dpdata Documentation

Release 0.0.0-rc

Han Wang

May 08, 2022

CONTENTS:

CHAPTER

ONE

SUPPORTED FORMATS

dpdata supports the following formats:

Table 1: Supported

Class	Alias	Supported Functio
ListFormat	list	*LabeledSystem.t*
DeePMDRawFormat	deepmd/raw deepmd	*MultiSystems.lo*
DeePMDCompFormat	deepmd/comp deepmd/npy	*MultiSystems.lo*
DeePMDHDF5Format	deepmd/hdf5	*MultiSystems.lo*
GaussianLogFormat	gaussian/log	*LabeledSystem()*
GaussianMDFormat	gaussian/md	*LabeledSystem()*
GromacsGroFormat	gromacs/gro gro	*LabeledSystem.t*
VASPPoscarFormat	vasp/contcar vasp/poscar contcar poscar	*LabeledSystem.t*
VASPStringFormat	vasp/string	*LabeledSystem.t*
VASPOutcarFormat	vasp/outcar outcar	*LabeledSystem()*
VASPXMLFormat	vasp/xml xml	*LabeledSystem()*
CP2KAIMDOutputFormat	cp2k/aimd_output	*LabeledSystem()*
CP2KOutputFormat	cp2k/output	*LabeledSystem()*
XYZFormat	xyz	*LabeledSystem.t*
QuipGapXYZFormat	quip/gap/xyz_file quip/gap/xyz	*MultiSystems.lo*
AbacusSCFFormat	abacus/lcao/scf abacus/pw/scf abacus/scf	*LabeledSystem()*
AbacusMDFormat	abacus/lcao/md abacus/pw/md abacus/md	*LabeledSystem()*
SiestaOutputFormat	siesta/output	*LabeledSystem()*
SiestaAIMDOutputFormat	siesta/aimd_output	*LabeledSystem()*
MolFormat	mol_file mol	BondOrderSystem
SdfFormat	sdf_file sdf	BondOrderSystem
LAMMPSLmpFormat	lammps/lmp lmp	*LabeledSystem.t*
LAMMPSDumpFormat	lammps/dump dump	*System()*
AmberMDFormat	amber/md	*LabeledSystem()*
SQMOutFormat	sqm/out	*LabeledSystem()*
SQMINFormat	sqm/in	*LabeledSystem.t*
PyMatgenStructureFormat	pymatgen/structure	*LabeledSystem.t*
PyMatgenMoleculeFormat	pymatgen/molecule	*LabeledSystem.t*
PyMatgenCSEFormat	pymatgen/computedstructureentry	*LabeledSystem.t*
PwmatOutputFormat	pwmat/output pwmat/mlmd pwmat/movement mlmd movement	*LabeledSystem()*
PwmatAtomconfigFormat	pwmat/final.config pwmat/atom.config final.config atom.config	*LabeledSystem.t*
QECPTrajFormat	qe/cp/traj	*LabeledSystem()*
QECPPWSCFFormat	qe/pw/scf	*LabeledSystem()*
FhiMDFormat	fhi_aims/output fhi_aims/md	*LabeledSystem()*

Table 1 – continued from

Class	Alias	Supported Functi
FhiSCFFormat	fhi_aims/scf	*LabeledSystem()*
ASEStructureFormat	ase/structure	*MultiSystems.lo*

CHAPTER

TWO

API DOCUMENTATION

2.1 dpdata package

2.1.1 Subpackages

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.

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*, *folder: str*, *data: dict*, *set_size=5000*, *comp_prec=<class 'numpy.float32'>*) → None

Dump data to a HDF5 file.

Parameters

f [h5py.File] HDF5 file 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: h5py._hl.files.File*, *folder: str*, *type_map: Optional[list] = None*, *labels: bool = True*)

Load a HDF5 file.

Parameters

f [h5py.File] HDF5 file object

folder [str] path in the HDF5 file

type_map [list] type map

labels [bool] labels

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.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*)

dpdata.fhi_aims.output.**get_info**(*lines*, *type_idx_zero=False*)

dpdata.gaussian package

Submodules

dpdata.gaussian.log module

dpdata.gaussian.log.**to_system_data**(*file_name*, *md=False*)

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

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, 0.0, 0.0])*)

dpdata.lammps.dump.**split_traj**(*dump_lines*)

dpdata.lammps.dump.**system_data**(*lines*, *type_map=None*, *type_idx_zero=True*)

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.plugins package

Submodules

dpdata.plugins.abacus module

class dpdata.plugins.abacus.**AbacusMDFormat**

Bases: *dpdata.format.Format*

Methods

MultiModes()	File mode for MultiSystems 0 (default): not implemented 1: every directory under the top-level directory is a system
from_labeled_system(file_name, **kwargs)	

from_multi_systems(directory, **kwargs)	MultiSystems.from
from_system(file_name, **kwargs)	System.from
register(key)	Register a virtual subclass of an ABC.
to_system(data, *args, **kwargs)	System.to

from_bond_order_system	
get_formats	
get_from_methods	
get_to_methods	
post	
register_from	
register_to	
to_bond_order_system	
to_labeled_system	
to_multi_systems	

from_labeled_system(*file_name*, ***kwargs*)

class dpdata.plugins.abacus.**AbacusSCFFormat**

Bases: *dpdata.format.Format*

Methods

MultiModes()	File mode for MultiSystems 0 (default): not implemented 1: every directory under the top-level directory is a system
from_labeled_system(file_name, **kwargs)	

from_multi_systems(directory, **kwargs)	MultiSystems.from
from_system(file_name, **kwargs)	System.from
register(key)	Register a virtual subclass of an ABC.
to_system(data, *args, **kwargs)	System.to

from_bond_order_system	
get_formats	
get_from_methods	
get_to_methods	
post	
register_from	
register_to	
to_bond_order_system	
to_labeled_system	
to_multi_systems	

from_labeled_system(*file_name*, ***kwargs*)

dpdata.plugins.amber module

class dpdata.plugins.amber.**AmberMDFormat**

Bases: *dpdata.format.Format*

Methods

MultiModes()	File mode for MultiSystems 0 (default): not implemented 1: every directory under the top-level directory is a system
from_multi_systems(directory, **kwargs)	MultiSystems.from
from_system([file_name, parm7_file, ...])	System.from
register(key)	Register a virtual subclass of an ABC.
to_system(data, *args, **kwargs)	System.to

from_bond_order_system	
from_labeled_system	
get_formats	
get_from_methods	
get_to_methods	
post	
register_from	
register_to	
to_bond_order_system	
to_labeled_system	
to_multi_systems	

