

---

# **dpdata Documentation**

***Release 0.0.0-rc***

**Han Wang**

**May 15, 2024**



## CONTENTS:

<b>1</b>	<b>Try dpdata online</b>	<b>1</b>
<b>2</b>	<b>Command line interface</b>	<b>3</b>
2.1	Positional Arguments . . . . .	3
2.2	Named Arguments . . . . .	3
<b>3</b>	<b>Supported Formats</b>	<b>5</b>
3.1	ase/structure format . . . . .	6
3.2	ase/traj format . . . . .	8
3.3	abacus/lcao/md format . . . . .	9
3.4	abacus/pw/md format . . . . .	9
3.5	abacus/md format . . . . .	9
3.6	abacus/lcao/relax format . . . . .	10
3.7	abacus/pw/relax format . . . . .	10
3.8	abacus/relax format . . . . .	10
3.9	abacus/lcao/scf format . . . . .	10
3.10	abacus/pw/scf format . . . . .	10
3.11	abacus/scf format . . . . .	10
3.12	stru format . . . . .	11
3.13	abacus/stru format . . . . .	11
3.14	amber/md format . . . . .	13
3.15	cp2k/aimd_output format . . . . .	13
3.16	cp2k/output format . . . . .	14
3.17	dftbplus format . . . . .	14
3.18	deepmd/comp format . . . . .	15
3.19	deepmd/npv format . . . . .	15
3.20	deepmd/hdf5 format . . . . .	18
3.21	deepmd/npv/mixed format . . . . .	20
3.22	deepmd/raw format . . . . .	22
3.23	deepmd format . . . . .	22
3.24	fhi_aims/output format . . . . .	24
3.25	fhi_aims/md format . . . . .	24
3.26	fhi_aims/scf format . . . . .	25
3.27	gaussian/gif format . . . . .	25
3.28	gaussian/log format . . . . .	26
3.29	gaussian/md format . . . . .	27
3.30	gromacs/gro format . . . . .	27
3.31	gro format . . . . .	27
3.32	lammps/dump format . . . . .	28
3.33	dump format . . . . .	28

3.34	lammmps/lmp format . . . . .	29
3.35	lmp format . . . . .	29
3.36	list format . . . . .	30
3.37	mol_file format . . . . .	31
3.38	mol format . . . . .	31
3.39	n2p2 format . . . . .	32
3.40	openmx/md format . . . . .	32
3.41	orca/spout format . . . . .	33
3.42	psi4/inp format . . . . .	34
3.43	psi4/out format . . . . .	35
3.44	pwmat/final.config format . . . . .	36
3.45	pwmat/atom.config format . . . . .	36
3.46	final.config format . . . . .	36
3.47	atom.config format . . . . .	36
3.48	pwmat/output format . . . . .	37
3.49	pwmat/mlmd format . . . . .	37
3.50	pwmat/movement format . . . . .	37
3.51	mlmd format . . . . .	37
3.52	movement format . . . . .	37
3.53	3dmol format . . . . .	38
3.54	pymatgen/computedstructureentry format . . . . .	40
3.55	pymatgen/molecule format . . . . .	40
3.56	pymatgen/structure format . . . . .	41
3.57	qe/pw/scf format . . . . .	42
3.58	qe/cp/traj format . . . . .	42
3.59	quip/gap/xyz_file format . . . . .	43
3.60	quip/gap/xyz format . . . . .	43
3.61	sqm/in format . . . . .	43
3.62	sqm/out format . . . . .	45
3.63	sdf_file format . . . . .	46
3.64	sdf format . . . . .	46
3.65	siesta/aimd_output format . . . . .	46
3.66	siesta/output format . . . . .	47
3.67	vasp/outcar format . . . . .	48
3.68	outcar format . . . . .	48
3.69	vasp/contcar format . . . . .	48
3.70	vasp/poscar format . . . . .	48
3.71	contcar format . . . . .	48
3.72	poscar format . . . . .	48
3.73	vasp/string format . . . . .	50
3.74	vasp/xml format . . . . .	51
3.75	xml format . . . . .	51
3.76	xyz format . . . . .	51
<b>4</b>	<b>Supported Drivers</b>	<b>53</b>
<b>5</b>	<b>Supported Minimizers</b>	<b>55</b>
<b>6</b>	<b>API documentation</b>	<b>57</b>
6.1	dpdata package . . . . .	57
<b>7</b>	<b>Authors</b>	<b>293</b>
<b>8</b>	<b>dpdata</b>	<b>297</b>
8.1	Installation . . . . .	297

8.2	Quick start . . . . .	297
8.3	BondOrderSystem . . . . .	302
8.4	Mixed Type Format . . . . .	303
8.5	Plugins . . . . .	303
<b>9</b>	<b>Indices and tables</b>	<b>305</b>
	<b>Bibliography</b>	<b>307</b>
	<b>Python Module Index</b>	<b>309</b>
	<b>Index</b>	<b>311</b>



**TRY DPDATA ONLINE**





## COMMAND LINE INTERFACE

dpdata: Manipulating multiple atomic simulation data formats

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

### 2.1 Positional Arguments

**from\_file**            read data from a file

### 2.2 Named Arguments

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



## SUPPORTED FORMATS

dpdata supports the following formats:

Table 1: Supported Formats

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

Table 1 – continued from previous

Format	Alias	Support
<i>fhi_aims/scf format</i>	fhi_aims/scf	Labeled:
<i>qe/cp/traj format</i>	qe/cp/traj	System()
<i>qe/pw/scf format</i>	qe/pw/scf	Labeled:
<i>3dmol format</i>	3dmol	System.t
<i>xyz format</i>	xyz	System()
<i>quip/gap/xyz_file format</i>	quip/gap/xyz_file quip/gap/xyz	Labeled:
<i>gromacs/gro format</i>	gromacs/gro gro	System()
<i>n2p2 format</i>	n2p2	Labeled:
<i>dftbplus format</i>	dftbplus	Labeled:
<i>amber/md format</i>	amber/md	System()
<i>sqm/out format</i>	sqm/out	System()
<i>sqm/in format</i>	sqm/in	System.t
<i>openmx/md format</i>	openmx/md	System()

## 3.1 ase/structure format

Class: *ASEStructureFormat*

Format for the Atomic Simulation Environment (ase).

ASE supports parsing a few dozen of data formats. As described in [the documentation](#), many of these formats can be determined automatically. Use the *ase\_fmt* keyword argument to supply the format if automatic detection fails.

### 3.1.1 Conversions

#### Convert from this format to System

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

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

Convert ase.Atoms to a System.

#### Parameters

##### atoms

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

#### Returns

##### System

converted system

### Convert from System to this format

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

```
dpdata.System.to_ase_structure()
```

Convert System to ASE Atom obj.

### Convert from LabeledSystem to this format

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

```
dpdata.LabeledSystem.to_ase_structure()
```

Convert System to ASE Atoms object.

### Convert from this format to LabeledSystem

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

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

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

#### Parameters

##### atoms

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

#### Returns

##### LabeledSystem

converted system

#### Raises

##### RuntimeError

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

### Convert from this format to MultiSystems

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

Convert a ASE supported file to ASE Atoms.

It will finally be converted to MultiSystems.

#### Parameters

##### file\_name

[str] path to file

##### begin

[int, optional] begin frame index

**end**

[int, optional] end frame index

**step**

[int, optional] frame index step

**ase\_fmt**

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

#### Returns

**MultiSystems**

converted system

## 3.2 ase/traj format

Class: *ASETrajFormat*

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

### 3.2.1 Conversions

#### Convert from this format to System

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

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

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

#### Parameters

**file\_name**

[str] ASE's trajectory file

**begin**

[int, optional] begin frame index

**end**

[int, optional] end frame index

**step**

[int, optional] frame index step

#### Returns

**System**

converted system

### Convert from this format to LabeledSystem

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

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

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

#### Parameters

**file\_name**  
[str] ASE's trajectory file

**begin**  
[int, optional] begin frame index

**end**  
[int, optional] end frame index

**step**  
[int, optional] frame index step

#### Returns

**LabeledSystem**  
converted system

## 3.3 abacus/lcao/md format

## 3.4 abacus/pw/md format

## 3.5 abacus/md format

Class: *AbacusMDFormat*

### 3.5.1 Conversions

#### Convert from this format to LabeledSystem

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

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

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

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

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

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

Convert this format to LabeledSystem.

**Returns**

**LabeledSystem**  
converted system

## 3.6 abacus/lcao/relax format

## 3.7 abacus/pw/relax format

## 3.8 abacus/relax format

Class: *AbacusRelaxFormat*

### 3.8.1 Conversions

#### Convert from this format to LabeledSystem

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

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

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

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

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

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

Convert this format to LabeledSystem.

**Returns**

**LabeledSystem**  
converted system

## 3.9 abacus/lcao/scf format

## 3.10 abacus/pw/scf format

## 3.11 abacus/scf format

Class: *AbacusSCFFormat*



### 3.11.1 Conversions

#### Convert from this format to LabeledSystem

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

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

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

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

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

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

Convert this format to LabeledSystem.

#### Returns

**LabeledSystem**  
converted system

## 3.12 stru format

## 3.13 abacus/stru format

Class: *AbacusSTRUFormat*

### 3.13.1 Conversions

#### Convert from this format to System

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

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

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

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

Convert this format to System.

#### Returns

**System**  
converted system

### Convert from System to this format

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

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

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

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

Dump the system into ABACUS STRU format file.

#### Parameters

**file\_name**

[str] The output file name

**frame\_idx**

[int] The index of the frame to dump

**pp\_file**

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

**numerical\_orbital**

[list of string, optional] List of orbital files

**mass**

[list of float, optional] List of atomic masses

**numerical\_descriptor**

[str, optional] numerical descriptor file

### Convert from LabeledSystem to this format

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

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

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

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

Dump the system into ABACUS STRU format file.

#### Parameters

**file\_name**

[str] The output file name

**frame\_idx**

[int] The index of the frame to dump

**pp\_file**

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

**numerical\_orbital**

[list of string, optional] List of orbital files

**mass**

[list of float, optional] List of atomic masses

**numerical\_descriptor**

[str, optional] numerical descriptor file

## 3.14 amber/md format

Class: *AmberMDFormat*

### 3.14.1 Conversions

#### Convert from this format to System

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

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

Convert this format to System.

##### Returns

**System**  
converted system

#### Convert from this format to LabeledSystem

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

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

Convert this format to LabeledSystem.

##### Returns

**LabeledSystem**  
converted system

## 3.15 cp2k/aimd\_output format

Class: *CP2KAIMDOutputFormat*

### 3.15.1 Conversions

#### Convert from this format to LabeledSystem

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

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

Convert this format to LabeledSystem.

**Returns**

**LabeledSystem**  
converted system

## 3.16 cp2k/output format

Class: *CP2KOutputFormat*

### 3.16.1 Conversions

#### Convert from this format to LabeledSystem

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

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

Convert this format to LabeledSystem.

**Returns**

**LabeledSystem**  
converted system

## 3.17 dftbplus format

Class: *DFTBplusFormat*

The DFTBplusFormat class handles files in the DFTB+ format.

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

**Attributes**

**None**

**Methods**

<b>from_labeled_system(file_paths, **kwargs):</b> Reads system information from files.
--

### 3.17.1 Conversions

#### Convert from this format to LabeledSystem

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

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

Reads system information from the given DFTB+ file paths.

##### Parameters

##### **file\_paths**

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

##### Returns

##### **LabeledSystem**

converted system

## 3.18 deepmd/comp format

## 3.19 deepmd/npv format

Class: *DeepPMDCompFormat*

### 3.19.1 Conversions

#### Convert from this format to System

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

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

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

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

Convert this format to System.

##### Returns

##### **System**

converted system

### Convert from System to this format

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

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

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

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

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

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

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

#### Parameters

**file\_name**

[str] The output folder

**set\_size**

[int] The size of each set.

**prec**

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

### Convert from LabeledSystem to this format

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

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

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

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

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

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

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

#### Parameters

**file\_name**

[str] The output folder

**set\_size**

[int] The size of each set.

**prec**

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

### Convert from this format to LabeledSystem

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

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

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

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

Convert this format to LabeledSystem.

#### Returns

##### **LabeledSystem**

converted system

### Convert from this format to MultiSystems

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

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

Convert this format to MultiSystems.

#### Parameters

##### **directory**

[str] directory of systems

#### Returns

##### **MultiSystems**

converted system

### Convert from MultiSystems to this format

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

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

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

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

Convert MultiSystems to this format.

#### Parameters

##### **directory**

[str] directory to save systems

#### Returns

##### **MultiSystems**

this system

## 3.20 deepmd/hdf5 format

Class: `DeePMDHDF5Format`

HDF5 format for DeePMD-kit.

### Examples

Dump a MultiSystems to a HDF5 file:

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

### 3.20.1 Conversions

#### Convert from this format to System

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

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

Convert HDF5 file to System data.

##### Parameters

###### **file\_name**

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

###### **type\_map**

[dict[str]] type map

##### Returns

###### **System**

converted system

##### Raises

###### **TypeError**

file\_name is not str or h5py.Group or h5py.File

#### Convert from System to this format

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

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

Convert System data to HDF5 file.

##### Parameters



**file\_name**

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

**set\_size**

[int, default=5000] set size

**comp\_prec**

[np.dtype] data precision

**Convert from LabeledSystem to this format**

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

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

Convert System data to HDF5 file.

**Parameters****file\_name**

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

**set\_size**

[int, default=5000] set size

**comp\_prec**

[np.dtype] data precision

**Convert from this format to LabeledSystem**

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

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

Convert HDF5 file to LabeledSystem data.

**Parameters****file\_name**

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

**type\_map**

[dict[str]] type map

**Returns****LabeledSystem**

converted system

**Raises****TypeError**

file\_name is not str or h5py.Group or h5py.File

### Convert from this format to MultiSystems

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

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

#### Parameters

##### directory

[str] HDF5 file name

#### Returns

##### MultiSystems

converted system

### Convert from MultiSystems to this format

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

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

Generate HDF5 groups, which will be passed to *to\_system*.

#### Parameters

##### directory

[str] HDF5 file name

#### Returns

##### MultiSystems

this system

## 3.21 deepmd/npz/mixed format

Class: *DeePMDMixedFormat*

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

### Examples

Dump a MultiSystems into a mixed type numpy directory:

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

Load a mixed type data into a MultiSystems:

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

### 3.21.1 Conversions

#### Convert from System to this format

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

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

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

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

##### Parameters

###### **file\_name**

[str] The output folder

###### **set\_size**

[int, default=2000] set size

###### **prec**

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

#### Convert from LabeledSystem to this format

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

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

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

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

##### Parameters

###### **file\_name**

[str] The output folder

###### **set\_size**

[int, default=2000] set size

###### **prec**

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

### Convert from this format to MultiSystems

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

Convert this format to MultiSystems.

#### Parameters

**directory**  
[str] directory of systems

#### Returns

**MultiSystems**  
converted system

### Convert from MultiSystems to this format

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

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

Convert MultiSystems to this format.

#### Parameters

**directory**  
[str] directory to save systems

#### Returns

**MultiSystems**  
this system

## 3.22 deepmd/raw format

## 3.23 deepmd format

Class: *DeePMDRawFormat*

### 3.23.1 Conversions

#### Convert from this format to System

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

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

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

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

Convert this format to System.

#### Returns

**System**

converted system

**Convert from System to this format**`dpdata.System.to(fmt: Literal['deepmd/raw'], file_name)``dpdata.System.to(fmt: Literal['deepmd/raw'], file_name)``dpdata.System.to_deepmd_raw(file_name)``dpdata.System.to_deepmd(file_name)`Dump the system in deepmd raw format to directory *file\_name*.**Convert from LabeledSystem to this format**`dpdata.LabeledSystem.to(fmt: Literal['deepmd/raw'], file_name)``dpdata.LabeledSystem.to(fmt: Literal['deepmd/raw'], file_name)``dpdata.LabeledSystem.to_deepmd_raw(file_name)``dpdata.LabeledSystem.to_deepmd(file_name)`Dump the system in deepmd raw format to directory *file\_name*.**Convert from this format to LabeledSystem**`dpdata.LabeledSystem(file_name, type_map=None, fmt: Literal['deepmd/raw'] = None) →`  
*dpdata.system.LabeledSystem*`dpdata.LabeledSystem(file_name, type_map=None, fmt: Literal['deepmd'] = None) →`  
*dpdata.system.LabeledSystem*`dpdata.LabeledSystem.from_deepmd_raw(file_name, type_map=None) → dpdata.system.LabeledSystem``dpdata.LabeledSystem.from_deepmd(file_name, type_map=None) → dpdata.system.LabeledSystem`

Convert this format to LabeledSystem.

**Returns****LabeledSystem**

converted system

**Convert from this format to MultiSystems**`dpdata.MultiSystems.from_deepmd_raw(directory) → dpdata.system.MultiSystems``dpdata.MultiSystems.from_deepmd(directory) → dpdata.system.MultiSystems`

Convert this format to MultiSystems.

**Parameters**

**directory**  
[str] directory of systems

**Returns**

**MultiSystems**  
converted system

### Convert from MultiSystems to this format

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

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

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

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

Convert MultiSystems to this format.

**Parameters**

**directory**  
[str] directory to save systems

**Returns**

**MultiSystems**  
this system

## 3.24 fhi\_aims/output format

## 3.25 fhi\_aims/md format

Class: *FhiMDFormat*

### 3.25.1 Conversions

#### Convert from this format to LabeledSystem

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

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

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

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

Convert this format to LabeledSystem.

**Returns**

**LabeledSystem**  
converted system

## 3.26 fhi\_aims/scf format

Class: *FhiSCFFormat*

### 3.26.1 Conversions

#### Convert from this format to LabeledSystem

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

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

Convert this format to LabeledSystem.

#### Returns

**LabeledSystem**  
converted system

## 3.27 gaussian/gjf format

Class: *GaussiaGJFFormat*

Gaussian input file.

### 3.27.1 Conversions

#### Convert from this format to System

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

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

Read Gaussian input file.

#### Parameters

**file\_name**  
[str] file name

#### Returns

**System**  
converted system

### Convert from System to this format

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

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

Generate Gaussian input file.

#### Parameters

**file\_name**

[str] file name

### Convert from LabeledSystem to this format

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

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

Generate Gaussian input file.

#### Parameters

**file\_name**

[str] file name

## 3.28 gaussian/log format

Class: *GaussianLogFormat*

### 3.28.1 Conversions

#### Convert from this format to LabeledSystem

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

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

Convert this format to LabeledSystem.

#### Returns

**LabeledSystem**

converted system



## 3.29 gaussian/md format

Class: *GaussianMDFormat*

### 3.29.1 Conversions

#### Convert from this format to LabeledSystem

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

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

Convert this format to LabeledSystem.

#### Returns

**LabeledSystem**  
converted system

## 3.30 gromacs/gro format

## 3.31 gro format

Class: *GromacsGroFormat*

### 3.31.1 Conversions

#### Convert from this format to System

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

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

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

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

Load gromacs .gro file.

#### Parameters

**file\_name**  
[str] The input file name

**format\_atom\_name**  
[bool] Whether to format the atom name

#### Returns

**System**  
converted system

### Convert from System to this format

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

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

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

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

Dump the system in gromacs .gro format.

#### Parameters

**file\_name**

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

**frame\_idx**

[int] The index of the frame to dump

### Convert from LabeledSystem to this format

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

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

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

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

Dump the system in gromacs .gro format.

#### Parameters

**file\_name**

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

**frame\_idx**

[int] The index of the frame to dump

## 3.32 lammmps/dump format

## 3.33 dump format

Class: [\*LAMMPSDumpFormat\*](#)

### 3.33.1 Conversions

#### Convert from this format to System

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

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

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

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

Convert this format to System.

#### Returns

##### System

converted system

## 3.34 lammps/lmp format

### 3.35 lmp format

Class: `LAMMPSLmpFormat`

#### 3.35.1 Conversions

##### Convert from this format to System

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

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

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

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

Convert this format to System.

#### Returns

##### System

converted system

### Convert from System to this format

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

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

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

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

Dump the system in lammps data format.

#### Parameters

**file\_name**

[str] The output file name

**frame\_idx**

[int] The index of the frame to dump

### Convert from LabeledSystem to this format

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

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

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

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

Dump the system in lammps data format.

#### Parameters

**file\_name**

[str] The output file name

**frame\_idx**

[int] The index of the frame to dump

## 3.36 list format

Class: *ListFormat*

### 3.36.1 Conversions

#### Convert from System to this format

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

```
dpdata.System.to_list()
```

Convert system to list, usefull for data collection.

### Convert from LabeledSystem to this format

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

`dpdata.LabeledSystem.to_list()`

Convert system to list, usefull for data collection.

## 3.37 mol\_file format

## 3.38 mol format

Class: *MolFormat*

### 3.38.1 Conversions

#### Convert from this format to BondOrderSystem

`dpdata.BondOrderSystem(file_name, fmt: Literal['mol_file'] = None) →`  
*dpdata.bond\_order\_system.BondOrderSystem*

`dpdata.BondOrderSystem(file_name, fmt: Literal['mol'] = None) →`  
*dpdata.bond\_order\_system.BondOrderSystem*

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

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

Convert this format to BondOrderSystem.

#### Returns

**BondOrderSystem**  
converted system

#### Convert from BondOrderSystem to this format

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

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

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

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

Convert BondOrderSystem to this format.

## 3.39 n2p2 format

Class: *N2P2Format*

n2p2.

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

### 3.39.1 Conversions

#### Convert from this format to LabeledSystem

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

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

Read from n2p2 format.

##### Parameters

**file\_name**

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

##### Returns

**LabeledSystem**

converted system

#### Convert from LabeledSystem to this format

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

`dpdata.LabeledSystem.to_n2p2(file_name)`

Write n2p2 format.

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

##### Parameters

**file\_name**

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

## 3.40 openmx/md format

Class: *OPENMXFormat*

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

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

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

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

### 3.40.1 Conversions

#### Convert from this format to System

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

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

Read from OpenMX output.

##### Parameters

###### file\_name

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

##### Returns

###### System

converted system

#### Convert from this format to LabeledSystem

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

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

Read from OpenMX output.

##### Parameters

###### file\_name

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

##### Returns

###### LabeledSystem

converted system

## 3.41 orca/spout format

Class: `ORCASPOutFormat`

ORCA single point energy output.

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

### 3.41.1 Conversions

#### Convert from this format to LabeledSystem

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

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

Read from ORCA single point energy output.

##### Parameters

**file\_name**  
[str] file name

#### Returns

**LabeledSystem**  
converted system

## 3.42 psi4/inp format

Class: *PSI4InputFormat*

Psi4 input file.

### 3.42.1 Conversions

#### Convert from System to this format

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

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

Write PSI4 input.

#### Parameters

**file\_name**  
[str] file name

**method**  
[str] computational method

**basis**  
[str] basis set; see [https://psicode.org/psi4manual/master/basissets\\_tables.html](https://psicode.org/psi4manual/master/basissets_tables.html)

**charge**  
[int, default=0] charge of system

**multiplicity**  
[int, default=1] multiplicity of system

**frame\_idx**  
[int, default=0] The index of the frame to dump

#### Convert from LabeledSystem to this format

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

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

Write PSI4 input.

#### Parameters



**file\_name**  
[str] file name

**method**  
[str] computational method

**basis**  
[str] basis set; see [https://psicode.org/psi4manual/master/basissets\\_tables.html](https://psicode.org/psi4manual/master/basissets_tables.html)

**charge**  
[int, default=0] charge of system

**multiplicity**  
[int, default=1] multiplicity of system

**frame\_idx**  
[int, default=0] The index of the frame to dump

### 3.43 psi4/out format

Class: *PSI4OutFormat*

Psi4 output.

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

#### 3.43.1 Conversions

##### Convert from this format to LabeledSystem

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

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

Read from Psi4 output.

##### Parameters

**file\_name**  
[str] file name

##### Returns

**LabeledSystem**  
converted system

## 3.44 pwmat/final.config format

## 3.45 pwmat/atom.config format

## 3.46 final.config format

## 3.47 atom.config format

Class: *PwmatAtomconfigFormat*

### 3.47.1 Conversions

#### Convert from this format to System

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

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

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

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

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

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

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

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

Convert this format to System.

#### Returns

##### System

converted system

#### Convert from System to this format

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

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

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

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

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

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

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

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

Dump the system in pwmat atom.config format.

#### Parameters

**file\_name**

[str] The output file name

**frame\_idx**

[int] The index of the frame to dump

### Convert from LabeledSystem to this format

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

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

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

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

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

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

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

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

Dump the system in pwmat atom.config format.

#### Parameters

**file\_name**

[str] The output file name

**frame\_idx**

[int] The index of the frame to dump

## 3.48 pwmat/output format

## 3.49 pwmat/mlmd format

## 3.50 pwmat/movement format

## 3.51 mlmd format

## 3.52 movement format

Class: *PwmatOutputFormat*

### 3.52.1 Conversions

#### Convert from this format to LabeledSystem

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

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

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

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

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

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

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

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

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

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

Convert this format to LabeledSystem.

#### Returns

**LabeledSystem**  
converted system

## 3.53 3dmol format

Class: `Py3DMolFormat`

3DMol format.

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

### 3.53.1 Conversions

#### Convert from System to this format

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

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

Show 3D structure of a frame in jupyter.

#### Parameters

##### f\_idx

[int] frame index to show

##### size

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

##### style

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

#### Examples

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

#### Convert from LabeledSystem to this format

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

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

Show 3D structure of a frame in jupyter.

#### Parameters

##### f\_idx

[int] frame index to show

##### size

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

##### style

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

## Examples

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

## 3.54 pymatgen/computedstructureentry format

Class: *PyMatgenCSEFormat*

### 3.54.1 Conversions

#### Convert from LabeledSystem to this format

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

```
dpdata.LabeledSystem.to_pymatgen_computedstructureentry()
```

Convert System to Pymagen ComputedStructureEntry obj.

## 3.55 pymatgen/molecule format

Class: *PyMatgenMoleculeFormat*

### 3.55.1 Conversions

#### Convert from this format to System

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

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

Convert this format to System.

#### Returns

##### System

converted system

#### Convert from System to this format

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

```
dpdata.System.to_pymatgen_molecule()
```

Convert System to Pymatgen Molecule obj.

### Convert from LabeledSystem to this format

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

```
dpdata.LabeledSystem.to_pymatgen_molecule()
```

Convert System to Pymatgen Molecule obj.

## 3.56 pymatgen/structure format

Class: *PyMatgenStructureFormat*

### 3.56.1 Conversions

#### Convert from this format to System

```
dpdata.System(structure, fmt: Literal['pymatgen/structure'] = None) → dpdata.system.System
```

```
dpdata.System.from_pymatgen_structure(structure) → dpdata.system.System
```

Convert pymatgen.core.Structure to System.

##### Parameters

###### **structure**

[pymatgen.core.Structure] a Pymatgen Structure, containing a structure

##### Returns

###### **System**

converted system

#### Convert from System to this format

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

```
dpdata.System.to_pymatgen_structure()
```

Convert System to Pymatgen Structure obj.

#### Convert from LabeledSystem to this format

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

```
dpdata.LabeledSystem.to_pymatgen_structure()
```

Convert System to Pymatgen Structure obj.

## 3.57 qe/pw/scf format

Class: *QECPPWSCFFormat*

### 3.57.1 Conversions

#### Convert from this format to LabeledSystem

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

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

Convert this format to LabeledSystem.

##### Returns

**LabeledSystem**  
converted system

## 3.58 qe/cp/traj format

Class: *QECPTrajFormat*

### 3.58.1 Conversions

#### Convert from this format to System

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

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

Convert this format to System.

##### Returns

**System**  
converted system

#### Convert from this format to LabeledSystem

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

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

Convert this format to LabeledSystem.

##### Returns

**LabeledSystem**  
converted system



## 3.59 quip/gap/xyz\_file format

## 3.60 quip/gap/xyz format

Class: *QuipGapXYZFormat*

### 3.60.1 Conversions

#### Convert from this format to LabeledSystem

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

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

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

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

Convert this format to LabeledSystem.

#### Returns

**LabeledSystem**  
converted system

#### Convert from this format to MultiSystems

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

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

Convert this format to MultiSystems.

#### Parameters

**directory**  
[str] directory of systems

#### Returns

**MultiSystems**  
converted system

## 3.61 sqm/in format

Class: *SQMInFormat*

### 3.61.1 Conversions

#### Convert from System to this format

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

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

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

##### Parameters

**fname**

[str] output file name

**frame\_idx**

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

##### Other Parameters

**\*\*kwargs**

[dict]

**valid parameters are:**

**qm\_theory**

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

**charge**

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

**maxcyc**

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

**mult**

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

#### Convert from LabeledSystem to this format

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

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

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

##### Parameters

**fname**

[str] output file name

**frame\_idx**

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

##### Other Parameters

**\*\*kwargs**

[dict]

**valid parameters are:**

**qm\_theory**

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

**charge**

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

**maxcyc**

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

**mult**

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

## 3.62 sqm/out format

Class: *SQMOutFormat*

### 3.62.1 Conversions

#### Convert from this format to System

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

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

Read from ambertools sqm.out.

**Returns****System**

converted system

#### Convert from this format to LabeledSystem

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

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

Read from ambertools sqm.out.

**Returns****LabeledSystem**

converted system

## 3.63 sdf\_file format

## 3.64 sdf format

Class: *SdfFormat*

### 3.64.1 Conversions

#### Convert from this format to BondOrderSystem

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

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

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

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

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

#### Returns

**BondOrderSystem**  
converted system

#### Convert from BondOrderSystem to this format

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

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

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

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

Convert BondOrderSystem to this format.

## 3.65 siesta/aimd\_output format

Class: *SiestaAIMDOutputFormat*

### 3.65.1 Conversions

#### Convert from this format to System

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

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

Convert this format to System.

**Returns**

**System**

converted system

### Convert from this format to LabeledSystem

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

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

Convert this format to LabeledSystem.

**Returns**

**LabeledSystem**

converted system

## 3.66 siesta/output format

Class: *SiestaOutputFormat*

### 3.66.1 Conversions

#### Convert from this format to System

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

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

Convert this format to System.

**Returns**

**System**

converted system

#### Convert from this format to LabeledSystem

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

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

Convert this format to LabeledSystem.

**Returns**

**LabeledSystem**

converted system

## 3.67 vasp/outcar format

## 3.68 outcar format

Class: *VASPOutcarFormat*

### 3.68.1 Conversions

Convert from this format to **LabeledSystem**

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

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

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

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

Convert this format to **LabeledSystem**.

**Returns**

**LabeledSystem**  
converted system

## 3.69 vasp/contcar format

## 3.70 vasp/poscar format

## 3.71 contcar format

## 3.72 poscar format

Class: *VASPPoscarFormat*

### 3.72.1 Conversions

#### Convert from this format to System

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

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

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

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

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

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

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

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

Convert this format to System.

