5 groups from a HDF5 file, which will be passed to *from_system*.

Parameters

directory
[str] HDF5 file name

Returns

MultiSystems
converted system

Convert from MultiSystems to this format

dpdata.MultiSystems.**to**(*fmt: Literal['deepmd/hdf5']*, *directory: 'str'*) → *dpdata.system.MultiSystems*

dpdata.MultiSystems.**to_deepmd_hdf5**(*directory: 'str'*) → *dpdata.system.MultiSystems*

Generate HDF5 groups, which will be passed to *to_system*.

Parameters

directory
[str] HDF5 file name

Returns

MultiSystems
this system

3.21 deepmd/npy/mixed format

Class: *DeePMDMixedFormat*

Mixed type numpy format for DeePMD-kit. Under this format, systems with the same number of atoms but different formula can be put together for a larger system, especially when the frame numbers in systems are sparse. This also helps to mixture the type information together for model training with type embedding network.

Examples

Dump a MultiSystems into a mixed type numpy directory:

```
>>> import dpdata
>>> dpdata.MultiSystems(*systems).to_deepmd_npy_mixed("mixed_dir")
```

Load a mixed type data into a MultiSystems:

```
>>> import dpdata
>>> dpdata.MultiSystems().load_systems_from_file("mixed_dir", fmt="deepmd/npy/mixed")
```

3.21.1 Conversions

Convert from System to this format

dpdata.System.**to**(*fmt: Literal['deepmd/npy/mixed'], file_name, set_size: 'int' = 2000, prec=<class 'numpy.float64'>*)

dpdata.System.**to_deepmd_npy_mixed**(*file_name*, *set_size: 'int' = 2000*, *prec=<class 'numpy.float64'>*)

Dump the system in deepmd mixed type format (numpy binary) to *folder*.

The frames were already split to different systems, so these frames can be dumped to one single subfolders named as *folder/set.000*, containing less than *set_size* frames.

Parameters

file_name
[str] The output folder

set_size
[int, default=2000] set size

prec
[{numpy.float32, numpy.float64}] The floating point precision of the compressed data

Convert from LabeledSystem to this format

dpdata.LabeledSystem.**to**(*fmt: Literal['deepmd/npy/mixed'], file_name, set_size: 'int' = 2000, prec=<class 'numpy.float64'>*)

dpdata.LabeledSystem.**to_deepmd_npy_mixed**(*file_name*, *set_size: 'int' = 2000*, *prec=<class 'numpy.float64'>*)

Dump the system in deepmd mixed type format (numpy binary) to *folder*.

The frames were already split to different systems, so these frames can be dumped to one single subfolders named as *folder/set.000*, containing less than *set_size* frames.

Parameters

file_name
[str] The output folder

set_size
[int, default=2000] set size

prec
[{numpy.float32, numpy.float64}] The floating point precision of the compressed data

Convert from this format to MultiSystems

dpdata.MultiSystems.**from_deepmd_npy_mixed**(*directory*) → *dpdata.system.MultiSystems*

Convert this format to `MultiSystems`.

Parameters

directory
[str] directory of systems

Returns

MultiSystems
converted system

Convert from MultiSystems to this format

dpdata.MultiSystems.**to**(*fmt: Literal['deepmd/npy/mixed']*, *directory*) → *dpdata.system.MultiSystems*

dpdata.MultiSystems.**to_deepmd_npy_mixed**(*directory*) → *dpdata.system.MultiSystems*

Convert `MultiSystems` to this format.

Parameters

directory
[str] directory to save systems

Returns

MultiSystems
this system

3.22 deepmd/raw format

3.23 deepmd format

Class: *DeePMDRawFormat*

3.23.1 Conversions

Convert from this format to System

dpdata.**System**(*file_name*, *type_map=None*, *fmt: Literal['deepmd/raw'] = None*) → *dpdata.system.System*

dpdata.**System**(*file_name*, *type_map=None*, *fmt: Literal['deepmd'] = None*) → *dpdata.system.System*

dpdata.System.**from_deepmd_raw**(*file_name*, *type_map=None*) → *dpdata.system.System*

dpdata.System.**from_deepmd**(*file_name*, *type_map=None*) → *dpdata.system.System*

Convert this format to `System`.

Returns

System
converted system

Convert from System to this format

dpdata.System.**to**(*fmt: Literal['deepmd/raw']*, *file_name*)

dpdata.System.**to**(*fmt: Literal['deepmd/raw']*, *file_name*)

dpdata.System.**to_deepmd_raw**(*file_name*)

dpdata.System.**to_deepmd**(*file_name*)

Dump the system in deepmd raw format to directory *file_name*.

Convert from LabeledSystem to this format

dpdata.LabeledSystem.**to**(*fmt: Literal['deepmd/raw']*, *file_name*)

dpdata.LabeledSystem.**to**(*fmt: Literal['deepmd/raw']*, *file_name*)

dpdata.LabeledSystem.**to_deepmd_raw**(*file_name*)

dpdata.LabeledSystem.**to_deepmd**(*file_name*)

Dump the system in deepmd raw format to directory *file_name*.

Convert from this format to LabeledSystem

dpdata.**LabeledSystem**(*file_name*, *type_map=None*, *fmt: Literal['deepmd/raw'] = None*) → *dpdata.system.LabeledSystem*

dpdata.**LabeledSystem**(*file_name*, *type_map=None*, *fmt: Literal['deepmd'] = None*) → *dpdata.system.LabeledSystem*

dpdata.LabeledSystem.**from_deepmd_raw**(*file_name*, *type_map=None*) → *dpdata.system.LabeledSystem*

dpdata.LabeledSystem.**from_deepmd**(*file_name*, *type_map=None*) → *dpdata.system.LabeledSystem*

Convert this format to `LabeledSystem`.

Returns

LabeledSystem
converted system

Convert from this format to MultiSystems

dpdata.MultiSystems.**from_deepmd_raw**(*directory*) → *dpdata.system.MultiSystems*

dpdata.MultiSystems.**from_deepmd**(*directory*) → *dpdata.system.MultiSystems*

Convert this format to `MultiSystems`.

Parameters

directory
[str] directory of systems

Returns

MultiSystems
converted system

Convert from MultiSystems to this format

dpdata.MultiSystems.**to**(*fmt: Literal['deepmd/raw']*, *directory*) → *dpdata.system.MultiSystems*

dpdata.MultiSystems.**to**(*fmt: Literal['deepmd/raw']*, *directory*) → *dpdata.system.MultiSystems*

dpdata.MultiSystems.**to_deepmd_raw**(*directory*) → *dpdata.system.MultiSystems*

dpdata.MultiSystems.**to_deepmd**(*directory*) → *dpdata.system.MultiSystems*

Convert `MultiSystems` to this format.

Parameters

directory
[str] directory to save systems

Returns

MultiSystems
this system

3.24 fhi_aims/output format

3.25 fhi_aims/md format

Class: *FhiMDFormat*

3.25.1 Conversions

Convert from this format to LabeledSystem

dpdata.**LabeledSystem**(*file_name*, *md=True*, *begin=0*, *step=1*, *convergence_check=True*, *fmt: Literal['fhi_aims/output'] = None*) → *dpdata.system.LabeledSystem*

dpdata.**LabeledSystem**(*file_name*, *md=True*, *begin=0*, *step=1*, *convergence_check=True*, *fmt: Literal['fhi_aims/md'] = None*) → *dpdata.system.LabeledSystem*

dpdata.LabeledSystem.**from_fhi_aims_output**(*file_name*, *md=True*, *begin=0*, *step=1*, *convergence_check=True*) → *dpdata.system.LabeledSystem*

dpdata.LabeledSystem.**from_fhi_aims_md**(*file_name*, *md=True*, *begin=0*, *step=1*, *convergence_check=True*) → *dpdata.system.LabeledSystem*

Convert this format to `LabeledSystem`.

Returns

LabeledSystem
converted system

3.26 fhi_aims/scf format

Class: *FhiSCFFormat*

3.26.1 Conversions

Convert from this format to LabeledSystem

dpdata.**LabeledSystem**(*file_name*, *fmt: Literal['fhi_aims/scf'] = None*) → *dpdata.system.LabeledSystem*

dpdata.LabeledSystem.**from_fhi_aims_scf**(*file_name*) → *dpdata.system.LabeledSystem*

Convert this format to LabeledSystem.

Returns

LabeledSystem
converted system

3.27 gaussian/gjf format

Class: *GaussiaGJFFormat*

Gaussian input file.

3.27.1 Conversions

Convert from this format to System

dpdata.**System**(*file_name: str*, *fmt: Literal['gaussian/gjf'] = None*) → *dpdata.system.System*

dpdata.System.**from_gaussian_gjf**(*file_name: str*) → *dpdata.system.System*

Read Gaussian input file.

Parameters

file_name
[str] file name

Returns

System
converted system

Convert from System to this format

dpdata.System.**to**(*fmt: Literal['gaussian/gjf']*, *file_name: str*)

dpdata.System.**to_gaussian_gjf**(*file_name: str*)

Generate Gaussian input file.

Parameters

file_name
[str] file name

Convert from LabeledSystem to this format

dpdata.LabeledSystem.**to**(*fmt: Literal['gaussian/gjf']*, *file_name: str*)

dpdata.LabeledSystem.**to_gaussian_gjf**(*file_name: str*)

Generate Gaussian input file.

Parameters

file_name
[str] file name

3.28 gaussian/log format

Class: *GaussianLogFormat*

3.28.1 Conversions

Convert from this format to LabeledSystem

dpdata.**LabeledSystem**(*file_name*, *md=False*, *fmt: Literal['gaussian/log'] = None*) → *dpdata.system.LabeledSystem*

dpdata.LabeledSystem.**from_gaussian_log**(*file_name*, *md=False*) → *dpdata.system.LabeledSystem*

Convert this format to `LabeledSystem`.

Returns

LabeledSystem
converted system

3.29 gaussian/md format

Class: *GaussianMDFormat*

3.29.1 Conversions

Convert from this format to LabeledSystem

dpdata.**LabeledSystem**(*file_name*, *fmt: Literal['gaussian/md'] = None*) → *dpdata.system.LabeledSystem*

dpdata.LabeledSystem.**from_gaussian_md**(*file_name*) → *dpdata.system.LabeledSystem*

Convert this format to `LabeledSystem`.

Returns

LabeledSystem
converted system

3.30 gromacs/gro format

3.31 gro format

Class: *GromacsGroFormat*

3.31.1 Conversions

Convert from this format to System

dpdata.**System**(*file_name*, *format_atom_name=True*, *fmt: Literal['gromacs/gro'] = None*) → *dpdata.system.System*

dpdata.**System**(*file_name*, *format_atom_name=True*, *fmt: Literal['gro'] = None*) → *dpdata.system.System*

dpdata.System.**from_gromacs_gro**(*file_name*, *format_atom_name=True*) → *dpdata.system.System*

dpdata.System.**from_gro**(*file_name*, *format_atom_name=True*) → *dpdata.system.System*

Load gromacs .gro file.

Parameters

file_name
[str] The input file name

format_atom_name
[bool] Whether to format the atom name

Returns

System
converted system

Convert from System to this format

dpdata.System.**to**(*fmt: Literal['gromacs/gro']*, *file_name=None*, *frame_idx=-1*)

dpdata.System.**to**(*fmt: Literal['gromacs/gro']*, *file_name=None*, *frame_idx=-1*)

dpdata.System.**to_gromacs_gro**(*file_name=None*, *frame_idx=-1*)

dpdata.System.**to_gro**(*file_name=None*, *frame_idx=-1*)

Dump the system in gromacs .gro format.

Parameters

file_name
[str or None] The output file name. If None, return the file content as a string

frame_idx
[int] The index of the frame to dump

Convert from LabeledSystem to this format

dpdata.LabeledSystem.**to**(*fmt: Literal['gromacs/gro']*, *file_name=None*, *frame_idx=-1*)

dpdata.LabeledSystem.**to**(*fmt: Literal['gromacs/gro']*, *file_name=None*, *frame_idx=-1*)

dpdata.LabeledSystem.**to_gromacs_gro**(*file_name=None*, *frame_idx=-1*)

dpdata.LabeledSystem.**to_gro**(*file_name=None*, *frame_idx=-1*)

Dump the system in gromacs .gro format.

Parameters

file_name
[str or None] The output file name. If None, return the file content as a string

frame_idx
[int] The index of the frame to dump

3.32 lammps/dump format

3.33 dump format

Class: *LAMMPSDumpFormat*

3.33.1 Conversions

Convert from this format to System

dpdata.**System**(*file_name*, *type_map=None*, *begin=0*, *step=1*, *unwrap=False*, *fmt: Literal['lammps/dump'] = None*) → *dpdata.system.System*

dpdata.**System**(*file_name*, *type_map=None*, *begin=0*, *step=1*, *unwrap=False*, *fmt: Literal['dump'] = None*) → *dpdata.system.System*

dpdata.System.**from_lammps_dump**(*file_name*, *type_map=None*, *begin=0*, *step=1*, *unwrap=False*) → *dpdata.system.System*

dpdata.System.**from_dump**(*file_name*, *type_map=None*, *begin=0*, *step=1*, *unwrap=False*) → *dpdata.system.System*

Convert this format to `System`.

Returns

System
converted system

3.34 lammps/lmp format

3.35 lmp format

Class: *LAMMPSLmpFormat*

3.35.1 Conversions

Convert from this format to System

dpdata.**System**(*file_name*, *type_map=None*, *fmt: Literal['lammps/lmp'] = None*) → *dpdata.system.System*

dpdata.**System**(*file_name*, *type_map=None*, *fmt: Literal['lmp'] = None*) → *dpdata.system.System*

dpdata.System.**from_lammps_lmp**(*file_name*, *type_map=None*) → *dpdata.system.System*

dpdata.System.**from_lmp**(*file_name*, *type_map=None*) → *dpdata.system.System*

Convert this format to `System`.

Returns

System
converted system

Convert from System to this format

dpdata.System.**to**(*fmt: Literal['lammps/lmp']*, *file_name*, *frame_idx=0*)

dpdata.System.**to**(*fmt: Literal['lammps/lmp']*, *file_name*, *frame_idx=0*)

dpdata.System.**to_lammps_lmp**(*file_name*, *frame_idx=0*)

dpdata.System.**to_lmp**(*file_name*, *frame_idx=0*)

Dump the system in lammps data format.

Parameters

file_name
[str] The output file name

frame_idx
[int] The index of the frame to dump

Convert from LabeledSystem to this format

dpdata.LabeledSystem.**to**(*fmt: Literal['lammps/lmp']*, *file_name*, *frame_idx=0*)

dpdata.LabeledSystem.**to**(*fmt: Literal['lammps/lmp']*, *file_name*, *frame_idx=0*)

dpdata.LabeledSystem.**to_lammps_lmp**(*file_name*, *frame_idx=0*)

dpdata.LabeledSystem.**to_lmp**(*file_name*, *frame_idx=0*)

Dump the system in lammps data format.

Parameters

file_name
[str] The output file name

frame_idx
[int] The index of the frame to dump

3.36 list format

Class: *ListFormat*

3.36.1 Conversions

Convert from System to this format

dpdata.System.**to**(*fmt: Literal['list']*)

dpdata.System.**to_list**()

Convert system to list, usefull for data collection.

Convert from LabeledSystem to this format

dpdata.LabeledSystem.**to**(*fmt: Literal['list']*)

dpdata.LabeledSystem.**to_list**()

Convert system to list, usefull for data collection.

3.37 mol_file format

3.38 mol format

Class: *MolFormat*

3.38.1 Conversions

Convert from this format to BondOrderSystem

dpdata.**BondOrderSystem**(*file_name*, *fmt: Literal['mol_file'] = None*) → *dpdata.bond_order_system.BondOrderSystem*

dpdata.**BondOrderSystem**(*file_name*, *fmt: Literal['mol'] = None*) → *dpdata.bond_order_system.BondOrderSystem*

dpdata.BondOrderSystem.**from_mol_file**(*file_name*) → *dpdata.bond_order_system.BondOrderSystem*

dpdata.BondOrderSystem.**from_mol**(*file_name*) → *dpdata.bond_order_system.BondOrderSystem*

Convert this format to BondOrderSystem.

Returns

BondOrderSystem
converted system

Convert from BondOrderSystem to this format

dpdata.BondOrderSystem.**to**(*fmt: Literal['mol_file']*, *mol*, *file_name*, *frame_idx=0*)

dpdata.BondOrderSystem.**to**(*fmt: Literal['mol_file']*, *mol*, *file_name*, *frame_idx=0*)

dpdata.BondOrderSystem.**to_mol_file**(*mol*, *file_name*, *frame_idx=0*)

dpdata.BondOrderSystem.**to_mol**(*mol*, *file_name*, *frame_idx=0*)

Convert BondOrderSystem to this format.

3.39 n2p2 format

Class: *N2P2Format*

n2p2.

This class support the conversion from and to the training data of n2p2 format. For more information about the n2p2 format, please refer to https://compphysvienna.github.io/n2p2/topics/cfg_file.html

3.39.1 Conversions

Convert from this format to LabeledSystem

`dpdata.LabeledSystem`(*file_name*, *fmt: Literal['n2p2'] = None*) → *dpdata.system.LabeledSystem*

`dpdata.LabeledSystem.from_n2p2`(*file_name*) → *dpdata.system.LabeledSystem*

Read from n2p2 format.

Parameters

file_name
[str] file name, i.e. the first argument

Returns

LabeledSystem
converted system

Convert from LabeledSystem to this format

`dpdata.LabeledSystem.to`(*fmt: Literal['n2p2']*, *file_name*)

`dpdata.LabeledSystem.to_n2p2`(*file_name*)

Write n2p2 format.

By default, LabeledSystem.to will fallback to System.to.

Parameters

file_name
[str] file name, where the data will be written

3.40 openmx/md format

Class: *OPENMXFormat*

Format for the *OpenMX <https://www.openmx-square.org/>*.

OpenMX (Open source package for Material eXplorer) is a nano-scale material simulation package based on DFT, norm-conserving pseudopotentials, and pseudo-atomic localized basis functions.

Note that two output files, System.Name.dat and System.Name.md, are required.

Use the *openmx/md* keyword argument to supply this format.

3.40.1 Conversions

Convert from this format to System

dpdata.**System**(*file_name: str*, *fmt: Literal['openmx/md'] = None*) → *dpdata.system.System*

dpdata.System.**from_openmx_md**(*file_name: str*) → *dpdata.system.System*

Read from OpenMX output.

Parameters

file_name
[str] file name, which is specified by a input file, i.e. System.Name.dat

Returns

System
converted system

Convert from this format to LabeledSystem

dpdata.**LabeledSystem**(*file_name: str*, *fmt: Literal['openmx/md'] = None*) → *dpdata.system.LabeledSystem*

dpdata.LabeledSystem.**from_openmx_md**(*file_name: str*) → *dpdata.system.LabeledSystem*

Read from OpenMX output.

Parameters

file_name
[str] file name, which is specified by a input file, i.e. System.Name.dat

Returns

LabeledSystem
converted system

3.41 orca/spout format

Class: *ORCASPOutFormat*

ORCA single point energy output.

Note that both the energy and the gradient should be printed into the output file.

3.41.1 Conversions

Convert from this format to LabeledSystem

dpdata.**LabeledSystem**(*file_name: str*, *fmt: Literal['orca/spout'] = None*) → *dpdata.system.LabeledSystem*

dpdata.LabeledSystem.**from_orca_spout**(*file_name: str*) → *dpdata.system.LabeledSystem*

Read from ORCA single point energy output.

Parameters

file_name
[str] file name

Returns

LabeledSystem
converted system

3.42 psi4/inp format

Class: *PSI4InputFormat*

Psi4 input file.

3.42.1 Conversions

Convert from System to this format

dpdata.System.**to**(*fmt: Literal['psi4/inp']*, *file_name: str*, *method: str*, *basis: str*, *charge: int = 0*, *multiplicity: int = 1*, *frame_idx=0*)

dpdata.System.**to_psi4_inp**(*file_name: str*, *method: str*, *basis: str*, *charge: int = 0*, *multiplicity: int = 1*, *frame_idx=0*)

Write PSI4 input.

Parameters

file_name
[str] file name

method
[str] computational method

basis
[str] basis set; see https://psicode.org/psi4manual/master/basissets_tables.html

charge
[int, default=0] charge of system

multiplicity
[int, default=1] multiplicity of system

frame_idx
[int, default=0] The index of the frame to dump

Convert from LabeledSystem to this format

dpdata.LabeledSystem.**to**(*fmt: Literal['psi4/inp']*, *file_name: str*, *method: str*, *basis: str*, *charge: int = 0*, *multiplicity: int = 1*, *frame_idx=0*)

dpdata.LabeledSystem.**to_psi4_inp**(*file_name: str*, *method: str*, *basis: str*, *charge: int = 0*, *multiplicity: int = 1*, *frame_idx=0*)

Write PSI4 input.

Parameters

file_name
[str] file name

method
[str] computational method

basis
[str] basis set; see https://psicode.org/psi4manual/master/basissets_tables.html

charge
[int, default=0] charge of system

multiplicity
[int, default=1] multiplicity of system

frame_idx
[int, default=0] The index of the frame to dump

3.43 psi4/out format

Class: *PSI4OutFormat*

Psi4 output.

Note that both the energy and the gradient should be printed into the output file.

3.43.1 Conversions

Convert from this format to LabeledSystem

dpdata.**LabeledSystem**(*file_name: str*, *fmt: Literal['psi4/out'] = None*) → *dpdata.system.LabeledSystem*

dpdata.LabeledSystem.**from_psi4_out**(*file_name: str*) → *dpdata.system.LabeledSystem*

Read from Psi4 output.

Parameters

file_name
[str] file name

Returns

LabeledSystem
converted system

3.44 pwmat/final.config format

3.45 pwmat/atom.config format

3.46 final.config format

3.47 atom.config format

Class: *PwmatAtomconfigFormat*

3.47.1 Conversions

Convert from this format to System

dpdata.**System**(*file_name*, *fmt: Literal['pwmat/final.config'] = None*) → *dpdata.system.System*

dpdata.**System**(*file_name*, *fmt: Literal['pwmat/atom.config'] = None*) → *dpdata.system.System*

dpdata.**System**(*file_name*, *fmt: Literal['final.config'] = None*) → *dpdata.system.System*

dpdata.**System**(*file_name*, *fmt: Literal['atom.config'] = None*) → *dpdata.system.System*

dpdata.System.**from_pwmat_finalconfig**(*file_name*) → *dpdata.system.System*

dpdata.System.**from_pwmat_atomconfig**(*file_name*) → *dpdata.system.System*

dpdata.System.**from_finalconfig**(*file_name*) → *dpdata.system.System*

dpdata.System.**from_atomconfig**(*file_name*) → *dpdata.system.System*

Convert this format to System.

Returns

System
converted system

Convert from System to this format

dpdata.System.**to**(*fmt: Literal['pwmat/final.config']*, *file_name*, *frame_idx=0*)

dpdata.System.**to**(*fmt: Literal['pwmat/final.config']*, *file_name*, *frame_idx=0*)

dpdata.System.**to**(*fmt: Literal['pwmat/final.config']*, *file_name*, *frame_idx=0*)

dpdata.System.**to**(*fmt: Literal['pwmat/final.config']*, *file_name*, *frame_idx=0*)

dpdata.System.**to_pwmat_finalconfig**(*file_name*, *frame_idx=0*)

dpdata.System.**to_pwmat_atomconfig**(*file_name*, *frame_idx=0*)

dpdata.System.**to_finalconfig**(*file_name*, *frame_idx=0*)

dpdata.System.**to_atomconfig**(*file_name*, *frame_idx=0*)

Dump the system in pwmat atom.config format.

Parameters

file_name
[str] The output file name

frame_idx
[int] The index of the frame to dump

Convert from LabeledSystem to this format

dpdata.LabeledSystem.**to**(*fmt: Literal['pwmat/final.config']*, *file_name*, *frame_idx=0*)

dpdata.LabeledSystem.**to**(*fmt: Literal['pwmat/final.config']*, *file_name*, *frame_idx=0*)

dpdata.LabeledSystem.**to**(*fmt: Literal['pwmat/final.config']*, *file_name*, *frame_idx=0*)

dpdata.LabeledSystem.**to**(*fmt: Literal['pwmat/final.config']*, *file_name*, *frame_idx=0*)

dpdata.LabeledSystem.**to_pwmat_finalconfig**(*file_name*, *frame_idx=0*)

dpdata.LabeledSystem.**to_pwmat_atomconfig**(*file_name*, *frame_idx=0*)

dpdata.LabeledSystem.**to_finalconfig**(*file_name*, *frame_idx=0*)

dpdata.LabeledSystem.**to_atomconfig**(*file_name*, *frame_idx=0*)

Dump the system in pwmat atom.config format.

Parameters

file_name
[str] The output file name

frame_idx
[int] The index of the frame to dump

3.48 pwmat/output format

3.49 pwmat/mlmd format

3.50 pwmat/movement format

3.51 mlmd format

3.52 movement format

Class: *PwmatOutputFormat*

3.52.1 Conversions

Convert from this format to LabeledSystem

dpdata.**LabeledSystem**(*file_name*, *begin=0*, *step=1*, *convergence_check=True*, *fmt: Literal['pwmat/output'] = None*) → *dpdata.system.LabeledSystem*

dpdata.**LabeledSystem**(*file_name*, *begin=0*, *step=1*, *convergence_check=True*, *fmt: Literal['pwmat/mlmd'] = None*) → *dpdata.system.LabeledSystem*

dpdata.**LabeledSystem**(*file_name*, *begin=0*, *step=1*, *convergence_check=True*, *fmt: Literal['pwmat/movement'] = None*) → *dpdata.system.LabeledSystem*

dpdata.**LabeledSystem**(*file_name*, *begin=0*, *step=1*, *convergence_check=True*, *fmt: Literal['mlmd'] = None*) → *dpdata.system.LabeledSystem*

dpdata.**LabeledSystem**(*file_name*, *begin=0*, *step=1*, *convergence_check=True*, *fmt: Literal['movement'] = None*) → *dpdata.system.LabeledSystem*

dpdata.LabeledSystem.**from_pwmat_output**(*file_name*, *begin=0*, *step=1*, *convergence_check=True*) → *dpdata.system.LabeledSystem*

dpdata.LabeledSystem.**from_pwmat_mlmd**(*file_name*, *begin=0*, *step=1*, *convergence_check=True*) → *dpdata.system.LabeledSystem*

dpdata.LabeledSystem.**from_pwmat_movement**(*file_name*, *begin=0*, *step=1*, *convergence_check=True*) → *dpdata.system.LabeledSystem*

dpdata.LabeledSystem.**from_mlmd**(*file_name*, *begin=0*, *step=1*, *convergence_check=True*) → *dpdata.system.LabeledSystem*

dpdata.LabeledSystem.**from_movement**(*file_name*, *begin=0*, *step=1*, *convergence_check=True*) → *dpdata.system.LabeledSystem*

Convert this format to `LabeledSystem`.

Returns

LabeledSystem
converted system

3.53 3dmol format

Class: *Py3DMolFormat*

3DMol format.

To use this format, py3Dmol should be installed in advance.

3.53.1 Conversions

Convert from System to this format

dpdata.System.**to**(*fmt: Literal['3dmol']*, *f_idx: int = 0*, *size: Tuple[int] = (300, 300)*, *style: dict = {'stick': {}, 'sphere': {'radius': 0.4}}*)

dpdata.System.**to_3dmol**(*f_idx: int = 0*, *size: Tuple[int] = (300, 300)*, *style: dict = {'stick': {}, 'sphere': {'radius': 0.4}}*)

Show 3D structure of a frame in jupyter.

Parameters

f_idx
[int] frame index to show

size
[tuple[int]] (width, height) of the widget

style
[dict] style of 3DMol. Read 3DMol documentation for details.

Examples

```
>>> system.to_3dmol()
```

Convert from LabeledSystem to this format

dpdata.LabeledSystem.**to**(*fmt: Literal['3dmol']*, *f_idx: int = 0*, *size: Tuple[int] = (300, 300)*, *style: dict = {'stick': {}, 'sphere': {'radius': 0.4}}*)

dpdata.LabeledSystem.**to_3dmol**(*f_idx: int = 0*, *size: Tuple[int] = (300, 300)*, *style: dict = {'stick': {}, 'sphere': {'radius': 0.4}}*)

Show 3D structure of a frame in jupyter.

Parameters

f_idx
[int] frame index to show

size
[tuple[int]] (width, height) of the widget

style
[dict] style of 3DMol. Read 3DMol documentation for details.

Examples

```
>>> system.to_3dmol()
```

3.54 pymatgen/computedstructureentry format

Class: *PyMatgenCSEFormat*

3.54.1 Conversions

Convert from LabeledSystem to this format

dpdata.LabeledSystem.**to**(*fmt: Literal['pymatgen/computedstructureentry']*)

dpdata.LabeledSystem.**to_pymatgen_computedstructureentry**()

Convert System to Pymagen ComputedStructureEntry obj.

3.55 pymatgen/molecule format

Class: *PyMatgenMoleculeFormat*

3.55.1 Conversions

Convert from this format to System

dpdata.**System**(*file_name*, *fmt: Literal['pymatgen/molecule'] = None*) → *dpdata.system.System*

dpdata.System.**from_pymatgen_molecule**(*file_name*) → *dpdata.system.System*

Convert this format to System.

Returns

System
converted system

Convert from System to this format

dpdata.System.**to**(*fmt: Literal['pymatgen/molecule']*)

dpdata.System.**to_pymatgen_molecule**()

Convert System to Pymatgen Molecule obj.

Convert from LabeledSystem to this format

dpdata.LabeledSystem.**to**(*fmt: Literal['pymatgen/molecule']*)

dpdata.LabeledSystem.**to_pymatgen_molecule**()

Convert System to Pymatgen Molecule obj.

3.56 pymatgen/structure format

Class: *PyMatgenStructureFormat*

3.56.1 Conversions

Convert from System to this format

dpdata.System.**to**(*fmt: Literal['pymatgen/structure']*)

dpdata.System.**to_pymatgen_structure**()

Convert System to Pymatgen Structure obj.

Convert from LabeledSystem to this format

dpdata.LabeledSystem.**to**(*fmt: Literal['pymatgen/structure']*)

dpdata.LabeledSystem.**to_pymatgen_structure**()

Convert System to Pymatgen Structure obj.

3.57 qe/pw/scf format

Class: *QECPPWSCFFormat*

3.57.1 Conversions

Convert from this format to LabeledSystem

dpdata.**LabeledSystem**(*file_name*, *fmt: Literal['qe/pw/scf'] = None*) → *dpdata.system.LabeledSystem*

dpdata.LabeledSystem.**from_qe_pw_scf**(*file_name*) → *dpdata.system.LabeledSystem*

Convert this format to `LabeledSystem`.

Returns

LabeledSystem
converted system

3.58 qe/cp/traj format

Class: *QECPTrajFormat*

3.58.1 Conversions

Convert from this format to System

dpdata.**System**(*file_name*, *begin=0*, *step=1*, *fmt: Literal['qe/cp/traj'] = None*) → *dpdata.system.System*

dpdata.System.**from_qe_cp_traj**(*file_name*, *begin=0*, *step=1*) → *dpdata.system.System*

Convert this format to `System`.

Returns

System
converted system

Convert from this format to LabeledSystem

dpdata.**LabeledSystem**(*file_name*, *begin=0*, *step=1*, *fmt: Literal['qe/cp/traj'] = None*) → *dpdata.system.LabeledSystem*

dpdata.LabeledSystem.**from_qe_cp_traj**(*file_name*, *begin=0*, *step=1*) → *dpdata.system.LabeledSystem*

Convert this format to `LabeledSystem`.

Returns

LabeledSystem
converted system

3.59 quip/gap/xyz_file format

3.60 quip/gap/xyz format

Class: *QuipGapXYZFormat*

3.60.1 Conversions

Convert from this format to LabeledSystem

dpdata.**LabeledSystem**(*data*, *fmt: Literal['quip/gap/xyz_file'] = None*) → *dpdata.system.LabeledSystem*

dpdata.**LabeledSystem**(*data*, *fmt: Literal['quip/gap/xyz'] = None*) → *dpdata.system.LabeledSystem*

dpdata.LabeledSystem.**from_quip_gap_xyz_file**(*data*) → *dpdata.system.LabeledSystem*

dpdata.LabeledSystem.**from_quip_gap_xyz**(*data*) → *dpdata.system.LabeledSystem*

Convert this format to LabeledSystem.

Returns

LabeledSystem
converted system

Convert from this format to MultiSystems

dpdata.MultiSystems.**from_quip_gap_xyz_file**(*file_name*) → *dpdata.system.MultiSystems*

dpdata.MultiSystems.**from_quip_gap_xyz**(*file_name*) → *dpdata.system.MultiSystems*

Convert this format to MultiSystems.

Parameters

directory
[str] directory of systems

Returns

MultiSystems
converted system

3.61 sqm/in format

Class: *SQMINFormat*

3.61.1 Conversions

Convert from System to this format

dpdata.System.**to**(*fmt: Literal['sqm/in']*, *fname=None*, *frame_idx=0*)

dpdata.System.**to_sqm_in**(*fname=None*, *frame_idx=0*)

Generate input files for semi-emperical calculation in sqm software.

Parameters

fname
[str] output file name

frame_idx
[int, default=0] index of frame to write

Other Parameters

****kwargs**
[dict]

valid parameters are:

qm_theory
[str, default=dftb3] level of theory. Options includes AM1, RM1, MNDO, PM3-PDDG, MNDO-PDDG, PM3-CARB1, MNDO/d, AM1/d, PM6, DFTB2, DFTB3

charge
[int, default=0] total charge in electron units

maxcyc
[int, default=0] maximum number of minimization cycles to allow. 0 represents a single-point calculation

mult
[int, default=1] multiplicity. Only 1 is allowed.

Convert from LabeledSystem to this format

`dpdata.LabeledSystem.to`(*fmt: Literal['sqm/in']*, *fname=None*, *frame_idx=0*)

`dpdata.LabeledSystem.to_sqm_in`(*fname=None*, *frame_idx=0*)

Generate input files for semi-emperical calculation in sqm software.

Parameters

fname
[str] output file name

frame_idx
[int, default=0] index of frame to write

Other Parameters

****kwargs**
[dict]

valid parameters are:

qm_theory
[str, default=dftb3] level of theory. Options includes AM1, RM1, MNDO, PM3-PDDG, MNDO-PDDG, PM3-CARB1, MNDO/d, AM1/d, PM6, DFTB2, DFTB3

charge
[int, default=0] total charge in electron units

maxcyc
[int, default=0] maximum number of minimization cycles to allow. 0 represents a single-point calculation

mult
[int, default=1] multiplicity. Only 1 is allowed.

3.62 sqm/out format

Class: *SQMOutFormat*

3.62.1 Conversions

Convert from this format to System

dpdata.**System**(*fname*, *fmt: Literal['sqm/out'] = None*) → *dpdata.system.System*

dpdata.System.**from_sqm_out**(*fname*) → *dpdata.system.System*

Read from ambertools sqm.out.

Returns

System
converted system

Convert from this format to LabeledSystem

dpdata.**LabeledSystem**(*fname*, *fmt: Literal['sqm/out'] = None*) → *dpdata.system.LabeledSystem*

dpdata.LabeledSystem.**from_sqm_out**(*fname*) → *dpdata.system.LabeledSystem*

Read from ambertools sqm.out.

Returns

LabeledSystem
converted system

3.63 sdf_file format

3.64 sdf format

Class: *SdfFormat*

3.64.1 Conversions

Convert from this format to BondOrderSystem

dpdata.**BondOrderSystem**(*file_name*, *fmt: Literal['sdf_file'] = None*) → *dpdata.bond_order_system.BondOrderSystem*

dpdata.**BondOrderSystem**(*file_name*, *fmt: Literal['sdf'] = None*) → *dpdata.bond_order_system.BondOrderSystem*

dpdata.BondOrderSystem.**from_sdf_file**(*file_name*) → *dpdata.bond_order_system.BondOrderSystem*

dpdata.BondOrderSystem.**from_sdf**(*file_name*) → *dpdata.bond_order_system.BondOrderSystem*

Note that it requires all molecules in .sdf file must be of the same topology.

Returns

BondOrderSystem
converted system

Convert from BondOrderSystem to this format

dpdata.BondOrderSystem.**to**(*fmt: Literal['sdf_file']*, *mol*, *file_name*, *frame_idx=-1*)

dpdata.BondOrderSystem.**to**(*fmt: Literal['sdf_file']*, *mol*, *file_name*, *frame_idx=-1*)

dpdata.BondOrderSystem.**to_sdf_file**(*mol*, *file_name*, *frame_idx=-1*)

dpdata.BondOrderSystem.**to_sdf**(*mol*, *file_name*, *frame_idx=-1*)

Convert `BondOrderSystem` to this format.

3.65 siesta/aimd_output format

Class: *SiestaAIMDOutputFormat*

3.65.1 Conversions

Convert from this format to System

dpdata.**System**(*file_name*, *fmt: Literal['siesta/aimd_output'] = None*) → *dpdata.system.System*

dpdata.System.**from_siesta_aimd_output**(*file_name*) → *dpdata.system.System*

Convert this format to `System`.

Returns

System
converted system

Convert from this format to LabeledSystem

dpdata.**LabeledSystem**(*file_name*, *fmt: Literal['siesta/aimd_output'] = None*) → *dpdata.system.LabeledSystem*

dpdata.LabeledSystem.**from_siesta_aimd_output**(*file_name*) → *dpdata.system.LabeledSystem*

Convert this format to `LabeledSystem`.

Returns

LabeledSystem
converted system

3.66 siesta/output format

Class: *SiestaOutputFormat*

3.66.1 Conversions

Convert from this format to System

dpdata.**System**(*file_name*, *fmt: Literal['siesta/output'] = None*) → *dpdata.system.System*

dpdata.System.**from_siesta_output**(*file_name*) → *dpdata.system.System*

Convert this format to `System`.

Returns

System
converted system

Convert from this format to LabeledSystem

dpdata.**LabeledSystem**(*file_name*, *fmt: Literal['siesta/output'] = None*) → *dpdata.system.LabeledSystem*

dpdata.LabeledSystem.**from_siesta_output**(*file_name*) → *dpdata.system.LabeledSystem*

Convert this format to `LabeledSystem`.

Returns

LabeledSystem
converted system

3.67 vasp/outcar format

3.68 outcar format

Class: *VASPOutcarFormat*

3.68.1 Conversions

Convert from this format to LabeledSystem

dpdata.**LabeledSystem**(*file_name*, *begin=0*, *step=1*, *convergence_check=True*, *fmt: Literal['vasp/outcar'] = None*) → *dpdata.system.LabeledSystem*

dpdata.**LabeledSystem**(*file_name*, *begin=0*, *step=1*, *convergence_check=True*, *fmt: Literal['outcar'] = None*) → *dpdata.system.LabeledSystem*

dpdata.LabeledSystem.**from_vasp_outcar**(*file_name*, *begin=0*, *step=1*, *convergence_check=True*) → *dpdata.system.LabeledSystem*

dpdata.LabeledSystem.**from_outcar**(*file_name*, *begin=0*, *step=1*, *convergence_check=True*) → *dpdata.system.LabeledSystem*

Convert this format to `LabeledSystem`.

Returns

LabeledSystem
converted system

3.69 vasp/contcar format

3.70 vasp/poscar format

3.71 contcar format

3.72 poscar format

Class: *VASPPoscarFormat*

3.72.1 Conversions

Convert from this format to System

dpdata.**System**(*file_name*, *fmt: Literal['vasp/contcar'] = None*) → *dpdata.system.System*

dpdata.**System**(*file_name*, *fmt: Literal['vasp/poscar'] = None*) → *dpdata.system.System*

dpdata.**System**(*file_name*, *fmt: Literal['contcar'] = None*) → *dpdata.system.System*

dpdata.**System**(*file_name*, *fmt: Literal['poscar'] = None*) → *dpdata.system.System*

dpdata.System.**from_vasp_contcar**(*file_name*) → *dpdata.system.System*

dpdata.System.**from_vasp_poscar**(*file_name*) → *dpdata.system.System*

dpdata.System.**from_contcar**(*file_name*) → *dpdata.system.System*

dpdata.System.**from_poscar**(*file_name*) → *dpdata.system.System*

Convert this format to System.

Returns

System
converted system

Convert from System to this format

dpdata.System.**to**(*fmt: Literal['vasp/contcar']*, *file_name*, *frame_idx=0*)

dpdata.System.**to**(*fmt: Literal['vasp/contcar']*, *file_name*, *frame_idx=0*)

dpdata.System.**to**(*fmt: Literal['vasp/contcar']*, *file_name*, *frame_idx=0*)

dpdata.System.**to**(*fmt: Literal['vasp/contcar']*, *file_name*, *frame_idx=0*)

dpdata.System.**to_vasp_contcar**(*file_name*, *frame_idx=0*)

dpdata.System.**to_vasp_poscar**(*file_name*, *frame_idx=0*)

dpdata.System.**to_contcar**(*file_name*, *frame_idx=0*)

dpdata.System.**to_poscar**(*file_name*, *frame_idx=0*)

Dump the system in vasp POSCAR format.

Parameters

file_name
[str] The output file name

frame_idx
[int] The index of the frame to dump

Convert from LabeledSystem to this format

dpdata.LabeledSystem.**to**(*fmt: Literal['vasp/contcar']*, *file_name*, *frame_idx=0*)

dpdata.LabeledSystem.**to**(*fmt: Literal['vasp/contcar']*, *file_name*, *frame_idx=0*)

dpdata.LabeledSystem.**to**(*fmt: Literal['vasp/contcar']*, *file_name*, *frame_idx=0*)

dpdata.LabeledSystem.**to**(*fmt: Literal['vasp/contcar']*, *file_name*, *frame_idx=0*)

dpdata.LabeledSystem.**to_vasp_contcar**(*file_name*, *frame_idx=0*)

dpdata.LabeledSystem.**to_vasp_poscar**(*file_name*, *frame_idx=0*)

dpdata.LabeledSystem.**to_contcar**(*file_name*, *frame_idx=0*)

dpdata.LabeledSystem.**to_poscar**(*file_name*, *frame_idx=0*)

Dump the system in vasp POSCAR format.

Parameters

file_name
[str] The output file name

frame_idx
[int] The index of the frame to dump

3.73 vasp/string format

Class: *VASPStringFormat*

3.73.1 Conversions

Convert from System to this format

dpdata.System.**to**(*fmt: Literal['vasp/string']*, *frame_idx=0*)

dpdata.System.**to_vasp_string**(*frame_idx=0*)

Dump the system in vasp POSCAR format string.

Parameters

frame_idx
[int] The index of the frame to dump

Convert from LabeledSystem to this format

dpdata.LabeledSystem.**to**(*fmt: Literal['vasp/string']*, *frame_idx=0*)

dpdata.LabeledSystem.**to_vasp_string**(*frame_idx=0*)

Dump the system in vasp POSCAR format string.

Parameters

frame_idx
[int] The index of the frame to dump

3.74 vasp/xml format

3.75 xml format

Class: *VASPXMLFormat*

3.75.1 Conversions

Convert from this format to LabeledSystem

dpdata.**LabeledSystem**(*file_name*, *begin=0*, *step=1*, *fmt: Literal['vasp/xml'] = None*) → *dpdata.system.LabeledSystem*

dpdata.**LabeledSystem**(*file_name*, *begin=0*, *step=1*, *fmt: Literal['xml'] = None*) → *dpdata.system.LabeledSystem*

dpdata.LabeledSystem.**from_vasp_xml**(*file_name*, *begin=0*, *step=1*) → *dpdata.system.LabeledSystem*

dpdata.LabeledSystem.**from_xml**(*file_name*, *begin=0*, *step=1*) → *dpdata.system.LabeledSystem*

Convert this format to `LabeledSystem`.

Returns

LabeledSystem
converted system

3.76 xyz format

Class: *XYZFormat*

XYZ foramt.

Examples

```
>>> s.to("xyz", "a.xyz")
```

3.76.1 Conversions

Convert from this format to System

dpdata.**System**(*file_name*, *fmt: Literal['xyz'] = None*) → *dpdata.system.System*

dpdata.System.**from_xyz**(*file_name*) → *dpdata.system.System*

Convert this format to `System`.

Returns

System
converted system

Convert from System to this format

dpdata.System.**to**(*fmt: Literal['xyz']*, *file_name*)

dpdata.System.**to_xyz**(*file_name*)

Convert `System` to this format.

Convert from LabeledSystem to this format

dpdata.LabeledSystem.**to**(*fmt: Literal['xyz']*, *file_name*)

dpdata.LabeledSystem.**to_xyz**(*file_name*)

Convert `LabeledSystem` to this format.

CHAPTER

FOUR

SUPPORTED DRIVERS

dpdata supports the following drivers:

Table 1: Supported Drivers

Class	Alias
HybridDriver	hybrid
DPDriver	deepmd-kit deepmd dp
ASEDriver	ase
GaussianDriver	gaussian
SQMDriver	sqm

CHAPTER

FIVE

SUPPORTED MINIMIZERS

dpdata supports the following minimizers:

Table 1: Supported Minimizers

Class	Alias
ASEMinimizer	`ase`
SQMMinimizer	`sqm`

CHAPTER

SIX

API DOCUMENTATION

6.1 dpdata package

class dpdata.**BondOrderSystem**(*file_name=None*, *fmt='auto'*, *type_map=None*, *begin=0*, *step=1*, *data=None*, *rdkit_mol=None*, *sanitize_level='medium'*, *raise_errors=True*, *verbose=False*, ***kwargs*)

Bases: *System*

The system with chemical bond and formal charges information.

For example, a labeled methane system named *d_example* has one molecule (5 atoms, 4 bonds) and *n_frames* frames. The bond order and formal charge information can be accessed by

- ***d_example['bonds']***
  [a numpy array of size 4 x 3, and] the first column represents the index of begin atom, the second column represents the index of end atom, the third columen represents the bond order:

  1 - single bond, 2 - double bond, 3 - triple bond, 1.5 - aromatic bond

- *d_example['formal_charges']* : a numpy array of size 5 x 1

Attributes

formula
Return the formula of this system, like C3H5O2.

formula_hash
Return the hash of the formula of this system.

nopbc

short_formula
Return the short formula of this system.

short_name
Return the short name of this system (no more than 255 bytes), in the following order: - formula - short_formula - formula_hash.

uniq_formula
Return the uniq_formula of this system.

Methods

add_atom_names(atom_names)	Add atom_names that do not exist.
append(system)	Append a system to this system.
apply_pbc()	Append periodic boundary condition.
as_dict()	Returns data dict of System instance.
check_data()	Check if data is correct.
check_type_map(type_map)	Assign atom_names to type_map if type_map is given and different from atom_names.
convert_to_mixed_type([type_map])	Convert the data dict to mixed type format structure, in order to append systems with different formula but the same number of atoms.
copy()	Returns a copy of the system.
dump(filename[, indent])	Dump .json or .yaml file.
extend(systems)	Extend a system list to this system.
from_3dmol(file_name, **kwargs)	Read data from *dpdata.plugins.3dmol.Py3DMolFormat* format.
from_abacus_lcao_md(file_name, **kwargs)	Read data from *dpdata.plugins.abacus.AbacusMDFormat* format.
from_abacus_lcao_relax(file_name, **kwargs)	Read data from *dpdata.plugins.abacus.AbacusRelaxFormat* format.
from_abacus_lcao_scf(file_name, **kwargs)	Read data from *dpdata.plugins.abacus.AbacusSCFFormat* format.
from_abacus_md(file_name, **kwargs)	Read data from *dpdata.plugins.abacus.AbacusMDFormat* format.
from_abacus_pw_md(file_name, **kwargs)	Read data from *dpdata.plugins.abacus.AbacusMDFormat* format.
from_abacus_pw_relax(file_name, **kwargs)	Read data from *dpdata.plugins.abacus.AbacusRelaxFormat* format.
from_abacus_pw_scf(file_name, **kwargs)	Read data from *dpdata.plugins.abacus.AbacusSCFFormat* format.
from_abacus_relax(file_name, **kwargs)	Read data from *dpdata.plugins.abacus.AbacusRelaxFormat* format.
from_abacus_scf(file_name, **kwargs)	Read data from *dpdata.plugins.abacus.AbacusSCFFormat* format.
from_abacus_stru(file_name, **kwargs)	Read data from *dpdata.plugins.abacus.AbacusSTRUFormat* format.
from_amber_md(file_name, **kwargs)	Read data from *dpdata.plugins.amber.AmberMDFormat* format.
from_ase_structure(file_name, **kwargs)	Read data from *dpdata.plugins.ase.ASEStructureFormat* format.
from_ase_traj(file_name, **kwargs)	Read data from *dpdata.plugins.ase.ASETrajFormat* format.
from_atomconfig(file_name, **kwargs)	Read data from *dpdata.plugins.pwmat.PwmatAtomconfigFormat* format.
from_contcar(file_name, **kwargs)	Read data from *dpdata.plugins.vasp.VASPPoscarFormat* format.
from_cp2k_aimd_output(file_name, **kwargs)	Read data from *dpdata.plugins.cp2k.CP2KAIMDOutputFormat* format.
from_cp2k_output(file_name, **kwargs)	Read data from *dpdata.plugins.cp2k.CP2KOutputFormat* format.

continues on next page

Table 1 – continued from previous page

from_deepmd(file_name, **kwargs)	Read data from *dpdata.plugins.deepmd.DeePMDRawFormat* format.
from_deepmd_comp(file_name, **kwargs)	Read data from *dpdata.plugins.deepmd.DeePMDCompFormat* format.
from_deepmd_hdf5(file_name, **kwargs)	Read data from *dpdata.plugins.deepmd.DeePMDHDF5Format* format.
from_deepmd_npy(file_name, **kwargs)	Read data from *dpdata.plugins.deepmd.DeePMDCompFormat* format.
from_deepmd_npy_mixed(file_name, **kwargs)	Read data from *dpdata.plugins.deepmd.DeePMDMixedFormat* format.
from_deepmd_raw(file_name, **kwargs)	Read data from *dpdata.plugins.deepmd.DeePMDRawFormat* format.
from_dftbplus(file_name, **kwargs)	Read data from *dpdata.plugins.dftbplus.DFTBplusFormat* format.
from_dict(d)	**param d** Dict representation.
from_dump(file_name, **kwargs)	Read data from *dpdata.plugins.lammps.LAMMPSDumpFormat* format.
from_fhi_aims_md(file_name, **kwargs)	Read data from *dpdata.plugins.fhi_aims.FhiMDFormat* format.
from_fhi_aims_output(file_name, **kwargs)	Read data from *dpdata.plugins.fhi_aims.FhiMDFormat* format.
from_fhi_aims_scf(file_name, **kwargs)	Read data from *dpdata.plugins.fhi_aims.FhiSCFFormat* format.
from_finalconfig(file_name, **kwargs)	Read data from *dpdata.plugins.pwmat.PwmatAtomconfigFormat* format.
from_gaussian_gjf(file_name, **kwargs)	Read data from *dpdata.plugins.gaussian.GaussiaGJFFormat* format.
from_gaussian_log(file_name, **kwargs)	Read data from *dpdata.plugins.gaussian.GaussianLogFormat* format.
from_gaussian_md(file_name, **kwargs)	Read data from *dpdata.plugins.gaussian.GaussianMDFormat* format.
from_gro(file_name, **kwargs)	Read data from *dpdata.plugins.gromacs.GromacsGroFormat* format.
from_gromacs_gro(file_name, **kwargs)	Read data from *dpdata.plugins.gromacs.GromacsGroFormat* format.
from_lammps_dump(file_name, **kwargs)	Read data from *dpdata.plugins.lammps.LAMMPSDumpFormat* format.
from_lammps_lmp(file_name, **kwargs)	Read data from *dpdata.plugins.lammps.LAMMPSLmpFormat* format.
from_list(file_name, **kwargs)	Read data from *dpdata.plugins.list.ListFormat* format.
from_lmp(file_name, **kwargs)	Read data from *dpdata.plugins.lammps.LAMMPSLmpFormat* format.
from_mlmd(file_name, **kwargs)	Read data from *dpdata.plugins.pwmat.PwmatOutputFormat* format.
from_mol(file_name, **kwargs)	Read data from *dpdata.plugins.rdkit.MolFormat* format.
from_mol_file(file_name, **kwargs)	Read data from *dpdata.plugins.rdkit.MolFormat* format.

continues on next page

Table 1 – continued from previous page

from_movement(file_name, **kwargs)	Read data from *dpdata.plugins.pwmat.PwmatOutputFormat* format.
from_n2p2(file_name, **kwargs)	Read data from *dpdata.plugins.n2p2.N2P2Format* format.
from_openmx_md(file_name, **kwargs)	Read data from *dpdata.plugins.openmx.OPENMXFormat* format.
from_orca_spout(file_name, **kwargs)	Read data from *dpdata.plugins.orca.ORCASPOutFormat* format.
from_outcar(file_name, **kwargs)	Read data from *dpdata.plugins.vasp.VASPOutcarFormat* format.
from_poscar(file_name, **kwargs)	Read data from *dpdata.plugins.vasp.VASPPoscarFormat* format.
from_psi4_inp(file_name, **kwargs)	Read data from *dpdata.plugins.psi4.PSI4InputFormat* format.
from_psi4_out(file_name, **kwargs)	Read data from *dpdata.plugins.psi4.PSI4OutFormat* format.
from_pwmat_atomconfig(file_name, **kwargs)	Read data from *dpdata.plugins.pwmat.PwmatAtomconfigFormat* format.
from_pwmat_finalconfig(file_name, **kwargs)	Read data from *dpdata.plugins.pwmat.PwmatAtomconfigFormat* format.
from_pwmat_mlmd(file_name, **kwargs)	Read data from *dpdata.plugins.pwmat.PwmatOutputFormat* format.
from_pwmat_movement(file_name, **kwargs)	Read data from *dpdata.plugins.pwmat.PwmatOutputFormat* format.
from_pwmat_output(file_name, **kwargs)	Read data from *dpdata.plugins.pwmat.PwmatOutputFormat* format.
from_pymatgen_computedstructureentry(...)	Read data from *dpdata.plugins.pymatgen.PyMatgenCSEFormat* format.
from_pymatgen_molecule(file_name, **kwargs)	Read data from *dpdata.plugins.pymatgen.PyMatgenMoleculeFormat* format.
from_pymatgen_structure(file_name, **kwargs)	Read data from *dpdata.plugins.pymatgen.PyMatgenStructureFormat* format.
from_qe_cp_traj(file_name, **kwargs)	Read data from *dpdata.plugins.qe.QECPTrajFormat* format.
from_qe_pw_scf(file_name, **kwargs)	Read data from *dpdata.plugins.qe.QECPPWSCFFormat* format.
from_quip_gap_xyz(file_name, **kwargs)	Read data from *dpdata.plugins.xyz.QuipGapXYZFormat* format.
from_quip_gap_xyz_file(file_name, **kwargs)	Read data from *dpdata.plugins.xyz.QuipGapXYZFormat* format.
from_rdkit_mol(rdkit_mol)	Initialize from a rdkit.Chem.rdchem.Mol object.
from_sdf(file_name, **kwargs)	Read data from *dpdata.plugins.rdkit.SdfFormat* format.
from_sdf_file(file_name, **kwargs)	Read data from *dpdata.plugins.rdkit.SdfFormat* format.
from_siesta_aiMD_output(file_name, **kwargs)	Read data from *dpdata.plugins.siesta.SiestaAIMDOutputFormat* format.
from_siesta_aimd_output(file_name, **kwargs)	Read data from *dpdata.plugins.siesta.SiestaAIMDOutputFormat* format.
from_siesta_output(file_name, **kwargs)	Read data from *dpdata.plugins.siesta.SiestaOutputFormat* format.

continues on next page

Table 1 – continued from previous page

from_sqm_in(file_name, **kwargs)	Read data from *dpdata.plugins.amber.SQMINFormat* format.
from_sqm_out(file_name, **kwargs)	Read data from *dpdata.plugins.amber.SQMOutFormat* format.
from_stru(file_name, **kwargs)	Read data from *dpdata.plugins.abacus.AbacusSTRUFormat* format.
from_vasp_contcar(file_name, **kwargs)	Read data from *dpdata.plugins.vasp.VASPPoscarFormat* format.
from_vasp_outcar(file_name, **kwargs)	Read data from *dpdata.plugins.vasp.VASPOutcarFormat* format.
from_vasp_poscar(file_name, **kwargs)	Read data from *dpdata.plugins.vasp.VASPPoscarFormat* format.
from_vasp_string(file_name, **kwargs)	Read data from *dpdata.plugins.vasp.VASPStringFormat* format.
from_vasp_xml(file_name, **kwargs)	Read data from *dpdata.plugins.vasp.VASPXMLFormat* format.
from_xml(file_name, **kwargs)	Read data from *dpdata.plugins.vasp.VASPXMLFormat* format.
from_xyz(file_name, **kwargs)	Read data from *dpdata.plugins.xyz.XYZFormat* format.
get_atom_names()	Returns name of atoms.
get_atom_numbs()	Returns number of atoms.
get_atom_types()	Returns type of atoms.
get_bond_order(begin_atom_idx, end_atom_idx)	Return the bond order between given atoms.
get_charge()	Return the total formal charge of the moleclue.
get_formal_charges()	Return the formal charges on each atom.
get_mol()	Return the rdkit.Mol object.
get_natoms()	Returns total number of atoms in the system.
get_nbonds()	Return the number of bonds.
get_nframes()	Returns number of frames in the system.
get_ntypes()	Returns total number of atom types in the system.
load(filename)	Rebuild System obj.
map_atom_types([type_map])	Map the atom types of the system.
minimize(*args, minimizer, **kwargs)	Minimize the geometry.
perturb(pert_num, cell_pert_fraction, ...[, ...])	Perturb each frame in the system randomly.
pick_atom_idx(idx[, nopbc])	Pick atom index.
pick_by_amber_mask(param, maskstr[, ...])	Pick atoms by amber mask.
predict(*args[, driver])	Predict energies and forces by a driver.
register_data_type(*data_type)	Register data type.
remove_atom_names(atom_names)	Remove atom names and all such atoms.
remove_pbc([protect_layer])	This method does NOT delete the definition of the cells, it (1) revises the cell to a cubic cell and ensures that the cell boundary to any atom in the system is no less than *protect_layer* (2) translates the system such that the center-of-geometry of the system locates at the center of the cell.
replicate(ncopy)	Replicate the each frame in the system in 3 dimensions.
shuffle()	Shuffle frames randomly.
sort_atom_names([type_map])	Sort atom_names of the system and reorder atom_numbs and atom_types accoarding to atom_names.

continues on next page

Table 1 – continued from previous page

sort_atom_types()	Sort atom types.
sub_system(f_idx)	Construct a subsystem from the system.
to(fmt, *args, **kwargs)	Dump systems to the specific format.
to_3dmol(*args, **kwargs)	Dump data to *dpdata.plugins.3dmol.Py3DMolFormat* format.
to_abacus_lcao_md(*args, **kwargs)	Dump data to *dpdata.plugins.abacus.AbacusMDFormat* format.
to_abacus_lcao_relax(*args, **kwargs)	Dump data to *dpdata.plugins.abacus.AbacusRelaxFormat* format.
to_abacus_lcao_scf(*args, **kwargs)	Dump data to *dpdata.plugins.abacus.AbacusSCFFormat* format.
to_abacus_md(*args, **kwargs)	Dump data to *dpdata.plugins.abacus.AbacusMDFormat* format.
to_abacus_pw_md(*args, **kwargs)	Dump data to *dpdata.plugins.abacus.AbacusMDFormat* format.
to_abacus_pw_relax(*args, **kwargs)	Dump data to *dpdata.plugins.abacus.AbacusRelaxFormat* format.
to_abacus_pw_scf(*args, **kwargs)	Dump data to *dpdata.plugins.abacus.AbacusSCFFormat* format.
to_abacus_relax(*args, **kwargs)	Dump data to *dpdata.plugins.abacus.AbacusRelaxFormat* format.
to_abacus_scf(*args, **kwargs)	Dump data to *dpdata.plugins.abacus.AbacusSCFFormat* format.
to_abacus_stru(*args, **kwargs)	Dump data to *dpdata.plugins.abacus.AbacusSTRUFormat* format.
to_amber_md(*args, **kwargs)	Dump data to *dpdata.plugins.amber.AmberMDFormat* format.
to_ase_structure(*args, **kwargs)	Dump data to *dpdata.plugins.ase.ASEStructureFormat* format.
to_ase_traj(*args, **kwargs)	Dump data to *dpdata.plugins.ase.ASETrajFormat* format.
to_atomconfig(*args, **kwargs)	Dump data to *dpdata.plugins.pwmat.PwmatAtomconfigFormat* format.
to_contcar(*args, **kwargs)	Dump data to *dpdata.plugins.vasp.VASPPoscarFormat* format.
to_cp2k_aimd_output(*args, **kwargs)	Dump data to *dpdata.plugins.cp2k.CP2KAIMDOutputFormat* format.
to_cp2k_output(*args, **kwargs)	Dump data to *dpdata.plugins.cp2k.CP2KOutputFormat* format.
to_deepmd(*args, **kwargs)	Dump data to *dpdata.plugins.deepmd.DeePMDRawFormat* format.
to_deepmd_comp(*args, **kwargs)	Dump data to *dpdata.plugins.deepmd.DeePMDCompFormat* format.
to_deepmd_hdf5(*args, **kwargs)	Dump data to *dpdata.plugins.deepmd.DeePMDHDF5Format* format.
to_deepmd_npy(*args, **kwargs)	Dump data to *dpdata.plugins.deepmd.DeePMDCompFormat* format.
to_deepmd_npy_mixed(*args, **kwargs)	Dump data to *dpdata.plugins.deepmd.DeePMDMixedFormat* format.
to_deepmd_raw(*args, **kwargs)	Dump data to *dpdata.plugins.deepmd.DeePMDRawFormat* format.

continues on next page

Table 1 – continued from previous page

to_dftbplus(*args, **kwargs)	Dump data to *dpdata.plugins.dftbplus.DFTBplusFormat* format.
to_dump(*args, **kwargs)	Dump data to *dpdata.plugins.lammps.LAMMPSDumpFormat* format.
to_fhi_aims_md(*args, **kwargs)	Dump data to *dpdata.plugins.fhi_aims.FhiMDFormat* format.
to_fhi_aims_output(*args, **kwargs)	Dump data to *dpdata.plugins.fhi_aims.FhiMDFormat* format.
to_fhi_aims_scf(*args, **kwargs)	Dump data to *dpdata.plugins.fhi_aims.FhiSCFFormat* format.
to_finalconfig(*args, **kwargs)	Dump data to *dpdata.plugins.pwmat.PwmatAtomconfigFormat* format.
to_gaussian_gjf(*args, **kwargs)	Dump data to *dpdata.plugins.gaussian.GaussiaGJFFormat* format.
to_gaussian_log(*args, **kwargs)	Dump data to *dpdata.plugins.gaussian.GaussianLogFormat* format.
to_gaussian_md(*args, **kwargs)	Dump data to *dpdata.plugins.gaussian.GaussianMDFormat* format.
to_gro(*args, **kwargs)	Dump data to *dpdata.plugins.gromacs.GromacsGroFormat* format.
to_gromacs_gro(*args, **kwargs)	Dump data to *dpdata.plugins.gromacs.GromacsGroFormat* format.
to_json()	Returns a json string representation of the MSONable object.
to_lammps_dump(*args, **kwargs)	Dump data to *dpdata.plugins.lammps.LAMMPSDumpFormat* format.
to_lammps_lmp(*args, **kwargs)	Dump data to *dpdata.plugins.lammps.LAMMPSLmpFormat* format.
to_list(*args, **kwargs)	Dump data to *dpdata.plugins.list.ListFormat* format.
to_lmp(*args, **kwargs)	Dump data to *dpdata.plugins.lammps.LAMMPSLmpFormat* format.
to_mlmd(*args, **kwargs)	Dump data to *dpdata.plugins.pwmat.PwmatOutputFormat* format.
to_mol(*args, **kwargs)	Dump data to *dpdata.plugins.rdkit.MolFormat* format.
to_mol_file(*args, **kwargs)	Dump data to *dpdata.plugins.rdkit.MolFormat* format.
to_movement(*args, **kwargs)	Dump data to *dpdata.plugins.pwmat.PwmatOutputFormat* format.
to_n2p2(*args, **kwargs)	Dump data to *dpdata.plugins.n2p2.N2P2Format* format.
to_openmx_md(*args, **kwargs)	Dump data to *dpdata.plugins.openmx.OPENMXFormat* format.
to_orca_spout(*args, **kwargs)	Dump data to *dpdata.plugins.orca.ORCASPOutFormat* format.
to_outcar(*args, **kwargs)	Dump data to *dpdata.plugins.vasp.VASPOutcarFormat* format.
to_poscar(*args, **kwargs)	Dump data to *dpdata.plugins.vasp.VASPPoscarFormat* format.
to_psi4_inp(*args, **kwargs)	Dump data to *dpdata.plugins.psi4.PSI4InputFormat* format.

continues on next page

Table 1 – continued from previous page

to_psi4_out(*args, **kwargs)	Dump data to *dpdata.plugins.psi4.PSI4OutFormat* format.
to_pwmat_atomconfig(*args, **kwargs)	Dump data to *dpdata.plugins.pwmat.PwmatAtomconfigFormat* format.
to_pwmat_finalconfig(*args, **kwargs)	Dump data to *dpdata.plugins.pwmat.PwmatAtomconfigFormat* format.
to_pwmat_mlmd(*args, **kwargs)	Dump data to *dpdata.plugins.pwmat.PwmatOutputFormat* format.
to_pwmat_movement(*args, **kwargs)	Dump data to *dpdata.plugins.pwmat.PwmatOutputFormat* format.
to_pwmat_output(*args, **kwargs)	Dump data to *dpdata.plugins.pwmat.PwmatOutputFormat* format.
to_pymatgen_ComputedStructureEntry(*args, ...)	Dump data to *dpdata.plugins.pymatgen.PyMatgenCSEFormat* format.
to_pymatgen_computedstructureentry(*args, ...)	Dump data to *dpdata.plugins.pymatgen.PyMatgenCSEFormat* format.
to_pymatgen_molecule(*args, **kwargs)	Dump data to *dpdata.plugins.pymatgen.PyMatgenMoleculeFormat* format.
to_pymatgen_structure(*args, **kwargs)	Dump data to *dpdata.plugins.pymatgen.PyMatgenStructureFormat* format.
to_qe_cp_traj(*args, **kwargs)	Dump data to *dpdata.plugins.qe.QECPTrajFormat* format.
to_qe_pw_scf(*args, **kwargs)	Dump data to *dpdata.plugins.qe.QECPPWSCFFormat* format.
to_quip_gap_xyz(*args, **kwargs)	Dump data to *dpdata.plugins.xyz.QuipGapXYZFormat* format.
to_quip_gap_xyz_file(*args, **kwargs)	Dump data to *dpdata.plugins.xyz.QuipGapXYZFormat* format.
to_sdf(*args, **kwargs)	Dump data to *dpdata.plugins.rdkit.SdfFormat* format.
to_sdf_file(*args, **kwargs)	Dump data to *dpdata.plugins.rdkit.SdfFormat* format.
to_siesta_aimd_output(*args, **kwargs)	Dump data to *dpdata.plugins.siesta.SiestaAIMDOutputFormat* format.
to_siesta_output(*args, **kwargs)	Dump data to *dpdata.plugins.siesta.SiestaOutputFormat* format.
to_sqm_in(*args, **kwargs)	Dump data to *dpdata.plugins.amber.SQMINFormat* format.
to_sqm_out(*args, **kwargs)	Dump data to *dpdata.plugins.amber.SQMOutFormat* format.
to_stru(*args, **kwargs)	Dump data to *dpdata.plugins.abacus.AbacusSTRUFormat* format.
to_vasp_contcar(*args, **kwargs)	Dump data to *dpdata.plugins.vasp.VASPPoscarFormat* format.
to_vasp_outcar(*args, **kwargs)	Dump data to *dpdata.plugins.vasp.VASPOutcarFormat* format.
to_vasp_poscar(*args, **kwargs)	Dump data to *dpdata.plugins.vasp.VASPPoscarFormat* format.
to_vasp_string(*args, **kwargs)	Dump data to *dpdata.plugins.vasp.VASPStringFormat* format.
to_vasp_xml(*args, **kwargs)	Dump data to *dpdata.plugins.vasp.VASPXMLFormat* format.

continues on next page

Table 1 – continued from previous page

to_xml(*args, **kwargs)	Dump data to *dpdata.plugins.vasp.VASPXMLFormat* format.
to_xyz(*args, **kwargs)	Dump data to *dpdata.plugins.xyz.XYZFormat* format.
unsafe_hash()	Returns an hash of the current object.
validate_monty_v1(_MSONable__input_value)	Pydantic validator with correct signature for pydantic v1.x
validate_monty_v2(_MSONable__input_value, _)	Pydantic validator with correct signature for pydantic v2.x

affine_map
apply_type_map
from_fmt
from_fmt_obj
replace
rot_frame_lower_triangular
rot_lower_triangular
to_fmt_obj

DTYPES = (<dpdata.data_type.DataType object>, <dpdata.data_type.DataType object>, <dpdata.data_type.DataType object>, <dpdata.data_type.DataType object>, <dpdata.data_type.DataType object>, <dpdata.data_type.DataType object>, <dpdata.data_type.DataType object>, <dpdata.data_type.DataType object>, <dpdata.data_type.DataType object>)

copy()

Returns a copy of the system.

from_fmt_obj(*fmtobj*, *file_name*, ***kwargs*)

from_rdkit_mol(*rdkit_mol*)

Initialize from a rdkit.Chem.rdchem.Mol object.

get_bond_order(*begin_atom_idx*, *end_atom_idx*)

Return the bond order between given atoms.

get_charge()

Return the total formal charge of the moleclue.

get_formal_charges()

Return the formal charges on each atom.

get_mol()

Return the rdkit.Mol object.

get_nbonds()

Return the number of bonds.

to_fmt_obj(*fmtobj*, **args*, ***kwargs*)

class dpdata.**LabeledSystem**(*file_name=None*, *fmt='auto'*, *type_map=None*, *begin=0*, *step=1*, *data=None*, *convergence_check=True*, ***kwargs*)

Bases: *System*

The labeled data System.