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*)

from_system(*file_name=None*, *parm7_file=None*, *nc_file=None*, *use_element_symbols=None*, ***kwargs*)

System.from

Parameters

file_name: str file name

Returns

data: dict system data

class dpdata.plugins.amber.**SQMINFormat**

Bases: *dpdata.format.Format*

Methods

MultiModes()	File mode for MultiSystems 0 (default): not implemented 1: every directory under the top-level directory is a system
from_multi_systems(directory, **kwargs)	MultiSystems.from
from_system(file_name, **kwargs)	System.from
register(key)	Register a virtual subclass of an ABC.
to_system(data[, fname, frame_idx])	Generate input files for semi-emperical calculation in sqm software

from_bond_order_system
from_labeled_system
get_formats
get_from_methods
get_to_methods
post
register_from
register_to
to_bond_order_system
to_labeled_system
to_multi_systems

to_system(*data*, *fname=None*, *frame_idx=0*, ***kwargs*)

Generate input files for semi-emperical calculation in sqm software

class dpdata.plugins.amber.**SQMOutFormat**

Bases: *dpdata.format.Format*

Methods

MultiModes()	File mode for MultiSystems 0 (default): not implemented 1: every directory under the top-level directory is a system
from_labeled_system(fname, **kwargs)	Read from ambertools sqm.out
from_multi_systems(directory, **kwargs)	MultiSystems.from
from_system(fname, **kwargs)	Read from ambertools sqm.out
register(key)	Register a virtual subclass of an ABC.
to_system(data, *args, **kwargs)	System.to

from_bond_order_system	
get_formats	
get_from_methods	
get_to_methods	
post	
register_from	
register_to	
to_bond_order_system	
to_labeled_system	
to_multi_systems	

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.**ASEStructureFormat**

Bases: *dpdata.format.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 0 (default): not implemented 1: every directory under the top-level directory is a system
from_multi_systems(file_name[, begin, end, ...])	MultiSystems.from
from_system(file_name, **kwargs)	System.from
register(key)	Register a virtual subclass of an ABC.
to_labeled_system(data, *args, **kwargs)	Convert System to ASE Atoms object.
to_system(data, **kwargs)	convert System to ASE Atom obj

from_bond_order_system	
from_labeled_system	
get_formats	
get_from_methods	
get_to_methods	
post	
register_from	
register_to	
to_bond_order_system	
to_multi_systems	

from_labeled_system(*data*, ***kwargs*)

from_multi_systems(*file_name*, *begin=None*, *end=None*, *step=None*, *ase_fmt=None*, ***kwargs*)

MultiSystems.from

Parameters

directory: str directory of system

Returns

filenames: list[str] list of filenames

to_labeled_system(*data*, **args*, ***kwargs*)

Convert System to ASE Atoms object.

to_system(*data*, ***kwargs*)

convert System to ASE Atom obj

dpdata.plugins.cp2k module

class dpdata.plugins.cp2k.**CP2KAIMDOutputFormat**

Bases: *dpdata.format.Format*

Methods

MultiModes()	File mode for MultiSystems 0 (default): not implemented 1: every directory under the top-level directory is a system
from_multi_systems(directory, **kwargs)	MultiSystems.from
from_system(file_name, **kwargs)	System.from
register(key)	Register a virtual subclass of an ABC.
to_system(data, *args, **kwargs)	System.to

from_bond_order_system	
from_labeled_system	
get_formats	
get_from_methods	
get_to_methods	
post	
register_from	
register_to	
to_bond_order_system	
to_labeled_system	
to_multi_systems	

from_labeled_system(*file_name*, *restart=False*, ***kwargs*)

class dpdata.plugins.cp2k.**CP2KOutputFormat**

Bases: *dpdata.format.Format*

Methods

MultiModes()	File mode for MultiSystems 0 (default): not implemented 1: every directory under the top-level directory is a system
from_multi_systems(directory, **kwargs)	MultiSystems.from
from_system(file_name, **kwargs)	System.from
register(key)	Register a virtual subclass of an ABC.
to_system(data, *args, **kwargs)	System.to

from_bond_order_system	
from_labeled_system	
get_formats	
get_from_methods	
get_to_methods	
post	
register_from	
register_to	
to_bond_order_system	
to_labeled_system	
to_multi_systems	

from_labeled_system(*file_name*, *restart=False*, ***kwargs*)

dpdata.plugins.deepmd module

class dpdata.plugins.deepmd.**DeePMDCompFormat**

Bases: *dpdata.format.Format*

Methods

MultiModes()	File mode for MultiSystems 0 (default): not implemented 1: every directory under the top-level directory is a system
from_multi_systems(directory, **kwargs)	MultiSystems.from
from_system(file_name[, type_map])	System.from
register(key)	Register a virtual subclass of an ABC.
to_system(data, file_name[, set_size, prec])	Dump the system in deepmd compressed format (numpy binary) to *folder*.

from_bond_order_system	
from_labeled_system	
get_formats	
get_from_methods	
get_to_methods	
post	
register_from	
register_to	
to_bond_order_system	
to_labeled_system	
to_multi_systems	

`MultiMode = 1`

`from_labeled_system`(*file_name*, *type_map=None*, ***kwargs*)

`from_system`(*file_name*, *type_map=None*, ***kwargs*)

System.from

Parameters

file_name: str file name

Returns

data: dict system data

`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

`class` dpdata.plugins.deepmd.**DeePMDHDF5Format**

Bases: *dpdata.format.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 0 (default): not implemented 1: every directory under the top-level directory is a system
from_multi_systems(directory, **kwargs)	MultiSystems.from
from_system(file_name[, type_map])	System.from
`register`(key)	Register a virtual subclass of an ABC.
to_system(data, file_name[, set_size, comp_prec])	System.to

from_bond_order_system
from_labeled_system
get_formats
get_from_methods
get_to_methods
post
register_from
register_to
to_bond_order_system
to_labeled_system
to_multi_systems

from_labeled_system(*file_name*, *type_map=None*, ***kwargs*)

from_multi_systems(*directory*, ***kwargs*)

MultiSystems.from

Parameters

directory: str directory of system

Returns

filenames: list[str] list of filenames

from_system(*file_name*, *type_map=None*, ***kwargs*)