#### Returns

##### System

converted system

#### Convert from System to this format

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

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

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

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

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

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

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

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

Dump the system in vasp POSCAR format.

#### Parameters

##### file\_name

[str] The output file name

##### frame\_idx

[int] The index of the frame to dump

### Convert from LabeledSystem to this format

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

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

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

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

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

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

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

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

Dump the system in vasp POSCAR format.

#### Parameters

##### **file\_name**

[str] The output file name

##### **frame\_idx**

[int] The index of the frame to dump

## 3.73 vasp/string format

Class: *VASPStringFormat*

### 3.73.1 Conversions

#### Convert from System to this format

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

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

Dump the system in vasp POSCAR format string.

#### Parameters

##### **frame\_idx**

[int] The index of the frame to dump



### Convert from LabeledSystem to this format

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

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

Dump the system in vasp POSCAR format string.

#### Parameters

**frame\_idx**

[int] The index of the frame to dump

## 3.74 vasp/xml format

### 3.75 xml format

Class: *VASXMLFormat*

#### 3.75.1 Conversions

##### Convert from this format to LabeledSystem

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

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

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

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

Convert this format to LabeledSystem.

#### Returns

**LabeledSystem**

converted system

## 3.76 xyz format

Class: *XYZFormat*

XYZ format.

## Examples

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

### 3.76.1 Conversions

#### Convert from this format to System

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

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

Convert this format to System.

##### Returns

**System**  
converted system

#### Convert from System to this format

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

`dpdata.System.to_xyz(file_name)`

Convert System to this format.

#### Convert from LabeledSystem to this format

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

`dpdata.LabeledSystem.to_xyz(file_name)`

Convert LabeledSystem to this format.

## SUPPORTED DRIVERS

dpdata supports the following drivers:

Table 1: Supported Drivers

Class	Alias
<i>HybridDriver</i>	hybrid
<i>DPDriver</i>	deepmd-kit deepmd dp
<i>ASEDriver</i>	ase
<i>GaussianDriver</i>	gaussian
<i>SQMDriver</i>	sqm



## SUPPORTED MINIMIZERS

dpdata supports the following minimizers:

Table 1: Supported Minimizers

Class	Alias
<i>ASEMinimizer</i>	ase
<i>SQMMinimizer</i>	sqm



## API DOCUMENTATION

### 6.1 dpdata package

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

Bases: *System*

The system with chemical bond and formal charges information.

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

- *d\_example['bonds']*  
[a numpy array of size 4 x 3, and] the first column represents the index of begin atom, the second column represents the index of end atom, the third column represents the bond order:  
1 - single bond, 2 - double bond, 3 - triple bond, 1.5 - aromatic bond
- *d\_example['formal\_charges']* : a numpy array of size 5 x 1

#### Attributes

##### **formula**

Return the formula of this system, like C3H5O2.

##### **formula\_hash**

Return the hash of the formula of this system.

##### **nopbc**

##### **short\_formula**

Return the short formula of this system.

##### **short\_name**

Return the short name of this system (no more than 255 bytes), in the following order: -  
formula - short\_formula - formula\_hash.

##### **uniq\_formula**

Return the uniq\_formula of this system.

## Methods

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

continues on next page



Table 1 – continued from previous page

<code>from_cp2k_output(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.cp2k.CP2KOutputFormat</code> format.
<code>from_deepmd(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.deepmd.DeePMDRawFormat</code> format.
<code>from_deepmd_comp(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.deepmd.DeePMDCompFormat</code> format.
<code>from_deepmd_hdf5(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.deepmd.DeePMDHDF5Format</code> format.
<code>from_deepmd_npy(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.deepmd.DeePMDCompFormat</code> format.
<code>from_deepmd_npy_mixed(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.deepmd.DeePMDMixedFormat</code> format.
<code>from_deepmd_raw(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.deepmd.DeePMDRawFormat</code> format.
<code>from_dftbplus(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.dftbplus.DFTBplusFormat</code> format.
<code>from_dict(d)</code>	<p><b>param d</b> Dict representation.</p>
<code>from_dump(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.lammps.LAMMPSDumpFormat</code> format.
<code>from_fhi_aims_md(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.fhi_aims.FhiMDFormat</code> format.
<code>from_fhi_aims_output(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.fhi_aims.FhiMDFormat</code> format.
<code>from_fhi_aims_scf(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.fhi_aims.FhiSCFFormat</code> format.
<code>from_finalconfig(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.pwmat.PwmatAtomconfigFormat</code> format.
<code>from_gaussian_gjf(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.gaussian.GaussianGJFFormat</code> format.
<code>from_gaussian_log(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.gaussian.GaussianLogFormat</code> format.
<code>from_gaussian_md(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.gaussian.GaussianMDFormat</code> format.
<code>from_gro(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.gromacs.GromacsGroFormat</code> format.
<code>from_gromacs_gro(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.gromacs.GromacsGroFormat</code> format.
<code>from_lammps_dump(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.lammps.LAMMPSDumpFormat</code> format.
<code>from_lammps_lmp(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.lammps.LAMMPSLmpFormat</code> format.
<code>from_list(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.list.ListFormat</code> format.
<code>from_lmp(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.lammps.LAMMPSLmpFormat</code> format.
<code>from_mlmd(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.pwmat.PwmatOutputFormat</code> format.
<code>from_mol(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.rdkit.MolFormat</code> format.

continues on next page

Table 1 – continued from previous page

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

continues on next page

Table 1 – continued from previous page

<code>from_siesta_output(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.siesta.SiestaOutputFormat</code> format.
<code>from_sqm_in(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.amber.SQMInFormat</code> format.
<code>from_sqm_out(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.amber.SQMOutFormat</code> format.
<code>from_stru(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.abacus.AbacusSTRUFormat</code> format.
<code>from_vasp_contcar(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.vasp.VASPPoscarFormat</code> format.
<code>from_vasp_outcar(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.vasp.VASPOutcarFormat</code> format.
<code>from_vasp_poscar(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.vasp.VASPPoscarFormat</code> format.
<code>from_vasp_string(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.vasp.VASPStringFormat</code> format.
<code>from_vasp_xml(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.vasp.VASPXMLFormat</code> format.
<code>from_xml(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.vasp.VASPXMLFormat</code> format.
<code>from_xyz(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.xyz.XYZFormat</code> format.
<code>get_atom_names()</code>	Returns name of atoms.
<code>get_atom_nums()</code>	Returns number of atoms.
<code>get_atom_types()</code>	Returns type of atoms.
<code>get_bond_order(begin_atom_idx, end_atom_idx)</code>	Return the bond order between given atoms.
<code>get_charge()</code>	Return the total formal charge of the molecule.
<code>get_formal_charges()</code>	Return the formal charges on each atom.
<code>get_mol()</code>	Return the rdkit.Mol object.
<code>get_natoms()</code>	Returns total number of atoms in the system.
<code>get_nbonds()</code>	Return the number of bonds.
<code>get_nframes()</code>	Returns number of frames in the system.
<code>get_ntypes()</code>	Returns total number of atom types in the system.
<code>load(filename)</code>	Rebuild System obj.
<code>map_atom_types([type_map])</code>	Map the atom types of the system.
<code>minimize(*args, minimizer, **kwargs)</code>	Minimize the geometry.
<code>perturb(pert_num, cell_pert_fraction, ..., ...)</code>	Perturb each frame in the system randomly.
<code>pick_atom_idx(idx[, nopbc])</code>	Pick atom index.
<code>pick_by_amber_mask(param, maskstr[, ...])</code>	Pick atoms by amber mask.
<code>predict(*args[, driver])</code>	Predict energies and forces by a driver.
<code>register_data_type(*data_type)</code>	Register data type.
<code>remove_atom_names(atom_names)</code>	Remove atom names and all such atoms.
<code>remove_pbc([protect_layer])</code>	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 <i>protect_layer</i> (2) translates the system such that the center-of-geometry of the system locates at the center of the cell.
<code>replicate(ncopy)</code>	Replicate the each frame in the system in 3 dimensions.
<code>shuffle()</code>	Shuffle frames randomly.

continues on next page

Table 1 – continued from previous page

<code>sort_atom_names([type_map])</code>	Sort atom_names of the system and reorder atom_nums and atom_types according to atom_names.
<code>sort_atom_types()</code>	Sort atom types.
<code>sub_system(f_idx)</code>	Construct a subsystem from the system.
<code>to(fmt, *args, **kwargs)</code>	Dump systems to the specific format.
<code>to_3dmol(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.3dmol.Py3DMolFormat</code> format.
<code>to_abacus_lcao_md(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.abacus.AbacusMDFormat</code> format.
<code>to_abacus_lcao_relax(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.abacus.AbacusRelaxFormat</code> format.
<code>to_abacus_lcao_scf(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.abacus.AbacusSCFFormat</code> format.
<code>to_abacus_md(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.abacus.AbacusMDFormat</code> format.
<code>to_abacus_pw_md(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.abacus.AbacusMDFormat</code> format.
<code>to_abacus_pw_relax(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.abacus.AbacusRelaxFormat</code> format.
<code>to_abacus_pw_scf(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.abacus.AbacusSCFFormat</code> format.
<code>to_abacus_relax(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.abacus.AbacusRelaxFormat</code> format.
<code>to_abacus_scf(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.abacus.AbacusSCFFormat</code> format.
<code>to_abacus_stru(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.abacus.AbacusSTRUFormat</code> format.
<code>to_amber_md(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.amber.AmberMDFormat</code> format.
<code>to_ase_structure(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.ase.ASEStructureFormat</code> format.
<code>to_ase_traj(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.ase.ASETrajFormat</code> format.
<code>to_atomconfig(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.pwmat.PwmatAtomconfigFormat</code> format.
<code>to_contcar(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.vasp.VASPPoscarFormat</code> format.
<code>to_cp2k_aimd_output(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.cp2k.CP2KAIMDOutputFormat</code> format.
<code>to_cp2k_output(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.cp2k.CP2KOutputFormat</code> format.
<code>to_deepmd(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.deepmd.DeepMDRawFormat</code> format.
<code>to_deepmd_comp(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.deepmd.DeepMDCompFormat</code> format.
<code>to_deepmd_hdf5(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.deepmd.DeepMDHDF5Format</code> format.
<code>to_deepmd_npy(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.deepmd.DeepMDCompFormat</code> format.
<code>to_deepmd_npy_mixed(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.deepmd.DeepMDMixedFormat</code> format.

continues on next page

Table 1 – continued from previous page

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

continues on next page



Table 1 – continued from previous page

<code>to_psi4_inp(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.psi4.PSI4InputFormat</code> format.
<code>to_psi4_out(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.psi4.PSI4OutputFormat</code> format.
<code>to_pwmat_atomconfig(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.pwmat.PwmatAtomconfigFormat</code> format.
<code>to_pwmat_finalconfig(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.pwmat.PwmatAtomconfigFormat</code> format.
<code>to_pwmat_mlmd(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.pwmat.PwmatOutputFormat</code> format.
<code>to_pwmat_movement(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.pwmat.PwmatOutputFormat</code> format.
<code>to_pwmat_output(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.pwmat.PwmatOutputFormat</code> format.
<code>to_pymatgen_ComputedStructureEntry(*args, ...)</code>	Dump data to <code>dpdata.plugins.pymatgen.PyMatgenCSEFormat</code> format.
<code>to_pymatgen_computedstructureentry(*args, ...)</code>	Dump data to <code>dpdata.plugins.pymatgen.PyMatgenCSEFormat</code> format.
<code>to_pymatgen_molecule(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.pymatgen.PyMatgenMoleculeFormat</code> format.
<code>to_pymatgen_structure(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.pymatgen.PyMatgenStructureFormat</code> format.
<code>to_qe_cp_traj(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.qe.QECPTrajFormat</code> format.
<code>to_qe_pw_scf(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.qe.QECPPWSCFFormat</code> format.
<code>to_quip_gap_xyz(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.xyz.QuipGapXYZFormat</code> format.
<code>to_quip_gap_xyz_file(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.xyz.QuipGapXYZFormat</code> format.
<code>to_sdf(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.rdkit.SdfFormat</code> format.
<code>to_sdf_file(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.rdkit.SdfFormat</code> format.
<code>to_siesta_aimd_output(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.siesta.SiestaAIMDOutputFormat</code> format.
<code>to_siesta_output(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.siesta.SiestaOutputFormat</code> format.
<code>to_sqm_in(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.amber.SQMInFormat</code> format.
<code>to_sqm_out(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.amber.SQMOutFormat</code> format.
<code>to_stru(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.abacus.AbacusSTRUFormat</code> format.
<code>to_vasp_contcar(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.vasp.VASPPoscarFormat</code> format.
<code>to_vasp_outcar(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.vasp.VASPOutcarFormat</code> format.
<code>to_vasp_poscar(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.vasp.VASPPoscarFormat</code> format.
<code>to_vasp_string(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.vasp.VASPStringFormat</code> format.

continues on next page

Table 1 – continued from previous page

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

<code>affine_map</code>
<code>from_fmt</code>
<code>from_fmt_obj</code>
<code>replace</code>
<code>rot_frame_lower_triangular</code>
<code>rot_lower_triangular</code>
<code>to_fmt_obj</code>

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

`copy()`

Returns a copy of the system.

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

`from_rdkit_mol(rdkit_mol)`

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

`get_bond_order(begin_atom_idx, end_atom_idx)`

Return the bond order between given atoms.

`get_charge()`

Return the total formal charge of the molecule.

`get_formal_charges()`

Return the formal charges on each atom.

`get_mol()`

Return the rdkit.Mol object.

`get_nbonds()`

Return the number of bonds.

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

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

Bases: [System](#)

The labeled data System.

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

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

It is noted that

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

### Parameters

#### **file\_name**

[str] The file to load the system

#### **fmt**

[str]

#### **Format of the file, supported formats are**

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



**type\_map**

[list of str] Maps atom type to name. The atom with type *ii* is mapped to *type\_map[ii]*. If not provided the atom names are assigned to 'Type\_1', 'Type\_2', 'Type\_3'...

**begin**

[int] The beginning frame when loading MD trajectory.

**step**

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

**Attributes****formula**

Return the formula of this system, like C3H5O2.

**formula\_hash**

Return the hash of the formula of this system.

**nopbc****short\_formula**

Return the short formula of this system.

**short\_name**

Return the short name of this system (no more than 255 bytes), in the following order: - formula - short\_formula - formula\_hash.

**uniq\_formula**

Return the uniq\_formula of this system.

**Methods**

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

continues on next page

Table 2 – continued from previous page

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

continues on next page

Table 2 – continued from previous page

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

continues on next page

Table 2 – continued from previous page

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

continues on next page

Table 2 – continued from previous page

<code>get_nframes()</code>	Returns number of frames in the system.
<code>get_ntypes()</code>	Returns total number of atom types in the system.
<code>load(filename)</code>	Rebuild System obj.
<code>map_atom_types([type_map])</code>	Map the atom types of the system.
<code>minimize(*args, minimizer, **kwargs)</code>	Minimize the geometry.
<code>perturb(pert_num, cell_pert_fraction, ..., ...)</code>	Perturb each frame in the system randomly.
<code>pick_atom_idx(idx[, nopbc])</code>	Pick atom index.
<code>pick_by_amber_mask(param, maskstr[, ...])</code>	Pick atoms by amber mask.
<code>predict(*args[, driver])</code>	Predict energies and forces by a driver.
<code>register_data_type(*data_type)</code>	Register data type.
<code>remove_atom_names(atom_names)</code>	Remove atom names and all such atoms.
<code>remove_outlier([threshold])</code>	Remove outlier frames from the system.
<code>remove_pbc([protect_layer])</code>	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 <i>protect_layer</i> (2) translates the system such that the center-of-geometry of the system locates at the center of the cell.
<code>replicate(ncopy)</code>	Replicate the each frame in the system in 3 dimensions.
<code>shuffle()</code>	Shuffle frames randomly.
<code>sort_atom_names([type_map])</code>	Sort <code>atom_names</code> of the system and reorder <code>atom_nums</code> and <code>atom_types</code> according to <code>atom_names</code> .
<code>sort_atom_types()</code>	Sort atom types.
<code>sub_system(f_idx)</code>	Construct a subsystem from the system.
<code>to(fmt, *args, **kwargs)</code>	Dump systems to the specific format.
<code>to_3dmol(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.3dmol.Py3DMolFormat</code> format.
<code>to_abacus_lcao_md(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.abacus.AbacusMDFormat</code> format.
<code>to_abacus_lcao_relax(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.abacus.AbacusRelaxFormat</code> format.
<code>to_abacus_lcao_scf(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.abacus.AbacusSCFFormat</code> format.
<code>to_abacus_md(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.abacus.AbacusMDFormat</code> format.
<code>to_abacus_pw_md(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.abacus.AbacusMDFormat</code> format.
<code>to_abacus_pw_relax(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.abacus.AbacusRelaxFormat</code> format.
<code>to_abacus_pw_scf(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.abacus.AbacusSCFFormat</code> format.
<code>to_abacus_relax(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.abacus.AbacusRelaxFormat</code> format.
<code>to_abacus_scf(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.abacus.AbacusSCFFormat</code> format.
<code>to_abacus_stru(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.abacus.AbacusSTRUFormat</code> format.
<code>to_amber_md(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.amber.AmberMDFormat</code> format.

continues on next page

Table 2 – continued from previous page

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

continues on next page



Table 2 – continued from previous page

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

continues on next page

Table 2 – continued from previous page

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

<code>affine_map</code>
<code>affine_map_fv</code>
<code>from_fmt</code>
<code>from_fmt_obj</code>
<code>has_virial</code>
<code>replace</code>
<code>rot_frame_lower_triangular</code>
<code>rot_lower_triangular</code>
<code>to_fmt_obj</code>

```

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

```



**affine\_map\_fv**(*trans*, *f\_idx*)

**correction**(*hl\_sys*)

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

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

#### Parameters

**hl\_sys**  
[LabeledSystem] high-level LabeledSystem

#### Returns

**corrected\_sys: LabeledSystem**  
Corrected LabeledSystem

**from\_3dmol**(*file\_name*, *\*\*kwargs*)

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

**from\_abacus\_lcao\_md**(*file\_name*, *\*\*kwargs*)

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

**from\_abacus\_lcao\_relax**(*file\_name*, *\*\*kwargs*)

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

**from\_abacus\_lcao\_scf**(*file\_name*, *\*\*kwargs*)

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

**from\_abacus\_md**(*file\_name*, *\*\*kwargs*)

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

**from\_abacus\_pw\_md**(*file\_name*, *\*\*kwargs*)

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

**from\_abacus\_pw\_relax**(*file\_name*, *\*\*kwargs*)

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

**from\_abacus\_pw\_scf**(*file\_name*, *\*\*kwargs*)

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

**from\_abacus\_relax**(*file\_name*, *\*\*kwargs*)

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

**from\_abacus\_scf**(*file\_name*, *\*\*kwargs*)

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

**from\_abacus\_stru**(*file\_name*, *\*\*kwargs*)

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

**from\_amber\_md**(*file\_name*, *\*\*kwargs*)

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

**from\_ase\_structure**(*file\_name*, *\*\*kwargs*)

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

**from\_ase\_traj**(*file\_name*, *\*\*kwargs*)

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

**from\_atomconfig**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.pwmat.PwmatAtomconfigFormat` format.

**from\_contcar**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.vasp.VASPPoscarFormat` format.

**from\_cp2k\_aimd\_output**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.cp2k.CP2KAIMDOutputFormat` format.

**from\_cp2k\_output**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.cp2k.CP2KOutputFormat` format.

**from\_deepmd**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.deepmd.DeePMDRawFormat` format.

**from\_deepmd\_comp**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.deepmd.DeePMDCompFormat` format.

**from\_deepmd\_hdf5**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.deepmd.DeePMDHDF5Format` format.

**from\_deepmd\_npy**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.deepmd.DeePMDCompFormat` format.

**from\_deepmd\_npy\_mixed**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.deepmd.DeePMDMixedFormat` format.

**from\_deepmd\_raw**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.deepmd.DeePMDRawFormat` format.

**from\_dftbplus**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.dftbplus.DFTBplusFormat` format.

**from\_dump**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.lammps.LAMMPSDumpFormat` format.

**from\_fhi\_aims\_md**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.fhi_aims.FhiMDFormat` format.

**from\_fhi\_aims\_output**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.fhi_aims.FhiMDFormat` format.

**from\_fhi\_aims\_scf**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.fhi_aims.FhiSCFFormat` format.

**from\_finalconfig**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.pwmat.PwmatAtomconfigFormat` format.

**from\_fmt\_obj**(*fntobj*, *file\_name*, **\*\*kwargs**)

**from\_gaussian\_gjf**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.gaussian.GaussianGJFFormat` format.

**from\_gaussian\_log**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.gaussian.GaussianLogFormat` format.

**from\_gaussian\_md**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.gaussian.GaussianMDFormat` format.

**from\_gro**(file\_name, \*\*kwargs)  
Read data from `dpdata.plugins.gromacs.GromacsGroFormat` format.

**from\_gromacs\_gro**(file\_name, \*\*kwargs)  
Read data from `dpdata.plugins.gromacs.GromacsGroFormat` format.

**from\_lammps\_dump**(file\_name, \*\*kwargs)  
Read data from `dpdata.plugins.lammps.LAMMPSDumpFormat` format.

**from\_lammps\_lmp**(file\_name, \*\*kwargs)  
Read data from `dpdata.plugins.lammps.LAMMPSLmpFormat` format.

**from\_list**(file\_name, \*\*kwargs)  
Read data from `dpdata.plugins.list.ListFormat` format.

**from\_lmp**(file\_name, \*\*kwargs)  
Read data from `dpdata.plugins.lammps.LAMMPSLmpFormat` format.

**from\_mlmd**(file\_name, \*\*kwargs)  
Read data from `dpdata.plugins.pwmat.PwmatOutputFormat` format.

**from\_mol**(file\_name, \*\*kwargs)  
Read data from `dpdata.plugins.rdkit.MolFormat` format.

**from\_mol\_file**(file\_name, \*\*kwargs)  
Read data from `dpdata.plugins.rdkit.MolFormat` format.

**from\_movement**(file\_name, \*\*kwargs)  
Read data from `dpdata.plugins.pwmat.PwmatOutputFormat` format.

**from\_n2p2**(file\_name, \*\*kwargs)  
Read data from `dpdata.plugins.n2p2.N2P2Format` format.

**from\_openmx\_md**(file\_name, \*\*kwargs)  
Read data from `dpdata.plugins.openmx.OPENMXFormat` format.

**from\_orca\_spout**(file\_name, \*\*kwargs)  
Read data from `dpdata.plugins.orca.ORCASPOutFormat` format.

**from\_outcar**(file\_name, \*\*kwargs)  
Read data from `dpdata.plugins.vasp.VASPOutcarFormat` format.

**from\_poscar**(file\_name, \*\*kwargs)  
Read data from `dpdata.plugins.vasp.VASPPoscarFormat` format.

**from\_psi4\_inp**(file\_name, \*\*kwargs)  
Read data from `dpdata.plugins.psi4.PSI4InputFormat` format.

**from\_psi4\_out**(file\_name, \*\*kwargs)  
Read data from `dpdata.plugins.psi4.PSI4OutFormat` format.

**from\_pwmat\_atomconfig**(file\_name, \*\*kwargs)  
Read data from `dpdata.plugins.pwmat.PwmatAtomconfigFormat` format.

**from\_pwmat\_finalconfig**(file\_name, \*\*kwargs)  
Read data from `dpdata.plugins.pwmat.PwmatAtomconfigFormat` format.

**from\_pwmat\_mlmd**(file\_name, \*\*kwargs)  
Read data from `dpdata.plugins.pwmat.PwmatOutputFormat` format.

**from\_pwmat\_movement**(file\_name, \*\*kwargs)  
Read data from `dpdata.plugins.pwmat.PwmatOutputFormat` format.

**from\_pwmat\_output**(file\_name, \*\*kwargs)  
Read data from `dpdata.plugins.pwmat.PwmatOutputFormat` format.

**from\_pymatgen\_computedstructureentry**(file\_name, \*\*kwargs)  
Read data from `dpdata.plugins.pymatgen.PyMatgenCSEFormat` format.

**from\_pymatgen\_molecule**(file\_name, \*\*kwargs)  
Read data from `dpdata.plugins.pymatgen.PyMatgenMoleculeFormat` format.

**from\_pymatgen\_structure**(file\_name, \*\*kwargs)  
Read data from `dpdata.plugins.pymatgen.PyMatgenStructureFormat` format.

**from\_qe\_cp\_traj**(file\_name, \*\*kwargs)  
Read data from `dpdata.plugins.qe.QECPTrajFormat` format.

**from\_qe\_pw\_scf**(file\_name, \*\*kwargs)  
Read data from `dpdata.plugins.qe.QECPPWSCFFormat` format.

**from\_quip\_gap\_xyz**(file\_name, \*\*kwargs)  
Read data from `dpdata.plugins.xyz.QuipGapXYZFormat` format.

**from\_quip\_gap\_xyz\_file**(file\_name, \*\*kwargs)  
Read data from `dpdata.plugins.xyz.QuipGapXYZFormat` format.

**from\_sdf**(file\_name, \*\*kwargs)  
Read data from `dpdata.plugins.rdkit.SdfFormat` format.

**from\_sdf\_file**(file\_name, \*\*kwargs)  
Read data from `dpdata.plugins.rdkit.SdfFormat` format.

**from\_siesta\_aiMD\_output**(file\_name, \*\*kwargs)  
Read data from `dpdata.plugins.siesta.SiestaAIMDOutputFormat` format.

**from\_siesta\_aimd\_output**(file\_name, \*\*kwargs)  
Read data from `dpdata.plugins.siesta.SiestaAIMDOutputFormat` format.

**from\_siesta\_output**(file\_name, \*\*kwargs)  
Read data from `dpdata.plugins.siesta.SiestaOutputFormat` format.

**from\_sqm\_in**(file\_name, \*\*kwargs)  
Read data from `dpdata.plugins.amber.SQMInFormat` format.

**from\_sqm\_out**(file\_name, \*\*kwargs)  
Read data from `dpdata.plugins.amber.SQMOutFormat` format.

**from\_stru**(file\_name, \*\*kwargs)  
Read data from `dpdata.plugins.abacus.AbacusSTRUFormat` format.

**from\_vasp\_contcar**(file\_name, \*\*kwargs)  
Read data from `dpdata.plugins.vasp.VASPPoscarFormat` format.

**from\_vasp\_outcar**(*file\_name*, **\*\*kwargs**)

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

**from\_vasp\_poscar**(*file\_name*, **\*\*kwargs**)

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

**from\_vasp\_string**(*file\_name*, **\*\*kwargs**)

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

**from\_vasp\_xml**(*file\_name*, **\*\*kwargs**)

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

**from\_xml**(*file\_name*, **\*\*kwargs**)

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

**from\_xyz**(*file\_name*, **\*\*kwargs**)

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

**has\_virial**()

**post\_funcs** = <dpdata.plugin.Plugin object>

**remove\_outlier**(*threshold: float = 8.0*) → *LabeledSystem*

Remove outlier frames from the system.

Remove the frames whose energies satisfy the condition

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

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

#### Parameters

##### **threshold**

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

#### Returns

##### **LabeledSystem**

The system without outlier frames.

#### References

[1], [2]

**rot\_frame\_lower\_triangular**(*f\_idx=0*)

**to\_3dmol**(*\*args*, **\*\*kwargs**)

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

**to\_abacus\_lcao\_md**(*\*args*, **\*\*kwargs**)

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

**to\_abacus\_lcao\_relax**(*\*args*, **\*\*kwargs**)

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

**to\_abacus\_lcao\_scf**(*\*args*, **\*\*kwargs**)

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

**to\_abacus\_md**(\*args, \*\*kwargs)

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

**to\_abacus\_pw\_md**(\*args, \*\*kwargs)

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

**to\_abacus\_pw\_relax**(\*args, \*\*kwargs)

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

**to\_abacus\_pw\_scf**(\*args, \*\*kwargs)

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

**to\_abacus\_relax**(\*args, \*\*kwargs)

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

**to\_abacus\_scf**(\*args, \*\*kwargs)

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

**to\_abacus\_stru**(\*args, \*\*kwargs)

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

**to\_amber\_md**(\*args, \*\*kwargs)

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

**to\_ase\_structure**(\*args, \*\*kwargs)

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

**to\_ase\_traj**(\*args, \*\*kwargs)

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

**to\_atomconfig**(\*args, \*\*kwargs)

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

**to\_contcar**(\*args, \*\*kwargs)

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

**to\_cp2k\_aimd\_output**(\*args, \*\*kwargs)

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

**to\_cp2k\_output**(\*args, \*\*kwargs)

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

**to\_deepmd**(\*args, \*\*kwargs)

Dump data to `dpdata.plugins.deepmd.DeepMDRawFormat` format.

**to\_deepmd\_comp**(\*args, \*\*kwargs)

Dump data to `dpdata.plugins.deepmd.DeepMDCompFormat` format.

**to\_deepmd\_hdf5**(\*args, \*\*kwargs)

Dump data to `dpdata.plugins.deepmd.DeepMDHDF5Format` format.

**to\_deepmd\_npy**(\*args, \*\*kwargs)

Dump data to `dpdata.plugins.deepmd.DeepMDCompFormat` format.

**to\_deepmd\_npy\_mixed**(\*args, \*\*kwargs)

Dump data to `dpdata.plugins.deepmd.DeepMDMixedFormat` format.

**to\_deepmd\_raw**(\*args, \*\*kwargs)

Dump data to `dpdata.plugins.deepmd.DeepMDRawFormat` format.

**to\_dftbplus**(\*args, \*\*kwargs)

Dump data to `dpdata.plugins.dftbplus.DFTBplusFormat` format.

**to\_dump**(\*args, \*\*kwargs)

Dump data to `dpdata.plugins.lammps.LAMMPSDumpFormat` format.

**to\_fhi\_aims\_md**(\*args, \*\*kwargs)

Dump data to `dpdata.plugins.fhi_aims.FhiMDFormat` format.

**to\_fhi\_aims\_output**(\*args, \*\*kwargs)

Dump data to `dpdata.plugins.fhi_aims.FhiMDFormat` format.

**to\_fhi\_aims\_scf**(\*args, \*\*kwargs)

Dump data to `dpdata.plugins.fhi_aims.FhiSCFFormat` format.

**to\_finalconfig**(\*args, \*\*kwargs)

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

**to\_fmt\_obj**(fmtobj, \*args, \*\*kwargs)

**to\_gaussian\_gjf**(\*args, \*\*kwargs)

Dump data to `dpdata.plugins.gaussian.GaussianGJFFormat` format.

**to\_gaussian\_log**(\*args, \*\*kwargs)

Dump data to `dpdata.plugins.gaussian.GaussianLogFormat` format.