For example, a labeled water system named *d_example* has two molecules (6 atoms) and *nframes* frames. The labels can be accessed by

- *d_example['energies']* : a numpy array of size nframes
- *d_example['forces']* : a numpy array of size nframes x 6 x 3
- *d_example['virials']* : optional, a numpy array of size nframes x 3 x 3

It is noted that

- The order of frames stored in *'energies'*, *'forces'* and *'virials'* should be consistent with *'atom_types'*, *'cells'* and *'coords'*.
- The order of atoms in **every** frame of *'forces'* should be consistent with *'coords'* and *'atom_types'*.

Parameters

file_name
[str] The file to load the system

fmt
[str]

Format of the file, supported formats are

- `auto`: infered from *file_name*'s extension
- `vasp/xml`: vasp xml
- `vasp/outcar`: vasp OUTCAR
- `deepmd/raw`: deepmd-kit raw
- `deepmd/npy`: deepmd-kit compressed format (numpy binary)
- `qe/cp/traj`: Quantum Espresso CP trajectory files. should have: file_name+'.in', file_name+'.pos', file_name+'.evp' and file_name+'.for'
- `qe/pw/scf`: Quantum Espresso PW single point calculations. Both input and output files are required. If file_name is a string, it denotes the output file name. Input file name is obtained by replacing 'out' by 'in' from file_name. Or file_name is a list, with the first element being the input file name and the second element being the output filename.
- `siesta/output`: siesta SCF output file
- `siesta/aimd_output`: siesta aimd output file
- `gaussian/log`: gaussian logs
- `gaussian/md`: gaussian ab initio molecular dynamics
- `cp2k/output`: cp2k output file
- `cp2k/aimd_output`: cp2k aimd output dir(contains *pos*.xyz and *.log file); optional *restart=True* if it is a cp2k restarted task.
- `pwmat/movement`: pwmat md output file
- `pwmat/out.mlmd`: pwmat scf output file

type_map
[list of str] Maps atom type to name. The atom with type *ii* is mapped to *type_map[ii]*. If not provided the atom names are assigned to *'Type_1'*, *'Type_2'*, *'Type_3'*...

begin
[int] The beginning frame when loading MD trajectory.

step
[int] The number of skipped frames when loading MD trajectory.

Attributes

formula
Return the formula of this system, like C3H5O2.

formula_hash
Return the hash of the formula of this system.

nopbc

short_formula
Return the short formula of this system.

short_name
Return the short name of this system (no more than 255 bytes), in the following order: - formula - short_formula - formula_hash.

uniq_formula
Return the uniq_formula of this system.

Methods

add_atom_names(atom_names)	Add atom_names that do not exist.
append(system)	Append a system to this system.
apply_pbc()	Append periodic boundary condition.
as_dict()	Returns data dict of System instance.
check_data()	Check if data is correct.
check_type_map(type_map)	Assign atom_names to type_map if type_map is given and different from atom_names.
convert_to_mixed_type([type_map])	Convert the data dict to mixed type format structure, in order to append systems with different formula but the same number of atoms.
copy()	Returns a copy of the system.
correction(hl_sys)	Get energy and force correction between self and a high-level LabeledSystem.
dump(filename[, indent])	Dump .json or .yaml file.
extend(systems)	Extend a system list to this system.
from_3dmol(file_name, **kwargs)	Read data from *dpdata.plugins.3dmol.Py3DMolFormat* format.
from_abacus_lcao_md(file_name, **kwargs)	Read data from *dpdata.plugins.abacus.AbacusMDFormat* format.
from_abacus_lcao_relax(file_name, **kwargs)	Read data from *dpdata.plugins.abacus.AbacusRelaxFormat* format.
from_abacus_lcao_scf(file_name, **kwargs)	Read data from *dpdata.plugins.abacus.AbacusSCFFormat* format.

continues on next page

Table 2 – continued from previous page

from_abacus_md(file_name, **kwargs)	Read data from *dpdata.plugins.abacus.AbacusMDFormat* format.
from_abacus_pw_md(file_name, **kwargs)	Read data from *dpdata.plugins.abacus.AbacusMDFormat* format.
from_abacus_pw_relax(file_name, **kwargs)	Read data from *dpdata.plugins.abacus.AbacusRelaxFormat* format.
from_abacus_pw_scf(file_name, **kwargs)	Read data from *dpdata.plugins.abacus.AbacusSCFFormat* format.
from_abacus_relax(file_name, **kwargs)	Read data from *dpdata.plugins.abacus.AbacusRelaxFormat* format.
from_abacus_scf(file_name, **kwargs)	Read data from *dpdata.plugins.abacus.AbacusSCFFormat* format.
from_abacus_stru(file_name, **kwargs)	Read data from *dpdata.plugins.abacus.AbacusSTRUFormat* format.
from_amber_md(file_name, **kwargs)	Read data from *dpdata.plugins.amber.AmberMDFormat* format.
from_ase_structure(file_name, **kwargs)	Read data from *dpdata.plugins.ase.ASEStructureFormat* format.
from_ase_traj(file_name, **kwargs)	Read data from *dpdata.plugins.ase.ASETrajFormat* format.
from_atomconfig(file_name, **kwargs)	Read data from *dpdata.plugins.pwmat.PwmatAtomconfigFormat* format.
from_contcar(file_name, **kwargs)	Read data from *dpdata.plugins.vasp.VASPPoscarFormat* format.
from_cp2k_aimd_output(file_name, **kwargs)	Read data from *dpdata.plugins.cp2k.CP2KAIMDOutputFormat* format.
from_cp2k_output(file_name, **kwargs)	Read data from *dpdata.plugins.cp2k.CP2KOutputFormat* format.
from_deepmd(file_name, **kwargs)	Read data from *dpdata.plugins.deepmd.DeePMDRawFormat* format.
from_deepmd_comp(file_name, **kwargs)	Read data from *dpdata.plugins.deepmd.DeePMDCompFormat* format.
from_deepmd_hdf5(file_name, **kwargs)	Read data from *dpdata.plugins.deepmd.DeePMDHDF5Format* format.
from_deepmd_npy(file_name, **kwargs)	Read data from *dpdata.plugins.deepmd.DeePMDCompFormat* format.
from_deepmd_npy_mixed(file_name, **kwargs)	Read data from *dpdata.plugins.deepmd.DeePMDMixedFormat* format.
from_deepmd_raw(file_name, **kwargs)	Read data from *dpdata.plugins.deepmd.DeePMDRawFormat* format.
from_dftbplus(file_name, **kwargs)	Read data from *dpdata.plugins.dftbplus.DFTBplusFormat* format.
from_dict(d)	**param d** Dict representation.
from_dump(file_name, **kwargs)	Read data from *dpdata.plugins.lammps.LAMMPSDumpFormat* format.
from_fhi_aims_md(file_name, **kwargs)	Read data from *dpdata.plugins.fhi_aims.FhiMDFormat* format.
from_fhi_aims_output(file_name, **kwargs)	Read data from *dpdata.plugins.fhi_aims.FhiMDFormat* format.

continues on next page

Table 2 – continued from previous page

from_fhi_aims_scf(file_name, **kwargs)	Read data from *dpdata.plugins.fhi_aims.FhiSCFFormat* format.
from_finalconfig(file_name, **kwargs)	Read data from *dpdata.plugins.pwmat.PwmatAtomconfigFormat* format.
from_gaussian_gjf(file_name, **kwargs)	Read data from *dpdata.plugins.gaussian.GaussiaGJFFormat* format.
from_gaussian_log(file_name, **kwargs)	Read data from *dpdata.plugins.gaussian.GaussianLogFormat* format.
from_gaussian_md(file_name, **kwargs)	Read data from *dpdata.plugins.gaussian.GaussianMDFormat* format.
from_gro(file_name, **kwargs)	Read data from *dpdata.plugins.gromacs.GromacsGroFormat* format.
from_gromacs_gro(file_name, **kwargs)	Read data from *dpdata.plugins.gromacs.GromacsGroFormat* format.
from_lammps_dump(file_name, **kwargs)	Read data from *dpdata.plugins.lammps.LAMMPSDumpFormat* format.
from_lammps_lmp(file_name, **kwargs)	Read data from *dpdata.plugins.lammps.LAMMPSLmpFormat* format.
from_list(file_name, **kwargs)	Read data from *dpdata.plugins.list.ListFormat* format.
from_lmp(file_name, **kwargs)	Read data from *dpdata.plugins.lammps.LAMMPSLmpFormat* format.
from_mlmd(file_name, **kwargs)	Read data from *dpdata.plugins.pwmat.PwmatOutputFormat* format.
from_mol(file_name, **kwargs)	Read data from *dpdata.plugins.rdkit.MolFormat* format.
from_mol_file(file_name, **kwargs)	Read data from *dpdata.plugins.rdkit.MolFormat* format.
from_movement(file_name, **kwargs)	Read data from *dpdata.plugins.pwmat.PwmatOutputFormat* format.
from_n2p2(file_name, **kwargs)	Read data from *dpdata.plugins.n2p2.N2P2Format* format.
from_openmx_md(file_name, **kwargs)	Read data from *dpdata.plugins.openmx.OPENMXFormat* format.
from_orca_spout(file_name, **kwargs)	Read data from *dpdata.plugins.orca.ORCASPOutFormat* format.
from_outcar(file_name, **kwargs)	Read data from *dpdata.plugins.vasp.VASPOutcarFormat* format.
from_poscar(file_name, **kwargs)	Read data from *dpdata.plugins.vasp.VASPPoscarFormat* format.
from_psi4_inp(file_name, **kwargs)	Read data from *dpdata.plugins.psi4.PSI4InputFormat* format.
from_psi4_out(file_name, **kwargs)	Read data from *dpdata.plugins.psi4.PSI4OutFormat* format.
from_pwmat_atomconfig(file_name, **kwargs)	Read data from *dpdata.plugins.pwmat.PwmatAtomconfigFormat* format.
from_pwmat_finalconfig(file_name, **kwargs)	Read data from *dpdata.plugins.pwmat.PwmatAtomconfigFormat* format.
from_pwmat_mlmd(file_name, **kwargs)	Read data from *dpdata.plugins.pwmat.PwmatOutputFormat* format.
from_pwmat_movement(file_name, **kwargs)	Read data from *dpdata.plugins.pwmat.PwmatOutputFormat* format.

continues on next page

Table 2 – continued from previous page

from_pwmat_output(file_name, **kwargs)	Read data from *dpdata.plugins.pwmat.PwmatOutputFormat* format.
from_pymatgen_computedstructureentry(...)	Read data from *dpdata.plugins.pymatgen.PyMatgenCSEFormat* format.
from_pymatgen_molecule(file_name, **kwargs)	Read data from *dpdata.plugins.pymatgen.PyMatgenMoleculeFormat* format.
from_pymatgen_structure(file_name, **kwargs)	Read data from *dpdata.plugins.pymatgen.PyMatgenStructureFormat* format.
from_qe_cp_traj(file_name, **kwargs)	Read data from *dpdata.plugins.qe.QECPTrajFormat* format.
from_qe_pw_scf(file_name, **kwargs)	Read data from *dpdata.plugins.qe.QECPPWSCFFormat* format.
from_quip_gap_xyz(file_name, **kwargs)	Read data from *dpdata.plugins.xyz.QuipGapXYZFormat* format.
from_quip_gap_xyz_file(file_name, **kwargs)	Read data from *dpdata.plugins.xyz.QuipGapXYZFormat* format.
from_sdf(file_name, **kwargs)	Read data from *dpdata.plugins.rdkit.SdfFormat* format.
from_sdf_file(file_name, **kwargs)	Read data from *dpdata.plugins.rdkit.SdfFormat* format.
from_siesta_aiMD_output(file_name, **kwargs)	Read data from *dpdata.plugins.siesta.SiestaAIMDOutputFormat* format.
from_siesta_aimd_output(file_name, **kwargs)	Read data from *dpdata.plugins.siesta.SiestaAIMDOutputFormat* format.
from_siesta_output(file_name, **kwargs)	Read data from *dpdata.plugins.siesta.SiestaOutputFormat* format.
from_sqm_in(file_name, **kwargs)	Read data from *dpdata.plugins.amber.SQMINFormat* format.
from_sqm_out(file_name, **kwargs)	Read data from *dpdata.plugins.amber.SQMOutFormat* format.
from_stru(file_name, **kwargs)	Read data from *dpdata.plugins.abacus.AbacusSTRUFormat* format.
from_vasp_contcar(file_name, **kwargs)	Read data from *dpdata.plugins.vasp.VASPPoscarFormat* format.
from_vasp_outcar(file_name, **kwargs)	Read data from *dpdata.plugins.vasp.VASPOutcarFormat* format.
from_vasp_poscar(file_name, **kwargs)	Read data from *dpdata.plugins.vasp.VASPPoscarFormat* format.
from_vasp_string(file_name, **kwargs)	Read data from *dpdata.plugins.vasp.VASPStringFormat* format.
from_vasp_xml(file_name, **kwargs)	Read data from *dpdata.plugins.vasp.VASPXMLFormat* format.
from_xml(file_name, **kwargs)	Read data from *dpdata.plugins.vasp.VASPXMLFormat* format.
from_xyz(file_name, **kwargs)	Read data from *dpdata.plugins.xyz.XYZFormat* format.
get_atom_names()	Returns name of atoms.
get_atom_numbs()	Returns number of atoms.
get_atom_types()	Returns type of atoms.
get_natoms()	Returns total number of atoms in the system.
get_nframes()	Returns number of frames in the system.
get_ntypes()	Returns total number of atom types in the system.

continues on next page

Table 2 – continued from previous page

load(filename)	Rebuild System obj.
map_atom_types([type_map])	Map the atom types of the system.
minimize(*args, minimizer, **kwargs)	Minimize the geometry.
perturb(pert_num, cell_pert_fraction, ...[, ...])	Perturb each frame in the system randomly.
pick_atom_idx(idx[, nopbc])	Pick atom index.
pick_by_amber_mask(param, maskstr[, ...])	Pick atoms by amber mask.
predict(*args[, driver])	Predict energies and forces by a driver.
register_data_type(*data_type)	Register data type.
remove_atom_names(atom_names)	Remove atom names and all such atoms.
remove_outlier([threshold])	Remove outlier frames from the system.
remove_pbc([protect_layer])	This method does NOT delete the definition of the cells, it (1) revises the cell to a cubic cell and ensures that the cell boundary to any atom in the system is no less than *protect_layer* (2) translates the system such that the center-of-geometry of the system locates at the center of the cell.
replicate(ncopy)	Replicate the each frame in the system in 3 dimensions.
shuffle()	Shuffle frames randomly.
sort_atom_names([type_map])	Sort atom_names of the system and reorder atom_numbs and atom_types accoarding to atom_names.
sort_atom_types()	Sort atom types.
sub_system(f_idx)	Construct a subsystem from the system.
to(fmt, *args, **kwargs)	Dump systems to the specific format.
to_3dmol(*args, **kwargs)	Dump data to *dpdata.plugins.3dmol.Py3DMolFormat* format.
to_abacus_lcao_md(*args, **kwargs)	Dump data to *dpdata.plugins.abacus.AbacusMDFormat* format.
to_abacus_lcao_relax(*args, **kwargs)	Dump data to *dpdata.plugins.abacus.AbacusRelaxFormat* format.
to_abacus_lcao_scf(*args, **kwargs)	Dump data to *dpdata.plugins.abacus.AbacusSCFFormat* format.
to_abacus_md(*args, **kwargs)	Dump data to *dpdata.plugins.abacus.AbacusMDFormat* format.
to_abacus_pw_md(*args, **kwargs)	Dump data to *dpdata.plugins.abacus.AbacusMDFormat* format.
to_abacus_pw_relax(*args, **kwargs)	Dump data to *dpdata.plugins.abacus.AbacusRelaxFormat* format.
to_abacus_pw_scf(*args, **kwargs)	Dump data to *dpdata.plugins.abacus.AbacusSCFFormat* format.
to_abacus_relax(*args, **kwargs)	Dump data to *dpdata.plugins.abacus.AbacusRelaxFormat* format.
to_abacus_scf(*args, **kwargs)	Dump data to *dpdata.plugins.abacus.AbacusSCFFormat* format.
to_abacus_stru(*args, **kwargs)	Dump data to *dpdata.plugins.abacus.AbacusSTRUFormat* format.
to_amber_md(*args, **kwargs)	Dump data to *dpdata.plugins.amber.AmberMDFormat* format.
to_ase_structure(*args, **kwargs)	Dump data to *dpdata.plugins.ase.ASEStructureFormat* format.

continues on next page

Table 2 – continued from previous page

to_ase_traj(*args, **kwargs)	Dump data to dpdata.plugins.ase.ASETrajFormat format.
to_atomconfig(*args, **kwargs)	Dump data to dpdata.plugins.pwmat.PwmatAtomconfigFormat format.
to_contcar(*args, **kwargs)	Dump data to dpdata.plugins.vasp.VASPPoscarFormat format.
to_cp2k_aimd_output(*args, **kwargs)	Dump data to dpdata.plugins.cp2k.CP2KAIMDOutputFormat format.
to_cp2k_output(*args, **kwargs)	Dump data to dpdata.plugins.cp2k.CP2KOutputFormat format.
to_deepmd(*args, **kwargs)	Dump data to dpdata.plugins.deepmd.DeePMDRawFormat format.
to_deepmd_comp(*args, **kwargs)	Dump data to dpdata.plugins.deepmd.DeePMDCompFormat format.
to_deepmd_hdf5(*args, **kwargs)	Dump data to dpdata.plugins.deepmd.DeePMDHDF5Format format.
to_deepmd_npy(*args, **kwargs)	Dump data to dpdata.plugins.deepmd.DeePMDCompFormat format.
to_deepmd_npy_mixed(*args, **kwargs)	Dump data to dpdata.plugins.deepmd.DeePMDMixedFormat format.
to_deepmd_raw(*args, **kwargs)	Dump data to dpdata.plugins.deepmd.DeePMDRawFormat format.
to_dftbplus(*args, **kwargs)	Dump data to dpdata.plugins.dftbplus.DFTBplusFormat format.
to_dump(*args, **kwargs)	Dump data to dpdata.plugins.lammps.LAMMPSDumpFormat format.
to_fhi_aims_md(*args, **kwargs)	Dump data to dpdata.plugins.fhi_aims.FhiMDFormat format.
to_fhi_aims_output(*args, **kwargs)	Dump data to dpdata.plugins.fhi_aims.FhiMDFormat format.
to_fhi_aims_scf(*args, **kwargs)	Dump data to dpdata.plugins.fhi_aims.FhiSCFFormat format.
to_finalconfig(*args, **kwargs)	Dump data to dpdata.plugins.pwmat.PwmatAtomconfigFormat format.
to_gaussian_gjf(*args, **kwargs)	Dump data to dpdata.plugins.gaussian.GaussiaGJFFormat format.
to_gaussian_log(*args, **kwargs)	Dump data to dpdata.plugins.gaussian.GaussianLogFormat format.
to_gaussian_md(*args, **kwargs)	Dump data to dpdata.plugins.gaussian.GaussianMDFormat format.
to_gro(*args, **kwargs)	Dump data to dpdata.plugins.gromacs.GromacsGroFormat format.
to_gromacs_gro(*args, **kwargs)	Dump data to dpdata.plugins.gromacs.GromacsGroFormat format.
to_json()	Returns a json string representation of the MSONable object.
to_lammps_dump(*args, **kwargs)	Dump data to dpdata.plugins.lammps.LAMMPSDumpFormat format.
to_lammps_lmp(*args, **kwargs)	Dump data to dpdata.plugins.lammps.LAMMPSLmpFormat format.
to_list(*args, **kwargs)	Dump data to dpdata.plugins.list.ListFormat format.

continues on next page

Table 2 – continued from previous page

to_lmp(*args, **kwargs)	Dump data to dpdata.plugins.lammps.LAMMPSLmpFormat format.
to_mlmd(*args, **kwargs)	Dump data to dpdata.plugins.pwmat.PwmatOutputFormat format.
to_mol(*args, **kwargs)	Dump data to dpdata.plugins.rdkit.MolFormat format.
to_mol_file(*args, **kwargs)	Dump data to dpdata.plugins.rdkit.MolFormat format.
to_movement(*args, **kwargs)	Dump data to dpdata.plugins.pwmat.PwmatOutputFormat format.
to_n2p2(*args, **kwargs)	Dump data to dpdata.plugins.n2p2.N2P2Format format.
to_openmx_md(*args, **kwargs)	Dump data to dpdata.plugins.openmx.OPENMXFormat format.
to_orca_spout(*args, **kwargs)	Dump data to dpdata.plugins.orca.ORCASPOutFormat format.
to_outcar(*args, **kwargs)	Dump data to dpdata.plugins.vasp.VASPOutcarFormat format.
to_poscar(*args, **kwargs)	Dump data to dpdata.plugins.vasp.VASPPoscarFormat format.
to_psi4_inp(*args, **kwargs)	Dump data to dpdata.plugins.psi4.PSI4InputFormat format.
to_psi4_out(*args, **kwargs)	Dump data to dpdata.plugins.psi4.PSI4OutFormat format.
to_pwmat_atomconfig(*args, **kwargs)	Dump data to dpdata.plugins.pwmat.PwmatAtomconfigFormat format.
to_pwmat_finalconfig(*args, **kwargs)	Dump data to dpdata.plugins.pwmat.PwmatAtomconfigFormat format.
to_pwmat_mlmd(*args, **kwargs)	Dump data to dpdata.plugins.pwmat.PwmatOutputFormat format.
to_pwmat_movement(*args, **kwargs)	Dump data to dpdata.plugins.pwmat.PwmatOutputFormat format.
to_pwmat_output(*args, **kwargs)	Dump data to dpdata.plugins.pwmat.PwmatOutputFormat format.
to_pymatgen_ComputedStructureEntry(*args, ...)	Dump data to dpdata.plugins.pymatgen.PyMatgenCSEFormat format.
to_pymatgen_computedstructureentry(*args, ...)	Dump data to dpdata.plugins.pymatgen.PyMatgenCSEFormat format.
to_pymatgen_molecule(*args, **kwargs)	Dump data to dpdata.plugins.pymatgen.PyMatgenMoleculeFormat format.
to_pymatgen_structure(*args, **kwargs)	Dump data to dpdata.plugins.pymatgen.PyMatgenStructureFormat format.
to_qe_cp_traj(*args, **kwargs)	Dump data to dpdata.plugins.qe.QECPTrajFormat format.
to_qe_pw_scf(*args, **kwargs)	Dump data to dpdata.plugins.qe.QECPPWSCFFormat format.
to_quip_gap_xyz(*args, **kwargs)	Dump data to dpdata.plugins.xyz.QuipGapXYZFormat format.
to_quip_gap_xyz_file(*args, **kwargs)	Dump data to dpdata.plugins.xyz.QuipGapXYZFormat format.
to_sdf(*args, **kwargs)	Dump data to dpdata.plugins.rdkit.SdfFormat format.

continues on next page

Table 2 – continued from previous page

to_sdf_file(*args, **kwargs)	Dump data to *dpdata.plugins.rdkit.SdfFormat* format.
to_siesta_aimd_output(*args, **kwargs)	Dump data to *dpdata.plugins.siesta.SiestaAIMDOutputFormat* format.
to_siesta_output(*args, **kwargs)	Dump data to *dpdata.plugins.siesta.SiestaOutputFormat* format.
to_sqm_in(*args, **kwargs)	Dump data to *dpdata.plugins.amber.SQMINFormat* format.
to_sqm_out(*args, **kwargs)	Dump data to *dpdata.plugins.amber.SQMOutFormat* format.
to_stru(*args, **kwargs)	Dump data to *dpdata.plugins.abacus.AbacusSTRUFormat* format.
to_vasp_contcar(*args, **kwargs)	Dump data to *dpdata.plugins.vasp.VASPPoscarFormat* format.
to_vasp_outcar(*args, **kwargs)	Dump data to *dpdata.plugins.vasp.VASPOutcarFormat* format.
to_vasp_poscar(*args, **kwargs)	Dump data to *dpdata.plugins.vasp.VASPPoscarFormat* format.
to_vasp_string(*args, **kwargs)	Dump data to *dpdata.plugins.vasp.VASPStringFormat* format.
to_vasp_xml(*args, **kwargs)	Dump data to *dpdata.plugins.vasp.VASPXMLFormat* format.
to_xml(*args, **kwargs)	Dump data to *dpdata.plugins.vasp.VASPXMLFormat* format.
to_xyz(*args, **kwargs)	Dump data to *dpdata.plugins.xyz.XYZFormat* format.
unsafe_hash()	Returns an hash of the current object.
validate_monty_v1(_MSONable__input_value)	Pydantic validator with correct signature for pydantic v1.x
validate_monty_v2(_MSONable__input_value, _)	Pydantic validator with correct signature for pydantic v2.x

affine_map
affine_map_fv
apply_type_map
from_fmt
from_fmt_obj
has_virial
replace
rot_frame_lower_triangular
rot_lower_triangular
to_fmt_obj

```
DTYPES = (<dpdata.data_type.DataType object>, <dpdata.data_type.DataType object>,
<dpdata.data_type.DataType object>, <dpdata.data_type.DataType object>,
<dpdata.data_type.DataType object>, <dpdata.data_type.DataType object>,
<dpdata.data_type.DataType object>, <dpdata.data_type.DataType object>,
<dpdata.data_type.DataType object>, <dpdata.data_type.DataType object>,
<dpdata.data_type.DataType object>)
```

affine_map_fv(*trans*, *f_idx*)

correction(*hl_sys*)

Get energy and force correction between self and a high-level LabeledSystem. The self's coordinates will be kept, but energy and forces will be replaced by the correction between these two systems.

Note: The function will not check whether coordinates and elements of two systems are the same. The user should make sure by itself.

Parameters

hl_sys
[LabeledSystem] high-level LabeledSystem

Returns

corrected_sys: LabeledSystem
Corrected LabeledSystem

from_3dmol(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.3dmol.Py3DMolFormat* format.

from_abacus_lcao_md(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.abacus.AbacusMDFormat* format.

from_abacus_lcao_relax(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.abacus.AbacusRelaxFormat* format.

from_abacus_lcao_scf(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.abacus.AbacusSCFFormat* format.

from_abacus_md(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.abacus.AbacusMDFormat* format.

from_abacus_pw_md(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.abacus.AbacusMDFormat* format.

from_abacus_pw_relax(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.abacus.AbacusRelaxFormat* format.

from_abacus_pw_scf(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.abacus.AbacusSCFFormat* format.

from_abacus_relax(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.abacus.AbacusRelaxFormat* format.

from_abacus_scf(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.abacus.AbacusSCFFormat* format.

from_abacus_stru(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.abacus.AbacusSTRUFormat* format.

from_amber_md(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.amber.AmberMDFormat* format.

from_ase_structure(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.ase.ASEStructureFormat* format.

from_ase_traj(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.ase.ASETrajFormat* format.

from_atomconfig(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.pwmat.PwmatAtomconfigFormat* format.

from_contcar(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.vasp.VASPPoscarFormat* format.

from_cp2k_aimd_output(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.cp2k.CP2KAIMDOutputFormat* format.

from_cp2k_output(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.cp2k.CP2KOutputFormat* format.

from_deepmd(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.deepmd.DeePMDRawFormat* format.

from_deepmd_comp(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.deepmd.DeePMDCompFormat* format.

from_deepmd_hdf5(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.deepmd.DeePMDHDF5Format* format.

from_deepmd_npy(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.deepmd.DeePMDCompFormat* format.

from_deepmd_npy_mixed(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.deepmd.DeePMDMixedFormat* format.

from_deepmd_raw(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.deepmd.DeePMDRawFormat* format.

from_dftbplus(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.dftbplus.DFTBplusFormat* format.

from_dump(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.lammps.LAMMPSDumpFormat* format.

from_fhi_aims_md(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.fhi_aims.FhiMDFormat* format.

from_fhi_aims_output(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.fhi_aims.FhiMDFormat* format.

from_fhi_aims_scf(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.fhi_aims.FhiSCFFormat* format.

from_finalconfig(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.pwmat.PwmatAtomconfigFormat* format.

from_fmt_obj(*fmtobj*, *file_name*, ***kwargs*)

from_gaussian_gjf(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.gaussian.GaussiaGJFFormat* format.

from_gaussian_log(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.gaussian.GaussianLogFormat* format.

from_gaussian_md(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.gaussian.GaussianMDFormat* format.

from_gro(*file_name*, ***kwargs*)
Read data from *dpdata.plugins.gromacs.GromacsGroFormat* format.

from_gromacs_gro(*file_name*, ***kwargs*)
Read data from *dpdata.plugins.gromacs.GromacsGroFormat* format.

from_lammps_dump(*file_name*, ***kwargs*)
Read data from *dpdata.plugins.lammps.LAMMPSDumpFormat* format.

from_lammps_lmp(*file_name*, ***kwargs*)
Read data from *dpdata.plugins.lammps.LAMMPSLmpFormat* format.

from_list(*file_name*, ***kwargs*)
Read data from *dpdata.plugins.list.ListFormat* format.

from_lmp(*file_name*, ***kwargs*)
Read data from *dpdata.plugins.lammps.LAMMPSLmpFormat* format.

from_mlmd(*file_name*, ***kwargs*)
Read data from *dpdata.plugins.pwmat.PwmatOutputFormat* format.

from_mol(*file_name*, ***kwargs*)
Read data from *dpdata.plugins.rdkit.MolFormat* format.

from_mol_file(*file_name*, ***kwargs*)
Read data from *dpdata.plugins.rdkit.MolFormat* format.

from_movement(*file_name*, ***kwargs*)
Read data from *dpdata.plugins.pwmat.PwmatOutputFormat* format.

from_n2p2(*file_name*, ***kwargs*)
Read data from *dpdata.plugins.n2p2.N2P2Format* format.

from_openmx_md(*file_name*, ***kwargs*)
Read data from *dpdata.plugins.openmx.OPENMXFormat* format.

from_orca_spout(*file_name*, ***kwargs*)
Read data from *dpdata.plugins.orca.ORCASPOutFormat* format.

from_outcar(*file_name*, ***kwargs*)
Read data from *dpdata.plugins.vasp.VASPOutcarFormat* format.

from_poscar(*file_name*, ***kwargs*)
Read data from *dpdata.plugins.vasp.VASPPoscarFormat* format.

from_psi4_inp(*file_name*, ***kwargs*)
Read data from *dpdata.plugins.psi4.PSI4InputFormat* format.

from_psi4_out(*file_name*, ***kwargs*)
Read data from *dpdata.plugins.psi4.PSI4OutFormat* format.

from_pwmat_atomconfig(*file_name*, ***kwargs*)
Read data from *dpdata.plugins.pwmat.PwmatAtomconfigFormat* format.

from_pwmat_finalconfig(*file_name*, ***kwargs*)
Read data from *dpdata.plugins.pwmat.PwmatAtomconfigFormat* format.

from_pwmat_mlmd(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.pwmat.PwmatOutputFormat* format.

from_pwmat_movement(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.pwmat.PwmatOutputFormat* format.

from_pwmat_output(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.pwmat.PwmatOutputFormat* format.

from_pymatgen_computedstructureentry(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.pymatgen.PyMatgenCSEFormat* format.

from_pymatgen_molecule(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.pymatgen.PyMatgenMoleculeFormat* format.

from_pymatgen_structure(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.pymatgen.PyMatgenStructureFormat* format.

from_qe_cp_traj(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.qe.QECPTrajFormat* format.

from_qe_pw_scf(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.qe.QECPPWSCFFormat* format.

from_quip_gap_xyz(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.xyz.QuipGapXYZFormat* format.

from_quip_gap_xyz_file(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.xyz.QuipGapXYZFormat* format.

from_sdf(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.rdkit.SdfFormat* format.

from_sdf_file(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.rdkit.SdfFormat* format.

from_siesta_aiMD_output(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.siesta.SiestaAIMDOutputFormat* format.

from_siesta_aimd_output(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.siesta.SiestaAIMDOutputFormat* format.

from_siesta_output(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.siesta.SiestaOutputFormat* format.

from_sqm_in(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.amber.SQMINFormat* format.

from_sqm_out(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.amber.SQMOutFormat* format.

from_stru(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.abacus.AbacusSTRUFormat* format.

from_vasp_contcar(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.vasp.VASPPoscarFormat* format.

from_vasp_outcar(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.vasp.VASPOutcarFormat* format.

from_vasp_poscar(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.vasp.VASPPoscarFormat* format.

from_vasp_string(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.vasp.VASPStringFormat* format.

from_vasp_xml(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.vasp.VASPXMLFormat* format.

from_xml(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.vasp.VASPXMLFormat* format.

from_xyz(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.xyz.XYZFormat* format.

has_virial()

post_funcs = <dpdata.plugin.Plugin object>

remove_outlier(*threshold: float = 8.0*) → *LabeledSystem*

Remove outlier frames from the system.

Remove the frames whose energies satisfy the condition

$$\frac{\left\|E-\bar{E}\right\|}{\sigma(E)} \geq \text{threshold}$$

where $\bar{E}$ and $\sigma(E)$ are the mean and standard deviation of the energies in the system.

Parameters

threshold
[float] The threshold of outlier detection. The default value is 8.0.

Returns

LabeledSystem
The system without outlier frames.

References

[1], [2]

rot_frame_lower_triangular(*f_idx=0*)

to_3dmol(**args*, ***kwargs*)

Dump data to *dpdata.plugins.3dmol.Py3DMolFormat* format.

to_abacus_lcao_md(**args*, ***kwargs*)

Dump data to *dpdata.plugins.abacus.AbacusMDFormat* format.

to_abacus_lcao_relax(**args*, ***kwargs*)

Dump data to *dpdata.plugins.abacus.AbacusRelaxFormat* format.

to_abacus_lcao_scf(**args*, ***kwargs*)

Dump data to *dpdata.plugins.abacus.AbacusSCFFormat* format.

to_abacus_md(**args*, ***kwargs*)

Dump data to *dpdata.plugins.abacus.AbacusMDFormat* format.

to_abacus_pw_md(**args*, ***kwargs*)

Dump data to *dpdata.plugins.abacus.AbacusMDFormat* format.

to_abacus_pw_relax(**args*, ***kwargs*)

Dump data to *dpdata.plugins.abacus.AbacusRelaxFormat* format.

to_abacus_pw_scf(**args*, ***kwargs*)

Dump data to *dpdata.plugins.abacus.AbacusSCFFormat* format.

to_abacus_relax(**args*, ***kwargs*)

Dump data to *dpdata.plugins.abacus.AbacusRelaxFormat* format.

to_abacus_scf(**args*, ***kwargs*)

Dump data to *dpdata.plugins.abacus.AbacusSCFFormat* format.

to_abacus_stru(**args*, ***kwargs*)

Dump data to *dpdata.plugins.abacus.AbacusSTRUFormat* format.

to_amber_md(**args*, ***kwargs*)

Dump data to *dpdata.plugins.amber.AmberMDFormat* format.

to_ase_structure(**args*, ***kwargs*)

Dump data to *dpdata.plugins.ase.ASEStructureFormat* format.

to_ase_traj(**args*, ***kwargs*)

Dump data to *dpdata.plugins.ase.ASETrajFormat* format.

to_atomconfig(**args*, ***kwargs*)

Dump data to *dpdata.plugins.pwmat.PwmatAtomconfigFormat* format.

to_contcar(**args*, ***kwargs*)

Dump data to *dpdata.plugins.vasp.VASPPoscarFormat* format.

to_cp2k_aimd_output(**args*, ***kwargs*)

Dump data to *dpdata.plugins.cp2k.CP2KAIMDOutputFormat* format.

to_cp2k_output(**args*, ***kwargs*)

Dump data to *dpdata.plugins.cp2k.CP2KOutputFormat* format.

to_deepmd(**args*, ***kwargs*)

Dump data to *dpdata.plugins.deepmd.DeePMDRawFormat* format.

to_deepmd_comp(**args*, ***kwargs*)

Dump data to *dpdata.plugins.deepmd.DeePMDCompFormat* format.

to_deepmd_hdf5(**args*, ***kwargs*)

Dump data to *dpdata.plugins.deepmd.DeePMDHDF5Format* format.

to_deepmd_npy(**args*, ***kwargs*)

Dump data to *dpdata.plugins.deepmd.DeePMDCompFormat* format.

to_deepmd_npy_mixed(**args*, ***kwargs*)

Dump data to *dpdata.plugins.deepmd.DeePMDMixedFormat* format.

to_deepmd_raw(**args*, ***kwargs*)
Dump data to *dpdata.plugins.deepmd.DeePMDRawFormat* format.

to_dftbplus(**args*, ***kwargs*)
Dump data to *dpdata.plugins.dftbplus.DFTBplusFormat* format.

to_dump(**args*, ***kwargs*)
Dump data to *dpdata.plugins.lammps.LAMMPSDumpFormat* format.

to_fhi_aims_md(**args*, ***kwargs*)
Dump data to *dpdata.plugins.fhi_aims.FhiMDFormat* format.

to_fhi_aims_output(**args*, ***kwargs*)
Dump data to *dpdata.plugins.fhi_aims.FhiMDFormat* format.

to_fhi_aims_scf(**args*, ***kwargs*)
Dump data to *dpdata.plugins.fhi_aims.FhiSCFFormat* format.

to_finalconfig(**args*, ***kwargs*)
Dump data to *dpdata.plugins.pwmat.PwmatAtomconfigFormat* format.

to_fmt_obj(*fmtobj*, **args*, ***kwargs*)

to_gaussian_gjf(**args*, ***kwargs*)
Dump data to *dpdata.plugins.gaussian.GaussiaGJFFormat* format.

to_gaussian_log(**args*, ***kwargs*)
Dump data to *dpdata.plugins.gaussian.GaussianLogFormat* format.

to_gaussian_md(**args*, ***kwargs*)
Dump data to *dpdata.plugins.gaussian.GaussianMDFormat* format.

to_gro(**args*, ***kwargs*)
Dump data to *dpdata.plugins.gromacs.GromacsGroFormat* format.

to_gromacs_gro(**args*, ***kwargs*)
Dump data to *dpdata.plugins.gromacs.GromacsGroFormat* format.

to_lammps_dump(**args*, ***kwargs*)
Dump data to *dpdata.plugins.lammps.LAMMPSDumpFormat* format.

to_lammps_lmp(**args*, ***kwargs*)
Dump data to *dpdata.plugins.lammps.LAMMPSLmpFormat* format.

to_list(**args*, ***kwargs*)
Dump data to *dpdata.plugins.list.ListFormat* format.

to_lmp(**args*, ***kwargs*)
Dump data to *dpdata.plugins.lammps.LAMMPSLmpFormat* format.

to_mlmd(**args*, ***kwargs*)
Dump data to *dpdata.plugins.pwmat.PwmatOutputFormat* format.

to_mol(**args*, ***kwargs*)
Dump data to *dpdata.plugins.rdkit.MolFormat* format.

to_mol_file(**args*, ***kwargs*)
Dump data to *dpdata.plugins.rdkit.MolFormat* format.

to_movement(**args*, ***kwargs*)

Dump data to *dpdata.plugins.pwmat.PwmatOutputFormat* format.

to_n2p2(**args*, ***kwargs*)

Dump data to *dpdata.plugins.n2p2.N2P2Format* format.

to_openmx_md(**args*, ***kwargs*)

Dump data to *dpdata.plugins.openmx.OPENMXFormat* format.

to_orca_spout(**args*, ***kwargs*)

Dump data to *dpdata.plugins.orca.ORCASPOutFormat* format.

to_outcar(**args*, ***kwargs*)

Dump data to *dpdata.plugins.vasp.VASPOutcarFormat* format.

to_poscar(**args*, ***kwargs*)

Dump data to *dpdata.plugins.vasp.VASPPoscarFormat* format.

to_psi4_inp(**args*, ***kwargs*)

Dump data to *dpdata.plugins.psi4.PSI4InputFormat* format.

to_psi4_out(**args*, ***kwargs*)

Dump data to *dpdata.plugins.psi4.PSI4OutFormat* format.

to_pwmat_atomconfig(**args*, ***kwargs*)

Dump data to *dpdata.plugins.pwmat.PwmatAtomconfigFormat* format.

to_pwmat_finalconfig(**args*, ***kwargs*)

Dump data to *dpdata.plugins.pwmat.PwmatAtomconfigFormat* format.

to_pwmat_mlmd(**args*, ***kwargs*)

Dump data to *dpdata.plugins.pwmat.PwmatOutputFormat* format.

to_pwmat_movement(**args*, ***kwargs*)

Dump data to *dpdata.plugins.pwmat.PwmatOutputFormat* format.

to_pwmat_output(**args*, ***kwargs*)

Dump data to *dpdata.plugins.pwmat.PwmatOutputFormat* format.

to_pymatgen_ComputedStructureEntry(**args*, ***kwargs*)

Dump data to *dpdata.plugins.pymatgen.PyMatgenCSEFormat* format.

to_pymatgen_computedstructureentry(**args*, ***kwargs*)

Dump data to *dpdata.plugins.pymatgen.PyMatgenCSEFormat* format.

to_pymatgen_molecule(**args*, ***kwargs*)

Dump data to *dpdata.plugins.pymatgen.PyMatgenMoleculeFormat* format.

to_pymatgen_structure(**args*, ***kwargs*)

Dump data to *dpdata.plugins.pymatgen.PyMatgenStructureFormat* format.

to_qe_cp_traj(**args*, ***kwargs*)

Dump data to *dpdata.plugins.qe.QECPTrajFormat* format.

to_qe_pw_scf(**args*, ***kwargs*)

Dump data to *dpdata.plugins.qe.QECPPWSCFFormat* format.

to_quip_gap_xyz(**args*, ***kwargs*)

Dump data to *dpdata.plugins.xyz.QuipGapXYZFormat* format.

to_quip_gap_xyz_file(**args*, ***kwargs*)

Dump data to *dpdata.plugins.xyz.QuipGapXYZFormat* format.

to_sdf(**args*, ***kwargs*)

Dump data to *dpdata.plugins.rdkit.SdfFormat* format.

to_sdf_file(**args*, ***kwargs*)

Dump data to *dpdata.plugins.rdkit.SdfFormat* format.

to_siesta_aimd_output(**args*, ***kwargs*)

Dump data to *dpdata.plugins.siesta.SiestaAIMDOutputFormat* format.

to_siesta_output(**args*, ***kwargs*)

Dump data to *dpdata.plugins.siesta.SiestaOutputFormat* format.

to_sqm_in(**args*, ***kwargs*)

Dump data to *dpdata.plugins.amber.SQMINFormat* format.

to_sqm_out(**args*, ***kwargs*)

Dump data to *dpdata.plugins.amber.SQMOutFormat* format.

to_stru(**args*, ***kwargs*)

Dump data to *dpdata.plugins.abacus.AbacusSTRUFormat* format.

to_vasp_contcar(**args*, ***kwargs*)

Dump data to *dpdata.plugins.vasp.VASPPoscarFormat* format.

to_vasp_outcar(**args*, ***kwargs*)

Dump data to *dpdata.plugins.vasp.VASPOutcarFormat* format.

to_vasp_poscar(**args*, ***kwargs*)

Dump data to *dpdata.plugins.vasp.VASPPoscarFormat* format.

to_vasp_string(**args*, ***kwargs*)

Dump data to *dpdata.plugins.vasp.VASPStringFormat* format.

to_vasp_xml(**args*, ***kwargs*)

Dump data to *dpdata.plugins.vasp.VASPXMLFormat* format.

to_xml(**args*, ***kwargs*)

Dump data to *dpdata.plugins.vasp.VASPXMLFormat* format.

to_xyz(**args*, ***kwargs*)

Dump data to *dpdata.plugins.xyz.XYZFormat* format.

class dpdata.**MultiSystems**(**systems*, *type_map=None*)

Bases: `object`

A set containing several systems.

Methods

append(*systems)	Append systems or MultiSystems to systems.
check_atom_names(system)	Make atom_names in all systems equal, prevent inconsistent atom_types.
correction(hl_sys)	Get energy and force correction between self (assumed low-level) and a high-level MultiSystems.
from_3dmol(file_name, **kwargs)	Read data from *dpdata.plugins.3dmol.Py3DMolFormat* format.
from_abacus_lcao_md(file_name, **kwargs)	Read data from *dpdata.plugins.abacus.AbacusMDFormat* format.
from_abacus_lcao_relax(file_name, **kwargs)	Read data from *dpdata.plugins.abacus.AbacusRelaxFormat* format.
from_abacus_lcao_scf(file_name, **kwargs)	Read data from *dpdata.plugins.abacus.AbacusSCFFormat* format.
from_abacus_md(file_name, **kwargs)	Read data from *dpdata.plugins.abacus.AbacusMDFormat* format.
from_abacus_pw_md(file_name, **kwargs)	Read data from *dpdata.plugins.abacus.AbacusMDFormat* format.
from_abacus_pw_relax(file_name, **kwargs)	Read data from *dpdata.plugins.abacus.AbacusRelaxFormat* format.
from_abacus_pw_scf(file_name, **kwargs)	Read data from *dpdata.plugins.abacus.AbacusSCFFormat* format.
from_abacus_relax(file_name, **kwargs)	Read data from *dpdata.plugins.abacus.AbacusRelaxFormat* format.
from_abacus_scf(file_name, **kwargs)	Read data from *dpdata.plugins.abacus.AbacusSCFFormat* format.
from_abacus_stru(file_name, **kwargs)	Read data from *dpdata.plugins.abacus.AbacusSTRUFormat* format.
from_amber_md(file_name, **kwargs)	Read data from *dpdata.plugins.amber.AmberMDFormat* format.
from_ase_structure(file_name, **kwargs)	Read data from *dpdata.plugins.ase.ASEStructureFormat* format.
from_ase_traj(file_name, **kwargs)	Read data from *dpdata.plugins.ase.ASETrajFormat* format.
from_atomconfig(file_name, **kwargs)	Read data from *dpdata.plugins.pwmat.PwmatAtomconfigFormat* format.
from_contcar(file_name, **kwargs)	Read data from *dpdata.plugins.vasp.VASPPoscarFormat* format.
from_cp2k_aimd_output(file_name, **kwargs)	Read data from *dpdata.plugins.cp2k.CP2KAIMDOutputFormat* format.
from_cp2k_output(file_name, **kwargs)	Read data from *dpdata.plugins.cp2k.CP2KOutputFormat* format.
from_deepmd(file_name, **kwargs)	Read data from *dpdata.plugins.deepmd.DeePMDRawFormat* format.
from_deepmd_comp(file_name, **kwargs)	Read data from *dpdata.plugins.deepmd.DeePMDCompFormat* format.
from_deepmd_hdf5(file_name, **kwargs)	Read data from *dpdata.plugins.deepmd.DeePMDHDF5Format* format.
from_deepmd_npy(file_name, **kwargs)	Read data from *dpdata.plugins.deepmd.DeePMDCompFormat* format.

continues on next page

Table 3 – continued from previous page

from_deepmd_npy_mixed(file_name, **kwargs)	Read data from *dpdata.plugins.deepmd.DeePMDMixedFormat* format.
from_deepmd_raw(file_name, **kwargs)	Read data from *dpdata.plugins.deepmd.DeePMDRawFormat* format.
from_dftbplus(file_name, **kwargs)	Read data from *dpdata.plugins.dftbplus.DFTBplusFormat* format.
from_dump(file_name, **kwargs)	Read data from *dpdata.plugins.lammps.LAMMPSDumpFormat* format.
from_fhi_aims_md(file_name, **kwargs)	Read data from *dpdata.plugins.fhi_aims.FhiMDFormat* format.
from_fhi_aims_output(file_name, **kwargs)	Read data from *dpdata.plugins.fhi_aims.FhiMDFormat* format.
from_fhi_aims_scf(file_name, **kwargs)	Read data from *dpdata.plugins.fhi_aims.FhiSCFFormat* format.
from_finalconfig(file_name, **kwargs)	Read data from *dpdata.plugins.pwmat.PwmatAtomconfigFormat* format.
from_gaussian_gjf(file_name, **kwargs)	Read data from *dpdata.plugins.gaussian.GaussiaGJFFormat* format.
from_gaussian_log(file_name, **kwargs)	Read data from *dpdata.plugins.gaussian.GaussianLogFormat* format.
from_gaussian_md(file_name, **kwargs)	Read data from *dpdata.plugins.gaussian.GaussianMDFormat* format.
from_gro(file_name, **kwargs)	Read data from *dpdata.plugins.gromacs.GromacsGroFormat* format.
from_gromacs_gro(file_name, **kwargs)	Read data from *dpdata.plugins.gromacs.GromacsGroFormat* format.
from_lammps_dump(file_name, **kwargs)	Read data from *dpdata.plugins.lammps.LAMMPSDumpFormat* format.
from_lammps_lmp(file_name, **kwargs)	Read data from *dpdata.plugins.lammps.LAMMPSLmpFormat* format.
from_list(file_name, **kwargs)	Read data from *dpdata.plugins.list.ListFormat* format.
from_lmp(file_name, **kwargs)	Read data from *dpdata.plugins.lammps.LAMMPSLmpFormat* format.
from_mlmd(file_name, **kwargs)	Read data from *dpdata.plugins.pwmat.PwmatOutputFormat* format.
from_mol(file_name, **kwargs)	Read data from *dpdata.plugins.rdkit.MolFormat* format.
from_mol_file(file_name, **kwargs)	Read data from *dpdata.plugins.rdkit.MolFormat* format.
from_movement(file_name, **kwargs)	Read data from *dpdata.plugins.pwmat.PwmatOutputFormat* format.
from_n2p2(file_name, **kwargs)	Read data from *dpdata.plugins.n2p2.N2P2Format* format.
from_openmx_md(file_name, **kwargs)	Read data from *dpdata.plugins.openmx.OPENMXFormat* format.
from_orca_spout(file_name, **kwargs)	Read data from *dpdata.plugins.orca.ORCASPOutFormat* format.
from_outcar(file_name, **kwargs)	Read data from *dpdata.plugins.vasp.VASPOutcarFormat* format.
from_poscar(file_name, **kwargs)	Read data from *dpdata.plugins.vasp.VASPPoscarFormat* format.

continues on next page

Table 3 – continued from previous page

from_psi4_inp(file_name, **kwargs)	Read data from *dpdata.plugins.psi4.PSI4InputFormat* format.
from_psi4_out(file_name, **kwargs)	Read data from *dpdata.plugins.psi4.PSI4OutFormat* format.
from_pwmat_atomconfig(file_name, **kwargs)	Read data from *dpdata.plugins.pwmat.PwmatAtomconfigFormat* format.
from_pwmat_finalconfig(file_name, **kwargs)	Read data from *dpdata.plugins.pwmat.PwmatAtomconfigFormat* format.
from_pwmat_mlmd(file_name, **kwargs)	Read data from *dpdata.plugins.pwmat.PwmatOutputFormat* format.
from_pwmat_movement(file_name, **kwargs)	Read data from *dpdata.plugins.pwmat.PwmatOutputFormat* format.
from_pwmat_output(file_name, **kwargs)	Read data from *dpdata.plugins.pwmat.PwmatOutputFormat* format.
from_pymatgen_computedstructureentry(...)	Read data from *dpdata.plugins.pymatgen.PyMatgenCSEFormat* format.
from_pymatgen_molecule(file_name, **kwargs)	Read data from *dpdata.plugins.pymatgen.PyMatgenMoleculeFormat* format.
from_pymatgen_structure(file_name, **kwargs)	Read data from *dpdata.plugins.pymatgen.PyMatgenStructureFormat* format.
from_qe_cp_traj(file_name, **kwargs)	Read data from *dpdata.plugins.qe.QECPTrajFormat* format.
from_qe_pw_scf(file_name, **kwargs)	Read data from *dpdata.plugins.qe.QECPPWSCFFormat* format.
from_quip_gap_xyz(file_name, **kwargs)	Read data from *dpdata.plugins.xyz.QuipGapXYZFormat* format.
from_quip_gap_xyz_file(file_name, **kwargs)	Read data from *dpdata.plugins.xyz.QuipGapXYZFormat* format.
from_sdf(file_name, **kwargs)	Read data from *dpdata.plugins.rdkit.SdfFormat* format.
from_sdf_file(file_name, **kwargs)	Read data from *dpdata.plugins.rdkit.SdfFormat* format.
from_siesta_aiMD_output(file_name, **kwargs)	Read data from *dpdata.plugins.siesta.SiestaAIMDOutputFormat* format.
from_siesta_aimd_output(file_name, **kwargs)	Read data from *dpdata.plugins.siesta.SiestaAIMDOutputFormat* format.
from_siesta_output(file_name, **kwargs)	Read data from *dpdata.plugins.siesta.SiestaOutputFormat* format.
from_sqm_in(file_name, **kwargs)	Read data from *dpdata.plugins.amber.SQMINFormat* format.
from_sqm_out(file_name, **kwargs)	Read data from *dpdata.plugins.amber.SQMOutFormat* format.
from_stru(file_name, **kwargs)	Read data from *dpdata.plugins.abacus.AbacusSTRUFormat* format.
from_vasp_contcar(file_name, **kwargs)	Read data from *dpdata.plugins.vasp.VASPPoscarFormat* format.
from_vasp_outcar(file_name, **kwargs)	Read data from *dpdata.plugins.vasp.VASPOutcarFormat* format.
from_vasp_poscar(file_name, **kwargs)	Read data from *dpdata.plugins.vasp.VASPPoscarFormat* format.
from_vasp_string(file_name, **kwargs)	Read data from *dpdata.plugins.vasp.VASPStringFormat* format.

continues on next page

Table 3 – continued from previous page

from_vasp_xml(file_name, **kwargs)	Read data from *dpdata.plugins.vasp.VASPXMLFormat* format.
from_xml(file_name, **kwargs)	Read data from *dpdata.plugins.vasp.VASPXMLFormat* format.
from_xyz(file_name, **kwargs)	Read data from *dpdata.plugins.xyz.XYZFormat* format.
get_nframes()	Returns number of frames in all systems.
minimize(*args, minimizer, **kwargs)	Minimize geometry by a minimizer.
pick_atom_idx(idx[, nopbc])	Pick atom index.
predict(*args[, driver])	Predict energies and forces by a driver.
to(fmt, *args, **kwargs)	Dump systems to the specific format.
to_3dmol(*args, **kwargs)	Dump data to *dpdata.plugins.3dmol.Py3DMolFormat* format.
to_abacus_lcao_md(*args, **kwargs)	Dump data to *dpdata.plugins.abacus.AbacusMDFormat* format.
to_abacus_lcao_relax(*args, **kwargs)	Dump data to *dpdata.plugins.abacus.AbacusRelaxFormat* format.
to_abacus_lcao_scf(*args, **kwargs)	Dump data to *dpdata.plugins.abacus.AbacusSCFFormat* format.
to_abacus_md(*args, **kwargs)	Dump data to *dpdata.plugins.abacus.AbacusMDFormat* format.
to_abacus_pw_md(*args, **kwargs)	Dump data to *dpdata.plugins.abacus.AbacusMDFormat* format.
to_abacus_pw_relax(*args, **kwargs)	Dump data to *dpdata.plugins.abacus.AbacusRelaxFormat* format.
to_abacus_pw_scf(*args, **kwargs)	Dump data to *dpdata.plugins.abacus.AbacusSCFFormat* format.
to_abacus_relax(*args, **kwargs)	Dump data to *dpdata.plugins.abacus.AbacusRelaxFormat* format.
to_abacus_scf(*args, **kwargs)	Dump data to *dpdata.plugins.abacus.AbacusSCFFormat* format.
to_abacus_stru(*args, **kwargs)	Dump data to *dpdata.plugins.abacus.AbacusSTRUFormat* format.
to_amber_md(*args, **kwargs)	Dump data to *dpdata.plugins.amber.AmberMDFormat* format.
to_ase_structure(*args, **kwargs)	Dump data to *dpdata.plugins.ase.ASEStructureFormat* format.
to_ase_traj(*args, **kwargs)	Dump data to *dpdata.plugins.ase.ASETrajFormat* format.
to_atomconfig(*args, **kwargs)	Dump data to *dpdata.plugins.pwmat.PwmatAtomconfigFormat* format.
to_contcar(*args, **kwargs)	Dump data to *dpdata.plugins.vasp.VASPPoscarFormat* format.
to_cp2k_aimd_output(*args, **kwargs)	Dump data to *dpdata.plugins.cp2k.CP2KAIMDOutputFormat* format.
to_cp2k_output(*args, **kwargs)	Dump data to *dpdata.plugins.cp2k.CP2KOutputFormat* format.
to_deepmd(*args, **kwargs)	Dump data to *dpdata.plugins.deepmd.DeePMDRawFormat* format.
to_deepmd_comp(*args, **kwargs)	Dump data to *dpdata.plugins.deepmd.DeePMDCompFormat* format.

continues on next page

Table 3 – continued from previous page

to_deepmd_hdf5(*args, **kwargs)	Dump data to dpdata.plugins.deepmd.DeePMDHDF5Format format.
to_deepmd_npy(*args, **kwargs)	Dump data to dpdata.plugins.deepmd.DeePMDCompFormat format.
to_deepmd_npy_mixed(*args, **kwargs)	Dump data to dpdata.plugins.deepmd.DeePMDMixedFormat format.
to_deepmd_raw(*args, **kwargs)	Dump data to dpdata.plugins.deepmd.DeePMDRawFormat format.
to_dftbplus(*args, **kwargs)	Dump data to dpdata.plugins.dftbplus.DFTBplusFormat format.
to_dump(*args, **kwargs)	Dump data to dpdata.plugins.lammps.LAMMPSDumpFormat format.
to_fhi_aims_md(*args, **kwargs)	Dump data to dpdata.plugins.fhi_aims.FhiMDFormat format.
to_fhi_aims_output(*args, **kwargs)	Dump data to dpdata.plugins.fhi_aims.FhiMDFormat format.
to_fhi_aims_scf(*args, **kwargs)	Dump data to dpdata.plugins.fhi_aims.FhiSCFFormat format.
to_finalconfig(*args, **kwargs)	Dump data to dpdata.plugins.pwmat.PwmatAtomconfigFormat format.
to_gaussian_gjf(*args, **kwargs)	Dump data to dpdata.plugins.gaussian.GaussiaGJFFormat format.
to_gaussian_log(*args, **kwargs)	Dump data to dpdata.plugins.gaussian.GaussianLogFormat format.
to_gaussian_md(*args, **kwargs)	Dump data to dpdata.plugins.gaussian.GaussianMDFormat format.
to_gro(*args, **kwargs)	Dump data to dpdata.plugins.gromacs.GromacsGroFormat format.
to_gromacs_gro(*args, **kwargs)	Dump data to dpdata.plugins.gromacs.GromacsGroFormat format.
to_lammps_dump(*args, **kwargs)	Dump data to dpdata.plugins.lammps.LAMMPSDumpFormat format.
to_lammps_lmp(*args, **kwargs)	Dump data to dpdata.plugins.lammps.LAMMPSLmpFormat format.
to_list(*args, **kwargs)	Dump data to dpdata.plugins.list.ListFormat format.
to_lmp(*args, **kwargs)	Dump data to dpdata.plugins.lammps.LAMMPSLmpFormat format.
to_mlmd(*args, **kwargs)	Dump data to dpdata.plugins.pwmat.PwmatOutputFormat format.
to_mol(*args, **kwargs)	Dump data to dpdata.plugins.rdkit.MolFormat format.
to_mol_file(*args, **kwargs)	Dump data to dpdata.plugins.rdkit.MolFormat format.
to_movement(*args, **kwargs)	Dump data to dpdata.plugins.pwmat.PwmatOutputFormat format.
to_n2p2(*args, **kwargs)	Dump data to dpdata.plugins.n2p2.N2P2Format format.
to_openmx_md(*args, **kwargs)	Dump data to dpdata.plugins.openmx.OPENMXFormat format.
to_orca_spout(*args, **kwargs)	Dump data to dpdata.plugins.orca.ORCASPOutFormat format.

continues on next page

Table 3 – continued from previous page

to_outcar(*args, **kwargs)	Dump data to dpdata.plugins.vasp.VASPOutcarFormat format.
to_poscar(*args, **kwargs)	Dump data to dpdata.plugins.vasp.VASPPoscarFormat format.
to_psi4_inp(*args, **kwargs)	Dump data to dpdata.plugins.psi4.PSI4InputFormat format.
to_psi4_out(*args, **kwargs)	Dump data to dpdata.plugins.psi4.PSI4OutFormat format.
to_pwmat_atomconfig(*args, **kwargs)	Dump data to dpdata.plugins.pwmat.PwmatAtomconfigFormat format.
to_pwmat_finalconfig(*args, **kwargs)	Dump data to dpdata.plugins.pwmat.PwmatAtomconfigFormat format.
to_pwmat_mlmd(*args, **kwargs)	Dump data to dpdata.plugins.pwmat.PwmatOutputFormat format.
to_pwmat_movement(*args, **kwargs)	Dump data to dpdata.plugins.pwmat.PwmatOutputFormat format.
to_pwmat_output(*args, **kwargs)	Dump data to dpdata.plugins.pwmat.PwmatOutputFormat format.
to_pymatgen_ComputedStructureEntry(*args, ...)	Dump data to dpdata.plugins.pymatgen.PyMatgenCSEFormat format.
to_pymatgen_computedstructureentry(*args, ...)	Dump data to dpdata.plugins.pymatgen.PyMatgenCSEFormat format.
to_pymatgen_molecule(*args, **kwargs)	Dump data to dpdata.plugins.pymatgen.PyMatgenMoleculeFormat format.
to_pymatgen_structure(*args, **kwargs)	Dump data to dpdata.plugins.pymatgen.PyMatgenStructureFormat format.
to_qe_cp_traj(*args, **kwargs)	Dump data to dpdata.plugins.qe.QECPTrajFormat format.
to_qe_pw_scf(*args, **kwargs)	Dump data to dpdata.plugins.qe.QECPPWSCFFormat format.
to_quip_gap_xyz(*args, **kwargs)	Dump data to dpdata.plugins.xyz.QuipGapXYZFormat format.
to_quip_gap_xyz_file(*args, **kwargs)	Dump data to dpdata.plugins.xyz.QuipGapXYZFormat format.
to_sdf(*args, **kwargs)	Dump data to dpdata.plugins.rdkit.SdfFormat format.
to_sdf_file(*args, **kwargs)	Dump data to dpdata.plugins.rdkit.SdfFormat format.
to_siesta_aimd_output(*args, **kwargs)	Dump data to dpdata.plugins.siesta.SiestaAIMDOutputFormat format.
to_siesta_output(*args, **kwargs)	Dump data to dpdata.plugins.siesta.SiestaOutputFormat format.
to_sqm_in(*args, **kwargs)	Dump data to dpdata.plugins.amber.SQMINFormat format.
to_sqm_out(*args, **kwargs)	Dump data to dpdata.plugins.amber.SQMOutFormat format.
to_stru(*args, **kwargs)	Dump data to dpdata.plugins.abacus.AbacusSTRUFormat format.
to_vasp_contcar(*args, **kwargs)	Dump data to dpdata.plugins.vasp.VASPPoscarFormat format.
to_vasp_outcar(*args, **kwargs)	Dump data to dpdata.plugins.vasp.VASPOutcarFormat format.

continues on next page

Table 3 – continued from previous page

to_vasp_poscar(*args, **kwargs)	Dump data to *dpdata.plugins.vasp.VASPPoscarFormat* format.
to_vasp_string(*args, **kwargs)	Dump data to *dpdata.plugins.vasp.VASPStringFormat* format.
to_vasp_xml(*args, **kwargs)	Dump data to *dpdata.plugins.vasp.VASPXMLFormat* format.
to_xml(*args, **kwargs)	Dump data to *dpdata.plugins.vasp.VASPXMLFormat* format.
to_xyz(*args, **kwargs)	Dump data to *dpdata.plugins.xyz.XYZFormat* format.
train_test_split(test_size[, seed])	Split systems into random train and test subsets.

from_dir
from_file
from_fmt_obj
load_systems_from_file
to_fmt_obj

append(**systems*)

Append systems or MultiSystems to systems.