System.from

Parameters

file_name: str file name

Returns

data: dict system data

to_multi_systems(*formulas*, *directory*, ***kwargs*)

to_system(*data: dict*, *file_name: str*, *set_size: int = 5000*, *comp_prec: numpy.dtype = <class 'numpy.float64'>*, ***kwargs*)

System.to

Parameters

data: dict system data

class dpdata.plugins.deepmd.**DeePMDRawFormat**

Bases: *dpdata.format.Format*

Methods

MultiModes()	File mode for MultiSystems 0 (default): not implemented 1: every directory under the top-level directory is a system
from_multi_systems(directory, **kwargs)	MultiSystems.from
from_system(file_name[, type_map])	System.from
register(key)	Register a virtual subclass of an ABC.
to_system(data, file_name, **kwargs)	Dump the system in deepmd raw format to directory *file_name*

from_bond_order_system	
from_labeled_system	
get_formats	
get_from_methods	
get_to_methods	
post	
register_from	
register_to	
to_bond_order_system	
to_labeled_system	
to_multi_systems	

MultiMode = 1

from_labeled_system(*file_name*, *type_map=None*, ***kwargs*)

from_system(*file_name*, *type_map=None*, ***kwargs*)

System.from

Parameters

file_name: str file name

Returns

data: dict system data

to_system(*data*, *file_name*, ***kwargs*)

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

dpdata.plugins.fhi_aims module

class dpdata.plugins.fhi_aims.**FhiMDFormat**

Bases: *dpdata.format.Format*

Methods

MultiModes()	File mode for MultiSystems 0 (default): not implemented 1: every directory under the top-level directory is a system
from_multi_systems(directory, **kwargs)	MultiSystems.from
from_system(file_name, **kwargs)	System.from
register(key)	Register a virtual subclass of an ABC.
to_system(data, *args, **kwargs)	System.to

from_bond_order_system	
from_labeled_system	
get_formats	
get_from_methods	
get_to_methods	
post	
register_from	
register_to	
to_bond_order_system	
to_labeled_system	
to_multi_systems	

from_labeled_system(*file_name*, *md=True*, *begin=0*, *step=1*, ***kwargs*)

class dpdata.plugins.fhi_aims.**FhiSCFFormat**

Bases: *dpdata.format.Format*

Methods

MultiModes()	File mode for MultiSystems 0 (default): not implemented 1: every directory under the top-level directory is a system
from_multi_systems(directory, **kwargs)	MultiSystems.from
from_system(file_name, **kwargs)	System.from
register(key)	Register a virtual subclass of an ABC.
to_system(data, *args, **kwargs)	System.to

from_bond_order_system	
from_labeled_system	
get_formats	
get_from_methods	
get_to_methods	
post	
register_from	
register_to	
to_bond_order_system	
to_labeled_system	
to_multi_systems	

from_labeled_system(*file_name*, ***kwargs*)

dpdata.plugins.gaussian module

class dpdata.plugins.gaussian.**GaussianLogFormat**

Bases: *dpdata.format.Format*

Methods

MultiModes()	File mode for MultiSystems 0 (default): not implemented 1: every directory under the top-level directory is a system
from_multi_systems(directory, **kwargs)	MultiSystems.from
from_system(file_name, **kwargs)	System.from
register(key)	Register a virtual subclass of an ABC.
to_system(data, *args, **kwargs)	System.to

from_bond_order_system	
from_labeled_system	
get_formats	
get_from_methods	
get_to_methods	
post	
register_from	
register_to	
to_bond_order_system	
to_labeled_system	
to_multi_systems	

from_labeled_system(*file_name*, *md=False*, ***kwargs*)

class dpdata.plugins.gaussian.**GaussianMDFormat**

Bases: *dpdata.format.Format*

Methods

MultiModes()	File mode for MultiSystems 0 (default): not implemented 1: every directory under the top-level directory is a system
from_multi_systems(directory, **kwargs)	MultiSystems.from
from_system(file_name, **kwargs)	System.from
register(key)	Register a virtual subclass of an ABC.
to_system(data, *args, **kwargs)	System.to

from_bond_order_system
from_labeled_system
get_formats
get_from_methods
get_to_methods
post
register_from
register_to
to_bond_order_system
to_labeled_system
to_multi_systems

from_labeled_system(*file_name*, ***kwargs*)

dpdata.plugins.gromacs module

class dpdata.plugins.gromacs.**GromacsGroFormat**

Bases: *dpdata.format.Format*

Methods

MultiModes()	File mode for MultiSystems 0 (default): not implemented 1: every directory under the top-level directory is a system
from_multi_systems(directory, **kwargs)	MultiSystems.from
from_system(file_name[, format_atom_name])	Load gromacs .gro file
register(key)	Register a virtual subclass of an ABC.
to_system(data[, file_name, frame_idx])	Dump the system in gromacs .gro format

from_bond_order_system	
from_labeled_system	
get_formats	
get_from_methods	
get_to_methods	
post	
register_from	
register_to	
to_bond_order_system	
to_labeled_system	
to_multi_systems	

from_system(*file_name*, *format_atom_name=True*, ***kwargs*)

Load gromacs .gro file

Parameters

file_name [str] The input file name

to_system(*data*, *file_name=None*, *frame_idx=- 1*, ***kwargs*)

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

dpdata.plugins.lammps module

class dpdata.plugins.lammps.**LAMMPSDumpFormat**

Bases: *dpdata.format.Format*

Methods

MultiModes()	File mode for MultiSystems 0 (default): not implemented 1: every directory under the top-level directory is a system
from_multi_systems(directory, **kwargs)	MultiSystems.from
from_system(file_name[, type_map, begin, step])	System.from
register(key)	Register a virtual subclass of an ABC.
to_system(data, *args, **kwargs)	System.to

from_bond_order_system	
from_labeled_system	
get_formats	
get_from_methods	
get_to_methods	
post	
register_from	
register_to	
to_bond_order_system	
to_labeled_system	
to_multi_systems	

from_system(*file_name*, *type_map=None*, *begin=0*, *step=1*, ***kwargs*)