**to\_gaussian\_md**(\*args, \*\*kwargs)

Dump data to `dpdata.plugins.gaussian.GaussianMDFormat` format.

**to\_gro**(\*args, \*\*kwargs)

Dump data to `dpdata.plugins.gromacs.GromacsGroFormat` format.

**to\_gromacs\_gro**(\*args, \*\*kwargs)

Dump data to `dpdata.plugins.gromacs.GromacsGroFormat` format.

**to\_lammps\_dump**(\*args, \*\*kwargs)

Dump data to `dpdata.plugins.lammps.LAMMPSDumpFormat` format.

**to\_lammps\_lmp**(\*args, \*\*kwargs)

Dump data to `dpdata.plugins.lammps.LAMMPSLmpFormat` format.

**to\_list**(\*args, \*\*kwargs)

Dump data to `dpdata.plugins.list.ListFormat` format.

**to\_lmp**(\*args, \*\*kwargs)

Dump data to `dpdata.plugins.lammps.LAMMPSLmpFormat` format.

**to\_mlmd**(\*args, \*\*kwargs)

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

**to\_mol**(\*args, \*\*kwargs)

Dump data to `dpdata.plugins.rdkit.MolFormat` format.

**to\_mol\_file**(\*args, \*\*kwargs)

Dump data to `dpdata.plugins.rdkit.MolFormat` format.



**to\_movement**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.pwmat.PwmatOutputFormat` format.

**to\_n2p2**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.n2p2.N2P2Format` format.

**to\_openmx\_md**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.openmx.OPENMXFormat` format.

**to\_orca\_spout**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.orca.ORCASPOutFormat` format.

**to\_outcar**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.vasp.VASPOutcarFormat` format.

**to\_poscar**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.vasp.VASPPoscarFormat` format.

**to\_psi4\_inp**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.psi4.PSI4InputFormat` format.

**to\_psi4\_out**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.psi4.PSI4OutFormat` format.

**to\_pwmat\_atomconfig**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.pwmat.PwmatAtomconfigFormat` format.

**to\_pwmat\_finalconfig**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.pwmat.PwmatAtomconfigFormat` format.

**to\_pwmat\_mlmd**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.pwmat.PwmatOutputFormat` format.

**to\_pwmat\_movement**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.pwmat.PwmatOutputFormat` format.

**to\_pwmat\_output**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.pwmat.PwmatOutputFormat` format.

**to\_pymatgen\_ComputedStructureEntry**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.pymatgen.PyMatgenCSEFormat` format.

**to\_pymatgen\_computedstructureentry**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.pymatgen.PyMatgenCSEFormat` format.

**to\_pymatgen\_molecule**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.pymatgen.PyMatgenMoleculeFormat` format.

**to\_pymatgen\_structure**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.pymatgen.PyMatgenStructureFormat` format.

**to\_qe\_cp\_traj**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.qe.QECPTrajFormat` format.

**to\_qe\_pw\_scf**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.qe.QECPPWSCFFormat` format.



```

to_quip_gap_xyz(*args, **kwargs)
    Dump data to dpdata.plugins.xyz.QuipGapXYZFormat format.
to_quip_gap_xyz_file(*args, **kwargs)
    Dump data to dpdata.plugins.xyz.QuipGapXYZFormat format.
to_sdf(*args, **kwargs)
    Dump data to dpdata.plugins.rdkit.SdfFormat format.
to_sdf_file(*args, **kwargs)
    Dump data to dpdata.plugins.rdkit.SdfFormat format.
to_siesta_aimd_output(*args, **kwargs)
    Dump data to dpdata.plugins.siesta.SiestaAIMDOutputFormat format.
to_siesta_output(*args, **kwargs)
    Dump data to dpdata.plugins.siesta.SiestaOutputFormat format.
to_sqm_in(*args, **kwargs)
    Dump data to dpdata.plugins.amber.SQMInFormat format.
to_sqm_out(*args, **kwargs)
    Dump data to dpdata.plugins.amber.SQMOutFormat format.
to_stru(*args, **kwargs)
    Dump data to dpdata.plugins.abacus.AbacusSTRUFormat format.
to_vasp_contcar(*args, **kwargs)
    Dump data to dpdata.plugins.vasp.VASPPoscarFormat format.
to_vasp_outcar(*args, **kwargs)
    Dump data to dpdata.plugins.vasp.VASPOutcarFormat format.
to_vasp_poscar(*args, **kwargs)
    Dump data to dpdata.plugins.vasp.VASPPoscarFormat format.
to_vasp_string(*args, **kwargs)
    Dump data to dpdata.plugins.vasp.VASPStringFormat format.
to_vasp_xml(*args, **kwargs)
    Dump data to dpdata.plugins.vasp.VASPXMLFormat format.
to_xml(*args, **kwargs)
    Dump data to dpdata.plugins.vasp.VASPXMLFormat format.
to_xyz(*args, **kwargs)
    Dump data to dpdata.plugins.xyz.XYZFormat format.
class dpdata.MultiSystems(*systems, type_map=None)
    Bases: object
    A set containing several systems.

```

## Methods

<code>append(*systems)</code>	Append systems or MultiSystems to systems.
<code>check_atom_names(system)</code>	Make atom_names in all systems equal, prevent inconsistent atom_types.
<code>correction(hl_sys)</code>	Get energy and force correction between self (assumed low-level) and a high-level MultiSystems.
<code>from_3dmol(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.3dmol.Py3DMolFormat</code> format.
<code>from_abacus_lcao_md(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.abacus.AbacusMDFormat</code> format.
<code>from_abacus_lcao_relax(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.abacus.AbacusRelaxFormat</code> format.
<code>from_abacus_lcao_scf(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.abacus.AbacusSCFFormat</code> format.
<code>from_abacus_md(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.abacus.AbacusMDFormat</code> format.
<code>from_abacus_pw_md(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.abacus.AbacusMDFormat</code> format.
<code>from_abacus_pw_relax(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.abacus.AbacusRelaxFormat</code> format.
<code>from_abacus_pw_scf(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.abacus.AbacusSCFFormat</code> format.
<code>from_abacus_relax(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.abacus.AbacusRelaxFormat</code> format.
<code>from_abacus_scf(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.abacus.AbacusSCFFormat</code> format.
<code>from_abacus_stru(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.abacus.AbacusSTRUFormat</code> format.
<code>from_amber_md(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.amber.AmberMDFormat</code> format.
<code>from_ase_structure(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.ase.ASEStructureFormat</code> format.
<code>from_ase_traj(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.ase.ASETrajFormat</code> format.
<code>from_atomconfig(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.pwmat.PwmatAtomconfigFormat</code> format.
<code>from_contcar(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.vasp.VASPPoscarFormat</code> format.
<code>from_cp2k_aimd_output(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.cp2k.CP2KAIMDOutputFormat</code> format.
<code>from_cp2k_output(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.cp2k.CP2KOutputFormat</code> format.
<code>from_deepmd(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.deepmd.DeepMDRawFormat</code> format.
<code>from_deepmd_comp(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.deepmd.DeepMDCompFormat</code> format.
<code>from_deepmd_hdf5(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.deepmd.DeepMDHDF5Format</code> format.
<code>from_deepmd_npy(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.deepmd.DeepMDCompFormat</code> format.

continues on next page

Table 3 – continued from previous page

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

continues on next page

Table 3 – continued from previous page

<i>from_psi4_inp</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.psi4.PSI4InputFormat</i> format.
<i>from_psi4_out</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.psi4.PSI4OutputFormat</i> format.
<i>from_pwmat_atomconfig</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.pwmat.PwmatAtomconfigFormat</i> format.
<i>from_pwmat_finalconfig</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.pwmat.PwmatAtomconfigFormat</i> format.
<i>from_pwmat_mlmd</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.pwmat.PwmatOutputFormat</i> format.
<i>from_pwmat_movement</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.pwmat.PwmatOutputFormat</i> format.
<i>from_pwmat_output</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.pwmat.PwmatOutputFormat</i> format.
<i>from_pymatgen_computedstructureentry</i> (...)	Read data from <i>dpdata.plugins.pymatgen.PyMatgenCSEFormat</i> format.
<i>from_pymatgen_molecule</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.pymatgen.PyMatgenMoleculeFormat</i> format.
<i>from_pymatgen_structure</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.pymatgen.PyMatgenStructureFormat</i> format.
<i>from_qe_cp_traj</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.qe.QECPTrajFormat</i> format.
<i>from_qe_pw_scf</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.qe.QECPPWSCFFormat</i> format.
<i>from_quip_gap_xyz</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.xyz.QuipGapXYZFormat</i> format.
<i>from_quip_gap_xyz_file</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.xyz.QuipGapXYZFormat</i> format.
<i>from_sdf</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.rdkit.SdfFormat</i> format.
<i>from_sdf_file</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.rdkit.SdfFormat</i> format.
<i>from_siesta_aiMD_output</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.siesta.SiestaAIMDOutputFormat</i> format.
<i>from_siesta_aimd_output</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.siesta.SiestaAIMDOutputFormat</i> format.
<i>from_siesta_output</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.siesta.SiestaOutputFormat</i> format.
<i>from_sqm_in</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.amber.SQMInFormat</i> format.
<i>from_sqm_out</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.amber.SQMOutFormat</i> format.
<i>from_stru</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.abacus.AbacusSTRUFormat</i> format.
<i>from_vasp_contcar</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.vasp.VASPPoscarFormat</i> format.
<i>from_vasp_outcar</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.vasp.VASPOutcarFormat</i> format.
<i>from_vasp_poscar</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.vasp.VASPPoscarFormat</i> format.
<i>from_vasp_string</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.vasp.VASPStringFormat</i> format.

continues on next page

Table 3 – continued from previous page

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

continues on next page

Table 3 – continued from previous page

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

continues on next page



Table 3 – continued from previous page

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

continues on next page

Table 3 – continued from previous page

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

<code>from_dir</code>
<code>from_file</code>
<code>from_fmt_obj</code>
<code>load_systems_from_file</code>
<code>to_fmt_obj</code>

**append**(\*systems)

Append systems or MultiSystems to systems.

**Parameters****\*systems**

[System] The system to append

**check\_atom\_names**(system)

Make atom\_names in all systems equal, prevent inconsistent atom\_types.

**correction**(hl\_sys: MultiSystems)

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

**Parameters****hl\_sys**

[MultiSystems] high-level MultiSystems

**Returns****corrected\_sys**

[MultiSystems] Corrected MultiSystems



## Notes

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

## Examples

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

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

**from\_3dmol**(*file\_name*, *\*\*kwargs*)

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

**from\_abacus\_lcao\_md**(*file\_name*, *\*\*kwargs*)

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

**from\_abacus\_lcao\_relax**(*file\_name*, *\*\*kwargs*)

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

**from\_abacus\_lcao\_scf**(*file\_name*, *\*\*kwargs*)

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

**from\_abacus\_md**(*file\_name*, *\*\*kwargs*)

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

**from\_abacus\_pw\_md**(*file\_name*, *\*\*kwargs*)

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

**from\_abacus\_pw\_relax**(*file\_name*, *\*\*kwargs*)

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

**from\_abacus\_pw\_scf**(*file\_name*, *\*\*kwargs*)

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

**from\_abacus\_relax**(*file\_name*, *\*\*kwargs*)

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

**from\_abacus\_scf**(*file\_name*, *\*\*kwargs*)

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

**from\_abacus\_stru**(*file\_name*, *\*\*kwargs*)

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

**from\_amber\_md**(*file\_name*, *\*\*kwargs*)

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

**from\_ase\_structure**(*file\_name*, *\*\*kwargs*)

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

**from\_ase\_traj**(*file\_name*, *\*\*kwargs*)

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

**from\_atomconfig**(*file\_name*, *\*\*kwargs*)  
Read data from `dpdata.plugins.pwmat.PwmatAtomconfigFormat` format.

**from\_contcar**(*file\_name*, *\*\*kwargs*)  
Read data from `dpdata.plugins.vasp.VASPPoscarFormat` format.

**from\_cp2k\_aimd\_output**(*file\_name*, *\*\*kwargs*)  
Read data from `dpdata.plugins.cp2k.CP2KAIMDOutputFormat` format.

**from\_cp2k\_output**(*file\_name*, *\*\*kwargs*)  
Read data from `dpdata.plugins.cp2k.CP2KOutputFormat` format.

**from\_deepmd**(*file\_name*, *\*\*kwargs*)  
Read data from `dpdata.plugins.deepmd.DeePMDRawFormat` format.

**from\_deepmd\_comp**(*file\_name*, *\*\*kwargs*)  
Read data from `dpdata.plugins.deepmd.DeePMDCompFormat` format.

**from\_deepmd\_hdf5**(*file\_name*, *\*\*kwargs*)  
Read data from `dpdata.plugins.deepmd.DeePMDHDF5Format` format.

**from\_deepmd\_npy**(*file\_name*, *\*\*kwargs*)  
Read data from `dpdata.plugins.deepmd.DeePMDCompFormat` format.

**from\_deepmd\_npy\_mixed**(*file\_name*, *\*\*kwargs*)  
Read data from `dpdata.plugins.deepmd.DeePMDMixedFormat` format.

**from\_deepmd\_raw**(*file\_name*, *\*\*kwargs*)  
Read data from `dpdata.plugins.deepmd.DeePMDRawFormat` format.

**from\_dftbplus**(*file\_name*, *\*\*kwargs*)  
Read data from `dpdata.plugins.dftbplus.DFTBplusFormat` format.

**classmethod from\_dir**(*dir\_name*, *file\_name*, *fmt='auto'*, *type\_map=None*)

**from\_dump**(*file\_name*, *\*\*kwargs*)  
Read data from `dpdata.plugins.lammps.LAMMPSDumpFormat` format.

**from\_fhi\_aims\_md**(*file\_name*, *\*\*kwargs*)  
Read data from `dpdata.plugins.fhi_aims.FhiMDFormat` format.

**from\_fhi\_aims\_output**(*file\_name*, *\*\*kwargs*)  
Read data from `dpdata.plugins.fhi_aims.FhiMDFormat` format.

**from\_fhi\_aims\_scf**(*file\_name*, *\*\*kwargs*)  
Read data from `dpdata.plugins.fhi_aims.FhiSCFFormat` format.

**classmethod from\_file**(*file\_name*, *fmt*, *\*\*kwargs*)

**from\_finalconfig**(*file\_name*, *\*\*kwargs*)  
Read data from `dpdata.plugins.pwmat.PwmatAtomconfigFormat` format.

**from\_fmt\_obj**(*fmtobj*, *directory*, *labeled=True*, *\*\*kwargs*)

**from\_gaussian\_gjf**(*file\_name*, *\*\*kwargs*)  
Read data from `dpdata.plugins.gaussian.GaussianGJFFormat` format.

**from\_gaussian\_log**(*file\_name*, *\*\*kwargs*)  
Read data from `dpdata.plugins.gaussian.GaussianLogFormat` format.

**from\_gaussian\_md**(file\_name, \*\*kwargs)  
Read data from `dpdata.plugins.gaussian.GaussianMDFormat` format.

**from\_gro**(file\_name, \*\*kwargs)  
Read data from `dpdata.plugins.gromacs.GromacsGroFormat` format.

**from\_gromacs\_gro**(file\_name, \*\*kwargs)  
Read data from `dpdata.plugins.gromacs.GromacsGroFormat` format.

**from\_lammps\_dump**(file\_name, \*\*kwargs)  
Read data from `dpdata.plugins.lammps.LAMMPSDumpFormat` format.

**from\_lammps\_lmp**(file\_name, \*\*kwargs)  
Read data from `dpdata.plugins.lammps.LAMMPSLmpFormat` format.

**from\_list**(file\_name, \*\*kwargs)  
Read data from `dpdata.plugins.list.ListFormat` format.

**from\_lmp**(file\_name, \*\*kwargs)  
Read data from `dpdata.plugins.lammps.LAMMPSLmpFormat` format.

**from\_mlmd**(file\_name, \*\*kwargs)  
Read data from `dpdata.plugins.pwmat.PwmatOutputFormat` format.

**from\_mol**(file\_name, \*\*kwargs)  
Read data from `dpdata.plugins.rdkit.MolFormat` format.

**from\_mol\_file**(file\_name, \*\*kwargs)  
Read data from `dpdata.plugins.rdkit.MolFormat` format.

**from\_movement**(file\_name, \*\*kwargs)  
Read data from `dpdata.plugins.pwmat.PwmatOutputFormat` format.

**from\_n2p2**(file\_name, \*\*kwargs)  
Read data from `dpdata.plugins.n2p2.N2P2Format` format.

**from\_openmx\_md**(file\_name, \*\*kwargs)  
Read data from `dpdata.plugins.openmx.OPENMXFormat` format.

**from\_orca\_spout**(file\_name, \*\*kwargs)  
Read data from `dpdata.plugins.orca.ORBASPOutFormat` format.

**from\_outcar**(file\_name, \*\*kwargs)  
Read data from `dpdata.plugins.vasp.VASPOutcarFormat` format.

**from\_poscar**(file\_name, \*\*kwargs)  
Read data from `dpdata.plugins.vasp.VASPPoscarFormat` format.

**from\_psi4\_inp**(file\_name, \*\*kwargs)  
Read data from `dpdata.plugins.psi4.PSI4InputFormat` format.

**from\_psi4\_out**(file\_name, \*\*kwargs)  
Read data from `dpdata.plugins.psi4.PSI4OutFormat` format.

**from\_pwmat\_atomconfig**(file\_name, \*\*kwargs)  
Read data from `dpdata.plugins.pwmat.PwmatAtomconfigFormat` format.

**from\_pwmat\_finalconfig**(file\_name, \*\*kwargs)  
Read data from `dpdata.plugins.pwmat.PwmatAtomconfigFormat` format.

**from\_pwmat\_mlmd**(file\_name, \*\*kwargs)  
Read data from `dpdata.plugins.pwmat.PwmatOutputFormat` format.

**from\_pwmat\_movement**(file\_name, \*\*kwargs)  
Read data from `dpdata.plugins.pwmat.PwmatOutputFormat` format.

**from\_pwmat\_output**(file\_name, \*\*kwargs)  
Read data from `dpdata.plugins.pwmat.PwmatOutputFormat` format.

**from\_pymatgen\_computedstructureentry**(file\_name, \*\*kwargs)  
Read data from `dpdata.plugins.pymatgen.PyMatgenCSEFormat` format.

**from\_pymatgen\_molecule**(file\_name, \*\*kwargs)  
Read data from `dpdata.plugins.pymatgen.PyMatgenMoleculeFormat` format.

**from\_pymatgen\_structure**(file\_name, \*\*kwargs)  
Read data from `dpdata.plugins.pymatgen.PyMatgenStructureFormat` format.

**from\_qe\_cp\_traj**(file\_name, \*\*kwargs)  
Read data from `dpdata.plugins.qe.QECPTrajFormat` format.

**from\_qe\_pw\_scf**(file\_name, \*\*kwargs)  
Read data from `dpdata.plugins.qe.QECPPWSCFFormat` format.

**from\_quip\_gap\_xyz**(file\_name, \*\*kwargs)  
Read data from `dpdata.plugins.xyz.QuipGapXYZFormat` format.

**from\_quip\_gap\_xyz\_file**(file\_name, \*\*kwargs)  
Read data from `dpdata.plugins.xyz.QuipGapXYZFormat` format.

**from\_sdf**(file\_name, \*\*kwargs)  
Read data from `dpdata.plugins.rdkit.SdfFormat` format.

**from\_sdf\_file**(file\_name, \*\*kwargs)  
Read data from `dpdata.plugins.rdkit.SdfFormat` format.

**from\_siesta\_aiMD\_output**(file\_name, \*\*kwargs)  
Read data from `dpdata.plugins.siesta.SiestaAIMDOutputFormat` format.

**from\_siesta\_aimd\_output**(file\_name, \*\*kwargs)  
Read data from `dpdata.plugins.siesta.SiestaAIMDOutputFormat` format.

**from\_siesta\_output**(file\_name, \*\*kwargs)  
Read data from `dpdata.plugins.siesta.SiestaOutputFormat` format.

**from\_sqm\_in**(file\_name, \*\*kwargs)  
Read data from `dpdata.plugins.amber.SQMInFormat` format.

**from\_sqm\_out**(file\_name, \*\*kwargs)  
Read data from `dpdata.plugins.amber.SQMOutFormat` format.

**from\_stru**(file\_name, \*\*kwargs)  
Read data from `dpdata.plugins.abacus.AbacusSTRUFormat` format.

**from\_vasp\_contcar**(*file\_name*, **\*\*kwargs**)

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

**from\_vasp\_outcar**(*file\_name*, **\*\*kwargs**)

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

**from\_vasp\_poscar**(*file\_name*, **\*\*kwargs**)

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

**from\_vasp\_string**(*file\_name*, **\*\*kwargs**)

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

**from\_vasp\_xml**(*file\_name*, **\*\*kwargs**)

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

**from\_xml**(*file\_name*, **\*\*kwargs**)

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

**from\_xyz**(*file\_name*, **\*\*kwargs**)

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

**get\_nframes**()

Returns number of frames in all systems.

**load\_systems\_from\_file**(*file\_name=None*, *fmt=None*, **\*\*kwargs**)

**minimize**(\*args: *Any*, minimizer: *str* | `Minimizer`, **\*\*kwargs**: *Any*) → *MultiSystems*

Minimize geometry by a minimizer.

#### Parameters

**\*args**

[iterable] Arguments passing to the minimizer

**minimizer**

[str or `Minimizer`] The assigned minimizer

**\*\*kwargs**

[dict] Other arguments passing to the minimizer

#### Returns

**MultiSystems**

A new labeled MultiSystems.

### Examples

Minimize a system using ASE BFGS along with a DP driver:

```
>>> from dpdata.driver import Driver
>>> from ase.optimize import BFGS
>>> driver = Driver.get_driver("dp")("some_model.pb")
>>> some_system.minimize(minimizer="ase", driver=driver, optimizer=BFGS,
↪ fmax=1e-5)
```

**pick\_atom\_idx**(*idx*, *nopbc=None*)

Pick atom index.

#### Parameters

**idx**

[int or list or slice] atom index

**nopbc**

[Boolean (default: None)] If nopbc is True or False, set nopbc

**Returns****new\_sys: MultiSystems**

new system

**predict** (\*args: *Any*, driver='dp', \*\*kwargs: *Any*) → *MultiSystems*

Predict energies and forces by a driver.

**Parameters****\*args**

[iterable] Arguments passing to the driver

**driver**

[str, default=dp] The assigned driver. For compatibility, default is dp

**\*\*kwargs**

[dict] Other arguments passing to the driver

**Returns****MultiSystems**

A new labeled MultiSystems.

**to**(fmt: *str*, \*args, \*\*kwargs) → *MultiSystems*

Dump systems to the specific format.

**Parameters****fmt**

[str] format

**\*args**

[list] arguments

**\*\*kwargs**

[dict] keyword arguments

**Returns****MultiSystems**

self

**to\_3dmol** (\*args, \*\*kwargs)Dump data to `dpdata.plugins.3dmol.Py3DMolFormat` format.**to\_abacus\_lcao\_md** (\*args, \*\*kwargs)Dump data to `dpdata.plugins.abacus.AbacusMDFormat` format.**to\_abacus\_lcao\_relax** (\*args, \*\*kwargs)Dump data to `dpdata.plugins.abacus.AbacusRelaxFormat` format.**to\_abacus\_lcao\_scf** (\*args, \*\*kwargs)Dump data to `dpdata.plugins.abacus.AbacusSCFFormat` format.**to\_abacus\_md** (\*args, \*\*kwargs)Dump data to `dpdata.plugins.abacus.AbacusMDFormat` format.

**to\_abacus\_pw\_md**(\*args, \*\*kwargs)  
 Dump data to `dpdata.plugins.abacus.AbacusMDFormat` format.

**to\_abacus\_pw\_relax**(\*args, \*\*kwargs)  
 Dump data to `dpdata.plugins.abacus.AbacusRelaxFormat` format.

**to\_abacus\_pw\_scf**(\*args, \*\*kwargs)  
 Dump data to `dpdata.plugins.abacus.AbacusSCFFormat` format.

**to\_abacus\_relax**(\*args, \*\*kwargs)  
 Dump data to `dpdata.plugins.abacus.AbacusRelaxFormat` format.

**to\_abacus\_scf**(\*args, \*\*kwargs)  
 Dump data to `dpdata.plugins.abacus.AbacusSCFFormat` format.

**to\_abacus\_stru**(\*args, \*\*kwargs)  
 Dump data to `dpdata.plugins.abacus.AbacusSTRUFormat` format.

**to\_amber\_md**(\*args, \*\*kwargs)  
 Dump data to `dpdata.plugins.amber.AmberMDFormat` format.

**to\_ase\_structure**(\*args, \*\*kwargs)  
 Dump data to `dpdata.plugins.ase.ASEStructureFormat` format.

**to\_ase\_traj**(\*args, \*\*kwargs)  
 Dump data to `dpdata.plugins.ase.ASETrajFormat` format.

**to\_atomconfig**(\*args, \*\*kwargs)  
 Dump data to `dpdata.plugins.pwmat.PwmatAtomconfigFormat` format.

**to\_contcar**(\*args, \*\*kwargs)  
 Dump data to `dpdata.plugins.vasp.VASPPoscarFormat` format.

**to\_cp2k\_aimd\_output**(\*args, \*\*kwargs)  
 Dump data to `dpdata.plugins.cp2k.CP2KAIMDOutputFormat` format.

**to\_cp2k\_output**(\*args, \*\*kwargs)  
 Dump data to `dpdata.plugins.cp2k.CP2KOutputFormat` format.

**to\_deepmd**(\*args, \*\*kwargs)  
 Dump data to `dpdata.plugins.deepmd.DeepMDRawFormat` format.

**to\_deepmd\_comp**(\*args, \*\*kwargs)  
 Dump data to `dpdata.plugins.deepmd.DeepMDCompFormat` format.

**to\_deepmd\_hdf5**(\*args, \*\*kwargs)  
 Dump data to `dpdata.plugins.deepmd.DeepMDHDF5Format` format.

**to\_deepmd\_npy**(\*args, \*\*kwargs)  
 Dump data to `dpdata.plugins.deepmd.DeepMDCompFormat` format.

**to\_deepmd\_npy\_mixed**(\*args, \*\*kwargs)  
 Dump data to `dpdata.plugins.deepmd.DeepMDMixedFormat` format.

**to\_deepmd\_raw**(\*args, \*\*kwargs)  
 Dump data to `dpdata.plugins.deepmd.DeepMDRawFormat` format.

**to\_dftbplus**(\*args, \*\*kwargs)

Dump data to `dpdata.plugins.dftbplus.DFTBplusFormat` format.

**to\_dump**(\*args, \*\*kwargs)

Dump data to `dpdata.plugins.lammps.LAMMPSDumpFormat` format.

**to\_fhi\_aims\_md**(\*args, \*\*kwargs)

Dump data to `dpdata.plugins.fhi_aims.FhiMDFormat` format.

**to\_fhi\_aims\_output**(\*args, \*\*kwargs)

Dump data to `dpdata.plugins.fhi_aims.FhiMDFormat` format.

**to\_fhi\_aims\_scf**(\*args, \*\*kwargs)

Dump data to `dpdata.plugins.fhi_aims.FhiSCFFormat` format.

**to\_finalconfig**(\*args, \*\*kwargs)

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

**to\_fmt\_obj**(fmtobj, directory, \*args, \*\*kwargs)

**to\_gaussian\_gjf**(\*args, \*\*kwargs)

Dump data to `dpdata.plugins.gaussian.GaussianGJFFormat` format.

**to\_gaussian\_log**(\*args, \*\*kwargs)

Dump data to `dpdata.plugins.gaussian.GaussianLogFormat` format.

**to\_gaussian\_md**(\*args, \*\*kwargs)

Dump data to `dpdata.plugins.gaussian.GaussianMDFormat` format.

**to\_gro**(\*args, \*\*kwargs)

Dump data to `dpdata.plugins.gromacs.GromacsGroFormat` format.

**to\_gromacs\_gro**(\*args, \*\*kwargs)

Dump data to `dpdata.plugins.gromacs.GromacsGroFormat` format.

**to\_lammps\_dump**(\*args, \*\*kwargs)

Dump data to `dpdata.plugins.lammps.LAMMPSDumpFormat` format.

**to\_lammps\_lmp**(\*args, \*\*kwargs)

Dump data to `dpdata.plugins.lammps.LAMMPSLmpFormat` format.

**to\_list**(\*args, \*\*kwargs)

Dump data to `dpdata.plugins.list.ListFormat` format.

**to\_lmp**(\*args, \*\*kwargs)

Dump data to `dpdata.plugins.lammps.LAMMPSLmpFormat` format.

**to\_mlmd**(\*args, \*\*kwargs)

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

**to\_mol**(\*args, \*\*kwargs)

Dump data to `dpdata.plugins.rdkit.MolFormat` format.

**to\_mol\_file**(\*args, \*\*kwargs)

Dump data to `dpdata.plugins.rdkit.MolFormat` format.

**to\_movement**(\*args, \*\*kwargs)

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



**to\_n2p2**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.n2p2.N2P2Format` format.

**to\_openmx\_md**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.openmx.OPENMXFormat` format.

**to\_orca\_spout**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.orca.ORBASPOutFormat` format.

**to\_outcar**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.vasp.VASPOutcarFormat` format.

**to\_poscar**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.vasp.VASPPoscarFormat` format.

**to\_psi4\_inp**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.psi4.PSI4InputFormat` format.

**to\_psi4\_out**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.psi4.PSI4OutFormat` format.

**to\_pwmat\_atomconfig**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.pwmat.PwmatAtomconfigFormat` format.

**to\_pwmat\_finalconfig**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.pwmat.PwmatAtomconfigFormat` format.

**to\_pwmat\_mlmd**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.pwmat.PwmatOutputFormat` format.

**to\_pwmat\_movement**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.pwmat.PwmatOutputFormat` format.

**to\_pwmat\_output**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.pwmat.PwmatOutputFormat` format.

**to\_pymatgen\_ComputedStructureEntry**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.pymatgen.PyMatgenCSEFormat` format.

**to\_pymatgen\_computedstructureentry**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.pymatgen.PyMatgenCSEFormat` format.

**to\_pymatgen\_molecule**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.pymatgen.PyMatgenMoleculeFormat` format.

**to\_pymatgen\_structure**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.pymatgen.PyMatgenStructureFormat` format.