Parameters

***systems**
[System] The system to append

check_atom_names(*system*)

Make atom_names in all systems equal, prevent inconsistent atom_types.

correction(*hl_sys:* MultiSystems)

Get energy and force correction between self (assumed low-level) and a high-level MultiSystems. The self's coordinates will be kept, but energy and forces will be replaced by the correction between these two systems.

Parameters

hl_sys
[MultiSystems] high-level MultiSystems

Returns

corrected_sys
[MultiSystems] Corrected MultiSystems

Notes

This method will not check whether coordinates and elements of two systems are the same. The user should make sure by itself.

Examples

Get correction between a low-level system and a high-level system:

```
>>> low_level = dpdata.MultiSystems().from_deepmd_hdf5("low_level.hdf5")
>>> high_level = dpdata.MultiSystems().from_deepmd_hdf5("high_level.hdf5")
>>> corr = low_level.correction(high_lebel)
>>> corr.to_deepmd_hdf5("corr.hdf5")
```

from_3dmol(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.3dmol.Py3DMolFormat* format.

from_abacus_lcao_md(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.abacus.AbacusMDFormat* format.

from_abacus_lcao_relax(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.abacus.AbacusRelaxFormat* format.

from_abacus_lcao_scf(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.abacus.AbacusSCFFormat* format.

from_abacus_md(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.abacus.AbacusMDFormat* format.

from_abacus_pw_md(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.abacus.AbacusMDFormat* format.

from_abacus_pw_relax(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.abacus.AbacusRelaxFormat* format.

from_abacus_pw_scf(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.abacus.AbacusSCFFormat* format.

from_abacus_relax(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.abacus.AbacusRelaxFormat* format.

from_abacus_scf(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.abacus.AbacusSCFFormat* format.

from_abacus_stru(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.abacus.AbacusSTRUFormat* format.

from_amber_md(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.amber.AmberMDFormat* format.

from_ase_structure(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.ase.ASEStructureFormat* format.

from_ase_traj(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.ase.ASETrajFormat* format.

from_atomconfig(*file_name*, ***kwargs*)
Read data from *dpdata.plugins.pwmat.PwmatAtomconfigFormat* format.

from_contcar(*file_name*, ***kwargs*)
Read data from *dpdata.plugins.vasp.VASPPoscarFormat* format.

from_cp2k_aimd_output(*file_name*, ***kwargs*)
Read data from *dpdata.plugins.cp2k.CP2KAIMDOutputFormat* format.

from_cp2k_output(*file_name*, ***kwargs*)
Read data from *dpdata.plugins.cp2k.CP2KOutputFormat* format.

from_deepmd(*file_name*, ***kwargs*)
Read data from *dpdata.plugins.deepmd.DeePMDRawFormat* format.

from_deepmd_comp(*file_name*, ***kwargs*)
Read data from *dpdata.plugins.deepmd.DeePMDCompFormat* format.

from_deepmd_hdf5(*file_name*, ***kwargs*)
Read data from *dpdata.plugins.deepmd.DeePMDHDF5Format* format.

from_deepmd_npy(*file_name*, ***kwargs*)
Read data from *dpdata.plugins.deepmd.DeePMDCompFormat* format.

from_deepmd_npy_mixed(*file_name*, ***kwargs*)
Read data from *dpdata.plugins.deepmd.DeePMDMixedFormat* format.

from_deepmd_raw(*file_name*, ***kwargs*)
Read data from *dpdata.plugins.deepmd.DeePMDRawFormat* format.

from_dftbplus(*file_name*, ***kwargs*)
Read data from *dpdata.plugins.dftbplus.DFTBplusFormat* format.

classmethod from_dir(*dir_name*, *file_name*, *fmt='auto'*, *type_map=None*)

from_dump(*file_name*, ***kwargs*)
Read data from *dpdata.plugins.lammps.LAMMPSDumpFormat* format.

from_fhi_aims_md(*file_name*, ***kwargs*)
Read data from *dpdata.plugins.fhi_aims.FhiMDFormat* format.

from_fhi_aims_output(*file_name*, ***kwargs*)
Read data from *dpdata.plugins.fhi_aims.FhiMDFormat* format.

from_fhi_aims_scf(*file_name*, ***kwargs*)
Read data from *dpdata.plugins.fhi_aims.FhiSCFFormat* format.

classmethod from_file(*file_name*, *fmt*, ***kwargs*)

from_finalconfig(*file_name*, ***kwargs*)
Read data from *dpdata.plugins.pwmat.PwmatAtomconfigFormat* format.

from_fmt_obj(*fmtobj*, *directory*, *labeled=True*, ***kwargs*)

from_gaussian_gjf(*file_name*, ***kwargs*)
Read data from *dpdata.plugins.gaussian.GaussiaGJFFormat* format.

from_gaussian_log(*file_name*, ***kwargs*)
Read data from *dpdata.plugins.gaussian.GaussianLogFormat* format.

from_gaussian_md(*file_name*, ***kwargs*)
Read data from *dpdata.plugins.gaussian.GaussianMDFormat* format.

from_gro(*file_name*, ***kwargs*)
Read data from *dpdata.plugins.gromacs.GromacsGroFormat* format.

from_gromacs_gro(*file_name*, ***kwargs*)
Read data from *dpdata.plugins.gromacs.GromacsGroFormat* format.

from_lammps_dump(*file_name*, ***kwargs*)
Read data from *dpdata.plugins.lammps.LAMMPSDumpFormat* format.

from_lammps_lmp(*file_name*, ***kwargs*)
Read data from *dpdata.plugins.lammps.LAMMPSLmpFormat* format.

from_list(*file_name*, ***kwargs*)
Read data from *dpdata.plugins.list.ListFormat* format.

from_lmp(*file_name*, ***kwargs*)
Read data from *dpdata.plugins.lammps.LAMMPSLmpFormat* format.

from_mlmd(*file_name*, ***kwargs*)
Read data from *dpdata.plugins.pwmat.PwmatOutputFormat* format.

from_mol(*file_name*, ***kwargs*)
Read data from *dpdata.plugins.rdkit.MolFormat* format.

from_mol_file(*file_name*, ***kwargs*)
Read data from *dpdata.plugins.rdkit.MolFormat* format.

from_movement(*file_name*, ***kwargs*)
Read data from *dpdata.plugins.pwmat.PwmatOutputFormat* format.

from_n2p2(*file_name*, ***kwargs*)
Read data from *dpdata.plugins.n2p2.N2P2Format* format.

from_openmx_md(*file_name*, ***kwargs*)
Read data from *dpdata.plugins.openmx.OPENMXFormat* format.

from_orca_spout(*file_name*, ***kwargs*)
Read data from *dpdata.plugins.orca.ORCASPOutFormat* format.

from_outcar(*file_name*, ***kwargs*)
Read data from *dpdata.plugins.vasp.VASPOutcarFormat* format.

from_poscar(*file_name*, ***kwargs*)
Read data from *dpdata.plugins.vasp.VASPPoscarFormat* format.

from_psi4_inp(*file_name*, ***kwargs*)
Read data from *dpdata.plugins.psi4.PSI4InputFormat* format.

from_psi4_out(*file_name*, ***kwargs*)
Read data from *dpdata.plugins.psi4.PSI4OutFormat* format.

from_pwmat_atomconfig(*file_name*, ***kwargs*)
Read data from *dpdata.plugins.pwmat.PwmatAtomconfigFormat* format.

from_pwmat_finalconfig(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.pwmat.PwmatAtomconfigFormat* format.

from_pwmat_mlmd(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.pwmat.PwmatOutputFormat* format.

from_pwmat_movement(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.pwmat.PwmatOutputFormat* format.

from_pwmat_output(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.pwmat.PwmatOutputFormat* format.

from_pymatgen_computedstructureentry(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.pymatgen.PyMatgenCSEFormat* format.

from_pymatgen_molecule(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.pymatgen.PyMatgenMoleculeFormat* format.

from_pymatgen_structure(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.pymatgen.PyMatgenStructureFormat* format.

from_qe_cp_traj(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.qe.QECPTrajFormat* format.

from_qe_pw_scf(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.qe.QECPPWSCFFormat* format.

from_quip_gap_xyz(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.xyz.QuipGapXYZFormat* format.

from_quip_gap_xyz_file(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.xyz.QuipGapXYZFormat* format.

from_sdf(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.rdkit.SdfFormat* format.

from_sdf_file(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.rdkit.SdfFormat* format.

from_siesta_aiMD_output(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.siesta.SiestaAIMDOutputFormat* format.

from_siesta_aimd_output(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.siesta.SiestaAIMDOutputFormat* format.

from_siesta_output(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.siesta.SiestaOutputFormat* format.

from_sqm_in(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.amber.SQMINFormat* format.

from_sqm_out(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.amber.SQMOutFormat* format.

from_stru(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.abacus.AbacusSTRUFormat* format.

from_vasp_contcar(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.vasp.VASPPoscarFormat* format.

from_vasp_outcar(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.vasp.VASPOutcarFormat* format.

from_vasp_poscar(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.vasp.VASPPoscarFormat* format.

from_vasp_string(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.vasp.VASPStringFormat* format.

from_vasp_xml(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.vasp.VASPXMLFormat* format.

from_xml(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.vasp.VASPXMLFormat* format.

from_xyz(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.xyz.XYZFormat* format.

get_nframes()

Returns number of frames in all systems.

load_systems_from_file(*file_name=None*, *fmt=None*, ***kwargs*)

minimize(**args: Any*, *minimizer: str | Minimizer*, ***kwargs: Any*) → *MultiSystems*

Minimize geometry by a minimizer.

Parameters

***args**
[iterable] Arguments passing to the minimizer

minimizer
[str or Minimizer] The assigned minimizer

****kwargs**
[dict] Other arguments passing to the minimizer

Returns

MultiSystems
A new labeled MultiSystems.

Examples

Minimize a system using ASE BFGS along with a DP driver:

```
>>> from dpdata.driver import Driver
>>> from ase.optimize import BFGS
>>> driver = Driver.get_driver("dp")("some_model.pb")
>>> some_system.minimize(minimizer="ase", driver=driver, optimizer=BFGS,␣
↪fmax=1e-5)
```

pick_atom_idx(*idx*, *nopbc=None*)

Pick atom index.

Parameters

idx
[int or list or slice] atom index

nopbc
[Boolen (default: None)] If nopbc is True or False, set nopbc

Returns

new_sys: MultiSystems
new system

predict(**args: Any*, *driver='dp'*, ***kwargs: Any*) → *MultiSystems*

Predict energies and forces by a driver.

Parameters

***args**
[iterable] Arguments passing to the driver

driver
[str, default=dp] The assigned driver. For compatibility, default is dp

****kwargs**
[dict] Other arguments passing to the driver

Returns

MultiSystems
A new labeled MultiSystems.

to(*fmt: str*, **args*, ***kwargs*) → *MultiSystems*

Dump systems to the specific format.

Parameters

fmt
[str] format

***args**
[list] arguments

****kwargs**
[dict] keyword arguments

Returns

MultiSystems
self

to_3dmol(**args*, ***kwargs*)

Dump data to *dpdata.plugins.3dmol.Py3DMolFormat* format.

to_abacus_lcao_md(**args*, ***kwargs*)

Dump data to *dpdata.plugins.abacus.AbacusMDFormat* format.

to_abacus_lcao_relax(**args*, ***kwargs*)

Dump data to *dpdata.plugins.abacus.AbacusRelaxFormat* format.

to_abacus_lcao_scf(**args*, ***kwargs*)

Dump data to *dpdata.plugins.abacus.AbacusSCFFormat* format.

to_abacus_md(**args*, ***kwargs*)

Dump data to *dpdata.plugins.abacus.AbacusMDFormat* format.

to_abacus_pw_md(**args*, ***kwargs*)

Dump data to *dpdata.plugins.abacus.AbacusMDFormat* format.

to_abacus_pw_relax(**args*, ***kwargs*)

Dump data to *dpdata.plugins.abacus.AbacusRelaxFormat* format.

to_abacus_pw_scf(**args*, ***kwargs*)

Dump data to *dpdata.plugins.abacus.AbacusSCFFormat* format.

to_abacus_relax(**args*, ***kwargs*)

Dump data to *dpdata.plugins.abacus.AbacusRelaxFormat* format.

to_abacus_scf(**args*, ***kwargs*)

Dump data to *dpdata.plugins.abacus.AbacusSCFFormat* format.

to_abacus_stru(**args*, ***kwargs*)

Dump data to *dpdata.plugins.abacus.AbacusSTRUFormat* format.

to_amber_md(**args*, ***kwargs*)

Dump data to *dpdata.plugins.amber.AmberMDFormat* format.

to_ase_structure(**args*, ***kwargs*)

Dump data to *dpdata.plugins.ase.ASEStructureFormat* format.

to_ase_traj(**args*, ***kwargs*)

Dump data to *dpdata.plugins.ase.ASETrajFormat* format.

to_atomconfig(**args*, ***kwargs*)

Dump data to *dpdata.plugins.pwmat.PwmatAtomconfigFormat* format.

to_contcar(**args*, ***kwargs*)

Dump data to *dpdata.plugins.vasp.VASPPoscarFormat* format.

to_cp2k_aimd_output(**args*, ***kwargs*)

Dump data to *dpdata.plugins.cp2k.CP2KAIMDOutputFormat* format.

to_cp2k_output(**args*, ***kwargs*)

Dump data to *dpdata.plugins.cp2k.CP2KOutputFormat* format.

to_deepmd(**args*, ***kwargs*)

Dump data to *dpdata.plugins.deepmd.DeePMDRawFormat* format.

to_deepmd_comp(**args*, ***kwargs*)

Dump data to *dpdata.plugins.deepmd.DeePMDCompFormat* format.

to_deepmd_hdf5(**args*, ***kwargs*)

Dump data to *dpdata.plugins.deepmd.DeePMDHDF5Format* format.

to_deepmd_npy(**args*, ***kwargs*)

Dump data to *dpdata.plugins.deepmd.DeePMDCompFormat* format.

to_deepmd_npy_mixed(**args*, ***kwargs*)

Dump data to *dpdata.plugins.deepmd.DeePMDMixedFormat* format.

to_deepmd_raw(**args*, ***kwargs*)

Dump data to *dpdata.plugins.deepmd.DeePMDRawFormat* format.

to_dftbplus(**args*, ***kwargs*)

Dump data to *dpdata.plugins.dftbplus.DFTBplusFormat* format.

to_dump(**args*, ***kwargs*)

Dump data to *dpdata.plugins.lammps.LAMMPSDumpFormat* format.

to_fhi_aims_md(**args*, ***kwargs*)

Dump data to *dpdata.plugins.fhi_aims.FhiMDFormat* format.

to_fhi_aims_output(**args*, ***kwargs*)

Dump data to *dpdata.plugins.fhi_aims.FhiMDFormat* format.

to_fhi_aims_scf(**args*, ***kwargs*)

Dump data to *dpdata.plugins.fhi_aims.FhiSCFFormat* format.

to_finalconfig(**args*, ***kwargs*)

Dump data to *dpdata.plugins.pwmat.PwmatAtomconfigFormat* format.

to_fmt_obj(*fmtobj*, *directory*, **args*, ***kwargs*)

to_gaussian_gjf(**args*, ***kwargs*)

Dump data to *dpdata.plugins.gaussian.GaussiaGJFFormat* format.

to_gaussian_log(**args*, ***kwargs*)

Dump data to *dpdata.plugins.gaussian.GaussianLogFormat* format.

to_gaussian_md(**args*, ***kwargs*)

Dump data to *dpdata.plugins.gaussian.GaussianMDFormat* format.

to_gro(**args*, ***kwargs*)

Dump data to *dpdata.plugins.gromacs.GromacsGroFormat* format.

to_gromacs_gro(**args*, ***kwargs*)

Dump data to *dpdata.plugins.gromacs.GromacsGroFormat* format.

to_lammps_dump(**args*, ***kwargs*)

Dump data to *dpdata.plugins.lammps.LAMMPSDumpFormat* format.

to_lammps_lmp(**args*, ***kwargs*)

Dump data to *dpdata.plugins.lammps.LAMMPSLmpFormat* format.

to_list(**args*, ***kwargs*)

Dump data to *dpdata.plugins.list.ListFormat* format.

to_lmp(**args*, ***kwargs*)

Dump data to *dpdata.plugins.lammps.LAMMPSLmpFormat* format.

to_mlmd(**args*, ***kwargs*)

Dump data to *dpdata.plugins.pwmat.PwmatOutputFormat* format.

to_mol(**args*, ***kwargs*)

Dump data to *dpdata.plugins.rdkit.MolFormat* format.

to_mol_file(**args*, ***kwargs*)

Dump data to *dpdata.plugins.rdkit.MolFormat* format.

to_movement(**args*, ***kwargs*)

Dump data to *dpdata.plugins.pwmat.PwmatOutputFormat* format.

to_n2p2(**args*, ***kwargs*)

Dump data to *dpdata.plugins.n2p2.N2P2Format* format.

to_openmx_md(**args*, ***kwargs*)

Dump data to *dpdata.plugins.openmx.OPENMXFormat* format.

to_orca_spout(**args*, ***kwargs*)

Dump data to *dpdata.plugins.orca.ORCASPOutFormat* format.

to_outcar(**args*, ***kwargs*)

Dump data to *dpdata.plugins.vasp.VASPOutcarFormat* format.

to_poscar(**args*, ***kwargs*)

Dump data to *dpdata.plugins.vasp.VASPPoscarFormat* format.

to_psi4_inp(**args*, ***kwargs*)

Dump data to *dpdata.plugins.psi4.PSI4InputFormat* format.

to_psi4_out(**args*, ***kwargs*)

Dump data to *dpdata.plugins.psi4.PSI4OutFormat* format.

to_pwmat_atomconfig(**args*, ***kwargs*)

Dump data to *dpdata.plugins.pwmat.PwmatAtomconfigFormat* format.

to_pwmat_finalconfig(**args*, ***kwargs*)

Dump data to *dpdata.plugins.pwmat.PwmatAtomconfigFormat* format.

to_pwmat_mlmd(**args*, ***kwargs*)

Dump data to *dpdata.plugins.pwmat.PwmatOutputFormat* format.

to_pwmat_movement(**args*, ***kwargs*)

Dump data to *dpdata.plugins.pwmat.PwmatOutputFormat* format.

to_pwmat_output(**args*, ***kwargs*)

Dump data to *dpdata.plugins.pwmat.PwmatOutputFormat* format.

to_pymatgen_ComputedStructureEntry(**args*, ***kwargs*)

Dump data to *dpdata.plugins.pymatgen.PyMatgenCSEFormat* format.

to_pymatgen_computedstructureentry(**args*, ***kwargs*)

Dump data to *dpdata.plugins.pymatgen.PyMatgenCSEFormat* format.

to_pymatgen_molecule(**args*, ***kwargs*)

Dump data to *dpdata.plugins.pymatgen.PyMatgenMoleculeFormat* format.

to_pymatgen_structure(**args*, ***kwargs*)

Dump data to *dpdata.plugins.pymatgen.PyMatgenStructureFormat* format.

to_qe_cp_traj(**args*, ***kwargs*)

Dump data to *dpdata.plugins.qe.QECPTrajFormat* format.

to_qe_pw_scf(**args*, ***kwargs*)

Dump data to *dpdata.plugins.qe.QECPPWSCFFormat* format.

to_quip_gap_xyz(**args*, ***kwargs*)

Dump data to *dpdata.plugins.xyz.QuipGapXYZFormat* format.

to_quip_gap_xyz_file(**args*, ***kwargs*)

Dump data to *dpdata.plugins.xyz.QuipGapXYZFormat* format.

to_sdf(**args*, ***kwargs*)

Dump data to *dpdata.plugins.rdkit.SdfFormat* format.

to_sdf_file(**args*, ***kwargs*)

Dump data to *dpdata.plugins.rdkit.SdfFormat* format.

to_siesta_aimd_output(**args*, ***kwargs*)

Dump data to *dpdata.plugins.siesta.SiestaAIMDOutputFormat* format.

to_siesta_output(**args*, ***kwargs*)

Dump data to *dpdata.plugins.siesta.SiestaOutputFormat* format.

to_sqm_in(**args*, ***kwargs*)

Dump data to *dpdata.plugins.amber.SQMINFormat* format.

to_sqm_out(**args*, ***kwargs*)

Dump data to *dpdata.plugins.amber.SQMOutFormat* format.

to_stru(**args*, ***kwargs*)

Dump data to *dpdata.plugins.abacus.AbacusSTRUFormat* format.

to_vasp_contcar(**args*, ***kwargs*)

Dump data to *dpdata.plugins.vasp.VASPPoscarFormat* format.

to_vasp_outcar(**args*, ***kwargs*)

Dump data to *dpdata.plugins.vasp.VASPOutcarFormat* format.

to_vasp_poscar(**args*, ***kwargs*)

Dump data to *dpdata.plugins.vasp.VASPPoscarFormat* format.

to_vasp_string(**args*, ***kwargs*)

Dump data to *dpdata.plugins.vasp.VASPStringFormat* format.

to_vasp_xml(**args*, ***kwargs*)

Dump data to *dpdata.plugins.vasp.VASPXMLFormat* format.

to_xml(**args*, ***kwargs*)

Dump data to *dpdata.plugins.vasp.VASPXMLFormat* format.

to_xyz(**args*, ***kwargs*)

Dump data to *dpdata.plugins.xyz.XYZFormat* format.

train_test_split(*test_size: float | int*, *seed: int | None = None*) → Tuple[*MultiSystems*, *MultiSystems*, Dict[str, ndarray]]

Split systems into random train and test subsets.

Parameters

test_size
[float or int] If float, should be between 0.0 and 1.0 and represent the proportion of the dataset to include in the test split. If int, represents the absolute number of test samples.

seed
[int, default=None] Random seed

Returns

MultiSystems
The training set

MultiSystems
The testing set

Dict[str, np.ndarray]
The bool array of training and testing sets for each system. False for training set and True for testing set.

class dpdata.**System**(*file_name=None*, *fmt='auto'*, *type_map=None*, *begin=0*, *step=1*, *data=None*, *convergence_check=True*, ***kwargs*)

Bases: `MSONable`

The data System.

A data System (a concept used by deepmd-kit) contains frames (e.g. produced by an MD simulation) that has the same number of atoms of the same type. The order of the atoms should be consistent among the frames in one System.

For example, a water system named *d_example* has two molecules. The properties can be accessed by

- *d_example['atom_numbs']* : [2, 4]
- *d_example['atom_names']* : ['O', 'H']
- *d_example['atom_types']* : [0, 1, 1, 0, 1, 1]
- *d_example['orig']* : [0, 0, 0]
- *d_example['cells']* : a numpy array of size nframes x 3 x 3
- *d_example['coords']* : a numpy array of size nframes x natoms x 3

It is noted that

- The order of frames stored in *'atom_types'*, *'cells'* and *'coords'* should be consistent.
- The order of atoms in **all** frames of *'atom_types'* and *'coords'* should be consistent.

Restrictions:

- *d_example['orig']* is always [0, 0, 0]
- *d_example['cells'][ii]* is always lower triangular (lammps cell tensor convention)

Attributes

DTYPES
[tuple[DataType]] data types of this class

Methods

add_atom_names(atom_names)	Add atom_names that do not exist.
append(system)	Append a system to this system.
apply_pbc()	Append periodic boundary condition.
as_dict()	Returns data dict of System instance.
check_data()	Check if data is correct.
check_type_map(type_map)	Assign atom_names to type_map if type_map is given and different from atom_names.

continues on next page

Table 4 – continued from previous page

convert_to_mixed_type([type_map])	Convert the data dict to mixed type format structure, in order to append systems with different formula but the same number of atoms.
copy()	Returns a copy of the system.
dump(filename[, indent])	Dump .json or .yaml file.
extend(systems)	Extend a system list to this system.
from_3dmol(file_name, **kwargs)	Read data from dpdata.plugins.3dmol.Py3DMolFormat format.
from_abacus_lcao_md(file_name, **kwargs)	Read data from dpdata.plugins.abacus.AbacusMDFormat format.
from_abacus_lcao_relax(file_name, **kwargs)	Read data from dpdata.plugins.abacus.AbacusRelaxFormat format.
from_abacus_lcao_scf(file_name, **kwargs)	Read data from dpdata.plugins.abacus.AbacusSCFFormat format.
from_abacus_md(file_name, **kwargs)	Read data from dpdata.plugins.abacus.AbacusMDFormat format.
from_abacus_pw_md(file_name, **kwargs)	Read data from dpdata.plugins.abacus.AbacusMDFormat format.
from_abacus_pw_relax(file_name, **kwargs)	Read data from dpdata.plugins.abacus.AbacusRelaxFormat format.
from_abacus_pw_scf(file_name, **kwargs)	Read data from dpdata.plugins.abacus.AbacusSCFFormat format.
from_abacus_relax(file_name, **kwargs)	Read data from dpdata.plugins.abacus.AbacusRelaxFormat format.
from_abacus_scf(file_name, **kwargs)	Read data from dpdata.plugins.abacus.AbacusSCFFormat format.
from_abacus_stru(file_name, **kwargs)	Read data from dpdata.plugins.abacus.AbacusSTRUFormat format.
from_amber_md(file_name, **kwargs)	Read data from dpdata.plugins.amber.AmberMDFormat format.
from_ase_structure(file_name, **kwargs)	Read data from dpdata.plugins.ase.ASEStructureFormat format.
from_ase_traj(file_name, **kwargs)	Read data from dpdata.plugins.ase.ASETrajFormat format.
from_atomconfig(file_name, **kwargs)	Read data from dpdata.plugins.pwmat.PwmatAtomconfigFormat format.
from_contcar(file_name, **kwargs)	Read data from dpdata.plugins.vasp.VASPPoscarFormat format.
from_cp2k_aimd_output(file_name, **kwargs)	Read data from dpdata.plugins.cp2k.CP2KAIMDOutputFormat format.
from_cp2k_output(file_name, **kwargs)	Read data from dpdata.plugins.cp2k.CP2KOutputFormat format.
from_deepmd(file_name, **kwargs)	Read data from dpdata.plugins.deepmd.DeePMDRawFormat format.
from_deepmd_comp(file_name, **kwargs)	Read data from dpdata.plugins.deepmd.DeePMDCompFormat format.
from_deepmd_hdf5(file_name, **kwargs)	Read data from dpdata.plugins.deepmd.DeePMDHDF5Format format.
from_deepmd_npy(file_name, **kwargs)	Read data from dpdata.plugins.deepmd.DeePMDCompFormat format.
from_deepmd_npy_mixed(file_name, **kwargs)	Read data from dpdata.plugins.deepmd.DeePMDMixedFormat format.

continues on next page

Table 4 – continued from previous page

from_deepmd_raw(file_name, **kwargs)	Read data from *dpdata.plugins.deepmd.DeePMDRawFormat* format.
from_dftbplus(file_name, **kwargs)	Read data from *dpdata.plugins.dftbplus.DFTBplusFormat* format.
from_dict(d)	**param d** Dict representation.
from_dump(file_name, **kwargs)	Read data from *dpdata.plugins.lammps.LAMMPSDumpFormat* format.
from_fhi_aims_md(file_name, **kwargs)	Read data from *dpdata.plugins.fhi_aims.FhiMDFormat* format.
from_fhi_aims_output(file_name, **kwargs)	Read data from *dpdata.plugins.fhi_aims.FhiMDFormat* format.
from_fhi_aims_scf(file_name, **kwargs)	Read data from *dpdata.plugins.fhi_aims.FhiSCFFormat* format.
from_finalconfig(file_name, **kwargs)	Read data from *dpdata.plugins.pwmat.PwmatAtomconfigFormat* format.
from_gaussian_gjf(file_name, **kwargs)	Read data from *dpdata.plugins.gaussian.GaussiaGJFFormat* format.
from_gaussian_log(file_name, **kwargs)	Read data from *dpdata.plugins.gaussian.GaussianLogFormat* format.
from_gaussian_md(file_name, **kwargs)	Read data from *dpdata.plugins.gaussian.GaussianMDFormat* format.
from_gro(file_name, **kwargs)	Read data from *dpdata.plugins.gromacs.GromacsGroFormat* format.
from_gromacs_gro(file_name, **kwargs)	Read data from *dpdata.plugins.gromacs.GromacsGroFormat* format.
from_lammps_dump(file_name, **kwargs)	Read data from *dpdata.plugins.lammps.LAMMPSDumpFormat* format.
from_lammps_lmp(file_name, **kwargs)	Read data from *dpdata.plugins.lammps.LAMMPSLmpFormat* format.
from_list(file_name, **kwargs)	Read data from *dpdata.plugins.list.ListFormat* format.
from_lmp(file_name, **kwargs)	Read data from *dpdata.plugins.lammps.LAMMPSLmpFormat* format.
from_mlmd(file_name, **kwargs)	Read data from *dpdata.plugins.pwmat.PwmatOutputFormat* format.
from_mol(file_name, **kwargs)	Read data from *dpdata.plugins.rdkit.MolFormat* format.
from_mol_file(file_name, **kwargs)	Read data from *dpdata.plugins.rdkit.MolFormat* format.
from_movement(file_name, **kwargs)	Read data from *dpdata.plugins.pwmat.PwmatOutputFormat* format.
from_n2p2(file_name, **kwargs)	Read data from *dpdata.plugins.n2p2.N2P2Format* format.
from_openmx_md(file_name, **kwargs)	Read data from *dpdata.plugins.openmx.OPENMXFormat* format.
from_orca_spout(file_name, **kwargs)	Read data from *dpdata.plugins.orca.ORCASPOutFormat* format.
from_outcar(file_name, **kwargs)	Read data from *dpdata.plugins.vasp.VASPOutcarFormat* format.

continues on next page

Table 4 – continued from previous page

from_poscar(file_name, **kwargs)	Read data from *dpdata.plugins.vasp.VASPPoscarFormat* format.
from_psi4_inp(file_name, **kwargs)	Read data from *dpdata.plugins.psi4.PSI4InputFormat* format.
from_psi4_out(file_name, **kwargs)	Read data from *dpdata.plugins.psi4.PSI4OutFormat* format.
from_pwmat_atomconfig(file_name, **kwargs)	Read data from *dpdata.plugins.pwmat.PwmatAtomconfigFormat* format.
from_pwmat_finalconfig(file_name, **kwargs)	Read data from *dpdata.plugins.pwmat.PwmatAtomconfigFormat* format.
from_pwmat_mlmd(file_name, **kwargs)	Read data from *dpdata.plugins.pwmat.PwmatOutputFormat* format.
from_pwmat_movement(file_name, **kwargs)	Read data from *dpdata.plugins.pwmat.PwmatOutputFormat* format.
from_pwmat_output(file_name, **kwargs)	Read data from *dpdata.plugins.pwmat.PwmatOutputFormat* format.
from_pymatgen_computedstructureentry(...)	Read data from *dpdata.plugins.pymatgen.PyMatgenCSEFormat* format.
from_pymatgen_molecule(file_name, **kwargs)	Read data from *dpdata.plugins.pymatgen.PyMatgenMoleculeFormat* format.
from_pymatgen_structure(file_name, **kwargs)	Read data from *dpdata.plugins.pymatgen.PyMatgenStructureFormat* format.
from_qe_cp_traj(file_name, **kwargs)	Read data from *dpdata.plugins.qe.QECPTrajFormat* format.
from_qe_pw_scf(file_name, **kwargs)	Read data from *dpdata.plugins.qe.QECPPWSCFFormat* format.
from_quip_gap_xyz(file_name, **kwargs)	Read data from *dpdata.plugins.xyz.QuipGapXYZFormat* format.
from_quip_gap_xyz_file(file_name, **kwargs)	Read data from *dpdata.plugins.xyz.QuipGapXYZFormat* format.
from_sdf(file_name, **kwargs)	Read data from *dpdata.plugins.rdkit.SdfFormat* format.
from_sdf_file(file_name, **kwargs)	Read data from *dpdata.plugins.rdkit.SdfFormat* format.
from_siesta_aiMD_output(file_name, **kwargs)	Read data from *dpdata.plugins.siesta.SiestaAIMDOutputFormat* format.
from_siesta_aimd_output(file_name, **kwargs)	Read data from *dpdata.plugins.siesta.SiestaAIMDOutputFormat* format.
from_siesta_output(file_name, **kwargs)	Read data from *dpdata.plugins.siesta.SiestaOutputFormat* format.
from_sqm_in(file_name, **kwargs)	Read data from *dpdata.plugins.amber.SQMINFormat* format.
from_sqm_out(file_name, **kwargs)	Read data from *dpdata.plugins.amber.SQMOutFormat* format.
from_stru(file_name, **kwargs)	Read data from *dpdata.plugins.abacus.AbacusSTRUFormat* format.
from_vasp_contcar(file_name, **kwargs)	Read data from *dpdata.plugins.vasp.VASPPoscarFormat* format.
from_vasp_outcar(file_name, **kwargs)	Read data from *dpdata.plugins.vasp.VASPOutcarFormat* format.
from_vasp_poscar(file_name, **kwargs)	Read data from *dpdata.plugins.vasp.VASPPoscarFormat* format.

continues on next page

Table 4 – continued from previous page

from_vasp_string(file_name, **kwargs)	Read data from *dpdata.plugins.vasp.VASPStringFormat* format.
from_vasp_xml(file_name, **kwargs)	Read data from *dpdata.plugins.vasp.VASPXMLFormat* format.
from_xml(file_name, **kwargs)	Read data from *dpdata.plugins.vasp.VASPXMLFormat* format.
from_xyz(file_name, **kwargs)	Read data from *dpdata.plugins.xyz.XYZFormat* format.
get_atom_names()	Returns name of atoms.
get_atom_numbs()	Returns number of atoms.
get_atom_types()	Returns type of atoms.
get_natoms()	Returns total number of atoms in the system.
get_nframes()	Returns number of frames in the system.
get_ntypes()	Returns total number of atom types in the system.
load(filename)	Rebuild System obj.
map_atom_types([type_map])	Map the atom types of the system.
minimize(*args, minimizer, **kwargs)	Minimize the geometry.
perturb(pert_num, cell_pert_fraction, ...[, ...])	Perturb each frame in the system randomly.
pick_atom_idx(idx[, nopbc])	Pick atom index.
pick_by_amber_mask(param, maskstr[, ...])	Pick atoms by amber mask.
predict(*args[, driver])	Predict energies and forces by a driver.
register_data_type(*data_type)	Register data type.
remove_atom_names(atom_names)	Remove atom names and all such atoms.
remove_pbc([protect_layer])	This method does NOT delete the definition of the cells, it (1) revises the cell to a cubic cell and ensures that the cell boundary to any atom in the system is no less than *protect_layer* (2) translates the system such that the center-of-geometry of the system locates at the center of the cell.
replicate(ncopy)	Replicate the each frame in the system in 3 dimensions.
shuffle()	Shuffle frames randomly.
sort_atom_names([type_map])	Sort atom_names of the system and reorder atom_numbs and atom_types accoarding to atom_names.
sort_atom_types()	Sort atom types.
sub_system(f_idx)	Construct a subsystem from the system.
to(fmt, *args, **kwargs)	Dump systems to the specific format.
to_3dmol(*args, **kwargs)	Dump data to *dpdata.plugins.3dmol.Py3DMolFormat* format.
to_abacus_lcao_md(*args, **kwargs)	Dump data to *dpdata.plugins.abacus.AbacusMDFormat* format.
to_abacus_lcao_relax(*args, **kwargs)	Dump data to *dpdata.plugins.abacus.AbacusRelaxFormat* format.
to_abacus_lcao_scf(*args, **kwargs)	Dump data to *dpdata.plugins.abacus.AbacusSCFFormat* format.
to_abacus_md(*args, **kwargs)	Dump data to *dpdata.plugins.abacus.AbacusMDFormat* format.
to_abacus_pw_md(*args, **kwargs)	Dump data to *dpdata.plugins.abacus.AbacusMDFormat* format.
to_abacus_pw_relax(*args, **kwargs)	Dump data to *dpdata.plugins.abacus.AbacusRelaxFormat* format.

continues on next page

Table 4 – continued from previous page

to_abacus_pw_scf(*args, **kwargs)	Dump data to *dpdata.plugins.abacus.AbacusSCFFormat* format.
to_abacus_relax(*args, **kwargs)	Dump data to *dpdata.plugins.abacus.AbacusRelaxFormat* format.
to_abacus_scf(*args, **kwargs)	Dump data to *dpdata.plugins.abacus.AbacusSCFFormat* format.
to_abacus_stru(*args, **kwargs)	Dump data to *dpdata.plugins.abacus.AbacusSTRUFormat* format.
to_amber_md(*args, **kwargs)	Dump data to *dpdata.plugins.amber.AmberMDFormat* format.
to_ase_structure(*args, **kwargs)	Dump data to *dpdata.plugins.ase.ASEStructureFormat* format.
to_ase_traj(*args, **kwargs)	Dump data to *dpdata.plugins.ase.ASETrajFormat* format.
to_atomconfig(*args, **kwargs)	Dump data to *dpdata.plugins.pwmat.PwmatAtomconfigFormat* format.
to_contcar(*args, **kwargs)	Dump data to *dpdata.plugins.vasp.VASPPoscarFormat* format.
to_cp2k_aimd_output(*args, **kwargs)	Dump data to *dpdata.plugins.cp2k.CP2KAIMDOutputFormat* format.
to_cp2k_output(*args, **kwargs)	Dump data to *dpdata.plugins.cp2k.CP2KOutputFormat* format.
to_deepmd(*args, **kwargs)	Dump data to *dpdata.plugins.deepmd.DeePMDRawFormat* format.
to_deepmd_comp(*args, **kwargs)	Dump data to *dpdata.plugins.deepmd.DeePMDCompFormat* format.
to_deepmd_hdf5(*args, **kwargs)	Dump data to *dpdata.plugins.deepmd.DeePMDHDF5Format* format.
to_deepmd_npy(*args, **kwargs)	Dump data to *dpdata.plugins.deepmd.DeePMDCompFormat* format.
to_deepmd_npy_mixed(*args, **kwargs)	Dump data to *dpdata.plugins.deepmd.DeePMDMixedFormat* format.
to_deepmd_raw(*args, **kwargs)	Dump data to *dpdata.plugins.deepmd.DeePMDRawFormat* format.
to_dftbplus(*args, **kwargs)	Dump data to *dpdata.plugins.dftbplus.DFTBplusFormat* format.
to_dump(*args, **kwargs)	Dump data to *dpdata.plugins.lammps.LAMMPSDumpFormat* format.
to_fhi_aims_md(*args, **kwargs)	Dump data to *dpdata.plugins.fhi_aims.FhiMDFormat* format.
to_fhi_aims_output(*args, **kwargs)	Dump data to *dpdata.plugins.fhi_aims.FhiMDFormat* format.
to_fhi_aims_scf(*args, **kwargs)	Dump data to *dpdata.plugins.fhi_aims.FhiSCFFormat* format.
to_finalconfig(*args, **kwargs)	Dump data to *dpdata.plugins.pwmat.PwmatAtomconfigFormat* format.
to_gaussian_gjf(*args, **kwargs)	Dump data to *dpdata.plugins.gaussian.GaussiaGJFFormat* format.
to_gaussian_log(*args, **kwargs)	Dump data to *dpdata.plugins.gaussian.GaussianLogFormat* format.
to_gaussian_md(*args, **kwargs)	Dump data to *dpdata.plugins.gaussian.GaussianMDFormat* format.

continues on next page

Table 4 – continued from previous page

to_gro(*args, **kwargs)	Dump data to *dpdata.plugins.gromacs.GromacsGroFormat* format.
to_gromacs_gro(*args, **kwargs)	Dump data to *dpdata.plugins.gromacs.GromacsGroFormat* format.
to_json()	Returns a json string representation of the MSONable object.
to_lammps_dump(*args, **kwargs)	Dump data to *dpdata.plugins.lammps.LAMMPSDumpFormat* format.
to_lammps_lmp(*args, **kwargs)	Dump data to *dpdata.plugins.lammps.LAMMPSLmpFormat* format.
to_list(*args, **kwargs)	Dump data to *dpdata.plugins.list.ListFormat* format.
to_lmp(*args, **kwargs)	Dump data to *dpdata.plugins.lammps.LAMMPSLmpFormat* format.
to_mlmd(*args, **kwargs)	Dump data to *dpdata.plugins.pwmat.PwmatOutputFormat* format.
to_mol(*args, **kwargs)	Dump data to *dpdata.plugins.rdkit.MolFormat* format.
to_mol_file(*args, **kwargs)	Dump data to *dpdata.plugins.rdkit.MolFormat* format.
to_movement(*args, **kwargs)	Dump data to *dpdata.plugins.pwmat.PwmatOutputFormat* format.
to_n2p2(*args, **kwargs)	Dump data to *dpdata.plugins.n2p2.N2P2Format* format.
to_openmx_md(*args, **kwargs)	Dump data to *dpdata.plugins.openmx.OPENMXFormat* format.
to_orca_spout(*args, **kwargs)	Dump data to *dpdata.plugins.orca.ORCASPOutFormat* format.
to_outcar(*args, **kwargs)	Dump data to *dpdata.plugins.vasp.VASPOutcarFormat* format.
to_poscar(*args, **kwargs)	Dump data to *dpdata.plugins.vasp.VASPPoscarFormat* format.
to_psi4_inp(*args, **kwargs)	Dump data to *dpdata.plugins.psi4.PSI4InputFormat* format.
to_psi4_out(*args, **kwargs)	Dump data to *dpdata.plugins.psi4.PSI4OutFormat* format.
to_pwmat_atomconfig(*args, **kwargs)	Dump data to *dpdata.plugins.pwmat.PwmatAtomconfigFormat* format.
to_pwmat_finalconfig(*args, **kwargs)	Dump data to *dpdata.plugins.pwmat.PwmatAtomconfigFormat* format.
to_pwmat_mlmd(*args, **kwargs)	Dump data to *dpdata.plugins.pwmat.PwmatOutputFormat* format.
to_pwmat_movement(*args, **kwargs)	Dump data to *dpdata.plugins.pwmat.PwmatOutputFormat* format.
to_pwmat_output(*args, **kwargs)	Dump data to *dpdata.plugins.pwmat.PwmatOutputFormat* format.
to_pymatgen_ComputedStructureEntry(*args, ...)	Dump data to *dpdata.plugins.pymatgen.PyMatgenCSEFormat* format.
to_pymatgen_computedstructureentry(*args, ...)	Dump data to *dpdata.plugins.pymatgen.PyMatgenCSEFormat* format.
to_pymatgen_molecule(*args, **kwargs)	Dump data to *dpdata.plugins.pymatgen.PyMatgenMoleculeFormat* format.

continues on next page

Table 4 – continued from previous page

to_pymatgen_structure(*args, **kwargs)	Dump data to dpdata.plugins.pymatgen.PyMatgenStructureFormat format.
to_qe_cp_traj(*args, **kwargs)	Dump data to dpdata.plugins.qe.QECPTrajFormat format.
to_qe_pw_scf(*args, **kwargs)	Dump data to dpdata.plugins.qe.QECPPWSCFFormat format.
to_quip_gap_xyz(*args, **kwargs)	Dump data to dpdata.plugins.xyz.QuipGapXYZFormat format.
to_quip_gap_xyz_file(*args, **kwargs)	Dump data to dpdata.plugins.xyz.QuipGapXYZFormat format.
to_sdf(*args, **kwargs)	Dump data to dpdata.plugins.rdkit.SdfFormat format.
to_sdf_file(*args, **kwargs)	Dump data to dpdata.plugins.rdkit.SdfFormat format.
to_siesta_aimd_output(*args, **kwargs)	Dump data to dpdata.plugins.siesta.SiestaAIMDOutputFormat format.
to_siesta_output(*args, **kwargs)	Dump data to dpdata.plugins.siesta.SiestaOutputFormat format.
to_sqm_in(*args, **kwargs)	Dump data to dpdata.plugins.amber.SQMINFormat format.
to_sqm_out(*args, **kwargs)	Dump data to dpdata.plugins.amber.SQMOutFormat format.
to_stru(*args, **kwargs)	Dump data to dpdata.plugins.abacus.AbacusSTRUFormat format.
to_vasp_contcar(*args, **kwargs)	Dump data to dpdata.plugins.vasp.VASPPoscarFormat format.
to_vasp_outcar(*args, **kwargs)	Dump data to dpdata.plugins.vasp.VASPOutcarFormat format.
to_vasp_poscar(*args, **kwargs)	Dump data to dpdata.plugins.vasp.VASPPoscarFormat format.
to_vasp_string(*args, **kwargs)	Dump data to dpdata.plugins.vasp.VASPStringFormat format.
to_vasp_xml(*args, **kwargs)	Dump data to dpdata.plugins.vasp.VASPXMLFormat format.
to_xml(*args, **kwargs)	Dump data to dpdata.plugins.vasp.VASPXMLFormat format.
to_xyz(*args, **kwargs)	Dump data to dpdata.plugins.xyz.XYZFormat format.
unsafe_hash()	Returns an hash of the current object.
validate_monty_v1(_MSONable__input_value)	Pydantic validator with correct signature for pydantic v1.x
validate_monty_v2(_MSONable__input_value, _)	Pydantic validator with correct signature for pydantic v2.x

affine_map
apply_type_map
from_fmt
from_fmt_obj
replace
rot_frame_lower_triangular
rot_lower_triangular
to_fmt_obj

DTYPES = (<dpdata.data_type.DataType object>, <dpdata.data_type.DataType object>, <dpdata.data_type.DataType object>, <dpdata.data_type.DataType object>, <dpdata.data_type.DataType object>, <dpdata.data_type.DataType object>, <dpdata.data_type.DataType object>, <dpdata.data_type.DataType object>, <dpdata.data_type.DataType object>)

add_atom_names(*atom_names*)

Add atom_names that do not exist.

affine_map(*trans*, *f_idx=0*)

append(*system*)

Append a system to this system.

Parameters

system
[System] The system to append

apply_pbc()

Append periodic boundary condition.

apply_type_map(*type_map*)

as_dict()

Returns data dict of System instance.

check_data()

Check if data is correct.

Raises

DataError
if data is not correct

check_type_map(*type_map*)

Assign atom_names to type_map if type_map is given and different from atom_names.

Parameters

type_map
[list] type_map

convert_to_mixed_type(*type_map=None*)

Convert the data dict to mixed type format structure, in order to append systems with different formula but the same number of atoms. Change the 'atom_names' to one placeholder type 'MIXED_TOKEN' and add 'real_atom_types' to store the real type vectors according to the given type_map.

Parameters

type_map
[list] type_map

copy()

Returns a copy of the system.

dump(*filename*, *indent=4*)

Dump .json or .yaml file.

extend(*systems*)

Extend a system list to this system.

Parameters

systems
[[System1, System2, System3]] The list to extend

property formula

Return the formula of this system, like C3H5O2.

property formula_hash: str

Return the hash of the formula of this system.

from_3dmol(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.3dmol.Py3DMolFormat* format.

from_abacus_lcao_md(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.abacus.AbacusMDFormat* format.

from_abacus_lcao_relax(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.abacus.AbacusRelaxFormat* format.

from_abacus_lcao_scf(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.abacus.AbacusSCFFormat* format.

from_abacus_md(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.abacus.AbacusMDFormat* format.

from_abacus_pw_md(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.abacus.AbacusMDFormat* format.

from_abacus_pw_relax(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.abacus.AbacusRelaxFormat* format.

from_abacus_pw_scf(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.abacus.AbacusSCFFormat* format.

from_abacus_relax(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.abacus.AbacusRelaxFormat* format.

from_abacus_scf(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.abacus.AbacusSCFFormat* format.

from_abacus_stru(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.abacus.AbacusSTRUFormat* format.

from_amber_md(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.amber.AmberMDFormat* format.

from_ase_structure(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.ase.ASEStructureFormat* format.

from_ase_traj(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.ase.ASETrajFormat* format.

from_atomconfig(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.pwmat.PwmatAtomconfigFormat* format.

from_contcar(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.vasp.VASPPoscarFormat* format.

from_cp2k_aimd_output(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.cp2k.CP2KAIMDOutputFormat* format.

from_cp2k_output(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.cp2k.CP2KOutputFormat* format.

from_deepmd(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.deepmd.DeePMDRawFormat* format.

from_deepmd_comp(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.deepmd.DeePMDCompFormat* format.

from_deepmd_hdf5(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.deepmd.DeePMDHDF5Format* format.

from_deepmd_npy(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.deepmd.DeePMDCompFormat* format.

from_deepmd_npy_mixed(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.deepmd.DeePMDMixedFormat* format.

from_deepmd_raw(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.deepmd.DeePMDRawFormat* format.

from_dftbplus(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.dftbplus.DFTBplusFormat* format.

from_dump(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.lammps.LAMMPSDumpFormat* format.

from_fhi_aims_md(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.fhi_aims.FhiMDFormat* format.

from_fhi_aims_output(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.fhi_aims.FhiMDFormat* format.

from_fhi_aims_scf(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.fhi_aims.FhiSCFFormat* format.

from_finalconfig(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.pwmat.PwmatAtomconfigFormat* format.

from_fmt(*file_name*, *fmt='auto'*, ***kwargs*)

from_fmt_obj(*fmtobj*, *file_name*, ***kwargs*)

from_gaussian_gjf(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.gaussian.GaussiaGJFFormat* format.

from_gaussian_log(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.gaussian.GaussianLogFormat* format.

from_gaussian_md(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.gaussian.GaussianMDFormat* format.

from_gro(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.gromacs.GromacsGroFormat* format.

from_gromacs_gro(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.gromacs.GromacsGroFormat* format.

from_lammps_dump(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.lammps.LAMMPSDumpFormat* format.

from_lammps_lmp(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.lammps.LAMMPSLmpFormat* format.

from_list(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.list.ListFormat* format.

from_lmp(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.lammps.LAMMPSLmpFormat* format.

from_mlmd(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.pwmat.PwmatOutputFormat* format.

from_mol(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.rdkit.MolFormat* format.

from_mol_file(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.rdkit.MolFormat* format.

from_movement(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.pwmat.PwmatOutputFormat* format.

from_n2p2(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.n2p2.N2P2Format* format.

from_openmx_md(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.openmx.OPENMXFormat* format.

from_orca_spout(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.orca.ORCASPOutFormat* format.

from_outcar(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.vasp.VASPOutcarFormat* format.

from_poscar(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.vasp.VASPPoscarFormat* format.

from_psi4_inp(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.psi4.PSI4InputFormat* format.

from_psi4_out(*file_name*, ***kwargs*)
Read data from *dpdata.plugins.psi4.PSI4OutFormat* format.

from_pwmat_atomconfig(*file_name*, ***kwargs*)
Read data from *dpdata.plugins.pwmat.PwmatAtomconfigFormat* format.

from_pwmat_finalconfig(*file_name*, ***kwargs*)
Read data from *dpdata.plugins.pwmat.PwmatAtomconfigFormat* format.

from_pwmat_mlmd(*file_name*, ***kwargs*)
Read data from *dpdata.plugins.pwmat.PwmatOutputFormat* format.

from_pwmat_movement(*file_name*, ***kwargs*)
Read data from *dpdata.plugins.pwmat.PwmatOutputFormat* format.

from_pwmat_output(*file_name*, ***kwargs*)
Read data from *dpdata.plugins.pwmat.PwmatOutputFormat* format.

from_pymatgen_computedstructureentry(*file_name*, ***kwargs*)
Read data from *dpdata.plugins.pymatgen.PyMatgenCSEFormat* format.

from_pymatgen_molecule(*file_name*, ***kwargs*)
Read data from *dpdata.plugins.pymatgen.PyMatgenMoleculeFormat* format.

from_pymatgen_structure(*file_name*, ***kwargs*)
Read data from *dpdata.plugins.pymatgen.PyMatgenStructureFormat* format.

from_qe_cp_traj(*file_name*, ***kwargs*)
Read data from *dpdata.plugins.qe.QECPTrajFormat* format.

from_qe_pw_scf(*file_name*, ***kwargs*)
Read data from *dpdata.plugins.qe.QECPPWSCFFormat* format.

from_quip_gap_xyz(*file_name*, ***kwargs*)
Read data from *dpdata.plugins.xyz.QuipGapXYZFormat* format.

from_quip_gap_xyz_file(*file_name*, ***kwargs*)
Read data from *dpdata.plugins.xyz.QuipGapXYZFormat* format.

from_sdf(*file_name*, ***kwargs*)
Read data from *dpdata.plugins.rdkit.SdfFormat* format.

from_sdf_file(*file_name*, ***kwargs*)
Read data from *dpdata.plugins.rdkit.SdfFormat* format.

from_siesta_aiMD_output(*file_name*, ***kwargs*)
Read data from *dpdata.plugins.siesta.SiestaAIMDOutputFormat* format.

from_siesta_aimd_output(*file_name*, ***kwargs*)
Read data from *dpdata.plugins.siesta.SiestaAIMDOutputFormat* format.

from_siesta_output(*file_name*, ***kwargs*)
Read data from *dpdata.plugins.siesta.SiestaOutputFormat* format.

from_sqm_in(*file_name*, ***kwargs*)
Read data from *dpdata.plugins.amber.SQMINFormat* format.

from_sqm_out(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.amber.SQMOutFormat* format.

from_stru(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.abacus.AbacusSTRUFormat* format.

from_vasp_contcar(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.vasp.VASPPoscarFormat* format.

from_vasp_outcar(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.vasp.VASPOutcarFormat* format.

from_vasp_poscar(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.vasp.VASPPoscarFormat* format.

from_vasp_string(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.vasp.VASPStringFormat* format.

from_vasp_xml(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.vasp.VASPXMLFormat* format.

from_xml(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.vasp.VASPXMLFormat* format.

from_xyz(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.xyz.XYZFormat* format.

get_atom_names()

Returns name of atoms.

get_atom_numbs()

Returns number of atoms.

get_atom_types()

Returns type of atoms.

get_natoms()

Returns total number of atoms in the system.

get_nframes()

Returns number of frames in the system.

get_ntypes() → int

Returns total number of atom types in the system.

static load(*filename*)

Rebuild System obj. from .json or .yaml file.

map_atom_types(*type_map=None*) → ndarray

Map the atom types of the system.

Parameters

type_map

dict : {"H":0,"O":1} or list ["H","C","O","N"] The map between elements and index if no map_dict is given, index will be set according to atomic number

Returns

new_atom_types
[np.ndarray] The mapped atom types

`minimize`(**args: Any*, *minimizer: str | Minimizer*, ***kwargs: Any*) → *LabeledSystem*

Minimize the geometry.

Parameters

***args**
[iterable] Arguments passing to the minimizer

minimizer
[str or Minimizer] The assigned minimizer

****kwargs**
[dict] Other arguments passing to the minimizer

Returns

labeled_sys
[LabeledSystem] A new labeled system.

`property nopbc`

`perturb`(*pert_num*, *cell_pert_fraction*, *atom_pert_distance*, *atom_pert_style='normal'*)

Perturb each frame in the system randomly. The cell will be deformed randomly, and atoms will be displaced by a random distance in random direction.

Parameters

pert_num
[int] Each frame in the system will make *pert_num* copies, and all the copies will be perturbed. That means the system to be returned will contain *pert_num* * frame_num of the input system.

cell_pert_fraction
[float] A fraction determines how much (relatively) will cell deform. The cell of each frame is deformed by a symmetric matrix perturbed from identity. The perturbation to the diagonal part is subject to a uniform distribution in [-cell_pert_fraction, cell_pert_fraction), and the perturbation to the off-diagonal part is subject to a uniform distribution in [-0.5*cell_pert_fraction, 0.5*cell_pert_fraction).

atom_pert_distance
[float] unit: Angstrom. A distance determines how far atoms will move. Atoms will move about *atom_pert_distance* in random direction. The distribution of the distance atoms move is determined by atom_pert_style

atom_pert_style
[str] Determines the distribution of the distance atoms move is subject to. Avaliable options are

- ***'normal'*: the *distance* will be object to *chi-square distribution with 3 degrees of freedom* after normalization.**
  The mean value of the distance is *atom_pert_fraction*side_length*
- ***'uniform'*: will generate uniformly random points in a 3D-balls with radius as *atom_pert_distance*.**
  These points are treated as vector used by atoms to move. Obviously, the max length of the distance atoms move is *atom_pert_distance*.
- *'const'*: The distance atoms move will be a constant *atom_pert_distance*.

Returns

perturbed_system
[System] The perturbed_system. It contains *pert_num* * frame_num of the input system frames.

`pick_atom_idx`(*idx*, *nopbc=None*)

Pick atom index.

Parameters

idx
[int or list or slice] atom index

nopbc
[Boolen (default: None)] If nopbc is True or False, set nopbc

Returns

new_sys: System
new system

`pick_by_amber_mask`(*param*, *maskstr*, *pass_coords=False*, *nopbc=None*)

Pick atoms by amber mask.

Parameters

param
[str or parmed.Structure] filename of Amber param file or parmed.Structure

maskstr
[str] Amber masks

pass_coords
[Boolen (default: False)] If pass_coords is true, the function will pass coordinates and return a MultiSystem. Otherwise, the result is coordinate-independent, and the function will return System or LabeledSystem.

nopbc
[Boolen (default: None)] If nopbc is True or False, set nopbc

`post_funcs = <dpdata.plugin.Plugin object>`

`predict`(**args: Any*, *driver: str = 'dp'*, ***kwargs: Any*) → *LabeledSystem*

Predict energies and forces by a driver.

Parameters

***args**
[iterable] Arguments passing to the driver

driver
[str, default=dp] The assigned driver. For compatibility, default is dp

****kwargs**
[dict] Other arguments passing to the driver

Returns

labeled_sys
[LabeledSystem] A new labeled system.

Examples

The default driver is DP:

```
>>> labeled_sys = ori_sys.predict("frozen_model_compressed.pb")
```

classmethod register_data_type(**data_type: Tuple[DataType]*)

Register data type.

Parameters

***data_type**
[tuple[DataType]] data type to be regiestered

remove_atom_names(*atom_names*)

Remove atom names and all such atoms. For example, you may not remove EP atoms in TIP4P/Ew water, which is not a real atom.

remove_pbc(*protect_layer=9*)

This method does NOT delete the definition of the cells, it (1) revises the cell to a cubic cell and ensures that the cell boundary to any atom in the system is no less than *protect_layer* (2) translates the system such that the center-of-geometry of the system locates at the center of the cell.

Parameters

protect_layer
[the protect layer between the atoms and the cell] boundary

replace(*initial_atom_type*, *end_atom_type*, *replace_num*)

replicate(*ncopy*)

Replicate the each frame in the system in 3 dimensions. Each frame in the system will become a supercell.

Parameters

ncopy
list: [4,2,3] or tuple: (4,2,3,) make *ncopy[0]* copys in x dimensions, make *ncopy[1]* copys in y dimensions, make *ncopy[2]* copys in z dimensions.

Returns

tmp
[System] The system after replication.

rot_frame_lower_triangular(*f_idx=0*)

rot_lower_triangular()

property short_formula: str

Return the short formula of this system. Elements with zero number will be removed.

property short_name: str

Return the short name of this system (no more than 255 bytes), in the following order:

- formula
- short_formula
- formula_hash.

`shuffle()`

Shuffle frames randomly.

`sort_atom_names(type_map=None)`

Sort atom_names of the system and reorder atom_numbs and atom_types accoarding to atom_names. If type_map is not given, atom_names will be sorted by alphabetical order. If type_map is given, atom_names will be type_map.

Parameters

type_map
[list] type_map

`sort_atom_types()` → ndarray

Sort atom types.

Returns

idx
[np.ndarray] new atom index in the Axis.NATOMS

`sub_system(f_idx)`

Construct a subsystem from the system.

Parameters

f_idx
[int or index] Which frame to use in the subsystem

Returns

sub_system
[System] The subsystem

`to(fmt: str, *args, **kwargs)` → *System*

Dump systems to the specific format.

Parameters

fmt
[str] format

***args**
arguments

****kwargs**
keyword arguments

Returns

System
self

`to_3dmol(*args, **kwargs)`

Dump data to *dpdata.plugins.3dmol.Py3DMolFormat* format.

`to_abacus_lcao_md(*args, **kwargs)`

Dump data to *dpdata.plugins.abacus.AbacusMDFormat* format.