System.from

Parameters

file_name: str file name

Returns

data: dict system data

class dpdata.plugins.lammps.**LAMMPSLmpFormat**

Bases: *dpdata.format.Format*

Methods

MultiModes()	File mode for MultiSystems 0 (default): not implemented 1: every directory under the top-level directory is a system
from_multi_systems(directory, **kwargs)	MultiSystems.from
from_system(file_name[, type_map])	System.from
register(key)	Register a virtual subclass of an ABC.
to_system(data, file_name[, frame_idx])	Dump the system in lammps data format

from_bond_order_system	
from_labeled_system	
get_formats	
get_from_methods	
get_to_methods	
post	
register_from	
register_to	
to_bond_order_system	
to_labeled_system	
to_multi_systems	

from_system(*file_name*, *type_map=None*, ***kwargs*)

System.from

Parameters

file_name: str file name

Returns

data: dict system data

`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

dpdata.plugins.list module

`class` dpdata.plugins.list.`ListFormat`

Bases: *dpdata.format.Format*

Methods

`MultiModes`()	File mode for MultiSystems 0 (default): not implemented 1: every directory under the top-level directory is a system
`from_multi_systems`(directory, **kwargs)	MultiSystems.from
`from_system`(file_name, **kwargs)	System.from
`register`(key)	Register a virtual subclass of an ABC.
to_system(data, **kwargs)	convert system to list, usefull for data collection

from_bond_order_system	
from_labeled_system	
get_formats	
get_from_methods	
get_to_methods	
post	
register_from	
register_to	
to_bond_order_system	
to_labeled_system	
to_multi_systems	

`to_system`(*data*, ***kwargs*)

convert system to list, usefull for data collection

dpdata.plugins.pwmat module

class dpdata.plugins.pwmat.**PwmatAtomconfigFormat**

Bases: *dpdata.format.Format*

Methods

MultiModes()	File mode for MultiSystems 0 (default): not implemented 1: every directory under the top-level directory is a system
from_multi_systems(directory, **kwargs)	MultiSystems.from
from_system(file_name, **kwargs)	System.from
register(key)	Register a virtual subclass of an ABC.
to_system(data, file_name[, frame_idx])	Dump the system in pwmat atom.config format

from_bond_order_system	
from_labeled_system	
get_formats	
get_from_methods	
get_to_methods	
post	
register_from	
register_to	
to_bond_order_system	
to_labeled_system	
to_multi_systems	

from_system(*file_name*, ***kwargs*)

System.from

Parameters

file_name: str file name

Returns

data: dict system data

to_system(*data*, *file_name*, *frame_idx=0*, **args*, ***kwargs*)

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

class dpdata.plugins.pwmat.**PwmatOutputFormat**

Bases: *dpdata.format.Format*

Methods

MultiModes()	File mode for MultiSystems 0 (default): not implemented 1: every directory under the top-level directory is a system
from_multi_systems(directory, **kwargs)	MultiSystems.from
from_system(file_name, **kwargs)	System.from
register(key)	Register a virtual subclass of an ABC.
to_system(data, *args, **kwargs)	System.to

from_bond_order_system	
from_labeled_system	
get_formats	
get_from_methods	
get_to_methods	
post	
register_from	
register_to	
to_bond_order_system	
to_labeled_system	
to_multi_systems	

from_labeled_system(*file_name*, *begin=0*, *step=1*, ***kwargs*)

dpdata.plugins.pymatgen module

class dpdata.plugins.pymatgen.**PyMatgenCSEFormat**

Bases: *dpdata.format.Format*

Methods

MultiModes()	File mode for MultiSystems 0 (default): not implemented 1: every directory under the top-level directory is a system
from_multi_systems(directory, **kwargs)	MultiSystems.from
from_system(file_name, **kwargs)	System.from
register(key)	Register a virtual subclass of an ABC.
to_labeled_system(data, *args, **kwargs)	convert System to Pymagen ComputedStructureEntry obj
to_system(data, *args, **kwargs)	System.to

from_bond_order_system	
from_labeled_system	
get_formats	
get_from_methods	
get_to_methods	
post	
register_from	
register_to	
to_bond_order_system	
to_multi_systems	

to_labeled_system(*data*, **args*, ***kwargs*)

convert System to Pymagen ComputedStructureEntry obj

class dpdata.plugins.pymatgen.**PyMatgenMoleculeFormat**

Bases: *dpdata.format.Format*

Methods

MultiModes()	File mode for MultiSystems 0 (default): not implemented 1: every directory under the top-level directory is a system
from_multi_systems(directory, **kwargs)	MultiSystems.from
from_system(file_name, **kwargs)	System.from
register(key)	Register a virtual subclass of an ABC.
to_system(data, **kwargs)	convert System to Pymatgen Molecule obj

from_bond_order_system	
from_labeled_system	
get_formats	
get_from_methods	
get_to_methods	
post	
register_from	
register_to	
to_bond_order_system	
to_labeled_system	
to_multi_systems	

from_system(*file_name*, ***kwargs*)

System.from

Parameters

file_name: str file name

Returns

data: dict system data

to_system(*data*, ***kwargs*)

convert System to Pymatgen Molecule obj

class dpdata.plugins.pymatgen.**PyMatgenStructureFormat**

Bases: *dpdata.format.Format*

Methods

MultiModes()	File mode for MultiSystems 0 (default): not implemented 1: every directory under the top-level directory is a system
from_multi_systems(directory, **kwargs)	MultiSystems.from
from_system(file_name, **kwargs)	System.from
register(key)	Register a virtual subclass of an ABC.
to_system(data, **kwargs)	convert System to Pymatgen Structure obj

from_bond_order_system	
from_labeled_system	
get_formats	
get_from_methods	
get_to_methods	
post	
register_from	
register_to	
to_bond_order_system	
to_labeled_system	
to_multi_systems	

to_system(*data*, ***kwargs*)

convert System to Pymatgen Structure obj

dpdata.plugins.qe module

class dpdata.plugins.qe.**QECPPWSCFFormat**

Bases: *dpdata.format.Format*

Methods

MultiModes()	File mode for MultiSystems 0 (default): not implemented 1: every directory under the top-level directory is a system
from_multi_systems(directory, **kwargs)	MultiSystems.from
from_system(file_name, **kwargs)	System.from
register(key)	Register a virtual subclass of an ABC.
to_system(data, *args, **kwargs)	System.to

from_bond_order_system	
from_labeled_system	
get_formats	
get_from_methods	
get_to_methods	
post	
register_from	
register_to	
to_bond_order_system	
to_labeled_system	
to_multi_systems	

from_labeled_system(*file_name*, ***kwargs*)

class dpdata.plugins.qe.**QECPTrajFormat**

Bases: *dpdata.format.Format*

Methods

MultiModes()	File mode for MultiSystems 0 (default): not implemented 1: every directory under the top-level directory is a system
from_multi_systems(directory, **kwargs)	MultiSystems.from
from_system(file_name[, begin, step])	System.from
register(key)	Register a virtual subclass of an ABC.
to_system(data, *args, **kwargs)	System.to

from_bond_order_system	
from_labeled_system	
get_formats	
get_from_methods	
get_to_methods	
post	
register_from	
register_to	
to_bond_order_system	
to_labeled_system	
to_multi_systems	

from_labeled_system(*file_name*, *begin=0*, *step=1*, ***kwargs*)

from_system(*file_name*, *begin=0*, *step=1*, ***kwargs*)