**to\_qe\_cp\_traj**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.qe.QECPTrajFormat` format.

**to\_qe\_pw\_scf**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.qe.QECPWSCFFormat` format.

**to\_quip\_gap\_xyz**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.xyz.QuipGapXYZFormat` format.

**to\_quip\_gap\_xyz\_file**(\*args, \*\*kwargs)

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

**to\_sdf**(\*args, \*\*kwargs)

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

**to\_sdf\_file**(\*args, \*\*kwargs)

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

**to\_siesta\_aimd\_output**(\*args, \*\*kwargs)

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

**to\_siesta\_output**(\*args, \*\*kwargs)

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

**to\_sqm\_in**(\*args, \*\*kwargs)

Dump data to `dpdata.plugins.amber.SQMInFormat` format.

**to\_sqm\_out**(\*args, \*\*kwargs)

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

**to\_stru**(\*args, \*\*kwargs)

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

**to\_vasp\_contcar**(\*args, \*\*kwargs)

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

**to\_vasp\_outcar**(\*args, \*\*kwargs)

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

**to\_vasp\_poscar**(\*args, \*\*kwargs)

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

**to\_vasp\_string**(\*args, \*\*kwargs)

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

**to\_vasp\_xml**(\*args, \*\*kwargs)

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

**to\_xml**(\*args, \*\*kwargs)

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

**to\_xyz**(\*args, \*\*kwargs)

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

**train\_test\_split**(test\_size: float | int, seed: int | None = None) → Tuple[MultiSystems, MultiSystems, Dict[str, ndarray]]

Split systems into random train and test subsets.

#### Parameters

##### test\_size

[float or int] If float, should be between 0.0 and 1.0 and represent the proportion of the dataset to include in the test split. If int, represents the absolute number of test samples.

##### seed

[int, default=None] Random seed

#### Returns

**MultiSystems**

The training set

**MultiSystems**

The testing set

**Dict[str, np.ndarray]**

The bool array of training and testing sets for each system. False for training set and True for testing set.

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

Bases: `MSONable`

The data System.

A data System (a concept used by [deepmd-kit](#)) contains frames (e.g. produced by an MD simulation) that has the same number of atoms of the same type. The order of the atoms should be consistent among the frames in one System.

**For example, a water system named *d\_example* has two molecules. The properties can be accessed by**

- `d_example['atom_nums']` : [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 (lammmps cell tensor convention)

**Attributes****DTYPES**

[tuple[DataType]] data types of this class

**Methods**

<code>add_atom_names(atom_names)</code>	Add atom_names that do not exist.
<code>append(system)</code>	Append a system to this system.
<code>apply_pbc()</code>	Append periodic boundary condition.
<code>apply_type_map(type_map)</code>	Customize the element symbol order and it should maintain order consistency in dpn or deepmd-kit.
<code>as_dict()</code>	Returns data dict of System instance.
<code>check_data()</code>	Check if data is correct.

continues on next page

Table 4 – continued from previous page

<code>check_type_map(type_map)</code>	Assign <code>atom_names</code> to <code>type_map</code> if <code>type_map</code> is given and different from <code>atom_names</code> .
<code>convert_to_mixed_type([type_map])</code>	Convert the data dict to mixed type format structure, in order to append systems with different formula but the same number of atoms.
<code>copy()</code>	Returns a copy of the system.
<code>dump(filename[, indent])</code>	Dump .json or .yaml file.
<code>extend(systems)</code>	Extend a system list to this system.
<code>from_3dmol(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.3dmol.Py3DMolFormat</code> format.
<code>from_abacus_lcao_md(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.abacus.AbacusMDFormat</code> format.
<code>from_abacus_lcao_relax(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.abacus.AbacusRelaxFormat</code> format.
<code>from_abacus_lcao_scf(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.abacus.AbacusSCFFormat</code> format.
<code>from_abacus_md(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.abacus.AbacusMDFormat</code> format.
<code>from_abacus_pw_md(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.abacus.AbacusMDFormat</code> format.
<code>from_abacus_pw_relax(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.abacus.AbacusRelaxFormat</code> format.
<code>from_abacus_pw_scf(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.abacus.AbacusSCFFormat</code> format.
<code>from_abacus_relax(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.abacus.AbacusRelaxFormat</code> format.
<code>from_abacus_scf(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.abacus.AbacusSCFFormat</code> format.
<code>from_abacus_stru(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.abacus.AbacusSTRUFormat</code> format.
<code>from_amber_md(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.amber.AmberMDFormat</code> format.
<code>from_ase_structure(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.ase.ASEStructureFormat</code> format.
<code>from_ase_traj(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.ase.ASETrajFormat</code> format.
<code>from_atomconfig(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.pwmat.PwmatAtomconfigFormat</code> format.
<code>from_contcar(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.vasp.VASPPoscarFormat</code> format.
<code>from_cp2k_aimd_output(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.cp2k.CP2KAIMDOutputFormat</code> format.
<code>from_cp2k_output(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.cp2k.CP2KOutputFormat</code> format.
<code>from_deepmd(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.deepmd.DeepMDRawFormat</code> format.
<code>from_deepmd_comp(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.deepmd.DeepMDCompFormat</code> format.
<code>from_deepmd_hdf5(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.deepmd.DeepMDHDF5Format</code> format.
<code>from_deepmd_npy(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.deepmd.DeepMDCompFormat</code> format.

continues on next page

Table 4 – continued from previous page

<code>from_deepmd_npy_mixed(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.deepmd.DeePMDMixedFormat</code> format.
<code>from_deepmd_raw(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.deepmd.DeePMDRawFormat</code> format.
<code>from_dftbplus(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.dftbplus.DFTBplusFormat</code> format.
<code>from_dict(d)</code>	<p><b>param d</b> Dict representation.</p>
<code>from_dump(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.lammps.LAMMPSDumpFormat</code> format.
<code>from_fhi_aims_md(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.fhi_aims.FhiMDFormat</code> format.
<code>from_fhi_aims_output(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.fhi_aims.FhiMDFormat</code> format.
<code>from_fhi_aims_scf(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.fhi_aims.FhiSCFFormat</code> format.
<code>from_finalconfig(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.pwmat.PwmatAtomconfigFormat</code> format.
<code>from_gaussian_gjf(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.gaussian.GaussianGJFFormat</code> format.
<code>from_gaussian_log(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.gaussian.GaussianLogFormat</code> format.
<code>from_gaussian_md(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.gaussian.GaussianMDFormat</code> format.
<code>from_gro(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.gromacs.GromacsGroFormat</code> format.
<code>from_gromacs_gro(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.gromacs.GromacsGroFormat</code> format.
<code>from_lammps_dump(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.lammps.LAMMPSDumpFormat</code> format.
<code>from_lammps_lmp(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.lammps.LAMMPSLmpFormat</code> format.
<code>from_list(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.list.ListFormat</code> format.
<code>from_lmp(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.lammps.LAMMPSLmpFormat</code> format.
<code>from_mlmd(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.pwmat.PwmatOutputFormat</code> format.
<code>from_mol(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.rdkit.MolFormat</code> format.
<code>from_mol_file(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.rdkit.MolFormat</code> format.
<code>from_movement(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.pwmat.PwmatOutputFormat</code> format.
<code>from_n2p2(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.n2p2.N2P2Format</code> format.
<code>from_openmx_md(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.openmx.OPENMXFormat</code> format.
<code>from_orca_spout(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.orca.OCASPOutFormat</code> format.

continues on next page

Table 4 – continued from previous page

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

continues on next page



Table 4 – continued from previous page

<i>from_vasp_poscar</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.vasp.VASPPoscarFormat</i> format.
<i>from_vasp_string</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.vasp.VASPStringFormat</i> format.
<i>from_vasp_xml</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.vasp.VASPXMLFormat</i> format.
<i>from_xml</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.vasp.VASPXMLFormat</i> format.
<i>from_xyz</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.xyz.XYZFormat</i> format.
<i>get_atom_names</i> ()	Returns name of atoms.
<i>get_atom_numbs</i> ()	Returns number of atoms.
<i>get_atom_types</i> ()	Returns type of atoms.
<i>get_natoms</i> ()	Returns total number of atoms in the system.
<i>get_nframes</i> ()	Returns number of frames in the system.
<i>get_ntypes</i> ()	Returns total number of atom types in the system.
<i>load</i> (filename)	Rebuild System obj.
<i>map_atom_types</i> ([type_map])	Map the atom types of the system.
<i>minimize</i> (*args, minimizer, **kwargs)	Minimize the geometry.
<i>perturb</i> (pert_num, cell_pert_fraction, ..., ...)	Perturb each frame in the system randomly.
<i>pick_atom_idx</i> (idx[, nopbc])	Pick atom index.
<i>pick_by_amber_mask</i> (param, maskstr[, ...])	Pick atoms by amber mask.
<i>predict</i> (*args[, driver])	Predict energies and forces by a driver.
<i>register_data_type</i> (*data_type)	Register data type.
<i>remove_atom_names</i> (atom_names)	Remove atom names and all such atoms.
<i>remove_pbc</i> ([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 <i>protect_layer</i> (2) translates the system such that the center-of-geometry of the system locates at the center of the cell.
<i>replicate</i> (ncopy)	Replicate the each frame in the system in 3 dimensions.
<i>shuffle</i> ()	Shuffle frames randomly.
<i>sort_atom_names</i> ([type_map])	Sort atom_names of the system and reorder atom_numbs and atom_types according to atom_names.
<i>sort_atom_types</i> ()	Sort atom types.
<i>sub_system</i> (f_idx)	Construct a subsystem from the system.
<i>to</i> (fmt, *args, **kwargs)	Dump systems to the specific format.
<i>to_3dmol</i> (*args, **kwargs)	Dump data to <i>dpdata.plugins.3dmol.Py3DMolFormat</i> format.
<i>to_abacus_lcao_md</i> (*args, **kwargs)	Dump data to <i>dpdata.plugins.abacus.AbacusMDFormat</i> format.
<i>to_abacus_lcao_relax</i> (*args, **kwargs)	Dump data to <i>dpdata.plugins.abacus.AbacusRelaxFormat</i> format.
<i>to_abacus_lcao_scf</i> (*args, **kwargs)	Dump data to <i>dpdata.plugins.abacus.AbacusSCFFormat</i> format.
<i>to_abacus_md</i> (*args, **kwargs)	Dump data to <i>dpdata.plugins.abacus.AbacusMDFormat</i> format.
<i>to_abacus_pw_md</i> (*args, **kwargs)	Dump data to <i>dpdata.plugins.abacus.AbacusMDFormat</i> format.

continues on next page

Table 4 – continued from previous page

<code>to_abacus_pw_relax(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.abacus.AbacusRelaxFormat</code> format.
<code>to_abacus_pw_scf(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.abacus.AbacusSCFFormat</code> format.
<code>to_abacus_relax(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.abacus.AbacusRelaxFormat</code> format.
<code>to_abacus_scf(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.abacus.AbacusSCFFormat</code> format.
<code>to_abacus_stru(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.abacus.AbacusSTRUFormat</code> format.
<code>to_amber_md(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.amber.AmberMDFormat</code> format.
<code>to_ase_structure(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.ase.ASEStructureFormat</code> format.
<code>to_ase_traj(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.ase.ASETrajFormat</code> format.
<code>to_atomconfig(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.pwmat.PwmatAtomconfigFormat</code> format.
<code>to_contcar(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.vasp.VASPPoscarFormat</code> format.
<code>to_cp2k_aimd_output(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.cp2k.CP2KAIMDOutputFormat</code> format.
<code>to_cp2k_output(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.cp2k.CP2KOutputFormat</code> format.
<code>to_deepmd(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.deepmd.DeePMDRawFormat</code> format.
<code>to_deepmd_comp(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.deepmd.DeePMDCompFormat</code> format.
<code>to_deepmd_hdf5(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.deepmd.DeePMDHDF5Format</code> format.
<code>to_deepmd_npy(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.deepmd.DeePMDCompFormat</code> format.
<code>to_deepmd_npy_mixed(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.deepmd.DeePMDMixedFormat</code> format.
<code>to_deepmd_raw(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.deepmd.DeePMDRawFormat</code> format.
<code>to_dftbplus(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.dftbplus.DFTBplusFormat</code> format.
<code>to_dump(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.lammps.LAMMPSDumpFormat</code> format.
<code>to_fhi_aims_md(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.fhi_aims.FhiMDFormat</code> format.
<code>to_fhi_aims_output(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.fhi_aims.FhiMDFormat</code> format.
<code>to_fhi_aims_scf(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.fhi_aims.FhiSCFFormat</code> format.
<code>to_finalconfig(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.pwmat.PwmatAtomconfigFormat</code> format.
<code>to_gaussian_gjf(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.gaussian.GaussianGJFFormat</code> format.
<code>to_gaussian_log(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.gaussian.GaussianLogFormat</code> format.

continues on next page



Table 4 – continued from previous page

<code>to_gaussian_md(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.gaussian.GaussianMDFormat</code> format.
<code>to_gro(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.gromacs.GromacsGroFormat</code> format.
<code>to_gromacs_gro(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.gromacs.GromacsGroFormat</code> format.
<code>to_json()</code>	Returns a json string representation of the MSONable object.
<code>to_lammps_dump(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.lammps.LAMMPSDumpFormat</code> format.
<code>to_lammps_lmp(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.lammps.LAMMPSLmpFormat</code> format.
<code>to_list(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.list.ListFormat</code> format.
<code>to_lmp(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.lammps.LAMMPSLmpFormat</code> format.
<code>to_mlmd(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.pwmat.PwmatOutputFormat</code> format.
<code>to_mol(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.rdkit.MolFormat</code> format.
<code>to_mol_file(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.rdkit.MolFormat</code> format.
<code>to_movement(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.pwmat.PwmatOutputFormat</code> format.
<code>to_n2p2(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.n2p2.N2P2Format</code> format.
<code>to_openmx_md(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.openmx.OPENMXFormat</code> format.
<code>to_orca_spout(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.orca.ORBASPOutFormat</code> format.
<code>to_outcar(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.vasp.VASPOutcarFormat</code> format.
<code>to_poscar(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.vasp.VASPPoscarFormat</code> format.
<code>to_psi4_inp(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.psi4.PSI4InputFormat</code> format.
<code>to_psi4_out(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.psi4.PSI4OutFormat</code> format.
<code>to_pwmat_atomconfig(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.pwmat.PwmatAtomconfigFormat</code> format.
<code>to_pwmat_finalconfig(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.pwmat.PwmatAtomconfigFormat</code> format.
<code>to_pwmat_mlmd(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.pwmat.PwmatOutputFormat</code> format.
<code>to_pwmat_movement(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.pwmat.PwmatOutputFormat</code> format.
<code>to_pwmat_output(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.pwmat.PwmatOutputFormat</code> format.
<code>to_pymatgen_ComputedStructureEntry(*args, ...)</code>	Dump data to <code>dpdata.plugins.pymatgen.PyMatgenCSEFormat</code> format.
<code>to_pymatgen_computedstructureentry(*args, ...)</code>	Dump data to <code>dpdata.plugins.pymatgen.PyMatgenCSEFormat</code> format.

continues on next page

Table 4 – continued from previous page

<code>to_pymatgen_molecule(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.pymatgen.PyMatgenMoleculeFormat</code> format.
<code>to_pymatgen_structure(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.pymatgen.PyMatgenStructureFormat</code> format.
<code>to_qe_cp_traj(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.qe.QECPTrajFormat</code> format.
<code>to_qe_pw_scf(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.qe.QECPWSCFFormat</code> format.
<code>to_quip_gap_xyz(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.xyz.QuipGapXYZFormat</code> format.
<code>to_quip_gap_xyz_file(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.xyz.QuipGapXYZFormat</code> format.
<code>to_sdf(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.rdkit.SdfFormat</code> format.
<code>to_sdf_file(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.rdkit.SdfFormat</code> format.
<code>to_siesta_aimd_output(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.siesta.SiestaAIMDOutputFormat</code> format.
<code>to_siesta_output(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.siesta.SiestaOutputFormat</code> format.
<code>to_sqm_in(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.amber.SQMInFormat</code> format.
<code>to_sqm_out(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.amber.SQMOutFormat</code> format.
<code>to_stru(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.abacus.AbacusSTRUFormat</code> format.
<code>to_vasp_contcar(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.vasp.VASPPoscarFormat</code> format.
<code>to_vasp_outcar(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.vasp.VASPOutcarFormat</code> format.
<code>to_vasp_poscar(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.vasp.VASPPoscarFormat</code> format.
<code>to_vasp_string(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.vasp.VASPStringFormat</code> format.
<code>to_vasp_xml(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.vasp.VASPXMLFormat</code> format.
<code>to_xml(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.vasp.VASPXMLFormat</code> format.
<code>to_xyz(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.xyz.XYZFormat</code> format.
<code>unsafe_hash()</code>	Returns an hash of the current object.
<code>validate_monty_v1(_MSONable__input_value)</code>	Pydantic validator with correct signature for pydantic v1.x
<code>validate_monty_v2(_MSONable__input_value, _)</code>	Pydantic validator with correct signature for pydantic v2.x

<b>affine_map</b>
<b>from_fmt</b>
<b>from_fmt_obj</b>
<b>replace</b>
<b>rot_frame_lower_triangular</b>
<b>rot_lower_triangular</b>
<b>to_fmt_obj</b>

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

**add\_atom\_names**(*atom\_names*)

Add atom\_names that do not exist.

**affine\_map**(*trans*, *f\_idx=0*)

**append**(*system*)

Append a system to this system.

#### Parameters

**system**

[System] The system to append

**apply\_pbc**()

Append periodic boundary condition.

**apply\_type\_map**(*type\_map*)

Customize the element symbol order and it should maintain order consistency in dpngen or deepmd-kit. It is especially recommended for multiple complex systems with multiple elements.

#### Parameters

**type\_map**

[list] type\_map

**as\_dict**()

Returns data dict of System instance.

**check\_data**()

Check if data is correct.

#### Raises

**DataError**

if data is not correct

**check\_type\_map**(*type\_map*)

Assign atom\_names to type\_map if type\_map is given and different from atom\_names.

#### Parameters

**type\_map**

[list] type\_map

**convert\_to\_mixed\_type**(*type\_map=None*)

Convert the data dict to mixed type format structure, in order to append systems with different formula but the same number of atoms. Change the 'atom\_names' to one placeholder type 'MIXED\_TOKEN' and add 'real\_atom\_types' to store the real type vectors according to the given *type\_map*.

**Parameters**

**type\_map**  
[list] *type\_map*

**copy**()

Returns a copy of the system.

**dump**(*filename, indent=4*)

Dump .json or .yaml file.

**extend**(*systems*)

Extend a system list to this system.

**Parameters**

**systems**  
[[System1, System2, System3 ]] The list to extend

**property formula**

Return the formula of this system, like C3H5O2.

**property formula\_hash: str**

Return the hash of the formula of this system.

**from\_3dmol**(*file\_name, \*\*kwargs*)

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

**from\_abacus\_lcao\_md**(*file\_name, \*\*kwargs*)

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

**from\_abacus\_lcao\_relax**(*file\_name, \*\*kwargs*)

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

**from\_abacus\_lcao\_scf**(*file\_name, \*\*kwargs*)

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

**from\_abacus\_md**(*file\_name, \*\*kwargs*)

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

**from\_abacus\_pw\_md**(*file\_name, \*\*kwargs*)

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

**from\_abacus\_pw\_relax**(*file\_name, \*\*kwargs*)

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

**from\_abacus\_pw\_scf**(*file\_name, \*\*kwargs*)

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

**from\_abacus\_relax**(*file\_name, \*\*kwargs*)

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

**from\_abacus\_scf**(*file\_name, \*\*kwargs*)

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

**from\_abacus\_stru**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.abacus.AbacusSTRUFormat` format.

**from\_amber\_md**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.amber.AmberMDFormat` format.

**from\_ase\_structure**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.ase.ASEStructureFormat` format.

**from\_ase\_traj**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.ase.ASETrajFormat` format.

**from\_atomconfig**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.pwmat.PwmatAtomconfigFormat` format.

**from\_contcar**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.vasp.VASPPoscarFormat` format.

**from\_cp2k\_aimd\_output**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.cp2k.CP2KAIMDOutputFormat` format.

**from\_cp2k\_output**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.cp2k.CP2KOutputFormat` format.

**from\_deepmd**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.deepmd.DeepMDRawFormat` format.

**from\_deepmd\_comp**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.deepmd.DeepMDCompFormat` format.

**from\_deepmd\_hdf5**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.deepmd.DeepMDHDF5Format` format.

**from\_deepmd\_npy**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.deepmd.DeepMDCompFormat` format.

**from\_deepmd\_npy\_mixed**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.deepmd.DeepMDMixedFormat` format.

**from\_deepmd\_raw**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.deepmd.DeepMDRawFormat` format.

**from\_dftbplus**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.dftbplus.DFTBplusFormat` format.

**from\_dump**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.lammps.LAMMPSDumpFormat` format.

**from\_fhi\_aims\_md**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.fhi_aims.FhiMDFormat` format.

**from\_fhi\_aims\_output**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.fhi_aims.FhiMDFormat` format.

**from\_fhi\_aims\_scf**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.fhi_aims.FhiSCFFormat` format.

**from\_finalconfig**(file\_name, \*\*kwargs)  
Read data from `dpdata.plugins.pwmat.PwmatAtomconfigFormat` format.

**from\_fmt**(file\_name, fmt='auto', \*\*kwargs)

**from\_fmt\_obj**(fmtobj, file\_name, \*\*kwargs)

**from\_gaussian\_gjf**(file\_name, \*\*kwargs)  
Read data from `dpdata.plugins.gaussian.GaussianGJFFormat` format.

**from\_gaussian\_log**(file\_name, \*\*kwargs)  
Read data from `dpdata.plugins.gaussian.GaussianLogFormat` format.

**from\_gaussian\_md**(file\_name, \*\*kwargs)  
Read data from `dpdata.plugins.gaussian.GaussianMDFormat` format.

**from\_gro**(file\_name, \*\*kwargs)  
Read data from `dpdata.plugins.gromacs.GromacsGroFormat` format.

**from\_gromacs\_gro**(file\_name, \*\*kwargs)  
Read data from `dpdata.plugins.gromacs.GromacsGroFormat` format.

**from\_lammps\_dump**(file\_name, \*\*kwargs)  
Read data from `dpdata.plugins.lammps.LAMMPSDumpFormat` format.

**from\_lammps\_lmp**(file\_name, \*\*kwargs)  
Read data from `dpdata.plugins.lammps.LAMMPSLmpFormat` format.

**from\_list**(file\_name, \*\*kwargs)  
Read data from `dpdata.plugins.list.ListFormat` format.

**from\_lmp**(file\_name, \*\*kwargs)  
Read data from `dpdata.plugins.lammps.LAMMPSLmpFormat` format.

**from\_mlmd**(file\_name, \*\*kwargs)  
Read data from `dpdata.plugins.pwmat.PwmatOutputFormat` format.

**from\_mol**(file\_name, \*\*kwargs)  
Read data from `dpdata.plugins.rdkit.MolFormat` format.

**from\_mol\_file**(file\_name, \*\*kwargs)  
Read data from `dpdata.plugins.rdkit.MolFormat` format.

**from\_movement**(file\_name, \*\*kwargs)  
Read data from `dpdata.plugins.pwmat.PwmatOutputFormat` format.

**from\_n2p2**(file\_name, \*\*kwargs)  
Read data from `dpdata.plugins.n2p2.N2P2Format` format.

**from\_openmx\_md**(file\_name, \*\*kwargs)  
Read data from `dpdata.plugins.openmx.OPENMXFormat` format.

**from\_orca\_spout**(file\_name, \*\*kwargs)  
Read data from `dpdata.plugins.orca.ORCASPOutFormat` format.

**from\_outcar**(file\_name, \*\*kwargs)  
Read data from `dpdata.plugins.vasp.VASPOutcarFormat` format.

**from\_poscar**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.vasp.VASPPoscarFormat` format.

**from\_psi4\_inp**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.psi4.PSI4InputFormat` format.

**from\_psi4\_out**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.psi4.PSI4OutFormat` format.

**from\_pwmat\_atomconfig**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.pwmat.PwmatAtomconfigFormat` format.

**from\_pwmat\_finalconfig**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.pwmat.PwmatAtomconfigFormat` format.

**from\_pwmat\_mlmd**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.pwmat.PwmatOutputFormat` format.

**from\_pwmat\_movement**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.pwmat.PwmatOutputFormat` format.

**from\_pwmat\_output**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.pwmat.PwmatOutputFormat` format.

**from\_pymatgen\_computedstructureentry**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.pymatgen.PyMatgenCSEFormat` format.

**from\_pymatgen\_molecule**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.pymatgen.PyMatgenMoleculeFormat` format.

**from\_pymatgen\_structure**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.pymatgen.PyMatgenStructureFormat` format.

**from\_qe\_cp\_traj**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.qe.QECPTrajFormat` format.

**from\_qe\_pw\_scf**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.qe.QECPPWSCFFormat` format.

**from\_quip\_gap\_xyz**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.xyz.QuipGapXYZFormat` format.

**from\_quip\_gap\_xyz\_file**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.xyz.QuipGapXYZFormat` format.

**from\_sdf**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.rdkit.SdfFormat` format.

**from\_sdf\_file**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.rdkit.SdfFormat` format.

**from\_siesta\_aiMD\_output**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.siesta.SiestaAIMDOutputFormat` format.

**from\_siesta\_aimd\_output**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.siesta.SiestaAIMDOutputFormat` format.

**from\_siesta\_output**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.siesta.SiestaOutputFormat` format.

**from\_sqm\_in**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.amber.SQMInFormat` format.

**from\_sqm\_out**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.amber.SQMOutFormat` format.

**from\_stru**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.abacus.AbacusSTRUFormat` format.

**from\_vasp\_contcar**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.vasp.VASPPoscarFormat` format.

**from\_vasp\_outcar**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.vasp.VASPOutcarFormat` format.

**from\_vasp\_poscar**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.vasp.VASPPoscarFormat` format.

**from\_vasp\_string**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.vasp.VASPStringFormat` format.

**from\_vasp\_xml**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.vasp.VASPXMLFormat` format.

**from\_xml**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.vasp.VASPXMLFormat` format.

**from\_xyz**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.xyz.XYZFormat` format.

**get\_atom\_names**()  
Returns name of atoms.

**get\_atom\_numbs**()  
Returns number of atoms.

**get\_atom\_types**()  
Returns type of atoms.

**get\_natoms**()  
Returns total number of atoms in the system.

**get\_nframes**()  
Returns number of frames in the system.

**get\_ntypes**() → `int`  
Returns total number of atom types in the system.

**static load**(*filename*)  
Rebuild System obj. from .json or .yaml file.

**map\_atom\_types**(*type\_map=None*) → `ndarray`  
Map the atom types of the system.

#### Parameters



**type\_map**

dict : { "H":0,"O":1 } or list [ "H","C","O","N" ] The map between elements and index if no map\_dict is given, index will be set according to atomic number

**Returns****new\_atom\_types**

[np.ndarray] The mapped atom types

**minimize**(\*args: *Any*, minimizer: *str* | *Minimizer*, \*\*kwargs: *Any*) → *LabeledSystem*

Minimize the geometry.

**Parameters****\*args**

[iterable] Arguments passing to the minimizer

**minimizer**

[str or Minimizer] The assigned minimizer

**\*\*kwargs**

[dict] Other arguments passing to the minimizer

**Returns****labeled\_sys**

[LabeledSystem] A new labeled system.

**property nopbc**

**perturb**(pert\_num, cell\_pert\_fraction, atom\_pert\_distance, atom\_pert\_style='normal')

Perturb each frame in the system randomly. The cell will be deformed randomly, and atoms will be displaced by a random distance in random direction.

**Parameters****pert\_num**

[int] Each frame in the system will make *pert\_num* copies, and all the copies will be perturbed. That means the system to be returned will contain *pert\_num* \* frame\_num of the input system.

**cell\_pert\_fraction**

[float] A fraction determines how much (relatively) will cell deform. The cell of each frame is deformed by a symmetric matrix perturbed from identity. The perturbation to the diagonal part is subject to a uniform distribution in [-cell\_pert\_fraction, cell\_pert\_fraction), and the perturbation to the off-diagonal part is subject to a uniform distribution in [-0.5\*cell\_pert\_fraction, 0.5\*cell\_pert\_fraction).

**atom\_pert\_distance**

[float] unit: Angstrom. A distance determines how far atoms will move. Atoms will move about *atom\_pert\_distance* in random direction. The distribution of the distance atoms move is determined by atom\_pert\_style

**atom\_pert\_style**

[str] Determines the distribution of the distance atoms move is subject to. Available options are

- **'normal': the distance will be object to chi-square distribution with 3 degrees of freedom after normalization.**

The mean value of the distance is *atom\_pert\_fraction*\**side\_length*

- **‘uniform’**: will generate uniformly random points in a 3D-balls with radius as *atom\_pert\_distance*.  
These points are treated as vector used by atoms to move. Obviously, the max length of the distance atoms move is *atom\_pert\_distance*.
- **‘const’**: The distance atoms move will be a constant *atom\_pert\_distance*.

#### Returns

##### **perturbed\_system**

[System] The perturbed\_system. It contains *pert\_num* \* frame\_num of the input system frames.

**pick\_atom\_idx**(*idx*, *nopbc*=None)

Pick atom index.

#### Parameters

##### **idx**

[int or list or slice] atom index

##### **nopbc**

[Boolean (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**

[Boolean (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**

[Boolean (default: None)] If nopbc is True or False, set nopbc

**post\_funcs** = <dpdata.plugin.Plugin object>

**predict**(\*args: Any, driver: str = 'dp', \*\*kwargs: Any) → LabeledSystem

Predict energies and forces by a driver.

#### Parameters

##### **\*args**

[iterable] Arguments passing to the driver

##### **driver**

[str, default=dp] The assigned driver. For compatibility, default is dp

##### **\*\*kwargs**

[dict] Other arguments passing to the driver

**Returns****labeled\_sys**

[LabeledSystem] A new labeled system.

**Examples**

The default driver is DP:

```
>>> labeled_sys = ori_sys.predict("frozen_model_compressed.pb")
```

**classmethod register\_data\_type**(\*data\_type: *Tuple*[DataType])

Register data type.

**Parameters****\*data\_type**

[tuple[DataType]] data type to be registered

**remove\_atom\_names**(atom\_names)

Remove atom names and all such atoms. For example, you may not remove EP atoms in TIP4P/Ew water, which is not a real atom.

**remove\_pbc**(protect\_layer=9)

This method does NOT delete the definition of the cells, it (1) revises the cell to a cubic cell and ensures that the cell boundary to any atom in the system is no less than *protect\_layer* (2) translates the system such that the center-of-geometry of the system locates at the center of the cell.

**Parameters****protect\_layer**

[the protect layer between the atoms and the cell] boundary

**replace**(initial\_atom\_type, end\_atom\_type, replace\_num)

**replicate**(ncopy)

Replicate the each frame in the system in 3 dimensions. Each frame in the system will become a supercell.