`to_abacus_lcao_relax(*args, **kwargs)`

Dump data to *dpdata.plugins.abacus.AbacusRelaxFormat* format.

to_abacus_lcao_scf(**args*, ***kwargs*)

Dump data to *dpdata.plugins.abacus.AbacusSCFFormat* format.

to_abacus_md(**args*, ***kwargs*)

Dump data to *dpdata.plugins.abacus.AbacusMDFormat* format.

to_abacus_pw_md(**args*, ***kwargs*)

Dump data to *dpdata.plugins.abacus.AbacusMDFormat* format.

to_abacus_pw_relax(**args*, ***kwargs*)

Dump data to *dpdata.plugins.abacus.AbacusRelaxFormat* format.

to_abacus_pw_scf(**args*, ***kwargs*)

Dump data to *dpdata.plugins.abacus.AbacusSCFFormat* format.

to_abacus_relax(**args*, ***kwargs*)

Dump data to *dpdata.plugins.abacus.AbacusRelaxFormat* format.

to_abacus_scf(**args*, ***kwargs*)

Dump data to *dpdata.plugins.abacus.AbacusSCFFormat* format.

to_abacus_stru(**args*, ***kwargs*)

Dump data to *dpdata.plugins.abacus.AbacusSTRUFormat* format.

to_amber_md(**args*, ***kwargs*)

Dump data to *dpdata.plugins.amber.AmberMDFormat* format.

to_ase_structure(**args*, ***kwargs*)

Dump data to *dpdata.plugins.ase.ASEStructureFormat* format.

to_ase_traj(**args*, ***kwargs*)

Dump data to *dpdata.plugins.ase.ASETrajFormat* format.

to_atomconfig(**args*, ***kwargs*)

Dump data to *dpdata.plugins.pwmat.PwmatAtomconfigFormat* format.

to_contcar(**args*, ***kwargs*)

Dump data to *dpdata.plugins.vasp.VASPPoscarFormat* format.

to_cp2k_aimd_output(**args*, ***kwargs*)

Dump data to *dpdata.plugins.cp2k.CP2KAIMDOutputFormat* format.

to_cp2k_output(**args*, ***kwargs*)

Dump data to *dpdata.plugins.cp2k.CP2KOutputFormat* format.

to_deepmd(**args*, ***kwargs*)

Dump data to *dpdata.plugins.deepmd.DeePMDRawFormat* format.

to_deepmd_comp(**args*, ***kwargs*)

Dump data to *dpdata.plugins.deepmd.DeePMDCompFormat* format.

to_deepmd_hdf5(**args*, ***kwargs*)

Dump data to *dpdata.plugins.deepmd.DeePMDHDF5Format* format.

to_deepmd_npy(**args*, ***kwargs*)

Dump data to *dpdata.plugins.deepmd.DeePMDCompFormat* format.

to_deepmd_npy_mixed(**args*, ***kwargs*)

Dump data to *dpdata.plugins.deepmd.DeePMDMixedFormat* format.

to_deepmd_raw(**args*, ***kwargs*)

Dump data to *dpdata.plugins.deepmd.DeePMDRawFormat* format.

to_dftbplus(**args*, ***kwargs*)

Dump data to *dpdata.plugins.dftbplus.DFTBplusFormat* format.

to_dump(**args*, ***kwargs*)

Dump data to *dpdata.plugins.lammps.LAMMPSDumpFormat* format.

to_fhi_aims_md(**args*, ***kwargs*)

Dump data to *dpdata.plugins.fhi_aims.FhiMDFormat* format.

to_fhi_aims_output(**args*, ***kwargs*)

Dump data to *dpdata.plugins.fhi_aims.FhiMDFormat* format.

to_fhi_aims_scf(**args*, ***kwargs*)

Dump data to *dpdata.plugins.fhi_aims.FhiSCFFormat* format.

to_finalconfig(**args*, ***kwargs*)

Dump data to *dpdata.plugins.pwmat.PwmatAtomconfigFormat* format.

to_fmt_obj(*fmtobj*, **args*, ***kwargs*)

to_gaussian_gjf(**args*, ***kwargs*)

Dump data to *dpdata.plugins.gaussian.GaussiaGJFFormat* format.

to_gaussian_log(**args*, ***kwargs*)

Dump data to *dpdata.plugins.gaussian.GaussianLogFormat* format.

to_gaussian_md(**args*, ***kwargs*)

Dump data to *dpdata.plugins.gaussian.GaussianMDFormat* format.

to_gro(**args*, ***kwargs*)

Dump data to *dpdata.plugins.gromacs.GromacsGroFormat* format.

to_gromacs_gro(**args*, ***kwargs*)

Dump data to *dpdata.plugins.gromacs.GromacsGroFormat* format.

to_lammps_dump(**args*, ***kwargs*)

Dump data to *dpdata.plugins.lammps.LAMMPSDumpFormat* format.

to_lammps_lmp(**args*, ***kwargs*)

Dump data to *dpdata.plugins.lammps.LAMMPSLmpFormat* format.

to_list(**args*, ***kwargs*)

Dump data to *dpdata.plugins.list.ListFormat* format.

to_lmp(**args*, ***kwargs*)

Dump data to *dpdata.plugins.lammps.LAMMPSLmpFormat* format.

to_mlmd(**args*, ***kwargs*)

Dump data to *dpdata.plugins.pwmat.PwmatOutputFormat* format.

to_mol(**args*, ***kwargs*)

Dump data to *dpdata.plugins.rdkit.MolFormat* format.

to_mol_file(**args*, ***kwargs*)

Dump data to *dpdata.plugins.rdkit.MolFormat* format.

to_movement(**args*, ***kwargs*)

Dump data to *dpdata.plugins.pwmat.PwmatOutputFormat* format.

to_n2p2(**args*, ***kwargs*)

Dump data to *dpdata.plugins.n2p2.N2P2Format* format.

to_openmx_md(**args*, ***kwargs*)

Dump data to *dpdata.plugins.openmx.OPENMXFormat* format.

to_orca_spout(**args*, ***kwargs*)

Dump data to *dpdata.plugins.orca.ORCASPOutFormat* format.

to_outcar(**args*, ***kwargs*)

Dump data to *dpdata.plugins.vasp.VASPOutcarFormat* format.

to_poscar(**args*, ***kwargs*)

Dump data to *dpdata.plugins.vasp.VASPPoscarFormat* format.

to_psi4_inp(**args*, ***kwargs*)

Dump data to *dpdata.plugins.psi4.PSI4InputFormat* format.

to_psi4_out(**args*, ***kwargs*)

Dump data to *dpdata.plugins.psi4.PSI4OutFormat* format.

to_pwmat_atomconfig(**args*, ***kwargs*)

Dump data to *dpdata.plugins.pwmat.PwmatAtomconfigFormat* format.

to_pwmat_finalconfig(**args*, ***kwargs*)

Dump data to *dpdata.plugins.pwmat.PwmatAtomconfigFormat* format.

to_pwmat_mlmd(**args*, ***kwargs*)

Dump data to *dpdata.plugins.pwmat.PwmatOutputFormat* format.

to_pwmat_movement(**args*, ***kwargs*)

Dump data to *dpdata.plugins.pwmat.PwmatOutputFormat* format.

to_pwmat_output(**args*, ***kwargs*)

Dump data to *dpdata.plugins.pwmat.PwmatOutputFormat* format.

to_pymatgen_ComputedStructureEntry(**args*, ***kwargs*)

Dump data to *dpdata.plugins.pymatgen.PyMatgenCSEFormat* format.

to_pymatgen_computedstructureentry(**args*, ***kwargs*)

Dump data to *dpdata.plugins.pymatgen.PyMatgenCSEFormat* format.

to_pymatgen_molecule(**args*, ***kwargs*)

Dump data to *dpdata.plugins.pymatgen.PyMatgenMoleculeFormat* format.

to_pymatgen_structure(**args*, ***kwargs*)

Dump data to *dpdata.plugins.pymatgen.PyMatgenStructureFormat* format.

to_qe_cp_traj(**args*, ***kwargs*)

Dump data to *dpdata.plugins.qe.QECPTrajFormat* format.

to_qe_pw_scf(**args*, ***kwargs*)

Dump data to *dpdata.plugins.qe.QECPPWSCFFormat* format.

to_quip_gap_xyz(**args*, ***kwargs*)

Dump data to *dpdata.plugins.xyz.QuipGapXYZFormat* format.

to_quip_gap_xyz_file(**args*, ***kwargs*)

Dump data to *dpdata.plugins.xyz.QuipGapXYZFormat* format.

to_sdf(**args*, ***kwargs*)

Dump data to *dpdata.plugins.rdkit.SdfFormat* format.

to_sdf_file(**args*, ***kwargs*)

Dump data to *dpdata.plugins.rdkit.SdfFormat* format.

to_siesta_aimd_output(**args*, ***kwargs*)

Dump data to *dpdata.plugins.siesta.SiestaAIMDOutputFormat* format.

to_siesta_output(**args*, ***kwargs*)

Dump data to *dpdata.plugins.siesta.SiestaOutputFormat* format.

to_sqm_in(**args*, ***kwargs*)

Dump data to *dpdata.plugins.amber.SQMINFormat* format.

to_sqm_out(**args*, ***kwargs*)

Dump data to *dpdata.plugins.amber.SQMOutFormat* format.

to_stru(**args*, ***kwargs*)

Dump data to *dpdata.plugins.abacus.AbacusSTRUFormat* format.

to_vasp_contcar(**args*, ***kwargs*)

Dump data to *dpdata.plugins.vasp.VASPPoscarFormat* format.

to_vasp_outcar(**args*, ***kwargs*)

Dump data to *dpdata.plugins.vasp.VASPOutcarFormat* format.

to_vasp_poscar(**args*, ***kwargs*)

Dump data to *dpdata.plugins.vasp.VASPPoscarFormat* format.

to_vasp_string(**args*, ***kwargs*)

Dump data to *dpdata.plugins.vasp.VASPStringFormat* format.

to_vasp_xml(**args*, ***kwargs*)

Dump data to *dpdata.plugins.vasp.VASPXMLFormat* format.

to_xml(**args*, ***kwargs*)

Dump data to *dpdata.plugins.vasp.VASPXMLFormat* format.

to_xyz(**args*, ***kwargs*)

Dump data to *dpdata.plugins.xyz.XYZFormat* format.

property uniq_formula

Return the uniq_formula of this system. The uniq_formula sort the elements in formula by names. Systems with the same uniq_formula can be append together.

6.1.1 Subpackages

dpdata.abacus package

Submodules

dpdata.abacus.md module

dpdata.abacus.md.**get_coord_dump_freq**(*inlines*)

dpdata.abacus.md.**get_coords_from_dump**(*dumplines*, *natoms*)

dpdata.abacus.md.**get_energy**(*outlines*, *ndump*, *dump_freq*)

dpdata.abacus.md.**get_frame**(*fname*)

dpdata.abacus.md.**get_path_out**(*fname*, *inlines*)

dpdata.abacus.relax module

dpdata.abacus.relax.**get_coords_from_log**(*loglines*, *natoms*)

NOTICE: unit of coords and cells is Angstrom order:

coordinate cell (no output if cell is not changed) energy (no output, if SCF is not converged) force (no output, if cal_force is not setted or abnormal ending) stress (no output, if set cal_stress is not setted or abnormal ending).

dpdata.abacus.relax.**get_frame**(*fname*)

dpdata.abacus.relax.**get_log_file**(*fname*, *inlines*)

dpdata.abacus.scf module

dpdata.abacus.scf.**CheckFile**(*ifile*)

dpdata.abacus.scf.**collect_force**(*outlines*)

dpdata.abacus.scf.**collect_stress**(*outlines*)

dpdata.abacus.scf.**get_block**(*lines*, *keyword*, *skip=0*, *nlines=None*)

dpdata.abacus.scf.**get_cell**(*geometry_inlines*)

dpdata.abacus.scf.**get_coords**(*celldm*, *cell*, *geometry_inlines*, *inlines=None*)

dpdata.abacus.scf.**get_energy**(*outlines*)

dpdata.abacus.scf.**get_force**(*outlines*, *natoms*)

dpdata.abacus.scf.**get_frame**(*fname*)

dpdata.abacus.scf.**get_frame_from_stru**(*fname*)

dpdata.abacus.scf.**get_geometry_in**(*fname*, *inlines*)

dpdata.abacus.scf.**get_nele_from_stru**(*geometry_inlines*)

dpdata.abacus.scf.**get_path_out**(*fname*, *inlines*)

dpdata.abacus.scf.**get_stress**(*outlines*)

dpdata.abacus.scf.**get_stru_block**(*lines*, *keyword*)

dpdata.abacus.scf.**make_unlabeled_stru**(*data*, *frame_idx*, *pp_file=None*, *numerical_orbital=None*, *numerical_descriptor=None*, *mass=None*)

dpdata.amber package

Submodules

dpdata.amber.mask module

Amber mask.

dpdata.amber.mask.**load_param_file**(*param_file*)

dpdata.amber.mask.**pick_by_amber_mask**(*param*, *maskstr*, *coords=None*)

Pick atoms by amber masks.

Parameters

param
[str or parmed.Structure] filename of Amber param file or parmed.Structure

maskstr
[str] Amber masks

coords
[np.ndarray (optional)] frame coordinates, shape: N*3

dpdata.amber.md module

dpdata.amber.md.**read_amber_traj**(*parm7_file*, *nc_file*, *mdfrc_file=None*, *mden_file=None*, *mdout_file=None*, *use_element_symbols=None*, *labeled=True*)

The amber trajectory includes: * nc, NetCDF format, stores coordinates * mdfrc, NetCDF format, stores forces * mden (optional), text format, stores energies * mdout (optional), text format, may store energies if there is no mden_file * parm7, text format, stores types.

Parameters

parm7_file, nc_file, mdfrc_file, mden_file, mdout_file:
filenames

use_element_symbols
[None or list or str] If use_element_symbols is a list of atom indexes, these atoms will use element symbols instead of amber types. For example, a ligand will use C, H, O, N, and so on instead of h1, hc, o, os, and so on. IF use_element_symbols is str, it will be considered as Amber mask.

labeled
[bool] Whether to return labeled data

dpdata.amber.sqm module

dpdata.amber.sqm.**make_sqm_in**(*data*, *fname=None*, *frame_idx=0*, ***kwargs*)

dpdata.amber.sqm.**parse_sqm_out**(*fname*)

Read atom symbols, charges and coordinates from ambertools sqm.out file.

dpdata.cp2k package

Submodules

dpdata.cp2k.cell module

dpdata.cp2k.cell.**cell_to_low_triangle**(*A*, *B*, *C*, *alpha*, *beta*, *gamma*)

Convert cell to low triangle matrix.

Parameters

A
[float] cell length A

B
[float] cell length B

C
[float] cell length C

alpha
[float] radian. The angle between vector B and vector C.

beta
[float] radian. The angle between vector A and vector C.

gamma
[float] radian. The angle between vector B and vector C.

Returns

cell
[list] The cell matrix used by dpdata in low triangle form.

dpdata.cp2k.output module

class dpdata.cp2k.output.**Cp2kSystems**(*log_file_name*, *xyz_file_name*, *restart=False*)

Bases: object

deal with cp2k outputfile.

Methods

get_log_block_generator
get_xyz_block_generator
handle_single_log_frame
handle_single_xyz_frame

get_log_block_generator()

get_xyz_block_generator()

handle_single_log_frame(*lines*)

handle_single_xyz_frame(*lines*)

dpdata.cp2k.output.**get_frames**(*fname*)

dpdata.deepmd package

Submodules

dpdata.deepmd.comp module

dpdata.deepmd.comp.**dump**(*folder*, *data*, *set_size=5000*, *comp_prec=<class 'numpy.float32'>*, *remove_sets=True*)

dpdata.deepmd.comp.**to_system_data**(*folder*, *type_map=None*, *labels=True*)

dpdata.deepmd.hdf5 module

Utils for deepmd/hdf5 format.

dpdata.deepmd.hdf5.**dump**(*f: ~h5py._hl.files.File | ~h5py._hl.group.Group*, *folder: str*, *data: dict*, *set_size=5000*, *comp_prec=<class 'numpy.float32'>*) → None

Dump data to a HDF5 file.

Parameters

f
[h5py.File or h5py.Group] HDF5 file or group object

folder
[str] path in the HDF5 file

data
[dict] System or LabeledSystem data

set_size
[int, default: 5000] size of a set

comp_prec
[np.dtype, default: np.float32] precision of data

dpdata.deepmd.hdf5.**to_system_data**(*f: File | Group*, *folder: str*, *type_map: list | None = None*, *labels: bool = True*)

Load a HDF5 file.

Parameters

f
[h5py.File or h5py.Group] HDF5 file or group object

folder
[str] path in the HDF5 file

type_map
[list] type map

labels
[bool] labels

dpdata.deepmd.mixed module

dpdata.deepmd.mixed.**dump**(*folder*, *data*, *set_size=2000*, *comp_prec=<class 'numpy.float32'>*, *remove_sets=True*)

dpdata.deepmd.mixed.**formula**(*atom_names*, *atom_numbs*)

Return the formula of this system, like C3H5O2.

dpdata.deepmd.mixed.**load_type**(*folder*)

dpdata.deepmd.mixed.**mix_system**(**system*, *type_map*, ***kwargs*)

Mix the systems into mixed_type ones according to the unified given type_map.

Parameters

***system**
[System] The systems to mix

type_map
[list of str] Maps atom type to name

****kwargs**
[dict] Other parameters

Returns

mixed_systems: dict
dict of mixed system with key 'atom_numbs'

dpdata.deepmd.mixed.**split_system**(*sys*, *split_num=10000*)

dpdata.deepmd.mixed.**to_system_data**(*folder*, *type_map=None*, *labels=True*)

dpdata.deepmd.raw module

dpdata.deepmd.raw.**dump**(*folder*, *data*)

dpdata.deepmd.raw.**load_type**(*folder*, *type_map=None*)

dpdata.deepmd.raw.**to_system_data**(*folder*, *type_map=None*, *labels=True*)

dpdata.dftbplus package

Submodules

dpdata.dftbplus.output module

dpdata.dftbplus.output.**read_dftb_plus**(*fn_1: str*, *fn_2: str*) → Tuple[str, ndarray, float, ndarray]

Read from DFTB+ input and output.

Parameters

fn_1
[str] DFTB+ input file name

fn_2
[str] DFTB+ output file name

Returns

str
atomic symbols

np.ndarray
atomic coordinates

float
total potential energy

np.ndarray
atomic forces

dpdata.fhi_aims package

Submodules

dpdata.fhi_aims.output module

dpdata.fhi_aims.output.**analyze_block**(*lines*, *first_blk=False*, *md=True*)

dpdata.fhi_aims.output.**get_fhi_aims_block**(*fp*)

dpdata.fhi_aims.output.**get_frames**(*fname*, *md=True*, *begin=0*, *step=1*, *convergence_check=True*)

dpdata.fhi_aims.output.**get_info**(*lines*, *type_idx_zero=False*)

dpdata.gaussian package

Submodules

dpdata.gaussian.gjf module

Generate Gaussian input file.

dpdata.gaussian.gjf.**detect_multiplicity**(*symbols: ndarray*) → int

Find the minimal multiplicity of the given molecules.

Parameters

symbols
[np.ndarray] element symbols; virtual elements are not supported

Returns

int
spin multiplicity

dpdata.gaussian.gjf.**make_gaussian_input**(*sys_data: dict*, *keywords: str | List[str]*, *multiplicity: str | int = 'auto'*, *charge: int = 0*, *fragment_guesses: bool = False*, *basis_set: str | None = None*, *keywords_high_multiplicity: str | None = None*, *nproc: int = 1*) → str

Make gaussian input file.

Parameters

sys_data
[dict] system data

keywords
[str or list[str]] Gaussian keywords, e.g. force b3lyp/6-31g**. If a list, run multiple steps

multiplicity
[str or int, default=auto] spin multiplicity state. It can be a number. If auto, multiplicity will be detected automatically, with the following rules:

fragment_guesses=True
multiplicity will +1 for each radical, and +2 for each oxygen molecule

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

charge
[int, default=0] molecule charge. Only used when charge is not provided by the system

fragment_guesses
[bool, default=False] initial guess generated from fragment guesses. If True, multiplicity should be auto

basis_set
[str, default=None] custom basis set

keywords_high_multiplicity
[str, default=None] keywords for points with multiple raicals. multiplicity should be auto. If not set, fallback to normal keywords

nproc
[int, default=1] Number of CPUs to use

Returns

str
gjf output string

dpdata.gaussian.gjf.**read_gaussian_input**(*inp: str*)

Read Gaussian input.

Parameters

inp
[str] Gaussian input str

Returns

dict
system data

dpdata.gaussian.log module

dpdata.gaussian.log.**to_system_data**(*file_name*, *md=False*)

Read Gaussian log file.

Parameters

file_name
[str] file name

md
[bool, default False] whether to read multiple frames

Returns

data
[dict] system data

Raises

RuntimeError
if the input orientation is not found

dpdata.gromacs package

Submodules

dpdata.gromacs.gro module

dpdata.gromacs.gro.**file_to_system_data**(*fname*, *format_atom_name=True*, ***kwargs*)

dpdata.gromacs.gro.**from_system_data**(*system*, *f_idx=0*, ***kwargs*)

dpdata.lammps package

Submodules

dpdata.lammps.dump module

exception dpdata.lammps.dump.**UnwrapWarning**

Bases: UserWarning

dpdata.lammps.dump.**box2dumpbox**(*orig*, *box*)

dpdata.lammps.dump.**dumpbox2box**(*bounds*, *tilt*)

dpdata.lammps.dump.**get_atype**(*lines*, *type_idx_zero=False*)

dpdata.lammps.dump.**get_coordtype_and_scalefactor**(*keys*)

dpdata.lammps.dump.**get_dumpbox**(*lines*)

dpdata.lammps.dump.**get_natoms**(*lines*)

dpdata.lammps.dump.**get_natoms_vec**(*lines*)

dpdata.lammps.dump.**get_natomtypes**(*lines*)

dpdata.lammps.dump.**load_file**(*fname*, *begin=0*, *step=1*)

dpdata.lammps.dump.**safe_get_posi**(*lines*, *cell*, *orig=array([0., 0., 0.])*, *unwrap=False*)

dpdata.lammps.dump.**split_traj**(*dump_lines*)

dpdata.lammps.dump.**system_data**(*lines*, *type_map=None*, *type_idx_zero=True*, *unwrap=False*)

dpdata.lammps.lmp module

dpdata.lammps.lmp.**box2lmpbox**(*orig*, *box*)

dpdata.lammps.lmp.**from_system_data**(*system*, *f_idx=0*)

dpdata.lammps.lmp.**get_atoms**(*lines*)

dpdata.lammps.lmp.**get_atype**(*lines*, *type_idx_zero=False*)

dpdata.lammps.lmp.**get_lmpbox**(*lines*)

dpdata.lammps.lmp.**get_natoms**(*lines*)

dpdata.lammps.lmp.**get_natoms_vec**(*lines*)

dpdata.lammps.lmp.**get_natomtypes**(*lines*)

dpdata.lammps.lmp.**get_posi**(*lines*)

dpdata.lammps.lmp.**lmpbox2box**(*lohi*, *tilt*)

dpdata.lammps.lmp.**system_data**(*lines*, *type_map=None*, *type_idx_zero=True*)

dpdata.lammps.lmp.**to_system_data**(*lines*, *type_map=None*, *type_idx_zero=True*)

dpdata.openmx package

Submodules

dpdata.openmx.omx module

dpdata.openmx.omx.**load_atom**(*lines*)

dpdata.openmx.omx.**load_cells**(*lines*)

dpdata.openmx.omx.**load_coords**(*lines*, *atom_names*, *natoms*)

dpdata.openmx.omx.**load_data**(*mdname*, *atom_names*, *natoms*)

dpdata.openmx.omx.**load_energy**(*lines*)

dpdata.openmx.omx.**load_force**(*lines*, *atom_names*, *atom_numbs*)

dpdata.openmx.omx.**load_param_file**(*fname*, *mdname*)

dpdata.openmx.omx.**to_system_data**(*fname*, *mdname*)

dpdata.openmx.omx.**to_system_label**(*fname*, *mdname*)

dpdata.orca package

Submodules

dpdata.orca.output module

dpdata.orca.output.**read_orca_sp_output**(*fn: str*) → Tuple[ndarray, ndarray, float, ndarray]

Read from ORCA output.

Note that both the energy and the gradient should be printed.

Parameters

fn
[str] file name

Returns

np.ndarray
atomic symbols

np.ndarray
atomic coordinates

float
total potential energy

np.ndarray
atomic forces

dpdata.plugins package

Submodules

dpdata.plugins.3dmol module

class dpdata.plugins.3dmol.**Py3DMolFormat**

Bases: *Format*

3DMol format.

To use this format, py3Dmol should be installed in advance.

Methods

MultiModes()	File mode for MultiSystems.
from_bond_order_system(file_name, **kwargs)	Implement BondOrderSystem.from that converts from this format to BondOrderSystem.
from_labeled_system(file_name, **kwargs)	Implement LabeledSystem.from that converts from this format to LabeledSystem.
from_multi_systems(directory, **kwargs)	Implement MultiSystems.from that converts from this format to MultiSystems.
from_system(file_name, **kwargs)	Implement System.from that converts from this format to System.
get_formats()	Get all registered formats.
get_from_methods()	Get all registered from methods.
get_to_methods()	Get all registered to methods.
mix_system(*system, type_map, **kwargs)	Mix the systems into mixed_type ones according to the unified given type_map.
post(func_name)	Register a post function for from method.
register(key)	Register a format plugin.
register_from(key)	Register a from method if the target method name is not default.
register_to(key)	Register a to method if the target method name is not default.
to_bond_order_system(data, rdkit_mol, *args, ...)	Implement BondOrderSystem.to that converts from BondOrderSystem to this format.
to_labeled_system(data, *args, **kwargs)	Implement LabeledSystem.to that converts from LabeledSystem to this format.
to_multi_systems(formulas, directory, **kwargs)	Implement MultiSystems.to that converts from MultiSystems to this format.
to_system(data[, f_idx, size, style])	Show 3D structure of a frame in jupyter.

to_system(*data: dict*, *f_idx: int = 0*, *size: Tuple[int] = (300, 300)*, *style: dict = {'sphere': {'radius': 0.4}, 'stick': {}}*, ***kwargs*)

Show 3D structure of a frame in jupyter.

Parameters

data
[dict] system data

f_idx
[int] frame index to show

size
[tuple[int]] (width, height) of the widget

style
[dict] style of 3DMol. Read 3DMol documentation for details.

****kwargs**
[dict] other parameters

Examples

```
>>> system.to_3dmol()
```

dpdata.plugins.abacus module

class dpdata.plugins.abacus.**AbacusMDFormat**

Bases: *Format*

Methods

MultiModes()	File mode for MultiSystems.
from_bond_order_system(file_name, **kwargs)	Implement BondOrderSystem.from that converts from this format to BondOrderSystem.
from_labeled_system(file_name, **kwargs)	Implement LabeledSystem.from that converts from this format to LabeledSystem.
from_multi_systems(directory, **kwargs)	Implement MultiSystems.from that converts from this format to MultiSystems.
from_system(file_name, **kwargs)	Implement System.from that converts from this format to System.
get_formats()	Get all registered formats.
get_from_methods()	Get all registered from methods.
get_to_methods()	Get all registered to methods.
mix_system(*system, type_map, **kwargs)	Mix the systems into mixed_type ones according to the unified given type_map.
post(func_name)	Register a post function for from method.
register(key)	Register a format plugin.
register_from(key)	Register a from method if the target method name is not default.
register_to(key)	Register a to method if the target method name is not default.
to_bond_order_system(data, rdkit_mol, *args, ...)	Implement BondOrderSystem.to that converts from BondOrderSystem to this format.
to_labeled_system(data, *args, **kwargs)	Implement LabeledSystem.to that converts from LabeledSystem to this format.
to_multi_systems(formulas, directory, **kwargs)	Implement MultiSystems.to that converts from MultiSystems to this format.
to_system(data, *args, **kwargs)	Implement System.to that converts from System to this format.

from_labeled_system(*file_name*, ***kwargs*)

Implement LabeledSystem.from that converts from this format to LabeledSystem.

Parameters

file_name
[str] file name, i.e. the first argument

****kwargs**
[dict] keyword arguments that will be passed from the method

Returns

data
[dict] system data, whose keys are defined in LabeledSystem.DTYPES

class dpdata.plugins.abacus.**AbacusRelaxFormat**

Bases: *Format*

Methods

MultiModes()	File mode for MultiSystems.
from_bond_order_system(file_name, **kwargs)	Implement BondOrderSystem.from that converts from this format to BondOrderSystem.
from_labeled_system(file_name, **kwargs)	Implement LabeledSystem.from that converts from this format to LabeledSystem.
from_multi_systems(directory, **kwargs)	Implement MultiSystems.from that converts from this format to MultiSystems.
from_system(file_name, **kwargs)	Implement System.from that converts from this format to System.
get_formats()	Get all registered formats.
get_from_methods()	Get all registered from methods.
get_to_methods()	Get all registered to methods.
mix_system(*system, type_map, **kwargs)	Mix the systems into mixed_type ones according to the unified given type_map.
post(func_name)	Register a post function for from method.
register(key)	Register a format plugin.
register_from(key)	Register a from method if the target method name is not default.
register_to(key)	Register a to method if the target method name is not default.
to_bond_order_system(data, rdkit_mol, *args, ...)	Implement BondOrderSystem.to that converts from BondOrderSystem to this format.
to_labeled_system(data, *args, **kwargs)	Implement LabeledSystem.to that converts from LabeledSystem to this format.
to_multi_systems(formulas, directory, **kwargs)	Implement MultiSystems.to that converts from MultiSystems to this format.
to_system(data, *args, **kwargs)	Implement System.to that converts from System to this format.

from_labeled_system(*file_name*, ***kwargs*)

Implement LabeledSystem.from that converts from this format to LabeledSystem.

Parameters

file_name
[str] file name, i.e. the first argument

****kwargs**
[dict] keyword arguments that will be passed from the method

Returns

data
[dict] system data, whose keys are defined in LabeledSystem.DTYPES

class dpdata.plugins.abacus.**AbacusSCFFormat**

Bases: *Format*

Methods

MultiModes()	File mode for MultiSystems.
from_bond_order_system(file_name, **kwargs)	Implement BondOrderSystem.from that converts from this format to BondOrderSystem.
from_labeled_system(file_name, **kwargs)	Implement LabeledSystem.from that converts from this format to LabeledSystem.
from_multi_systems(directory, **kwargs)	Implement MultiSystems.from that converts from this format to MultiSystems.
from_system(file_name, **kwargs)	Implement System.from that converts from this format to System.
get_formats()	Get all registered formats.
get_from_methods()	Get all registered from methods.
get_to_methods()	Get all registered to methods.
mix_system(*system, type_map, **kwargs)	Mix the systems into mixed_type ones according to the unified given type_map.
post(func_name)	Register a post function for from method.
register(key)	Register a format plugin.
register_from(key)	Register a from method if the target method name is not default.
register_to(key)	Register a to method if the target method name is not default.
to_bond_order_system(data, rdkit_mol, *args, ...)	Implement BondOrderSystem.to that converts from BondOrderSystem to this format.
to_labeled_system(data, *args, **kwargs)	Implement LabeledSystem.to that converts from LabeledSystem to this format.
to_multi_systems(formulas, directory, **kwargs)	Implement MultiSystems.to that converts from MultiSystems to this format.
to_system(data, *args, **kwargs)	Implement System.to that converts from System to this format.

from_labeled_system(*file_name*, ***kwargs*)

Implement LabeledSystem.from that converts from this format to LabeledSystem.

Parameters

file_name
[str] file name, i.e. the first argument

****kwargs**
[dict] keyword arguments that will be passed from the method

Returns

data
[dict] system data, whose keys are defined in LabeledSystem.DTYPES

class dpdata.plugins.abacus.**AbacusSTRUFormat**

Bases: *Format*

Methods

MultiModes()	File mode for MultiSystems.
from_bond_order_system(file_name, **kwargs)	Implement BondOrderSystem.from that converts from this format to BondOrderSystem.
from_labeled_system(file_name, **kwargs)	Implement LabeledSystem.from that converts from this format to LabeledSystem.
from_multi_systems(directory, **kwargs)	Implement MultiSystems.from that converts from this format to MultiSystems.
from_system(file_name, **kwargs)	Implement System.from that converts from this format to System.
get_formats()	Get all registered formats.
get_from_methods()	Get all registered from methods.
get_to_methods()	Get all registered to methods.
mix_system(*system, type_map, **kwargs)	Mix the systems into mixed_type ones according to the unified given type_map.
post(func_name)	Register a post function for from method.
register(key)	Register a format plugin.
register_from(key)	Register a from method if the target method name is not default.
register_to(key)	Register a to method if the target method name is not default.
to_bond_order_system(data, rdkit_mol, *args, ...)	Implement BondOrderSystem.to that converts from BondOrderSystem to this format.
to_labeled_system(data, *args, **kwargs)	Implement LabeledSystem.to that converts from LabeledSystem to this format.
to_multi_systems(formulas, directory, **kwargs)	Implement MultiSystems.to that converts from MultiSystems to this format.
to_system(data, file_name[, frame_idx])	Dump the system into ABACUS STRU format file.

from_system(*file_name*, ***kwargs*)

Implement System.from that converts from this format to System.

Parameters

file_name
[str] file name, i.e. the first argument

****kwargs**
[dict] keyword arguments that will be passed from the method

Returns

data
[dict] system data, whose keys are defined in System.DTYPES

`to_system`(*data*, *file_name*, *frame_idx=0*, ***kwargs*)

Dump the system into ABACUS STRU format file.

Parameters

data
[dict] System data

file_name
[str] The output file name

frame_idx
[int] The index of the frame to dump

pp_file
[list of string, optional] List of pseudo potential files

numerical_orbital
[list of string, optional] List of orbital files

mass
[list of float, optional] List of atomic masses

numerical_descriptor
[str, optional] numerical descriptor file

****kwargs**
[dict] other parameters

dpdata.plugins.amber module

`class` dpdata.plugins.amber.**AmberMDFormat**

Bases: *Format*

Methods

MultiModes()	File mode for MultiSystems.
from_bond_order_system(file_name, **kwargs)	Implement BondOrderSystem.from that converts from this format to BondOrderSystem.
from_labeled_system([file_name, parm7_file, ...])	Implement LabeledSystem.from that converts from this format to LabeledSystem.
from_multi_systems(directory, **kwargs)	Implement MultiSystems.from that converts from this format to MultiSystems.
from_system([file_name, parm7_file, ...])	Implement System.from that converts from this format to System.
get_formats()	Get all registered formats.
get_from_methods()	Get all registered from methods.
get_to_methods()	Get all registered to methods.
mix_system(*system, type_map, **kwargs)	Mix the systems into mixed_type ones according to the unified given type_map.
post(func_name)	Register a post function for from method.
register(key)	Register a format plugin.
register_from(key)	Register a from method if the target method name is not default.
register_to(key)	Register a to method if the target method name is not default.
to_bond_order_system(data, rdkit_mol, *args, ...)	Implement BondOrderSystem.to that converts from BondOrderSystem to this format.
to_labeled_system(data, *args, **kwargs)	Implement LabeledSystem.to that converts from LabeledSystem to this format.
to_multi_systems(formulas, directory, **kwargs)	Implement MultiSystems.to that converts from MultiSystems to this format.
to_system(data, *args, **kwargs)	Implement System.to that converts from System to this format.

from_labeled_system(*file_name=None*, *parm7_file=None*, *nc_file=None*, *mdfrc_file=None*, *mden_file=None*, *mdout_file=None*, *use_element_symbols=None*, ***kwargs*)

Implement LabeledSystem.from that converts from this format to LabeledSystem.

Parameters

file_name
[str] file name, i.e. the first argument

****kwargs**
[dict] keyword arguments that will be passed from the method

Returns

data
[dict] system data, whose keys are defined in LabeledSystem.DTYPES

from_system(*file_name=None*, *parm7_file=None*, *nc_file=None*, *use_element_symbols=None*, ***kwargs*)

Implement System.from that converts from this format to System.

Parameters

file_name
[str] file name, i.e. the first argument

****kwargs**
[dict] keyword arguments that will be passed from the method

Returns

data
[dict] system data, whose keys are defined in System.DTYPES

class dpdata.plugins.amber.**SQMDriver**(*sqm_exec: str = 'sqm'*, ***kwargs: dict*)

Bases: *Driver*

AMBER sqm program driver.

Parameters

sqm_exec
[str, default=sqm] path to sqm program

****kwargs**
[dict] other arguments to make input files. See *SQMINFormat*

Examples

Use DFTB3 method to calculate potential energy:

```
>>> labeled_system = system.predict(theory="DFTB3", driver="sqm")
>>> labeled_system['energies'][0]
-15.41111246
```

Attributes

ase_calculator
Returns an ase calculator based on this driver.

Methods

get_driver(key)	Get a driver plugin.
get_drivers()	Get all driver plugins.
label(data)	Label a system data.
register(key)	Register a driver plugin.

label(*data: dict*) → dict

Label a system data. Returns new data with energy, forces, and virials.

Parameters

data
[dict] data with coordinates and atom types

Returns

dict
labeled data with energies and forces

class dpdata.plugins.amber.**SQMINFormat**

Bases: *Format*

Methods

MultiModes()	File mode for MultiSystems.
from_bond_order_system(file_name, **kwargs)	Implement BondOrderSystem.from that converts from this format to BondOrderSystem.
from_labeled_system(file_name, **kwargs)	Implement LabeledSystem.from that converts from this format to LabeledSystem.
from_multi_systems(directory, **kwargs)	Implement MultiSystems.from that converts from this format to MultiSystems.
from_system(file_name, **kwargs)	Implement System.from that converts from this format to System.
get_formats()	Get all registered formats.
get_from_methods()	Get all registered from methods.
get_to_methods()	Get all registered to methods.
mix_system(*system, type_map, **kwargs)	Mix the systems into mixed_type ones according to the unified given type_map.
post(func_name)	Register a post function for from method.
register(key)	Register a format plugin.
register_from(key)	Register a from method if the target method name is not default.
register_to(key)	Register a to method if the target method name is not default.
to_bond_order_system(data, rdkit_mol, *args, ...)	Implement BondOrderSystem.to that converts from BondOrderSystem to this format.
to_labeled_system(data, *args, **kwargs)	Implement LabeledSystem.to that converts from LabeledSystem to this format.
to_multi_systems(formulas, directory, **kwargs)	Implement MultiSystems.to that converts from MultiSystems to this format.
to_system(data[, fname, frame_idx])	Generate input files for semi-emperical calculation in sqm software.

to_system(*data*, *fname=None*, *frame_idx=0*, ***kwargs*)

Generate input files for semi-emperical calculation in sqm software.

Parameters

data
[dict] system data

fname
[str] output file name

frame_idx
[int, default=0] index of frame to write

****kwargs**
[dict] other parameters

Other Parameters

****kwargs**
[dict]

valid parameters are:

qm_theory
[str, default=dftb3] level of theory. Options includes AM1, RM1, MNDO,

PM3-PDDG, MNDO-PDDG, PM3-CARB1, MNDO/d, AM1/d, PM6, DFTB2, DFTB3

charge
[int, default=0] total charge in electron units

maxcyc
[int, default=0] maximum number of minimization cycles to allow. 0 represents a single-point calculation

mult
[int, default=1] multiplicity. Only 1 is allowed.

class dpdata.plugins.amber.**SQMMinimizer**(*maxcyc=1000*, **args*, ***kwargs*)

Bases: *Minimizer*

SQM minimizer.

Parameters

maxcyc
[int, default=1000] maximun cycle to minimize

Methods

get_minimizer(key)	Get a minimizer plugin.
get_minimizers()	Get all minimizer plugins.
minimize(data)	Minimize the geometry.
register(key)	Register a minimizer plugin.

minimize(*data: dict*) → dict

Minimize the geometry.

Parameters

data
[dict] data with coordinates and atom types

Returns

dict
labeled data with minimized coordinates, energies, and forces

class dpdata.plugins.amber.**SQMOutFormat**

Bases: *Format*

Methods

MultiModes()	File mode for MultiSystems.
from_bond_order_system(file_name, **kwargs)	Implement BondOrderSystem.from that converts from this format to BondOrderSystem.
from_labeled_system(fname, **kwargs)	Read from ambertools sqm.out.
from_multi_systems(directory, **kwargs)	Implement MultiSystems.from that converts from this format to MultiSystems.
from_system(fname, **kwargs)	Read from ambertools sqm.out.
get_formats()	Get all registered formats.
get_from_methods()	Get all registered from methods.
get_to_methods()	Get all registered to methods.
mix_system(*system, type_map, **kwargs)	Mix the systems into mixed_type ones according to the unified given type_map.
post(func_name)	Register a post function for from method.
register(key)	Register a format plugin.
register_from(key)	Register a from method if the target method name is not default.
register_to(key)	Register a to method if the target method name is not default.
to_bond_order_system(data, rdkit_mol, *args, ...)	Implement BondOrderSystem.to that converts from BondOrderSystem to this format.
to_labeled_system(data, *args, **kwargs)	Implement LabeledSystem.to that converts from LabeledSystem to this format.
to_multi_systems(formulas, directory, **kwargs)	Implement MultiSystems.to that converts from MultiSystems to this format.
to_system(data, *args, **kwargs)	Implement System.to that converts from System to this format.

from_labeled_system(*fname*, ***kwargs*)

Read from ambertools sqm.out.

from_system(*fname*, ***kwargs*)

Read from ambertools sqm.out.

dpdata.plugins.ase module

class dpdata.plugins.ase.**ASEDriver**(*calculator: ase.calculators.calculator.Calculator*)

Bases: *Driver*

ASE Driver.

Parameters

calculator
[ase.calculators.calculator.Calculato] ASE calculator

Attributes

ase_calculator
Returns an ase calculator based on this driver.

Methods

get_driver(key)	Get a driver plugin.
get_drivers()	Get all driver plugins.
label(data)	Label a system data.
register(key)	Register a driver plugin.

label(*data: dict*) → dict

Label a system data. Returns new data with energy, forces, and virials.

Parameters

data
[dict] data with coordinates and atom types

Returns

dict
labeled data with energies and forces

class dpdata.plugins.ase.**ASEMinimizer**(*driver: Driver*, *optimizer: Type[Optimizer] | None = None*, *fmax: float = 0.005*, *max_steps: int | None = None*, *optimizer_kwargs: dict = {}*)

Bases: *Minimizer*

ASE minimizer.

Parameters

driver
[Driver] dpdata driver

optimizer
[type, optional] ase optimizer class

fmax
[float, optional, default=5e-3] force convergence criterion

max_steps
[int, optional] max steps to optimize

optimizer_kwargs
[dict, optional] other parameters for optimizer

Methods

get_minimizer(key)	Get a minimizer plugin.
get_minimizers()	Get all minimizer plugins.
minimize(data)	Minimize the geometry.
register(key)	Register a minimizer plugin.

minimize(*data: dict*) → dict

Minimize the geometry.

Parameters

data
[dict] data with coordinates and atom types

Returns

dict
labeled data with minimized coordinates, energies, and forces

class dpdata.plugins.ase.**ASEStructureFormat**

Bases: *Format*

Format for the Atomic Simulation Environment (ase).

ASE supports parsing a few dozen of data formats. As described in i the documentation, many of these formats can be determined automatically. Use the *ase_fmt* keyword argument to supply the format if automatic detection fails.

Methods

MultiModes()	File mode for MultiSystems.
from_bond_order_system(file_name, **kwargs)	Implement BondOrderSystem.from that converts from this format to BondOrderSystem.
from_labeled_system(atoms, **kwargs)	Convert ase.Atoms to a LabeledSystem.
from_multi_systems(file_name[, begin, end, ...])	Convert a ASE supported file to ASE Atoms.
from_system(atoms, **kwargs)	Convert ase.Atoms to a System.
get_formats()	Get all registered formats.
get_from_methods()	Get all registered from methods.
get_to_methods()	Get all registered to methods.
mix_system(*system, type_map, **kwargs)	Mix the systems into mixed_type ones according to the unified given type_map.
post(func_name)	Register a post function for from method.
register(key)	Register a format plugin.
register_from(key)	Register a from method if the target method name is not default.
register_to(key)	Register a to method if the target method name is not default.
to_bond_order_system(data, rdkit_mol, *args, ...)	Implement BondOrderSystem.to that converts from BondOrderSystem to this format.
to_labeled_system(data, *args, **kwargs)	Convert System to ASE Atoms object.
to_multi_systems(formulas, directory, **kwargs)	Implement MultiSystems.to that converts from MultiSystems to this format.
to_system(data, **kwargs)	Convert System to ASE Atom obj.

from_labeled_system(*atoms: ase.Atoms*, ***kwargs*) → dict

Convert ase.Atoms to a LabeledSystem. Energies and forces are calculated by the calculator.

Parameters

atoms
[ase.Atoms] an ASE Atoms, containing a structure

****kwargs**
[dict] other parameters

Returns

dict
data dict

Raises

RuntimeError
ASE will raise RuntimeError if the atoms does not have a calculator

`from_multi_systems`(*file_name: str*, *begin: int | None = None*, *end: int | None = None*, *step: int | None = None*, *ase_fmt: str | None = None*, ***kwargs*) → ase.Atoms

Convert a ASE supported file to ASE Atoms.

It will finally be converted to MultiSystems.

Parameters

file_name
[str] path to file

begin
[int, optional] begin frame index

end
[int, optional] end frame index

step
[int, optional] frame index step

ase_fmt
[str, optional] ASE format. See the ASE documentation about supported formats

****kwargs**
[dict] other parameters

Yields

ase.Atoms
ASE atoms in the file

`from_system`(*atoms: ase.Atoms*, ***kwargs*) → dict

Convert ase.Atoms to a System.

Parameters

atoms
[ase.Atoms] an ASE Atoms, containing a structure

****kwargs**
[dict] other parameters

Returns

dict
data dict

`to_labeled_system`(*data*, **args*, ***kwargs*)

Convert System to ASE Atoms object.

`to_system`(*data*, ***kwargs*)

Convert System to ASE Atom obj.

class dpdata.plugins.ase.**ASETrajFormat**

Bases: *Format*

Format for the ASE's trajectory format <https://wiki.fysik.dtu.dk/ase/ase/io/trajectory.html#module-ase.io.trajectory>`_ (ase).' a `traj' contains a sequence of frames, each of which is an `Atoms' object.

Methods

MultiModes()	File mode for MultiSystems.
from_bond_order_system(file_name, **kwargs)	Implement BondOrderSystem.from that converts from this format to BondOrderSystem.
from_labeled_system(file_name[, begin, end, ...])	Read ASE's trajectory file to *System* of multiple frames.
from_multi_systems(directory, **kwargs)	Implement MultiSystems.from that converts from this format to MultiSystems.
from_system(file_name[, begin, end, step])	Read ASE's trajectory file to *System* of multiple frames.
get_formats()	Get all registered formats.
get_from_methods()	Get all registered from methods.
get_to_methods()	Get all registered to methods.
mix_system(*system, type_map, **kwargs)	Mix the systems into mixed_type ones according to the unified given type_map.
post(func_name)	Register a post function for from method.
register(key)	Register a format plugin.
register_from(key)	Register a from method if the target method name is not default.
register_to(key)	Register a to method if the target method name is not default.
to_bond_order_system(data, rdkit_mol, *args, ...)	Implement BondOrderSystem.to that converts from BondOrderSystem to this format.
to_labeled_system(data, *args, **kwargs)	Implement LabeledSystem.to that converts from LabeledSystem to this format.
to_multi_systems(formulas, directory, **kwargs)	Implement MultiSystems.to that converts from MultiSystems to this format.
to_system(data, *args, **kwargs)	Implement System.to that converts from System to this format.

from_labeled_system(*file_name: str*, *begin: int | None = 0*, *end: int | None = None*, *step: int | None = 1*, ***kwargs*) → dict

Read ASE's trajectory file to *System* of multiple frames.

Parameters

file_name
[str] ASE's trajectory file

begin
[int, optional] begin frame index

end
[int, optional] end frame index

step
[int, optional] frame index step

****kwargs**
[dict] other parameters

Returns

dict_frames: dict
a dictionary containing data of multiple frames

from_system(*file_name: str*, *begin: int | None = 0*, *end: int | None = None*, *step: int | None = 1*, ***kwargs*) → dict

Read ASE's trajectory file to *System* of multiple frames.

Parameters

file_name
[str] ASE's trajectory file

begin
[int, optional] begin frame index

end
[int, optional] end frame index

step
[int, optional] frame index step

****kwargs**
[dict] other parameters

Returns

dict_frames: dict
a dictionary containing data of multiple frames

dpdata.plugins.cp2k module

class dpdata.plugins.cp2k.**CP2KAIMDOutputFormat**

Bases: *Format*

Methods

MultiModes()	File mode for MultiSystems.
from_bond_order_system(file_name, **kwargs)	Implement BondOrderSystem.from that converts from this format to BondOrderSystem.
from_labeled_system(file_name[, restart])	Implement LabeledSystem.from that converts from this format to LabeledSystem.
from_multi_systems(directory, **kwargs)	Implement MultiSystems.from that converts from this format to MultiSystems.
from_system(file_name, **kwargs)	Implement System.from that converts from this format to System.
get_formats()	Get all registered formats.
get_from_methods()	Get all registered from methods.
get_to_methods()	Get all registered to methods.
mix_system(*system, type_map, **kwargs)	Mix the systems into mixed_type ones according to the unified given type_map.
post(func_name)	Register a post function for from method.
register(key)	Register a format plugin.
register_from(key)	Register a from method if the target method name is not default.
register_to(key)	Register a to method if the target method name is not default.
to_bond_order_system(data, rdkit_mol, *args, ...)	Implement BondOrderSystem.to that converts from BondOrderSystem to this format.
to_labeled_system(data, *args, **kwargs)	Implement LabeledSystem.to that converts from LabeledSystem to this format.
to_multi_systems(formulas, directory, **kwargs)	Implement MultiSystems.to that converts from MultiSystems to this format.
to_system(data, *args, **kwargs)	Implement System.to that converts from System to this format.

from_labeled_system(*file_name*, *restart=False*, ***kwargs*)

Implement LabeledSystem.from that converts from this format to LabeledSystem.

Parameters

file_name
[str] file name, i.e. the first argument

****kwargs**
[dict] keyword arguments that will be passed from the method

Returns

data
[dict] system data, whose keys are defined in LabeledSystem.DTYPES

class dpdata.plugins.cp2k.**CP2KOutputFormat**

Bases: *Format*

Methods

MultiModes()	File mode for MultiSystems.
from_bond_order_system(file_name, **kwargs)	Implement BondOrderSystem.from that converts from this format to BondOrderSystem.
from_labeled_system(file_name[, restart])	Implement LabeledSystem.from that converts from this format to LabeledSystem.
from_multi_systems(directory, **kwargs)	Implement MultiSystems.from that converts from this format to MultiSystems.
from_system(file_name, **kwargs)	Implement System.from that converts from this format to System.
get_formats()	Get all registered formats.
get_from_methods()	Get all registered from methods.
get_to_methods()	Get all registered to methods.
mix_system(*system, type_map, **kwargs)	Mix the systems into mixed_type ones according to the unified given type_map.
post(func_name)	Register a post function for from method.
register(key)	Register a format plugin.
register_from(key)	Register a from method if the target method name is not default.
register_to(key)	Register a to method if the target method name is not default.
to_bond_order_system(data, rdkit_mol, *args, ...)	Implement BondOrderSystem.to that converts from BondOrderSystem to this format.
to_labeled_system(data, *args, **kwargs)	Implement LabeledSystem.to that converts from LabeledSystem to this format.
to_multi_systems(formulas, directory, **kwargs)	Implement MultiSystems.to that converts from MultiSystems to this format.
to_system(data, *args, **kwargs)	Implement System.to that converts from System to this format.

from_labeled_system(*file_name*, *restart=False*, ***kwargs*)

Implement LabeledSystem.from that converts from this format to LabeledSystem.

Parameters

file_name
[str] file name, i.e. the first argument

****kwargs**
[dict] keyword arguments that will be passed from the method

Returns

data
[dict] system data, whose keys are defined in LabeledSystem.DTYPES

dpdata.plugins.deepmd module

class dpdata.plugins.deepmd.**DPDriver**(*dp: str*)

Bases: *Driver*

DeePMD-kit driver.

Parameters

dp
[deepmd.DeepPot or str] The deepmd-kit potential class or the filename of the model.

Examples

```
>>> DPDriver("frozen_model.pb")
```

Attributes

ase_calculator
Returns an ase calculator based on this driver.

Methods

get_driver(key)	Get a driver plugin.
get_drivers()	Get all driver plugins.
label(data)	Label a system data by deepmd-kit.
register(key)	Register a driver plugin.

label(*data: dict*) → dict

Label a system data by deepmd-kit. Returns new data with energy, forces, and virials.

Parameters

data
[dict] data with coordinates and atom types

Returns

dict
labeled data with energies and forces

class dpdata.plugins.deepmd.**DeePMDCompFormat**

Bases: *Format*

Methods

MultiModes()	File mode for MultiSystems.
from_bond_order_system(file_name, **kwargs)	Implement BondOrderSystem.from that converts from this format to BondOrderSystem.
from_labeled_system(file_name[, type_map])	Implement LabeledSystem.from that converts from this format to LabeledSystem.
from_multi_systems(directory, **kwargs)	Implement MultiSystems.from that converts from this format to MultiSystems.
from_system(file_name[, type_map])	Implement System.from that converts from this format to System.
get_formats()	Get all registered formats.
get_from_methods()	Get all registered from methods.
get_to_methods()	Get all registered to methods.
mix_system(*system, type_map, **kwargs)	Mix the systems into mixed_type ones according to the unified given type_map.
post(func_name)	Register a post function for from method.
register(key)	Register a format plugin.
register_from(key)	Register a from method if the target method name is not default.
register_to(key)	Register a to method if the target method name is not default.
to_bond_order_system(data, rdkit_mol, *args, ...)	Implement BondOrderSystem.to that converts from BondOrderSystem to this format.
to_labeled_system(data, *args, **kwargs)	Implement LabeledSystem.to that converts from LabeledSystem to this format.
to_multi_systems(formulas, directory, **kwargs)	Implement MultiSystems.to that converts from MultiSystems to this format.
to_system(data, file_name[, set_size, prec])	Dump the system in deepmd compressed format (numpy binary) to *folder*.

MultiMode = 1

from_labeled_system(*file_name*, *type_map=None*, ***kwargs*)

Implement LabeledSystem.from that converts from this format to LabeledSystem.

Parameters

file_name
[str] file name, i.e. the first argument

****kwargs**
[dict] keyword arguments that will be passed from the method

Returns

data
[dict] system data, whose keys are defined in LabeledSystem.DTYPES

from_system(*file_name*, *type_map=None*, ***kwargs*)

Implement System.from that converts from this format to System.

Parameters

file_name
[str] file name, i.e. the first argument

****kwargs**
[dict] keyword arguments that will be passed from the method

Returns

data
[dict] system data, whose keys are defined in System.DTYPES

to_system(*data*, *file_name*, *set_size=5000*, *prec=<class 'numpy.float64'>*, ***kwargs*)

Dump the system in deepmd compressed format (numpy binary) to *folder*.

The frames are firstly split to sets, then dumped to seperated subfolders named as *folder/set.000*, *folder/set.001*, ...

Each set contains *set_size* frames. The last set may have less frames than *set_size*.

Parameters

data
[dict] System data

file_name
[str] The output folder

set_size
[int] The size of each set.

prec
[{numpy.float32, numpy.float64}] The floating point precision of the compressed data

****kwargs**
[dict] other parameters

class dpdata.plugins.deepmd.**DeePMDHDF5Format**

Bases: *Format*

HDF5 format for DeePMD-kit.

Examples

Dump a MultiSystems to a HDF5 file:

```
>>> import dpdata
>>> dpdata.MultiSystems().from_deepmd_npy("data").to_deepmd_hdf5("data.hdf5")
```

Methods

MultiModes()	File mode for MultiSystems.
from_bond_order_system(file_name, **kwargs)	Implement BondOrderSystem.from that converts from this format to BondOrderSystem.
from_labeled_system(file_name[, type_map])	Convert HDF5 file to LabeledSystem data.
from_multi_systems(directory, **kwargs)	Generate HDF5 groups from a HDF5 file, which will be passed to *from_system*.
from_system(file_name[, type_map])	Convert HDF5 file to System data.
get_formats()	Get all registered formats.
get_from_methods()	Get all registered from methods.
get_to_methods()	Get all registered to methods.
mix_system(*system, type_map, **kwargs)	Mix the systems into mixed_type ones according to the unified given type_map.
post(func_name)	Register a post function for from method.
register(key)	Register a format plugin.
register_from(key)	Register a from method if the target method name is not default.
register_to(key)	Register a to method if the target method name is not default.
to_bond_order_system(data, rdkit_mol, *args, ...)	Implement BondOrderSystem.to that converts from BondOrderSystem to this format.
to_labeled_system(data, *args, **kwargs)	Implement LabeledSystem.to that converts from LabeledSystem to this format.
to_multi_systems(formulas, directory, **kwargs)	Generate HDF5 groups, which will be passed to *to_system*.
to_system(data, file_name[, set_size, comp_prec])	Convert System data to HDF5 file.

from_labeled_system(*file_name: str | Group | File*, *type_map: list[str] | None = None*, ***kwargs*) → dict

Convert HDF5 file to LabeledSystem data.

Parameters

file_name
[str or h5py.Group or h5py.File] file name of the HDF5 file or HDF5 object. If it is a string, hashtag is used to split path to the HDF5 file and the HDF5 group

type_map
[dict[str]] type map

****kwargs**
[dict] other parameters

Returns

dict
LabeledSystem data

Raises

TypeError
file_name is not str or h5py.Group or h5py.File

from_multi_systems(*directory: str*, ***kwargs*) → Group

Generate HDF5 groups from a HDF5 file, which will be passed to *from_system*.

Parameters

directory
[str] HDF5 file name

****kwargs**
[dict] other parameters

Yields

h5py.Group
a HDF5 group in the HDF5 file

`from_system`(*file_name: str | Group | File*, *type_map: list[str] | None = None*, ***kwargs*) → dict

Convert HDF5 file to System data.

Parameters

file_name
[str or h5py.Group or h5py.File] file name of the HDF5 file or HDF5 object. If it is a string, hashtag is used to split path to the HDF5 file and the HDF5 group

type_map
[dict[str]] type map

****kwargs**
[dict] other parameters

Returns

dict
System data

Raises

TypeError
file_name is not str or h5py.Group or h5py.File

`to_multi_systems`(*formulas: list[str]*, *directory: str*, ***kwargs*) → Group

Generate HDF5 groups, which will be passed to *to_system*.

Parameters

formulas
[list[str]] formulas of MultiSystems

directory
[str] HDF5 file name

****kwargs**
[dict] other parameters

Yields

h5py.Group
a HDF5 group with the name of formula

`to_system`(*data: dict*, *file_name: str | ~h5py._hl.group.Group | ~h5py._hl.files.File*, *set_size: int = 5000*, *comp_prec: ~numpy.dtype = <class 'numpy.float64'>*, ***kwargs*)

Convert System data to HDF5 file.

Parameters

data
[dict] data dict

file_name
[str or h5py.Group or h5py.File] file name of the HDF5 file or HDF5 object. If it is a string, hashtag is used to split path to the HDF5 file and the HDF5 group

set_size
[int, default=5000] set size

comp_prec
[np.dtype] data precision

****kwargs**
[dict] other parameters

class dpdata.plugins.deepmd.**DeePMDMixedFormat**

Bases: *Format*

Mixed type numpy format for DeePMD-kit. Under this format, systems with the same number of atoms but different formula can be put together for a larger system, especially when the frame numbers in systems are sparse. This also helps to mixture the type information together for model training with type embedding network.

Examples

Dump a MultiSystems into a mixed type numpy directory:

```
>>> import dpdata
>>> dpdata.MultiSystems(*systems).to_deepmd_npy_mixed("mixed_dir")
```

Load a mixed type data into a MultiSystems:

```
>>> import dpdata
>>> dpdata.MultiSystems().load_systems_from_file("mixed_dir", fmt="deepmd/npy/mixed
↪")
```

Methods

MultiModes()	File mode for MultiSystems.
from_bond_order_system(file_name, **kwargs)	Implement BondOrderSystem.from that converts from this format to BondOrderSystem.
from_labeled_system(file_name, **kwargs)	Implement LabeledSystem.from that converts from this format to LabeledSystem.
from_multi_systems(directory, **kwargs)	Implement MultiSystems.from that converts from this format to MultiSystems.
from_system(file_name, **kwargs)	Implement System.from that converts from this format to System.
get_formats()	Get all registered formats.
get_from_methods()	Get all registered from methods.
get_to_methods()	Get all registered to methods.
mix_system(*system, type_map, **kwargs)	Mix the systems into mixed_type ones according to the unified given type_map.
post(func_name)	Register a post function for from method.
register(key)	Register a format plugin.
register_from(key)	Register a from method if the target method name is not default.
register_to(key)	Register a to method if the target method name is not default.
to_bond_order_system(data, rdkit_mol, *args, ...)	Implement BondOrderSystem.to that converts from BondOrderSystem to this format.
to_labeled_system(data, *args, **kwargs)	Implement LabeledSystem.to that converts from LabeledSystem to this format.
to_multi_systems(formulas, directory, **kwargs)	Implement MultiSystems.to that converts from MultiSystems to this format.
to_system(data, file_name[, set_size, prec])	Dump the system in deepmd mixed type format (numpy binary) to *folder*.

from_labeled_system_mix
from_system_mix

MultiMode = 1

from_labeled_system_mix(*file_name*, *type_map=None*, ***kwargs*)

from_multi_systems(*directory*, ***kwargs*)

Implement MultiSystems.from that converts from this format to MultiSystems.

By default, this method follows MultiMode to implement the conversion.

Parameters

directory
[str] directory of system

****kwargs**
[dict] keyword arguments that will be passed from the method

Returns

filenames: list[str]
list of filenames

from_system_mix(*file_name*, *type_map=None*, ***kwargs*)

mix_system(**system*, *type_map*, ***kwargs*)

Mix the systems into mixed_type ones according to the unified given type_map.

Parameters

***system**
[System] The systems to mix

type_map
[list of str] Maps atom type to name

****kwargs**
[dict] other parameters

Returns

mixed_systems: dict
dict of mixed system with key 'atom_numbs'

to_system(*data*, *file_name*, *set_size: int = 2000*, *prec=<class 'numpy.float64'>*, ***kwargs*)

Dump the system in deepmd mixed type format (numpy binary) to *folder*.

The frames were already split to different systems, so these frames can be dumped to one single subfolders
named as *folder/set.000*, containing less than *set_size* frames.

Parameters

data
[dict] System data

file_name
[str] The output folder

set_size
[int, default=2000] set size

prec
[{numpy.float32, numpy.float64}] The floating point precision of the compressed data

****kwargs**
[dict] other parameters

class dpdata.plugins.deepmd.**DeePMDRawFormat**

Bases: *Format*

Methods

MultiModes()	File mode for MultiSystems.
from_bond_order_system(file_name, **kwargs)	Implement BondOrderSystem.from that converts from this format to BondOrderSystem.
from_labeled_system(file_name[, type_map])	Implement LabeledSystem.from that converts from this format to LabeledSystem.
from_multi_systems(directory, **kwargs)	Implement MultiSystems.from that converts from this format to MultiSystems.
from_system(file_name[, type_map])	Implement System.from that converts from this format to System.
get_formats()	Get all registered formats.
get_from_methods()	Get all registered from methods.
get_to_methods()	Get all registered to methods.
mix_system(*system, type_map, **kwargs)	Mix the systems into mixed_type ones according to the unified given type_map.
post(func_name)	Register a post function for from method.
register(key)	Register a format plugin.
register_from(key)	Register a from method if the target method name is not default.
register_to(key)	Register a to method if the target method name is not default.
to_bond_order_system(data, rdkit_mol, *args, ...)	Implement BondOrderSystem.to that converts from BondOrderSystem to this format.
to_labeled_system(data, *args, **kwargs)	Implement LabeledSystem.to that converts from LabeledSystem to this format.
to_multi_systems(formulas, directory, **kwargs)	Implement MultiSystems.to that converts from MultiSystems to this format.
to_system(data, file_name, **kwargs)	Dump the system in deepmd raw format to directory *file_name*.

MultiMode = 1

from_labeled_system(*file_name*, *type_map=None*, ***kwargs*)

Implement LabeledSystem.from that converts from this format to LabeledSystem.

Parameters

file_name
[str] file name, i.e. the first argument

****kwargs**
[dict] keyword arguments that will be passed from the method

Returns

data
[dict] system data, whose keys are defined in LabeledSystem.DTYPES

from_system(*file_name*, *type_map=None*, ***kwargs*)

Implement System.from that converts from this format to System.

Parameters

file_name
[str] file name, i.e. the first argument

****kwargs**
[dict] keyword arguments that will be passed from the method

Returns

data
[dict] system data, whose keys are defined in System.DTYPES

`to_system`(*data*, *file_name*, ***kwargs*)

Dump the system in deepmd raw format to directory *file_name*.

dpdata.plugins.dftbplus module

`class` dpdata.plugins.dftbplus.**DFTBplusFormat**

Bases: *Format*

The DFTBplusFormat class handles files in the DFTB+ format.

This class provides a method to read DFTB+ files from a labeled system and returns a dictionary containing various properties of the system.For more information, please refer to the official documentation at the following URL: https://dftbplus.org/documentation

Attributes

None

Methods

from_labeled_system(file_paths, **kwargs): Reads system information from files.

`from_labeled_system`(*file_paths*, ***kwargs*)

Reads system information from the given DFTB+ file paths.

Parameters

file_paths
[tuple] A tuple containing the input and output file paths. - Input file (file_in): Contains information about symbols and coord. - Output file (file_out): Contains information about energy and force.

****kwargs**
[dict] other parameters

dpdata.plugins.fhi_aims module

`class` dpdata.plugins.fhi_aims.**FhiMDFormat**

Bases: *Format*

Methods

MultiModes()	File mode for MultiSystems.
from_bond_order_system(file_name, **kwargs)	Implement BondOrderSystem.from that converts from this format to BondOrderSystem.
from_labeled_system(file_name[, md, begin, ...])	Implement LabeledSystem.from that converts from this format to LabeledSystem.
from_multi_systems(directory, **kwargs)	Implement MultiSystems.from that converts from this format to MultiSystems.
from_system(file_name, **kwargs)	Implement System.from that converts from this format to System.
get_formats()	Get all registered formats.
get_from_methods()	Get all registered from methods.
get_to_methods()	Get all registered to methods.
mix_system(*system, type_map, **kwargs)	Mix the systems into mixed_type ones according to the unified given type_map.
post(func_name)	Register a post function for from method.
register(key)	Register a format plugin.
register_from(key)	Register a from method if the target method name is not default.
register_to(key)	Register a to method if the target method name is not default.
to_bond_order_system(data, rdkit_mol, *args, ...)	Implement BondOrderSystem.to that converts from BondOrderSystem to this format.
to_labeled_system(data, *args, **kwargs)	Implement LabeledSystem.to that converts from LabeledSystem to this format.
to_multi_systems(formulas, directory, **kwargs)	Implement MultiSystems.to that converts from MultiSystems to this format.
to_system(data, *args, **kwargs)	Implement System.to that converts from System to this format.

from_labeled_system(*file_name*, *md=True*, *begin=0*, *step=1*, *convergence_check=True*, ***kwargs*)

Implement LabeledSystem.from that converts from this format to LabeledSystem.