System.from

Parameters

file_name: str file name

Returns

data: dict system data

dpdata.plugins.rdkit module

class dpdata.plugins.rdkit.**MolFormat**

Bases: *dpdata.format.Format*

Methods

MultiModes()	File mode for MultiSystems 0 (default): not implemented 1: every directory under the top-level directory is a system
from_multi_systems(directory, **kwargs)	MultiSystems.from
from_system(file_name, **kwargs)	System.from
register(key)	Register a virtual subclass of an ABC.
to_system(data, *args, **kwargs)	System.to

from_bond_order_system
from_labeled_system
get_formats
get_from_methods
get_to_methods
post
register_from
register_to
to_bond_order_system
to_labeled_system
to_multi_systems

from_bond_order_system(*file_name*, ***kwargs*)

to_bond_order_system(*data*, *mol*, *file_name*, *frame_idx=0*, ***kwargs*)

class dpdata.plugins.rdkit.**SdfFormat**

Bases: *dpdata.format.Format*

Methods

MultiModes()	File mode for MultiSystems 0 (default): not implemented 1: every directory under the top-level directory is a system
from_bond_order_system(file_name, **kwargs)	Note that it requires all molecules in .sdf file must be of the same topology
from_multi_systems(directory, **kwargs)	MultiSystems.from
from_system(file_name, **kwargs)	System.from
register(key)	Register a virtual subclass of an ABC.
to_system(data, *args, **kwargs)	System.to

from_labeled_system	
get_formats	
get_from_methods	
get_to_methods	
post	
register_from	
register_to	
to_bond_order_system	
to_labeled_system	
to_multi_systems	

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*)

dpdata.plugins.siesta module

class dpdata.plugins.siesta.**SiestaAIMDOutputFormat**

Bases: *dpdata.format.Format*

Methods

MultiModes()	File mode for MultiSystems 0 (default): not implemented 1: every directory under the top-level directory is a system
from_multi_systems(directory, **kwargs)	MultiSystems.from
from_system(file_name, **kwargs)	System.from
register(key)	Register a virtual subclass of an ABC.
to_system(data, *args, **kwargs)	System.to

from_bond_order_system	
from_labeled_system	
get_formats	
get_from_methods	
get_to_methods	
post	
register_from	
register_to	
to_bond_order_system	
to_labeled_system	
to_multi_systems	

from_labeled_system(*file_name*, ***kwargs*)

from_system(*file_name*, ***kwargs*)

System.from

Parameters

file_name: str file name

Returns

data: dict system data

class dpdata.plugins.siesta.**SiestaOutputFormat**

Bases: *dpdata.format.Format*

Methods

MultiModes()	File mode for MultiSystems 0 (default): not implemented 1: every directory under the top-level directory is a system
from_multi_systems(directory, **kwargs)	MultiSystems.from
from_system(file_name, **kwargs)	System.from
register(key)	Register a virtual subclass of an ABC.
to_system(data, *args, **kwargs)	System.to

from_bond_order_system	
from_labeled_system	
get_formats	
get_from_methods	
get_to_methods	
post	
register_from	
register_to	
to_bond_order_system	
to_labeled_system	
to_multi_systems	

from_labeled_system(*file_name*, ***kwargs*)

from_system(*file_name*, ***kwargs*)

System.from

Parameters

file_name: str file name

Returns

data: dict system data

dpdata.plugins.vasp module

class dpdata.plugins.vasp.**VASPOutcarFormat**

Bases: *dpdata.format.Format*

Methods

MultiModes()	File mode for MultiSystems 0 (default): not implemented 1: every directory under the top-level directory is a system
from_multi_systems(directory, **kwargs)	MultiSystems.from
from_system(file_name, **kwargs)	System.from
register(key)	Register a virtual subclass of an ABC.
to_system(data, *args, **kwargs)	System.to

from_bond_order_system
from_labeled_system
get_formats
get_from_methods
get_to_methods
post
register_from
register_to
to_bond_order_system
to_labeled_system
to_multi_systems

from_labeled_system(*file_name*, *begin=0*, *step=1*, ***kwargs*)

class dpdata.plugins.vasp.**VASPPoscarFormat**

Bases: *dpdata.format.Format*

Methods

MultiModes()	File mode for MultiSystems 0 (default): not implemented 1: every directory under the top-level directory is a system
from_multi_systems(directory, **kwargs)	MultiSystems.from
from_system(file_name, **kwargs)	System.from
register(key)	Register a virtual subclass of an ABC.
to_system(data, file_name[, frame_idx])	Dump the system in vasp POSCAR format

from_bond_order_system	
from_labeled_system	
get_formats	
get_from_methods	
get_to_methods	
post	
register_from	
register_to	
to_bond_order_system	
to_labeled_system	
to_multi_systems	

from_system(*file_name*, ***kwargs*)

System.from

Parameters

file_name: str file name

Returns

data: dict system data

to_system(*data*, *file_name*, *frame_idx=0*, ***kwargs*)

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

class dpdata.plugins.vasp.**VASPStringFormat**

Bases: *dpdata.format.Format*

Methods

MultiModes()	File mode for MultiSystems 0 (default): not implemented 1: every directory under the top-level directory is a system
from_multi_systems(directory, **kwargs)	MultiSystems.from
from_system(file_name, **kwargs)	System.from
register(key)	Register a virtual subclass of an ABC.
to_system(data[, frame_idx])	Dump the system in vasp POSCAR format string

from_bond_order_system	
from_labeled_system	
get_formats	
get_from_methods	
get_to_methods	
post	
register_from	
register_to	
to_bond_order_system	
to_labeled_system	
to_multi_systems	

to_system(*data*, *frame_idx=0*, ***kwargs*)