**Parameters****ncopy**

list: [4,2,3] or tuple: (4,2,3,) make *ncopy*[0] copys in x dimensions, make *ncopy*[1] copys in y dimensions, make *ncopy*[2] copys in z dimensions.

**Returns****tmp**

[System] The system after replication.

**rot\_frame\_lower\_triangular**(f\_idx=0)

**rot\_lower\_triangular**()

**property short\_formula:** **str**

Return the short formula of this system. Elements with zero number will be removed.

**property short\_name:** **str**

Return the short name of this system (no more than 255 bytes), in the following order:

- formula

- `short_formula`
- `formula_hash`.

**shuffle()**

Shuffle frames randomly.

**sort\_atom\_names**(*type\_map=None*)

Sort `atom_names` of the system and reorder `atom_nums` and `atom_types` according to `atom_names`. If `type_map` is not given, `atom_names` will be sorted by alphabetical order. If `type_map` is given, `atom_names` will be `type_map`.

**Parameters**

**type\_map**

[list] `type_map`

**sort\_atom\_types**() → `ndarray`

Sort atom types.

**Returns**

**idx**

[`np.ndarray`] new atom index in the `Axis.NATOMS`

**sub\_system**(*f\_idx*)

Construct a subsystem from the system.

**Parameters**

**f\_idx**

[int or index] Which frame to use in the subsystem

**Returns**

**sub\_system**

[`System`] The subsystem

**to**(*fmt: str, \*args, \*\*kwargs*) → `System`

Dump systems to the specific format.

**Parameters**

**fmt**

[str] format

**\*args**

arguments

**\*\*kwargs**

keyword arguments

**Returns**

**System**

self

**to\_3dmol**(*\*args, \*\*kwargs*)

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

**to\_abacus\_lcao\_md**(*\*args, \*\*kwargs*)

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

**to\_abacus\_lcao\_relax**(\*args, \*\*kwargs)

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

**to\_abacus\_lcao\_scf**(\*args, \*\*kwargs)

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

**to\_abacus\_md**(\*args, \*\*kwargs)

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

**to\_abacus\_pw\_md**(\*args, \*\*kwargs)

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

**to\_abacus\_pw\_relax**(\*args, \*\*kwargs)

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

**to\_abacus\_pw\_scf**(\*args, \*\*kwargs)

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

**to\_abacus\_relax**(\*args, \*\*kwargs)

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

**to\_abacus\_scf**(\*args, \*\*kwargs)

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

**to\_abacus\_stru**(\*args, \*\*kwargs)

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

**to\_amber\_md**(\*args, \*\*kwargs)

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

**to\_ase\_structure**(\*args, \*\*kwargs)

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

**to\_ase\_traj**(\*args, \*\*kwargs)

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

**to\_atomconfig**(\*args, \*\*kwargs)

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

**to\_contcar**(\*args, \*\*kwargs)

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

**to\_cp2k\_aimd\_output**(\*args, \*\*kwargs)

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

**to\_cp2k\_output**(\*args, \*\*kwargs)

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

**to\_deepmd**(\*args, \*\*kwargs)

Dump data to `dpdata.plugins.deepmd.DeepMDRawFormat` format.

**to\_deepmd\_comp**(\*args, \*\*kwargs)

Dump data to `dpdata.plugins.deepmd.DeepMDCompFormat` format.

**to\_deepmd\_hdf5**(\*args, \*\*kwargs)

Dump data to `dpdata.plugins.deepmd.DeepMDHDF5Format` format.

**to\_deepmd\_npy**(\*args, \*\*kwargs)

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

**to\_deepmd\_npy\_mixed**(\*args, \*\*kwargs)

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

**to\_deepmd\_raw**(\*args, \*\*kwargs)

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

**to\_dftbplus**(\*args, \*\*kwargs)

Dump data to `dpdata.plugins.dftbplus.DFTBplusFormat` format.

**to\_dump**(\*args, \*\*kwargs)

Dump data to `dpdata.plugins.lammps.LAMPSDumpFormat` format.

**to\_fhi\_aims\_md**(\*args, \*\*kwargs)

Dump data to `dpdata.plugins.fhi_aims.FhiMDFormat` format.

**to\_fhi\_aims\_output**(\*args, \*\*kwargs)

Dump data to `dpdata.plugins.fhi_aims.FhiMDFormat` format.

**to\_fhi\_aims\_scf**(\*args, \*\*kwargs)

Dump data to `dpdata.plugins.fhi_aims.FhiSCFFormat` format.

**to\_finalconfig**(\*args, \*\*kwargs)

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

**to\_fmt\_obj**(fmtobj, \*args, \*\*kwargs)

**to\_gaussian\_gjf**(\*args, \*\*kwargs)

Dump data to `dpdata.plugins.gaussian.GaussianGJFFormat` 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.LAMPSDumpFormat` format.

**to\_lammps\_lmp**(\*args, \*\*kwargs)

Dump data to `dpdata.plugins.lammps.LAMPSLmpFormat` format.

**to\_list**(\*args, \*\*kwargs)

Dump data to `dpdata.plugins.list.ListFormat` format.

**to\_lmp**(\*args, \*\*kwargs)

Dump data to `dpdata.plugins.lammps.LAMPSLmpFormat` format.

**to\_mlmd**(\*args, \*\*kwargs)

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

**to\_mol**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.rdkit.MolFormat` format.

**to\_mol\_file**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.rdkit.MolFormat` format.

**to\_movement**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.pwmat.PwmatOutputFormat` format.

**to\_n2p2**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.n2p2.N2P2Format` format.

**to\_openmx\_md**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.openmx.OPENMXFormat` format.

**to\_orca\_spout**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.orca.ORBASPOutFormat` format.

**to\_outcar**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.vasp.VASPOutcarFormat` format.

**to\_poscar**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.vasp.VASPPoscarFormat` format.

**to\_psi4\_inp**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.psi4.PSI4InputFormat` format.

**to\_psi4\_out**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.psi4.PSI4OutFormat` format.

**to\_pwmat\_atomconfig**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.pwmat.PwmatAtomconfigFormat` format.

**to\_pwmat\_finalconfig**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.pwmat.PwmatAtomconfigFormat` format.

**to\_pwmat\_mlmd**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.pwmat.PwmatOutputFormat` format.

**to\_pwmat\_movement**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.pwmat.PwmatOutputFormat` format.

**to\_pwmat\_output**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.pwmat.PwmatOutputFormat` format.

**to\_pymatgen\_ComputedStructureEntry**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.pymatgen.PyMatgenCSEFormat` format.

**to\_pymatgen\_computedstructureentry**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.pymatgen.PyMatgenCSEFormat` format.

**to\_pymatgen\_molecule**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.pymatgen.PyMatgenMoleculeFormat` format.

**to\_pymatgen\_structure**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.pymatgen.PyMatgenStructureFormat` format.

**to\_qe\_cp\_traj**(\*args, \*\*kwargs)

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

**to\_qe\_pw\_scf**(\*args, \*\*kwargs)

Dump data to `dpdata.plugins.qe.QECPWSCFFormat` format.

**to\_quip\_gap\_xyz**(\*args, \*\*kwargs)

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

**to\_quip\_gap\_xyz\_file**(\*args, \*\*kwargs)

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

**to\_sdf**(\*args, \*\*kwargs)

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

**to\_sdf\_file**(\*args, \*\*kwargs)

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

**to\_siesta\_aimd\_output**(\*args, \*\*kwargs)

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

**to\_siesta\_output**(\*args, \*\*kwargs)

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

**to\_sqm\_in**(\*args, \*\*kwargs)

Dump data to `dpdata.plugins.amber.SQMInFormat` format.

**to\_sqm\_out**(\*args, \*\*kwargs)

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

**to\_stru**(\*args, \*\*kwargs)

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

**to\_vasp\_contcar**(\*args, \*\*kwargs)

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

**to\_vasp\_outcar**(\*args, \*\*kwargs)

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

**to\_vasp\_poscar**(\*args, \*\*kwargs)

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

**to\_vasp\_string**(\*args, \*\*kwargs)

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

**to\_vasp\_xml**(\*args, \*\*kwargs)

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

**to\_xml**(\*args, \*\*kwargs)

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

**to\_xyz**(\*args, \*\*kwargs)

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

**property uniq\_formula**

Return the `uniq_formula` of this system. The `uniq_formula` sort the elements in formula by names. Systems with the same `uniq_formula` can be append together.



## 6.1.1 Subpackages

### dpdata.abacus package

#### Submodules

#### dpdata.abacus.md module

```
dpdata.abacus.md.get_coord_dump_freq(inlines)
dpdata.abacus.md.get_coords_from_dump(dumplines, natoms)
dpdata.abacus.md.get_energy(outlines, ndump, dump_freq)
dpdata.abacus.md.get_frame(fname)
dpdata.abacus.md.get_path_out(fname, inlines)
```

#### dpdata.abacus.relax module

```
dpdata.abacus.relax.get_coords_from_log(loglines, natoms)
```

NOTICE: unit of coords and cells is Angstrom order:

coordinate cell (no output if cell is not changed) energy (no output, if SCF is not converged) force (no output, if cal\_force is not setted or abnormal ending) stress (no output, if set cal\_stress is not setted or abnormal ending).

```
dpdata.abacus.relax.get_frame(fname)
dpdata.abacus.relax.get_log_file(fname, inlines)
```

#### dpdata.abacus.scf module

```
dpdata.abacus.scf.CheckFile(ifile)
dpdata.abacus.scf.collect_force(outlines)
dpdata.abacus.scf.collect_stress(outlines)
dpdata.abacus.scf.get_block(lines, keyword, skip=0, nlines=None)
dpdata.abacus.scf.get_cell(geometry_inlines)
dpdata.abacus.scf.get_coords(celldm, cell, geometry_inlines, inlines=None)
dpdata.abacus.scf.get_energy(outlines)
dpdata.abacus.scf.get_force(outlines, natoms)
dpdata.abacus.scf.get_frame(fname)
dpdata.abacus.scf.get_frame_from_stru(fname)
dpdata.abacus.scf.get_geometry_in(fname, inlines)
```

```
dpdata.abacus.scf.get_nele_from_stru(geometry_inlines)
```

```
dpdata.abacus.scf.get_path_out(fname, inlines)
```

```
dpdata.abacus.scf.get_stress(outlines)
```

```
dpdata.abacus.scf.get_stru_block(lines, keyword)
```

```
dpdata.abacus.scf.make_unlabeled_stru(data, frame_idx, pp_file=None, numerical_orbital=None,  
                                     numerical_descriptor=None, mass=None)
```

## dpdata.amber package

### Submodules

#### dpdata.amber.mask module

Amber mask.

```
dpdata.amber.mask.load_param_file(param_file)
```

```
dpdata.amber.mask.pick_by_amber_mask(param, maskstr, coords=None)
```

Pick atoms by amber masks.

##### Parameters

###### param

[str or parmed.Structure] filename of Amber param file or parmed.Structure

###### maskstr

[str] Amber masks

###### coords

[np.ndarray (optional)] frame coordinates, shape: N\*3

#### dpdata.amber.md module

```
dpdata.amber.md.read_amber_traj(parm7_file, nc_file, mdfrc_file=None, mden_file=None, mdout_file=None,  
                                use_element_symbols=None, labeled=True)
```

The amber trajectory includes: \* nc, NetCDF format, stores coordinates \* mdfrc, NetCDF format, stores forces \* mden (optional), text format, stores energies \* mdout (optional), text format, may store energies if there is no mden\_file \* parm7, text format, stores types.

##### Parameters

**parm7\_file, nc\_file, mdfrc\_file, mden\_file, mdout\_file:**  
filenames

###### use\_element\_symbols

[None or list or str] If use\_element\_symbols is a list of atom indexes, these atoms will use element symbols instead of amber types. For example, a ligand will use C, H, O, N, and so on instead of h1, hc, o, os, and so on. IF use\_element\_symbols is str, it will be considered as Amber mask.

###### labeled

[bool] Whether to return labeled data

## dpdata.amber.sqm module

`dpdata.amber.sqm.make_sqm_in(data, fname=None, frame_idx=0, **kwargs)`

`dpdata.amber.sqm.parse_sqm_out(fname)`

Read atom symbols, charges and coordinates from ambertools sqm.out file.

## dpdata.cp2k package

### Submodules

### dpdata.cp2k.cell module

`dpdata.cp2k.cell.cell_to_low_triangle(A, B, C, alpha, beta, gamma)`

Convert cell to low triangle matrix.

#### Parameters

**A**

[float] cell length A

**B**

[float] cell length B

**C**

[float] cell length C

**alpha**

[float] radian. The angle between vector B and vector C.

**beta**

[float] radian. The angle between vector A and vector C.

**gamma**

[float] radian. The angle between vector B and vector C.

#### Returns

**cell**

[list] The cell matrix used by dpdata in low triangle form.

## dpdata.cp2k.output module

`class dpdata.cp2k.output.Cp2kSystems(log_file_name, xyz_file_name, restart=False)`

Bases: `object`

deal with cp2k outputfile.

## Methods

<code>get_log_block_generator</code>
<code>get_xyz_block_generator</code>
<code>handle_single_log_frame</code>
<code>handle_single_xyz_frame</code>

`get_log_block_generator()`

`get_xyz_block_generator()`

`handle_single_log_frame(lines)`

`handle_single_xyz_frame(lines)`

`dpdata.cp2k.output.get_frames(fname)`

## dpdata.deepmd package

### Submodules

#### dpdata.deepmd.comp module

`dpdata.deepmd.comp.dump(folder, data, set_size=5000, comp_prec=<class 'numpy.float32'>, remove_sets=True)`

`dpdata.deepmd.comp.to_system_data(folder, type_map=None, labels=True)`

#### dpdata.deepmd.hdf5 module

Utils for deepmd/hdf5 format.

`dpdata.deepmd.hdf5.dump(f: ~h5py._hl.files.File | ~h5py._hl.group.Group, folder: str, data: dict, set_size=5000, comp_prec=<class 'numpy.float32'>) → None`

Dump data to a HDF5 file.

##### Parameters

**f**

[h5py.File or h5py.Group] HDF5 file or group object

**folder**

[str] path in the HDF5 file

**data**

[dict] System or LabeledSystem data

**set\_size**

[int, default: 5000] size of a set

**comp\_prec**

[np.dtype, default: np.float32] precision of data

`dpdata.deepmd.hdf5.to_system_data(f: File | Group, folder: str, type_map: list | None = None, labels: bool = True)`

Load a HDF5 file.

#### Parameters

**f**  
[h5py.File or h5py.Group] HDF5 file or group object

**folder**  
[str] path in the HDF5 file

**type\_map**  
[list] type map

**labels**  
[bool] labels

### dpdata.deepmd.mixed module

`dpdata.deepmd.mixed.dump(folder, data, set_size=2000, comp_prec=<class 'numpy.float32'>, remove_sets=True)`

`dpdata.deepmd.mixed.formula(atom_names, atom_numbs)`

Return the formula of this system, like C3H5O2.

`dpdata.deepmd.mixed.load_type(folder)`

`dpdata.deepmd.mixed.mix_system(*system, type_map, **kwargs)`

Mix the systems into mixed\_type ones according to the unified given type\_map.

#### Parameters

**\*system**  
[System] The systems to mix

**type\_map**  
[list of str] Maps atom type to name

**\*\*kwargs**  
[dict] Other parameters

#### Returns

**mixed\_systems: dict**  
dict of mixed system with key 'atom\_numbs'

`dpdata.deepmd.mixed.split_system(sys, split_num=10000)`

`dpdata.deepmd.mixed.to_system_data(folder, type_map=None, labels=True)`

## dpdata.deepmd.raw module

`dpdata.deepmd.raw.dump(folder, data)`

`dpdata.deepmd.raw.load_type(folder, type_map=None)`

`dpdata.deepmd.raw.to_system_data(folder, type_map=None, labels=True)`

## dpdata.dftbplus package

### Submodules

## dpdata.dftbplus.output module

`dpdata.dftbplus.output.read_dftb_plus(fn_1: str, fn_2: str) → Tuple[str, ndarray, float, ndarray]`

Read from DFTB+ input and output.

#### Parameters

**fn\_1**

[str] DFTB+ input file name

**fn\_2**

[str] DFTB+ output file name

#### Returns

**str**

atomic symbols

**np.ndarray**

atomic coordinates

**float**

total potential energy

**np.ndarray**

atomic forces

## dpdata.fhi\_aims package

### Submodules

## dpdata.fhi\_aims.output module

`dpdata.fhi_aims.output.analyze_block(lines, first_blk=False, md=True)`

`dpdata.fhi_aims.output.get_fhi_aims_block(fp)`

`dpdata.fhi_aims.output.get_frames(fname, md=True, begin=0, step=1, convergence_check=True)`

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

## dpdata.gaussian package

### Submodules

#### dpdata.gaussian.gjf module

Generate Gaussian input file.

`dpdata.gaussian.gjf.detect_multiplicity(symbols: ndarray) → int`

Find the minimal multiplicity of the given molecules.

##### Parameters

###### symbols

[np.ndarray] element symbols; virtual elements are not supported

##### Returns

###### int

spin multiplicity

`dpdata.gaussian.gjf.make_gaussian_input(sys_data: dict, keywords: str | List[str], multiplicity: str | int = 'auto', charge: int = 0, fragment_guesses: bool = False, basis_set: str | None = None, keywords_high_multiplicity: str | None = None, nproc: int = 1) → str`

Make gaussian input file.

##### Parameters

###### sys\_data

[dict] system data

###### keywords

[str or list[str]] Gaussian keywords, e.g. force b3lyp/6-31g\*\*. If a list, run multiple steps

###### multiplicity

[str or int, default=auto] spin multiplicity state. It can be a number. If auto, multiplicity will be detected automatically, with the following rules:

###### fragment\_guesses=True

multiplicity will +1 for each radical, and +2 for each oxygen molecule

###### fragment\_guesses=False

multiplicity will be 1 or 2, but +2 for each oxygen molecule

###### charge

[int, default=0] molecule charge. Only used when charge is not provided by the system

###### fragment\_guesses

[bool, default=False] initial guess generated from fragment guesses. If True, multiplicity should be auto

###### basis\_set

[str, default=None] custom basis set

###### keywords\_high\_multiplicity

[str, default=None] keywords for points with multiple raicals. multiplicity should be auto. If not set, fallback to normal keywords

###### nproc

[int, default=1] Number of CPUs to use

**Returns****str**

gif output string

`dpdata.gaussian.gjf.read_gaussian_input(inp: str)`

Read Gaussian input.

**Parameters****inp**

[str] Gaussian input str

**Returns****dict**

system data

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

Read Gaussian log file.

**Parameters****file\_name**

[str] file name

**md**

[bool, default False] whether to read multiple frames

**Returns****data**

[dict] system data

**Raises****RuntimeError**

if the input orientation is not found

**dpdata.gromacs package****Submodules****dpdata.gromacs.gro module**`dpdata.gromacs.gro.file_to_system_data(fname, format_atom_name=True, **kwargs)``dpdata.gromacs.gro.from_system_data(system, f_idx=0, **kwargs)`



## dpdata.lammps package

### Submodules

#### dpdata.lammps.dump module

**exception** dpdata.lammps.dump.UnwrapWarning

Bases: `UserWarning`

dpdata.lammps.dump.**box2dumpbox**(*orig*, *box*)

dpdata.lammps.dump.**dumpbox2box**(*bounds*, *tilt*)

dpdata.lammps.dump.**get\_atype**(*lines*, *type\_idx\_zero=False*)

dpdata.lammps.dump.**get\_coordtype\_and\_scalefactor**(*keys*)

dpdata.lammps.dump.**get\_dumpbox**(*lines*)

dpdata.lammps.dump.**get\_natoms**(*lines*)

dpdata.lammps.dump.**get\_natoms\_vec**(*lines*)

dpdata.lammps.dump.**get\_natomtypes**(*lines*)

dpdata.lammps.dump.**load\_file**(*fname*, *begin=0*, *step=1*)

dpdata.lammps.dump.**safe\_get\_posi**(*lines*, *cell*, *orig=array([0., 0., 0.])*, *unwrap=False*)

dpdata.lammps.dump.**split\_traj**(*dump\_lines*)

dpdata.lammps.dump.**system\_data**(*lines*, *type\_map=None*, *type\_idx\_zero=True*, *unwrap=False*)

#### dpdata.lammps.lmp module

dpdata.lammps.lmp.**box2lmpbox**(*orig*, *box*)

dpdata.lammps.lmp.**from\_system\_data**(*system*, *f\_idx=0*)

dpdata.lammps.lmp.**get\_atoms**(*lines*)

dpdata.lammps.lmp.**get\_atype**(*lines*, *type\_idx\_zero=False*)

dpdata.lammps.lmp.**get\_lmpbox**(*lines*)

dpdata.lammps.lmp.**get\_natoms**(*lines*)

dpdata.lammps.lmp.**get\_natoms\_vec**(*lines*)

dpdata.lammps.lmp.**get\_natomtypes**(*lines*)

dpdata.lammps.lmp.**get\_posi**(*lines*)

dpdata.lammps.lmp.**lmpbox2box**(*lohi*, *tilt*)

dpdata.lammps.lmp.**system\_data**(*lines*, *type\_map=None*, *type\_idx\_zero=True*)

dpdata.lammps.lmp.**to\_system\_data**(*lines*, *type\_map=None*, *type\_idx\_zero=True*)

## dpdata.openmx package

### Submodules

#### dpdata.openmx.omx module

`dpdata.openmx.omx.load_atom(lines)`  
`dpdata.openmx.omx.load_cells(lines)`  
`dpdata.openmx.omx.load_coords(lines, atom_names, natoms)`  
`dpdata.openmx.omx.load_data(mdname, atom_names, natoms)`  
`dpdata.openmx.omx.load_energy(lines)`  
`dpdata.openmx.omx.load_force(lines, atom_names, atom_numbs)`  
`dpdata.openmx.omx.load_param_file(fname, mdname)`  
`dpdata.openmx.omx.to_system_data(fname, mdname)`  
`dpdata.openmx.omx.to_system_label(fname, mdname)`

## dpdata.orca package

### Submodules

#### dpdata.orca.output module

`dpdata.orca.output.read_orca_sp_output(fn: str) → Tuple[ndarray, ndarray, float, ndarray]`

Read from ORCA output.

Note that both the energy and the gradient should be printed.

#### Parameters

**fn**  
[str] file name

#### Returns

**np.ndarray**  
atomic symbols

**np.ndarray**  
atomic coordinates

**float**  
total potential energy

**np.ndarray**  
atomic forces

## dpdata.plugins package

### Submodules

### dpdata.plugins.3dmol module

**class** dpdata.plugins.3dmol.Py3DMolFormat

Bases: *Format*

3DMol format.

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

### Methods

<code>MultiModes()</code>	File mode for MultiSystems.
<code>from_bond_order_system(file_name, **kwargs)</code>	Implement BondOrderSystem.from that converts from this format to BondOrderSystem.
<code>from_labeled_system(file_name, **kwargs)</code>	Implement LabeledSystem.from that converts from this format to LabeledSystem.
<code>from_multi_systems(directory, **kwargs)</code>	Implement MultiSystems.from that converts from this format to MultiSystems.
<code>from_system(file_name, **kwargs)</code>	Implement System.from that converts from this format to System.
<code>get_formats()</code>	Get all registered formats.
<code>get_from_methods()</code>	Get all registered from methods.
<code>get_to_methods()</code>	Get all registered to methods.
<code>mix_system(*system, type_map, **kwargs)</code>	Mix the systems into mixed_type ones according to the unified given type_map.
<code>post(func_name)</code>	Register a post function for from method.
<code>register(key)</code>	Register a format plugin.
<code>register_from(key)</code>	Register a from method if the target method name is not default.
<code>register_to(key)</code>	Register a to method if the target method name is not default.
<code>to_bond_order_system(data, rdkit_mol, *args, ...)</code>	Implement BondOrderSystem.to that converts from BondOrderSystem to this format.
<code>to_labeled_system(data, *args, **kwargs)</code>	Implement LabeledSystem.to that converts from LabeledSystem to this format.
<code>to_multi_systems(formulas, directory, **kwargs)</code>	Implement MultiSystems.to that converts from MultiSystems to this format.
<code>to_system(data[, f_idx, size, style])</code>	Show 3D structure of a frame in jupyter.

**to\_system**(data: dict, f\_idx: int = 0, size: Tuple[int] = (300, 300), style: dict = {'sphere': {'radius': 0.4}, 'stick': {}}, \*\*kwargs)

Show 3D structure of a frame in jupyter.

#### Parameters

**data**

[dict] system data

**f\_idx**  
[int] frame index to show

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

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

**\*\*kwargs**  
[dict] other parameters

## Examples

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

## dpdata.plugins.abacus module

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

Bases: *Format*

## Methods

<b>MultiModes()</b>	File mode for MultiSystems.
<b>from_bond_order_system(file_name, **kwargs)</b>	Implement BondOrderSystem.from that converts from this format to BondOrderSystem.
<b>from_labeled_system(file_name, **kwargs)</b>	Implement LabeledSystem.from that converts from this format to LabeledSystem.
<b>from_multi_systems(directory, **kwargs)</b>	Implement MultiSystems.from that converts from this format to MultiSystems.
<b>from_system(file_name, **kwargs)</b>	Implement System.from that converts from this format to System.
<b>get_formats()</b>	Get all registered formats.
<b>get_from_methods()</b>	Get all registered from methods.
<b>get_to_methods()</b>	Get all registered to methods.
<b>mix_system(*system, type_map, **kwargs)</b>	Mix the systems into mixed_type ones according to the unified given type_map.
<b>post(func_name)</b>	Register a post function for from method.
<b>register(key)</b>	Register a format plugin.
<b>register_from(key)</b>	Register a from method if the target method name is not default.
<b>register_to(key)</b>	Register a to method if the target method name is not default.
<b>to_bond_order_system(data, rdkit_mol, *args, ...)</b>	Implement BondOrderSystem.to that converts from BondOrderSystem to this format.
<b>to_labeled_system(data, *args, **kwargs)</b>	Implement LabeledSystem.to that converts from LabeledSystem to this format.
<b>to_multi_systems(formulas, directory, **kwargs)</b>	Implement MultiSystems.to that converts from MultiSystems to this format.
<b>to_system(data, *args, **kwargs)</b>	Implement System.to that converts from System to this format.

**from\_labeled\_system**(*file\_name*, **\*\*kwargs**)

Implement LabeledSystem.from that converts from this format to LabeledSystem.

#### Parameters

**file\_name**

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

**\*\*kwargs**

[dict] keyword arguments that will be passed from the method

#### Returns

**data**

[dict] system data, whose keys are defined in LabeledSystem.DTYPES

**class** dpdata.plugins.abacus.**AbacusRelaxFormat**

Bases: *Format*

#### Methods

<b>MultiModes()</b>	File mode for MultiSystems.
<b>from_bond_order_system</b> ( <i>file_name</i> , <b>**kwargs</b> )	Implement BondOrderSystem.from that converts from this format to BondOrderSystem.
<i>from_labeled_system</i> ( <i>file_name</i> , <b>**kwargs</b> )	Implement LabeledSystem.from that converts from this format to LabeledSystem.
<b>from_multi_systems</b> ( <i>directory</i> , <b>**kwargs</b> )	Implement MultiSystems.from that converts from this format to MultiSystems.
<b>from_system</b> ( <i>file_name</i> , <b>**kwargs</b> )	Implement System.from that converts from this format to System.
<b>get_formats()</b>	Get all registered formats.
<b>get_from_methods()</b>	Get all registered from methods.
<b>get_to_methods()</b>	Get all registered to methods.
<b>mix_system</b> (* <i>system</i> , <i>type_map</i> , <b>**kwargs</b> )	Mix the systems into mixed_type ones according to the unified given type_map.
<b>post</b> ( <i>func_name</i> )	Register a post function for from method.
<b>register</b> ( <i>key</i> )	Register a format plugin.
<b>register_from</b> ( <i>key</i> )	Register a from method if the target method name is not default.
<b>register_to</b> ( <i>key</i> )	Register a to method if the target method name is not default.
<b>to_bond_order_system</b> ( <i>data</i> , <i>rdkit_mol</i> , <i>*args</i> , ...)	Implement BondOrderSystem.to that converts from BondOrderSystem to this format.
<b>to_labeled_system</b> ( <i>data</i> , <i>*args</i> , <b>**kwargs</b> )	Implement LabeledSystem.to that converts from LabeledSystem to this format.
<b>to_multi_systems</b> ( <i>formulas</i> , <i>directory</i> , <b>**kwargs</b> )	Implement MultiSystems.to that converts from MultiSystems to this format.
<b>to_system</b> ( <i>data</i> , <i>*args</i> , <b>**kwargs</b> )	Implement System.to that converts from System to this format.

**from\_labeled\_system**(*file\_name*, **\*\*kwargs**)

Implement LabeledSystem.from that converts from this format to LabeledSystem.

#### Parameters

**file\_name**

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

**\*\*kwargs**

[dict] keyword arguments that will be passed from the method

**Returns****data**

[dict] system data, whose keys are defined in LabeledSystem.DTYPES

**class** dpdata.plugins.abacus.**AbacusSCFFormat**Bases: *Format***Methods**

<b>MultiModes()</b>	File mode for MultiSystems.
<b>from_bond_order_system(file_name, **kwargs)</b>	Implement BondOrderSystem.from that converts from this format to BondOrderSystem.
<b>from_labeled_system(file_name, **kwargs)</b>	Implement LabeledSystem.from that converts from this format to LabeledSystem.
<b>from_multi_systems(directory, **kwargs)</b>	Implement MultiSystems.from that converts from this format to MultiSystems.
<b>from_system(file_name, **kwargs)</b>	Implement System.from that converts from this format to System.
<b>get_formats()</b>	Get all registered formats.
<b>get_from_methods()</b>	Get all registered from methods.
<b>get_to_methods()</b>	Get all registered to methods.
<b>mix_system(*system, type_map, **kwargs)</b>	Mix the systems into mixed_type ones according to the unified given type_map.
<b>post(func_name)</b>	Register a post function for from method.
<b>register(key)</b>	Register a format plugin.
<b>register_from(key)</b>	Register a from method if the target method name is not default.
<b>register_to(key)</b>	Register a to method if the target method name is not default.
<b>to_bond_order_system(data, rdkit_mol, *args, ...)</b>	Implement BondOrderSystem.to that converts from BondOrderSystem to this format.
<b>to_labeled_system(data, *args, **kwargs)</b>	Implement LabeledSystem.to that converts from LabeledSystem to this format.
<b>to_multi_systems(formulas, directory, **kwargs)</b>	Implement MultiSystems.to that converts from MultiSystems to this format.
<b>to_system(data, *args, **kwargs)</b>	Implement System.to that converts from System to this format.

**from\_labeled\_system(file\_name, \*\*kwargs)**

Implement LabeledSystem.from that converts from this format to LabeledSystem.