Parameters

file_name
[str] file name, i.e. the first argument

****kwargs**
[dict] keyword arguments that will be passed from the method

Returns

data
[dict] system data, whose keys are defined in LabeledSystem.DTYPES

class dpdata.plugins.fhi_aims.**FhiSCFFormat**

Bases: *Format*

Methods

MultiModes()	File mode for MultiSystems.
from_bond_order_system(file_name, **kwargs)	Implement BondOrderSystem.from that converts from this format to BondOrderSystem.
from_labeled_system(file_name, **kwargs)	Implement LabeledSystem.from that converts from this format to LabeledSystem.
from_multi_systems(directory, **kwargs)	Implement MultiSystems.from that converts from this format to MultiSystems.
from_system(file_name, **kwargs)	Implement System.from that converts from this format to System.
get_formats()	Get all registered formats.
get_from_methods()	Get all registered from methods.
get_to_methods()	Get all registered to methods.
mix_system(*system, type_map, **kwargs)	Mix the systems into mixed_type ones according to the unified given type_map.
post(func_name)	Register a post function for from method.
register(key)	Register a format plugin.
register_from(key)	Register a from method if the target method name is not default.
register_to(key)	Register a to method if the target method name is not default.
to_bond_order_system(data, rdkit_mol, *args, ...)	Implement BondOrderSystem.to that converts from BondOrderSystem to this format.
to_labeled_system(data, *args, **kwargs)	Implement LabeledSystem.to that converts from LabeledSystem to this format.
to_multi_systems(formulas, directory, **kwargs)	Implement MultiSystems.to that converts from MultiSystems to this format.
to_system(data, *args, **kwargs)	Implement System.to that converts from System to this format.

from_labeled_system(*file_name*, ***kwargs*)

Implement LabeledSystem.from that converts from this format to LabeledSystem.

Parameters

file_name
[str] file name, i.e. the first argument

****kwargs**
[dict] keyword arguments that will be passed from the method

Returns

data
[dict] system data, whose keys are defined in LabeledSystem.DTYPES

dpdata.plugins.gaussian module

class dpdata.plugins.gaussian.**GaussiaGJFFormat**

Bases: *Format*

Gaussian input file.

Methods

MultiModes()	File mode for MultiSystems.
from_bond_order_system(file_name, **kwargs)	Implement BondOrderSystem.from that converts from this format to BondOrderSystem.
from_labeled_system(file_name, **kwargs)	Implement LabeledSystem.from that converts from this format to LabeledSystem.
from_multi_systems(directory, **kwargs)	Implement MultiSystems.from that converts from this format to MultiSystems.
from_system(file_name, **kwargs)	Read Gaussian input file.
get_formats()	Get all registered formats.
get_from_methods()	Get all registered from methods.
get_to_methods()	Get all registered to methods.
mix_system(*system, type_map, **kwargs)	Mix the systems into mixed_type ones according to the unified given type_map.
post(func_name)	Register a post function for from method.
register(key)	Register a format plugin.
register_from(key)	Register a from method if the target method name is not default.
register_to(key)	Register a to method if the target method name is not default.
to_bond_order_system(data, rdkit_mol, *args, ...)	Implement BondOrderSystem.to that converts from BondOrderSystem to this format.
to_labeled_system(data, *args, **kwargs)	Implement LabeledSystem.to that converts from LabeledSystem to this format.
to_multi_systems(formulas, directory, **kwargs)	Implement MultiSystems.to that converts from MultiSystems to this format.
to_system(data, file_name, **kwargs)	Generate Gaussian input file.

from_system(*file_name: str*, ***kwargs*)

Read Gaussian input file.

Parameters

file_name
[str] file name

****kwargs**
[dict] keyword arguments

to_system(*data: dict*, *file_name: str*, ***kwargs*)

Generate Gaussian input file.

Parameters

data
[dict] system data

file_name
[str] file name

****kwargs**
[dict] Other parameters to make input files. See *dpdata.gaussian.gjf.make_gaussian_input()*

class dpdata.plugins.gaussian.**GaussianDriver**(*gaussian_exec: str = 'g16'*, ***kwargs: dict*)

Bases: *Driver*

Gaussian driver.

Note that "force" keyword must be added. If the number of atoms is large, "Geom=PrintInputOrient" should be added.

Parameters

gaussian_exec
[str, default=g16] path to gaussian program

****kwargs**
[dict] other arguments to make input files. See *dpdata.gaussian.gjf.make_gaussian_input()*

Examples

Use B3LYP method to calculate potential energy of a methane molecule:

```
>>> labeled_system = system.predict(keywords="force b3lyp/6-31g**", driver="gaussian
↪")
>>> labeled_system['energies'][0]
-1102.714590995794
```

Attributes

ase_calculator
Returns an ase calculator based on this driver.

Methods

get_driver(key)	Get a driver plugin.
get_drivers()	Get all driver plugins.
label(data)	Label a system data.
register(key)	Register a driver plugin.

label(*data: dict*) → dict

Label a system data. Returns new data with energy, forces, and virials.

Parameters

data
[dict] data with coordinates and atom types

Returns

dict
labeled data with energies and forces

class dpdata.plugins.gaussian.**GaussianLogFormat**

Bases: *Format*

Methods

MultiModes()	File mode for MultiSystems.
from_bond_order_system(file_name, **kwargs)	Implement BondOrderSystem.from that converts from this format to BondOrderSystem.
from_labeled_system(file_name[, md])	Implement LabeledSystem.from that converts from this format to LabeledSystem.
from_multi_systems(directory, **kwargs)	Implement MultiSystems.from that converts from this format to MultiSystems.
from_system(file_name, **kwargs)	Implement System.from that converts from this format to System.
get_formats()	Get all registered formats.
get_from_methods()	Get all registered from methods.
get_to_methods()	Get all registered to methods.
mix_system(*system, type_map, **kwargs)	Mix the systems into mixed_type ones according to the unified given type_map.
post(func_name)	Register a post function for from method.
register(key)	Register a format plugin.
register_from(key)	Register a from method if the target method name is not default.
register_to(key)	Register a to method if the target method name is not default.
to_bond_order_system(data, rdkit_mol, *args, ...)	Implement BondOrderSystem.to that converts from BondOrderSystem to this format.
to_labeled_system(data, *args, **kwargs)	Implement LabeledSystem.to that converts from LabeledSystem to this format.
to_multi_systems(formulas, directory, **kwargs)	Implement MultiSystems.to that converts from MultiSystems to this format.
to_system(data, *args, **kwargs)	Implement System.to that converts from System to this format.

from_labeled_system(*file_name*, *md=False*, ***kwargs*)

Implement LabeledSystem.from that converts from this format to LabeledSystem.

Parameters

file_name
[str] file name, i.e. the first argument

****kwargs**
[dict] keyword arguments that will be passed from the method

Returns

data
[dict] system data, whose keys are defined in LabeledSystem.DTYPES

class dpdata.plugins.gaussian.**GaussianMDFormat**

Bases: *Format*

Methods

MultiModes()	File mode for MultiSystems.
from_bond_order_system(file_name, **kwargs)	Implement BondOrderSystem.from that converts from this format to BondOrderSystem.
from_labeled_system(file_name, **kwargs)	Implement LabeledSystem.from that converts from this format to LabeledSystem.
from_multi_systems(directory, **kwargs)	Implement MultiSystems.from that converts from this format to MultiSystems.
from_system(file_name, **kwargs)	Implement System.from that converts from this format to System.
get_formats()	Get all registered formats.
get_from_methods()	Get all registered from methods.
get_to_methods()	Get all registered to methods.
mix_system(*system, type_map, **kwargs)	Mix the systems into mixed_type ones according to the unified given type_map.
post(func_name)	Register a post function for from method.
register(key)	Register a format plugin.
register_from(key)	Register a from method if the target method name is not default.
register_to(key)	Register a to method if the target method name is not default.
to_bond_order_system(data, rdkit_mol, *args, ...)	Implement BondOrderSystem.to that converts from BondOrderSystem to this format.
to_labeled_system(data, *args, **kwargs)	Implement LabeledSystem.to that converts from LabeledSystem to this format.
to_multi_systems(formulas, directory, **kwargs)	Implement MultiSystems.to that converts from MultiSystems to this format.
to_system(data, *args, **kwargs)	Implement System.to that converts from System to this format.

from_labeled_system(*file_name*, ***kwargs*)

Implement LabeledSystem.from that converts from this format to LabeledSystem.

Parameters

file_name
[str] file name, i.e. the first argument

****kwargs**
[dict] keyword arguments that will be passed from the method

Returns

data
[dict] system data, whose keys are defined in LabeledSystem.DTYPES

dpdata.plugins.gromacs module

class dpdata.plugins.gromacs.**GromacsGroFormat**

Bases: *Format*

Methods

MultiModes()	File mode for MultiSystems.
from_bond_order_system(file_name, **kwargs)	Implement BondOrderSystem.from that converts from this format to BondOrderSystem.
from_labeled_system(file_name, **kwargs)	Implement LabeledSystem.from that converts from this format to LabeledSystem.
from_multi_systems(directory, **kwargs)	Implement MultiSystems.from that converts from this format to MultiSystems.
from_system(file_name[, format_atom_name])	Load gromacs .gro file.
get_formats()	Get all registered formats.
get_from_methods()	Get all registered from methods.
get_to_methods()	Get all registered to methods.
mix_system(*system, type_map, **kwargs)	Mix the systems into mixed_type ones according to the unified given type_map.
post(func_name)	Register a post function for from method.
register(key)	Register a format plugin.
register_from(key)	Register a from method if the target method name is not default.
register_to(key)	Register a to method if the target method name is not default.
to_bond_order_system(data, rdkit_mol, *args, ...)	Implement BondOrderSystem.to that converts from BondOrderSystem to this format.
to_labeled_system(data, *args, **kwargs)	Implement LabeledSystem.to that converts from LabeledSystem to this format.
to_multi_systems(formulas, directory, **kwargs)	Implement MultiSystems.to that converts from MultiSystems to this format.
to_system(data[, file_name, frame_idx])	Dump the system in gromacs .gro format.

from_system(*file_name*, *format_atom_name=True*, ***kwargs*)

Load gromacs .gro file.

Parameters

file_name
[str] The input file name

format_atom_name
[bool] Whether to format the atom name

****kwargs**
[dict] other parameters

to_system(*data*, *file_name=None*, *frame_idx=-1*, ***kwargs*)

Dump the system in gromacs .gro format.

Parameters

data
[dict] System data

file_name
[str or None] The output file name. If None, return the file content as a string

frame_idx
[int] The index of the frame to dump

****kwargs**
[dict] other parameters

dpdata.plugins.lammps module

class dpdata.plugins.lammps.**LAMMPSDumpFormat**

Bases: *Format*

Methods

MultiModes()	File mode for MultiSystems.
from_bond_order_system(file_name, **kwargs)	Implement BondOrderSystem.from that converts from this format to BondOrderSystem.
from_labeled_system(file_name, **kwargs)	Implement LabeledSystem.from that converts from this format to LabeledSystem.
from_multi_systems(directory, **kwargs)	Implement MultiSystems.from that converts from this format to MultiSystems.
from_system(file_name[, type_map, begin, ...])	Implement System.from that converts from this format to System.
get_formats()	Get all registered formats.
get_from_methods()	Get all registered from methods.
get_to_methods()	Get all registered to methods.
mix_system(*system, type_map, **kwargs)	Mix the systems into mixed_type ones according to the unified given type_map.
post(func_name)	Register a post function for from method.
register(key)	Register a format plugin.
register_from(key)	Register a from method if the target method name is not default.
register_to(key)	Register a to method if the target method name is not default.
to_bond_order_system(data, rdkit_mol, *args, ...)	Implement BondOrderSystem.to that converts from BondOrderSystem to this format.
to_labeled_system(data, *args, **kwargs)	Implement LabeledSystem.to that converts from LabeledSystem to this format.
to_multi_systems(formulas, directory, **kwargs)	Implement MultiSystems.to that converts from MultiSystems to this format.
to_system(data, *args, **kwargs)	Implement System.to that converts from System to this format.

from_system(*file_name*, *type_map=None*, *begin=0*, *step=1*, *unwrap=False*, ***kwargs*)

Implement System.from that converts from this format to System.

Parameters

file_name
[str] file name, i.e. the first argument

****kwargs**
[dict] keyword arguments that will be passed from the method

Returns

data
[dict] system data, whose keys are defined in System.DTYPES

class dpdata.plugins.lammps.**LAMMPSLmpFormat**

Bases: *Format*

Methods

MultiModes()	File mode for MultiSystems.
from_bond_order_system(file_name, **kwargs)	Implement BondOrderSystem.from that converts from this format to BondOrderSystem.
from_labeled_system(file_name, **kwargs)	Implement LabeledSystem.from that converts from this format to LabeledSystem.
from_multi_systems(directory, **kwargs)	Implement MultiSystems.from that converts from this format to MultiSystems.
from_system(file_name[, type_map])	Implement System.from that converts from this format to System.
get_formats()	Get all registered formats.
get_from_methods()	Get all registered from methods.
get_to_methods()	Get all registered to methods.
mix_system(*system, type_map, **kwargs)	Mix the systems into mixed_type ones according to the unified given type_map.
post(func_name)	Register a post function for from method.
register(key)	Register a format plugin.
register_from(key)	Register a from method if the target method name is not default.
register_to(key)	Register a to method if the target method name is not default.
to_bond_order_system(data, rdkit_mol, *args, ...)	Implement BondOrderSystem.to that converts from BondOrderSystem to this format.
to_labeled_system(data, *args, **kwargs)	Implement LabeledSystem.to that converts from LabeledSystem to this format.
to_multi_systems(formulas, directory, **kwargs)	Implement MultiSystems.to that converts from MultiSystems to this format.
to_system(data, file_name[, frame_idx])	Dump the system in lammps data format.

from_system(*file_name*, *type_map=None*, ***kwargs*)

Implement System.from that converts from this format to System.

Parameters

file_name
[str] file name, i.e. the first argument

****kwargs**
[dict] keyword arguments that will be passed from the method

Returns

data
[dict] system data, whose keys are defined in System.DTYPES

`to_system`(*data*, *file_name*, *frame_idx=0*, ***kwargs*)

Dump the system in lammps data format.

Parameters

data
[dict] System data

file_name
[str] The output file name

frame_idx
[int] The index of the frame to dump

****kwargs**
[dict] other parameters

dpdata.plugins.list module

`class` dpdata.plugins.list.`ListFormat`

Bases: *Format*

Methods

MultiModes()	File mode for MultiSystems.
from_bond_order_system(file_name, **kwargs)	Implement BondOrderSystem.from that converts from this format to BondOrderSystem.
from_labeled_system(file_name, **kwargs)	Implement LabeledSystem.from that converts from this format to LabeledSystem.
from_multi_systems(directory, **kwargs)	Implement MultiSystems.from that converts from this format to MultiSystems.
from_system(file_name, **kwargs)	Implement System.from that converts from this format to System.
get_formats()	Get all registered formats.
get_from_methods()	Get all registered from methods.
get_to_methods()	Get all registered to methods.
mix_system(*system, type_map, **kwargs)	Mix the systems into mixed_type ones according to the unified given type_map.
post(func_name)	Register a post function for from method.
register(key)	Register a format plugin.
register_from(key)	Register a from method if the target method name is not default.
register_to(key)	Register a to method if the target method name is not default.
to_bond_order_system(data, rdkit_mol, *args, ...)	Implement BondOrderSystem.to that converts from BondOrderSystem to this format.
to_labeled_system(data, *args, **kwargs)	Implement LabeledSystem.to that converts from LabeledSystem to this format.
to_multi_systems(formulas, directory, **kwargs)	Implement MultiSystems.to that converts from MultiSystems to this format.
to_system(data, **kwargs)	Convert system to list, usefull for data collection.

to_system(*data*, ***kwargs*)

Convert system to list, usefull for data collection.

dpdata.plugins.n2p2 module

class dpdata.plugins.n2p2.**N2P2Format**

Bases: *Format*

n2p2.

This class support the conversion from and to the training data of n2p2 format. For more information about the n2p2 format, please refer to https://compphysvienna.github.io/n2p2/topics/cfg_file.html

Methods

MultiModes()	File mode for MultiSystems.
from_bond_order_system(file_name, **kwargs)	Implement BondOrderSystem.from that converts from this format to BondOrderSystem.
from_labeled_system(file_name, **kwargs)	Read from n2p2 format.
from_multi_systems(directory, **kwargs)	Implement MultiSystems.from that converts from this format to MultiSystems.
from_system(file_name, **kwargs)	Implement System.from that converts from this format to System.
get_formats()	Get all registered formats.
get_from_methods()	Get all registered from methods.
get_to_methods()	Get all registered to methods.
mix_system(*system, type_map, **kwargs)	Mix the systems into mixed_type ones according to the unified given type_map.
post(func_name)	Register a post function for from method.
register(key)	Register a format plugin.
register_from(key)	Register a from method if the target method name is not default.
register_to(key)	Register a to method if the target method name is not default.
to_bond_order_system(data, rdkit_mol, *args, ...)	Implement BondOrderSystem.to that converts from BondOrderSystem to this format.
to_labeled_system(data, file_name, **kwargs)	Write n2p2 format.
to_multi_systems(formulas, directory, **kwargs)	Implement MultiSystems.to that converts from MultiSystems to this format.
to_system(data, *args, **kwargs)	Implement System.to that converts from System to this format.

from_labeled_system(*file_name*, ***kwargs*)

Read from n2p2 format.

Parameters

file_name
[str] file name, i.e. the first argument

****kwargs**
[dict] keyword arguments that will be passed from the method

Returns

data
[dict] system data, whose keys are defined in LabeledSystem.DTYPES

to_labeled_system(*data*, *file_name*, ***kwargs*)

Write n2p2 format.

By default, LabeledSystem.to will fallback to System.to.

Parameters

data
[dict] system data, whose keys are defined in LabeledSystem.DTYPES

file_name
[str] file name, where the data will be written

***args**
[list] arguments that will be passed from the method

****kwargs**
[dict] keyword arguments that will be passed from the method

dpdata.plugins.n2p2.**match_indices**(*atype1*, *atype2*)

dpdata.plugins.openmx module

class dpdata.plugins.openmx.**OPENMXFormat**

Bases: *Format*

Format for the *OpenMX <https://www.openmx-square.org/>*.

OpenMX (Open source package for Material eXplorer) is a nano-scale material simulation package based on DFT, norm-conserving pseudopotentials, and pseudo-atomic localized basis functions.

Note that two output files, System.Name.dat and System.Name.md, are required.

Use the *openmx/md* keyword argument to supply this format.

Methods

MultiModes()	File mode for MultiSystems.
from_bond_order_system(file_name, **kwargs)	Implement BondOrderSystem.from that converts from this format to BondOrderSystem.
from_labeled_system(file_name, **kwargs)	Read from OpenMX output.
from_multi_systems(directory, **kwargs)	Implement MultiSystems.from that converts from this format to MultiSystems.
from_system(file_name, **kwargs)	Read from OpenMX output.
get_formats()	Get all registered formats.
get_from_methods()	Get all registered from methods.
get_to_methods()	Get all registered to methods.
mix_system(*system, type_map, **kwargs)	Mix the systems into mixed_type ones according to the unified given type_map.
post(func_name)	Register a post function for from method.
register(key)	Register a format plugin.
register_from(key)	Register a from method if the target method name is not default.
register_to(key)	Register a to method if the target method name is not default.
to_bond_order_system(data, rdkit_mol, *args, ...)	Implement BondOrderSystem.to that converts from BondOrderSystem to this format.
to_labeled_system(data, *args, **kwargs)	Implement LabeledSystem.to that converts from LabeledSystem to this format.
to_multi_systems(formulas, directory, **kwargs)	Implement MultiSystems.to that converts from MultiSystems to this format.
to_system(data, *args, **kwargs)	Implement System.to that converts from System to this format.

from_labeled_system(*file_name: str*, ***kwargs*) → dict

Read from OpenMX output.

Parameters

file_name
[str] file name, which is specified by a input file, i.e. System.Name.dat

****kwargs**
[dict] other parameters

Returns

dict
data dict

from_system(*file_name: str*, ***kwargs*) → dict

Read from OpenMX output.

Parameters

file_name
[str] file name, which is specified by a input file, i.e. System.Name.dat

****kwargs**
[dict] other parameters

Returns

dict
data dict

dpdata.plugins.orca module

class dpdata.plugins.orca.**ORCASPOutFormat**

Bases: *Format*

ORCA single point energy output.

Note that both the energy and the gradient should be printed into the output file.

Methods

MultiModes()	File mode for MultiSystems.
from_bond_order_system(file_name, **kwargs)	Implement BondOrderSystem.from that converts from this format to BondOrderSystem.
from_labeled_system(file_name, **kwargs)	Read from ORCA single point energy output.
from_multi_systems(directory, **kwargs)	Implement MultiSystems.from that converts from this format to MultiSystems.
from_system(file_name, **kwargs)	Implement System.from that converts from this format to System.
get_formats()	Get all registered formats.
get_from_methods()	Get all registered from methods.
get_to_methods()	Get all registered to methods.
mix_system(*system, type_map, **kwargs)	Mix the systems into mixed_type ones according to the unified given type_map.
post(func_name)	Register a post function for from method.
register(key)	Register a format plugin.
register_from(key)	Register a from method if the target method name is not default.
register_to(key)	Register a to method if the target method name is not default.
to_bond_order_system(data, rdkit_mol, *args, ...)	Implement BondOrderSystem.to that converts from BondOrderSystem to this format.
to_labeled_system(data, *args, **kwargs)	Implement LabeledSystem.to that converts from LabeledSystem to this format.
to_multi_systems(formulas, directory, **kwargs)	Implement MultiSystems.to that converts from MultiSystems to this format.
to_system(data, *args, **kwargs)	Implement System.to that converts from System to this format.

from_labeled_system(*file_name: str*, ***kwargs*) → dict

Read from ORCA single point energy output.

Parameters

file_name
[str] file name

****kwargs**
keyword arguments

Returns

dict
system data

dpdata.plugins.psi4 module

class dpdata.plugins.psi4.**PSI4InputFormat**

Bases: *Format*

Psi4 input file.

Methods

MultiModes()	File mode for MultiSystems.
from_bond_order_system(file_name, **kwargs)	Implement BondOrderSystem.from that converts from this format to BondOrderSystem.
from_labeled_system(file_name, **kwargs)	Implement LabeledSystem.from that converts from this format to LabeledSystem.
from_multi_systems(directory, **kwargs)	Implement MultiSystems.from that converts from this format to MultiSystems.
from_system(file_name, **kwargs)	Implement System.from that converts from this format to System.
get_formats()	Get all registered formats.
get_from_methods()	Get all registered from methods.
get_to_methods()	Get all registered to methods.
mix_system(*system, type_map, **kwargs)	Mix the systems into mixed_type ones according to the unified given type_map.
post(func_name)	Register a post function for from method.
register(key)	Register a format plugin.
register_from(key)	Register a from method if the target method name is not default.
register_to(key)	Register a to method if the target method name is not default.
to_bond_order_system(data, rdkit_mol, *args, ...)	Implement BondOrderSystem.to that converts from BondOrderSystem to this format.
to_labeled_system(data, *args, **kwargs)	Implement LabeledSystem.to that converts from LabeledSystem to this format.
to_multi_systems(formulas, directory, **kwargs)	Implement MultiSystems.to that converts from MultiSystems to this format.
to_system(data, file_name, method, basis[, ...])	Write PSI4 input.

to_system(*data: dict*, *file_name: str*, *method: str*, *basis: str*, *charge: int = 0*, *multiplicity: int = 1*, *frame_idx=0*, ***kwargs*)

Write PSI4 input.

Parameters

data
[dict] system data

file_name
[str] file name

method
[str] computational method

basis
[str] basis set; see https://psicode.org/psi4manual/master/basissets_tables.html

charge
[int, default=0] charge of system

multiplicity
[int, default=1] multiplicity of system

frame_idx
[int, default=0] The index of the frame to dump

****kwargs**
keyword arguments

class dpdata.plugins.psi4.**PSI4OutFormat**

Bases: *Format*

Psi4 output.

Note that both the energy and the gradient should be printed into the output file.

Methods

MultiModes()	File mode for MultiSystems.
from_bond_order_system(file_name, **kwargs)	Implement BondOrderSystem.from that converts from this format to BondOrderSystem.
from_labeled_system(file_name, **kwargs)	Read from Psi4 output.
from_multi_systems(directory, **kwargs)	Implement MultiSystems.from that converts from this format to MultiSystems.
from_system(file_name, **kwargs)	Implement System.from that converts from this format to System.
get_formats()	Get all registered formats.
get_from_methods()	Get all registered from methods.
get_to_methods()	Get all registered to methods.
mix_system(*system, type_map, **kwargs)	Mix the systems into mixed_type ones according to the unified given type_map.
post(func_name)	Register a post function for from method.
register(key)	Register a format plugin.
register_from(key)	Register a from method if the target method name is not default.
register_to(key)	Register a to method if the target method name is not default.
to_bond_order_system(data, rdkit_mol, *args, ...)	Implement BondOrderSystem.to that converts from BondOrderSystem to this format.
to_labeled_system(data, *args, **kwargs)	Implement LabeledSystem.to that converts from LabeledSystem to this format.
to_multi_systems(formulas, directory, **kwargs)	Implement MultiSystems.to that converts from MultiSystems to this format.
to_system(data, *args, **kwargs)	Implement System.to that converts from System to this format.

from_labeled_system(*file_name: str*, ***kwargs*) → dict

Read from Psi4 output.

Parameters

file_name
[str] file name

****kwargs**
keyword arguments

Returns

dict
system data

dpdata.plugins.pwmat module

class dpdata.plugins.pwmat.**PwmatAtomconfigFormat**

Bases: *Format*

Methods

MultiModes()	File mode for MultiSystems.
from_bond_order_system(file_name, **kwargs)	Implement BondOrderSystem.from that converts from this format to BondOrderSystem.
from_labeled_system(file_name, **kwargs)	Implement LabeledSystem.from that converts from this format to LabeledSystem.
from_multi_systems(directory, **kwargs)	Implement MultiSystems.from that converts from this format to MultiSystems.
from_system(file_name, **kwargs)	Implement System.from that converts from this format to System.
get_formats()	Get all registered formats.
get_from_methods()	Get all registered from methods.
get_to_methods()	Get all registered to methods.
mix_system(*system, type_map, **kwargs)	Mix the systems into mixed_type ones according to the unified given type_map.
post(func_name)	Register a post function for from method.
register(key)	Register a format plugin.
register_from(key)	Register a from method if the target method name is not default.
register_to(key)	Register a to method if the target method name is not default.
to_bond_order_system(data, rdkit_mol, *args, ...)	Implement BondOrderSystem.to that converts from BondOrderSystem to this format.
to_labeled_system(data, *args, **kwargs)	Implement LabeledSystem.to that converts from LabeledSystem to this format.
to_multi_systems(formulas, directory, **kwargs)	Implement MultiSystems.to that converts from MultiSystems to this format.
to_system(data, file_name[, frame_idx])	Dump the system in pwmat atom.config format.

from_system(*file_name*, ***kwargs*)

Implement System.from that converts from this format to System.

Parameters

file_name
[str] file name, i.e. the first argument

****kwargs**
[dict] keyword arguments that will be passed from the method

Returns

data
[dict] system data, whose keys are defined in System.DTYPES

`to_system`(*data*, *file_name*, *frame_idx=0*, **args*, ***kwargs*)

Dump the system in pwmat atom.config format.

Parameters

data
[dict] The system data

file_name
[str] The output file name

frame_idx
[int] The index of the frame to dump

***args**
[list] other parameters

****kwargs**
[dict] other parameters

`class` dpdata.plugins.pwmat.`PwmatOutputFormat`

Bases: *Format*

Methods

MultiModes()	File mode for MultiSystems.
from_bond_order_system(file_name, **kwargs)	Implement BondOrderSystem.from that converts from this format to BondOrderSystem.
from_labeled_system(file_name[, begin, ...])	Implement LabeledSystem.from that converts from this format to LabeledSystem.
from_multi_systems(directory, **kwargs)	Implement MultiSystems.from that converts from this format to MultiSystems.
from_system(file_name, **kwargs)	Implement System.from that converts from this format to System.
get_formats()	Get all registered formats.
get_from_methods()	Get all registered from methods.
get_to_methods()	Get all registered to methods.
mix_system(*system, type_map, **kwargs)	Mix the systems into mixed_type ones according to the unified given type_map.
post(func_name)	Register a post function for from method.
register(key)	Register a format plugin.
register_from(key)	Register a from method if the target method name is not default.
register_to(key)	Register a to method if the target method name is not default.
to_bond_order_system(data, rdkit_mol, *args, ...)	Implement BondOrderSystem.to that converts from BondOrderSystem to this format.
to_labeled_system(data, *args, **kwargs)	Implement LabeledSystem.to that converts from LabeledSystem to this format.
to_multi_systems(formulas, directory, **kwargs)	Implement MultiSystems.to that converts from MultiSystems to this format.
to_system(data, *args, **kwargs)	Implement System.to that converts from System to this format.

from_labeled_system(*file_name*, *begin=0*, *step=1*, *convergence_check=True*, ***kwargs*)

Implement LabeledSystem.from that converts from this format to LabeledSystem.

Parameters

file_name
[str] file name, i.e. the first argument

****kwargs**
[dict] keyword arguments that will be passed from the method

Returns

data
[dict] system data, whose keys are defined in LabeledSystem.DTYPES

dpdata.plugins.pymatgen module

class dpdata.plugins.pymatgen.**PyMatgenCSEFormat**

Bases: *Format*

Methods

MultiModes()	File mode for MultiSystems.
from_bond_order_system(file_name, **kwargs)	Implement BondOrderSystem.from that converts from this format to BondOrderSystem.
from_labeled_system(file_name, **kwargs)	Implement LabeledSystem.from that converts from this format to LabeledSystem.
from_multi_systems(directory, **kwargs)	Implement MultiSystems.from that converts from this format to MultiSystems.
from_system(file_name, **kwargs)	Implement System.from that converts from this format to System.
get_formats()	Get all registered formats.
get_from_methods()	Get all registered from methods.
get_to_methods()	Get all registered to methods.
mix_system(*system, type_map, **kwargs)	Mix the systems into mixed_type ones according to the unified given type_map.
post(func_name)	Register a post function for from method.
register(key)	Register a format plugin.
register_from(key)	Register a from method if the target method name is not default.
register_to(key)	Register a to method if the target method name is not default.
to_bond_order_system(data, rdkit_mol, *args, ...)	Implement BondOrderSystem.to that converts from BondOrderSystem to this format.
to_labeled_system(data, *args, **kwargs)	Convert System to Pymagen ComputedStructureEntry obj.
to_multi_systems(formulas, directory, **kwargs)	Implement MultiSystems.to that converts from MultiSystems to this format.
to_system(data, *args, **kwargs)	Implement System.to that converts from System to this format.

to_labeled_system(*data*, **args*, ***kwargs*)

Convert System to Pymagen ComputedStructureEntry obj.

class dpdata.plugins.pymatgen.**PyMatgenMoleculeFormat**

Bases: *Format*

Methods

MultiModes()	File mode for MultiSystems.
from_bond_order_system(file_name, **kwargs)	Implement BondOrderSystem.from that converts from this format to BondOrderSystem.
from_labeled_system(file_name, **kwargs)	Implement LabeledSystem.from that converts from this format to LabeledSystem.
from_multi_systems(directory, **kwargs)	Implement MultiSystems.from that converts from this format to MultiSystems.
from_system(file_name, **kwargs)	Implement System.from that converts from this format to System.
get_formats()	Get all registered formats.
get_from_methods()	Get all registered from methods.
get_to_methods()	Get all registered to methods.
mix_system(*system, type_map, **kwargs)	Mix the systems into mixed_type ones according to the unified given type_map.
post(func_name)	Register a post function for from method.
register(key)	Register a format plugin.
register_from(key)	Register a from method if the target method name is not default.
register_to(key)	Register a to method if the target method name is not default.
to_bond_order_system(data, rdkit_mol, *args, ...)	Implement BondOrderSystem.to that converts from BondOrderSystem to this format.
to_labeled_system(data, *args, **kwargs)	Implement LabeledSystem.to that converts from LabeledSystem to this format.
to_multi_systems(formulas, directory, **kwargs)	Implement MultiSystems.to that converts from MultiSystems to this format.
to_system(data, **kwargs)	Convert System to Pymatgen Molecule obj.

from_system(*file_name*, ***kwargs*)

Implement System.from that converts from this format to System.

Parameters

file_name
[str] file name, i.e. the first argument

****kwargs**
[dict] keyword arguments that will be passed from the method

Returns

data
[dict] system data, whose keys are defined in System.DTYPES

to_system(*data*, ***kwargs*)

Convert System to Pymatgen Molecule obj.

class dpdata.plugins.pymatgen.**PyMatgenStructureFormat**

Bases: *Format*

Methods

MultiModes()	File mode for MultiSystems.
from_bond_order_system(file_name, **kwargs)	Implement BondOrderSystem.from that converts from this format to BondOrderSystem.
from_labeled_system(file_name, **kwargs)	Implement LabeledSystem.from that converts from this format to LabeledSystem.
from_multi_systems(directory, **kwargs)	Implement MultiSystems.from that converts from this format to MultiSystems.
from_system(file_name, **kwargs)	Implement System.from that converts from this format to System.
get_formats()	Get all registered formats.
get_from_methods()	Get all registered from methods.
get_to_methods()	Get all registered to methods.
mix_system(*system, type_map, **kwargs)	Mix the systems into mixed_type ones according to the unified given type_map.
post(func_name)	Register a post function for from method.
register(key)	Register a format plugin.
register_from(key)	Register a from method if the target method name is not default.
register_to(key)	Register a to method if the target method name is not default.
to_bond_order_system(data, rdkit_mol, *args, ...)	Implement BondOrderSystem.to that converts from BondOrderSystem to this format.
to_labeled_system(data, *args, **kwargs)	Implement LabeledSystem.to that converts from LabeledSystem to this format.
to_multi_systems(formulas, directory, **kwargs)	Implement MultiSystems.to that converts from MultiSystems to this format.
to_system(data, **kwargs)	Convert System to Pymatgen Structure obj.

to_system(*data*, ***kwargs*)

Convert System to Pymatgen Structure obj.

dpdata.plugins.qe module

class dpdata.plugins.qe.**QECPPWSCFFormat**

Bases: *Format*

Methods

MultiModes()	File mode for MultiSystems.
from_bond_order_system(file_name, **kwargs)	Implement BondOrderSystem.from that converts from this format to BondOrderSystem.
from_labeled_system(file_name, **kwargs)	Implement LabeledSystem.from that converts from this format to LabeledSystem.
from_multi_systems(directory, **kwargs)	Implement MultiSystems.from that converts from this format to MultiSystems.
from_system(file_name, **kwargs)	Implement System.from that converts from this format to System.
get_formats()	Get all registered formats.
get_from_methods()	Get all registered from methods.
get_to_methods()	Get all registered to methods.
mix_system(*system, type_map, **kwargs)	Mix the systems into mixed_type ones according to the unified given type_map.
post(func_name)	Register a post function for from method.
register(key)	Register a format plugin.
register_from(key)	Register a from method if the target method name is not default.
register_to(key)	Register a to method if the target method name is not default.
to_bond_order_system(data, rdkit_mol, *args, ...)	Implement BondOrderSystem.to that converts from BondOrderSystem to this format.
to_labeled_system(data, *args, **kwargs)	Implement LabeledSystem.to that converts from LabeledSystem to this format.
to_multi_systems(formulas, directory, **kwargs)	Implement MultiSystems.to that converts from MultiSystems to this format.
to_system(data, *args, **kwargs)	Implement System.to that converts from System to this format.

from_labeled_system(*file_name*, ***kwargs*)

Implement LabeledSystem.from that converts from this format to LabeledSystem.

Parameters

file_name
[str] file name, i.e. the first argument

****kwargs**
[dict] keyword arguments that will be passed from the method

Returns

data
[dict] system data, whose keys are defined in LabeledSystem.DTYPES

class dpdata.plugins.qe.**QECPTrajFormat**

Bases: *Format*

Methods

MultiModes()	File mode for MultiSystems.
from_bond_order_system(file_name, **kwargs)	Implement BondOrderSystem.from that converts from this format to BondOrderSystem.
from_labeled_system(file_name[, begin, step])	Implement LabeledSystem.from that converts from this format to LabeledSystem.
from_multi_systems(directory, **kwargs)	Implement MultiSystems.from that converts from this format to MultiSystems.
from_system(file_name[, begin, step])	Implement System.from that converts from this format to System.
get_formats()	Get all registered formats.
get_from_methods()	Get all registered from methods.
get_to_methods()	Get all registered to methods.
mix_system(*system, type_map, **kwargs)	Mix the systems into mixed_type ones according to the unified given type_map.
post(func_name)	Register a post function for from method.
register(key)	Register a format plugin.
register_from(key)	Register a from method if the target method name is not default.
register_to(key)	Register a to method if the target method name is not default.
to_bond_order_system(data, rdkit_mol, *args, ...)	Implement BondOrderSystem.to that converts from BondOrderSystem to this format.
to_labeled_system(data, *args, **kwargs)	Implement LabeledSystem.to that converts from LabeledSystem to this format.
to_multi_systems(formulas, directory, **kwargs)	Implement MultiSystems.to that converts from MultiSystems to this format.
to_system(data, *args, **kwargs)	Implement System.to that converts from System to this format.

from_labeled_system(*file_name*, *begin=0*, *step=1*, ***kwargs*)

Implement LabeledSystem.from that converts from this format to LabeledSystem.

Parameters

file_name
[str] file name, i.e. the first argument

****kwargs**
[dict] keyword arguments that will be passed from the method

Returns

data
[dict] system data, whose keys are defined in LabeledSystem.DTYPES

from_system(*file_name*, *begin=0*, *step=1*, ***kwargs*)

Implement System.from that converts from this format to System.

Parameters

file_name
[str] file name, i.e. the first argument

****kwargs**
[dict] keyword arguments that will be passed from the method

Returns

data
[dict] system data, whose keys are defined in System.DTYPES

dpdata.plugins.rdkit module

class dpdata.plugins.rdkit.**MolFormat**

Bases: *Format*

Methods

MultiModes()	File mode for MultiSystems.
from_bond_order_system(file_name, **kwargs)	Implement BondOrderSystem.from that converts from this format to BondOrderSystem.
from_labeled_system(file_name, **kwargs)	Implement LabeledSystem.from that converts from this format to LabeledSystem.
from_multi_systems(directory, **kwargs)	Implement MultiSystems.from that converts from this format to MultiSystems.
from_system(file_name, **kwargs)	Implement System.from that converts from this format to System.
get_formats()	Get all registered formats.
get_from_methods()	Get all registered from methods.
get_to_methods()	Get all registered to methods.
mix_system(*system, type_map, **kwargs)	Mix the systems into mixed_type ones according to the unified given type_map.
post(func_name)	Register a post function for from method.
register(key)	Register a format plugin.
register_from(key)	Register a from method if the target method name is not default.
register_to(key)	Register a to method if the target method name is not default.
to_bond_order_system(data, mol, file_name[, ...])	Implement BondOrderSystem.to that converts from BondOrderSystem to this format.
to_labeled_system(data, *args, **kwargs)	Implement LabeledSystem.to that converts from LabeledSystem to this format.
to_multi_systems(formulas, directory, **kwargs)	Implement MultiSystems.to that converts from MultiSystems to this format.
to_system(data, *args, **kwargs)	Implement System.to that converts from System to this format.

from_bond_order_system(*file_name*, ***kwargs*)

Implement BondOrderSystem.from that converts from this format to BondOrderSystem.

Parameters

file_name
[str] file name, i.e. the first argument

****kwargs**
[dict] keyword arguments that will be passed from the method

Returns

data
[dict] system data

`to_bond_order_system`(*data*, *mol*, *file_name*, *frame_idx=0*, ***kwargs*)

Implement BondOrderSystem.to that converts from BondOrderSystem to this format.

By default, BondOrderSystem.to will fallback to LabeledSystem.to.

Parameters

data
[dict] system data

rdkit_mol
[rdkit.Chem.rdchem.Mol] rdkit mol object

***args**
[list] arguments that will be passed from the method

****kwargs**
[dict] keyword arguments that will be passed from the method

`class dpdata.plugins.rdkit.SdfFormat`

Bases: *Format*

Methods

`MultiModes`()	File mode for MultiSystems.
from_bond_order_system(file_name, **kwargs)	Note that it requires all molecules in .sdf file must be of the same topology.
`from_labeled_system`(file_name, **kwargs)	Implement LabeledSystem.from that converts from this format to LabeledSystem.
`from_multi_systems`(directory, **kwargs)	Implement MultiSystems.from that converts from this format to MultiSystems.
`from_system`(file_name, **kwargs)	Implement System.from that converts from this format to System.
`get_formats`()	Get all registered formats.
`get_from_methods`()	Get all registered from methods.
`get_to_methods`()	Get all registered to methods.
`mix_system`(*system, type_map, **kwargs)	Mix the systems into mixed_type ones according to the unified given type_map.
`post`(func_name)	Register a post function for from method.
`register`(key)	Register a format plugin.
`register_from`(key)	Register a from method if the target method name is not default.
`register_to`(key)	Register a to method if the target method name is not default.
to_bond_order_system(data, mol, file_name[, ...])	Implement BondOrderSystem.to that converts from BondOrderSystem to this format.
`to_labeled_system`(data, *args, **kwargs)	Implement LabeledSystem.to that converts from LabeledSystem to this format.
`to_multi_systems`(formulas, directory, **kwargs)	Implement MultiSystems.to that converts from MultiSystems to this format.
`to_system`(data, *args, **kwargs)	Implement System.to that converts from System to this format.

from_bond_order_system(*file_name*, ***kwargs*)

Note that it requires all molecules in .sdf file must be of the same topology.

to_bond_order_system(*data*, *mol*, *file_name*, *frame_idx=-1*, ***kwargs*)

Implement BondOrderSystem.to that converts from BondOrderSystem to this format.

By default, BondOrderSystem.to will fallback to LabeledSystem.to.

Parameters

data
[dict] system data

rdkit_mol
[rdkit.Chem.rdchem.Mol] rdkit mol object

***args**
[list] arguments that will be passed from the method

****kwargs**
[dict] keyword arguments that will be passed from the method

dpdata.plugins.siesta module

class dpdata.plugins.siesta.**SiestaAIMDOutputFormat**

Bases: *Format*

Methods

MultiModes()	File mode for MultiSystems.
from_bond_order_system(file_name, **kwargs)	Implement BondOrderSystem.from that converts from this format to BondOrderSystem.
from_labeled_system(file_name, **kwargs)	Implement LabeledSystem.from that converts from this format to LabeledSystem.
from_multi_systems(directory, **kwargs)	Implement MultiSystems.from that converts from this format to MultiSystems.
from_system(file_name, **kwargs)	Implement System.from that converts from this format to System.
get_formats()	Get all registered formats.
get_from_methods()	Get all registered from methods.
get_to_methods()	Get all registered to methods.
mix_system(*system, type_map, **kwargs)	Mix the systems into mixed_type ones according to the unified given type_map.
post(func_name)	Register a post function for from method.
register(key)	Register a format plugin.
register_from(key)	Register a from method if the target method name is not default.
register_to(key)	Register a to method if the target method name is not default.
to_bond_order_system(data, rdkit_mol, *args, ...)	Implement BondOrderSystem.to that converts from BondOrderSystem to this format.
to_labeled_system(data, *args, **kwargs)	Implement LabeledSystem.to that converts from LabeledSystem to this format.
to_multi_systems(formulas, directory, **kwargs)	Implement MultiSystems.to that converts from MultiSystems to this format.
to_system(data, *args, **kwargs)	Implement System.to that converts from System to this format.

from_labeled_system(*file_name*, ***kwargs*)

Implement LabeledSystem.from that converts from this format to LabeledSystem.

Parameters

file_name
[str] file name, i.e. the first argument

****kwargs**
[dict] keyword arguments that will be passed from the method

Returns

data
[dict] system data, whose keys are defined in LabeledSystem.DTYPES

from_system(*file_name*, ***kwargs*)

Implement System.from that converts from this format to System.

Parameters

file_name
[str] file name, i.e. the first argument

****kwargs**
[dict] keyword arguments that will be passed from the method

Returns

data
[dict] system data, whose keys are defined in System.DTYPES

class dpdata.plugins.siesta.**SiestaOutputFormat**

Bases: *Format*

Methods

MultiModes()	File mode for MultiSystems.
from_bond_order_system(file_name, **kwargs)	Implement BondOrderSystem.from that converts from this format to BondOrderSystem.
from_labeled_system(file_name, **kwargs)	Implement LabeledSystem.from that converts from this format to LabeledSystem.
from_multi_systems(directory, **kwargs)	Implement MultiSystems.from that converts from this format to MultiSystems.
from_system(file_name, **kwargs)	Implement System.from that converts from this format to System.
get_formats()	Get all registered formats.
get_from_methods()	Get all registered from methods.
get_to_methods()	Get all registered to methods.
mix_system(*system, type_map, **kwargs)	Mix the systems into mixed_type ones according to the unified given type_map.
post(func_name)	Register a post function for from method.
register(key)	Register a format plugin.
register_from(key)	Register a from method if the target method name is not default.
register_to(key)	Register a to method if the target method name is not default.
to_bond_order_system(data, rdkit_mol, *args, ...)	Implement BondOrderSystem.to that converts from BondOrderSystem to this format.
to_labeled_system(data, *args, **kwargs)	Implement LabeledSystem.to that converts from LabeledSystem to this format.
to_multi_systems(formulas, directory, **kwargs)	Implement MultiSystems.to that converts from MultiSystems to this format.
to_system(data, *args, **kwargs)	Implement System.to that converts from System to this format.

from_labeled_system(*file_name*, ***kwargs*)

Implement LabeledSystem.from that converts from this format to LabeledSystem.

Parameters

file_name
[str] file name, i.e. the first argument

****kwargs**
[dict] keyword arguments that will be passed from the method

Returns

data
[dict] system data, whose keys are defined in LabeledSystem.DTYPES

from_system(*file_name*, ***kwargs*)

Implement System.from that converts from this format to System.

Parameters

file_name
[str] file name, i.e. the first argument

****kwargs**
[dict] keyword arguments that will be passed from the method

Returns

data
[dict] system data, whose keys are defined in System.DTYPES

dpdata.plugins.vasp module

class dpdata.plugins.vasp.**VASPOutcarFormat**

Bases: *Format*

Methods

MultiModes()	File mode for MultiSystems.
from_bond_order_system(file_name, **kwargs)	Implement BondOrderSystem.from that converts from this format to BondOrderSystem.
from_labeled_system(file_name[, begin, ...])	Implement LabeledSystem.from that converts from this format to LabeledSystem.
from_multi_systems(directory, **kwargs)	Implement MultiSystems.from that converts from this format to MultiSystems.
from_system(file_name, **kwargs)	Implement System.from that converts from this format to System.
get_formats()	Get all registered formats.
get_from_methods()	Get all registered from methods.
get_to_methods()	Get all registered to methods.
mix_system(*system, type_map, **kwargs)	Mix the systems into mixed_type ones according to the unified given type_map.
post(func_name)	Register a post function for from method.
register(key)	Register a format plugin.
register_from(key)	Register a from method if the target method name is not default.
register_to(key)	Register a to method if the target method name is not default.
to_bond_order_system(data, rdkit_mol, *args, ...)	Implement BondOrderSystem.to that converts from BondOrderSystem to this format.
to_labeled_system(data, *args, **kwargs)	Implement LabeledSystem.to that converts from LabeledSystem to this format.
to_multi_systems(formulas, directory, **kwargs)	Implement MultiSystems.to that converts from MultiSystems to this format.
to_system(data, *args, **kwargs)	Implement System.to that converts from System to this format.

from_labeled_system(*file_name*, *begin=0*, *step=1*, *convergence_check=True*, ***kwargs*)

Implement LabeledSystem.from that converts from this format to LabeledSystem.

Parameters

file_name
[str] file name, i.e. the first argument

****kwargs**
[dict] keyword arguments that will be passed from the method

Returns

data
[dict] system data, whose keys are defined in LabeledSystem.DTYPES

class dpdata.plugins.vasp.**VASPPoscarFormat**

Bases: *Format*

Methods

MultiModes()	File mode for MultiSystems.
from_bond_order_system(file_name, **kwargs)	Implement BondOrderSystem.from that converts from this format to BondOrderSystem.
from_labeled_system(file_name, **kwargs)	Implement LabeledSystem.from that converts from this format to LabeledSystem.
from_multi_systems(directory, **kwargs)	Implement MultiSystems.from that converts from this format to MultiSystems.
from_system(file_name, **kwargs)	Implement System.from that converts from this format to System.
get_formats()	Get all registered formats.
get_from_methods()	Get all registered from methods.
get_to_methods()	Get all registered to methods.
mix_system(*system, type_map, **kwargs)	Mix the systems into mixed_type ones according to the unified given type_map.
post(func_name)	Register a post function for from method.
register(key)	Register a format plugin.
register_from(key)	Register a from method if the target method name is not default.
register_to(key)	Register a to method if the target method name is not default.
to_bond_order_system(data, rdkit_mol, *args, ...)	Implement BondOrderSystem.to that converts from BondOrderSystem to this format.
to_labeled_system(data, *args, **kwargs)	Implement LabeledSystem.to that converts from LabeledSystem to this format.
to_multi_systems(formulas, directory, **kwargs)	Implement MultiSystems.to that converts from MultiSystems to this format.
to_system(data, file_name[, frame_idx])	Dump the system in vasp POSCAR format.

from_system(*file_name*, ***kwargs*)

Implement System.from that converts from this format to System.

Parameters

file_name
[str] file name, i.e. the first argument

****kwargs**
[dict] keyword arguments that will be passed from the method

Returns

data
[dict] system data, whose keys are defined in System.DTYPES

`to_system`(*data*, *file_name*, *frame_idx=0*, ***kwargs*)

Dump the system in vasp POSCAR format.

Parameters

data
[dict] The system data

file_name
[str] The output file name

frame_idx
[int] The index of the frame to dump

****kwargs**
[dict] other parameters

`class` dpdata.plugins.vasp.`VASPStringFormat`

Bases: *Format*

Methods

MultiModes()	File mode for MultiSystems.
from_bond_order_system(file_name, **kwargs)	Implement BondOrderSystem.from that converts from this format to BondOrderSystem.
from_labeled_system(file_name, **kwargs)	Implement LabeledSystem.from that converts from this format to LabeledSystem.
from_multi_systems(directory, **kwargs)	Implement MultiSystems.from that converts from this format to MultiSystems.
from_system(file_name, **kwargs)	Implement System.from that converts from this format to System.
get_formats()	Get all registered formats.
get_from_methods()	Get all registered from methods.
get_to_methods()	Get all registered to methods.
mix_system(*system, type_map, **kwargs)	Mix the systems into mixed_type ones according to the unified given type_map.
post(func_name)	Register a post function for from method.
register(key)	Register a format plugin.
register_from(key)	Register a from method if the target method name is not default.
register_to(key)	Register a to method if the target method name is not default.
to_bond_order_system(data, rdkit_mol, *args, ...)	Implement BondOrderSystem.to that converts from BondOrderSystem to this format.
to_labeled_system(data, *args, **kwargs)	Implement LabeledSystem.to that converts from LabeledSystem to this format.
to_multi_systems(formulas, directory, **kwargs)	Implement MultiSystems.to that converts from MultiSystems to this format.
to_system(data[, frame_idx])	Dump the system in vasp POSCAR format string.

to_system(*data*, *frame_idx=0*, ***kwargs*)

Dump the system in vasp POSCAR format string.

Parameters

data
[dict] The system data

frame_idx
[int] The index of the frame to dump

****kwargs**
[dict] other parameters

class dpdata.plugins.vasp.**VASPXMLFormat**

Bases: *Format*

Methods

MultiModes()	File mode for MultiSystems.
from_bond_order_system(file_name, **kwargs)	Implement BondOrderSystem.from that converts from this format to BondOrderSystem.
from_labeled_system(file_name[, begin, step])	Implement LabeledSystem.from that converts from this format to LabeledSystem.
from_multi_systems(directory, **kwargs)	Implement MultiSystems.from that converts from this format to MultiSystems.
from_system(file_name, **kwargs)	Implement System.from that converts from this format to System.
get_formats()	Get all registered formats.
get_from_methods()	Get all registered from methods.
get_to_methods()	Get all registered to methods.
mix_system(*system, type_map, **kwargs)	Mix the systems into mixed_type ones according to the unified given type_map.
post(func_name)	Register a post function for from method.
register(key)	Register a format plugin.
register_from(key)	Register a from method if the target method name is not default.
register_to(key)	Register a to method if the target method name is not default.
to_bond_order_system(data, rdkit_mol, *args, ...)	Implement BondOrderSystem.to that converts from BondOrderSystem to this format.
to_labeled_system(data, *args, **kwargs)	Implement LabeledSystem.to that converts from LabeledSystem to this format.
to_multi_systems(formulas, directory, **kwargs)	Implement MultiSystems.to that converts from MultiSystems to this format.
to_system(data, *args, **kwargs)	Implement System.to that converts from System to this format.

from_labeled_system(*file_name*, *begin=0*, *step=1*, ***kwargs*)

Implement LabeledSystem.from that converts from this format to LabeledSystem.

Parameters

file_name
[str] file name, i.e. the first argument

****kwargs**
[dict] keyword arguments that will be passed from the method

Returns

data
[dict] system data, whose keys are defined in LabeledSystem.DTYPES

dpdata.plugins.xyz module

class dpdata.plugins.xyz.**QuipGapXYZFormat**

Bases: *Format*

Methods

MultiModes()	File mode for MultiSystems.
from_bond_order_system(file_name, **kwargs)	Implement BondOrderSystem.from that converts from this format to BondOrderSystem.
from_labeled_system(data, **kwargs)	Implement LabeledSystem.from that converts from this format to LabeledSystem.
from_multi_systems(file_name, **kwargs)	Implement MultiSystems.from that converts from this format to MultiSystems.
from_system(file_name, **kwargs)	Implement System.from that converts from this format to System.
get_formats()	Get all registered formats.
get_from_methods()	Get all registered from methods.
get_to_methods()	Get all registered to methods.
mix_system(*system, type_map, **kwargs)	Mix the systems into mixed_type ones according to the unified given type_map.
post(func_name)	Register a post function for from method.
register(key)	Register a format plugin.
register_from(key)	Register a from method if the target method name is not default.
register_to(key)	Register a to method if the target method name is not default.
to_bond_order_system(data, rdkit_mol, *args, ...)	Implement BondOrderSystem.to that converts from BondOrderSystem to this format.
to_labeled_system(data, *args, **kwargs)	Implement LabeledSystem.to that converts from LabeledSystem to this format.
to_multi_systems(formulas, directory, **kwargs)	Implement MultiSystems.to that converts from MultiSystems to this format.
to_system(data, *args, **kwargs)	Implement System.to that converts from System to this format.

from_labeled_system(*data*, ***kwargs*)

Implement LabeledSystem.from that converts from this format to LabeledSystem.

Parameters

file_name
[str] file name, i.e. the first argument

****kwargs**
[dict] keyword arguments that will be passed from the method

Returns

data
[dict] system data, whose keys are defined in LabeledSystem.DTYPES

from_multi_systems(*file_name*, ***kwargs*)

Implement MultiSystems.from that converts from this format to MultiSystems.

By default, this method follows MultiMode to implement the conversion.

Parameters

directory
[str] directory of system

****kwargs**
[dict] keyword arguments that will be passed from the method

Returns

filenames: list[str]
list of filenames

class dpdata.plugins.xyz.**XYZFormat**

Bases: *Format*

XYZ foramt.

Examples

```
>>> s.to("xyz", "a.xyz")
```

Methods

MultiModes()	File mode for MultiSystems.
from_bond_order_system(file_name, **kwargs)	Implement BondOrderSystem.from that converts from this format to BondOrderSystem.
from_labeled_system(file_name, **kwargs)	Implement LabeledSystem.from that converts from this format to LabeledSystem.
from_multi_systems(directory, **kwargs)	Implement MultiSystems.from that converts from this format to MultiSystems.
from_system(file_name, **kwargs)	Implement System.from that converts from this format to System.
get_formats()	Get all registered formats.
get_from_methods()	Get all registered from methods.
get_to_methods()	Get all registered to methods.
mix_system(*system, type_map, **kwargs)	Mix the systems into mixed_type ones according to the unified given type_map.
post(func_name)	Register a post function for from method.
register(key)	Register a format plugin.
register_from(key)	Register a from method if the target method name is not default.
register_to(key)	Register a to method if the target method name is not default.
to_bond_order_system(data, rdkit_mol, *args, ...)	Implement BondOrderSystem.to that converts from BondOrderSystem to this format.
to_labeled_system(data, *args, **kwargs)	Implement LabeledSystem.to that converts from LabeledSystem to this format.
to_multi_systems(formulas, directory, **kwargs)	Implement MultiSystems.to that converts from MultiSystems to this format.
to_system(data, file_name, **kwargs)	Implement System.to that converts from System to this format.

from_system(*file_name*, ***kwargs*)

Implement System.from that converts from this format to System.

Parameters

file_name
[str] file name, i.e. the first argument

****kwargs**
[dict] keyword arguments that will be passed from the method

Returns

data
[dict] system data, whose keys are defined in System.DTYPES

to_system(*data*, *file_name*, ***kwargs*)

Implement System.to that converts from System to this format.

Parameters

data
[dict] system data, whose keys are defined in System.DTYPES

***args**
[list] arguments that will be passed from the method

****kwargs**
[dict] keyword arguments that will be passed from the method

dpdata.psi4 package

Submodules

dpdata.psi4.input module

dpdata.psi4.input.**write_psi4_input**(*types: ndarray*, *coords: ndarray*, *method: str*, *basis: str*, *charge: int = 0*, *multiplicity: int = 1*) → str

Write Psi4 input file.

Parameters

types
[np.ndarray] atomic symbols

coords
[np.ndarray] atomic coordinates

method
[str] computational method

basis
[str] basis set; see https://psicode.org/psi4manual/master/basissets_tables.html

charge
[int, default=0] charge of system

multiplicity
[int, default=1] multiplicity of system

Returns

str
content of Psi4 input file

dpdata.psi4.output module

dpdata.psi4.output.**read_psi4_output**(*fn: str*) → Tuple[str, ndarray, float, ndarray]

Read from Psi4 output.

Note that both the energy and the gradient should be printed.

Parameters

fn
[str] file name

Returns

str
atomic symbols

np.ndarray
atomic coordinates

float
total potential energy

np.ndarray
atomic forces

dpdata.pwmat package

Submodules

dpdata.pwmat.atomconfig module

dpdata.pwmat.atomconfig.**from_system_data**(*system*, *f_idx=0*, *skip_zeros=True*)

dpdata.pwmat.atomconfig.**to_system_data**(*lines*)

dpdata.pwmat.movement module

dpdata.pwmat.movement.**analyze_block**(*lines*, *ntot*, *nelm*)

dpdata.pwmat.movement.**get_frames**(*fname*, *begin=0*, *step=1*, *convergence_check=True*)

dpdata.pwmat.movement.**get_movement_block**(*fp*)

dpdata.pwmat.movement.**system_info**(*lines*, *type_idx_zero=False*)

dpdata.pymatgen package

Submodules

dpdata.pymatgen.molecule module

dpdata.pymatgen.molecule.**to_system_data**(*file_name*, *protect_layer=9*)

dpdata.qe package

Submodules

dpdata.qe.scf module

dpdata.qe.scf.**get_block**(*lines*, *keyword*, *skip=0*)

dpdata.qe.scf.**get_cell**(*lines*)

dpdata.qe.scf.**get_coords**(*lines*, *cell*)

dpdata.qe.scf.**get_energy**(*lines*)

dpdata.qe.scf.**get_force**(*lines*, *natoms*)

dpdata.qe.scf.**get_frame**(*fname*)

dpdata.qe.scf.**get_stress**(*lines*)

dpdata.qe.traj module

dpdata.qe.traj.**convert_celldm**(*ibrav*, *celldm*)

dpdata.qe.traj.**load_atom_names**(*lines*, *ntypes*)

dpdata.qe.traj.**load_atom_types**(*lines*, *natoms*, *atom_names*)

dpdata.qe.traj.**load_block**(*lines*, *key*, *nlines*)

dpdata.qe.traj.**load_cell_parameters**(*lines*)

dpdata.qe.traj.**load_celldm**(*lines*)

dpdata.qe.traj.**load_data**(*fname*, *natoms*, *begin=0*, *step=1*, *convert=1.0*)

dpdata.qe.traj.**load_energy**(*fname*, *begin=0*, *step=1*)

dpdata.qe.traj.**load_key**(*lines*, *key*)

dpdata.qe.traj.**load_param_file**(*fname*)

dpdata.qe.traj.**to_system_data**(*input_name*, *prefix*, *begin=0*, *step=1*)

dpdata.qe.traj.**to_system_label**(*input_name*, *prefix*, *begin=0*, *step=1*)

dpdata.rdkit package

Submodules

dpdata.rdkit.sanitize module

exception dpdata.rdkit.sanitize.**SanitizeError**(*content='Sanitization Failed.'*)

Bases: Exception

class dpdata.rdkit.sanitize.**Sanitizer**(*level='medium'*, *raise_errors=True*, *verbose=False*)

Bases: object

Methods

sanitize(mol)	Sanitize mol according to *self.level*.

sanitize(*mol*)

Sanitize mol according to *self.level*. If failed, return None.

dpdata.rdkit.sanitize.**assign_formal_charge_for_atom**(*atom*, *verbose=False*)

Assigen formal charge according to 8-electron rule for element B,C,N,O,S,P,As.

dpdata.rdkit.sanitize.**contain_hetero_aromatic**(*mol*)

dpdata.rdkit.sanitize.**convert_by_obabel**(*mol*, *cache_dir='/home/docs/checkouts/readthedocs.org/user_builds/dpdata/checkouts/s*
obabel_path='obabel')

dpdata.rdkit.sanitize.**get_explicit_valence**(*atom*, *verbose=False*)

dpdata.rdkit.sanitize.**get_terminal_NR2s**(*atom*)

dpdata.rdkit.sanitize.**get_terminal_oxygens**(*atom*)

dpdata.rdkit.sanitize.**is_terminal_NR2**(*N_atom*)

dpdata.rdkit.sanitize.**is_terminal_nitrogen**(*N_atom*)

dpdata.rdkit.sanitize.**is_terminal_oxygen**(*O_atom*)

dpdata.rdkit.sanitize.**kekulize_aromatic_heterocycles**(*mol_in*, *assign_formal_charge=True*, *sanitize=True*)

dpdata.rdkit.sanitize.**mol_edit_log**(*mol*, *i*, *j*)

dpdata.rdkit.sanitize.**print_atoms**(*mol*)

dpdata.rdkit.sanitize.**print_bonds**(*mol*)

dpdata.rdkit.sanitize.**regularize_carbon_bond_order**(*atom*, *verbose=True*)

dpdata.rdkit.sanitize.**regularize_formal_charges**(*mol*, *sanitize=True*, *verbose=False*)

Regularize formal charges of atoms.

dpdata.rdkit.sanitize.**regularize_nitrogen_bond_order**(*atom*, *verbose=True*)

dpdata.rdkit.sanitize.**sanitize_carboxyl**(*mol*)

dpdata.rdkit.sanitize.**sanitize_carboxyl_Catom**(*C_atom*, *verbose=True*)

dpdata.rdkit.sanitize.**sanitize_guanidine**(*mol*)

dpdata.rdkit.sanitize.**sanitize_guanidine_Catom**(*C_atom*, *verbose=True*)

dpdata.rdkit.sanitize.**sanitize_mol**(*mol*, *verbose=False*)

dpdata.rdkit.sanitize.**sanitize_nitrine_Natom**(*atom*, *verbose=True*)

dpdata.rdkit.sanitize.**sanitize_nitro**(*mol*)

dpdata.rdkit.sanitize.**sanitize_nitro_Natom**(*N_atom*, *verbose=True*)

dpdata.rdkit.sanitize.**sanitize_phosphate**(*mol*)

dpdata.rdkit.sanitize.**sanitize_phosphate_Patom**(*P_atom*, *verbose=True*)

dpdata.rdkit.sanitize.**sanitize_sulfate**(*mol*)

dpdata.rdkit.sanitize.**sanitize_sulfate_Satom**(*S_atom*, *verbose=True*)

dpdata.rdkit.sanitize.**super_sanitize_mol**(*mol*, *name=None*, *verbose=True*)

dpdata.rdkit.utils module

dpdata.rdkit.utils.**check_molecule_list**(*mols*)

dpdata.rdkit.utils.**check_same_atom**(*atom_1*, *atom_2*)

dpdata.rdkit.utils.**check_same_molecule**(*mol_1*, *mol_2*)

dpdata.rdkit.utils.**combine_molecules**(*mols*)

dpdata.rdkit.utils.**mol_to_system_data**(*mol*)

dpdata.rdkit.utils.**system_data_to_mol**(*data*)

dpdata.siesta package

Submodules

dpdata.siesta.aiMD_output module

dpdata.siesta.aiMD_output.**covert_dimension**(*arr*, *num*)

dpdata.siesta.aiMD_output.**extract_keyword**(*fout*, *keyword*, *down_line_num*, *begin_column*, *read_column_num*, *is_repeated_read*, *column_num*)

dpdata.siesta.aiMD_output.**get_aiMD_frame**(*fname*)

dpdata.siesta.aiMD_output.**get_atom_name**(*fout*)

dpdata.siesta.aiMD_output.**get_atom_numbs**(*atomtypes*)

dpdata.siesta.aiMD_output.**get_atom_types**(*fout*, *atomnums*)

dpdata.siesta.aiMD_output.**get_single_line_tail**(*fin*, *keyword*, *num=1*)

dpdata.siesta.aiMD_output.**get_virial**(*fout*, *cell*)

dpdata.siesta.aiMD_output.**obtain_nframe**(*fname*)

dpdata.siesta.output module

dpdata.siesta.output.**extract_keyword**(*fout*, *keyword*, *down_line_num*, *begin_column*, *column_num*)

dpdata.siesta.output.**get_atom_name**(*fout*)

dpdata.siesta.output.**get_atom_numbs**(*atomtypes*)

dpdata.siesta.output.**get_atom_types**(*fout*, *atomnums*)

dpdata.siesta.output.**get_single_line_tail**(*fin*, *keyword*, *num=1*)

dpdata.siesta.output.**get_virial**(*fout*, *cells*)

dpdata.siesta.output.**obtain_frame**(*fname*)

dpdata.vasp package

Submodules

dpdata.vasp.outcar module

dpdata.vasp.outcar.**analyze_block**(*lines*, *ntot*, *nelm*, *ml=False*)

dpdata.vasp.outcar.**get_frames**(*fname*, *begin=0*, *step=1*, *ml=False*, *convergence_check=True*)

dpdata.vasp.outcar.**get_outcar_block**(*fp*, *ml=False*)

dpdata.vasp.outcar.**system_info**(*lines*, *type_idx_zero=False*)

dpdata.vasp.poscar module

dpdata.vasp.poscar.**from_system_data**(*system*, *f_idx=0*, *skip_zeros=True*)

dpdata.vasp.poscar.**to_system_data**(*lines*)

dpdata.vasp.xml module

dpdata.vasp.xml.**analyze**(*fname*, *type_idx_zero=False*, *begin=0*, *step=1*)

Deal with broken xml file.

dpdata.vasp.xml.**analyze_atominfo**(*atominfo_xml*)

dpdata.vasp.xml.**analyze_calculation**(*cc*)

dpdata.vasp.xml.**check_name**(*item*, *name*)

dpdata.vasp.xml.**formulate_config**(*eles*, *types*, *posi*, *cell*, *ener*, *forc*, *strs_*)

dpdata.vasp.xml.**get_varray**(*varray*)

dpdata.xyz package

Submodules

dpdata.xyz.quip_gap_xyz module

class dpdata.xyz.quip_gap_xyz.**QuipGapxyzSystems**(*file_name*)

Bases: `object`

deal with QuipGapxyzFile.