Dump the system in vasp POSCAR format string

Parameters

frame_idx [int] The index of the frame to dump

class dpdata.plugins.vasp.**VASPXMLFormat**

Bases: *dpdata.format.Format*

Methods

MultiModes()	File mode for MultiSystems 0 (default): not implemented 1: every directory under the top-level directory is a system
from_multi_systems(directory, **kwargs)	MultiSystems.from
from_system(file_name, **kwargs)	System.from
register(key)	Register a virtual subclass of an ABC.
to_system(data, *args, **kwargs)	System.to

from_bond_order_system	
from_labeled_system	
get_formats	
get_from_methods	
get_to_methods	
post	
register_from	
register_to	
to_bond_order_system	
to_labeled_system	
to_multi_systems	

from_labeled_system(*file_name*, *begin=0*, *step=1*, ***kwargs*)

dpdata.plugins.xyz module

class dpdata.plugins.xyz.**QuipGapXYZFormat**

Bases: *dpdata.format.Format*

Methods

MultiModes()	File mode for MultiSystems 0 (default): not implemented 1: every directory under the top-level directory is a system
from_multi_systems(file_name, **kwargs)	MultiSystems.from
from_system(file_name, **kwargs)	System.from
register(key)	Register a virtual subclass of an ABC.
to_system(data, *args, **kwargs)	System.to

from_bond_order_system	
from_labeled_system	
get_formats	
get_from_methods	
get_to_methods	
post	
register_from	
register_to	
to_bond_order_system	
to_labeled_system	
to_multi_systems	

from_labeled_system(*data*, ***kwargs*)

from_multi_systems(*file_name*, ***kwargs*)

MultiSystems.from

Parameters

directory: str directory of system

Returns

filenames: list[str] list of filenames

class dpdata.plugins.xyz.**XYZFormat**

Bases: *dpdata.format.Format*

XYZ foramt.

Examples

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

Methods

MultiModes()	File mode for MultiSystems 0 (default): not implemented 1: every directory under the top-level directory is a system
from_multi_systems(directory, **kwargs)	MultiSystems.from
from_system(file_name, **kwargs)	System.from
register(key)	Register a virtual subclass of an ABC.
to_system(data, file_name, **kwargs)	System.to

from_bond_order_system
from_labeled_system
get_formats
get_from_methods
get_to_methods
post
register_from
register_to
to_bond_order_system
to_labeled_system
to_multi_systems

to_system(*data*, *file_name*, ***kwargs*)

System.to

Parameters

data: dict system data

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*)

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*)

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

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*)

dpdata.vasp.outcar.**get_frames**(*fname*, *begin=0*, *step=1*)

dpdata.vasp.outcar.**get_outcar_block**(*fp*)

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*)
can 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*)

2.1.2 Submodules

2.1.3 dpdata.bond_order_system module

2.1.4 dpdata.cli module

Command line interface for dpdata.

dpdata.cli.**convert**(*, *from_file: str*, *from_format: str = 'auto'*, *to_file: Optional[str] = None*, *to_format: Optional[str] = None*, *no_labeled: bool = False*, *multi: bool = False*, *type_map: Optional[list] = 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

dpdata.cli.**dpdata_cli**()

dpdata cli.

Examples

```
$ dpdata -ioposcar POSCAR -odeepmd/npy -O data -n
```

2.1.5 dpdata.format module

Implement the format plugin system.

class dpdata.format.**Format**

Bases: abc.ABC

Methods

MultiModes()	File mode for MultiSystems 0 (default): not implemented 1: every directory under the top-level directory is a system
from_multi_systems(directory, **kwargs)	MultiSystems.from
from_system(file_name, **kwargs)	System.from
register(key)	Register a virtual subclass of an ABC.
to_system(data, *args, **kwargs)	System.to

from_bond_order_system	
from_labeled_system	
get_formats	
get_from_methods	
get_to_methods	
post	
register_from	
register_to	
to_bond_order_system	
to_labeled_system	
to_multi_systems	

`MultiMode = 0`

`class MultiModes`

Bases: `object`

File mode for MultiSystems 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*)

`from_labeled_system`(*file_name*, ***kwargs*)

`from_multi_systems`(*directory*, ***kwargs*)

MultiSystems.from

Parameters

directory: str directory of system

Returns

filenames: list[str] list of filenames

`from_system`(*file_name*, ***kwargs*)

System.from

Parameters

file_name: str file name

Returns

data: dict system data

`static get_formats`()

`static get_from_methods`()

`static get_to_methods`()

`static post`(*func_name*)

static register(*key*)

Register a virtual subclass of an ABC.

Returns the subclass, to allow usage as a class decorator.

static register_from(*key*)

static register_to(*key*)

to_bond_order_system(*data*, *rdkit_mol*, **args*, ***kwargs*)

to_labeled_system(*data*, **args*, ***kwargs*)

to_multi_systems(*formulas*, *directory*, ***kwargs*)

to_system(*data*, **args*, ***kwargs*)

System.to

Parameters

data: dict system data

2.1.6 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

2.1.7 dpdata.plugin module

Base of plugin systems.

class dpdata.plugin.**Plugin**

Bases: object

A class to register plugins.

Examples

```
>>> Plugin = Register()
>>> @Plugin.register("xx")
    def xxx():
        pass
>>> print(Plugin.plugins['xx'])
```

Methods

register(key)	Register a plugin.

get_plugin	

get_plugin(*key*)

register(*key*)

Register a plugin.

2.1.8 dpdata.system module

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

Bases: *dpdata.system.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

- *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'*.