**Parameters****file\_name**

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

**\*\*kwargs**

[dict] keyword arguments that will be passed from the method

**Returns****data**

[dict] system data, whose keys are defined in LabeledSystem.DTYPES

**class** dpdata.plugins.abacus.AbacusSTRUFormatBases: *Format***Methods**

<code>MultiModes()</code>	File mode for MultiSystems.
<code>from_bond_order_system(file_name, **kwargs)</code>	Implement BondOrderSystem.from that converts from this format to BondOrderSystem.
<code>from_labeled_system(file_name, **kwargs)</code>	Implement LabeledSystem.from that converts from this format to LabeledSystem.
<code>from_multi_systems(directory, **kwargs)</code>	Implement MultiSystems.from that converts from this format to MultiSystems.
<code>from_system(file_name, **kwargs)</code>	Implement System.from that converts from this format to System.
<code>get_formats()</code>	Get all registered formats.
<code>get_from_methods()</code>	Get all registered from methods.
<code>get_to_methods()</code>	Get all registered to methods.
<code>mix_system(*system, type_map, **kwargs)</code>	Mix the systems into mixed_type ones according to the unified given type_map.
<code>post(func_name)</code>	Register a post function for from method.
<code>register(key)</code>	Register a format plugin.
<code>register_from(key)</code>	Register a from method if the target method name is not default.
<code>register_to(key)</code>	Register a to method if the target method name is not default.
<code>to_bond_order_system(data, rdkit_mol, *args, ...)</code>	Implement BondOrderSystem.to that converts from BondOrderSystem to this format.
<code>to_labeled_system(data, *args, **kwargs)</code>	Implement LabeledSystem.to that converts from LabeledSystem to this format.
<code>to_multi_systems(formulas, directory, **kwargs)</code>	Implement MultiSystems.to that converts from MultiSystems to this format.
<code>to_system(data, file_name[, frame_idx])</code>	Dump the system into ABACUS STRU format file.

**from\_system**(file\_name, \*\*kwargs)

Implement System.from that converts from this format to System.

**Parameters****file\_name**

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

**\*\*kwargs**

[dict] keyword arguments that will be passed from the method

**Returns****data**

[dict] system data, whose keys are defined in System.DTYPES

**to\_system**(*data*, *file\_name*, *frame\_idx*=0, **\*\*kwargs**)

Dump the system into ABACUS STRU format file.

**Parameters**

**data**

[dict] System data

**file\_name**

[str] The output file name

**frame\_idx**

[int] The index of the frame to dump

**pp\_file**

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

**numerical\_orbital**

[list of string, optional] List of orbital files

**mass**

[list of float, optional] List of atomic masses

**numerical\_descriptor**

[str, optional] numerical descriptor file

**\*\*kwargs**

[dict] other parameters

**dpdata.plugins.amber module**

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

Bases: *Format*



## Methods

<code>MultiModes()</code>	File mode for MultiSystems.
<code>from_bond_order_system(file_name, **kwargs)</code>	Implement BondOrderSystem.from that converts from this format to BondOrderSystem.
<code>from_labeled_system([file_name, parm7_file, ...])</code>	Implement LabeledSystem.from that converts from this format to LabeledSystem.
<code>from_multi_systems(directory, **kwargs)</code>	Implement MultiSystems.from that converts from this format to MultiSystems.
<code>from_system([file_name, parm7_file, ...])</code>	Implement System.from that converts from this format to System.
<code>get_formats()</code>	Get all registered formats.
<code>get_from_methods()</code>	Get all registered from methods.
<code>get_to_methods()</code>	Get all registered to methods.
<code>mix_system(*system, type_map, **kwargs)</code>	Mix the systems into mixed_type ones according to the unified given type_map.
<code>post(func_name)</code>	Register a post function for from method.
<code>register(key)</code>	Register a format plugin.
<code>register_from(key)</code>	Register a from method if the target method name is not default.
<code>register_to(key)</code>	Register a to method if the target method name is not default.
<code>to_bond_order_system(data, rdkit_mol, *args, ...)</code>	Implement BondOrderSystem.to that converts from BondOrderSystem to this format.
<code>to_labeled_system(data, *args, **kwargs)</code>	Implement LabeledSystem.to that converts from LabeledSystem to this format.
<code>to_multi_systems(formulas, directory, **kwargs)</code>	Implement MultiSystems.to that converts from MultiSystems to this format.
<code>to_system(data, *args, **kwargs)</code>	Implement System.to that converts from System to this format.

**from\_labeled\_system**(*file\_name=None, parm7\_file=None, nc\_file=None, mdfrc\_file=None, mden\_file=None, mdout\_file=None, use\_element\_symbols=None, \*\*kwargs*)

Implement LabeledSystem.from that converts from this format to LabeledSystem.

### Parameters

#### **file\_name**

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

#### **\*\*kwargs**

[dict] keyword arguments that will be passed from the method

### Returns

#### **data**

[dict] system data, whose keys are defined in LabeledSystem.DTYPES

**from\_system**(*file\_name=None, parm7\_file=None, nc\_file=None, use\_element\_symbols=None, \*\*kwargs*)

Implement System.from that converts from this format to System.

### Parameters

#### **file\_name**

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

**\*\*kwargs**

[dict] keyword arguments that will be passed from the method

### Returns

**data**

[dict] system data, whose keys are defined in `System.DTYPES`

**class** `dpdata.plugins.amber.SQMDriver(sqm_exec: str = 'sqm', **kwargs: dict)`

Bases: [Driver](#)

AMBER sqm program driver.

### Parameters

**sqm\_exec**

[str, default=sqm] path to sqm program

**\*\*kwargs**

[dict] other arguments to make input files. See [SQMINFormat](#)

### Examples

Use DFTB3 method to calculate potential energy:

```
>>> labeled_system = system.predict(theory="DFTB3", driver="sqm")
>>> labeled_system['energies'][0]
-15.41111246
```

### Attributes

**ase\_calculator**

Returns an ase calculator based on this driver.

### Methods

<code>get_driver(key)</code>	Get a driver plugin.
<code>get_drivers()</code>	Get all driver plugins.
<code>label(data)</code>	Label a system data.
<code>register(key)</code>	Register a driver plugin.

**label**(data: dict) → dict

Label a system data. Returns new data with energy, forces, and virials.

### Parameters

**data**

[dict] data with coordinates and atom types

### Returns

**dict**

labeled data with energies and forces

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

Bases: [Format](#)

## Methods

<code>MultiModes()</code>	File mode for MultiSystems.
<code>from_bond_order_system(file_name, **kwargs)</code>	Implement BondOrderSystem.from that converts from this format to BondOrderSystem.
<code>from_labeled_system(file_name, **kwargs)</code>	Implement LabeledSystem.from that converts from this format to LabeledSystem.
<code>from_multi_systems(directory, **kwargs)</code>	Implement MultiSystems.from that converts from this format to MultiSystems.
<code>from_system(file_name, **kwargs)</code>	Implement System.from that converts from this format to System.
<code>get_formats()</code>	Get all registered formats.
<code>get_from_methods()</code>	Get all registered from methods.
<code>get_to_methods()</code>	Get all registered to methods.
<code>mix_system(*system, type_map, **kwargs)</code>	Mix the systems into mixed_type ones according to the unified given type_map.
<code>post(func_name)</code>	Register a post function for from method.
<code>register(key)</code>	Register a format plugin.
<code>register_from(key)</code>	Register a from method if the target method name is not default.
<code>register_to(key)</code>	Register a to method if the target method name is not default.
<code>to_bond_order_system(data, rdkit_mol, *args, ...)</code>	Implement BondOrderSystem.to that converts from BondOrderSystem to this format.
<code>to_labeled_system(data, *args, **kwargs)</code>	Implement LabeledSystem.to that converts from LabeledSystem to this format.
<code>to_multi_systems(formulas, directory, **kwargs)</code>	Implement MultiSystems.to that converts from MultiSystems to this format.
<code>to_system(data[, fname, frame_idx])</code>	Generate input files for semi-emperical calculation in sqm software.

**to\_system**(data, fname=None, frame\_idx=0, \*\*kwargs)

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

### Parameters

**data**

[dict] system data

**fname**

[str] output file name

**frame\_idx**

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

**\*\*kwargs**

[dict] other parameters

### Other Parameters

**\*\*kwargs**

[dict]

**valid parameters are:**

**qm\_theory**

[str, default=dftb3] level of theory. Options includes AM1, RM1, MNDO,

PM3-PDDG, MNDO-PDDG, PM3-CARB1, MNDO/d, AM1/d, PM6, DFTB2, DFTB3

**charge**

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

**maxcyc**

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

**mult**

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

**class** dpdata.plugins.amber.SQMMinimizer(maxcyc=1000, \*args, \*\*kwargs)

Bases: [Minimizer](#)

SQM minimizer.

**Parameters****maxcyc**

[int, default=1000] maximum cycle to minimize

**Methods**

<code>get_minimizer(key)</code>	Get a minimizer plugin.
<code>get_minimizers()</code>	Get all minimizer plugins.
<code>minimize(data)</code>	Minimize the geometry.
<code>register(key)</code>	Register a minimizer plugin.

**minimize**(data: *dict*) → *dict*

Minimize the geometry.

**Parameters****data**

[dict] data with coordinates and atom types

**Returns****dict**

labeled data with minimized coordinates, energies, and forces

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

Bases: [Format](#)

## Methods

<code>MultiModes()</code>	File mode for MultiSystems.
<code>from_bond_order_system(file_name, **kwargs)</code>	Implement BondOrderSystem.from that converts from this format to BondOrderSystem.
<code>from_labeled_system(fname, **kwargs)</code>	Read from ambertools sqm.out.
<code>from_multi_systems(directory, **kwargs)</code>	Implement MultiSystems.from that converts from this format to MultiSystems.
<code>from_system(fname, **kwargs)</code>	Read from ambertools sqm.out.
<code>get_formats()</code>	Get all registered formats.
<code>get_from_methods()</code>	Get all registered from methods.
<code>get_to_methods()</code>	Get all registered to methods.
<code>mix_system(*system, type_map, **kwargs)</code>	Mix the systems into mixed_type ones according to the unified given type_map.
<code>post(func_name)</code>	Register a post function for from method.
<code>register(key)</code>	Register a format plugin.
<code>register_from(key)</code>	Register a from method if the target method name is not default.
<code>register_to(key)</code>	Register a to method if the target method name is not default.
<code>to_bond_order_system(data, rdkit_mol, *args, ...)</code>	Implement BondOrderSystem.to that converts from BondOrderSystem to this format.
<code>to_labeled_system(data, *args, **kwargs)</code>	Implement LabeledSystem.to that converts from LabeledSystem to this format.
<code>to_multi_systems(formulas, directory, **kwargs)</code>	Implement MultiSystems.to that converts from MultiSystems to this format.
<code>to_system(data, *args, **kwargs)</code>	Implement System.to that converts from System to this format.

**from\_labeled\_system**(*fname*, *\*\*kwargs*)

Read from ambertools sqm.out.

**from\_system**(*fname*, *\*\*kwargs*)

Read from ambertools sqm.out.

## dpdata.plugins.ase module

**class** `dpdata.plugins.ase.ASEDriver`(*calculator*: *ase.calculators.calculator.Calculator*)

Bases: *Driver*

ASE Driver.

### Parameters

#### **calculator**

[*ase.calculators.calculator.Calculator*] ASE calculator

### Attributes

#### **ase\_calculator**

Returns an ase calculator based on this driver.

## Methods

<code>get_driver(key)</code>	Get a driver plugin.
<code>get_drivers()</code>	Get all driver plugins.
<code>label(data)</code>	Label a system data.
<code>register(key)</code>	Register a driver plugin.

**label**(*data*: *dict*) → *dict*

Label a system data. Returns new data with energy, forces, and virials.

### Parameters

**data**  
[dict] data with coordinates and atom types

### Returns

**dict**  
labeled data with energies and forces

**class** dpdata.plugins.ase.**ASEMinimizer**(*driver*: *Driver*, *optimizer*: *Type[Optimizer]* | *None* = *None*, *fmax*: *float* = 0.005, *max\_steps*: *int* | *None* = *None*, *optimizer\_kwargs*: *dict* = {})

Bases: *Minimizer*

ASE minimizer.

### Parameters

**driver**  
[Driver] dpdata driver

**optimizer**  
[type, optional] ase optimizer class

**fmax**  
[float, optional, default=5e-3] force convergence criterion

**max\_steps**  
[int, optional] max steps to optimize

**optimizer\_kwargs**  
[dict, optional] other parameters for optimizer

## Methods

<code>get_minimizer(key)</code>	Get a minimizer plugin.
<code>get_minimizers()</code>	Get all minimizer plugins.
<code>minimize(data)</code>	Minimize the geometry.
<code>register(key)</code>	Register a minimizer plugin.

**minimize**(*data*: *dict*) → *dict*

Minimize the geometry.

### Parameters

**data**

[dict] data with coordinates and atom types

**Returns****dict**

labeled data with minimized coordinates, energies, and forces

**class** dpdata.plugins.ase.ASEStructureFormatBases: *Format*

Format for the Atomic Simulation Environment (ase).

ASE supports parsing a few dozen of data formats. As described in [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**

<code>MultiModes()</code>	File mode for MultiSystems.
<code>from_bond_order_system(file_name, **kwargs)</code>	Implement BondOrderSystem.from that converts from this format to BondOrderSystem.
<code>from_labeled_system(atoms, **kwargs)</code>	Convert ase.Atoms to a LabeledSystem.
<code>from_multi_systems(file_name[, begin, end, ...])</code>	Convert a ASE supported file to ASE Atoms.
<code>from_system(atoms, **kwargs)</code>	Convert ase.Atoms to a System.
<code>get_formats()</code>	Get all registered formats.
<code>get_from_methods()</code>	Get all registered from methods.
<code>get_to_methods()</code>	Get all registered to methods.
<code>mix_system(*system, type_map, **kwargs)</code>	Mix the systems into mixed_type ones according to the unified given type_map.
<code>post(func_name)</code>	Register a post function for from method.
<code>register(key)</code>	Register a format plugin.
<code>register_from(key)</code>	Register a from method if the target method name is not default.
<code>register_to(key)</code>	Register a to method if the target method name is not default.
<code>to_bond_order_system(data, rdkit_mol, *args, ...)</code>	Implement BondOrderSystem.to that converts from BondOrderSystem to this format.
<code>to_labeled_system(data, *args, **kwargs)</code>	Convert System to ASE Atoms object.
<code>to_multi_systems(formulas, directory, **kwargs)</code>	Implement MultiSystems.to that converts from MultiSystems to this format.
<code>to_system(data, **kwargs)</code>	Convert System to ASE Atom obj.

**from\_labeled\_system**(atoms: *ase.Atoms*, \*\*kwargs) → dict

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

**Parameters****atoms**

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

**\*\*kwargs**

[dict] other parameters

**Returns**

**dict**

data dict

### Raises

#### **RuntimeError**

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

**from\_multi\_systems**(*file\_name*: *str*, *begin*: *int* | *None* = *None*, *end*: *int* | *None* = *None*, *step*: *int* | *None* = *None*, *ase\_fmt*: *str* | *None* = *None*, *\*\*kwargs*) → *ase.Atoms*

Convert a ASE supported file to ASE Atoms.

It will finally be converted to MultiSystems.

### Parameters

#### **file\_name**

[str] path to file

#### **begin**

[int, optional] begin frame index

#### **end**

[int, optional] end frame index

#### **step**

[int, optional] frame index step

#### **ase\_fmt**

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

#### **\*\*kwargs**

[dict] other parameters

### Yields

#### **ase.Atoms**

ASE atoms in the file

**from\_system**(*atoms*: *ase.Atoms*, *\*\*kwargs*) → *dict*

Convert ase.Atoms to a System.

### Parameters

#### **atoms**

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

#### **\*\*kwargs**

[dict] other parameters

### Returns

**dict**

data dict

**to\_labeled\_system**(*data*, *\*args*, *\*\*kwargs*)

Convert System to ASE Atoms object.

**to\_system**(*data*, *\*\*kwargs*)

Convert System to ASE Atom obj.



**class** dpdata.plugins.ase.ASETrajFormatBases: *Format*

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

**Methods**

<code>MultiModes()</code>	File mode for MultiSystems.
<code>from_bond_order_system(file_name, **kwargs)</code>	Implement BondOrderSystem.from that converts from this format to BondOrderSystem.
<code>from_labeled_system(file_name[, begin, end, ...])</code>	Read ASE's trajectory file to <i>System</i> of multiple frames.
<code>from_multi_systems(directory, **kwargs)</code>	Implement MultiSystems.from that converts from this format to MultiSystems.
<code>from_system(file_name[, begin, end, step])</code>	Read ASE's trajectory file to <i>System</i> of multiple frames.
<code>get_formats()</code>	Get all registered formats.
<code>get_from_methods()</code>	Get all registered from methods.
<code>get_to_methods()</code>	Get all registered to methods.
<code>mix_system(*system, type_map, **kwargs)</code>	Mix the systems into mixed_type ones according to the unified given type_map.
<code>post(func_name)</code>	Register a post function for from method.
<code>register(key)</code>	Register a format plugin.
<code>register_from(key)</code>	Register a from method if the target method name is not default.
<code>register_to(key)</code>	Register a to method if the target method name is not default.
<code>to_bond_order_system(data, rdkit_mol, *args, ...)</code>	Implement BondOrderSystem.to that converts from BondOrderSystem to this format.
<code>to_labeled_system(data, *args, **kwargs)</code>	Implement LabeledSystem.to that converts from LabeledSystem to this format.
<code>to_multi_systems(formulas, directory, **kwargs)</code>	Implement MultiSystems.to that converts from MultiSystems to this format.
<code>to_system(data, *args, **kwargs)</code>	Implement System.to that converts from System to this format.

**from\_labeled\_system**(*file\_name*: *str*, *begin*: *int* | *None* = 0, *end*: *int* | *None* = None, *step*: *int* | *None* = 1, *\*\*kwargs*) → *dict*

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

**Parameters****file\_name**

[*str*] ASE's trajectory file

**begin**

[*int*, optional] begin frame index

**end**

[*int*, optional] end frame index

**step**

[*int*, optional] frame index step

**\*\*kwargs**  
[dict] other parameters

#### Returns

**dict\_frames: dict**  
a dictionary containing data of multiple frames

**from\_system**(*file\_name: str*, *begin: int | None = 0*, *end: int | None = None*, *step: int | None = 1*, **\*\*kwargs**)  
→ dict

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

#### Parameters

**file\_name**  
[str] ASE's trajectory file

**begin**  
[int, optional] begin frame index

**end**  
[int, optional] end frame index

**step**  
[int, optional] frame index step

**\*\*kwargs**  
[dict] other parameters

#### Returns

**dict\_frames: dict**  
a dictionary containing data of multiple frames

### dpdata.plugins.cp2k module

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

Bases: *Format*

## Methods

<code>MultiModes()</code>	File mode for MultiSystems.
<code>from_bond_order_system(file_name, **kwargs)</code>	Implement BondOrderSystem.from that converts from this format to BondOrderSystem.
<code>from_labeled_system(file_name[, restart])</code>	Implement LabeledSystem.from that converts from this format to LabeledSystem.
<code>from_multi_systems(directory, **kwargs)</code>	Implement MultiSystems.from that converts from this format to MultiSystems.
<code>from_system(file_name, **kwargs)</code>	Implement System.from that converts from this format to System.
<code>get_formats()</code>	Get all registered formats.
<code>get_from_methods()</code>	Get all registered from methods.
<code>get_to_methods()</code>	Get all registered to methods.
<code>mix_system(*system, type_map, **kwargs)</code>	Mix the systems into mixed_type ones according to the unified given type_map.
<code>post(func_name)</code>	Register a post function for from method.
<code>register(key)</code>	Register a format plugin.
<code>register_from(key)</code>	Register a from method if the target method name is not default.
<code>register_to(key)</code>	Register a to method if the target method name is not default.
<code>to_bond_order_system(data, rdkit_mol, *args, ...)</code>	Implement BondOrderSystem.to that converts from BondOrderSystem to this format.
<code>to_labeled_system(data, *args, **kwargs)</code>	Implement LabeledSystem.to that converts from LabeledSystem to this format.
<code>to_multi_systems(formulas, directory, **kwargs)</code>	Implement MultiSystems.to that converts from MultiSystems to this format.
<code>to_system(data, *args, **kwargs)</code>	Implement System.to that converts from System to this format.

**from\_labeled\_system**(*file\_name*, *restart=False*, *\*\*kwargs*)

Implement LabeledSystem.from that converts from this format to LabeledSystem.

### Parameters

**file\_name**

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

**\*\*kwargs**

[dict] keyword arguments that will be passed from the method

### Returns

**data**

[dict] system data, whose keys are defined in LabeledSystem.DTYPES

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

Bases: *Format*

## Methods

<code>MultiModes()</code>	File mode for MultiSystems.
<code>from_bond_order_system(file_name, **kwargs)</code>	Implement BondOrderSystem.from that converts from this format to BondOrderSystem.
<code>from_labeled_system(file_name[, restart])</code>	Implement LabeledSystem.from that converts from this format to LabeledSystem.
<code>from_multi_systems(directory, **kwargs)</code>	Implement MultiSystems.from that converts from this format to MultiSystems.
<code>from_system(file_name, **kwargs)</code>	Implement System.from that converts from this format to System.
<code>get_formats()</code>	Get all registered formats.
<code>get_from_methods()</code>	Get all registered from methods.
<code>get_to_methods()</code>	Get all registered to methods.
<code>mix_system(*system, type_map, **kwargs)</code>	Mix the systems into mixed_type ones according to the unified given type_map.
<code>post(func_name)</code>	Register a post function for from method.
<code>register(key)</code>	Register a format plugin.
<code>register_from(key)</code>	Register a from method if the target method name is not default.
<code>register_to(key)</code>	Register a to method if the target method name is not default.
<code>to_bond_order_system(data, rdkit_mol, *args, ...)</code>	Implement BondOrderSystem.to that converts from BondOrderSystem to this format.
<code>to_labeled_system(data, *args, **kwargs)</code>	Implement LabeledSystem.to that converts from LabeledSystem to this format.
<code>to_multi_systems(formulas, directory, **kwargs)</code>	Implement MultiSystems.to that converts from MultiSystems to this format.
<code>to_system(data, *args, **kwargs)</code>	Implement System.to that converts from System to this format.

**from\_labeled\_system**(*file\_name*, *restart=False*, *\*\*kwargs*)

Implement LabeledSystem.from that converts from this format to LabeledSystem.

### Parameters

**file\_name**

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

**\*\*kwargs**

[dict] keyword arguments that will be passed from the method

### Returns

**data**

[dict] system data, whose keys are defined in LabeledSystem.DTYPES

## dpdata.plugins.deepmd module

**class** dpdata.plugins.deepmd.DPDriver(*dp: str*)

Bases: *Driver*

DeePMD-kit driver.

### Parameters

**dp**

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

### Examples

```
>>> DPDriver("frozen_model.pb")
```

### Attributes

**ase\_calculator**

Returns an ase calculator based on this driver.

### Methods

<code>get_driver(key)</code>	Get a driver plugin.
<code>get_drivers()</code>	Get all driver plugins.
<code>label(data)</code>	Label a system data by deepmd-kit.
<code>register(key)</code>	Register a driver plugin.

**label**(*data: dict*) → *dict*

Label a system data by deepmd-kit. Returns new data with energy, forces, and virials.

### Parameters

**data**

[dict] data with coordinates and atom types

### Returns

**dict**

labeled data with energies and forces

**class** dpdata.plugins.deepmd.DeepMDCompFormat

Bases: *Format*

## Methods

<code>MultiModes()</code>	File mode for MultiSystems.
<code>from_bond_order_system(file_name, **kwargs)</code>	Implement BondOrderSystem.from that converts from this format to BondOrderSystem.
<code>from_labeled_system(file_name[, type_map])</code>	Implement LabeledSystem.from that converts from this format to LabeledSystem.
<code>from_multi_systems(directory, **kwargs)</code>	Implement MultiSystems.from that converts from this format to MultiSystems.
<code>from_system(file_name[, type_map])</code>	Implement System.from that converts from this format to System.
<code>get_formats()</code>	Get all registered formats.
<code>get_from_methods()</code>	Get all registered from methods.
<code>get_to_methods()</code>	Get all registered to methods.
<code>mix_system(*system, type_map, **kwargs)</code>	Mix the systems into mixed_type ones according to the unified given type_map.
<code>post(func_name)</code>	Register a post function for from method.
<code>register(key)</code>	Register a format plugin.
<code>register_from(key)</code>	Register a from method if the target method name is not default.
<code>register_to(key)</code>	Register a to method if the target method name is not default.
<code>to_bond_order_system(data, rdkit_mol, *args, ...)</code>	Implement BondOrderSystem.to that converts from BondOrderSystem to this format.
<code>to_labeled_system(data, *args, **kwargs)</code>	Implement LabeledSystem.to that converts from LabeledSystem to this format.
<code>to_multi_systems(formulas, directory, **kwargs)</code>	Implement MultiSystems.to that converts from MultiSystems to this format.
<code>to_system(data, file_name[, set_size, prec])</code>	Dump the system in deepmd compressed format (numpy binary) to <i>folder</i> .

**MultiMode = 1**

**from\_labeled\_system**(*file\_name*, *type\_map*=None, \*\*kwargs)

Implement LabeledSystem.from that converts from this format to LabeledSystem.

### Parameters

**file\_name**

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

**\*\*kwargs**

[dict] keyword arguments that will be passed from the method

### Returns

**data**

[dict] system data, whose keys are defined in LabeledSystem.DTYPES

**from\_system**(*file\_name*, *type\_map*=None, \*\*kwargs)

Implement System.from that converts from this format to System.

### Parameters

**file\_name**

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

**\*\*kwargs**

[dict] keyword arguments that will be passed from the method

### Returns

**data**

[dict] system data, whose keys are defined in `System.DTYPES`

**to\_system**(*data*, *file\_name*, *set\_size*=5000, *prec*=<class 'numpy.float64'>, \*\*kwargs)

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

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

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

### Parameters

**data**

[dict] System data

**file\_name**

[str] The output folder

**set\_size**

[int] The size of each set.

**prec**

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

**\*\*kwargs**

[dict] other parameters

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

Bases: *Format*

HDF5 format for DeePMD-kit.

### Examples

Dump a MultiSystems to a HDF5 file:

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

## Methods

<code>MultiModes()</code>	File mode for MultiSystems.
<code>from_bond_order_system(file_name, **kwargs)</code>	Implement BondOrderSystem.from that converts from this format to BondOrderSystem.
<code>from_labeled_system(file_name[, type_map])</code>	Convert HDF5 file to LabeledSystem data.
<code>from_multi_systems(directory, **kwargs)</code>	Generate HDF5 groups from a HDF5 file, which will be passed to <i>from_system</i> .
<code>from_system(file_name[, type_map])</code>	Convert HDF5 file to System data.
<code>get_formats()</code>	Get all registered formats.
<code>get_from_methods()</code>	Get all registered from methods.
<code>get_to_methods()</code>	Get all registered to methods.
<code>mix_system(*system, type_map, **kwargs)</code>	Mix the systems into mixed_type ones according to the unified given type_map.
<code>post(func_name)</code>	Register a post function for from method.
<code>register(key)</code>	Register a format plugin.
<code>register_from(key)</code>	Register a from method if the target method name is not default.
<code>register_to(key)</code>	Register a to method if the target method name is not default.
<code>to_bond_order_system(data, rdkit_mol, *args, ...)</code>	Implement BondOrderSystem.to that converts from BondOrderSystem to this format.
<code>to_labeled_system(data, *args, **kwargs)</code>	Implement LabeledSystem.to that converts from LabeledSystem to this format.
<code>to_multi_systems(formulas, directory, **kwargs)</code>	Generate HDF5 groups, which will be passed to <i>to_system</i> .
<code>to_system(data, file_name[, set_size, comp_prec])</code>	Convert System data to HDF5 file.

**from\_labeled\_system**(*file\_name*: *str* | *Group* | *File*, *type\_map*: *list[str]* | *None* = *None*, *\*\*kwargs*) → *dict*

Convert HDF5 file to LabeledSystem data.

### Parameters

#### **file\_name**

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

#### **type\_map**

[*dict[str]*] type map

#### **\*\*kwargs**

[*dict*] other parameters

### Returns

#### **dict**

LabeledSystem data

### Raises

#### **TypeError**

*file\_name* is not *str* or *h5py.Group* or *h5py.File*

**from\_multi\_systems**(*directory*: *str*, *\*\*kwargs*) → *Group*

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

### Parameters



**directory**

[str] HDF5 file name

**\*\*kwargs**

[dict] other parameters

**Yields****h5py.Group**

a HDF5 group in the HDF5 file

**from\_system**(*file\_name*: str | Group | File, *type\_map*: list[str] | None = None, \*\*kwargs) → dict

Convert HDF5 file to System data.

**Parameters****file\_name**

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

**type\_map**

[dict[str]] type map

**\*\*kwargs**

[dict] other parameters

**Returns****dict**

System data

**Raises****TypeError**

file\_name is not str or h5py.Group or h5py.File

**to\_multi\_systems**(*formulas*: list[str], *directory*: str, \*\*kwargs) → GroupGenerate HDF5 groups, which will be passed to *to\_system*.**Parameters****formulas**

[list[str]] formulas of MultiSystems

**directory**

[str] HDF5 file name

**\*\*kwargs**

[dict] other parameters

**Yields****h5py.Group**

a HDF5 group with the name of formula

**to\_system**(*data*: dict, *file\_name*: str | ~h5py.\_hl.group.Group | ~h5py.\_hl.files.File, *set\_size*: int = 5000, *comp\_prec*: ~numpy.dtype = <class 'numpy.float64'>, \*\*kwargs)

Convert System data to HDF5 file.