Methods

get_block_generator
handle_single_xyz_frame

get_block_generator()

static handle_single_xyz_frame(*lines*)

dpdata.xyz.xyz module

dpdata.xyz.xyz.**coord_to_xyz**(*coord: ndarray*, *types: list*) → str

Convert coordinates and types to xyz format.

Parameters

coord
[np.ndarray] coordinates, Nx3 array

types
[list] list of types

Returns

str
xyz format string

Examples

```
>>> coord_to_xyz(np.ones((1,3)), ["C"])
1
```

C 1.000000 1.000000 1.000000

dpdata.xyz.xyz.**xyz_to_coord**(*xyz: str*) → Tuple[ndarray, list]

Convert xyz format to coordinates and types.

Parameters

xyz
[str] xyz format string

Returns

coords
[np.ndarray] coordinates, Nx3 array

types
[list] list of types

6.1.2 Submodules

6.1.3 dpdata.ase_calculator module

6.1.4 dpdata.bond_order_system module

class dpdata.bond_order_system.**BondOrderSystem**(*file_name=None*, *fmt='auto'*, *type_map=None*, *begin=0*, *step=1*, *data=None*, *rdkit_mol=None*, *sanitize_level='medium'*, *raise_errors=True*, *verbose=False*, ***kwargs*)

Bases: *System*

The system with chemical bond and formal charges information.

For example, a labeled methane system named *d_example* has one molecule (5 atoms, 4 bonds) and *n_frames* frames. The bond order and formal charge information can be accessed by

- ***d_example['bonds']***
  [a numpy array of size 4 x 3, and] the first column represents the index of begin atom, the second column represents the index of end atom, the third columen represents the bond order:

  1 - single bond, 2 - double bond, 3 - triple bond, 1.5 - aromatic bond

- *d_example['formal_charges']* : a numpy array of size 5 x 1

Attributes

formula
Return the formula of this system, like C3H5O2.

formula_hash
Return the hash of the formula of this system.

nopbc

short_formula
Return the short formula of this system.

short_name
Return the short name of this system (no more than 255 bytes), in the following order: - formula - short_formula - formula_hash.

uniq_formula
Return the uniq_formula of this system.

Methods

add_atom_names(atom_names)	Add atom_names that do not exist.
append(system)	Append a system to this system.
apply_pbc()	Append periodic boundary condition.
as_dict()	Returns data dict of System instance.
check_data()	Check if data is correct.
check_type_map(type_map)	Assign atom_names to type_map if type_map is given and different from atom_names.
convert_to_mixed_type([type_map])	Convert the data dict to mixed type format structure, in order to append systems with different formula but the same number of atoms.

continues on next page

Table 5 – continued from previous page

copy()	Returns a copy of the system.
dump(filename[, indent])	Dump .json or .yaml file.
extend(systems)	Extend a system list to this system.
from_3dmol(file_name, **kwargs)	Read data from *dpdata.plugins.3dmol.Py3DMolFormat* format.
from_abacus_lcao_md(file_name, **kwargs)	Read data from *dpdata.plugins.abacus.AbacusMDFormat* format.
from_abacus_lcao_relax(file_name, **kwargs)	Read data from *dpdata.plugins.abacus.AbacusRelaxFormat* format.
from_abacus_lcao_scf(file_name, **kwargs)	Read data from *dpdata.plugins.abacus.AbacusSCFFormat* format.
from_abacus_md(file_name, **kwargs)	Read data from *dpdata.plugins.abacus.AbacusMDFormat* format.
from_abacus_pw_md(file_name, **kwargs)	Read data from *dpdata.plugins.abacus.AbacusMDFormat* format.
from_abacus_pw_relax(file_name, **kwargs)	Read data from *dpdata.plugins.abacus.AbacusRelaxFormat* format.
from_abacus_pw_scf(file_name, **kwargs)	Read data from *dpdata.plugins.abacus.AbacusSCFFormat* format.
from_abacus_relax(file_name, **kwargs)	Read data from *dpdata.plugins.abacus.AbacusRelaxFormat* format.
from_abacus_scf(file_name, **kwargs)	Read data from *dpdata.plugins.abacus.AbacusSCFFormat* format.
from_abacus_stru(file_name, **kwargs)	Read data from *dpdata.plugins.abacus.AbacusSTRUFormat* format.
from_amber_md(file_name, **kwargs)	Read data from *dpdata.plugins.amber.AmberMDFormat* format.
from_ase_structure(file_name, **kwargs)	Read data from *dpdata.plugins.ase.ASEStructureFormat* format.
from_ase_traj(file_name, **kwargs)	Read data from *dpdata.plugins.ase.ASETrajFormat* format.
from_atomconfig(file_name, **kwargs)	Read data from *dpdata.plugins.pwmat.PwmatAtomconfigFormat* format.
from_contcar(file_name, **kwargs)	Read data from *dpdata.plugins.vasp.VASPPoscarFormat* format.
from_cp2k_aimd_output(file_name, **kwargs)	Read data from *dpdata.plugins.cp2k.CP2KAIMDOutputFormat* format.
from_cp2k_output(file_name, **kwargs)	Read data from *dpdata.plugins.cp2k.CP2KOutputFormat* format.
from_deepmd(file_name, **kwargs)	Read data from *dpdata.plugins.deepmd.DeePMDRawFormat* format.
from_deepmd_comp(file_name, **kwargs)	Read data from *dpdata.plugins.deepmd.DeePMDCompFormat* format.
from_deepmd_hdf5(file_name, **kwargs)	Read data from *dpdata.plugins.deepmd.DeePMDHDF5Format* format.
from_deepmd_npy(file_name, **kwargs)	Read data from *dpdata.plugins.deepmd.DeePMDCompFormat* format.
from_deepmd_npy_mixed(file_name, **kwargs)	Read data from *dpdata.plugins.deepmd.DeePMDMixedFormat* format.
from_deepmd_raw(file_name, **kwargs)	Read data from *dpdata.plugins.deepmd.DeePMDRawFormat* format.

continues on next page

Table 5 – continued from previous page

from_dftbplus(file_name, **kwargs)	Read data from *dpdata.plugins.dftbplus.DFTBplusFormat* format.
from_dict(d)	**param d** Dict representation.
from_dump(file_name, **kwargs)	Read data from *dpdata.plugins.lammps.LAMMPSDumpFormat* format.
from_fhi_aims_md(file_name, **kwargs)	Read data from *dpdata.plugins.fhi_aims.FhiMDFormat* format.
from_fhi_aims_output(file_name, **kwargs)	Read data from *dpdata.plugins.fhi_aims.FhiMDFormat* format.
from_fhi_aims_scf(file_name, **kwargs)	Read data from *dpdata.plugins.fhi_aims.FhiSCFFormat* format.
from_finalconfig(file_name, **kwargs)	Read data from *dpdata.plugins.pwmat.PwmatAtomconfigFormat* format.
from_gaussian_gjf(file_name, **kwargs)	Read data from *dpdata.plugins.gaussian.GaussiaGJFFormat* format.
from_gaussian_log(file_name, **kwargs)	Read data from *dpdata.plugins.gaussian.GaussianLogFormat* format.
from_gaussian_md(file_name, **kwargs)	Read data from *dpdata.plugins.gaussian.GaussianMDFormat* format.
from_gro(file_name, **kwargs)	Read data from *dpdata.plugins.gromacs.GromacsGroFormat* format.
from_gromacs_gro(file_name, **kwargs)	Read data from *dpdata.plugins.gromacs.GromacsGroFormat* format.
from_lammps_dump(file_name, **kwargs)	Read data from *dpdata.plugins.lammps.LAMMPSDumpFormat* format.
from_lammps_lmp(file_name, **kwargs)	Read data from *dpdata.plugins.lammps.LAMMPSLmpFormat* format.
from_list(file_name, **kwargs)	Read data from *dpdata.plugins.list.ListFormat* format.
from_lmp(file_name, **kwargs)	Read data from *dpdata.plugins.lammps.LAMMPSLmpFormat* format.
from_mlmd(file_name, **kwargs)	Read data from *dpdata.plugins.pwmat.PwmatOutputFormat* format.
from_mol(file_name, **kwargs)	Read data from *dpdata.plugins.rdkit.MolFormat* format.
from_mol_file(file_name, **kwargs)	Read data from *dpdata.plugins.rdkit.MolFormat* format.
from_movement(file_name, **kwargs)	Read data from *dpdata.plugins.pwmat.PwmatOutputFormat* format.
from_n2p2(file_name, **kwargs)	Read data from *dpdata.plugins.n2p2.N2P2Format* format.
from_openmx_md(file_name, **kwargs)	Read data from *dpdata.plugins.openmx.OPENMXFormat* format.
from_orca_spout(file_name, **kwargs)	Read data from *dpdata.plugins.orca.ORCASPOutFormat* format.
from_outcar(file_name, **kwargs)	Read data from *dpdata.plugins.vasp.VASPOutcarFormat* format.
from_poscar(file_name, **kwargs)	Read data from *dpdata.plugins.vasp.VASPPoscarFormat* format.

continues on next page

Table 5 – continued from previous page

from_psi4_inp(file_name, **kwargs)	Read data from *dpdata.plugins.psi4.PSI4InputFormat* format.
from_psi4_out(file_name, **kwargs)	Read data from *dpdata.plugins.psi4.PSI4OutFormat* format.
from_pwmat_atomconfig(file_name, **kwargs)	Read data from *dpdata.plugins.pwmat.PwmatAtomconfigFormat* format.
from_pwmat_finalconfig(file_name, **kwargs)	Read data from *dpdata.plugins.pwmat.PwmatAtomconfigFormat* format.
from_pwmat_mlmd(file_name, **kwargs)	Read data from *dpdata.plugins.pwmat.PwmatOutputFormat* format.
from_pwmat_movement(file_name, **kwargs)	Read data from *dpdata.plugins.pwmat.PwmatOutputFormat* format.
from_pwmat_output(file_name, **kwargs)	Read data from *dpdata.plugins.pwmat.PwmatOutputFormat* format.
from_pymatgen_computedstructureentry(...)	Read data from *dpdata.plugins.pymatgen.PyMatgenCSEFormat* format.
from_pymatgen_molecule(file_name, **kwargs)	Read data from *dpdata.plugins.pymatgen.PyMatgenMoleculeFormat* format.
from_pymatgen_structure(file_name, **kwargs)	Read data from *dpdata.plugins.pymatgen.PyMatgenStructureFormat* format.
from_qe_cp_traj(file_name, **kwargs)	Read data from *dpdata.plugins.qe.QECPTrajFormat* format.
from_qe_pw_scf(file_name, **kwargs)	Read data from *dpdata.plugins.qe.QECPPWSCFFormat* format.
from_quip_gap_xyz(file_name, **kwargs)	Read data from *dpdata.plugins.xyz.QuipGapXYZFormat* format.
from_quip_gap_xyz_file(file_name, **kwargs)	Read data from *dpdata.plugins.xyz.QuipGapXYZFormat* format.
from_rdkit_mol(rdkit_mol)	Initialize from a rdkit.Chem.rdchem.Mol object.
from_sdf(file_name, **kwargs)	Read data from *dpdata.plugins.rdkit.SdfFormat* format.
from_sdf_file(file_name, **kwargs)	Read data from *dpdata.plugins.rdkit.SdfFormat* format.
from_siesta_aiMD_output(file_name, **kwargs)	Read data from *dpdata.plugins.siesta.SiestaAIMDOutputFormat* format.
from_siesta_aimd_output(file_name, **kwargs)	Read data from *dpdata.plugins.siesta.SiestaAIMDOutputFormat* format.
from_siesta_output(file_name, **kwargs)	Read data from *dpdata.plugins.siesta.SiestaOutputFormat* format.
from_sqm_in(file_name, **kwargs)	Read data from *dpdata.plugins.amber.SQMINFormat* format.
from_sqm_out(file_name, **kwargs)	Read data from *dpdata.plugins.amber.SQMOutFormat* format.
from_stru(file_name, **kwargs)	Read data from *dpdata.plugins.abacus.AbacusSTRUFormat* format.
from_vasp_contcar(file_name, **kwargs)	Read data from *dpdata.plugins.vasp.VASPPoscarFormat* format.
from_vasp_outcar(file_name, **kwargs)	Read data from *dpdata.plugins.vasp.VASPOutcarFormat* format.
from_vasp_poscar(file_name, **kwargs)	Read data from *dpdata.plugins.vasp.VASPPoscarFormat* format.

continues on next page

Table 5 – continued from previous page

from_vasp_string(file_name, **kwargs)	Read data from *dpdata.plugins.vasp.VASPStringFormat* format.
from_vasp_xml(file_name, **kwargs)	Read data from *dpdata.plugins.vasp.VASPXMLFormat* format.
from_xml(file_name, **kwargs)	Read data from *dpdata.plugins.vasp.VASPXMLFormat* format.
from_xyz(file_name, **kwargs)	Read data from *dpdata.plugins.xyz.XYZFormat* format.
get_atom_names()	Returns name of atoms.
get_atom_numbs()	Returns number of atoms.
get_atom_types()	Returns type of atoms.
get_bond_order(begin_atom_idx, end_atom_idx)	Return the bond order between given atoms.
get_charge()	Return the total formal charge of the moleclue.
get_formal_charges()	Return the formal charges on each atom.
get_mol()	Return the rdkit.Mol object.
get_natoms()	Returns total number of atoms in the system.
get_nbonds()	Return the number of bonds.
get_nframes()	Returns number of frames in the system.
get_ntypes()	Returns total number of atom types in the system.
load(filename)	Rebuild System obj.
map_atom_types([type_map])	Map the atom types of the system.
minimize(*args, minimizer, **kwargs)	Minimize the geometry.
perturb(pert_num, cell_pert_fraction, ...[, ...])	Perturb each frame in the system randomly.
pick_atom_idx(idx[, nopbc])	Pick atom index.
pick_by_amber_mask(param, maskstr[, ...])	Pick atoms by amber mask.
predict(*args[, driver])	Predict energies and forces by a driver.
register_data_type(*data_type)	Register data type.
remove_atom_names(atom_names)	Remove atom names and all such atoms.
remove_pbc([protect_layer])	This method does NOT delete the definition of the cells, it (1) revises the cell to a cubic cell and ensures that the cell boundary to any atom in the system is no less than *protect_layer* (2) translates the system such that the center-of-geometry of the system locates at the center of the cell.
replicate(ncopy)	Replicate the each frame in the system in 3 dimensions.
shuffle()	Shuffle frames randomly.
sort_atom_names([type_map])	Sort atom_names of the system and reorder atom_numbs and atom_types accoarding to atom_names.
sort_atom_types()	Sort atom types.
sub_system(f_idx)	Construct a subsystem from the system.
to(fmt, *args, **kwargs)	Dump systems to the specific format.
to_3dmol(*args, **kwargs)	Dump data to *dpdata.plugins.3dmol.Py3DMolFormat* format.
to_abacus_lcao_md(*args, **kwargs)	Dump data to *dpdata.plugins.abacus.AbacusMDFormat* format.
to_abacus_lcao_relax(*args, **kwargs)	Dump data to *dpdata.plugins.abacus.AbacusRelaxFormat* format.
to_abacus_lcao_scf(*args, **kwargs)	Dump data to *dpdata.plugins.abacus.AbacusSCFFormat* format.

continues on next page

Table 5 – continued from previous page

to_abacus_md(*args, **kwargs)	Dump data to *dpdata.plugins.abacus.AbacusMDFormat* format.
to_abacus_pw_md(*args, **kwargs)	Dump data to *dpdata.plugins.abacus.AbacusMDFormat* format.
to_abacus_pw_relax(*args, **kwargs)	Dump data to *dpdata.plugins.abacus.AbacusRelaxFormat* format.
to_abacus_pw_scf(*args, **kwargs)	Dump data to *dpdata.plugins.abacus.AbacusSCFFormat* format.
to_abacus_relax(*args, **kwargs)	Dump data to *dpdata.plugins.abacus.AbacusRelaxFormat* format.
to_abacus_scf(*args, **kwargs)	Dump data to *dpdata.plugins.abacus.AbacusSCFFormat* format.
to_abacus_stru(*args, **kwargs)	Dump data to *dpdata.plugins.abacus.AbacusSTRUFormat* format.
to_amber_md(*args, **kwargs)	Dump data to *dpdata.plugins.amber.AmberMDFormat* format.
to_ase_structure(*args, **kwargs)	Dump data to *dpdata.plugins.ase.ASEStructureFormat* format.
to_ase_traj(*args, **kwargs)	Dump data to *dpdata.plugins.ase.ASETrajFormat* format.
to_atomconfig(*args, **kwargs)	Dump data to *dpdata.plugins.pwmat.PwmatAtomconfigFormat* format.
to_contcar(*args, **kwargs)	Dump data to *dpdata.plugins.vasp.VASPPoscarFormat* format.
to_cp2k_aimd_output(*args, **kwargs)	Dump data to *dpdata.plugins.cp2k.CP2KAIMDOutputFormat* format.
to_cp2k_output(*args, **kwargs)	Dump data to *dpdata.plugins.cp2k.CP2KOutputFormat* format.
to_deepmd(*args, **kwargs)	Dump data to *dpdata.plugins.deepmd.DeePMDRawFormat* format.
to_deepmd_comp(*args, **kwargs)	Dump data to *dpdata.plugins.deepmd.DeePMDCompFormat* format.
to_deepmd_hdf5(*args, **kwargs)	Dump data to *dpdata.plugins.deepmd.DeePMDHDF5Format* format.
to_deepmd_npy(*args, **kwargs)	Dump data to *dpdata.plugins.deepmd.DeePMDCompFormat* format.
to_deepmd_npy_mixed(*args, **kwargs)	Dump data to *dpdata.plugins.deepmd.DeePMDMixedFormat* format.
to_deepmd_raw(*args, **kwargs)	Dump data to *dpdata.plugins.deepmd.DeePMDRawFormat* format.
to_dftbplus(*args, **kwargs)	Dump data to *dpdata.plugins.dftbplus.DFTBplusFormat* format.
to_dump(*args, **kwargs)	Dump data to *dpdata.plugins.lammps.LAMMPSDumpFormat* format.
to_fhi_aims_md(*args, **kwargs)	Dump data to *dpdata.plugins.fhi_aims.FhiMDFormat* format.
to_fhi_aims_output(*args, **kwargs)	Dump data to *dpdata.plugins.fhi_aims.FhiMDFormat* format.
to_fhi_aims_scf(*args, **kwargs)	Dump data to *dpdata.plugins.fhi_aims.FhiSCFFormat* format.
to_finalconfig(*args, **kwargs)	Dump data to *dpdata.plugins.pwmat.PwmatAtomconfigFormat* format.

continues on next page

Table 5 – continued from previous page

to_gaussian_gjf(*args, **kwargs)	Dump data to *dpdata.plugins.gaussian.GaussiaGJFFormat* format.
to_gaussian_log(*args, **kwargs)	Dump data to *dpdata.plugins.gaussian.GaussianLogFormat* format.
to_gaussian_md(*args, **kwargs)	Dump data to *dpdata.plugins.gaussian.GaussianMDFormat* format.
to_gro(*args, **kwargs)	Dump data to *dpdata.plugins.gromacs.GromacsGroFormat* format.
to_gromacs_gro(*args, **kwargs)	Dump data to *dpdata.plugins.gromacs.GromacsGroFormat* format.
to_json()	Returns a json string representation of the MSONable object.
to_lammps_dump(*args, **kwargs)	Dump data to *dpdata.plugins.lammps.LAMMPSDumpFormat* format.
to_lammps_lmp(*args, **kwargs)	Dump data to *dpdata.plugins.lammps.LAMMPSLmpFormat* format.
to_list(*args, **kwargs)	Dump data to *dpdata.plugins.list.ListFormat* format.
to_lmp(*args, **kwargs)	Dump data to *dpdata.plugins.lammps.LAMMPSLmpFormat* format.
to_mlmd(*args, **kwargs)	Dump data to *dpdata.plugins.pwmat.PwmatOutputFormat* format.
to_mol(*args, **kwargs)	Dump data to *dpdata.plugins.rdkit.MolFormat* format.
to_mol_file(*args, **kwargs)	Dump data to *dpdata.plugins.rdkit.MolFormat* format.
to_movement(*args, **kwargs)	Dump data to *dpdata.plugins.pwmat.PwmatOutputFormat* format.
to_n2p2(*args, **kwargs)	Dump data to *dpdata.plugins.n2p2.N2P2Format* format.
to_openmx_md(*args, **kwargs)	Dump data to *dpdata.plugins.openmx.OPENMXFormat* format.
to_orca_spout(*args, **kwargs)	Dump data to *dpdata.plugins.orca.ORCASPOutFormat* format.
to_outcar(*args, **kwargs)	Dump data to *dpdata.plugins.vasp.VASPOutcarFormat* format.
to_poscar(*args, **kwargs)	Dump data to *dpdata.plugins.vasp.VASPPoscarFormat* format.
to_psi4_inp(*args, **kwargs)	Dump data to *dpdata.plugins.psi4.PSI4InputFormat* format.
to_psi4_out(*args, **kwargs)	Dump data to *dpdata.plugins.psi4.PSI4OutFormat* format.
to_pwmat_atomconfig(*args, **kwargs)	Dump data to *dpdata.plugins.pwmat.PwmatAtomconfigFormat* format.
to_pwmat_finalconfig(*args, **kwargs)	Dump data to *dpdata.plugins.pwmat.PwmatAtomconfigFormat* format.
to_pwmat_mlmd(*args, **kwargs)	Dump data to *dpdata.plugins.pwmat.PwmatOutputFormat* format.
to_pwmat_movement(*args, **kwargs)	Dump data to *dpdata.plugins.pwmat.PwmatOutputFormat* format.
to_pwmat_output(*args, **kwargs)	Dump data to *dpdata.plugins.pwmat.PwmatOutputFormat* format.

continues on next page

Table 5 – continued from previous page

to_pymatgen_ComputedStructureEntry(*args, ...)	Dump data to *dpdata.plugins.pymatgen.PyMatgenCSEFormat* format.
to_pymatgen_computedstructureentry(*args, ...)	Dump data to *dpdata.plugins.pymatgen.PyMatgenCSEFormat* format.
to_pymatgen_molecule(*args, **kwargs)	Dump data to *dpdata.plugins.pymatgen.PyMatgenMoleculeFormat* format.
to_pymatgen_structure(*args, **kwargs)	Dump data to *dpdata.plugins.pymatgen.PyMatgenStructureFormat* format.
to_qe_cp_traj(*args, **kwargs)	Dump data to *dpdata.plugins.qe.QECPTrajFormat* format.
to_qe_pw_scf(*args, **kwargs)	Dump data to *dpdata.plugins.qe.QECPPWSCFFormat* format.
to_quip_gap_xyz(*args, **kwargs)	Dump data to *dpdata.plugins.xyz.QuipGapXYZFormat* format.
to_quip_gap_xyz_file(*args, **kwargs)	Dump data to *dpdata.plugins.xyz.QuipGapXYZFormat* format.
to_sdf(*args, **kwargs)	Dump data to *dpdata.plugins.rdkit.SdfFormat* format.
to_sdf_file(*args, **kwargs)	Dump data to *dpdata.plugins.rdkit.SdfFormat* format.
to_siesta_aimd_output(*args, **kwargs)	Dump data to *dpdata.plugins.siesta.SiestaAIMDOutputFormat* format.
to_siesta_output(*args, **kwargs)	Dump data to *dpdata.plugins.siesta.SiestaOutputFormat* format.
to_sqm_in(*args, **kwargs)	Dump data to *dpdata.plugins.amber.SQMINFormat* format.
to_sqm_out(*args, **kwargs)	Dump data to *dpdata.plugins.amber.SQMOutFormat* format.
to_stru(*args, **kwargs)	Dump data to *dpdata.plugins.abacus.AbacusSTRUFormat* format.
to_vasp_contcar(*args, **kwargs)	Dump data to *dpdata.plugins.vasp.VASPPoscarFormat* format.
to_vasp_outcar(*args, **kwargs)	Dump data to *dpdata.plugins.vasp.VASPOutcarFormat* format.
to_vasp_poscar(*args, **kwargs)	Dump data to *dpdata.plugins.vasp.VASPPoscarFormat* format.
to_vasp_string(*args, **kwargs)	Dump data to *dpdata.plugins.vasp.VASPStringFormat* format.
to_vasp_xml(*args, **kwargs)	Dump data to *dpdata.plugins.vasp.VASPXMLFormat* format.
to_xml(*args, **kwargs)	Dump data to *dpdata.plugins.vasp.VASPXMLFormat* format.
to_xyz(*args, **kwargs)	Dump data to *dpdata.plugins.xyz.XYZFormat* format.
unsafe_hash()	Returns an hash of the current object.
validate_monty_v1(_MSONable__input_value)	Pydantic validator with correct signature for pydantic v1.x
validate_monty_v2(_MSONable__input_value, _)	Pydantic validator with correct signature for pydantic v2.x

affine_map
apply_type_map
from_fmt
from_fmt_obj
replace
rot_frame_lower_triangular
rot_lower_triangular
to_fmt_obj

DTYPES = (<dpdata.data_type.DataType object>, <dpdata.data_type.DataType object>, <dpdata.data_type.DataType object>, <dpdata.data_type.DataType object>, <dpdata.data_type.DataType object>, <dpdata.data_type.DataType object>, <dpdata.data_type.DataType object>, <dpdata.data_type.DataType object>, <dpdata.data_type.DataType object>, <dpdata.data_type.DataType object>)

copy()

Returns a copy of the system.

from_fmt_obj(*fmtobj*, *file_name*, ***kwargs*)

from_rdkit_mol(*rdkit_mol*)

Initialize from a rdkit.Chem.rdchem.Mol object.

get_bond_order(*begin_atom_idx*, *end_atom_idx*)

Return the bond order between given atoms.

get_charge()

Return the total formal charge of the moleclue.

get_formal_charges()

Return the formal charges on each atom.

get_mol()

Return the rdkit.Mol object.

get_nbonds()

Return the number of bonds.

to_fmt_obj(*fmtobj*, **args*, ***kwargs*)

6.1.5 dpdata.cli module

Command line interface for dpdata.

dpdata.cli.**convert**(*, *from_file: str*, *from_format: str = 'auto'*, *to_file: str | None = None*, *to_format: str | None = None*, *no_labeled: bool = False*, *multi: bool = False*, *type_map: list | None = None*, ***kwargs*)

Convert files from one format to another one.

Parameters

from_file
[str] read data from a file

from_format
[str] the format of from_file

to_file
[str] dump data to a file

to_format
[str] the format of to_file

no_labeled
[bool] labels aren't provided

multi
[bool] the system contains multiple directories

type_map
[list] type map

****kwargs**
[dict] Additional arguments for the format.

dpdata.cli.**dpdata_cli**()

Dpdata cli.

Examples

```
$ dpdata -iposcar POSCAR -odeepmd/npy -O data -n
```

dpdata.cli.**dpdata_parser**() → ArgumentParser

Returns dpdata cli parser.

Returns

argparse.ArgumentParser
dpdata cli parser

6.1.6 dpdata.data_type module

class dpdata.data_type.**AnyInt**

Bases: int

AnyInt equals to any other integer.

Attributes

denominator
the denominator of a rational number in lowest terms

imag
the imaginary part of a complex number

numerator
the numerator of a rational number in lowest terms

real
the real part of a complex number

Methods

as_integer_ratio(/)	Return integer ratio.
bit_count(/)	Number of ones in the binary representation of the absolute value of self.
bit_length(/)	Number of bits necessary to represent self in binary.
conjugate	Returns self, the complex conjugate of any int.
from_bytes(/, bytes[, byteorder, signed])	Return the integer represented by the given array of bytes.
to_bytes(/[, length, byteorder, signed])	Return an array of bytes representing an integer.

class dpdata.data_type.**Axis**(*value*, *names=None*, ***, *module=None*, *qualname=None*, *type=None*, *start=1*, *boundary=None*)

Bases: Enum

Data axis.

NATOMS = 'natoms'

NBONDS = 'nbonds'

NFRAMES = 'nframes'

NTYPES = 'ntypes'

exception dpdata.data_type.**DataError**

Bases: Exception

Data is not correct.

class dpdata.data_type.**DataType**(*name: str*, *dtype: type*, *shape: Tuple[int,* Axis*] = None*, *required: bool = True*)

Bases: object

DataType represents a type of data, like coordinates, energies, etc.

Parameters

name
[str] name of data

dtype
[type or tuple[type]] data type, e.g. np.ndarray

shape
[tuple[int], optional] shape of data. Used when data is list or np.ndarray. Use Axis to represents numbers

required
[bool, default=True] whether this data is required

Methods

check(system)	Check if a system has correct data of this type.
real_shape(system)	Returns expected real shape of a system.

check(*system:* System)

Check if a system has correct data of this type.

Parameters

system
[System] checked system

Raises

DataError
type or shape of data is not correct

real_shape(*system:* System) → Tuple[int]

Returns expected real shape of a system.

dpdata.data_type.**get_data_types**(*labeled: bool*)

Get all registered data types.

Parameters

labeled
[bool] whether this data type is for LabeledSystem

dpdata.data_type.**register_data_type**(*data_type:* DataType, *labeled: bool*)

Register a data type.

Parameters

data_type
[DataType] data type to be registered

labeled
[bool] whether this data type is for LabeledSystem

6.1.7 dpdata.driver module

Driver plugin system.

class dpdata.driver.**Driver**(**args*, ***kwargs*)

Bases: ABC

The base class for a driver plugin. A driver can label a pure System to generate the LabeledSystem.

See also:

dpdata.plugins.deepmd.DPDriver
an example of Driver

Attributes

ase_calculator
Returns an ase calculator based on this driver.

Methods

get_driver(key)	Get a driver plugin.
get_drivers()	Get all driver plugins.
label(data)	Label a system data.
register(key)	Register a driver plugin.

property ase_calculator: ase.calculators.calculator.Calculator

Returns an ase calculator based on this driver.

static get_driver(*key: str*) → *Driver*

Get a driver plugin.

Parameters

key
[str] key of the plugin.

Returns

Driver
the specific driver class

Raises

RuntimeError
if the requested driver is not implemented

static get_drivers() → dict

Get all driver plugins.

Returns

dict
dict for all driver plugisn

abstract label(*data: dict*) → dict

Label a system data. Returns new data with energy, forces, and virials.

Parameters

data
[dict] data with coordinates and atom types

Returns

dict
labeled data with energies and forces

static register(*key: str*) → Callable

Register a driver plugin. Used as decorators.

Parameters

key
[str] key of the plugin.

Returns

Callable
decorator of a class

Examples

```
>>> @Driver.register("some_driver")
... class SomeDriver(Driver):
...     pass
```

class dpdata.driver.**HybridDriver**(*drivers: List[dict | Driver]*)

Bases: *Driver*

Hybrid driver, with mixed drivers.

Parameters

drivers
[list[dict, Driver]] list of drivers or drivers dict. For a dict, it should contain *type* as the name of the driver, and others are arguments of the driver.

Raises

TypeError
The value of *drivers* is not a dict or *Driver*.

Examples

```
>>> driver = HybridDriver([
...     {"type": "sqm", "qm_theory": "DFTB3"},
...     {"type": "dp", "dp": "frozen_model.pb"},
... ])
```

This driver is the hybrid of SQM and DP.

Attributes

ase_calculator
Returns an ase calculator based on this driver.

Methods

get_driver(key)	Get a driver plugin.
get_drivers()	Get all driver plugins.
label(data)	Label a system data.
register(key)	Register a driver plugin.

label(*data: dict*) → dict

Label a system data.

Energies and forces are the sum of those of each driver.

Parameters

data
[dict] data with coordinates and atom types

Returns

dict
labeled data with energies and forces

class dpdata.driver.**Minimizer**(**args*, ***kwargs*)

Bases: ABC

The base class for a minimizer plugin. A minimizer can minimize geometry.

Methods

get_minimizer(key)	Get a minimizer plugin.
get_minimizers()	Get all minimizer plugins.
minimize(data)	Minimize the geometry.
register(key)	Register a minimizer plugin.

static get_minimizer(*key: str*) → *Minimizer*

Get a minimizer plugin.

Parameters

key
[str] key of the plugin.

Returns

Minimizer
the specific minimizer class

Raises

RuntimeError
if the requested minimizer is not implemented

static get_minimizers() → dict

Get all minimizer plugins.

Returns

dict
dict for all minimizer plugisn

abstract minimize(*data: dict*) → dict

Minimize the geometry.

Parameters

data
[dict] data with coordinates and atom types

Returns

dict
labeled data with minimized coordinates, energies, and forces

static register(*key: str*) → Callable

Register a minimizer plugin. Used as decorators.

Parameters

key
[str] key of the plugin.

Returns

Callable
decorator of a class

Examples

```
>>> @Minimizer.register("some_minimizer")
... class SomeMinimizer(Minimizer):
...     pass
```

6.1.8 dpdata.format module

Implement the format plugin system.

class dpdata.format.**Format**

Bases: ABC

The abstract base class for all formats.

To add a new format, one should create a new class inherited from this class, and then

- implement several methods, such as *from_system()*;
- register the format with a key;
- add documentation in the class docstring;

The new format can be either insider or outside the package.

Methods

MultiModes()	File mode for MultiSystems.
from_bond_order_system(file_name, **kwargs)	Implement BondOrderSystem.from that converts from this format to BondOrderSystem.
from_labeled_system(file_name, **kwargs)	Implement LabeledSystem.from that converts from this format to LabeledSystem.
from_multi_systems(directory, **kwargs)	Implement MultiSystems.from that converts from this format to MultiSystems.
from_system(file_name, **kwargs)	Implement System.from that converts from this format to System.
get_formats()	Get all registered formats.
get_from_methods()	Get all registered from methods.
get_to_methods()	Get all registered to methods.
mix_system(*system, type_map, **kwargs)	Mix the systems into mixed_type ones according to the unified given type_map.
post(func_name)	Register a post function for from method.
register(key)	Register a format plugin.
register_from(key)	Register a from method if the target method name is not default.
register_to(key)	Register a to method if the target method name is not default.
to_bond_order_system(data, rdkit_mol, *args, ...)	Implement BondOrderSystem.to that converts from BondOrderSystem to this format.
to_labeled_system(data, *args, **kwargs)	Implement LabeledSystem.to that converts from LabeledSystem to this format.
to_multi_systems(formulas, directory, **kwargs)	Implement MultiSystems.to that converts from MultiSystems to this format.
to_system(data, *args, **kwargs)	Implement System.to that converts from System to this format.

MultiMode = 0

class MultiModes

Bases: object

File mode for MultiSystems.

The current implemented modes are:

- 0 (default): not implemented
- 1: every directory under the top-level directory is a system.

Directory = 1

NotImplemented = 0

from_bond_order_system(*file_name*, ***kwargs*)

Implement BondOrderSystem.from that converts from this format to BondOrderSystem.

Parameters

file_name
[str] file name, i.e. the first argument

****kwargs**
[dict] keyword arguments that will be passed from the method

Returns

data
[dict] system data

`from_labeled_system`(*file_name*, ***kwargs*)

Implement LabeledSystem.from that converts from this format to LabeledSystem.

Parameters

file_name
[str] file name, i.e. the first argument

****kwargs**
[dict] keyword arguments that will be passed from the method

Returns

data
[dict] system data, whose keys are defined in LabeledSystem.DTYPES

`from_multi_systems`(*directory*, ***kwargs*)

Implement MultiSystems.from that converts from this format to MultiSystems.

By default, this method follows MultiMode to implement the conversion.

Parameters

directory
[str] directory of system

****kwargs**
[dict] keyword arguments that will be passed from the method

Returns

filenames: list[str]
list of filenames

`from_system`(*file_name*, ***kwargs*)

Implement System.from that converts from this format to System.

Parameters

file_name
[str] file name, i.e. the first argument

****kwargs**
[dict] keyword arguments that will be passed from the method

Returns

data
[dict] system data, whose keys are defined in System.DTYPES

`static get_formats`()

Get all registered formats.

`static get_from_methods`()

Get all registered from methods.

static get_to_methods()

Get all registered to methods.

mix_system(**system*, *type_map*, ***kwargs*)

Mix the systems into mixed_type ones according to the unified given type_map.

Parameters

***system**
[System] The systems to mix

type_map
[list of str] Maps atom type to name

****kwargs**
[dict] keyword arguments that will be passed from the method

Returns

mixed_systems: dict
dict of mixed system with key 'atom_numbs'

static post(*func_name*)

Register a post function for from method.

Such function will be called after the "from" method is called.

Parameters

func_name
[str or list of str] The name of the post function.

Returns

function
The decorator function.

Examples

Register a post function:

```
>>> @Format.post('remove_pbc')
... @Format.register('test')
... class TestFormat(Format):
...     pass
```

static register(*key*)

Register a format plugin.

By default, after a format plugin is registered, the following methods will be registered as well for `System()`, `LabeledSystem()`, `MultiSystems()`, and `BondOrderSystem()`:

- from_{key.replace('/', '_')}
- to_{key.replace('/', '_')}
- from({key}, ...)
- to({key}, ...)

The decorator should be explicitly executed before *dpdata.system* is imported. A module will be imported automatically if it

- is a submodule of *dpdata.plugins*;
- is registered at the *dpdata.plugins* entry point

Parameters

key
[str] The key to register the plugin.

Returns

function
The decorator function.

Examples

Register a format plugin:

```
>>> @Format.register('test')
... @Format.register('test2')
... class TestFormat(Format):
...     pass
```

static register_from(*key*)

Register a from method if the target method name is not default.

Parameters

key
[str] The key to register the plugin.

Returns

function
The decorator function.

Examples

Register a from method:

```
>>> @Format.register_from('from_test_haha')
... @Format.register('test)
... class TestFormat(Format):
...     pass
```

This will register a from method named from_test_haha, although the format name is test.

static register_to(*key*)

Register a to method if the target method name is not default.

Parameters

key
[str] The key to register the plugin.

Returns

function
The decorator function.

Examples

Register a to method:

```
>>> @Format.register_to('to_test_haha')
... @Format.register('test')
... class TestFormat(Format):
...     pass
```

This will register a to method named to_test_haha, although the format name is test.

`to_bond_order_system`(*data*, *rdkit_mol*, **args*, ***kwargs*)

Implement BondOrderSystem.to that converts from BondOrderSystem to this format.

By default, BondOrderSystem.to will fallback to LabeledSystem.to.

Parameters

data
[dict] system data

rdkit_mol
[rdkit.Chem.rdchem.Mol] rdkit mol object

***args**
[list] arguments that will be passed from the method

****kwargs**
[dict] keyword arguments that will be passed from the method

`to_labeled_system`(*data*, **args*, ***kwargs*)

Implement LabeledSystem.to that converts from LabeledSystem to this format.

By default, LabeledSystem.to will fallback to System.to.

Parameters

data
[dict] system data, whose keys are defined in LabeledSystem.DTYPES

***args**
[list] arguments that will be passed from the method

****kwargs**
[dict] keyword arguments that will be passed from the method

`to_multi_systems`(*formulas*, *directory*, ***kwargs*)

Implement MultiSystems.to that converts from MultiSystems to this format.

By default, this method follows MultiMode to implement the conversion.

Parameters

formulas
[list[str]] list of formulas

directory
[str] directory of system

****kwargs**
[dict] keyword arguments that will be passed from the method

`to_system`(*data*, **args*, ***kwargs*)

Implement System.to that converts from System to this format.

Parameters

data
[dict] system data, whose keys are defined in System.DTYPES

***args**
[list] arguments that will be passed from the method

****kwargs**
[dict] keyword arguments that will be passed from the method

6.1.9 dpdata.periodic_table module

`class` dpdata.periodic_table.`Element`(*symbol: str*)

Bases: `object`

Attributes

X
Z
calculated_radius
mass
name
radius

Methods

from_Z

`property X`

`property Z`

`property calculated_radius`

`classmethod from_Z`(*Z*)

`property mass`

`property name`

`property radius`

6.1.10 dpdata.plugin module

Base of plugin systems.

class dpdata.plugin.**Plugin**

Bases: object

A class to register plugins.

Examples

```
>>> example_plugin = Plugin()
>>> @example_plugin.register("xx")
    def xxx():
        pass
>>> print(example_plugin.plugins['xx'])
```

Methods

register(key)	Register a plugin.

get_plugin

get_plugin(*key*)

register(*key*)

Register a plugin.

Parameters

key
[str] Key of the plugin.

6.1.11 dpdata.stat module

class dpdata.stat.**Errors**(*system_1: object*, *system_2: object*)

Bases: *ErrorsBase*

Compute errors (deviations) between two LabeledSystems.

Parameters

system_1
[object] system 1

system_2
[object] system 2

Examples

Get errors between referenced system and predicted system:

```
>>> e = dpdata.stat.Errors(system_1, system_2)
>>> print("%.4f %.4f %.4f %.4f" % (e.e_mae, e.e_rmse, e.f_mae, e.f_rmse))
```

Attributes

e_errors
Energy errors.

e_mae
Energy MAE.

e_rmse
Energy RMSE.

f_errors
Force errors.

f_mae
Force MAE.

f_rmse
Force RMSE.

Methods

SYSTEM_TYPE	alias of *LabeledSystem*

SYSTEM_TYPE
alias of *LabeledSystem*

property e_errors: ndarray
Energy errors.

property f_errors: ndarray
Force errors.

class dpdata.stat.**ErrorsBase**(*system_1: object*, *system_2: object*)

Bases: object

Compute errors (deviations) between two systems. The type of system is assigned by SYSTEM_TYPE.

Parameters

system_1
[object] system 1

system_2
[object] system 2

Attributes

e_errors
Energy errors.

e_mae
Energy MAE.

e_rmse
Energy RMSE.

f_errors
Force errors.

f_mae
Force MAE.

f_rmse
Force RMSE.

Methods

SYSTEM_TYPE	alias of `object`

SYSTEM_TYPE
alias of `object`

abstract property e_errors: `ndarray`
Energy errors.

property e_mae: `float64`
Energy MAE.

property e_rmse: `float64`
Energy RMSE.

abstract property f_errors: `ndarray`
Force errors.

property f_mae: `float64`
Force MAE.

property f_rmse: `float64`
Force RMSE.

class dpdata.stat.**MultiErrors**(*system_1: object*, *system_2: object*)
Bases: *ErrorsBase*

Compute errors (deviations) between two MultiSystems.

Parameters

system_1
[object] system 1

system_2
[object] system 2

Examples

Get errors between referenced system and predicted system:

```
>>> e = dpdata.stat.MultiErrors(system_1, system_2)
>>> print("%.4f %.4f %.4f %.4f" % (e.e_mae, e.e_rmse, e.f_mae, e.f_rmse))
```

Attributes

e_errors
Energy errors.

e_mae
Energy MAE.

e_rmse
Energy RMSE.

f_errors
Force errors.

f_mae
Force MAE.

f_rmse
Force RMSE.

Methods

SYSTEM_TYPE	alias of *MultiSystems*

SYSTEM_TYPE
alias of *MultiSystems*

property e_errors: ndarray
Energy errors.

property f_errors: ndarray
Force errors.

dpdata.stat.**mae**(*errors: ndarray*) → float64
Compute the mean absolute error (MAE).

Parameters

errors
[np.ndarray] errors between two values

Returns

np.float64
mean absolute error (MAE)

dpdata.stat.**rmse**(*errors: ndarray*) → float64
Compute the root mean squared error (RMSE).

Parameters

errors
[np.ndarray] errors between two values

Returns

np.float64
root mean squared error (RMSE)

6.1.12 dpdata.system module

class dpdata.system.**LabeledSystem**(*file_name=None*, *fmt='auto'*, *type_map=None*, *begin=0*, *step=1*, *data=None*, *convergence_check=True*, ***kwargs*)

Bases: *System*

The labeled data System.

For example, a labeled water system named *d_example* has two molecules (6 atoms) and *nframes* frames. The labels can be accessed by

- *d_example['energies']* : a numpy array of size nframes
- *d_example['forces']* : a numpy array of size nframes x 6 x 3
- *d_example['virials']* : optional, a numpy array of size nframes x 3 x 3

It is noted that

- The order of frames stored in *'energies'*, *'forces'* and *'virials'* should be consistent with *'atom_types'*, *'cells'* and *'coords'*.
- The order of atoms in **every** frame of *'forces'* should be consistent with *'coords'* and *'atom_types'*.

Parameters

file_name
[str] The file to load the system

fmt
[str]

Format of the file, supported formats are

- `auto`: infered from *file_name*'s extension
- `vasp/xml`: vasp xml
- `vasp/outcar`: vasp OUTCAR
- `deepmd/raw`: deepmd-kit raw
- `deepmd/npy`: deepmd-kit compressed format (numpy binary)
- `qe/cp/traj`: Quantum Espresso CP trajectory files. should have: file_name+'.in', file_name+'.pos', file_name+'.evp' and file_name+'.for'
- `qe/pw/scf`: Quantum Espresso PW single point calculations. Both input and output files are required. If file_name is a string, it denotes the output file name. Input file name is obtained by replacing 'out' by 'in' from file_name. Or file_name is a list, with the first element being the input file name and the second element being the output filename.
- `siesta/output`: siesta SCF output file
- `siesta/aimd_output`: siesta aimd output file

- `gaussian/log`: gaussian logs
- `gaussian/md`: gaussian ab initio molecular dynamics
- `cp2k/output`: cp2k output file
- `cp2k/aimd_output`: cp2k aimd output dir(contains *pos*.xyz and *.log file); optional *restart=True* if it is a cp2k restarted task.
- `pwmat/movement`: pwmat md output file
- `pwmat/out.mlmd`: pwmat scf output file

type_map
[list of str] Maps atom type to name. The atom with type *ii* is mapped to *type_map[ii]*. If not provided the atom names are assigned to *'Type_1'*, *'Type_2'*, *'Type_3'*...

begin
[int] The beginning frame when loading MD trajectory.

step
[int] The number of skipped frames when loading MD trajectory.

Attributes

formula
Return the formula of this system, like C3H5O2.

formula_hash
Return the hash of the formula of this system.

nopbc

short_formula
Return the short formula of this system.

short_name
Return the short name of this system (no more than 255 bytes), in the following order: - formula - short_formula - formula_hash.

uniq_formula
Return the uniq_formula of this system.

Methods

add_atom_names(atom_names)	Add atom_names that do not exist.
append(system)	Append a system to this system.
apply_pbc()	Append periodic boundary condition.
as_dict()	Returns data dict of System instance.
check_data()	Check if data is correct.
check_type_map(type_map)	Assign atom_names to type_map if type_map is given and different from atom_names.
convert_to_mixed_type([type_map])	Convert the data dict to mixed type format structure, in order to append systems with different formula but the same number of atoms.
copy()	Returns a copy of the system.
correction(hl_sys)	Get energy and force correction between self and a high-level LabeledSystem.
dump(filename[, indent])	Dump .json or .yaml file.

continues on next page

Table 6 – continued from previous page

extend(systems)	Extend a system list to this system.
from_3dmol(file_name, **kwargs)	Read data from *dpdata.plugins.3dmol.Py3DMolFormat* format.
from_abacus_lcao_md(file_name, **kwargs)	Read data from *dpdata.plugins.abacus.AbacusMDFormat* format.
from_abacus_lcao_relax(file_name, **kwargs)	Read data from *dpdata.plugins.abacus.AbacusRelaxFormat* format.
from_abacus_lcao_scf(file_name, **kwargs)	Read data from *dpdata.plugins.abacus.AbacusSCFFormat* format.
from_abacus_md(file_name, **kwargs)	Read data from *dpdata.plugins.abacus.AbacusMDFormat* format.
from_abacus_pw_md(file_name, **kwargs)	Read data from *dpdata.plugins.abacus.AbacusMDFormat* format.
from_abacus_pw_relax(file_name, **kwargs)	Read data from *dpdata.plugins.abacus.AbacusRelaxFormat* format.
from_abacus_pw_scf(file_name, **kwargs)	Read data from *dpdata.plugins.abacus.AbacusSCFFormat* format.
from_abacus_relax(file_name, **kwargs)	Read data from *dpdata.plugins.abacus.AbacusRelaxFormat* format.
from_abacus_scf(file_name, **kwargs)	Read data from *dpdata.plugins.abacus.AbacusSCFFormat* format.
from_abacus_stru(file_name, **kwargs)	Read data from *dpdata.plugins.abacus.AbacusSTRUFormat* format.
from_amber_md(file_name, **kwargs)	Read data from *dpdata.plugins.amber.AmberMDFormat* format.
from_ase_structure(file_name, **kwargs)	Read data from *dpdata.plugins.ase.ASEStructureFormat* format.
from_ase_traj(file_name, **kwargs)	Read data from *dpdata.plugins.ase.ASETrajFormat* format.
from_atomconfig(file_name, **kwargs)	Read data from *dpdata.plugins.pwmat.PwmatAtomconfigFormat* format.
from_contcar(file_name, **kwargs)	Read data from *dpdata.plugins.vasp.VASPPoscarFormat* format.
from_cp2k_aimd_output(file_name, **kwargs)	Read data from *dpdata.plugins.cp2k.CP2KAIMDOutputFormat* format.
from_cp2k_output(file_name, **kwargs)	Read data from *dpdata.plugins.cp2k.CP2KOutputFormat* format.
from_deepmd(file_name, **kwargs)	Read data from *dpdata.plugins.deepmd.DeePMDRawFormat* format.
from_deepmd_comp(file_name, **kwargs)	Read data from *dpdata.plugins.deepmd.DeePMDCompFormat* format.
from_deepmd_hdf5(file_name, **kwargs)	Read data from *dpdata.plugins.deepmd.DeePMDHDF5Format* format.
from_deepmd_npy(file_name, **kwargs)	Read data from *dpdata.plugins.deepmd.DeePMDCompFormat* format.
from_deepmd_npy_mixed(file_name, **kwargs)	Read data from *dpdata.plugins.deepmd.DeePMDMixedFormat* format.
from_deepmd_raw(file_name, **kwargs)	Read data from *dpdata.plugins.deepmd.DeePMDRawFormat* format.
from_dftbplus(file_name, **kwargs)	Read data from *dpdata.plugins.dftbplus.DFTBplusFormat* format.

continues on next page

Table 6 – continued from previous page

from_dict(d) **param d** Dict representation.	
from_dump(file_name, **kwargs)	Read data from *dpdata.plugins.lammps.LAMMPSDumpFormat* format.
from_fhi_aims_md(file_name, **kwargs)	Read data from *dpdata.plugins.fhi_aims.FhiMDFormat* format.
from_fhi_aims_output(file_name, **kwargs)	Read data from *dpdata.plugins.fhi_aims.FhiMDFormat* format.
from_fhi_aims_scf(file_name, **kwargs)	Read data from *dpdata.plugins.fhi_aims.FhiSCFFormat* format.
from_finalconfig(file_name, **kwargs)	Read data from *dpdata.plugins.pwmat.PwmatAtomconfigFormat* format.
from_gaussian_gjf(file_name, **kwargs)	Read data from *dpdata.plugins.gaussian.GaussiaGJFFormat* format.
from_gaussian_log(file_name, **kwargs)	Read data from *dpdata.plugins.gaussian.GaussianLogFormat* format.
from_gaussian_md(file_name, **kwargs)	Read data from *dpdata.plugins.gaussian.GaussianMDFormat* format.
from_gro(file_name, **kwargs)	Read data from *dpdata.plugins.gromacs.GromacsGroFormat* format.
from_gromacs_gro(file_name, **kwargs)	Read data from *dpdata.plugins.gromacs.GromacsGroFormat* format.
from_lammps_dump(file_name, **kwargs)	Read data from *dpdata.plugins.lammps.LAMMPSDumpFormat* format.
from_lammps_lmp(file_name, **kwargs)	Read data from *dpdata.plugins.lammps.LAMMPSLmpFormat* format.
from_list(file_name, **kwargs)	Read data from *dpdata.plugins.list.ListFormat* format.
from_lmp(file_name, **kwargs)	Read data from *dpdata.plugins.lammps.LAMMPSLmpFormat* format.
from_mlmd(file_name, **kwargs)	Read data from *dpdata.plugins.pwmat.PwmatOutputFormat* format.
from_mol(file_name, **kwargs)	Read data from *dpdata.plugins.rdkit.MolFormat* format.
from_mol_file(file_name, **kwargs)	Read data from *dpdata.plugins.rdkit.MolFormat* format.
from_movement(file_name, **kwargs)	Read data from *dpdata.plugins.pwmat.PwmatOutputFormat* format.
from_n2p2(file_name, **kwargs)	Read data from *dpdata.plugins.n2p2.N2P2Format* format.
from_openmx_md(file_name, **kwargs)	Read data from *dpdata.plugins.openmx.OPENMXFormat* format.
from_orca_spout(file_name, **kwargs)	Read data from *dpdata.plugins.orca.ORCASPOutFormat* format.
from_outcar(file_name, **kwargs)	Read data from *dpdata.plugins.vasp.VASPOutcarFormat* format.
from_poscar(file_name, **kwargs)	Read data from *dpdata.plugins.vasp.VASPPoscarFormat* format.
from_psi4_inp(file_name, **kwargs)	Read data from *dpdata.plugins.psi4.PSI4InputFormat* format.

continues on next page

Table 6 – continued from previous page

from_psi4_out(file_name, **kwargs)	Read data from *dpdata.plugins.psi4.PSI4OutFormat* format.
from_pwmat_atomconfig(file_name, **kwargs)	Read data from *dpdata.plugins.pwmat.PwmatAtomconfigFormat* format.
from_pwmat_finalconfig(file_name, **kwargs)	Read data from *dpdata.plugins.pwmat.PwmatAtomconfigFormat* format.
from_pwmat_mlmd(file_name, **kwargs)	Read data from *dpdata.plugins.pwmat.PwmatOutputFormat* format.
from_pwmat_movement(file_name, **kwargs)	Read data from *dpdata.plugins.pwmat.PwmatOutputFormat* format.
from_pwmat_output(file_name, **kwargs)	Read data from *dpdata.plugins.pwmat.PwmatOutputFormat* format.
from_pymatgen_computedstructureentry(...)	Read data from *dpdata.plugins.pymatgen.PyMatgenCSEFormat* format.
from_pymatgen_molecule(file_name, **kwargs)	Read data from *dpdata.plugins.pymatgen.PyMatgenMoleculeFormat* format.
from_pymatgen_structure(file_name, **kwargs)	Read data from *dpdata.plugins.pymatgen.PyMatgenStructureFormat* format.
from_qe_cp_traj(file_name, **kwargs)	Read data from *dpdata.plugins.qe.QECPTrajFormat* format.
from_qe_pw_scf(file_name, **kwargs)	Read data from *dpdata.plugins.qe.QECPPWSCFFormat* format.
from_quip_gap_xyz(file_name, **kwargs)	Read data from *dpdata.plugins.xyz.QuipGapXYZFormat* format.
from_quip_gap_xyz_file(file_name, **kwargs)	Read data from *dpdata.plugins.xyz.QuipGapXYZFormat* format.
from_sdf(file_name, **kwargs)	Read data from *dpdata.plugins.rdkit.SdfFormat* format.
from_sdf_file(file_name, **kwargs)	Read data from *dpdata.plugins.rdkit.SdfFormat* format.
from_siesta_aiMD_output(file_name, **kwargs)	Read data from *dpdata.plugins.siesta.SiestaAIMDOutputFormat* format.
from_siesta_aimd_output(file_name, **kwargs)	Read data from *dpdata.plugins.siesta.SiestaAIMDOutputFormat* format.
from_siesta_output(file_name, **kwargs)	Read data from *dpdata.plugins.siesta.SiestaOutputFormat* format.
from_sqm_in(file_name, **kwargs)	Read data from *dpdata.plugins.amber.SQMINFormat* format.
from_sqm_out(file_name, **kwargs)	Read data from *dpdata.plugins.amber.SQMOutFormat* format.
from_stru(file_name, **kwargs)	Read data from *dpdata.plugins.abacus.AbacusSTRUFormat* format.
from_vasp_contcar(file_name, **kwargs)	Read data from *dpdata.plugins.vasp.VASPPoscarFormat* format.
from_vasp_outcar(file_name, **kwargs)	Read data from *dpdata.plugins.vasp.VASPOutcarFormat* format.
from_vasp_poscar(file_name, **kwargs)	Read data from *dpdata.plugins.vasp.VASPPoscarFormat* format.
from_vasp_string(file_name, **kwargs)	Read data from *dpdata.plugins.vasp.VASPStringFormat* format.
from_vasp_xml(file_name, **kwargs)	Read data from *dpdata.plugins.vasp.VASPXMLFormat* format.

continues on next page

Table 6 – continued from previous page

from_xml(file_name, **kwargs)	Read data from *dpdata.plugins.vasp.VASPXMLFormat* format.
from_xyz(file_name, **kwargs)	Read data from *dpdata.plugins.xyz.XYZFormat* format.
get_atom_names()	Returns name of atoms.
get_atom_numbs()	Returns number of atoms.
get_atom_types()	Returns type of atoms.
get_natoms()	Returns total number of atoms in the system.
get_nframes()	Returns number of frames in the system.
get_ntypes()	Returns total number of atom types in the system.
load(filename)	Rebuild System obj.
map_atom_types([type_map])	Map the atom types of the system.
minimize(*args, minimizer, **kwargs)	Minimize the geometry.
perturb(pert_num, cell_pert_fraction, ...[, ...])	Perturb each frame in the system randomly.
pick_atom_idx(idx[, nopbc])	Pick atom index.
pick_by_amber_mask(param, maskstr[, ...])	Pick atoms by amber mask.
predict(*args[, driver])	Predict energies and forces by a driver.
register_data_type(*data_type)	Register data type.
remove_atom_names(atom_names)	Remove atom names and all such atoms.
remove_outlier([threshold])	Remove outlier frames from the system.
remove_pbc([protect_layer])	This method does NOT delete the definition of the cells, it (1) revises the cell to a cubic cell and ensures that the cell boundary to any atom in the system is no less than *protect_layer* (2) translates the system such that the center-of-geometry of the system locates at the center of the cell.
replicate(ncopy)	Replicate the each frame in the system in 3 dimensions.
shuffle()	Shuffle frames randomly.
sort_atom_names([type_map])	Sort atom_names of the system and reorder atom_numbs and atom_types accoarding to atom_names.
sort_atom_types()	Sort atom types.
sub_system(f_idx)	Construct a subsystem from the system.
to(fmt, *args, **kwargs)	Dump systems to the specific format.
to_3dmol(*args, **kwargs)	Dump data to *dpdata.plugins.3dmol.Py3DMolFormat* format.
to_abacus_lcao_md(*args, **kwargs)	Dump data to *dpdata.plugins.abacus.AbacusMDFormat* format.
to_abacus_lcao_relax(*args, **kwargs)	Dump data to *dpdata.plugins.abacus.AbacusRelaxFormat* format.
to_abacus_lcao_scf(*args, **kwargs)	Dump data to *dpdata.plugins.abacus.AbacusSCFFormat* format.
to_abacus_md(*args, **kwargs)	Dump data to *dpdata.plugins.abacus.AbacusMDFormat* format.
to_abacus_pw_md(*args, **kwargs)	Dump data to *dpdata.plugins.abacus.AbacusMDFormat* format.
to_abacus_pw_relax(*args, **kwargs)	Dump data to *dpdata.plugins.abacus.AbacusRelaxFormat* format.
to_abacus_pw_scf(*args, **kwargs)	Dump data to *dpdata.plugins.abacus.AbacusSCFFormat* format.

continues on next page

Table 6 – continued from previous page

to_abacus_relax(*args, **kwargs)	Dump data to dpdata.plugins.abacus.AbacusRelaxFormat format.
to_abacus_scf(*args, **kwargs)	Dump data to dpdata.plugins.abacus.AbacusSCFFormat format.
to_abacus_stru(*args, **kwargs)	Dump data to dpdata.plugins.abacus.AbacusSTRUFormat format.
to_amber_md(*args, **kwargs)	Dump data to dpdata.plugins.amber.AmberMDFormat format.
to_ase_structure(*args, **kwargs)	Dump data to dpdata.plugins.ase.ASEStructureFormat format.
to_ase_traj(*args, **kwargs)	Dump data to dpdata.plugins.ase.ASETrajFormat format.
to_atomconfig(*args, **kwargs)	Dump data to dpdata.plugins.pwmat.PwmatAtomconfigFormat format.
to_contcar(*args, **kwargs)	Dump data to dpdata.plugins.vasp.VASPPoscarFormat format.
to_cp2k_aimd_output(*args, **kwargs)	Dump data to dpdata.plugins.cp2k.CP2KAIMDOutputFormat format.
to_cp2k_output(*args, **kwargs)	Dump data to dpdata.plugins.cp2k.CP2KOutputFormat format.
to_deepmd(*args, **kwargs)	Dump data to dpdata.plugins.deepmd.DeePMDRawFormat format.
to_deepmd_comp(*args, **kwargs)	Dump data to dpdata.plugins.deepmd.DeePMDCompFormat format.
to_deepmd_hdf5(*args, **kwargs)	Dump data to dpdata.plugins.deepmd.DeePMDHDF5Format format.
to_deepmd_npy(*args, **kwargs)	Dump data to dpdata.plugins.deepmd.DeePMDCompFormat format.
to_deepmd_npy_mixed(*args, **kwargs)	Dump data to dpdata.plugins.deepmd.DeePMDMixedFormat format.
to_deepmd_raw(*args, **kwargs)	Dump data to dpdata.plugins.deepmd.DeePMDRawFormat format.
to_dftbplus(*args, **kwargs)	Dump data to dpdata.plugins.dftbplus.DFTBplusFormat format.
to_dump(*args, **kwargs)	Dump data to dpdata.plugins.lammps.LAMMPSDumpFormat format.
to_fhi_aims_md(*args, **kwargs)	Dump data to dpdata.plugins.fhi_aims.FhiMDFormat format.
to_fhi_aims_output(*args, **kwargs)	Dump data to dpdata.plugins.fhi_aims.FhiMDFormat format.
to_fhi_aims_scf(*args, **kwargs)	Dump data to dpdata.plugins.fhi_aims.FhiSCFFormat format.
to_finalconfig(*args, **kwargs)	Dump data to dpdata.plugins.pwmat.PwmatAtomconfigFormat format.
to_gaussian_gjf(*args, **kwargs)	Dump data to dpdata.plugins.gaussian.GaussiaGJFFormat format.
to_gaussian_log(*args, **kwargs)	Dump data to dpdata.plugins.gaussian.GaussianLogFormat format.
to_gaussian_md(*args, **kwargs)	Dump data to dpdata.plugins.gaussian.GaussianMDFormat format.
to_gro(*args, **kwargs)	Dump data to dpdata.plugins.gromacs.GromacsGroFormat format.

continues on next page

Table 6 – continued from previous page

to_gromacs_gro(*args, **kwargs)	Dump data to dpdata.plugins.gromacs.GromacsGroFormat format.
to_json()	Returns a json string representation of the MSONable object.
to_lammps_dump(*args, **kwargs)	Dump data to dpdata.plugins.lammps.LAMMPSDumpFormat format.
to_lammps_lmp(*args, **kwargs)	Dump data to dpdata.plugins.lammps.LAMMPSLmpFormat format.
to_list(*args, **kwargs)	Dump data to dpdata.plugins.list.ListFormat format.
to_lmp(*args, **kwargs)	Dump data to dpdata.plugins.lammps.LAMMPSLmpFormat format.
to_mlmd(*args, **kwargs)	Dump data to dpdata.plugins.pwmat.PwmatOutputFormat format.
to_mol(*args, **kwargs)	Dump data to dpdata.plugins.rdkit.MolFormat format.
to_mol_file(*args, **kwargs)	Dump data to dpdata.plugins.rdkit.MolFormat format.
to_movement(*args, **kwargs)	Dump data to dpdata.plugins.pwmat.PwmatOutputFormat format.
to_n2p2(*args, **kwargs)	Dump data to dpdata.plugins.n2p2.N2P2Format format.
to_openmx_md(*args, **kwargs)	Dump data to dpdata.plugins.openmx.OPENMXFormat format.
to_orca_spout(*args, **kwargs)	Dump data to dpdata.plugins.orca.ORCASPOutFormat format.
to_outcar(*args, **kwargs)	Dump data to dpdata.plugins.vasp.VASPOutcarFormat format.
to_poscar(*args, **kwargs)	Dump data to dpdata.plugins.vasp.VASPPoscarFormat format.
to_psi4_inp(*args, **kwargs)	Dump data to dpdata.plugins.psi4.PSI4InputFormat format.
to_psi4_out(*args, **kwargs)	Dump data to dpdata.plugins.psi4.PSI4OutFormat format.
to_pwmat_atomconfig(*args, **kwargs)	Dump data to dpdata.plugins.pwmat.PwmatAtomconfigFormat format.
to_pwmat_finalconfig(*args, **kwargs)	Dump data to dpdata.plugins.pwmat.PwmatAtomconfigFormat format.
to_pwmat_mlmd(*args, **kwargs)	Dump data to dpdata.plugins.pwmat.PwmatOutputFormat format.
to_pwmat_movement(*args, **kwargs)	Dump data to dpdata.plugins.pwmat.PwmatOutputFormat format.
to_pwmat_output(*args, **kwargs)	Dump data to dpdata.plugins.pwmat.PwmatOutputFormat format.
to_pymatgen_ComputedStructureEntry(*args, ...)	Dump data to dpdata.plugins.pymatgen.PyMatgenCSEFormat format.
to_pymatgen_computedstructureentry(*args, ...)	Dump data to dpdata.plugins.pymatgen.PyMatgenCSEFormat format.
to_pymatgen_molecule(*args, **kwargs)	Dump data to dpdata.plugins.pymatgen.PyMatgenMoleculeFormat format.
to_pymatgen_structure(*args, **kwargs)	Dump data to dpdata.plugins.pymatgen.PyMatgenStructureFormat format.

continues on next page

Table 6 – continued from previous page

to_qe_cp_traj(*args, **kwargs)	Dump data to *dpdata.plugins.qe.QECPTrajFormat* format.
to_qe_pw_scf(*args, **kwargs)	Dump data to *dpdata.plugins.qe.QECPPWSCFFormat* format.
to_quip_gap_xyz(*args, **kwargs)	Dump data to *dpdata.plugins.xyz.QuipGapXYZFormat* format.
to_quip_gap_xyz_file(*args, **kwargs)	Dump data to *dpdata.plugins.xyz.QuipGapXYZFormat* format.
to_sdf(*args, **kwargs)	Dump data to *dpdata.plugins.rdkit.SdfFormat* format.
to_sdf_file(*args, **kwargs)	Dump data to *dpdata.plugins.rdkit.SdfFormat* format.
to_siesta_aimd_output(*args, **kwargs)	Dump data to *dpdata.plugins.siesta.SiestaAIMDOutputFormat* format.
to_siesta_output(*args, **kwargs)	Dump data to *dpdata.plugins.siesta.SiestaOutputFormat* format.
to_sqm_in(*args, **kwargs)	Dump data to *dpdata.plugins.amber.SQMINFormat* format.
to_sqm_out(*args, **kwargs)	Dump data to *dpdata.plugins.amber.SQMOutFormat* format.
to_stru(*args, **kwargs)	Dump data to *dpdata.plugins.abacus.AbacusSTRUFormat* format.
to_vasp_contcar(*args, **kwargs)	Dump data to *dpdata.plugins.vasp.VASPPoscarFormat* format.
to_vasp_outcar(*args, **kwargs)	Dump data to *dpdata.plugins.vasp.VASPOutcarFormat* format.
to_vasp_poscar(*args, **kwargs)	Dump data to *dpdata.plugins.vasp.VASPPoscarFormat* format.
to_vasp_string(*args, **kwargs)	Dump data to *dpdata.plugins.vasp.VASPStringFormat* format.
to_vasp_xml(*args, **kwargs)	Dump data to *dpdata.plugins.vasp.VASPXMLFormat* format.
to_xml(*args, **kwargs)	Dump data to *dpdata.plugins.vasp.VASPXMLFormat* format.
to_xyz(*args, **kwargs)	Dump data to *dpdata.plugins.xyz.XYZFormat* format.
unsafe_hash()	Returns an hash of the current object.
validate_monty_v1(_MSONable__input_value)	Pydantic validator with correct signature for pydantic v1.x
validate_monty_v2(_MSONable__input_value, _)	Pydantic validator with correct signature for pydantic v2.x

affine_map
affine_map_fv
apply_type_map
from_fmt
from_fmt_obj
has_virial
replace
rot_frame_lower_triangular
rot_lower_triangular
to_fmt_obj

DTYPES = (<dpdata.data_type.DataType object>, <dpdata.data_type.DataType object>, <dpdata.data_type.DataType object>, <dpdata.data_type.DataType object>, <dpdata.data_type.DataType object>, <dpdata.data_type.DataType object>, <dpdata.data_type.DataType object>, <dpdata.data_type.DataType object>, <dpdata.data_type.DataType object>, <dpdata.data_type.DataType object>)

affine_map_fv(*trans*, *f_idx*)

correction(*hl_sys*)

Get energy and force correction between self and a high-level LabeledSystem. The self's coordinates will be kept, but energy and forces will be replaced by the correction between these two systems.