Attributes

formula Return the formula of this system, like C3H5O2

nopbc

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_type_map(type_map)	Assign atom_names to type_map if type_map is given and different from atom_names.
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_abacus_lcao_md(file_name, **kwargs)	Read data from *dpdata.plugins.abacus.AbacusMDFormat* 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_scf(file_name, **kwargs)	Read data from *dpdata.plugins.abacus.AbacusSCFFormat* format.
from_abacus_scf(file_name, **kwargs)	Read data from *dpdata.plugins.abacus.AbacusSCFFormat* 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_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 1 – continued from previous page

from_deepmd_raw(file_name, **kwargs)	Read data from *dpdata.plugins.deepmd.DeePMDRawFormat* 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_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_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_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 1 – 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_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
load(filename)	rebuild System obj.

continues on next page

Table 1 – continued from previous page

map_atom_types([type_map])	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
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(dp)	Predict energies and forces by deepmd-kit.
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()	Also shuffle labeled data e.g.
sort_atom_names([type_map])	Sort atom_names of the system and reorder atom_numbs and atom_types accoarding to atom_names.
sub_system(f_idx)	Construct a subsystem from the system
to(fmt, *args, **kwargs)	Dump systems to the specific format.
to_abacus_lcao_md(*args, **kwargs)	Dump data to *dpdata.plugins.abacus.AbacusMDFormat* 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_scf(*args, **kwargs)	Dump data to *dpdata.plugins.abacus.AbacusSCFFormat* format.
to_abacus_scf(*args, **kwargs)	Dump data to *dpdata.plugins.abacus.AbacusSCFFormat* 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_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.

continues on next page

Table 1 – continued from previous page

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_raw(*args, **kwargs)	Dump data to dpdata.plugins.deepmd.DeePMDRawFormat 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_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_outcar(*args, **kwargs)	Dump data to dpdata.plugins.vasp.VASPOutcarFormat format.
to_poscar(*args, **kwargs)	Dump data to dpdata.plugins.vasp.VASPPoscarFormat format.
to_pwmat_atomconfig(*args, **kwargs)	Dump data to dpdata.plugins.pwmat.PwmatAtomconfigFormat format.

continues on next page

Table 1 – 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_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.

continues on next page

Table 1 – continued from previous page

validate_monty(v)	pydantic Validator for MSONable pattern

affine_map	
affine_map_fv	
apply_type_map	
from_fmt	
from_fmt_obj	
has_virial	
replace	
rot_frame_lower_triangular	
rot_lower_triangular	
sort_atom_types	
to_fmt_obj	

affine_map_fv(*trans*, *f_idx*)

append(*system*)

Append a system to this system

Parameters

system [System] The system to append

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_abacus_lcao_md(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.abacus.AbacusMDFormat* 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_scf(*file_name*, ***kwargs*)

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

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

Read data from *dpdata.plugins.abacus.AbacusSCFFormat* 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_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_raw(*file_name*, ***kwargs*)

Read data from *dpdata.plugins.deepmd.DeePMDRawFormat* 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_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_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_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_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()

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: LabeledSystem new system

post_funcs = <dpdata.plugin.Plugin object>

rot_frame_lower_triangular(*f_idx=0*)

rot_lower_triangular()

shuffle()

Also shuffle labeled data e.g. energies and forces.

sort_atom_types()

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 [LabeledSystem] The subsystem

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

Dump data to *dpdata.plugins.abacus.AbacusMDFormat* 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_scf(**args*, ***kwargs*)

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

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

Dump data to *dpdata.plugins.abacus.AbacusSCFFormat* 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_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_raw(**args*, ***kwargs*)
Dump data to *dpdata.plugins.deepmd.DeePMDRawFormat* 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_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_outcar(**args*, ***kwargs*)

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

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

Dump data to *dpdata.plugins.vasp.VASPPoscarFormat* 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_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.
from_abacus_lcao_md(file_name, **kwargs)	Read data from *dpdata.plugins.abacus.AbacusMDFormat* 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_scf(file_name, **kwargs)	Read data from *dpdata.plugins.abacus.AbacusSCFFormat* format.
from_abacus_scf(file_name, **kwargs)	Read data from *dpdata.plugins.abacus.AbacusSCFFormat* 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_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_raw(file_name, **kwargs)	Read data from *dpdata.plugins.deepmd.DeePMDRawFormat* 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_log(file_name, **kwargs)	Read data from *dpdata.plugins.gaussian.GaussianLogFormat* format.

continues on next page

Table 2 – continued from previous page

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_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_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.

continues on next page

Table 2 – continued from previous page

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_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
pick_atom_idx(idx[, nopbc])	Pick atom index
to(fmt, *args, **kwargs)	Dump systems to the specific format.
to_abacus_lcao_md(*args, **kwargs)	Dump data to *dpdata.plugins.abacus.AbacusMDFormat* 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_scf(*args, **kwargs)	Dump data to *dpdata.plugins.abacus.AbacusSCFFormat* format.
to_abacus_scf(*args, **kwargs)	Dump data to *dpdata.plugins.abacus.AbacusSCFFormat* 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_atomconfig(*args, **kwargs)	Dump data to *dpdata.plugins.pwmat.PwmatAtomconfigFormat* format.
to_contcar(*args, **kwargs)	Dump data to *dpdata.plugins.vasp.VASPPoscarFormat* format.

continues on next page

Table 2 – continued from previous page

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_raw(*args, **kwargs)	Dump data to dpdata.plugins.deepmd.DeePMDRawFormat 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_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_outcar(*args, **kwargs)	Dump data to dpdata.plugins.vasp.VASPOutcarFormat format.

continues on next page

Table 2 – continued from previous page

to_poscar(*args, **kwargs)	Dump data to *dpdata.plugins.vasp.VASPPoscarFormat* 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_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 2 – 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.

from_dir	
from_file	
from_fmt_obj	
load_systems_from_file	
predict	
to_fmt_obj	

append(**systems*)

Append systems or MultiSystems to systems

Parameters

system [System] The system to append

check_atom_names(*system*)

Make atom_names in all systems equal, prevent inconsistent atom_types.

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

Read data from dpdata.plugins.abacus.AbacusMDFormat 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_scf(*file_name*, ***kwargs*)

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

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

Read data from dpdata.plugins.abacus.AbacusSCFFormat 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_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_raw(*file_name*, ***kwargs*)
Read data from *dpdata.plugins.deepmd.DeePMDRawFormat* 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_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_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_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_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*)

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(*dp*)

to(*fmt: str*, **args*, ***kwargs*) → *dpdata.system.MultiSystems*

Dump systems to the specific format.