**Parameters****data**

[dict] data dict

**file\_name**

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

**set\_size**

[int, default=5000] set size

**comp\_prec**

[np.dtype] data precision

**\*\*kwargs**

[dict] other parameters

**class** dpdata.plugins.deepmd.DeepMDMixedFormat

Bases: *Format*

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

## Examples

Dump a MultiSystems into a mixed type numpy directory:

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

Load a mixed type data into a MultiSystems:

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

## Methods

<code>MultiModes()</code>	File mode for MultiSystems.
<code>from_bond_order_system(file_name, **kwargs)</code>	Implement BondOrderSystem.from that converts from this format to BondOrderSystem.
<code>from_labeled_system(file_name, **kwargs)</code>	Implement LabeledSystem.from that converts from this format to LabeledSystem.
<code>from_multi_systems(directory, **kwargs)</code>	Implement MultiSystems.from that converts from this format to MultiSystems.
<code>from_system(file_name, **kwargs)</code>	Implement System.from that converts from this format to System.
<code>get_formats()</code>	Get all registered formats.
<code>get_from_methods()</code>	Get all registered from methods.
<code>get_to_methods()</code>	Get all registered to methods.
<code>mix_system(*system, type_map, **kwargs)</code>	Mix the systems into mixed_type ones according to the unified given type_map.
<code>post(func_name)</code>	Register a post function for from method.
<code>register(key)</code>	Register a format plugin.
<code>register_from(key)</code>	Register a from method if the target method name is not default.
<code>register_to(key)</code>	Register a to method if the target method name is not default.
<code>to_bond_order_system(data, rdkit_mol, *args, ...)</code>	Implement BondOrderSystem.to that converts from BondOrderSystem to this format.
<code>to_labeled_system(data, *args, **kwargs)</code>	Implement LabeledSystem.to that converts from LabeledSystem to this format.
<code>to_multi_systems(formulas, directory, **kwargs)</code>	Implement MultiSystems.to that converts from MultiSystems to this format.
<code>to_system(data, file_name[, set_size, prec])</code>	Dump the system in deepmd mixed type format (numpy binary) to <i>folder</i> .

<b>from_labeled_system_mix</b> <b>from_system_mix</b>
--

**MultiMode = 1**

**from\_labeled\_system\_mix**(*file\_name*, *type\_map=None*, *\*\*kwargs*)

**from\_multi\_systems**(*directory*, *\*\*kwargs*)

Implement MultiSystems.from that converts from this format to MultiSystems.

By default, this method follows MultiMode to implement the conversion.

### Parameters

**directory**

[str] directory of system

**\*\*kwargs**

[dict] keyword arguments that will be passed from the method

### Returns

**filenames:** list[str]

list of filenames

**from\_system\_mix**(*file\_name*, *type\_map=None*, *\*\*kwargs*)

**mix\_system**(*\*system*, *type\_map*, *\*\*kwargs*)

Mix the systems into mixed\_type ones according to the unified given *type\_map*.

#### Parameters

**\*system**

[System] The systems to mix

**type\_map**

[list of str] Maps atom type to name

**\*\*kwargs**

[dict] other parameters

#### Returns

**mixed\_systems: dict**

dict of mixed system with key 'atom\_numbs'

**to\_system**(*data*, *file\_name*, *set\_size: int = 2000*, *prec=<class 'numpy.float64'>*, *\*\*kwargs*)

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

**The frames were already split to different systems, so these frames can be dumped to one single subfolders**

named as *folder/set.000*, containing less than *set\_size* frames.

#### Parameters

**data**

[dict] System data

**file\_name**

[str] The output folder

**set\_size**

[int, default=2000] set size

**prec**

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

**\*\*kwargs**

[dict] other parameters

**class** dpdata.plugins.deepmd.DeepMDRawFormat

Bases: [Format](#)

## Methods

<code>MultiModes()</code>	File mode for MultiSystems.
<code>from_bond_order_system(file_name, **kwargs)</code>	Implement BondOrderSystem.from that converts from this format to BondOrderSystem.
<code>from_labeled_system(file_name[, type_map])</code>	Implement LabeledSystem.from that converts from this format to LabeledSystem.
<code>from_multi_systems(directory, **kwargs)</code>	Implement MultiSystems.from that converts from this format to MultiSystems.
<code>from_system(file_name[, type_map])</code>	Implement System.from that converts from this format to System.
<code>get_formats()</code>	Get all registered formats.
<code>get_from_methods()</code>	Get all registered from methods.
<code>get_to_methods()</code>	Get all registered to methods.
<code>mix_system(*system, type_map, **kwargs)</code>	Mix the systems into mixed_type ones according to the unified given type_map.
<code>post(func_name)</code>	Register a post function for from method.
<code>register(key)</code>	Register a format plugin.
<code>register_from(key)</code>	Register a from method if the target method name is not default.
<code>register_to(key)</code>	Register a to method if the target method name is not default.
<code>to_bond_order_system(data, rdkit_mol, *args, ...)</code>	Implement BondOrderSystem.to that converts from BondOrderSystem to this format.
<code>to_labeled_system(data, *args, **kwargs)</code>	Implement LabeledSystem.to that converts from LabeledSystem to this format.
<code>to_multi_systems(formulas, directory, **kwargs)</code>	Implement MultiSystems.to that converts from MultiSystems to this format.
<code>to_system(data, file_name, **kwargs)</code>	Dump the system in deepmd raw format to directory <i>file_name</i> .

**MultiMode = 1**

**from\_labeled\_system**(*file\_name*, *type\_map=None*, *\*\*kwargs*)

Implement LabeledSystem.from that converts from this format to LabeledSystem.

### Parameters

**file\_name**

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

**\*\*kwargs**

[dict] keyword arguments that will be passed from the method

### Returns

**data**

[dict] system data, whose keys are defined in LabeledSystem.DTYPES

**from\_system**(*file\_name*, *type\_map=None*, *\*\*kwargs*)

Implement System.from that converts from this format to System.

### Parameters

**file\_name**

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

**\*\*kwargs**

[dict] keyword arguments that will be passed from the method

### Returns

**data**

[dict] system data, whose keys are defined in System.DTYPES

**to\_system**(*data*, *file\_name*, **\*\*kwargs**)

Dump the system in deepmd raw format to directory *file\_name*.

## dpdata.plugins.dftbplus module

**class** dpdata.plugins.dftbplus.DFTBplusFormat

Bases: *Format*

The DFTBplusFormat class handles files in the DFTB+ format.

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

### Attributes

None

### Methods

<b>from_labeled_system</b> ( <i>file_paths</i> , <b>**kwargs</b> ): Reads system information from files.
--

**from\_labeled\_system**(*file\_paths*, **\*\*kwargs**)

Reads system information from the given DFTB+ file paths.

### Parameters

**file\_paths**

[tuple] A tuple containing the input and output file paths. - Input file (*file\_in*): Contains information about symbols and coord. - Output file (*file\_out*): Contains information about energy and force.

**\*\*kwargs**

[dict] other parameters

## dpdata.plugins.fhi\_aims module

**class** dpdata.plugins.fhi\_aims.FhiMDFormat

Bases: *Format*

## Methods

<code>MultiModes()</code>	File mode for MultiSystems.
<code>from_bond_order_system(file_name, **kwargs)</code>	Implement BondOrderSystem.from that converts from this format to BondOrderSystem.
<code>from_labeled_system(file_name[, md, begin, ...])</code>	Implement LabeledSystem.from that converts from this format to LabeledSystem.
<code>from_multi_systems(directory, **kwargs)</code>	Implement MultiSystems.from that converts from this format to MultiSystems.
<code>from_system(file_name, **kwargs)</code>	Implement System.from that converts from this format to System.
<code>get_formats()</code>	Get all registered formats.
<code>get_from_methods()</code>	Get all registered from methods.
<code>get_to_methods()</code>	Get all registered to methods.
<code>mix_system(*system, type_map, **kwargs)</code>	Mix the systems into mixed_type ones according to the unified given type_map.
<code>post(func_name)</code>	Register a post function for from method.
<code>register(key)</code>	Register a format plugin.
<code>register_from(key)</code>	Register a from method if the target method name is not default.
<code>register_to(key)</code>	Register a to method if the target method name is not default.
<code>to_bond_order_system(data, rdkit_mol, *args, ...)</code>	Implement BondOrderSystem.to that converts from BondOrderSystem to this format.
<code>to_labeled_system(data, *args, **kwargs)</code>	Implement LabeledSystem.to that converts from LabeledSystem to this format.
<code>to_multi_systems(formulas, directory, **kwargs)</code>	Implement MultiSystems.to that converts from MultiSystems to this format.
<code>to_system(data, *args, **kwargs)</code>	Implement System.to that converts from System to this format.

**from\_labeled\_system**(*file\_name*, *md=True*, *begin=0*, *step=1*, *convergence\_check=True*, *\*\*kwargs*)

Implement LabeledSystem.from that converts from this format to LabeledSystem.

### Parameters

**file\_name**

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

**\*\*kwargs**

[dict] keyword arguments that will be passed from the method

### Returns

**data**

[dict] system data, whose keys are defined in LabeledSystem.DTYPES

**class** dpdata.plugins.fhi\_aims.FhiSCFFormat

Bases: *Format*

## Methods

<code>MultiModes()</code>	File mode for MultiSystems.
<code>from_bond_order_system(file_name, **kwargs)</code>	Implement BondOrderSystem.from that converts from this format to BondOrderSystem.
<code>from_labeled_system(file_name, **kwargs)</code>	Implement LabeledSystem.from that converts from this format to LabeledSystem.
<code>from_multi_systems(directory, **kwargs)</code>	Implement MultiSystems.from that converts from this format to MultiSystems.
<code>from_system(file_name, **kwargs)</code>	Implement System.from that converts from this format to System.
<code>get_formats()</code>	Get all registered formats.
<code>get_from_methods()</code>	Get all registered from methods.
<code>get_to_methods()</code>	Get all registered to methods.
<code>mix_system(*system, type_map, **kwargs)</code>	Mix the systems into mixed_type ones according to the unified given type_map.
<code>post(func_name)</code>	Register a post function for from method.
<code>register(key)</code>	Register a format plugin.
<code>register_from(key)</code>	Register a from method if the target method name is not default.
<code>register_to(key)</code>	Register a to method if the target method name is not default.
<code>to_bond_order_system(data, rdkit_mol, *args, ...)</code>	Implement BondOrderSystem.to that converts from BondOrderSystem to this format.
<code>to_labeled_system(data, *args, **kwargs)</code>	Implement LabeledSystem.to that converts from LabeledSystem to this format.
<code>to_multi_systems(formulas, directory, **kwargs)</code>	Implement MultiSystems.to that converts from MultiSystems to this format.
<code>to_system(data, *args, **kwargs)</code>	Implement System.to that converts from System to this format.

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

Implement LabeledSystem.from that converts from this format to LabeledSystem.

#### Parameters

##### **file\_name**

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

##### **\*\*kwargs**

[dict] keyword arguments that will be passed from the method

#### Returns

##### **data**

[dict] system data, whose keys are defined in LabeledSystem.DTYPES



**dpdata.plugins.gaussian module****class** dpdata.plugins.gaussian.GaussianGJFFormatBases: *Format*

Gaussian input file.

**Methods**

<code>MultiModes()</code>	File mode for MultiSystems.
<code>from_bond_order_system(file_name, **kwargs)</code>	Implement BondOrderSystem.from that converts from this format to BondOrderSystem.
<code>from_labeled_system(file_name, **kwargs)</code>	Implement LabeledSystem.from that converts from this format to LabeledSystem.
<code>from_multi_systems(directory, **kwargs)</code>	Implement MultiSystems.from that converts from this format to MultiSystems.
<code>from_system(file_name, **kwargs)</code>	Read Gaussian input file.
<code>get_formats()</code>	Get all registered formats.
<code>get_from_methods()</code>	Get all registered from methods.
<code>get_to_methods()</code>	Get all registered to methods.
<code>mix_system(*system, type_map, **kwargs)</code>	Mix the systems into mixed_type ones according to the unified given type_map.
<code>post(func_name)</code>	Register a post function for from method.
<code>register(key)</code>	Register a format plugin.
<code>register_from(key)</code>	Register a from method if the target method name is not default.
<code>register_to(key)</code>	Register a to method if the target method name is not default.
<code>to_bond_order_system(data, rdkit_mol, *args, ...)</code>	Implement BondOrderSystem.to that converts from BondOrderSystem to this format.
<code>to_labeled_system(data, *args, **kwargs)</code>	Implement LabeledSystem.to that converts from LabeledSystem to this format.
<code>to_multi_systems(formulas, directory, **kwargs)</code>	Implement MultiSystems.to that converts from MultiSystems to this format.
<code>to_system(data, file_name, **kwargs)</code>	Generate Gaussian input file.

**from\_system**(file\_name: *str*, \*\*kwargs)

Read Gaussian input file.

**Parameters****file\_name**

[str] file name

**\*\*kwargs**

[dict] keyword arguments

**to\_system**(data: *dict*, file\_name: *str*, \*\*kwargs)

Generate Gaussian input file.

**Parameters****data**

[dict] system data

**file\_name**  
[str] file name

**\*\*kwargs**  
[dict] Other parameters to make input files. See [dpdata.gaussian.gjf.make\\_gaussian\\_input\(\)](#)

**class** dpdata.plugins.gaussian.**GaussianDriver**(*gaussian\_exec: str = 'g16', \*\*kwargs: dict*)

Bases: [Driver](#)

Gaussian driver.

Note that “force” keyword must be added. If the number of atoms is large, “Geom=PrintInputOrient” should be added.

### Parameters

**gaussian\_exec**  
[str, default=g16] path to gaussian program

**\*\*kwargs**  
[dict] other arguments to make input files. See [dpdata.gaussian.gjf.make\\_gaussian\\_input\(\)](#)

### Examples

Use B3LYP method to calculate potential energy of a methane molecule:

```
>>> labeled_system = system.predict(keywords="force b3lyp/6-31g**", driver="gaussian")
>>> labeled_system['energies'][0]
-1102.714590995794
```

### Attributes

**ase\_calculator**  
Returns an ase calculator based on this driver.

### Methods

<code>get_driver(key)</code>	Get a driver plugin.
<code>get_drivers()</code>	Get all driver plugins.
<code>label(data)</code>	Label a system data.
<code>register(key)</code>	Register a driver plugin.

**label**(*data: dict*) → dict

Label a system data. Returns new data with energy, forces, and virials.

### Parameters

**data**  
[dict] data with coordinates and atom types

### Returns

**dict**

labeled data with energies and forces

**class** dpdata.plugins.gaussian.GaussianLogFormatBases: *Format***Methods**

<code>MultiModes()</code>	File mode for MultiSystems.
<code>from_bond_order_system(file_name, **kwargs)</code>	Implement BondOrderSystem.from that converts from this format to BondOrderSystem.
<code>from_labeled_system(file_name[, md])</code>	Implement LabeledSystem.from that converts from this format to LabeledSystem.
<code>from_multi_systems(directory, **kwargs)</code>	Implement MultiSystems.from that converts from this format to MultiSystems.
<code>from_system(file_name, **kwargs)</code>	Implement System.from that converts from this format to System.
<code>get_formats()</code>	Get all registered formats.
<code>get_from_methods()</code>	Get all registered from methods.
<code>get_to_methods()</code>	Get all registered to methods.
<code>mix_system(*system, type_map, **kwargs)</code>	Mix the systems into mixed_type ones according to the unified given type_map.
<code>post(func_name)</code>	Register a post function for from method.
<code>register(key)</code>	Register a format plugin.
<code>register_from(key)</code>	Register a from method if the target method name is not default.
<code>register_to(key)</code>	Register a to method if the target method name is not default.
<code>to_bond_order_system(data, rdkit_mol, *args, ...)</code>	Implement BondOrderSystem.to that converts from BondOrderSystem to this format.
<code>to_labeled_system(data, *args, **kwargs)</code>	Implement LabeledSystem.to that converts from LabeledSystem to this format.
<code>to_multi_systems(formulas, directory, **kwargs)</code>	Implement MultiSystems.to that converts from MultiSystems to this format.
<code>to_system(data, *args, **kwargs)</code>	Implement System.to that converts from System to this format.

**from\_labeled\_system**(file\_name, md=False, \*\*kwargs)

Implement LabeledSystem.from that converts from this format to LabeledSystem.

**Parameters****file\_name**

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

**\*\*kwargs**

[dict] keyword arguments that will be passed from the method

**Returns****data**

[dict] system data, whose keys are defined in LabeledSystem.DTYPES

**class** dpdata.plugins.gaussian.GaussianMDFormatBases: *Format*

## Methods

<code>MultiModes()</code>	File mode for MultiSystems.
<code>from_bond_order_system(file_name, **kwargs)</code>	Implement BondOrderSystem.from that converts from this format to BondOrderSystem.
<code>from_labeled_system(file_name, **kwargs)</code>	Implement LabeledSystem.from that converts from this format to LabeledSystem.
<code>from_multi_systems(directory, **kwargs)</code>	Implement MultiSystems.from that converts from this format to MultiSystems.
<code>from_system(file_name, **kwargs)</code>	Implement System.from that converts from this format to System.
<code>get_formats()</code>	Get all registered formats.
<code>get_from_methods()</code>	Get all registered from methods.
<code>get_to_methods()</code>	Get all registered to methods.
<code>mix_system(*system, type_map, **kwargs)</code>	Mix the systems into mixed_type ones according to the unified given type_map.
<code>post(func_name)</code>	Register a post function for from method.
<code>register(key)</code>	Register a format plugin.
<code>register_from(key)</code>	Register a from method if the target method name is not default.
<code>register_to(key)</code>	Register a to method if the target method name is not default.
<code>to_bond_order_system(data, rdkit_mol, *args, ...)</code>	Implement BondOrderSystem.to that converts from BondOrderSystem to this format.
<code>to_labeled_system(data, *args, **kwargs)</code>	Implement LabeledSystem.to that converts from LabeledSystem to this format.
<code>to_multi_systems(formulas, directory, **kwargs)</code>	Implement MultiSystems.to that converts from MultiSystems to this format.
<code>to_system(data, *args, **kwargs)</code>	Implement System.to that converts from System to this format.

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

Implement LabeledSystem.from that converts from this format to LabeledSystem.

#### Parameters

##### **file\_name**

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

##### **\*\*kwargs**

[dict] keyword arguments that will be passed from the method

#### Returns

##### **data**

[dict] system data, whose keys are defined in LabeledSystem.DTYPES

## dpdata.plugins.gromacs module

class dpdata.plugins.gromacs.GromacsGroFormat

Bases: *Format*

## Methods

<code>MultiModes()</code>	File mode for MultiSystems.
<code>from_bond_order_system(file_name, **kwargs)</code>	Implement BondOrderSystem.from that converts from this format to BondOrderSystem.
<code>from_labeled_system(file_name, **kwargs)</code>	Implement LabeledSystem.from that converts from this format to LabeledSystem.
<code>from_multi_systems(directory, **kwargs)</code>	Implement MultiSystems.from that converts from this format to MultiSystems.
<code>from_system(file_name[, format_atom_name])</code>	Load gromacs .gro file.
<code>get_formats()</code>	Get all registered formats.
<code>get_from_methods()</code>	Get all registered from methods.
<code>get_to_methods()</code>	Get all registered to methods.
<code>mix_system(*system, type_map, **kwargs)</code>	Mix the systems into mixed_type ones according to the unified given type_map.
<code>post(func_name)</code>	Register a post function for from method.
<code>register(key)</code>	Register a format plugin.
<code>register_from(key)</code>	Register a from method if the target method name is not default.
<code>register_to(key)</code>	Register a to method if the target method name is not default.
<code>to_bond_order_system(data, rdkit_mol, *args, ...)</code>	Implement BondOrderSystem.to that converts from BondOrderSystem to this format.
<code>to_labeled_system(data, *args, **kwargs)</code>	Implement LabeledSystem.to that converts from LabeledSystem to this format.
<code>to_multi_systems(formulas, directory, **kwargs)</code>	Implement MultiSystems.to that converts from MultiSystems to this format.
<code>to_system(data[, file_name, frame_idx])</code>	Dump the system in gromacs .gro format.

**from\_system**(file\_name, format\_atom\_name=True, \*\*kwargs)

Load gromacs .gro file.

## Parameters

**file\_name**

[str] The input file name

**format\_atom\_name**

[bool] Whether to format the atom name

**\*\*kwargs**

[dict] other parameters

**to\_system**(data, file\_name=None, frame\_idx=-1, \*\*kwargs)

Dump the system in gromacs .gro format.

## Parameters

**data**  
[dict] System data

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

**frame\_idx**  
[int] The index of the frame to dump

**\*\*kwargs**  
[dict] other parameters

## dpdata.plugins.lammps module

**class** dpdata.plugins.lammps.LAMPSDumpFormat

Bases: *Format*

### Methods

<b>MultiModes()</b>	File mode for MultiSystems.
<b>from_bond_order_system(file_name, **kwargs)</b>	Implement BondOrderSystem.from that converts from this format to BondOrderSystem.
<b>from_labeled_system(file_name, **kwargs)</b>	Implement LabeledSystem.from that converts from this format to LabeledSystem.
<b>from_multi_systems(directory, **kwargs)</b>	Implement MultiSystems.from that converts from this format to MultiSystems.
<b><i>from_system</i>(file_name[, type_map, begin, ...])</b>	Implement System.from that converts from this format to System.
<b>get_formats()</b>	Get all registered formats.
<b>get_from_methods()</b>	Get all registered from methods.
<b>get_to_methods()</b>	Get all registered to methods.
<b>mix_system(*system, type_map, **kwargs)</b>	Mix the systems into mixed_type ones according to the unified given type_map.
<b>post(func_name)</b>	Register a post function for from method.
<b>register(key)</b>	Register a format plugin.
<b>register_from(key)</b>	Register a from method if the target method name is not default.
<b>register_to(key)</b>	Register a to method if the target method name is not default.
<b>to_bond_order_system(data, rdkit_mol, *args, ...)</b>	Implement BondOrderSystem.to that converts from BondOrderSystem to this format.
<b>to_labeled_system(data, *args, **kwargs)</b>	Implement LabeledSystem.to that converts from LabeledSystem to this format.
<b>to_multi_systems(formulas, directory, **kwargs)</b>	Implement MultiSystems.to that converts from MultiSystems to this format.
<b>to_system(data, *args, **kwargs)</b>	Implement System.to that converts from System to this format.

**from\_system**(file\_name, type\_map=None, begin=0, step=1, unwrap=False, \*\*kwargs)

Implement System.from that converts from this format to System.

### Parameters

**file\_name**

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

**\*\*kwargs**

[dict] keyword arguments that will be passed from the method

**Returns****data**

[dict] system data, whose keys are defined in System.DTYPES

**class** dpdata.plugins.lammps.LAMPSLmpFormatBases: *Format***Methods**

<b>MultiModes()</b>	File mode for MultiSystems.
<b>from_bond_order_system(file_name, **kwargs)</b>	Implement BondOrderSystem.from that converts from this format to BondOrderSystem.
<b>from_labeled_system(file_name, **kwargs)</b>	Implement LabeledSystem.from that converts from this format to LabeledSystem.
<b>from_multi_systems(directory, **kwargs)</b>	Implement MultiSystems.from that converts from this format to MultiSystems.
<b>from_system(file_name[, type_map])</b>	Implement System.from that converts from this format to System.
<b>get_formats()</b>	Get all registered formats.
<b>get_from_methods()</b>	Get all registered from methods.
<b>get_to_methods()</b>	Get all registered to methods.
<b>mix_system(*system, type_map, **kwargs)</b>	Mix the systems into mixed_type ones according to the unified given type_map.
<b>post(func_name)</b>	Register a post function for from method.
<b>register(key)</b>	Register a format plugin.
<b>register_from(key)</b>	Register a from method if the target method name is not default.
<b>register_to(key)</b>	Register a to method if the target method name is not default.
<b>to_bond_order_system(data, rdkit_mol, *args, ...)</b>	Implement BondOrderSystem.to that converts from BondOrderSystem to this format.
<b>to_labeled_system(data, *args, **kwargs)</b>	Implement LabeledSystem.to that converts from LabeledSystem to this format.
<b>to_multi_systems(formulas, directory, **kwargs)</b>	Implement MultiSystems.to that converts from MultiSystems to this format.
<b>to_system(data, file_name[, frame_idx])</b>	Dump the system in lammps data format.

**from\_system**(file\_name, type\_map=None, \*\*kwargs)

Implement System.from that converts from this format to System.

**Parameters****file\_name**

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

**\*\*kwargs**

[dict] keyword arguments that will be passed from the method

### Returns

#### **data**

[dict] system data, whose keys are defined in System.DTYPES

**to\_system**(*data*, *file\_name*, *frame\_idx=0*, **\*\*kwargs**)

Dump the system in lammmps data format.

### Parameters

#### **data**

[dict] System data

#### **file\_name**

[str] The output file name

#### **frame\_idx**

[int] The index of the frame to dump

#### **\*\*kwargs**

[dict] other parameters

## dpdata.plugins.list module

**class** dpdata.plugins.list.**ListFormat**

Bases: *Format*



## Methods

<code>MultiModes()</code>	File mode for MultiSystems.
<code>from_bond_order_system(file_name, **kwargs)</code>	Implement BondOrderSystem.from that converts from this format to BondOrderSystem.
<code>from_labeled_system(file_name, **kwargs)</code>	Implement LabeledSystem.from that converts from this format to LabeledSystem.
<code>from_multi_systems(directory, **kwargs)</code>	Implement MultiSystems.from that converts from this format to MultiSystems.
<code>from_system(file_name, **kwargs)</code>	Implement System.from that converts from this format to System.
<code>get_formats()</code>	Get all registered formats.
<code>get_from_methods()</code>	Get all registered from methods.
<code>get_to_methods()</code>	Get all registered to methods.
<code>mix_system(*system, type_map, **kwargs)</code>	Mix the systems into mixed_type ones according to the unified given type_map.
<code>post(func_name)</code>	Register a post function for from method.
<code>register(key)</code>	Register a format plugin.
<code>register_from(key)</code>	Register a from method if the target method name is not default.
<code>register_to(key)</code>	Register a to method if the target method name is not default.
<code>to_bond_order_system(data, rdkit_mol, *args, ...)</code>	Implement BondOrderSystem.to that converts from BondOrderSystem to this format.
<code>to_labeled_system(data, *args, **kwargs)</code>	Implement LabeledSystem.to that converts from LabeledSystem to this format.
<code>to_multi_systems(formulas, directory, **kwargs)</code>	Implement MultiSystems.to that converts from MultiSystems to this format.
<code>to_system(data, **kwargs)</code>	Convert system to list, usefull for data collection.

`to_system(data, **kwargs)`

Convert system to list, usefull for data collection.

## dpdata.plugins.n2p2 module

**class** dpdata.plugins.n2p2.N2P2Format

Bases: *Format*

n2p2.

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

## Methods

<code>MultiModes()</code>	File mode for MultiSystems.
<code>from_bond_order_system(file_name, **kwargs)</code>	Implement BondOrderSystem.from that converts from this format to BondOrderSystem.
<code>from_labeled_system(file_name, **kwargs)</code>	Read from n2p2 format.
<code>from_multi_systems(directory, **kwargs)</code>	Implement MultiSystems.from that converts from this format to MultiSystems.
<code>from_system(file_name, **kwargs)</code>	Implement System.from that converts from this format to System.
<code>get_formats()</code>	Get all registered formats.
<code>get_from_methods()</code>	Get all registered from methods.
<code>get_to_methods()</code>	Get all registered to methods.
<code>mix_system(*system, type_map, **kwargs)</code>	Mix the systems into mixed_type ones according to the unified given type_map.
<code>post(func_name)</code>	Register a post function for from method.
<code>register(key)</code>	Register a format plugin.
<code>register_from(key)</code>	Register a from method if the target method name is not default.
<code>register_to(key)</code>	Register a to method if the target method name is not default.
<code>to_bond_order_system(data, rdkit_mol, *args, ...)</code>	Implement BondOrderSystem.to that converts from BondOrderSystem to this format.
<code>to_labeled_system(data, file_name, **kwargs)</code>	Write n2p2 format.
<code>to_multi_systems(formulas, directory, **kwargs)</code>	Implement MultiSystems.to that converts from MultiSystems to this format.
<code>to_system(data, *args, **kwargs)</code>	Implement System.to that converts from System to this format.

**from\_labeled\_system**(*file\_name*, *\*\*kwargs*)

Read from n2p2 format.

### Parameters

**file\_name**

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

**\*\*kwargs**

[dict] keyword arguments that will be passed from the method

### Returns

**data**

[dict] system data, whose keys are defined in LabeledSystem.DTYPES

**to\_labeled\_system**(*data*, *file\_name*, *\*\*kwargs*)

Write n2p2 format.

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

### Parameters

**data**

[dict] system data, whose keys are defined in LabeledSystem.DTYPES

**file\_name**

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

**\*args**

[list] arguments that will be passed from the method

**\*\*kwargs**

[dict] keyword arguments that will be passed from the method

`dpdata.plugins.n2p2.match_indices(atype1, atype2)`

## dpdata.plugins.openmx module

**class** `dpdata.plugins.openmx.OPENMXFormat`

Bases: *Format*

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

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

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

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

## Methods

<code>MultiModes()</code>	File mode for MultiSystems.
<code>from_bond_order_system(file_name, **kwargs)</code>	Implement <code>BondOrderSystem.from</code> that converts from this format to <code>BondOrderSystem</code> .
<code>from_labeled_system(file_name, **kwargs)</code>	Read from OpenMX output.
<code>from_multi_systems(directory, **kwargs)</code>	Implement <code>MultiSystems.from</code> that converts from this format to <code>MultiSystems</code> .
<code>from_system(file_name, **kwargs)</code>	Read from OpenMX output.
<code>get_formats()</code>	Get all registered formats.
<code>get_from_methods()</code>	Get all registered from methods.
<code>get_to_methods()</code>	Get all registered to methods.
<code>mix_system(*system, type_map, **kwargs)</code>	Mix the systems into <code>mixed_type</code> ones according to the unified given <code>type_map</code> .
<code>post(func_name)</code>	Register a post function for from method.
<code>register(key)</code>	Register a format plugin.
<code>register_from(key)</code>	Register a from method if the target method name is not default.
<code>register_to(key)</code>	Register a to method if the target method name is not default.
<code>to_bond_order_system(data, rdkit_mol, *args, ...)</code>	Implement <code>BondOrderSystem.to</code> that converts from <code>BondOrderSystem</code> to this format.
<code>to_labeled_system(data, *args, **kwargs)</code>	Implement <code>LabeledSystem.to</code> that converts from <code>LabeledSystem</code> to this format.
<code>to_multi_systems(formulas, directory, **kwargs)</code>	Implement <code>MultiSystems.to</code> that converts from <code>MultiSystems</code> to this format.
<code>to_system(data, *args, **kwargs)</code>	Implement <code>System.to</code> that converts from <code>System</code> to this format.

**from\_labeled\_system**(*file\_name*: *str*, *\*\*kwargs*) → *dict*

Read from OpenMX output.

**Parameters****file\_name**

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

**\*\*kwargs**

[dict] other parameters

**Returns****dict**

data dict

**from\_system**(file\_name: *str*, \*\*kwargs) → dict

Read from OpenMX output.

**Parameters****file\_name**

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

**\*\*kwargs**

[dict] other parameters

**Returns****dict**

data dict

**dpdata.plugins.orca module****class** dpdata.plugins.orca.ORBASPOutFormatBases: *Format*

ORCA single point energy output.