Note: The function will not check whether coordinates and elements of two systems are the same. The user should make sure by itself.

Parameters

hl_sys
[LabeledSystem] high-level LabeledSystem

Returns

corrected_sys: LabeledSystem
Corrected LabeledSystem

from_3dmol(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.3dmol.Py3DMolFormat* format.

from_abacus_lcao_md(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.abacus.AbacusMDFormat* format.

from_abacus_lcao_relax(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.abacus.AbacusRelaxFormat* format.

from_abacus_lcao_scf(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.abacus.AbacusSCFFormat* format.

from_abacus_md(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.abacus.AbacusMDFormat* format.

from_abacus_pw_md(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.abacus.AbacusMDFormat* format.

from_abacus_pw_relax(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.abacus.AbacusRelaxFormat* format.

from_abacus_pw_scf(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.abacus.AbacusSCFFormat* format.

from_abacus_relax(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.abacus.AbacusRelaxFormat* format.

from_abacus_scf(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.abacus.AbacusSCFFormat* format.

from_abacus_stru(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.abacus.AbacusSTRUFormat* format.

from_amber_md(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.amber.AmberMDFormat* format.

from_ase_structure(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.ase.ASEStructureFormat* format.

from_ase_traj(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.ase.ASETrajFormat* format.

from_atomconfig(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.pwmat.PwmatAtomconfigFormat* format.

from_contcar(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.vasp.VASPPoscarFormat* format.

from_cp2k_aimd_output(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.cp2k.CP2KAIMDOutputFormat* format.

from_cp2k_output(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.cp2k.CP2KOutputFormat* format.

from_deepmd(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.deepmd.DeePMDRawFormat* format.

from_deepmd_comp(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.deepmd.DeePMDCompFormat* format.

from_deepmd_hdf5(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.deepmd.DeePMDHDF5Format* format.

from_deepmd_npy(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.deepmd.DeePMDCompFormat* format.

from_deepmd_npy_mixed(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.deepmd.DeePMDMixedFormat* format.

from_deepmd_raw(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.deepmd.DeePMDRawFormat* format.

from_dftbplus(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.dftbplus.DFTBplusFormat* format.

from_dump(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.lammps.LAMMPSDumpFormat* format.

from_fhi_aims_md(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.fhi_aims.FhiMDFormat* format.

from_fhi_aims_output(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.fhi_aims.FhiMDFormat* format.

from_fhi_aims_scf(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.fhi_aims.FhiSCFFormat* format.

from_finalconfig(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.pwmat.PwmatAtomconfigFormat* format.

from_fmt_obj(*fmtobj*, *file_name*, ***kwargs*)

from_gaussian_gjf(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.gaussian.GaussiaGJFFormat* format.

from_gaussian_log(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.gaussian.GaussianLogFormat* format.

from_gaussian_md(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.gaussian.GaussianMDFormat* format.

from_gro(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.gromacs.GromacsGroFormat* format.

from_gromacs_gro(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.gromacs.GromacsGroFormat* format.

from_lammps_dump(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.lammps.LAMMPSDumpFormat* format.

from_lammps_lmp(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.lammps.LAMMPSLmpFormat* format.

from_list(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.list.ListFormat* format.

from_lmp(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.lammps.LAMMPSLmpFormat* format.

from_mlmd(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.pwmat.PwmatOutputFormat* format.

from_mol(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.rdkit.MolFormat* format.

from_mol_file(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.rdkit.MolFormat* format.

from_movement(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.pwmat.PwmatOutputFormat* format.

from_n2p2(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.n2p2.N2P2Format* format.

from_openmx_md(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.openmx.OPENMXFormat* format.

from_orca_spout(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.orca.ORCASPOutFormat* format.

from_outcar(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.vasp.VASPOutcarFormat* format.

from_poscar(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.vasp.VASPPoscarFormat* format.

from_psi4_inp(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.psi4.PSI4InputFormat* format.

from_psi4_out(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.psi4.PSI4OutFormat* format.

from_pwmat_atomconfig(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.pwmat.PwmatAtomconfigFormat* format.

from_pwmat_finalconfig(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.pwmat.PwmatAtomconfigFormat* format.

from_pwmat_mlmd(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.pwmat.PwmatOutputFormat* format.

from_pwmat_movement(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.pwmat.PwmatOutputFormat* format.

from_pwmat_output(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.pwmat.PwmatOutputFormat* format.

from_pymatgen_computedstructureentry(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.pymatgen.PyMatgenCSEFormat* format.

from_pymatgen_molecule(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.pymatgen.PyMatgenMoleculeFormat* format.

from_pymatgen_structure(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.pymatgen.PyMatgenStructureFormat* format.

from_qe_cp_traj(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.qe.QECPTrajFormat* format.

from_qe_pw_scf(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.qe.QECPPWSCFFormat* format.

from_quip_gap_xyz(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.xyz.QuipGapXYZFormat* format.

from_quip_gap_xyz_file(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.xyz.QuipGapXYZFormat* format.

from_sdf(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.rdkit.SdfFormat* format.

from_sdf_file(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.rdkit.SdfFormat* format.

from_siesta_aiMD_output(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.siesta.SiestaAIMDOutputFormat* format.

from_siesta_aimd_output(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.siesta.SiestaAIMDOutputFormat* format.

from_siesta_output(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.siesta.SiestaOutputFormat* format.

from_sqm_in(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.amber.SQMINFormat* format.

from_sqm_out(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.amber.SQMOutFormat* format.

from_stru(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.abacus.AbacusSTRUFormat* format.

from_vasp_contcar(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.vasp.VASPPoscarFormat* format.

from_vasp_outcar(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.vasp.VASPOutcarFormat* format.

from_vasp_poscar(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.vasp.VASPPoscarFormat* format.

from_vasp_string(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.vasp.VASPStringFormat* format.

from_vasp_xml(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.vasp.VASPXMLFormat* format.

from_xml(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.vasp.VASPXMLFormat* format.

from_xyz(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.xyz.XYZFormat* format.

has_virial()

post_funcs = <dpdata.plugin.Plugin object>

remove_outlier(*threshold: float = 8.0*) → *LabeledSystem*

Remove outlier frames from the system.

Remove the frames whose energies satisfy the condition

$$\frac{\left\|E-\bar{E}\right\|}{\sigma(E)} \geq \text{threshold}$$

where $\bar{E}$ and $\sigma(E)$ are the mean and standard deviation of the energies in the system.

Parameters

threshold

[float] The threshold of outlier detection. The default value is 8.0.

Returns

LabeledSystem
The system without outlier frames.

References

[1], [2]

rot_frame_lower_triangular(*f_idx=0*)

to_3dmol(**args*, ***kwargs*)
Dump data to *dpdata.plugins.3dmol.Py3DMolFormat* format.

to_abacus_lcao_md(**args*, ***kwargs*)
Dump data to *dpdata.plugins.abacus.AbacusMDFormat* format.

to_abacus_lcao_relax(**args*, ***kwargs*)
Dump data to *dpdata.plugins.abacus.AbacusRelaxFormat* format.

to_abacus_lcao_scf(**args*, ***kwargs*)
Dump data to *dpdata.plugins.abacus.AbacusSCFFormat* format.

to_abacus_md(**args*, ***kwargs*)
Dump data to *dpdata.plugins.abacus.AbacusMDFormat* format.

to_abacus_pw_md(**args*, ***kwargs*)
Dump data to *dpdata.plugins.abacus.AbacusMDFormat* format.

to_abacus_pw_relax(**args*, ***kwargs*)
Dump data to *dpdata.plugins.abacus.AbacusRelaxFormat* format.

to_abacus_pw_scf(**args*, ***kwargs*)
Dump data to *dpdata.plugins.abacus.AbacusSCFFormat* format.

to_abacus_relax(**args*, ***kwargs*)
Dump data to *dpdata.plugins.abacus.AbacusRelaxFormat* format.

to_abacus_scf(**args*, ***kwargs*)
Dump data to *dpdata.plugins.abacus.AbacusSCFFormat* format.

to_abacus_stru(**args*, ***kwargs*)
Dump data to *dpdata.plugins.abacus.AbacusSTRUFormat* format.

to_amber_md(**args*, ***kwargs*)
Dump data to *dpdata.plugins.amber.AmberMDFormat* format.

to_ase_structure(**args*, ***kwargs*)
Dump data to *dpdata.plugins.ase.ASEStructureFormat* format.

to_ase_traj(**args*, ***kwargs*)
Dump data to *dpdata.plugins.ase.ASETrajFormat* format.

to_atomconfig(**args*, ***kwargs*)
Dump data to *dpdata.plugins.pwmat.PwmatAtomconfigFormat* format.

to_contcar(**args*, ***kwargs*)
Dump data to *dpdata.plugins.vasp.VASPPoscarFormat* format.

to_cp2k_aimd_output(**args*, ***kwargs*)

Dump data to *dpdata.plugins.cp2k.CP2KAIMDOutputFormat* format.

to_cp2k_output(**args*, ***kwargs*)

Dump data to *dpdata.plugins.cp2k.CP2KOutputFormat* format.

to_deepmd(**args*, ***kwargs*)

Dump data to *dpdata.plugins.deepmd.DeePMDRawFormat* format.

to_deepmd_comp(**args*, ***kwargs*)

Dump data to *dpdata.plugins.deepmd.DeePMDCompFormat* format.

to_deepmd_hdf5(**args*, ***kwargs*)

Dump data to *dpdata.plugins.deepmd.DeePMDHDF5Format* format.

to_deepmd_npy(**args*, ***kwargs*)

Dump data to *dpdata.plugins.deepmd.DeePMDCompFormat* format.

to_deepmd_npy_mixed(**args*, ***kwargs*)

Dump data to *dpdata.plugins.deepmd.DeePMDMixedFormat* format.

to_deepmd_raw(**args*, ***kwargs*)

Dump data to *dpdata.plugins.deepmd.DeePMDRawFormat* format.

to_dftbplus(**args*, ***kwargs*)

Dump data to *dpdata.plugins.dftbplus.DFTBplusFormat* format.

to_dump(**args*, ***kwargs*)

Dump data to *dpdata.plugins.lammps.LAMMPSDumpFormat* format.

to_fhi_aims_md(**args*, ***kwargs*)

Dump data to *dpdata.plugins.fhi_aims.FhiMDFormat* format.

to_fhi_aims_output(**args*, ***kwargs*)

Dump data to *dpdata.plugins.fhi_aims.FhiMDFormat* format.

to_fhi_aims_scf(**args*, ***kwargs*)

Dump data to *dpdata.plugins.fhi_aims.FhiSCFFormat* format.

to_finalconfig(**args*, ***kwargs*)

Dump data to *dpdata.plugins.pwmat.PwmatAtomconfigFormat* format.

to_fmt_obj(*fmtobj*, **args*, ***kwargs*)

to_gaussian_gjf(**args*, ***kwargs*)

Dump data to *dpdata.plugins.gaussian.GaussiaGJFFormat* format.

to_gaussian_log(**args*, ***kwargs*)

Dump data to *dpdata.plugins.gaussian.GaussianLogFormat* format.

to_gaussian_md(**args*, ***kwargs*)

Dump data to *dpdata.plugins.gaussian.GaussianMDFormat* format.

to_gro(**args*, ***kwargs*)

Dump data to *dpdata.plugins.gromacs.GromacsGroFormat* format.

to_gromacs_gro(**args*, ***kwargs*)

Dump data to *dpdata.plugins.gromacs.GromacsGroFormat* format.

to_lammps_dump(**args*, ***kwargs*)

Dump data to *dpdata.plugins.lammps.LAMMPSDumpFormat* format.

to_lammps_lmp(**args*, ***kwargs*)

Dump data to *dpdata.plugins.lammps.LAMMPSLmpFormat* format.

to_list(**args*, ***kwargs*)

Dump data to *dpdata.plugins.list.ListFormat* format.

to_lmp(**args*, ***kwargs*)

Dump data to *dpdata.plugins.lammps.LAMMPSLmpFormat* format.

to_mlmd(**args*, ***kwargs*)

Dump data to *dpdata.plugins.pwmat.PwmatOutputFormat* format.

to_mol(**args*, ***kwargs*)

Dump data to *dpdata.plugins.rdkit.MolFormat* format.

to_mol_file(**args*, ***kwargs*)

Dump data to *dpdata.plugins.rdkit.MolFormat* format.

to_movement(**args*, ***kwargs*)

Dump data to *dpdata.plugins.pwmat.PwmatOutputFormat* format.

to_n2p2(**args*, ***kwargs*)

Dump data to *dpdata.plugins.n2p2.N2P2Format* format.

to_openmx_md(**args*, ***kwargs*)

Dump data to *dpdata.plugins.openmx.OPENMXFormat* format.

to_orca_spout(**args*, ***kwargs*)

Dump data to *dpdata.plugins.orca.ORCASPOutFormat* format.

to_outcar(**args*, ***kwargs*)

Dump data to *dpdata.plugins.vasp.VASPOutcarFormat* format.

to_poscar(**args*, ***kwargs*)

Dump data to *dpdata.plugins.vasp.VASPPoscarFormat* format.

to_psi4_inp(**args*, ***kwargs*)

Dump data to *dpdata.plugins.psi4.PSI4InputFormat* format.

to_psi4_out(**args*, ***kwargs*)

Dump data to *dpdata.plugins.psi4.PSI4OutFormat* format.

to_pwmat_atomconfig(**args*, ***kwargs*)

Dump data to *dpdata.plugins.pwmat.PwmatAtomconfigFormat* format.

to_pwmat_finalconfig(**args*, ***kwargs*)

Dump data to *dpdata.plugins.pwmat.PwmatAtomconfigFormat* format.

to_pwmat_mlmd(**args*, ***kwargs*)

Dump data to *dpdata.plugins.pwmat.PwmatOutputFormat* format.

to_pwmat_movement(**args*, ***kwargs*)

Dump data to *dpdata.plugins.pwmat.PwmatOutputFormat* format.

to_pwmat_output(**args*, ***kwargs*)

Dump data to *dpdata.plugins.pwmat.PwmatOutputFormat* format.

to_pymatgen_ComputedStructureEntry(**args*, ***kwargs*)

Dump data to *dpdata.plugins.pymatgen.PyMatgenCSEFormat* format.

to_pymatgen_computedstructureentry(**args*, ***kwargs*)

Dump data to *dpdata.plugins.pymatgen.PyMatgenCSEFormat* format.

to_pymatgen_molecule(**args*, ***kwargs*)

Dump data to *dpdata.plugins.pymatgen.PyMatgenMoleculeFormat* format.

to_pymatgen_structure(**args*, ***kwargs*)

Dump data to *dpdata.plugins.pymatgen.PyMatgenStructureFormat* format.

to_qe_cp_traj(**args*, ***kwargs*)

Dump data to *dpdata.plugins.qe.QECPTrajFormat* format.

to_qe_pw_scf(**args*, ***kwargs*)

Dump data to *dpdata.plugins.qe.QECPPWSCFFormat* format.

to_quip_gap_xyz(**args*, ***kwargs*)

Dump data to *dpdata.plugins.xyz.QuipGapXYZFormat* format.

to_quip_gap_xyz_file(**args*, ***kwargs*)

Dump data to *dpdata.plugins.xyz.QuipGapXYZFormat* format.

to_sdf(**args*, ***kwargs*)

Dump data to *dpdata.plugins.rdkit.SdfFormat* format.

to_sdf_file(**args*, ***kwargs*)

Dump data to *dpdata.plugins.rdkit.SdfFormat* format.

to_siesta_aimd_output(**args*, ***kwargs*)

Dump data to *dpdata.plugins.siesta.SiestaAIMDOutputFormat* format.

to_siesta_output(**args*, ***kwargs*)

Dump data to *dpdata.plugins.siesta.SiestaOutputFormat* format.

to_sqm_in(**args*, ***kwargs*)

Dump data to *dpdata.plugins.amber.SQMINFormat* format.

to_sqm_out(**args*, ***kwargs*)

Dump data to *dpdata.plugins.amber.SQMOutFormat* format.

to_stru(**args*, ***kwargs*)

Dump data to *dpdata.plugins.abacus.AbacusSTRUFormat* format.

to_vasp_contcar(**args*, ***kwargs*)

Dump data to *dpdata.plugins.vasp.VASPPoscarFormat* format.

to_vasp_outcar(**args*, ***kwargs*)

Dump data to *dpdata.plugins.vasp.VASPOutcarFormat* format.

to_vasp_poscar(**args*, ***kwargs*)

Dump data to *dpdata.plugins.vasp.VASPPoscarFormat* format.

to_vasp_string(**args*, ***kwargs*)

Dump data to *dpdata.plugins.vasp.VASPStringFormat* format.

to_vasp_xml(**args*, ***kwargs*)

Dump data to *dpdata.plugins.vasp.VASPXMLFormat* format.

to_xml(**args*, ***kwargs*)

Dump data to *dpdata.plugins.vasp.VASPXMLFormat* format.

to_xyz(**args*, ***kwargs*)

Dump data to *dpdata.plugins.xyz.XYZFormat* format.

class dpdata.system.**MultiSystems**(**systems*, *type_map=None*)

Bases: object

A set containing several systems.

Methods

append(*systems)	Append systems or MultiSystems to systems.
check_atom_names(system)	Make atom_names in all systems equal, prevent inconsistent atom_types.
correction(hl_sys)	Get energy and force correction between self (assumed low-level) and a high-level MultiSystems.
from_3dmol(file_name, **kwargs)	Read data from *dpdata.plugins.3dmol.Py3DMolFormat* format.
from_abacus_lcao_md(file_name, **kwargs)	Read data from *dpdata.plugins.abacus.AbacusMDFormat* format.
from_abacus_lcao_relax(file_name, **kwargs)	Read data from *dpdata.plugins.abacus.AbacusRelaxFormat* format.
from_abacus_lcao_scf(file_name, **kwargs)	Read data from *dpdata.plugins.abacus.AbacusSCFFormat* format.
from_abacus_md(file_name, **kwargs)	Read data from *dpdata.plugins.abacus.AbacusMDFormat* format.
from_abacus_pw_md(file_name, **kwargs)	Read data from *dpdata.plugins.abacus.AbacusMDFormat* format.
from_abacus_pw_relax(file_name, **kwargs)	Read data from *dpdata.plugins.abacus.AbacusRelaxFormat* format.
from_abacus_pw_scf(file_name, **kwargs)	Read data from *dpdata.plugins.abacus.AbacusSCFFormat* format.
from_abacus_relax(file_name, **kwargs)	Read data from *dpdata.plugins.abacus.AbacusRelaxFormat* format.
from_abacus_scf(file_name, **kwargs)	Read data from *dpdata.plugins.abacus.AbacusSCFFormat* format.
from_abacus_stru(file_name, **kwargs)	Read data from *dpdata.plugins.abacus.AbacusSTRUFormat* format.
from_amber_md(file_name, **kwargs)	Read data from *dpdata.plugins.amber.AmberMDFormat* format.
from_ase_structure(file_name, **kwargs)	Read data from *dpdata.plugins.ase.ASEStructureFormat* format.
from_ase_traj(file_name, **kwargs)	Read data from *dpdata.plugins.ase.ASETrajFormat* format.

continues on next page

Table 7 – continued from previous page

from_atomconfig(file_name, **kwargs)	Read data from *dpdata.plugins.pwmat.PwmatAtomconfigFormat* format.
from_contcar(file_name, **kwargs)	Read data from *dpdata.plugins.vasp.VASPPoscarFormat* format.
from_cp2k_aimd_output(file_name, **kwargs)	Read data from *dpdata.plugins.cp2k.CP2KAIMDOutputFormat* format.
from_cp2k_output(file_name, **kwargs)	Read data from *dpdata.plugins.cp2k.CP2KOutputFormat* format.
from_deepmd(file_name, **kwargs)	Read data from *dpdata.plugins.deepmd.DeePMDRawFormat* format.
from_deepmd_comp(file_name, **kwargs)	Read data from *dpdata.plugins.deepmd.DeePMDCompFormat* format.
from_deepmd_hdf5(file_name, **kwargs)	Read data from *dpdata.plugins.deepmd.DeePMDHDF5Format* format.
from_deepmd_npy(file_name, **kwargs)	Read data from *dpdata.plugins.deepmd.DeePMDCompFormat* format.
from_deepmd_npy_mixed(file_name, **kwargs)	Read data from *dpdata.plugins.deepmd.DeePMDMixedFormat* format.
from_deepmd_raw(file_name, **kwargs)	Read data from *dpdata.plugins.deepmd.DeePMDRawFormat* format.
from_dftbplus(file_name, **kwargs)	Read data from *dpdata.plugins.dftbplus.DFTBplusFormat* format.
from_dump(file_name, **kwargs)	Read data from *dpdata.plugins.lammps.LAMMPSDumpFormat* format.
from_fhi_aims_md(file_name, **kwargs)	Read data from *dpdata.plugins.fhi_aims.FhiMDFormat* format.
from_fhi_aims_output(file_name, **kwargs)	Read data from *dpdata.plugins.fhi_aims.FhiMDFormat* format.
from_fhi_aims_scf(file_name, **kwargs)	Read data from *dpdata.plugins.fhi_aims.FhiSCFFormat* format.
from_finalconfig(file_name, **kwargs)	Read data from *dpdata.plugins.pwmat.PwmatAtomconfigFormat* format.
from_gaussian_gjf(file_name, **kwargs)	Read data from *dpdata.plugins.gaussian.GaussiaGJFFormat* format.
from_gaussian_log(file_name, **kwargs)	Read data from *dpdata.plugins.gaussian.GaussianLogFormat* format.
from_gaussian_md(file_name, **kwargs)	Read data from *dpdata.plugins.gaussian.GaussianMDFormat* format.
from_gro(file_name, **kwargs)	Read data from *dpdata.plugins.gromacs.GromacsGroFormat* format.
from_gromacs_gro(file_name, **kwargs)	Read data from *dpdata.plugins.gromacs.GromacsGroFormat* format.
from_lammps_dump(file_name, **kwargs)	Read data from *dpdata.plugins.lammps.LAMMPSDumpFormat* format.
from_lammps_lmp(file_name, **kwargs)	Read data from *dpdata.plugins.lammps.LAMMPSLmpFormat* format.
from_list(file_name, **kwargs)	Read data from *dpdata.plugins.list.ListFormat* format.
from_lmp(file_name, **kwargs)	Read data from *dpdata.plugins.lammps.LAMMPSLmpFormat* format.
from_mlmd(file_name, **kwargs)	Read data from *dpdata.plugins.pwmat.PwmatOutputFormat* format.

continues on next page

Table 7 – continued from previous page

from_mol(file_name, **kwargs)	Read data from *dpdata.plugins.rdkit.MolFormat* format.
from_mol_file(file_name, **kwargs)	Read data from *dpdata.plugins.rdkit.MolFormat* format.
from_movement(file_name, **kwargs)	Read data from *dpdata.plugins.pwmat.PwmatOutputFormat* format.
from_n2p2(file_name, **kwargs)	Read data from *dpdata.plugins.n2p2.N2P2Format* format.
from_openmx_md(file_name, **kwargs)	Read data from *dpdata.plugins.openmx.OPENMXFormat* format.
from_orca_spout(file_name, **kwargs)	Read data from *dpdata.plugins.orca.ORCASPOutFormat* format.
from_outcar(file_name, **kwargs)	Read data from *dpdata.plugins.vasp.VASPOutcarFormat* format.
from_poscar(file_name, **kwargs)	Read data from *dpdata.plugins.vasp.VASPPoscarFormat* format.
from_psi4_inp(file_name, **kwargs)	Read data from *dpdata.plugins.psi4.PSI4InputFormat* format.
from_psi4_out(file_name, **kwargs)	Read data from *dpdata.plugins.psi4.PSI4OutFormat* format.
from_pwmat_atomconfig(file_name, **kwargs)	Read data from *dpdata.plugins.pwmat.PwmatAtomconfigFormat* format.
from_pwmat_finalconfig(file_name, **kwargs)	Read data from *dpdata.plugins.pwmat.PwmatAtomconfigFormat* format.
from_pwmat_mlmd(file_name, **kwargs)	Read data from *dpdata.plugins.pwmat.PwmatOutputFormat* format.
from_pwmat_movement(file_name, **kwargs)	Read data from *dpdata.plugins.pwmat.PwmatOutputFormat* format.
from_pwmat_output(file_name, **kwargs)	Read data from *dpdata.plugins.pwmat.PwmatOutputFormat* format.
from_pymatgen_computedstructureentry(...)	Read data from *dpdata.plugins.pymatgen.PyMatgenCSEFormat* format.
from_pymatgen_molecule(file_name, **kwargs)	Read data from *dpdata.plugins.pymatgen.PyMatgenMoleculeFormat* format.
from_pymatgen_structure(file_name, **kwargs)	Read data from *dpdata.plugins.pymatgen.PyMatgenStructureFormat* format.
from_qe_cp_traj(file_name, **kwargs)	Read data from *dpdata.plugins.qe.QECPTrajFormat* format.
from_qe_pw_scf(file_name, **kwargs)	Read data from *dpdata.plugins.qe.QECPPWSCFFormat* format.
from_quip_gap_xyz(file_name, **kwargs)	Read data from *dpdata.plugins.xyz.QuipGapXYZFormat* format.
from_quip_gap_xyz_file(file_name, **kwargs)	Read data from *dpdata.plugins.xyz.QuipGapXYZFormat* format.
from_sdf(file_name, **kwargs)	Read data from *dpdata.plugins.rdkit.SdfFormat* format.
from_sdf_file(file_name, **kwargs)	Read data from *dpdata.plugins.rdkit.SdfFormat* format.
from_siesta_aiMD_output(file_name, **kwargs)	Read data from *dpdata.plugins.siesta.SiestaAIMDOutputFormat* format.
from_siesta_aimd_output(file_name, **kwargs)	Read data from *dpdata.plugins.siesta.SiestaAIMDOutputFormat* format.

continues on next page

Table 7 – continued from previous page

from_siesta_output(file_name, **kwargs)	Read data from *dpdata.plugins.siesta.SiestaOutputFormat* format.
from_sqm_in(file_name, **kwargs)	Read data from *dpdata.plugins.amber.SQMINFormat* format.
from_sqm_out(file_name, **kwargs)	Read data from *dpdata.plugins.amber.SQMOutFormat* format.
from_stru(file_name, **kwargs)	Read data from *dpdata.plugins.abacus.AbacusSTRUFormat* format.
from_vasp_contcar(file_name, **kwargs)	Read data from *dpdata.plugins.vasp.VASPPoscarFormat* format.
from_vasp_outcar(file_name, **kwargs)	Read data from *dpdata.plugins.vasp.VASPOutcarFormat* format.
from_vasp_poscar(file_name, **kwargs)	Read data from *dpdata.plugins.vasp.VASPPoscarFormat* format.
from_vasp_string(file_name, **kwargs)	Read data from *dpdata.plugins.vasp.VASPStringFormat* format.
from_vasp_xml(file_name, **kwargs)	Read data from *dpdata.plugins.vasp.VASPXMLFormat* format.
from_xml(file_name, **kwargs)	Read data from *dpdata.plugins.vasp.VASPXMLFormat* format.
from_xyz(file_name, **kwargs)	Read data from *dpdata.plugins.xyz.XYZFormat* format.
get_nframes()	Returns number of frames in all systems.
minimize(*args, minimizer, **kwargs)	Minimize geometry by a minimizer.
pick_atom_idx(idx[, nopbc])	Pick atom index.
predict(*args[, driver])	Predict energies and forces by a driver.
to(fmt, *args, **kwargs)	Dump systems to the specific format.
to_3dmol(*args, **kwargs)	Dump data to *dpdata.plugins.3dmol.Py3DMolFormat* format.
to_abacus_lcao_md(*args, **kwargs)	Dump data to *dpdata.plugins.abacus.AbacusMDFormat* format.
to_abacus_lcao_relax(*args, **kwargs)	Dump data to *dpdata.plugins.abacus.AbacusRelaxFormat* format.
to_abacus_lcao_scf(*args, **kwargs)	Dump data to *dpdata.plugins.abacus.AbacusSCFFormat* format.
to_abacus_md(*args, **kwargs)	Dump data to *dpdata.plugins.abacus.AbacusMDFormat* format.
to_abacus_pw_md(*args, **kwargs)	Dump data to *dpdata.plugins.abacus.AbacusMDFormat* format.
to_abacus_pw_relax(*args, **kwargs)	Dump data to *dpdata.plugins.abacus.AbacusRelaxFormat* format.
to_abacus_pw_scf(*args, **kwargs)	Dump data to *dpdata.plugins.abacus.AbacusSCFFormat* format.
to_abacus_relax(*args, **kwargs)	Dump data to *dpdata.plugins.abacus.AbacusRelaxFormat* format.
to_abacus_scf(*args, **kwargs)	Dump data to *dpdata.plugins.abacus.AbacusSCFFormat* format.
to_abacus_stru(*args, **kwargs)	Dump data to *dpdata.plugins.abacus.AbacusSTRUFormat* format.
to_amber_md(*args, **kwargs)	Dump data to *dpdata.plugins.amber.AmberMDFormat* format.

continues on next page

Table 7 – continued from previous page

to_ase_structure(*args, **kwargs)	Dump data to dpdata.plugins.ase.ASEStructureFormat format.
to_ase_traj(*args, **kwargs)	Dump data to dpdata.plugins.ase.ASETrajFormat format.
to_atomconfig(*args, **kwargs)	Dump data to dpdata.plugins.pwmat.PwmatAtomconfigFormat format.
to_contcar(*args, **kwargs)	Dump data to dpdata.plugins.vasp.VASPPoscarFormat format.
to_cp2k_aimd_output(*args, **kwargs)	Dump data to dpdata.plugins.cp2k.CP2KAIMDOutputFormat format.
to_cp2k_output(*args, **kwargs)	Dump data to dpdata.plugins.cp2k.CP2KOutputFormat format.
to_deepmd(*args, **kwargs)	Dump data to dpdata.plugins.deepmd.DeePMDRawFormat format.
to_deepmd_comp(*args, **kwargs)	Dump data to dpdata.plugins.deepmd.DeePMDCompFormat format.
to_deepmd_hdf5(*args, **kwargs)	Dump data to dpdata.plugins.deepmd.DeePMDHDF5Format format.
to_deepmd_npy(*args, **kwargs)	Dump data to dpdata.plugins.deepmd.DeePMDCompFormat format.
to_deepmd_npy_mixed(*args, **kwargs)	Dump data to dpdata.plugins.deepmd.DeePMDMixedFormat format.
to_deepmd_raw(*args, **kwargs)	Dump data to dpdata.plugins.deepmd.DeePMDRawFormat format.
to_dftbplus(*args, **kwargs)	Dump data to dpdata.plugins.dftbplus.DFTBplusFormat format.
to_dump(*args, **kwargs)	Dump data to dpdata.plugins.lammps.LAMMPSDumpFormat format.
to_fhi_aims_md(*args, **kwargs)	Dump data to dpdata.plugins.fhi_aims.FhiMDFormat format.
to_fhi_aims_output(*args, **kwargs)	Dump data to dpdata.plugins.fhi_aims.FhiMDFormat format.
to_fhi_aims_scf(*args, **kwargs)	Dump data to dpdata.plugins.fhi_aims.FhiSCFFormat format.
to_finalconfig(*args, **kwargs)	Dump data to dpdata.plugins.pwmat.PwmatAtomconfigFormat format.
to_gaussian_gjf(*args, **kwargs)	Dump data to dpdata.plugins.gaussian.GaussiaGJFFormat format.
to_gaussian_log(*args, **kwargs)	Dump data to dpdata.plugins.gaussian.GaussianLogFormat format.
to_gaussian_md(*args, **kwargs)	Dump data to dpdata.plugins.gaussian.GaussianMDFormat format.
to_gro(*args, **kwargs)	Dump data to dpdata.plugins.gromacs.GromacsGroFormat format.
to_gromacs_gro(*args, **kwargs)	Dump data to dpdata.plugins.gromacs.GromacsGroFormat format.
to_lammps_dump(*args, **kwargs)	Dump data to dpdata.plugins.lammps.LAMMPSDumpFormat format.
to_lammps_lmp(*args, **kwargs)	Dump data to dpdata.plugins.lammps.LAMMPSLmpFormat format.
to_list(*args, **kwargs)	Dump data to dpdata.plugins.list.ListFormat format.

continues on next page

Table 7 – continued from previous page

to_lmp(*args, **kwargs)	Dump data to dpdata.plugins.lammps.LAMMPSLmpFormat format.
to_mlmd(*args, **kwargs)	Dump data to dpdata.plugins.pwmat.PwmatOutputFormat format.
to_mol(*args, **kwargs)	Dump data to dpdata.plugins.rdkit.MolFormat format.
to_mol_file(*args, **kwargs)	Dump data to dpdata.plugins.rdkit.MolFormat format.
to_movement(*args, **kwargs)	Dump data to dpdata.plugins.pwmat.PwmatOutputFormat format.
to_n2p2(*args, **kwargs)	Dump data to dpdata.plugins.n2p2.N2P2Format format.
to_openmx_md(*args, **kwargs)	Dump data to dpdata.plugins.openmx.OPENMXFormat format.
to_orca_spout(*args, **kwargs)	Dump data to dpdata.plugins.orca.ORCASPOutFormat format.
to_outcar(*args, **kwargs)	Dump data to dpdata.plugins.vasp.VASPOutcarFormat format.
to_poscar(*args, **kwargs)	Dump data to dpdata.plugins.vasp.VASPPoscarFormat format.
to_psi4_inp(*args, **kwargs)	Dump data to dpdata.plugins.psi4.PSI4InputFormat format.
to_psi4_out(*args, **kwargs)	Dump data to dpdata.plugins.psi4.PSI4OutFormat format.
to_pwmat_atomconfig(*args, **kwargs)	Dump data to dpdata.plugins.pwmat.PwmatAtomconfigFormat format.
to_pwmat_finalconfig(*args, **kwargs)	Dump data to dpdata.plugins.pwmat.PwmatAtomconfigFormat format.
to_pwmat_mlmd(*args, **kwargs)	Dump data to dpdata.plugins.pwmat.PwmatOutputFormat format.
to_pwmat_movement(*args, **kwargs)	Dump data to dpdata.plugins.pwmat.PwmatOutputFormat format.
to_pwmat_output(*args, **kwargs)	Dump data to dpdata.plugins.pwmat.PwmatOutputFormat format.
to_pymatgen_ComputedStructureEntry(*args, ...)	Dump data to dpdata.plugins.pymatgen.PyMatgenCSEFormat format.
to_pymatgen_computedstructureentry(*args, ...)	Dump data to dpdata.plugins.pymatgen.PyMatgenCSEFormat format.
to_pymatgen_molecule(*args, **kwargs)	Dump data to dpdata.plugins.pymatgen.PyMatgenMoleculeFormat format.
to_pymatgen_structure(*args, **kwargs)	Dump data to dpdata.plugins.pymatgen.PyMatgenStructureFormat format.
to_qe_cp_traj(*args, **kwargs)	Dump data to dpdata.plugins.qe.QECPTrajFormat format.
to_qe_pw_scf(*args, **kwargs)	Dump data to dpdata.plugins.qe.QECPPWSCFFormat format.
to_quip_gap_xyz(*args, **kwargs)	Dump data to dpdata.plugins.xyz.QuipGapXYZFormat format.
to_quip_gap_xyz_file(*args, **kwargs)	Dump data to dpdata.plugins.xyz.QuipGapXYZFormat format.
to_sdf(*args, **kwargs)	Dump data to dpdata.plugins.rdkit.SdfFormat format.

continues on next page

Table 7 – continued from previous page

to_sdf_file(*args, **kwargs)	Dump data to dpdata.plugins.rdkit.SdfFormat format.
to_siesta_aimd_output(*args, **kwargs)	Dump data to dpdata.plugins.siesta.SiestaAIMDOutputFormat format.
to_siesta_output(*args, **kwargs)	Dump data to dpdata.plugins.siesta.SiestaOutputFormat format.
to_sqm_in(*args, **kwargs)	Dump data to dpdata.plugins.amber.SQMINFormat format.
to_sqm_out(*args, **kwargs)	Dump data to dpdata.plugins.amber.SQMOutFormat format.
to_stru(*args, **kwargs)	Dump data to dpdata.plugins.abacus.AbacusSTRUFormat format.
to_vasp_contcar(*args, **kwargs)	Dump data to dpdata.plugins.vasp.VASPPoscarFormat format.
to_vasp_outcar(*args, **kwargs)	Dump data to dpdata.plugins.vasp.VASPOutcarFormat format.
to_vasp_poscar(*args, **kwargs)	Dump data to dpdata.plugins.vasp.VASPPoscarFormat format.
to_vasp_string(*args, **kwargs)	Dump data to dpdata.plugins.vasp.VASPStringFormat format.
to_vasp_xml(*args, **kwargs)	Dump data to dpdata.plugins.vasp.VASPXMLFormat format.
to_xml(*args, **kwargs)	Dump data to dpdata.plugins.vasp.VASPXMLFormat format.
to_xyz(*args, **kwargs)	Dump data to dpdata.plugins.xyz.XYZFormat format.
train_test_split(test_size[, seed])	Split systems into random train and test subsets.

from_dir
from_file
from_fmt_obj
load_systems_from_file
to_fmt_obj

append(**systems*)

Append systems or MultiSystems to systems.

Parameters

***systems**
[System] The system to append

check_atom_names(*system*)

Make atom_names in all systems equal, prevent inconsistent atom_types.

correction(*hl_sys:* MultiSystems)

Get energy and force correction between self (assumed low-level) and a high-level MultiSystems. The self's coordinates will be kept, but energy and forces will be replaced by the correction between these two systems.

Parameters

hl_sys
[MultiSystems] high-level MultiSystems

Returns

corrected_sys
[MultiSystems] Corrected MultiSystems

Notes

This method will not check whether coordinates and elements of two systems are the same. The user should make sure by itself.

Examples

Get correction between a low-level system and a high-level system:

```
>>> low_level = dpdata.MultiSystems().from_deepmd_hdf5("low_level.hdf5")
>>> high_level = dpdata.MultiSystems().from_deepmd_hdf5("high_level.hdf5")
>>> corr = low_level.correction(high_lebel)
>>> corr.to_deepmd_hdf5("corr.hdf5")
```

from_3dmol(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.3dmol.Py3DMolFormat* format.

from_abacus_lcao_md(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.abacus.AbacusMDFormat* format.

from_abacus_lcao_relax(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.abacus.AbacusRelaxFormat* format.

from_abacus_lcao_scf(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.abacus.AbacusSCFFormat* format.

from_abacus_md(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.abacus.AbacusMDFormat* format.

from_abacus_pw_md(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.abacus.AbacusMDFormat* format.

from_abacus_pw_relax(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.abacus.AbacusRelaxFormat* format.

from_abacus_pw_scf(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.abacus.AbacusSCFFormat* format.

from_abacus_relax(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.abacus.AbacusRelaxFormat* format.

from_abacus_scf(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.abacus.AbacusSCFFormat* format.

from_abacus_stru(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.abacus.AbacusSTRUFormat* format.

from_amber_md(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.amber.AmberMDFormat* format.

from_ase_structure(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.ase.ASEStructureFormat* format.

from_ase_traj(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.ase.ASETrajFormat* format.

from_atomconfig(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.pwmat.PwmatAtomconfigFormat* format.

from_contcar(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.vasp.VASPPoscarFormat* format.

from_cp2k_aimd_output(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.cp2k.CP2KAIMDOutputFormat* format.

from_cp2k_output(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.cp2k.CP2KOutputFormat* format.

from_deepmd(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.deepmd.DeePMDRawFormat* format.

from_deepmd_comp(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.deepmd.DeePMDCompFormat* format.

from_deepmd_hdf5(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.deepmd.DeePMDHDF5Format* format.

from_deepmd_npy(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.deepmd.DeePMDCompFormat* format.

from_deepmd_npy_mixed(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.deepmd.DeePMDMixedFormat* format.

from_deepmd_raw(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.deepmd.DeePMDRawFormat* format.

from_dftbplus(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.dftbplus.DFTBplusFormat* format.

classmethod from_dir(*dir_name*, *file_name*, *fmt='auto'*, *type_map=None*)

from_dump(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.lammps.LAMMPSDumpFormat* format.

from_fhi_aims_md(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.fhi_aims.FhiMDFormat* format.

from_fhi_aims_output(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.fhi_aims.FhiMDFormat* format.

from_fhi_aims_scf(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.fhi_aims.FhiSCFFormat* format.

classmethod from_file(*file_name*, *fmt*, ***kwargs*)

from_finalconfig(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.pwmat.PwmatAtomconfigFormat* format.

from_fmt_obj(*fmtobj*, *directory*, *labeled=True*, ***kwargs*)

from_gaussian_gjf(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.gaussian.GaussiaGJFFormat* format.

from_gaussian_log(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.gaussian.GaussianLogFormat* format.

from_gaussian_md(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.gaussian.GaussianMDFormat* format.

from_gro(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.gromacs.GromacsGroFormat* format.

from_gromacs_gro(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.gromacs.GromacsGroFormat* format.

from_lammps_dump(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.lammps.LAMMPSDumpFormat* format.

from_lammps_lmp(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.lammps.LAMMPSLmpFormat* format.

from_list(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.list.ListFormat* format.

from_lmp(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.lammps.LAMMPSLmpFormat* format.

from_mlmd(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.pwmat.PwmatOutputFormat* format.

from_mol(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.rdkit.MolFormat* format.

from_mol_file(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.rdkit.MolFormat* format.

from_movement(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.pwmat.PwmatOutputFormat* format.

from_n2p2(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.n2p2.N2P2Format* format.

from_openmx_md(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.openmx.OPENMXFormat* format.

from_orca_spout(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.orca.ORCASPOutFormat* format.

from_outcar(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.vasp.VASPOutcarFormat* format.

from_poscar(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.vasp.VASPPoscarFormat* format.

from_psi4_inp(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.psi4.PSI4InputFormat* format.

from_psi4_out(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.psi4.PSI4OutFormat* format.

from_pwmat_atomconfig(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.pwmat.PwmatAtomconfigFormat* format.

from_pwmat_finalconfig(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.pwmat.PwmatAtomconfigFormat* format.

from_pwmat_mlmd(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.pwmat.PwmatOutputFormat* format.

from_pwmat_movement(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.pwmat.PwmatOutputFormat* format.

from_pwmat_output(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.pwmat.PwmatOutputFormat* format.

from_pymatgen_computedstructureentry(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.pymatgen.PyMatgenCSEFormat* format.

from_pymatgen_molecule(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.pymatgen.PyMatgenMoleculeFormat* format.

from_pymatgen_structure(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.pymatgen.PyMatgenStructureFormat* format.

from_qe_cp_traj(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.qe.QECPTrajFormat* format.

from_qe_pw_scf(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.qe.QECPPWSCFFormat* format.

from_quip_gap_xyz(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.xyz.QuipGapXYZFormat* format.

from_quip_gap_xyz_file(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.xyz.QuipGapXYZFormat* format.

from_sdf(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.rdkit.SdfFormat* format.

from_sdf_file(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.rdkit.SdfFormat* format.

from_siesta_aiMD_output(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.siesta.SiestaAIMDOutputFormat* format.

from_siesta_aimd_output(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.siesta.SiestaAIMDOutputFormat* format.

from_siesta_output(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.siesta.SiestaOutputFormat* format.

from_sqm_in(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.amber.SQMINFormat* format.

from_sqm_out(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.amber.SQMOutFormat* format.

from_stru(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.abacus.AbacusSTRUFormat* format.

from_vasp_contcar(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.vasp.VASPPoscarFormat* format.

from_vasp_outcar(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.vasp.VASPOutcarFormat* format.

from_vasp_poscar(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.vasp.VASPPoscarFormat* format.

from_vasp_string(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.vasp.VASPStringFormat* format.

from_vasp_xml(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.vasp.VASPXMLFormat* format.

from_xml(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.vasp.VASPXMLFormat* format.

from_xyz(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.xyz.XYZFormat* format.

get_nframes()

Returns number of frames in all systems.

load_systems_from_file(*file_name=None*, *fmt=None*, ***kwargs*)

minimize(**args: Any*, *minimizer: str* | Minimizer, ***kwargs: Any*) → *MultiSystems*

Minimize geometry by a minimizer.

Parameters

***args**
[iterable] Arguments passing to the minimizer

minimizer
[str or Minimizer] The assigned minimizer

****kwargs**
[dict] Other arguments passing to the minimizer

Returns

MultiSystems
A new labeled MultiSystems.

Examples

Minimize a system using ASE BFGS along with a DP driver:

```
>>> from dpdata.driver import Driver
>>> from ase.optimize import BFGS
>>> driver = Driver.get_driver("dp")("some_model.pb")
>>> some_system.minimize(minimizer="ase", driver=driver, optimizer=BFGS,␣
↪fmax=1e-5)
```

pick_atom_idx(*idx*, *nopbc=None*)

Pick atom index.

Parameters

idx
[int or list or slice] atom index

nopbc
[Boolen (default: None)] If nopbc is True or False, set nopbc

Returns

new_sys: MultiSystems
new system

predict(**args: Any*, *driver='dp'*, ***kwargs: Any*) → *MultiSystems*

Predict energies and forces by a driver.

Parameters

***args**
[iterable] Arguments passing to the driver

driver
[str, default=dp] The assigned driver. For compatibility, default is dp

****kwargs**
[dict] Other arguments passing to the driver

Returns

MultiSystems
A new labeled MultiSystems.

to(*fmt: str*, **args*, ***kwargs*) → *MultiSystems*

Dump systems to the specific format.

Parameters

fmt
[str] format

***args**
[list] arguments

****kwargs**
[dict] keyword arguments

Returns

MultiSystems
self

to_3dmol(**args*, ***kwargs*)

Dump data to *dpdata.plugins.3dmol.Py3DMolFormat* format.

to_abacus_lcao_md(**args*, ***kwargs*)

Dump data to *dpdata.plugins.abacus.AbacusMDFormat* format.

to_abacus_lcao_relax(**args*, ***kwargs*)

Dump data to *dpdata.plugins.abacus.AbacusRelaxFormat* format.

to_abacus_lcao_scf(**args*, ***kwargs*)

Dump data to *dpdata.plugins.abacus.AbacusSCFFormat* format.

to_abacus_md(**args*, ***kwargs*)

Dump data to *dpdata.plugins.abacus.AbacusMDFormat* format.

to_abacus_pw_md(**args*, ***kwargs*)

Dump data to *dpdata.plugins.abacus.AbacusMDFormat* format.

to_abacus_pw_relax(**args*, ***kwargs*)

Dump data to *dpdata.plugins.abacus.AbacusRelaxFormat* format.

to_abacus_pw_scf(**args*, ***kwargs*)

Dump data to *dpdata.plugins.abacus.AbacusSCFFormat* format.

to_abacus_relax(**args*, ***kwargs*)

Dump data to *dpdata.plugins.abacus.AbacusRelaxFormat* format.

to_abacus_scf(**args*, ***kwargs*)

Dump data to *dpdata.plugins.abacus.AbacusSCFFormat* format.

to_abacus_stru(**args*, ***kwargs*)

Dump data to *dpdata.plugins.abacus.AbacusSTRUFormat* format.

to_amber_md(**args*, ***kwargs*)

Dump data to *dpdata.plugins.amber.AmberMDFormat* format.

to_ase_structure(**args*, ***kwargs*)

Dump data to *dpdata.plugins.ase.ASEStructureFormat* format.

to_ase_traj(**args*, ***kwargs*)

Dump data to *dpdata.plugins.ase.ASETrajFormat* format.

to_atomconfig(**args*, ***kwargs*)

Dump data to *dpdata.plugins.pwmat.PwmatAtomconfigFormat* format.

to_contcar(**args*, ***kwargs*)

Dump data to *dpdata.plugins.vasp.VASPPoscarFormat* format.

to_cp2k_aimd_output(**args*, ***kwargs*)

Dump data to *dpdata.plugins.cp2k.CP2KAIMDOutputFormat* format.

to_cp2k_output(**args*, ***kwargs*)

Dump data to *dpdata.plugins.cp2k.CP2KOutputFormat* format.

to_deepmd(**args*, ***kwargs*)

Dump data to *dpdata.plugins.deepmd.DeePMDRawFormat* format.

to_deepmd_comp(**args*, ***kwargs*)

Dump data to *dpdata.plugins.deepmd.DeePMDCompFormat* format.

to_deepmd_hdf5(**args*, ***kwargs*)

Dump data to *dpdata.plugins.deepmd.DeePMDHDF5Format* format.

to_deepmd_npy(**args*, ***kwargs*)

Dump data to *dpdata.plugins.deepmd.DeePMDCompFormat* format.

to_deepmd_npy_mixed(**args*, ***kwargs*)

Dump data to *dpdata.plugins.deepmd.DeePMDMixedFormat* format.

to_deepmd_raw(**args*, ***kwargs*)

Dump data to *dpdata.plugins.deepmd.DeePMDRawFormat* format.

to_dftbplus(**args*, ***kwargs*)

Dump data to *dpdata.plugins.dftbplus.DFTBplusFormat* format.

to_dump(**args*, ***kwargs*)

Dump data to *dpdata.plugins.lammps.LAMMPSDumpFormat* format.

to_fhi_aims_md(**args*, ***kwargs*)

Dump data to *dpdata.plugins.fhi_aims.FhiMDFormat* format.

to_fhi_aims_output(**args*, ***kwargs*)

Dump data to *dpdata.plugins.fhi_aims.FhiMDFormat* format.

to_fhi_aims_scf(**args*, ***kwargs*)

Dump data to *dpdata.plugins.fhi_aims.FhiSCFFormat* format.

to_finalconfig(**args*, ***kwargs*)

Dump data to *dpdata.plugins.pwmat.PwmatAtomconfigFormat* format.

to_fmt_obj(*fmtobj*, *directory*, **args*, ***kwargs*)

to_gaussian_gjf(**args*, ***kwargs*)

Dump data to *dpdata.plugins.gaussian.GaussiaGJFFormat* format.

to_gaussian_log(**args*, ***kwargs*)

Dump data to *dpdata.plugins.gaussian.GaussianLogFormat* format.

to_gaussian_md(**args*, ***kwargs*)

Dump data to *dpdata.plugins.gaussian.GaussianMDFormat* format.

to_gro(**args*, ***kwargs*)

Dump data to *dpdata.plugins.gromacs.GromacsGroFormat* format.

to_gromacs_gro(**args*, ***kwargs*)

Dump data to *dpdata.plugins.gromacs.GromacsGroFormat* format.

to_lammps_dump(**args*, ***kwargs*)

Dump data to *dpdata.plugins.lammps.LAMMPSDumpFormat* format.

to_lammps_lmp(**args*, ***kwargs*)

Dump data to *dpdata.plugins.lammps.LAMMPSLmpFormat* format.

to_list(**args*, ***kwargs*)

Dump data to *dpdata.plugins.list.ListFormat* format.

to_lmp(**args*, ***kwargs*)

Dump data to *dpdata.plugins.lammps.LAMMPSLmpFormat* format.

to_mlmd(**args*, ***kwargs*)

Dump data to *dpdata.plugins.pwmat.PwmatOutputFormat* format.

to_mol(**args*, ***kwargs*)

Dump data to *dpdata.plugins.rdkit.MolFormat* format.

to_mol_file(**args*, ***kwargs*)

Dump data to *dpdata.plugins.rdkit.MolFormat* format.

to_movement(**args*, ***kwargs*)

Dump data to *dpdata.plugins.pwmat.PwmatOutputFormat* format.

to_n2p2(**args*, ***kwargs*)

Dump data to *dpdata.plugins.n2p2.N2P2Format* format.

to_openmx_md(**args*, ***kwargs*)

Dump data to *dpdata.plugins.openmx.OPENMXFormat* format.

to_orca_spout(**args*, ***kwargs*)

Dump data to *dpdata.plugins.orca.ORCASPOutFormat* format.

to_outcar(**args*, ***kwargs*)

Dump data to *dpdata.plugins.vasp.VASPOutcarFormat* format.

to_poscar(**args*, ***kwargs*)

Dump data to *dpdata.plugins.vasp.VASPPoscarFormat* format.

to_psi4_inp(**args*, ***kwargs*)

Dump data to *dpdata.plugins.psi4.PSI4InputFormat* format.

to_psi4_out(**args*, ***kwargs*)

Dump data to *dpdata.plugins.psi4.PSI4OutFormat* format.

to_pwmat_atomconfig(**args*, ***kwargs*)

Dump data to *dpdata.plugins.pwmat.PwmatAtomconfigFormat* format.

to_pwmat_finalconfig(**args*, ***kwargs*)

Dump data to *dpdata.plugins.pwmat.PwmatAtomconfigFormat* format.

to_pwmat_mlmd(**args*, ***kwargs*)

Dump data to *dpdata.plugins.pwmat.PwmatOutputFormat* format.

to_pwmat_movement(**args*, ***kwargs*)

Dump data to *dpdata.plugins.pwmat.PwmatOutputFormat* format.

to_pwmat_output(**args*, ***kwargs*)

Dump data to *dpdata.plugins.pwmat.PwmatOutputFormat* format.

to_pymatgen_ComputedStructureEntry(**args*, ***kwargs*)

Dump data to *dpdata.plugins.pymatgen.PyMatgenCSEFormat* format.

to_pymatgen_computedstructureentry(**args*, ***kwargs*)

Dump data to *dpdata.plugins.pymatgen.PyMatgenCSEFormat* format.

to_pymatgen_molecule(**args*, ***kwargs*)

Dump data to *dpdata.plugins.pymatgen.PyMatgenMoleculeFormat* format.

to_pymatgen_structure(**args*, ***kwargs*)

Dump data to *dpdata.plugins.pymatgen.PyMatgenStructureFormat* format.

to_qe_cp_traj(**args*, ***kwargs*)

Dump data to *dpdata.plugins.qe.QECPTrajFormat* format.

to_qe_pw_scf(**args*, ***kwargs*)

Dump data to *dpdata.plugins.qe.QECPPWSCFFormat* format.

to_quip_gap_xyz(**args*, ***kwargs*)

Dump data to *dpdata.plugins.xyz.QuipGapXYZFormat* format.

to_quip_gap_xyz_file(**args*, ***kwargs*)

Dump data to *dpdata.plugins.xyz.QuipGapXYZFormat* format.

to_sdf(**args*, ***kwargs*)

Dump data to *dpdata.plugins.rdkit.SdfFormat* format.

to_sdf_file(**args*, ***kwargs*)

Dump data to *dpdata.plugins.rdkit.SdfFormat* format.

to_siesta_aimd_output(**args*, ***kwargs*)

Dump data to *dpdata.plugins.siesta.SiestaAIMDOutputFormat* format.

to_siesta_output(**args*, ***kwargs*)

Dump data to *dpdata.plugins.siesta.SiestaOutputFormat* format.

to_sqm_in(**args*, ***kwargs*)

Dump data to *dpdata.plugins.amber.SQMINFormat* format.

to_sqm_out(**args*, ***kwargs*)

Dump data to *dpdata.plugins.amber.SQMOutFormat* format.

to_stru(**args*, ***kwargs*)

Dump data to *dpdata.plugins.abacus.AbacusSTRUFormat* format.

to_vasp_contcar(**args*, ***kwargs*)

Dump data to *dpdata.plugins.vasp.VASPPoscarFormat* format.

to_vasp_outcar(**args*, ***kwargs*)

Dump data to *dpdata.plugins.vasp.VASPOutcarFormat* format.

to_vasp_poscar(**args*, ***kwargs*)

Dump data to *dpdata.plugins.vasp.VASPPoscarFormat* format.

to_vasp_string(**args*, ***kwargs*)

Dump data to *dpdata.plugins.vasp.VASPStringFormat* format.

to_vasp_xml(**args*, ***kwargs*)

Dump data to *dpdata.plugins.vasp.VASPXMLFormat* format.

to_xml(**args*, ***kwargs*)

Dump data to *dpdata.plugins.vasp.VASPXMLFormat* format.

to_xyz(**args*, ***kwargs*)

Dump data to *dpdata.plugins.xyz.XYZFormat* format.

train_test_split(*test_size: float | int*, *seed: int | None = None*) → Tuple[*MultiSystems*, *MultiSystems*, Dict[str, ndarray]]

Split systems into random train and test subsets.

Parameters

test_size
[float or int] If float, should be between 0.0 and 1.0 and represent the proportion of the dataset to include in the test split. If int, represents the absolute number of test samples.

seed
[int, default=None] Random seed

Returns

MultiSystems
The training set

MultiSystems
The testing set

Dict[str, np.ndarray]
The bool array of training and testing sets for each system. False for training set and True for testing set.

class dpdata.system.**System**(*file_name=None*, *fmt='auto'*, *type_map=None*, *begin=0*, *step=1*, *data=None*, *convergence_check=True*, ***kwargs*)

Bases: MSONable

The data System.

A data System (a concept used by deepmd-kit) contains frames (e.g. produced by an MD simulation) that has the same number of atoms of the same type. The order of the atoms should be consistent among the frames in one System.

For example, a water system named *d_example* has two molecules. The properties can be accessed by

- *d_example['atom_numbs']* : [2, 4]
- *d_example['atom_names']* : ['O', 'H']
- *d_example['atom_types']* : [0, 1, 1, 0, 1, 1]
- *d_example['orig']* : [0, 0, 0]
- *d_example['cells']* : a numpy array of size nframes x 3 x 3
- *d_example['coords']* : a numpy array of size nframes x natoms x 3

It is noted that

- The order of frames stored in *'atom_types'*, *'cells'* and *'coords'* should be consistent.
- The order of atoms in **all** frames of *'atom_types'* and *'coords'* should be consistent.