Parameters

fmt [str] format

Returns

MultiSystems self

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

Dump data to *dpdata.plugins.abacus.AbacusMDFormat* 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_scf(**args*, ***kwargs*)

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

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

Dump data to *dpdata.plugins.abacus.AbacusSCFFormat* 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_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_raw(**args*, ***kwargs*)

Dump data to *dpdata.plugins.deepmd.DeePMDRawFormat* 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_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_outcar(**args*, ***kwargs*)

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

to_poscar(**args*, ***kwargs*)
Dump data to *dpdata.plugins.vasp.VASPPoscarFormat* 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_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.**System**(*file_name=None*, *fmt='auto'*, *type_map=None*, *begin=0*, *step=1*, *data=None*, ***kwargs*)

Bases: `monty.json.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

formula Return the formula of this system, like C3H5O2

nopbc

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_type_map(type_map)	Assign atom_names to type_map if type_map is given and different from atom_names.
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_abacus_lcao_md(file_name, **kwargs)	Read data from *dpdata.plugins.abacus.AbacusMDFormat* 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_scf(file_name, **kwargs)	Read data from *dpdata.plugins.abacus.AbacusSCFFormat* format.
from_abacus_scf(file_name, **kwargs)	Read data from *dpdata.plugins.abacus.AbacusSCFFormat* 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_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_raw(file_name, **kwargs)	Read data from *dpdata.plugins.deepmd.DeePMDRawFormat* format.

continues on next page

Table 3 – 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_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_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_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.

continues on next page

Table 3 – continued from previous page

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_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
load(filename)	rebuild System obj.

continues on next page

Table 3 – continued from previous page

map_atom_types([type_map])	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
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(dp)	Predict energies and forces by deepmd-kit.
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.
sub_system(f_idx)	Construct a subsystem from the system
to(fmt, *args, **kwargs)	Dump systems to the specific format.
to_abacus_lcao_md(*args, **kwargs)	Dump data to *dpdata.plugins.abacus.AbacusMDFormat* 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_scf(*args, **kwargs)	Dump data to *dpdata.plugins.abacus.AbacusSCFFormat* format.
to_abacus_scf(*args, **kwargs)	Dump data to *dpdata.plugins.abacus.AbacusSCFFormat* 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_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.

continues on next page

Table 3 – continued from previous page

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_raw(*args, **kwargs)	Dump data to dpdata.plugins.deepmd.DeePMDRawFormat 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_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_outcar(*args, **kwargs)	Dump data to dpdata.plugins.vasp.VASPOutcarFormat format.
to_poscar(*args, **kwargs)	Dump data to dpdata.plugins.vasp.VASPPoscarFormat format.
to_pwmat_atomconfig(*args, **kwargs)	Dump data to dpdata.plugins.pwmat.PwmatAtomconfigFormat format.

continues on next page

Table 3 – 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_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.

continues on next page

Table 3 – continued from previous page

validate_monty(v)	pydantic Validator for MSONable pattern

affine_map	
apply_type_map	
from_fmt	
from_fmt_obj	
replace	
rot_frame_lower_triangular	
rot_lower_triangular	
sort_atom_types	
to_fmt_obj	

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_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

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

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

Read data from *dpdata.plugins.abacus.AbacusMDFormat* 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_scf(*file_name*, ***kwargs*)

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

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

Read data from *dpdata.plugins.abacus.AbacusSCFFormat* 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_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_raw(*file_name*, ***kwargs*)

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

static load(*filename*)

rebuild System obj. from .json or .yaml file

map_atom_types(*type_map=None*)

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 [list] The mapped atom types

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 normalizat** 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`(*dp*)

Predict energies and forces by deepmd-kit.

Parameters

dp [deepmd.DeepPot or str] The deepmd-kit potential class or the filename of the model.

Returns

labeled_sys [LabeledSystem] The labeled system.

`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()

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()

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*) → *dpdata.system.System*

Dump systems to the specific format.

Parameters

fmt [str] format

Returns

System self

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

Dump data to *dpdata.plugins.abacus.AbacusMDFormat* 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_scf(**args*, ***kwargs*)

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

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

Dump data to *dpdata.plugins.abacus.AbacusSCFFormat* 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_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_raw(**args*, ***kwargs*)

Dump data to *dpdata.plugins.deepmd.DeePMDRawFormat* 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_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_outcar(**args*, ***kwargs*)

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

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

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

Notes

Ensure all plugins have been loaded before execuating this function!

dpdata.system.**check_LabeledSystem**(*data*)

dpdata.system.**check_System**(*data*)

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*)

2.1.9 dpdata.unit module

class dpdata.unit.**Conversion**(*unitA*, *unitB*, *check=True*)

Bases: abc.ABC

Methods

set_value	
value	

set_value(*value*)

value()

class dpdata.unit.**EnergyConversion**(*unitA*, *unitB*)

Bases: *dpdata.unit.Conversion*

Methods

set_value	
value	

class dpdata.unit.**ForceConversion**(*unitA*, *unitB*)

Bases: *dpdata.unit.Conversion*

Methods

set_value	
value	

class dpdata.unit.**LengthConversion**(*unitA*, *unitB*)

Bases: *dpdata.unit.Conversion*

Methods

set_value	
value	

class dpdata.unit.**PressureConversion**(*unitA*, *unitB*)

Bases: *dpdata.unit.Conversion*

Methods

set_value	
value	

dpdata.unit.**check_unit**(*unit*)

2.1.10 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

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 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.x.

CHAPTER

THREE

INSTALLATION

One can download the source code of dpdata by

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

then use `setup.py` to install the module

```
cd dpdata
python setup.py install
```

`dpdata` can also by install via pip

```
pip3 install dpdata
```

CHAPTER

FOUR

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

4.1 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'

continues on next page

Table 1 – continued from previous page

Software	format	multi frames	labeled	class	format key
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'
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	output	False	True	LabeledSystem	'cp2k/output'
cp2k	aimd_output	True	True	LabeledSystem	'cp2k/aimd_output'
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'
ABACUS	STRU	False	True	LabeledSystem	'abacus/scf'
ABACUS	cif	True	True	LabeledSystem	'abacus/md'
ase	structure	True	True	MultiSystems	'ase/structure'

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')
```

(continues on next page)

(continued from previous page)

```
# 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')
```

4.2 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

4.3 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`.

4.4 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.

4.5 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)
```

4.6 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')
```

CHAPTER

FIVE

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)
```

5.1 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)
```

5.2 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.

CHAPTER

SIX

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 `setup.py`:

```
entry_points={
    'dpdata.plugins': [
        'random=dpdata_random:RandomFormat'
    ]
},
```

CHAPTER

SEVEN

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

d

INDEX

E

F

G

H

L

M

N

O

P

Q

R

S

T

U

V

X

Z