Restrictions:

- *d_example['orig']* is always [0, 0, 0]
- *d_example['cells'][ii]* is always lower triangular (lammps cell tensor convention)

Attributes

DTYPES
[tuple[DataType]] data types of this class

Methods

add_atom_names(atom_names)	Add atom_names that do not exist.
append(system)	Append a system to this system.
apply_pbc()	Append periodic boundary condition.
as_dict()	Returns data dict of System instance.
check_data()	Check if data is correct.
check_type_map(type_map)	Assign atom_names to type_map if type_map is given and different from atom_names.
convert_to_mixed_type([type_map])	Convert the data dict to mixed type format structure, in order to append systems with different formula but the same number of atoms.
copy()	Returns a copy of the system.
dump(filename[, indent])	Dump .json or .yaml file.
extend(systems)	Extend a system list to this system.
from_3dmol(file_name, **kwargs)	Read data from *dpdata.plugins.3dmol.Py3DMolFormat* format.
from_abacus_lcao_md(file_name, **kwargs)	Read data from *dpdata.plugins.abacus.AbacusMDFormat* format.
from_abacus_lcao_relax(file_name, **kwargs)	Read data from *dpdata.plugins.abacus.AbacusRelaxFormat* format.
from_abacus_lcao_scf(file_name, **kwargs)	Read data from *dpdata.plugins.abacus.AbacusSCFFormat* format.
from_abacus_md(file_name, **kwargs)	Read data from *dpdata.plugins.abacus.AbacusMDFormat* format.
from_abacus_pw_md(file_name, **kwargs)	Read data from *dpdata.plugins.abacus.AbacusMDFormat* format.
from_abacus_pw_relax(file_name, **kwargs)	Read data from *dpdata.plugins.abacus.AbacusRelaxFormat* format.
from_abacus_pw_scf(file_name, **kwargs)	Read data from *dpdata.plugins.abacus.AbacusSCFFormat* format.
from_abacus_relax(file_name, **kwargs)	Read data from *dpdata.plugins.abacus.AbacusRelaxFormat* format.
from_abacus_scf(file_name, **kwargs)	Read data from *dpdata.plugins.abacus.AbacusSCFFormat* format.
from_abacus_stru(file_name, **kwargs)	Read data from *dpdata.plugins.abacus.AbacusSTRUFormat* format.
from_amber_md(file_name, **kwargs)	Read data from *dpdata.plugins.amber.AmberMDFormat* format.
from_ase_structure(file_name, **kwargs)	Read data from *dpdata.plugins.ase.ASEStructureFormat* format.
from_ase_traj(file_name, **kwargs)	Read data from *dpdata.plugins.ase.ASETrajFormat* format.
from_atomconfig(file_name, **kwargs)	Read data from *dpdata.plugins.pwmat.PwmatAtomconfigFormat* format.
from_contcar(file_name, **kwargs)	Read data from *dpdata.plugins.vasp.VASPPoscarFormat* format.

continues on next page

Table 8 – continued from previous page

from_cp2k_aimd_output(file_name, **kwargs)	Read data from *dpdata.plugins.cp2k.CP2KAIMDOutputFormat* format.
from_cp2k_output(file_name, **kwargs)	Read data from *dpdata.plugins.cp2k.CP2KOutputFormat* format.
from_deepmd(file_name, **kwargs)	Read data from *dpdata.plugins.deepmd.DeePMDRawFormat* format.
from_deepmd_comp(file_name, **kwargs)	Read data from *dpdata.plugins.deepmd.DeePMDCompFormat* format.
from_deepmd_hdf5(file_name, **kwargs)	Read data from *dpdata.plugins.deepmd.DeePMDHDF5Format* format.
from_deepmd_npy(file_name, **kwargs)	Read data from *dpdata.plugins.deepmd.DeePMDCompFormat* format.
from_deepmd_npy_mixed(file_name, **kwargs)	Read data from *dpdata.plugins.deepmd.DeePMDMixedFormat* format.
from_deepmd_raw(file_name, **kwargs)	Read data from *dpdata.plugins.deepmd.DeePMDRawFormat* format.
from_dftbplus(file_name, **kwargs)	Read data from *dpdata.plugins.dftbplus.DFTBplusFormat* format.
from_dict(d)	**param d** Dict representation.
from_dump(file_name, **kwargs)	Read data from *dpdata.plugins.lammps.LAMMPSDumpFormat* format.
from_fhi_aims_md(file_name, **kwargs)	Read data from *dpdata.plugins.fhi_aims.FhiMDFormat* format.
from_fhi_aims_output(file_name, **kwargs)	Read data from *dpdata.plugins.fhi_aims.FhiMDFormat* format.
from_fhi_aims_scf(file_name, **kwargs)	Read data from *dpdata.plugins.fhi_aims.FhiSCFFormat* format.
from_finalconfig(file_name, **kwargs)	Read data from *dpdata.plugins.pwmat.PwmatAtomconfigFormat* format.
from_gaussian_gjf(file_name, **kwargs)	Read data from *dpdata.plugins.gaussian.GaussiaGJFFormat* format.
from_gaussian_log(file_name, **kwargs)	Read data from *dpdata.plugins.gaussian.GaussianLogFormat* format.
from_gaussian_md(file_name, **kwargs)	Read data from *dpdata.plugins.gaussian.GaussianMDFormat* format.
from_gro(file_name, **kwargs)	Read data from *dpdata.plugins.gromacs.GromacsGroFormat* format.
from_gromacs_gro(file_name, **kwargs)	Read data from *dpdata.plugins.gromacs.GromacsGroFormat* format.
from_lammps_dump(file_name, **kwargs)	Read data from *dpdata.plugins.lammps.LAMMPSDumpFormat* format.
from_lammps_lmp(file_name, **kwargs)	Read data from *dpdata.plugins.lammps.LAMMPSLmpFormat* format.
from_list(file_name, **kwargs)	Read data from *dpdata.plugins.list.ListFormat* format.
from_lmp(file_name, **kwargs)	Read data from *dpdata.plugins.lammps.LAMMPSLmpFormat* format.
from_mlmd(file_name, **kwargs)	Read data from *dpdata.plugins.pwmat.PwmatOutputFormat* format.

continues on next page

Table 8 – continued from previous page

from_mol(file_name, **kwargs)	Read data from *dpdata.plugins.rdkit.MolFormat* format.
from_mol_file(file_name, **kwargs)	Read data from *dpdata.plugins.rdkit.MolFormat* format.
from_movement(file_name, **kwargs)	Read data from *dpdata.plugins.pwmat.PwmatOutputFormat* format.
from_n2p2(file_name, **kwargs)	Read data from *dpdata.plugins.n2p2.N2P2Format* format.
from_openmx_md(file_name, **kwargs)	Read data from *dpdata.plugins.openmx.OPENMXFormat* format.
from_orca_spout(file_name, **kwargs)	Read data from *dpdata.plugins.orca.ORCASPOutFormat* format.
from_outcar(file_name, **kwargs)	Read data from *dpdata.plugins.vasp.VASPOutcarFormat* format.
from_poscar(file_name, **kwargs)	Read data from *dpdata.plugins.vasp.VASPPoscarFormat* format.
from_psi4_inp(file_name, **kwargs)	Read data from *dpdata.plugins.psi4.PSI4InputFormat* format.
from_psi4_out(file_name, **kwargs)	Read data from *dpdata.plugins.psi4.PSI4OutFormat* format.
from_pwmat_atomconfig(file_name, **kwargs)	Read data from *dpdata.plugins.pwmat.PwmatAtomconfigFormat* format.
from_pwmat_finalconfig(file_name, **kwargs)	Read data from *dpdata.plugins.pwmat.PwmatAtomconfigFormat* format.
from_pwmat_mlmd(file_name, **kwargs)	Read data from *dpdata.plugins.pwmat.PwmatOutputFormat* format.
from_pwmat_movement(file_name, **kwargs)	Read data from *dpdata.plugins.pwmat.PwmatOutputFormat* format.
from_pwmat_output(file_name, **kwargs)	Read data from *dpdata.plugins.pwmat.PwmatOutputFormat* format.
from_pymatgen_computedstructureentry(...)	Read data from *dpdata.plugins.pymatgen.PyMatgenCSEFormat* format.
from_pymatgen_molecule(file_name, **kwargs)	Read data from *dpdata.plugins.pymatgen.PyMatgenMoleculeFormat* format.
from_pymatgen_structure(file_name, **kwargs)	Read data from *dpdata.plugins.pymatgen.PyMatgenStructureFormat* format.
from_qe_cp_traj(file_name, **kwargs)	Read data from *dpdata.plugins.qe.QECPTrajFormat* format.
from_qe_pw_scf(file_name, **kwargs)	Read data from *dpdata.plugins.qe.QECPPWSCFFormat* format.
from_quip_gap_xyz(file_name, **kwargs)	Read data from *dpdata.plugins.xyz.QuipGapXYZFormat* format.
from_quip_gap_xyz_file(file_name, **kwargs)	Read data from *dpdata.plugins.xyz.QuipGapXYZFormat* format.
from_sdf(file_name, **kwargs)	Read data from *dpdata.plugins.rdkit.SdfFormat* format.
from_sdf_file(file_name, **kwargs)	Read data from *dpdata.plugins.rdkit.SdfFormat* format.
from_siesta_aiMD_output(file_name, **kwargs)	Read data from *dpdata.plugins.siesta.SiestaAIMDOutputFormat* format.
from_siesta_aimd_output(file_name, **kwargs)	Read data from *dpdata.plugins.siesta.SiestaAIMDOutputFormat* format.

continues on next page

Table 8 – continued from previous page

from_siesta_output(file_name, **kwargs)	Read data from *dpdata.plugins.siesta.SiestaOutputFormat* format.
from_sqm_in(file_name, **kwargs)	Read data from *dpdata.plugins.amber.SQMINFormat* format.
from_sqm_out(file_name, **kwargs)	Read data from *dpdata.plugins.amber.SQMOutFormat* format.
from_stru(file_name, **kwargs)	Read data from *dpdata.plugins.abacus.AbacusSTRUFormat* format.
from_vasp_contcar(file_name, **kwargs)	Read data from *dpdata.plugins.vasp.VASPPoscarFormat* format.
from_vasp_outcar(file_name, **kwargs)	Read data from *dpdata.plugins.vasp.VASPOutcarFormat* format.
from_vasp_poscar(file_name, **kwargs)	Read data from *dpdata.plugins.vasp.VASPPoscarFormat* format.
from_vasp_string(file_name, **kwargs)	Read data from *dpdata.plugins.vasp.VASPStringFormat* format.
from_vasp_xml(file_name, **kwargs)	Read data from *dpdata.plugins.vasp.VASPXMLFormat* format.
from_xml(file_name, **kwargs)	Read data from *dpdata.plugins.vasp.VASPXMLFormat* format.
from_xyz(file_name, **kwargs)	Read data from *dpdata.plugins.xyz.XYZFormat* format.
get_atom_names()	Returns name of atoms.
get_atom_numbs()	Returns number of atoms.
get_atom_types()	Returns type of atoms.
get_natoms()	Returns total number of atoms in the system.
get_nframes()	Returns number of frames in the system.
get_ntypes()	Returns total number of atom types in the system.
load(filename)	Rebuild System obj.
map_atom_types([type_map])	Map the atom types of the system.
minimize(*args, minimizer, **kwargs)	Minimize the geometry.
perturb(pert_num, cell_pert_fraction, ...[, ...])	Perturb each frame in the system randomly.
pick_atom_idx(idx[, nopbc])	Pick atom index.
pick_by_amber_mask(param, maskstr[, ...])	Pick atoms by amber mask.
predict(*args[, driver])	Predict energies and forces by a driver.
register_data_type(*data_type)	Register data type.
remove_atom_names(atom_names)	Remove atom names and all such atoms.
remove_pbc([protect_layer])	This method does NOT delete the definition of the cells, it (1) revises the cell to a cubic cell and ensures that the cell boundary to any atom in the system is no less than *protect_layer* (2) translates the system such that the center-of-geometry of the system locates at the center of the cell.
replicate(ncopy)	Replicate the each frame in the system in 3 dimensions.
shuffle()	Shuffle frames randomly.
sort_atom_names([type_map])	Sort atom_names of the system and reorder atom_numbs and atom_types accoarding to atom_names.
sort_atom_types()	Sort atom types.
sub_system(f_idx)	Construct a subsystem from the system.
to(fmt, *args, **kwargs)	Dump systems to the specific format.

continues on next page

Table 8 – continued from previous page

to_3dmol(*args, **kwargs)	Dump data to *dpdata.plugins.3dmol.Py3DMolFormat* format.
to_abacus_lcao_md(*args, **kwargs)	Dump data to *dpdata.plugins.abacus.AbacusMDFormat* format.
to_abacus_lcao_relax(*args, **kwargs)	Dump data to *dpdata.plugins.abacus.AbacusRelaxFormat* format.
to_abacus_lcao_scf(*args, **kwargs)	Dump data to *dpdata.plugins.abacus.AbacusSCFFormat* format.
to_abacus_md(*args, **kwargs)	Dump data to *dpdata.plugins.abacus.AbacusMDFormat* format.
to_abacus_pw_md(*args, **kwargs)	Dump data to *dpdata.plugins.abacus.AbacusMDFormat* format.
to_abacus_pw_relax(*args, **kwargs)	Dump data to *dpdata.plugins.abacus.AbacusRelaxFormat* format.
to_abacus_pw_scf(*args, **kwargs)	Dump data to *dpdata.plugins.abacus.AbacusSCFFormat* format.
to_abacus_relax(*args, **kwargs)	Dump data to *dpdata.plugins.abacus.AbacusRelaxFormat* format.
to_abacus_scf(*args, **kwargs)	Dump data to *dpdata.plugins.abacus.AbacusSCFFormat* format.
to_abacus_stru(*args, **kwargs)	Dump data to *dpdata.plugins.abacus.AbacusSTRUFormat* format.
to_amber_md(*args, **kwargs)	Dump data to *dpdata.plugins.amber.AmberMDFormat* format.
to_ase_structure(*args, **kwargs)	Dump data to *dpdata.plugins.ase.ASEStructureFormat* format.
to_ase_traj(*args, **kwargs)	Dump data to *dpdata.plugins.ase.ASETrajFormat* format.
to_atomconfig(*args, **kwargs)	Dump data to *dpdata.plugins.pwmat.PwmatAtomconfigFormat* format.
to_contcar(*args, **kwargs)	Dump data to *dpdata.plugins.vasp.VASPPoscarFormat* format.
to_cp2k_aimd_output(*args, **kwargs)	Dump data to *dpdata.plugins.cp2k.CP2KAIMDOutputFormat* format.
to_cp2k_output(*args, **kwargs)	Dump data to *dpdata.plugins.cp2k.CP2KOutputFormat* format.
to_deepmd(*args, **kwargs)	Dump data to *dpdata.plugins.deepmd.DeePMDRawFormat* format.
to_deepmd_comp(*args, **kwargs)	Dump data to *dpdata.plugins.deepmd.DeePMDCompFormat* format.
to_deepmd_hdf5(*args, **kwargs)	Dump data to *dpdata.plugins.deepmd.DeePMDHDF5Format* format.
to_deepmd_npy(*args, **kwargs)	Dump data to *dpdata.plugins.deepmd.DeePMDCompFormat* format.
to_deepmd_npy_mixed(*args, **kwargs)	Dump data to *dpdata.plugins.deepmd.DeePMDMixedFormat* format.
to_deepmd_raw(*args, **kwargs)	Dump data to *dpdata.plugins.deepmd.DeePMDRawFormat* format.
to_dftbplus(*args, **kwargs)	Dump data to *dpdata.plugins.dftbplus.DFTBplusFormat* format.
to_dump(*args, **kwargs)	Dump data to *dpdata.plugins.lammps.LAMMPSDumpFormat* format.

continues on next page

Table 8 – continued from previous page

to_fhi_aims_md(*args, **kwargs)	Dump data to *dpdata.plugins.fhi_aims.FhiMDFormat* format.
to_fhi_aims_output(*args, **kwargs)	Dump data to *dpdata.plugins.fhi_aims.FhiMDFormat* format.
to_fhi_aims_scf(*args, **kwargs)	Dump data to *dpdata.plugins.fhi_aims.FhiSCFFormat* format.
to_finalconfig(*args, **kwargs)	Dump data to *dpdata.plugins.pwmat.PwmatAtomconfigFormat* format.
to_gaussian_gjf(*args, **kwargs)	Dump data to *dpdata.plugins.gaussian.GaussiaGJFFormat* format.
to_gaussian_log(*args, **kwargs)	Dump data to *dpdata.plugins.gaussian.GaussianLogFormat* format.
to_gaussian_md(*args, **kwargs)	Dump data to *dpdata.plugins.gaussian.GaussianMDFormat* format.
to_gro(*args, **kwargs)	Dump data to *dpdata.plugins.gromacs.GromacsGroFormat* format.
to_gromacs_gro(*args, **kwargs)	Dump data to *dpdata.plugins.gromacs.GromacsGroFormat* format.
to_json()	Returns a json string representation of the MSONable object.
to_lammps_dump(*args, **kwargs)	Dump data to *dpdata.plugins.lammps.LAMMPSDumpFormat* format.
to_lammps_lmp(*args, **kwargs)	Dump data to *dpdata.plugins.lammps.LAMMPSLmpFormat* format.
to_list(*args, **kwargs)	Dump data to *dpdata.plugins.list.ListFormat* format.
to_lmp(*args, **kwargs)	Dump data to *dpdata.plugins.lammps.LAMMPSLmpFormat* format.
to_mlmd(*args, **kwargs)	Dump data to *dpdata.plugins.pwmat.PwmatOutputFormat* format.
to_mol(*args, **kwargs)	Dump data to *dpdata.plugins.rdkit.MolFormat* format.
to_mol_file(*args, **kwargs)	Dump data to *dpdata.plugins.rdkit.MolFormat* format.
to_movement(*args, **kwargs)	Dump data to *dpdata.plugins.pwmat.PwmatOutputFormat* format.
to_n2p2(*args, **kwargs)	Dump data to *dpdata.plugins.n2p2.N2P2Format* format.
to_openmx_md(*args, **kwargs)	Dump data to *dpdata.plugins.openmx.OPENMXFormat* format.
to_orca_spout(*args, **kwargs)	Dump data to *dpdata.plugins.orca.ORCASPOutFormat* format.
to_outcar(*args, **kwargs)	Dump data to *dpdata.plugins.vasp.VASPOutcarFormat* format.
to_poscar(*args, **kwargs)	Dump data to *dpdata.plugins.vasp.VASPPoscarFormat* format.
to_psi4_inp(*args, **kwargs)	Dump data to *dpdata.plugins.psi4.PSI4InputFormat* format.
to_psi4_out(*args, **kwargs)	Dump data to *dpdata.plugins.psi4.PSI4OutFormat* format.
to_pwmat_atomconfig(*args, **kwargs)	Dump data to *dpdata.plugins.pwmat.PwmatAtomconfigFormat* format.

continues on next page

Table 8 – continued from previous page

to_pwmat_finalconfig(*args, **kwargs)	Dump data to dpdata.plugins.pwmat.PwmatAtomconfigFormat format.
to_pwmat_mlmd(*args, **kwargs)	Dump data to dpdata.plugins.pwmat.PwmatOutputFormat format.
to_pwmat_movement(*args, **kwargs)	Dump data to dpdata.plugins.pwmat.PwmatOutputFormat format.
to_pwmat_output(*args, **kwargs)	Dump data to dpdata.plugins.pwmat.PwmatOutputFormat format.
to_pymatgen_ComputedStructureEntry(*args, ...)	Dump data to dpdata.plugins.pymatgen.PyMatgenCSEFormat format.
to_pymatgen_computedstructureentry(*args, ...)	Dump data to dpdata.plugins.pymatgen.PyMatgenCSEFormat format.
to_pymatgen_molecule(*args, **kwargs)	Dump data to dpdata.plugins.pymatgen.PyMatgenMoleculeFormat format.
to_pymatgen_structure(*args, **kwargs)	Dump data to dpdata.plugins.pymatgen.PyMatgenStructureFormat format.
to_qe_cp_traj(*args, **kwargs)	Dump data to dpdata.plugins.qe.QECPTrajFormat format.
to_qe_pw_scf(*args, **kwargs)	Dump data to dpdata.plugins.qe.QECPPWSCFFormat format.
to_quip_gap_xyz(*args, **kwargs)	Dump data to dpdata.plugins.xyz.QuipGapXYZFormat format.
to_quip_gap_xyz_file(*args, **kwargs)	Dump data to dpdata.plugins.xyz.QuipGapXYZFormat format.
to_sdf(*args, **kwargs)	Dump data to dpdata.plugins.rdkit.SdfFormat format.
to_sdf_file(*args, **kwargs)	Dump data to dpdata.plugins.rdkit.SdfFormat format.
to_siesta_aimd_output(*args, **kwargs)	Dump data to dpdata.plugins.siesta.SiestaAIMDOutputFormat format.
to_siesta_output(*args, **kwargs)	Dump data to dpdata.plugins.siesta.SiestaOutputFormat format.
to_sqm_in(*args, **kwargs)	Dump data to dpdata.plugins.amber.SQMINFormat format.
to_sqm_out(*args, **kwargs)	Dump data to dpdata.plugins.amber.SQMOutFormat format.
to_stru(*args, **kwargs)	Dump data to dpdata.plugins.abacus.AbacusSTRUFormat format.
to_vasp_contcar(*args, **kwargs)	Dump data to dpdata.plugins.vasp.VASPPoscarFormat format.
to_vasp_outcar(*args, **kwargs)	Dump data to dpdata.plugins.vasp.VASPOutcarFormat format.
to_vasp_poscar(*args, **kwargs)	Dump data to dpdata.plugins.vasp.VASPPoscarFormat format.
to_vasp_string(*args, **kwargs)	Dump data to dpdata.plugins.vasp.VASPStringFormat format.
to_vasp_xml(*args, **kwargs)	Dump data to dpdata.plugins.vasp.VASPXMLFormat format.
to_xml(*args, **kwargs)	Dump data to dpdata.plugins.vasp.VASPXMLFormat format.
to_xyz(*args, **kwargs)	Dump data to dpdata.plugins.xyz.XYZFormat format.

continues on next page

Table 8 – continued from previous page

unsafe_hash()	Returns an hash of the current object.
validate_monty_v1(_MSONable__input_value)	Pydantic validator with correct signature for pydantic v1.x
validate_monty_v2(_MSONable__input_value, _)	Pydantic validator with correct signature for pydantic v2.x

affine_map
apply_type_map
from_fmt
from_fmt_obj
replace
rot_frame_lower_triangular
rot_lower_triangular
to_fmt_obj

DTYPES = (<dpdata.data_type.DataType object>, <dpdata.data_type.DataType object>, <dpdata.data_type.DataType object>, <dpdata.data_type.DataType object>, <dpdata.data_type.DataType object>, <dpdata.data_type.DataType object>, <dpdata.data_type.DataType object>, <dpdata.data_type.DataType object>, <dpdata.data_type.DataType object>)

add_atom_names(*atom_names*)

Add atom_names that do not exist.

affine_map(*trans*, *f_idx=0*)

append(*system*)

Append a system to this system.

Parameters

system
[System] The system to append

apply_pbc()

Append periodic boundary condition.

apply_type_map(*type_map*)

as_dict()

Returns data dict of System instance.

check_data()

Check if data is correct.

Raises

DataError
if data is not correct

check_type_map(*type_map*)

Assign atom_names to type_map if type_map is given and different from atom_names.

Parameters

type_map
[list] type_map

convert_to_mixed_type(*type_map=None*)

Convert the data dict to mixed type format structure, in order to append systems with different formula but the same number of atoms. Change the 'atom_names' to one placeholder type 'MIXED_TOKEN' and add 'real_atom_types' to store the real type vectors according to the given type_map.

Parameters

type_map
[list] type_map

copy()

Returns a copy of the system.

dump(*filename*, *indent=4*)

Dump .json or .yaml file.

extend(*systems*)

Extend a system list to this system.

Parameters

systems
[[System1, System2, System3]] The list to extend

property formula

Return the formula of this system, like C3H5O2.

property formula_hash: str

Return the hash of the formula of this system.

from_3dmol(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.3dmol.Py3DMolFormat* format.

from_abacus_lcao_md(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.abacus.AbacusMDFormat* format.

from_abacus_lcao_relax(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.abacus.AbacusRelaxFormat* format.

from_abacus_lcao_scf(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.abacus.AbacusSCFFormat* format.

from_abacus_md(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.abacus.AbacusMDFormat* format.

from_abacus_pw_md(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.abacus.AbacusMDFormat* format.

from_abacus_pw_relax(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.abacus.AbacusRelaxFormat* format.

from_abacus_pw_scf(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.abacus.AbacusSCFFormat* format.

from_abacus_relax(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.abacus.AbacusRelaxFormat* format.

from_abacus_scf(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.abacus.AbacusSCFFormat* format.

from_abacus_stru(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.abacus.AbacusSTRUFormat* format.

from_amber_md(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.amber.AmberMDFormat* format.

from_ase_structure(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.ase.ASEStructureFormat* format.

from_ase_traj(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.ase.ASETrajFormat* format.

from_atomconfig(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.pwmat.PwmatAtomconfigFormat* format.

from_contcar(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.vasp.VASPPoscarFormat* format.

from_cp2k_aimd_output(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.cp2k.CP2KAIMDOutputFormat* format.

from_cp2k_output(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.cp2k.CP2KOutputFormat* format.

from_deepmd(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.deepmd.DeePMDRawFormat* format.

from_deepmd_comp(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.deepmd.DeePMDCompFormat* format.

from_deepmd_hdf5(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.deepmd.DeePMDHDF5Format* format.

from_deepmd_npy(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.deepmd.DeePMDCompFormat* format.

from_deepmd_npy_mixed(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.deepmd.DeePMDMixedFormat* format.

from_deepmd_raw(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.deepmd.DeePMDRawFormat* format.

from_dftbplus(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.dftbplus.DFTBplusFormat* format.

from_dump(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.lammps.LAMMPSDumpFormat* format.

from_fhi_aims_md(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.fhi_aims.FhiMDFormat* format.

from_fhi_aims_output(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.fhi_aims.FhiMDFormat* format.

from_fhi_aims_scf(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.fhi_aims.FhiSCFFormat* format.

from_finalconfig(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.pwmat.PwmatAtomconfigFormat* format.

from_fmt(*file_name*, *fmt='auto'*, ***kwargs*)

from_fmt_obj(*fmtobj*, *file_name*, ***kwargs*)

from_gaussian_gjf(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.gaussian.GaussiaGJFFormat* format.

from_gaussian_log(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.gaussian.GaussianLogFormat* format.

from_gaussian_md(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.gaussian.GaussianMDFormat* format.

from_gro(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.gromacs.GromacsGroFormat* format.

from_gromacs_gro(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.gromacs.GromacsGroFormat* format.

from_lammps_dump(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.lammps.LAMMPSDumpFormat* format.

from_lammps_lmp(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.lammps.LAMMPSLmpFormat* format.

from_list(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.list.ListFormat* format.

from_lmp(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.lammps.LAMMPSLmpFormat* format.

from_mlmd(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.pwmat.PwmatOutputFormat* format.

from_mol(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.rdkit.MolFormat* format.

from_mol_file(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.rdkit.MolFormat* format.

from_movement(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.pwmat.PwmatOutputFormat* format.

from_n2p2(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.n2p2.N2P2Format* format.

from_openmx_md(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.openmx.OPENMXFormat* format.

from_orca_spout(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.orca.ORCASPOutFormat* format.

from_outcar(*file_name*, ***kwargs*)
Read data from *dpdata.plugins.vasp.VASPOutcarFormat* format.

from_poscar(*file_name*, ***kwargs*)
Read data from *dpdata.plugins.vasp.VASPPoscarFormat* format.

from_psi4_inp(*file_name*, ***kwargs*)
Read data from *dpdata.plugins.psi4.PSI4InputFormat* format.

from_psi4_out(*file_name*, ***kwargs*)
Read data from *dpdata.plugins.psi4.PSI4OutFormat* format.

from_pwmat_atomconfig(*file_name*, ***kwargs*)
Read data from *dpdata.plugins.pwmat.PwmatAtomconfigFormat* format.

from_pwmat_finalconfig(*file_name*, ***kwargs*)
Read data from *dpdata.plugins.pwmat.PwmatAtomconfigFormat* format.

from_pwmat_mlmd(*file_name*, ***kwargs*)
Read data from *dpdata.plugins.pwmat.PwmatOutputFormat* format.

from_pwmat_movement(*file_name*, ***kwargs*)
Read data from *dpdata.plugins.pwmat.PwmatOutputFormat* format.

from_pwmat_output(*file_name*, ***kwargs*)
Read data from *dpdata.plugins.pwmat.PwmatOutputFormat* format.

from_pymatgen_computedstructureentry(*file_name*, ***kwargs*)
Read data from *dpdata.plugins.pymatgen.PyMatgenCSEFormat* format.

from_pymatgen_molecule(*file_name*, ***kwargs*)
Read data from *dpdata.plugins.pymatgen.PyMatgenMoleculeFormat* format.

from_pymatgen_structure(*file_name*, ***kwargs*)
Read data from *dpdata.plugins.pymatgen.PyMatgenStructureFormat* format.

from_qe_cp_traj(*file_name*, ***kwargs*)
Read data from *dpdata.plugins.qe.QECPTrajFormat* format.

from_qe_pw_scf(*file_name*, ***kwargs*)
Read data from *dpdata.plugins.qe.QECPPWSCFFormat* format.

from_quip_gap_xyz(*file_name*, ***kwargs*)
Read data from *dpdata.plugins.xyz.QuipGapXYZFormat* format.

from_quip_gap_xyz_file(*file_name*, ***kwargs*)
Read data from *dpdata.plugins.xyz.QuipGapXYZFormat* format.

from_sdf(*file_name*, ***kwargs*)
Read data from *dpdata.plugins.rdkit.SdfFormat* format.

from_sdf_file(*file_name*, ***kwargs*)
Read data from *dpdata.plugins.rdkit.SdfFormat* format.

from_siesta_aiMD_output(*file_name*, ***kwargs*)
Read data from *dpdata.plugins.siesta.SiestaAIMDOutputFormat* format.

from_siesta_aimd_output(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.siesta.SiestaAIMDOutputFormat* format.

from_siesta_output(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.siesta.SiestaOutputFormat* format.

from_sqm_in(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.amber.SQMINFormat* format.

from_sqm_out(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.amber.SQMOutFormat* format.

from_stru(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.abacus.AbacusSTRUFormat* format.

from_vasp_contcar(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.vasp.VASPPoscarFormat* format.

from_vasp_outcar(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.vasp.VASPOutcarFormat* format.

from_vasp_poscar(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.vasp.VASPPoscarFormat* format.

from_vasp_string(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.vasp.VASPStringFormat* format.

from_vasp_xml(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.vasp.VASPXMLFormat* format.

from_xml(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.vasp.VASPXMLFormat* format.

from_xyz(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.xyz.XYZFormat* format.

get_atom_names()

Returns name of atoms.

get_atom_numbs()

Returns number of atoms.

get_atom_types()

Returns type of atoms.

get_natoms()

Returns total number of atoms in the system.

get_nframes()

Returns number of frames in the system.

get_ntypes() → int

Returns total number of atom types in the system.

static load(*filename*)

Rebuild System obj. from .json or .yaml file.

map_atom_types(*type_map=None*) → ndarray

Map the atom types of the system.

Parameters

type_map
dict : {"H":0,"O":1} or list ["H","C","O","N"] The map between elements and index if no map_dict is given, index will be set according to atomic number

Returns

new_atom_types
[np.ndarray] The mapped atom types

minimize(**args: Any*, *minimizer: str | Minimizer*, ***kwargs: Any*) → *LabeledSystem*

Minimize the geometry.

Parameters

***args**
[iterable] Arguments passing to the minimizer

minimizer
[str or Minimizer] The assigned minimizer

****kwargs**
[dict] Other arguments passing to the minimizer

Returns

labeled_sys
[LabeledSystem] A new labeled system.

property nopbc

perturb(*pert_num*, *cell_pert_fraction*, *atom_pert_distance*, *atom_pert_style='normal'*)

Perturb each frame in the system randomly. The cell will be deformed randomly, and atoms will be displaced by a random distance in random direction.

Parameters

pert_num
[int] Each frame in the system will make *pert_num* copies, and all the copies will be perturbed. That means the system to be returned will contain *pert_num* * frame_num of the input system.

cell_pert_fraction
[float] A fraction determines how much (relatively) will cell deform. The cell of each frame is deformed by a symmetric matrix perturbed from identity. The perturbation to the diagonal part is subject to a uniform distribution in [-cell_pert_fraction, cell_pert_fraction), and the perturbation to the off-diagonal part is subject to a uniform distribution in [-0.5*cell_pert_fraction, 0.5*cell_pert_fraction).

atom_pert_distance
[float] unit: Angstrom. A distance determines how far atoms will move. Atoms will move about *atom_pert_distance* in random direction. The distribution of the distance atoms move is determined by atom_pert_style

atom_pert_style
[str] Determines the distribution of the distance atoms move is subject to. Avaliable options are

- ***'normal'*: the *distance* will be object to *chi-square distribution with 3 degrees of freedom* after normalization.**
  The mean value of the distance is *atom_pert_fraction*side_length*
- ***'uniform'*: will generate uniformly random points in a 3D-balls with radius as *atom_pert_distance*.**
  These points are treated as vector used by atoms to move. Obviously, the max length of the distance atoms move is *atom_pert_distance*.
- *'const'*: The distance atoms move will be a constant *atom_pert_distance*.

Returns

perturbed_system
[System] The perturbed_system. It contains *pert_num* * frame_num of the input system frames.

`pick_atom_idx`(*idx*, *nopbc=None*)

Pick atom index.

Parameters

idx
[int or list or slice] atom index

nopbc
[Boolen (default: None)] If nopbc is True or False, set nopbc

Returns

new_sys: System
new system

`pick_by_amber_mask`(*param*, *maskstr*, *pass_coords=False*, *nopbc=None*)

Pick atoms by amber mask.

Parameters

param
[str or parmed.Structure] filename of Amber param file or parmed.Structure

maskstr
[str] Amber masks

pass_coords
[Boolen (default: False)] If pass_coords is true, the function will pass coordinates and return a MultiSystem. Otherwise, the result is coordinate-independent, and the function will return System or LabeledSystem.

nopbc
[Boolen (default: None)] If nopbc is True or False, set nopbc

`post_funcs = <dpdata.plugin.Plugin object>`

`predict`(**args: Any*, *driver: str = 'dp'*, ***kwargs: Any*) → *LabeledSystem*

Predict energies and forces by a driver.

Parameters

***args**
[iterable] Arguments passing to the driver

driver
[str, default=dp] The assigned driver. For compatibility, default is dp

****kwargs**
[dict] Other arguments passing to the driver

Returns

labeled_sys
[LabeledSystem] A new labeled system.

Examples

The default driver is DP:

```
>>> labeled_sys = ori_sys.predict("frozen_model_compressed.pb")
```

classmethod register_data_type(**data_type: Tuple[DataType]*)

Register data type.

Parameters

***data_type**
[tuple[DataType]] data type to be regiestered

remove_atom_names(*atom_names*)

Remove atom names and all such atoms. For example, you may not remove EP atoms in TIP4P/Ew water, which is not a real atom.

remove_pbc(*protect_layer=9*)

This method does NOT delete the definition of the cells, it (1) revises the cell to a cubic cell and ensures that the cell boundary to any atom in the system is no less than *protect_layer* (2) translates the system such that the center-of-geometry of the system locates at the center of the cell.

Parameters

protect_layer
[the protect layer between the atoms and the cell] boundary

replace(*initial_atom_type*, *end_atom_type*, *replace_num*)

replicate(*ncopy*)

Replicate the each frame in the system in 3 dimensions. Each frame in the system will become a supercell.

Parameters

ncopy
list: [4,2,3] or tuple: (4,2,3,) make *ncopy[0]* copys in x dimensions, make *ncopy[1]* copys in y dimensions, make *ncopy[2]* copys in z dimensions.

Returns

tmp
[System] The system after replication.

rot_frame_lower_triangular(*f_idx=0*)

rot_lower_triangular()

property short_formula: str

Return the short formula of this system. Elements with zero number will be removed.

property short_name: str

Return the short name of this system (no more than 255 bytes), in the following order:

- formula
- short_formula
- formula_hash.

shuffle()

Shuffle frames randomly.

sort_atom_names(*type_map=None*)

Sort atom_names of the system and reorder atom_numbs and atom_types accoarding to atom_names. If type_map is not given, atom_names will be sorted by alphabetical order. If type_map is given, atom_names will be type_map.

Parameters

type_map
[list] type_map

sort_atom_types() → ndarray

Sort atom types.

Returns

idx
[np.ndarray] new atom index in the Axis.NATOMS

sub_system(*f_idx*)

Construct a subsystem from the system.

Parameters

f_idx
[int or index] Which frame to use in the subsystem

Returns

sub_system
[System] The subsystem

to(*fmt: str*, **args*, ***kwargs*) → *System*

Dump systems to the specific format.

Parameters

fmt
[str] format

***args**
arguments

****kwargs**
keyword arguments

Returns

System
self

to_3dmol(**args*, ***kwargs*)

Dump data to *dpdata.plugins.3dmol.Py3DMolFormat* format.

to_abacus_lcao_md(**args*, ***kwargs*)

Dump data to *dpdata.plugins.abacus.AbacusMDFormat* format.

to_abacus_lcao_relax(**args*, ***kwargs*)

Dump data to *dpdata.plugins.abacus.AbacusRelaxFormat* format.

to_abacus_lcao_scf(**args*, ***kwargs*)

Dump data to *dpdata.plugins.abacus.AbacusSCFFormat* format.

to_abacus_md(**args*, ***kwargs*)

Dump data to *dpdata.plugins.abacus.AbacusMDFormat* format.

to_abacus_pw_md(**args*, ***kwargs*)

Dump data to *dpdata.plugins.abacus.AbacusMDFormat* format.

to_abacus_pw_relax(**args*, ***kwargs*)

Dump data to *dpdata.plugins.abacus.AbacusRelaxFormat* format.

to_abacus_pw_scf(**args*, ***kwargs*)

Dump data to *dpdata.plugins.abacus.AbacusSCFFormat* format.

to_abacus_relax(**args*, ***kwargs*)

Dump data to *dpdata.plugins.abacus.AbacusRelaxFormat* format.

to_abacus_scf(**args*, ***kwargs*)

Dump data to *dpdata.plugins.abacus.AbacusSCFFormat* format.

to_abacus_stru(**args*, ***kwargs*)

Dump data to *dpdata.plugins.abacus.AbacusSTRUFormat* format.

to_amber_md(**args*, ***kwargs*)

Dump data to *dpdata.plugins.amber.AmberMDFormat* format.

to_ase_structure(**args*, ***kwargs*)

Dump data to *dpdata.plugins.ase.ASEStructureFormat* format.

to_ase_traj(**args*, ***kwargs*)

Dump data to *dpdata.plugins.ase.ASETrajFormat* format.

to_atomconfig(**args*, ***kwargs*)

Dump data to *dpdata.plugins.pwmat.PwmatAtomconfigFormat* format.

to_contcar(**args*, ***kwargs*)

Dump data to *dpdata.plugins.vasp.VASPPoscarFormat* format.

to_cp2k_aimd_output(**args*, ***kwargs*)

Dump data to *dpdata.plugins.cp2k.CP2KAIMDOutputFormat* format.

to_cp2k_output(**args*, ***kwargs*)

Dump data to *dpdata.plugins.cp2k.CP2KOutputFormat* format.

to_deepmd(**args*, ***kwargs*)

Dump data to *dpdata.plugins.deepmd.DeePMDRawFormat* format.

to_deepmd_comp(**args*, ***kwargs*)

Dump data to *dpdata.plugins.deepmd.DeePMDCompFormat* format.

to_deepmd_hdf5(**args*, ***kwargs*)

Dump data to *dpdata.plugins.deepmd.DeePMDHDF5Format* format.

to_deepmd_npy(**args*, ***kwargs*)

Dump data to *dpdata.plugins.deepmd.DeePMDCompFormat* format.

to_deepmd_npy_mixed(**args*, ***kwargs*)

Dump data to *dpdata.plugins.deepmd.DeePMDMixedFormat* format.

to_deepmd_raw(**args*, ***kwargs*)

Dump data to *dpdata.plugins.deepmd.DeePMDRawFormat* format.

to_dftbplus(**args*, ***kwargs*)

Dump data to *dpdata.plugins.dftbplus.DFTBplusFormat* format.

to_dump(**args*, ***kwargs*)

Dump data to *dpdata.plugins.lammps.LAMMPSDumpFormat* format.

to_fhi_aims_md(**args*, ***kwargs*)

Dump data to *dpdata.plugins.fhi_aims.FhiMDFormat* format.

to_fhi_aims_output(**args*, ***kwargs*)

Dump data to *dpdata.plugins.fhi_aims.FhiMDFormat* format.

to_fhi_aims_scf(**args*, ***kwargs*)

Dump data to *dpdata.plugins.fhi_aims.FhiSCFFormat* format.

to_finalconfig(**args*, ***kwargs*)

Dump data to *dpdata.plugins.pwmat.PwmatAtomconfigFormat* format.

to_fmt_obj(*fmtobj*, **args*, ***kwargs*)

to_gaussian_gjf(**args*, ***kwargs*)

Dump data to *dpdata.plugins.gaussian.GaussiaGJFFormat* format.

to_gaussian_log(**args*, ***kwargs*)

Dump data to *dpdata.plugins.gaussian.GaussianLogFormat* format.

to_gaussian_md(**args*, ***kwargs*)

Dump data to *dpdata.plugins.gaussian.GaussianMDFormat* format.

to_gro(**args*, ***kwargs*)

Dump data to *dpdata.plugins.gromacs.GromacsGroFormat* format.

to_gromacs_gro(**args*, ***kwargs*)

Dump data to *dpdata.plugins.gromacs.GromacsGroFormat* format.

to_lammps_dump(**args*, ***kwargs*)

Dump data to *dpdata.plugins.lammps.LAMMPSDumpFormat* format.

to_lammps_lmp(**args*, ***kwargs*)

Dump data to *dpdata.plugins.lammps.LAMMPSLmpFormat* format.

to_list(**args*, ***kwargs*)

Dump data to *dpdata.plugins.list.ListFormat* format.

to_lmp(**args*, ***kwargs*)

Dump data to *dpdata.plugins.lammps.LAMMPSLmpFormat* format.

to_mlmd(**args*, ***kwargs*)

Dump data to *dpdata.plugins.pwmat.PwmatOutputFormat* format.

to_mol(**args*, ***kwargs*)

Dump data to *dpdata.plugins.rdkit.MolFormat* format.

to_mol_file(**args*, ***kwargs*)

Dump data to *dpdata.plugins.rdkit.MolFormat* format.

to_movement(**args*, ***kwargs*)

Dump data to *dpdata.plugins.pwmat.PwmatOutputFormat* format.

to_n2p2(**args*, ***kwargs*)

Dump data to *dpdata.plugins.n2p2.N2P2Format* format.

to_openmx_md(**args*, ***kwargs*)

Dump data to *dpdata.plugins.openmx.OPENMXFormat* format.

to_orca_spout(**args*, ***kwargs*)

Dump data to *dpdata.plugins.orca.ORCASPOutFormat* format.

to_outcar(**args*, ***kwargs*)

Dump data to *dpdata.plugins.vasp.VASPOutcarFormat* format.

to_poscar(**args*, ***kwargs*)

Dump data to *dpdata.plugins.vasp.VASPPoscarFormat* format.

to_psi4_inp(**args*, ***kwargs*)

Dump data to *dpdata.plugins.psi4.PSI4InputFormat* format.

to_psi4_out(**args*, ***kwargs*)

Dump data to *dpdata.plugins.psi4.PSI4OutFormat* format.

to_pwmat_atomconfig(**args*, ***kwargs*)

Dump data to *dpdata.plugins.pwmat.PwmatAtomconfigFormat* format.

to_pwmat_finalconfig(**args*, ***kwargs*)

Dump data to *dpdata.plugins.pwmat.PwmatAtomconfigFormat* format.

to_pwmat_mlmd(**args*, ***kwargs*)

Dump data to *dpdata.plugins.pwmat.PwmatOutputFormat* format.

to_pwmat_movement(**args*, ***kwargs*)

Dump data to *dpdata.plugins.pwmat.PwmatOutputFormat* format.

to_pwmat_output(**args*, ***kwargs*)

Dump data to *dpdata.plugins.pwmat.PwmatOutputFormat* format.

to_pymatgen_ComputedStructureEntry(**args*, ***kwargs*)

Dump data to *dpdata.plugins.pymatgen.PyMatgenCSEFormat* format.

to_pymatgen_computedstructureentry(**args*, ***kwargs*)

Dump data to *dpdata.plugins.pymatgen.PyMatgenCSEFormat* format.

to_pymatgen_molecule(**args*, ***kwargs*)

Dump data to *dpdata.plugins.pymatgen.PyMatgenMoleculeFormat* format.

to_pymatgen_structure(**args*, ***kwargs*)

Dump data to *dpdata.plugins.pymatgen.PyMatgenStructureFormat* format.

to_qe_cp_traj(**args*, ***kwargs*)

Dump data to *dpdata.plugins.qe.QECPTrajFormat* format.

to_qe_pw_scf(**args*, ***kwargs*)

Dump data to *dpdata.plugins.qe.QECPPWSCFFormat* format.

to_quip_gap_xyz(**args*, ***kwargs*)

Dump data to *dpdata.plugins.xyz.QuipGapXYZFormat* format.

to_quip_gap_xyz_file(**args*, ***kwargs*)

Dump data to *dpdata.plugins.xyz.QuipGapXYZFormat* format.

to_sdf(**args*, ***kwargs*)

Dump data to *dpdata.plugins.rdkit.SdfFormat* format.

to_sdf_file(**args*, ***kwargs*)

Dump data to *dpdata.plugins.rdkit.SdfFormat* format.

to_siesta_aimd_output(**args*, ***kwargs*)

Dump data to *dpdata.plugins.siesta.SiestaAIMDOutputFormat* format.

to_siesta_output(**args*, ***kwargs*)

Dump data to *dpdata.plugins.siesta.SiestaOutputFormat* format.

to_sqm_in(**args*, ***kwargs*)

Dump data to *dpdata.plugins.amber.SQMINFormat* format.

to_sqm_out(**args*, ***kwargs*)

Dump data to *dpdata.plugins.amber.SQMOutFormat* format.

to_stru(**args*, ***kwargs*)

Dump data to *dpdata.plugins.abacus.AbacusSTRUFormat* format.

to_vasp_contcar(**args*, ***kwargs*)

Dump data to *dpdata.plugins.vasp.VASPPoscarFormat* format.

to_vasp_outcar(**args*, ***kwargs*)

Dump data to *dpdata.plugins.vasp.VASPOutcarFormat* format.

to_vasp_poscar(**args*, ***kwargs*)

Dump data to *dpdata.plugins.vasp.VASPPoscarFormat* format.

to_vasp_string(**args*, ***kwargs*)

Dump data to *dpdata.plugins.vasp.VASPStringFormat* format.

to_vasp_xml(**args*, ***kwargs*)

Dump data to *dpdata.plugins.vasp.VASPXMLFormat* format.

to_xml(**args*, ***kwargs*)

Dump data to *dpdata.plugins.vasp.VASPXMLFormat* format.

to_xyz(**args*, ***kwargs*)

Dump data to *dpdata.plugins.xyz.XYZFormat* format.

property uniq_formula

Return the uniq_formula of this system. The uniq_formula sort the elements in formula by names. Systems with the same uniq_formula can be append together.

dpdata.system.**add_format_methods**()

Add format methods to System, LabeledSystem, and MultiSystems; add data types to System and LabeledSystem.

Notes

Ensure all plugins have been loaded before executating this function!

dpdata.system.**get_atom_perturb_vector**(*atom_pert_distance*, *atom_pert_style='normal'*)

dpdata.system.**get_cell_perturb_matrix**(*cell_pert_fraction*)

dpdata.system.**get_cls_name**(*cls: object*) → str

Returns the fully qualified name of a class, such as *np.ndarray*.

Parameters

cls
[object] the class

Returns

str
the fully qualified name of a class

dpdata.system.**load_format**(*fmt*)

6.1.13 dpdata.unit module

class dpdata.unit.**Conversion**(*unitA*, *unitB*, *check=True*)

Bases: ABC

Methods

set_value
value

set_value(*value*)

value()

class dpdata.unit.**EnergyConversion**(*unitA*, *unitB*)

Bases: *Conversion*

Methods

set_value
value

class dpdata.unit.**ForceConversion**(*unitA*, *unitB*)

Bases: *Conversion*

Methods

set_value
value

class dpdata.unit.**LengthConversion**(*unitA*, *unitB*)

Bases: *Conversion*

Methods

set_value
value

class dpdata.unit.**PressureConversion**(*unitA*, *unitB*)

Bases: *Conversion*

Methods

set_value
value

dpdata.unit.**check_unit**(*unit*)

6.1.14 dpdata.utils module

dpdata.utils.**add_atom_names**(*data*, *atom_names*)

Add atom_names that do not exist.

dpdata.utils.**elements_index_map**(*elements*, *standard=False*, *inverse=False*)

dpdata.utils.**remove_pbc**(*system*, *protect_layer=9*)

dpdata.utils.**sort_atom_names**(*data*, *type_map=None*)

Sort atom_names of the system and reorder atom_numbs and atom_types accoarding to atom_names. If type_map is not given, atom_names will be sorted by alphabetical order. If type_map is given, atom_names will be type_map.

Parameters

data
[dict] system data

type_map
[list] type_map

dpdata.utils.**uniq_atom_names**(*data*)

Make the atom names uniq. For example [‘O’, ‘H’, ‘O’, ‘H’, ‘O’] -> [‘O’, ‘H’].

Parameters

data
[dict] data dict of *System*, *LabeledSystem*

dpdata.utils.**utf8len**(*s: str*) → int

Return the byte length of a string.

CHAPTER

SEVEN

AUTHORS

- A bot of @njzjz
- AngelJia
- AnguseZhang
- C. Thang Nguyen
- Chen Tao
- Chentao168
- Chenxing Luo
- Duo
- Ericwang6
- Feifei Tian
- Han Wang
- Haruka
- Hedley
- Hedley Dong
- HuangJiameng
- Jia-Xin Zhu
- Jingchao Zhang
- Jinzhe Zeng
- Junhan Chang
- L-RuiHao
- Levi Zhou
- LiangWenshuo1118
- Linfeng Zhang
- Liu Renxi
- Liu-RX
- LiuGroupHNU
- LiuHanyu

- LiuLiping
- Logan Ward
- Marián Rynik
- PKUfjh
- Pan Xiang
- Peng Xingliang
- Shigetomo Yanase
- Silvia-liu
- Wanrun Jiang
- Wenxin Zhang
- Yifan Li
- Yingze Wang
- Yixiao Chen
- Yongbin Zhuang
- Yu Liu
- Yuan Fengbo
- Zezhong Zhang
- deepmodeling
- ericwang6
- felix5572
- haidi
- hl2500
- hongriTianqi
- jameswind
- kiwi
- link89
- liuliping
- pee8379
- pre-commit-ci[bot]
- pxlxingliang
- repo-ranger[bot]
- robinzhuang
- robinzyb
- tianhongzhen
- tuoping
- wang laosi

- xfanak
- yuzhi
- zezhong-zhang

CHAPTER

EIGHT

DPDATA

dpdata is a python package for manipulating data formats of software in computational science, including DeePMD-kit, VASP, LAMMPS, GROMACS, Gaussian. dpdata only works with python 3.7 or above.

8.1 Installation

One can download the source code of dpdata by

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

then use `pip` to install the module from source

```
cd dpdata
pip install .
```

`dpdata` can also by install via pip without source

```
pip install dpdata
```

8.2 Quick start

This section gives some examples on how dpdata works. Firstly one needs to import the module in a python 3.x compatible code.

```
import dpdata
```

The typicall workflow of `dpdata` is

1. Load data from vasp or lammps or deepmd-kit data files.
2. Manipulate data
3. Dump data to in a desired format

8.2.1 Load data

```
d_poscar = dpdata.System("POSCAR", fmt="vasp/poscar")
```

or let dpdata infer the format (`vasp/poscar`) of the file from the file name extension

```
d_poscar = dpdata.System("my.POSCAR")
```

The number of atoms, atom types, coordinates are loaded from the `POSCAR` and stored to a data `System` called `d_poscar`. A data `System` (a concept used by deepmd-kit) contains frames that has the same number of atoms of the same type. The order of the atoms should be consistent among the frames in one `System`. It is noted that `POSCAR` only contains one frame. If the multiple frames stored in, for example, a `OUTCAR` is wanted,

```
d_outcar = dpdata.LabeledSystem("OUTCAR")
```

The labels provided in the `OUTCAR`, i.e. energies, forces and virials (if any), are loaded by `LabeledSystem`. It is noted that the forces of atoms are always assumed to exist. `LabeledSystem` is a derived class of `System`.

The `System` or `LabeledSystem` can be constructed from the following file formats with the `format key` in the table passed to argument `fmt`:

Software	format	multi frames	labeled	class	format key
vasp	poscar	False	False	System	'vasp/poscar'
vasp	outcar	True	True	LabeledSystem	'vasp/outcar'
vasp	xml	True	True	LabeledSystem	'vasp/xml'
lammps	lmp	False	False	System	'lammps/lmp'
lammps	dump	True	False	System	'lammps/dump'
deepmd	raw	True	False	System	'deepmd/raw'
deepmd	npy	True	False	System	'deepmd/npy'
deepmd	raw	True	True	LabeledSystem	'deepmd/raw'
deepmd	npy	True	True	LabeledSystem	'deepmd/npy'
deepmd	npy	True	True	MultiSystems	'deepmd/npy/mixed'
deepmd	npy	True	False	MultiSystems	'deepmd/npy/mixed'
gaussian	log	False	True	LabeledSystem	'gaussian/log'
gaussian	log	True	True	LabeledSystem	'gaussian/md'
siesta	output	False	True	LabeledSystem	'siesta/output'
siesta	aimd_output	True	True	LabeledSystem	'siesta/aimd_output'
cp2k(deprecated in future)	output	False	True	LabeledSystem	'cp2k/output'
cp2k(deprecated in future)	aimd_output	True	True	LabeledSystem	'cp2k/aimd_output'
cp2k(plug-in)	stdout	False	True	LabeledSystem	'cp2kdata/e_f'
cp2k(plug-in)	stdout	True	True	LabeledSystem	'cp2kdata/md'
QE	log	False	True	LabeledSystem	'qe/pw/scf'
QE	log	True	False	System	'qe/cp/traj'
QE	log	True	True	LabeledSystem	'qe/cp/traj'
Fhi-aims	output	True	True	LabeledSystem	'fhi_aims/md'
Fhi-aims	output	False	True	LabeledSystem	'fhi_aims/scf'
quip/gap	xyz	True	True	MultiSystems	'quip/gap/xyz'
PWmat	atom.config	False	False	System	'pwmat/atom.config'
PWmat	movement	True	True	LabeledSystem	'pwmat/movement'
PWmat	OUT.MLMD	True	True	LabeledSystem	'pwmat/out.mlmd'
Amber	multi	True	True	LabeledSystem	'amber/md'
Amber/sqm	sqm.out	False	False	System	'sqm/out'
Gromacs	gro	True	False	System	'gromacs/gro'

continues on next page

Table 1 – continued from previous page

Software	format	multi frames	labeled	class	format key
ABACUS	STRU	False	False	System	'abacus/stru'
ABACUS	STRU	False	True	LabeledSystem	'abacus/scf'
ABACUS	cif	True	True	LabeledSystem	'abacus/md'
ABACUS	STRU	True	True	LabeledSystem	'abacus/relax'
ase	structure	True	True	MultiSystems	'ase/structure'
DFTB+	dftbplus	False	True	LabeledSystem	'dftbplus'
n2p2	n2p2	True	True	LabeledSystem	'n2p2'

The Class `dpdata.MultiSystems` can read data from a dir which may contains many files of different systems, or from single xyz file which contains different systems.

Use `dpdata.MultiSystems.from_dir` to read from a directory, `dpdata.MultiSystems` will walk in the directory Recursively and find all file with specific file_name. Supports all the file formats that `dpdata.LabeledSystem` supports.

Use `dpdata.MultiSystems.from_file` to read from single file. Single-file support is available for the `quip/gap/xyz` and `ase/structure` formats.

For example, for `quip/gap xyz` files, single .xyz file may contain many different configurations with different atom numbers and atom type.

The following commands relating to `Class dpdata.MultiSystems` may be useful.

```
# load data

xyz_multi_systems = dpdata.MultiSystems.from_file(
    file_name="tests/xyz/xyz_unittest.xyz", fmt="quip/gap/xyz"
)
vasp_multi_systems = dpdata.MultiSystems.from_dir(
    dir_name="./mgal_outcar", file_name="OUTCAR", fmt="vasp/outcar"
)

# use wildcard
vasp_multi_systems = dpdata.MultiSystems.from_dir(
    dir_name="./mgal_outcar", file_name="*OUTCAR", fmt="vasp/outcar"
)

# print the multi_system infomation
print(xyz_multi_systems)
print(xyz_multi_systems.systems)  # return a dictionaries

# print the system infomation
print(xyz_multi_systems.systems["B1C9"].data)

# dump a system's data to ./my_work_dir/B1C9_raw folder
xyz_multi_systems.systems["B1C9"].to_deepmd_raw("./my_work_dir/B1C9_raw")

# dump all systems
xyz_multi_systems.to_deepmd_raw("./my_deepmd_data/")
```

You may also use the following code to parse muti-system:

```
from dpdata import LabeledSystem, MultiSystems
from glob import glob

"""
process multi systems
"""
fs = glob("./*/OUTCAR")  # remeber to change here !!!
ms = MultiSystems()
for f in fs:
    try:
        ls = LabeledSystem(f)
    except:
        print(f)
    if len(ls) > 0:
        ms.append(ls)

ms.to_deepmd_raw("deepmd")
ms.to_deepmd_npy("deepmd")
```

8.2.2 Access data

These properties stored in `System` and `LabeledSystem` can be accessed by operator `[]` with the key of the property supplied, for example

```
coords = d_outcar["coords"]
```

Available properties are (nframe: number of frames in the system, natoms: total number of atoms in the system)

key	type	dimension	are labels	description
'atom_names'	list of str	ntypes	False	The name of each atom type
'atom_numbs'	list of int	ntypes	False	The number of atoms of each atom type
'atom_types'	np.ndarray	natoms	False	Array assigning type to each atom
'cells'	np.ndarray	nframes x 3 x 3	False	The cell tensor of each frame
'coords'	np.ndarray	nframes x natoms x 3	False	The atom coordinates
'energies'	np.ndarray	nframes	True	The frame energies
'forces'	np.ndarray	nframes x natoms x 3	True	The atom forces
'virials'	np.ndarray	nframes x 3 x 3	True	The virial tensor of each frame

8.2.3 Dump data

The data stored in `System` or `LabeledSystem` can be dumped in 'lammps/lmp' or 'vasp/poscar' format, for example:

```
d_outcar.to("lammps/lmp", "conf.lmp", frame_idx=0)
```

The first frames of `d_outcar` will be dumped to 'conf.lmp'

```
d_outcar.to("vasp/poscar", "POSCAR", frame_idx=-1)
```

The last frames of `d_outcar` will be dumped to 'POSCAR'.

The data stored in `LabeledSystem` can be dumped to deepmd-kit raw format, for example

```
d_outcar.to("deepmd/raw", "dpmd_raw")
```

Or a simpler command:

```
dpdata.LabeledSystem("OUTCAR").to("deepmd/raw", "dpmd_raw")
```

Frame selection can be implemented by

```
dpdata.LabeledSystem("OUTCAR").sub_system([0, -1]).to("deepmd/raw", "dpmd_raw")
```

by which only the first and last frames are dumped to `dpmd_raw`.

8.2.4 replicate

dpdata will create a super cell of the current atom configuration.

```
dpdata.System("./POSCAR").replicate(
    (
        1,
        2,
        3,
    )
)
```

tuple(1,2,3) means don't copy atom configuration in x direction, make 2 copys in y direction, make 3 copys in z direction.

8.2.5 perturb

By the following example, each frame of the original system (`dpdata.System('./POSCAR')`) is perturbed to generate three new frames. For each frame, the cell is perturbed by 5% and the atom positions are perturbed by 0.6 Angstrom. `atom_pert_style` indicates that the perturbation to the atom positions is subject to normal distribution. Other available options to `atom_pert_style` are`uniform` (uniform in a ball), and `const` (uniform on a sphere).

```
perturbed_system = dpdata.System("./POSCAR").perturb(
    pert_num=3,
    cell_pert_fraction=0.05,
    atom_pert_distance=0.6,
    atom_pert_style="normal",
)
print(perturbed_system.data)
```

8.2.6 replace

By the following example, Random 8 Hf atoms in the system will be replaced by Zr atoms with the atom postion unchanged.

```
s = dpdata.System("tests/poscars/POSCAR.P42nmc", fmt="vasp/poscar")
s.replace("Hf", "Zr", 8)
s.to_vasp_poscar("POSCAR.P42nmc.replace")
```

8.3 BondOrderSystem

A new class `BondOrderSystem` which inherits from class `System` is introduced in dpdata. This new class contains information of chemical bonds and formal charges (stored in `BondOrderSystem.data['bonds']`, `BondOrderSystem.data['formal_charges']`). Now BondOrderSystem can only read from .mol/.sdf formats, because of its dependency on rdkit (which means rdkit must be installed if you want to use this function). Other formats, such as pdb, must be converted to .mol/.sdf format (maybe with software like open babel).

```
import dpdata

system_1 = dpdata.BondOrderSystem(
    "tests/bond_order/CH3OH.mol", fmt="mol"
)  # read from .mol file
system_2 = dpdata.BondOrderSystem(
    "tests/bond_order/methane.sdf", fmt="sdf"
)  # read from .sdf file
```

In sdf file, all molecules must be of the same topology (i.e. conformers of the same molecular configuration). `BondOrderSystem` also supports initialize from a `rdkit.Chem.rdchem.Mol` object directly.

```
from rdkit import Chem
from rdkit.Chem import AllChem
import dpdata

mol = Chem.MolFromSmiles("CC")
mol = Chem.AddHs(mol)
AllChem.EmbedMultipleConfs(mol, 10)
system = dpdata.BondOrderSystem(rdkit_mol=mol)
```

8.3.1 Bond Order Assignment

The `BondOrderSystem` implements a more robust sanitize procedure for rdkit Mol, as defined in `dpdata.rdkit.santizie.Sanitizer`. This class defines 3 level of sanitization process by: low, medium and high. (default is medium).

- low: use `rdkit.Chem.SanitizeMol()` function to sanitize molecule.
- medium: before using rdkit, the programm will first assign formal charge of each atom to avoid inappropriate valence exceptions. However, this mode requires the rightness of the bond order information in the given molecule.
- high: the program will try to fix inappropriate bond orders in aromatic hetreocycles, phosphate, sulfate, carboxyl, nitro, nitrine, guanidine groups. If this procedure fails to sanitize the given molecule, the program will then try to call `obabel` to pre-process the mol and repeat the sanitization procedure. **That is to say, if you wan't to use this level of sanitization, please ensure ``obabel`` is installed in the environment.** According to our test, our sanitization procedure can successfully read 4852 small molecules in the PDBBind-refined-set. It is necessary to point out that the in the molecule file (mol/sdf), the number of explicit hydrogens has to be correct. Thus, we recommend to use `obabel xxx -O xxx -h` to pre-process the file. The reason why we do not implement this hydrogen-adding procedure in dpdata is that we can not ensure its correctness.

```
import dpdata

for sdf_file in glob.glob("bond_order/refined-set-ligands/obabel/*sdf"):
    syst = dpdata.BondOrderSystem(sdf_file, sanitize_level="high", verbose=False)
```

8.3.2 Formal Charge Assignment

BondOrderSystem implement a method to assign formal charge for each atom based on the 8-electron rule (see below). Note that it only supports common elements in bio-system: B,C,N,O,P,S,As

```
import dpdata

syst = dpdata.BondOrderSystem("tests/bond_order/CH3NH3+.mol", fmt="mol")
print(syst.get_formal_charges())  # return the formal charge on each atom
print(syst.get_charge())  # return the total charge of the system
```

If a valence of 3 is detected on carbon, the formal charge will be assigned to -1. Because for most cases (in alkynyl anion, isonitrile, cyclopentadienyl anion), the formal charge on 3-valence carbon is -1, and this is also consisent with the 8-electron rule.

8.4 Mixed Type Format

The format `deepmd/npy/mixed` is the mixed type numpy format for DeePMD-kit, and can be loaded or dumped through class `dpdata.MultiSystems`.

Under this format, systems with the same number of atoms but different formula can be put together for a larger system, especially when the frame numbers in systems are sparse.

This also helps to mixture the type information together for model training with type embedding network.

Here are examples using `deepmd/npy/mixed` format:

- Dump a MultiSystems into a mixed type numpy directory: ```python import dpdata

dpdata.MultiSystems(*systems).to_deepmd_npy_mixed("mixed_dir")

```
- Load a mixed type data into a MultiSystems:
```python
import dpdata

dpdata.MultiSystems().load_systems_from_file("mixed_dir", fmt="deepmd/npy/mixed")
```

8.5 Plugins

One can follow a simple example to add their own format by creating and installing plugins. It's critical to add the Format class to `entry_points['dpdata.plugins']` in ``*pyproject.toml*`` <plugin_example/pyproject.toml>`_:

```
[project.entry-points.'dpdata.plugins']
random = "dpdata_random:RandomFormat"
```
```

CHAPTER

NINE

INDICES AND TABLES

- genindex
- modindex
- search

BIBLIOGRAPHY

[1] Gao, X.; Ramezanghorbani, F.; Isayev, O.; Smith, J. S.; Roitberg, A. E. TorchANI: A Free and Open Source PyTorch-Based Deep Learning Implementation of the ANI Neural Network Potentials. J. Chem. Inf. Model. 2020, 60, 3408-3415.

[2] Zeng, J.; Tao, Y.; Giese, T. J.; York, D. M.. QD: A Quantum Deep Potential Interaction Model for Drug Discovery. J. Comput. Chem. 2023, 19, 1261-1275.

[1] Gao, X.; Ramezanghorbani, F.; Isayev, O.; Smith, J. S.; Roitberg, A. E. TorchANI: A Free and Open Source PyTorch-Based Deep Learning Implementation of the ANI Neural Network Potentials. J. Chem. Inf. Model. 2020, 60, 3408-3415.

[2] Zeng, J.; Tao, Y.; Giese, T. J.; York, D. M.. QD: A Quantum Deep Potential Interaction Model for Drug Discovery. J. Comput. Chem. 2023, 19, 1261-1275.

PYTHON MODULE INDEX

d

INDEX

D

E

F

G

H

I

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Z