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

## Methods

<code>MultiModes()</code>	File mode for MultiSystems.
<code>from_bond_order_system(file_name, **kwargs)</code>	Implement BondOrderSystem.from that converts from this format to BondOrderSystem.
<code>from_labeled_system(file_name, **kwargs)</code>	Read from ORCA single point energy output.
<code>from_multi_systems(directory, **kwargs)</code>	Implement MultiSystems.from that converts from this format to MultiSystems.
<code>from_system(file_name, **kwargs)</code>	Implement System.from that converts from this format to System.
<code>get_formats()</code>	Get all registered formats.
<code>get_from_methods()</code>	Get all registered from methods.
<code>get_to_methods()</code>	Get all registered to methods.
<code>mix_system(*system, type_map, **kwargs)</code>	Mix the systems into mixed_type ones according to the unified given type_map.
<code>post(func_name)</code>	Register a post function for from method.
<code>register(key)</code>	Register a format plugin.
<code>register_from(key)</code>	Register a from method if the target method name is not default.
<code>register_to(key)</code>	Register a to method if the target method name is not default.
<code>to_bond_order_system(data, rdkit_mol, *args, ...)</code>	Implement BondOrderSystem.to that converts from BondOrderSystem to this format.
<code>to_labeled_system(data, *args, **kwargs)</code>	Implement LabeledSystem.to that converts from LabeledSystem to this format.
<code>to_multi_systems(formulas, directory, **kwargs)</code>	Implement MultiSystems.to that converts from MultiSystems to this format.
<code>to_system(data, *args, **kwargs)</code>	Implement System.to that converts from System to this format.

**from\_labeled\_system**(*file\_name*: *str*, *\*\*kwargs*) → *dict*

Read from ORCA single point energy output.

### Parameters

**file\_name**

[str] file name

**\*\*kwargs**

keyword arguments

### Returns

**dict**

system data

## dpdata.plugins.psi4 module

**class** dpdata.plugins.psi4.PSI4InputFormat

Bases: *Format*

Psi4 input file.

## Methods

<code>MultiModes()</code>	File mode for MultiSystems.
<code>from_bond_order_system(file_name, **kwargs)</code>	Implement BondOrderSystem.from that converts from this format to BondOrderSystem.
<code>from_labeled_system(file_name, **kwargs)</code>	Implement LabeledSystem.from that converts from this format to LabeledSystem.
<code>from_multi_systems(directory, **kwargs)</code>	Implement MultiSystems.from that converts from this format to MultiSystems.
<code>from_system(file_name, **kwargs)</code>	Implement System.from that converts from this format to System.
<code>get_formats()</code>	Get all registered formats.
<code>get_from_methods()</code>	Get all registered from methods.
<code>get_to_methods()</code>	Get all registered to methods.
<code>mix_system(*system, type_map, **kwargs)</code>	Mix the systems into mixed_type ones according to the unified given type_map.
<code>post(func_name)</code>	Register a post function for from method.
<code>register(key)</code>	Register a format plugin.
<code>register_from(key)</code>	Register a from method if the target method name is not default.
<code>register_to(key)</code>	Register a to method if the target method name is not default.
<code>to_bond_order_system(data, rdkit_mol, *args, ...)</code>	Implement BondOrderSystem.to that converts from BondOrderSystem to this format.
<code>to_labeled_system(data, *args, **kwargs)</code>	Implement LabeledSystem.to that converts from LabeledSystem to this format.
<code>to_multi_systems(formulas, directory, **kwargs)</code>	Implement MultiSystems.to that converts from MultiSystems to this format.
<code>to_system(data, file_name, method, basis[, ...])</code>	Write PSI4 input.

**to\_system**(data: *dict*, file\_name: *str*, method: *str*, basis: *str*, charge: *int* = 0, multiplicity: *int* = 1, frame\_idx=0, \*\*kwargs)

Write PSI4 input.

## Parameters

**data**

[dict] system data

**file\_name**

[str] file name

**method**

[str] computational method

**basis**

[str] basis set; see [https://psicode.org/psi4manual/master/basissets\\_tables.html](https://psicode.org/psi4manual/master/basissets_tables.html)

**charge**  
[int, default=0] charge of system

**multiplicity**  
[int, default=1] multiplicity of system

**frame\_idx**  
[int, default=0] The index of the frame to dump

**\*\*kwargs**  
keyword arguments

**class** dpdata.plugins.psi4.PSI4OutFormat

Bases: *Format*

Psi4 output.

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

## Methods

<code>MultiModes()</code>	File mode for MultiSystems.
<code>from_bond_order_system(file_name, **kwargs)</code>	Implement BondOrderSystem.from that converts from this format to BondOrderSystem.
<code>from_labeled_system(file_name, **kwargs)</code>	Read from Psi4 output.
<code>from_multi_systems(directory, **kwargs)</code>	Implement MultiSystems.from that converts from this format to MultiSystems.
<code>from_system(file_name, **kwargs)</code>	Implement System.from that converts from this format to System.
<code>get_formats()</code>	Get all registered formats.
<code>get_from_methods()</code>	Get all registered from methods.
<code>get_to_methods()</code>	Get all registered to methods.
<code>mix_system(*system, type_map, **kwargs)</code>	Mix the systems into mixed_type ones according to the unified given type_map.
<code>post(func_name)</code>	Register a post function for from method.
<code>register(key)</code>	Register a format plugin.
<code>register_from(key)</code>	Register a from method if the target method name is not default.
<code>register_to(key)</code>	Register a to method if the target method name is not default.
<code>to_bond_order_system(data, rdkit_mol, *args, ...)</code>	Implement BondOrderSystem.to that converts from BondOrderSystem to this format.
<code>to_labeled_system(data, *args, **kwargs)</code>	Implement LabeledSystem.to that converts from LabeledSystem to this format.
<code>to_multi_systems(formulas, directory, **kwargs)</code>	Implement MultiSystems.to that converts from MultiSystems to this format.
<code>to_system(data, *args, **kwargs)</code>	Implement System.to that converts from System to this format.

**from\_labeled\_system**(file\_name: *str*, \*\*kwargs) → dict

Read from Psi4 output.

### Parameters

**file\_name**  
[str] file name

**\*\*kwargs**  
keyword arguments

### Returns

**dict**  
system data

## dpdata.plugins.pwmat module

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

Bases: *Format*

### Methods

<b>MultiModes()</b>	File mode for MultiSystems.
<b>from_bond_order_system(file_name, **kwargs)</b>	Implement BondOrderSystem.from that converts from this format to BondOrderSystem.
<b>from_labeled_system(file_name, **kwargs)</b>	Implement LabeledSystem.from that converts from this format to LabeledSystem.
<b>from_multi_systems(directory, **kwargs)</b>	Implement MultiSystems.from that converts from this format to MultiSystems.
<b>from_system(file_name, **kwargs)</b>	Implement System.from that converts from this format to System.
<b>get_formats()</b>	Get all registered formats.
<b>get_from_methods()</b>	Get all registered from methods.
<b>get_to_methods()</b>	Get all registered to methods.
<b>mix_system(*system, type_map, **kwargs)</b>	Mix the systems into mixed_type ones according to the unified given type_map.
<b>post(func_name)</b>	Register a post function for from method.
<b>register(key)</b>	Register a format plugin.
<b>register_from(key)</b>	Register a from method if the target method name is not default.
<b>register_to(key)</b>	Register a to method if the target method name is not default.
<b>to_bond_order_system(data, rdkit_mol, *args, ...)</b>	Implement BondOrderSystem.to that converts from BondOrderSystem to this format.
<b>to_labeled_system(data, *args, **kwargs)</b>	Implement LabeledSystem.to that converts from LabeledSystem to this format.
<b>to_multi_systems(formulas, directory, **kwargs)</b>	Implement MultiSystems.to that converts from MultiSystems to this format.
<b>to_system(data, file_name[, frame_idx])</b>	Dump the system in pwmat atom.config format.

**from\_system(file\_name, \*\*kwargs)**

Implement System.from that converts from this format to System.

### Parameters

**file\_name**  
[str] file name, i.e. the first argument

**\*\*kwargs**  
[dict] keyword arguments that will be passed from the method



**Returns****data**

[dict] system data, whose keys are defined in `System.DTYPES`

**to\_system**(*data*, *file\_name*, *frame\_idx=0*, \**args*, \*\**kwargs*)

Dump the system in pwmat atom.config format.

**Parameters****data**

[dict] The system data

**file\_name**

[str] The output file name

**frame\_idx**

[int] The index of the frame to dump

**\*args**

[list] other parameters

**\*\*kwargs**

[dict] other parameters

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

Bases: [\*Format\*](#)

## Methods

<code>MultiModes()</code>	File mode for MultiSystems.
<code>from_bond_order_system(file_name, **kwargs)</code>	Implement BondOrderSystem.from that converts from this format to BondOrderSystem.
<code>from_labeled_system(file_name[, begin, ...])</code>	Implement LabeledSystem.from that converts from this format to LabeledSystem.
<code>from_multi_systems(directory, **kwargs)</code>	Implement MultiSystems.from that converts from this format to MultiSystems.
<code>from_system(file_name, **kwargs)</code>	Implement System.from that converts from this format to System.
<code>get_formats()</code>	Get all registered formats.
<code>get_from_methods()</code>	Get all registered from methods.
<code>get_to_methods()</code>	Get all registered to methods.
<code>mix_system(*system, type_map, **kwargs)</code>	Mix the systems into mixed_type ones according to the unified given type_map.
<code>post(func_name)</code>	Register a post function for from method.
<code>register(key)</code>	Register a format plugin.
<code>register_from(key)</code>	Register a from method if the target method name is not default.
<code>register_to(key)</code>	Register a to method if the target method name is not default.
<code>to_bond_order_system(data, rdkit_mol, *args, ...)</code>	Implement BondOrderSystem.to that converts from BondOrderSystem to this format.
<code>to_labeled_system(data, *args, **kwargs)</code>	Implement LabeledSystem.to that converts from LabeledSystem to this format.
<code>to_multi_systems(formulas, directory, **kwargs)</code>	Implement MultiSystems.to that converts from MultiSystems to this format.
<code>to_system(data, *args, **kwargs)</code>	Implement System.to that converts from System to this format.

**from\_labeled\_system**(*file\_name*, *begin=0*, *step=1*, *convergence\_check=True*, *\*\*kwargs*)

Implement LabeledSystem.from that converts from this format to LabeledSystem.

### Parameters

**file\_name**

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

**\*\*kwargs**

[dict] keyword arguments that will be passed from the method

### Returns

**data**

[dict] system data, whose keys are defined in LabeledSystem.DTYPES

**dpdata.plugins.pymatgen module****class** dpdata.plugins.pymatgen.PyMatgenCSEFormatBases: *Format***Methods**

<b>MultiModes()</b>	File mode for MultiSystems.
<b>from_bond_order_system(file_name, **kwargs)</b>	Implement BondOrderSystem.from that converts from this format to BondOrderSystem.
<b>from_labeled_system(file_name, **kwargs)</b>	Implement LabeledSystem.from that converts from this format to LabeledSystem.
<b>from_multi_systems(directory, **kwargs)</b>	Implement MultiSystems.from that converts from this format to MultiSystems.
<b>from_system(file_name, **kwargs)</b>	Implement System.from that converts from this format to System.
<b>get_formats()</b>	Get all registered formats.
<b>get_from_methods()</b>	Get all registered from methods.
<b>get_to_methods()</b>	Get all registered to methods.
<b>mix_system(*system, type_map, **kwargs)</b>	Mix the systems into mixed_type ones according to the unified given type_map.
<b>post(func_name)</b>	Register a post function for from method.
<b>register(key)</b>	Register a format plugin.
<b>register_from(key)</b>	Register a from method if the target method name is not default.
<b>register_to(key)</b>	Register a to method if the target method name is not default.
<b>to_bond_order_system(data, rdkit_mol, *args, ...)</b>	Implement BondOrderSystem.to that converts from BondOrderSystem to this format.
<b>to_labeled_system(data, *args, **kwargs)</b>	Convert System to Pymagen ComputedStructureEntry obj.
<b>to_multi_systems(formulas, directory, **kwargs)</b>	Implement MultiSystems.to that converts from MultiSystems to this format.
<b>to_system(data, *args, **kwargs)</b>	Implement System.to that converts from System to this format.

**to\_labeled\_system(data, \*args, \*\*kwargs)**

Convert System to Pymagen ComputedStructureEntry obj.

**class** dpdata.plugins.pymatgen.PyMatgenMoleculeFormatBases: *Format*

## Methods

<code>MultiModes()</code>	File mode for MultiSystems.
<code>from_bond_order_system(file_name, **kwargs)</code>	Implement BondOrderSystem.from that converts from this format to BondOrderSystem.
<code>from_labeled_system(file_name, **kwargs)</code>	Implement LabeledSystem.from that converts from this format to LabeledSystem.
<code>from_multi_systems(directory, **kwargs)</code>	Implement MultiSystems.from that converts from this format to MultiSystems.
<code>from_system(file_name, **kwargs)</code>	Implement System.from that converts from this format to System.
<code>get_formats()</code>	Get all registered formats.
<code>get_from_methods()</code>	Get all registered from methods.
<code>get_to_methods()</code>	Get all registered to methods.
<code>mix_system(*system, type_map, **kwargs)</code>	Mix the systems into mixed_type ones according to the unified given type_map.
<code>post(func_name)</code>	Register a post function for from method.
<code>register(key)</code>	Register a format plugin.
<code>register_from(key)</code>	Register a from method if the target method name is not default.
<code>register_to(key)</code>	Register a to method if the target method name is not default.
<code>to_bond_order_system(data, rdkit_mol, *args, ...)</code>	Implement BondOrderSystem.to that converts from BondOrderSystem to this format.
<code>to_labeled_system(data, *args, **kwargs)</code>	Implement LabeledSystem.to that converts from LabeledSystem to this format.
<code>to_multi_systems(formulas, directory, **kwargs)</code>	Implement MultiSystems.to that converts from MultiSystems to this format.
<code>to_system(data, **kwargs)</code>	Convert System to Pymatgen Molecule obj.

**from\_system**(*file\_name*, *\*\*kwargs*)

Implement System.from that converts from this format to System.

### Parameters

**file\_name**

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

**\*\*kwargs**

[dict] keyword arguments that will be passed from the method

### Returns

**data**

[dict] system data, whose keys are defined in System.DTYPES

**to\_system**(*data*, *\*\*kwargs*)

Convert System to Pymatgen Molecule obj.

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

Bases: *Format*

## Methods

<code>MultiModes()</code>	File mode for MultiSystems.
<code>from_bond_order_system(file_name, **kwargs)</code>	Implement BondOrderSystem.from that converts from this format to BondOrderSystem.
<code>from_labeled_system(file_name, **kwargs)</code>	Implement LabeledSystem.from that converts from this format to LabeledSystem.
<code>from_multi_systems(directory, **kwargs)</code>	Implement MultiSystems.from that converts from this format to MultiSystems.
<code>from_system(structure, **kwargs)</code>	Convert pymatgen.core.Structure to System.
<code>get_formats()</code>	Get all registered formats.
<code>get_from_methods()</code>	Get all registered from methods.
<code>get_to_methods()</code>	Get all registered to methods.
<code>mix_system(*system, type_map, **kwargs)</code>	Mix the systems into mixed_type ones according to the unified given type_map.
<code>post(func_name)</code>	Register a post function for from method.
<code>register(key)</code>	Register a format plugin.
<code>register_from(key)</code>	Register a from method if the target method name is not default.
<code>register_to(key)</code>	Register a to method if the target method name is not default.
<code>to_bond_order_system(data, rdkit_mol, *args, ...)</code>	Implement BondOrderSystem.to that converts from BondOrderSystem to this format.
<code>to_labeled_system(data, *args, **kwargs)</code>	Implement LabeledSystem.to that converts from LabeledSystem to this format.
<code>to_multi_systems(formulas, directory, **kwargs)</code>	Implement MultiSystems.to that converts from MultiSystems to this format.
<code>to_system(data, **kwargs)</code>	Convert System to Pymatgen Structure obj.

**from\_system**(*structure*, *\*\*kwargs*) → dict

Convert pymatgen.core.Structure to System.

### Parameters

#### **structure**

[pymatgen.core.Structure] a Pymatgen Structure, containing a structure

#### **\*\*kwargs**

[dict] other parameters

### Returns

#### **dict**

data dict

**to\_system**(*data*, *\*\*kwargs*)

Convert System to Pymatgen Structure obj.

## dpdata.plugins.qe module

**class** dpdata.plugins.qe.QECPWSCFFormat

Bases: *Format*

## Methods

<code>MultiModes()</code>	File mode for MultiSystems.
<code>from_bond_order_system(file_name, **kwargs)</code>	Implement BondOrderSystem.from that converts from this format to BondOrderSystem.
<i><code>from_labeled_system(file_name, **kwargs)</code></i>	Implement LabeledSystem.from that converts from this format to LabeledSystem.
<code>from_multi_systems(directory, **kwargs)</code>	Implement MultiSystems.from that converts from this format to MultiSystems.
<code>from_system(file_name, **kwargs)</code>	Implement System.from that converts from this format to System.
<code>get_formats()</code>	Get all registered formats.
<code>get_from_methods()</code>	Get all registered from methods.
<code>get_to_methods()</code>	Get all registered to methods.
<code>mix_system(*system, type_map, **kwargs)</code>	Mix the systems into mixed_type ones according to the unified given type_map.
<code>post(func_name)</code>	Register a post function for from method.
<code>register(key)</code>	Register a format plugin.
<code>register_from(key)</code>	Register a from method if the target method name is not default.
<code>register_to(key)</code>	Register a to method if the target method name is not default.
<code>to_bond_order_system(data, rdkit_mol, *args, ...)</code>	Implement BondOrderSystem.to that converts from BondOrderSystem to this format.
<code>to_labeled_system(data, *args, **kwargs)</code>	Implement LabeledSystem.to that converts from LabeledSystem to this format.
<code>to_multi_systems(formulas, directory, **kwargs)</code>	Implement MultiSystems.to that converts from MultiSystems to this format.
<code>to_system(data, *args, **kwargs)</code>	Implement System.to that converts from System to this format.

***from\_labeled\_system(file\_name, \*\*kwargs)***

Implement LabeledSystem.from that converts from this format to LabeledSystem.

## Parameters

**file\_name**

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

**\*\*kwargs**

[dict] keyword arguments that will be passed from the method

## Returns

**data**

[dict] system data, whose keys are defined in LabeledSystem.DTYPES

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

Bases: *Format*

## Methods

<code>MultiModes()</code>	File mode for MultiSystems.
<code>from_bond_order_system(file_name, **kwargs)</code>	Implement BondOrderSystem.from that converts from this format to BondOrderSystem.
<code>from_labeled_system(file_name[, begin, step])</code>	Implement LabeledSystem.from that converts from this format to LabeledSystem.
<code>from_multi_systems(directory, **kwargs)</code>	Implement MultiSystems.from that converts from this format to MultiSystems.
<code>from_system(file_name[, begin, step])</code>	Implement System.from that converts from this format to System.
<code>get_formats()</code>	Get all registered formats.
<code>get_from_methods()</code>	Get all registered from methods.
<code>get_to_methods()</code>	Get all registered to methods.
<code>mix_system(*system, type_map, **kwargs)</code>	Mix the systems into mixed_type ones according to the unified given type_map.
<code>post(func_name)</code>	Register a post function for from method.
<code>register(key)</code>	Register a format plugin.
<code>register_from(key)</code>	Register a from method if the target method name is not default.
<code>register_to(key)</code>	Register a to method if the target method name is not default.
<code>to_bond_order_system(data, rdkit_mol, *args, ...)</code>	Implement BondOrderSystem.to that converts from BondOrderSystem to this format.
<code>to_labeled_system(data, *args, **kwargs)</code>	Implement LabeledSystem.to that converts from LabeledSystem to this format.
<code>to_multi_systems(formulas, directory, **kwargs)</code>	Implement MultiSystems.to that converts from MultiSystems to this format.
<code>to_system(data, *args, **kwargs)</code>	Implement System.to that converts from System to this format.

**from\_labeled\_system**(*file\_name*, *begin=0*, *step=1*, *\*\*kwargs*)

Implement LabeledSystem.from that converts from this format to LabeledSystem.

### Parameters

#### **file\_name**

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

#### **\*\*kwargs**

[dict] keyword arguments that will be passed from the method

### Returns

#### **data**

[dict] system data, whose keys are defined in LabeledSystem.DTYPES

**from\_system**(*file\_name*, *begin=0*, *step=1*, *\*\*kwargs*)

Implement System.from that converts from this format to System.

### Parameters

#### **file\_name**

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

#### **\*\*kwargs**

[dict] keyword arguments that will be passed from the method

**Returns****data**

[dict] system data, whose keys are defined in System.DTYPES

**dpdata.plugins.rdkit module****class** dpdata.plugins.rdkit.**MolFormat**Bases: *Format***Methods**

<code>MultiModes()</code>	File mode for MultiSystems.
<code>from_bond_order_system(file_name, **kwargs)</code>	Implement BondOrderSystem.from that converts from this format to BondOrderSystem.
<code>from_labeled_system(file_name, **kwargs)</code>	Implement LabeledSystem.from that converts from this format to LabeledSystem.
<code>from_multi_systems(directory, **kwargs)</code>	Implement MultiSystems.from that converts from this format to MultiSystems.
<code>from_system(file_name, **kwargs)</code>	Implement System.from that converts from this format to System.
<code>get_formats()</code>	Get all registered formats.
<code>get_from_methods()</code>	Get all registered from methods.
<code>get_to_methods()</code>	Get all registered to methods.
<code>mix_system(*system, type_map, **kwargs)</code>	Mix the systems into mixed_type ones according to the unified given type_map.
<code>post(func_name)</code>	Register a post function for from method.
<code>register(key)</code>	Register a format plugin.
<code>register_from(key)</code>	Register a from method if the target method name is not default.
<code>register_to(key)</code>	Register a to method if the target method name is not default.
<code>to_bond_order_system(data, mol, file_name[, ...])</code>	Implement BondOrderSystem.to that converts from BondOrderSystem to this format.
<code>to_labeled_system(data, *args, **kwargs)</code>	Implement LabeledSystem.to that converts from LabeledSystem to this format.
<code>to_multi_systems(formulas, directory, **kwargs)</code>	Implement MultiSystems.to that converts from MultiSystems to this format.
<code>to_system(data, *args, **kwargs)</code>	Implement System.to that converts from System to this format.

**from\_bond\_order\_system**(*file\_name*, *\*\*kwargs*)

Implement BondOrderSystem.from that converts from this format to BondOrderSystem.

**Parameters****file\_name**

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

**\*\*kwargs**

[dict] keyword arguments that will be passed from the method

**Returns**



**data**

[dict] system data

**to\_bond\_order\_system**(data, mol, file\_name, frame\_idx=0, \*\*kwargs)

Implement BondOrderSystem.to that converts from BondOrderSystem to this format.

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

**Parameters****data**

[dict] system data

**rdkit\_mol**

[rdkit.Chem.rdchem.Mol] rdkit mol object

**\*args**

[list] arguments that will be passed from the method

**\*\*kwargs**

[dict] keyword arguments that will be passed from the method

**class** dpdata.plugins.rdkit.SdfFormatBases: *Format***Methods**

<b>MultiModes()</b>	File mode for MultiSystems.
<i>from_bond_order_system</i> (file_name, **kwargs)	Note that it requires all molecules in .sdf file must be of the same topology.
<i>from_labeled_system</i> (file_name, **kwargs)	Implement LabeledSystem.from that converts from this format to LabeledSystem.
<i>from_multi_systems</i> (directory, **kwargs)	Implement MultiSystems.from that converts from this format to MultiSystems.
<i>from_system</i> (file_name, **kwargs)	Implement System.from that converts from this format to System.
<i>get_formats</i> ()	Get all registered formats.
<i>get_from_methods</i> ()	Get all registered from methods.
<i>get_to_methods</i> ()	Get all registered to methods.
<i>mix_system</i> (*system, type_map, **kwargs)	Mix the systems into mixed_type ones according to the unified given type_map.
<i>post</i> (func_name)	Register a post function for from method.
<i>register</i> (key)	Register a format plugin.
<i>register_from</i> (key)	Register a from method if the target method name is not default.
<i>register_to</i> (key)	Register a to method if the target method name is not default.
<i>to_bond_order_system</i> (data, mol, file_name[, ...])	Implement BondOrderSystem.to that converts from BondOrderSystem to this format.
<i>to_labeled_system</i> (data, *args, **kwargs)	Implement LabeledSystem.to that converts from LabeledSystem to this format.
<i>to_multi_systems</i> (formulas, directory, **kwargs)	Implement MultiSystems.to that converts from MultiSystems to this format.
<i>to_system</i> (data, *args, **kwargs)	Implement System.to that converts from System to this format.

**from\_bond\_order\_system**(*file\_name*, **\*\*kwargs**)

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

**to\_bond\_order\_system**(*data*, *mol*, *file\_name*, *frame\_idx=-1*, **\*\*kwargs**)

Implement BondOrderSystem.to that converts from BondOrderSystem to this format.

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

#### Parameters

**data**

[dict] system data

**rdkit\_mol**

[rdkit.Chem.rdchem.Mol] rdkit mol object

**\*args**

[list] arguments that will be passed from the method

**\*\*kwargs**

[dict] keyword arguments that will be passed from the method

### dpdata.plugins.siesta module

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

Bases: *Format*

## Methods

<code>MultiModes()</code>	File mode for MultiSystems.
<code>from_bond_order_system(file_name, **kwargs)</code>	Implement BondOrderSystem.from that converts from this format to BondOrderSystem.
<code>from_labeled_system(file_name, **kwargs)</code>	Implement LabeledSystem.from that converts from this format to LabeledSystem.
<code>from_multi_systems(directory, **kwargs)</code>	Implement MultiSystems.from that converts from this format to MultiSystems.
<code>from_system(file_name, **kwargs)</code>	Implement System.from that converts from this format to System.
<code>get_formats()</code>	Get all registered formats.
<code>get_from_methods()</code>	Get all registered from methods.
<code>get_to_methods()</code>	Get all registered to methods.
<code>mix_system(*system, type_map, **kwargs)</code>	Mix the systems into mixed_type ones according to the unified given type_map.
<code>post(func_name)</code>	Register a post function for from method.
<code>register(key)</code>	Register a format plugin.
<code>register_from(key)</code>	Register a from method if the target method name is not default.
<code>register_to(key)</code>	Register a to method if the target method name is not default.
<code>to_bond_order_system(data, rdkit_mol, *args, ...)</code>	Implement BondOrderSystem.to that converts from BondOrderSystem to this format.
<code>to_labeled_system(data, *args, **kwargs)</code>	Implement LabeledSystem.to that converts from LabeledSystem to this format.
<code>to_multi_systems(formulas, directory, **kwargs)</code>	Implement MultiSystems.to that converts from MultiSystems to this format.
<code>to_system(data, *args, **kwargs)</code>	Implement System.to that converts from System to this format.

**from\_labeled\_system**(*file\_name*, *\*\*kwargs*)

Implement LabeledSystem.from that converts from this format to LabeledSystem.

### Parameters

**file\_name**

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

**\*\*kwargs**

[dict] keyword arguments that will be passed from the method

### Returns

**data**

[dict] system data, whose keys are defined in LabeledSystem.DTYPES

**from\_system**(*file\_name*, *\*\*kwargs*)

Implement System.from that converts from this format to System.

### Parameters

**file\_name**

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

**\*\*kwargs**

[dict] keyword arguments that will be passed from the method

**Returns****data**

[dict] system data, whose keys are defined in System.DTYPES

**class** dpdata.plugins.siesta.SiestaOutputFormatBases: *Format***Methods**

<b>MultiModes()</b>	File mode for MultiSystems.
<b>from_bond_order_system</b> (file_name, **kwargs)	Implement BondOrderSystem.from that converts from this format to BondOrderSystem.
<b>from_labeled_system</b> (file_name, **kwargs)	Implement LabeledSystem.from that converts from this format to LabeledSystem.
<b>from_multi_systems</b> (directory, **kwargs)	Implement MultiSystems.from that converts from this format to MultiSystems.
<b>from_system</b> (file_name, **kwargs)	Implement System.from that converts from this format to System.
<b>get_formats()</b>	Get all registered formats.
<b>get_from_methods()</b>	Get all registered from methods.
<b>get_to_methods()</b>	Get all registered to methods.
<b>mix_system</b> (*system, type_map, **kwargs)	Mix the systems into mixed_type ones according to the unified given type_map.
<b>post</b> (func_name)	Register a post function for from method.
<b>register</b> (key)	Register a format plugin.
<b>register_from</b> (key)	Register a from method if the target method name is not default.
<b>register_to</b> (key)	Register a to method if the target method name is not default.
<b>to_bond_order_system</b> (data, rdkit_mol, *args, ...)	Implement BondOrderSystem.to that converts from BondOrderSystem to this format.
<b>to_labeled_system</b> (data, *args, **kwargs)	Implement LabeledSystem.to that converts from LabeledSystem to this format.
<b>to_multi_systems</b> (formulas, directory, **kwargs)	Implement MultiSystems.to that converts from MultiSystems to this format.
<b>to_system</b> (data, *args, **kwargs)	Implement System.to that converts from System to this format.

**from\_labeled\_system**(file\_name, \*\*kwargs)

Implement LabeledSystem.from that converts from this format to LabeledSystem.

**Parameters****file\_name**

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

**\*\*kwargs**

[dict] keyword arguments that will be passed from the method

**Returns****data**

[dict] system data, whose keys are defined in LabeledSystem.DTYPES

**from\_system**(*file\_name*, **\*\*kwargs**)

Implement System.from that converts from this format to System.

#### Parameters

**file\_name**

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

**\*\*kwargs**

[dict] keyword arguments that will be passed from the method

#### Returns

**data**

[dict] system data, whose keys are defined in System.DTYPES

### dpdata.plugins.vasp module

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

Bases: *Format*

#### Methods

MultiModes()	File mode for MultiSystems.
from_bond_order_system( <i>file_name</i> , <b>**kwargs</b> )	Implement BondOrderSystem.from that converts from this format to BondOrderSystem.
<i>from_labeled_system</i> ( <i>file_name</i> [, <i>begin</i> , ...])	Implement LabeledSystem.from that converts from this format to LabeledSystem.
from_multi_systems( <i>directory</i> , <b>**kwargs</b> )	Implement MultiSystems.from that converts from this format to MultiSystems.
from_system( <i>file_name</i> , <b>**kwargs</b> )	Implement System.from that converts from this format to System.
get_formats()	Get all registered formats.
get_from_methods()	Get all registered from methods.
get_to_methods()	Get all registered to methods.
mix_system(* <i>system</i> , <i>type_map</i> , <b>**kwargs</b> )	Mix the systems into mixed_type ones according to the unified given type_map.
post( <i>func_name</i> )	Register a post function for from method.
register( <i>key</i> )	Register a format plugin.
register_from( <i>key</i> )	Register a from method if the target method name is not default.
register_to( <i>key</i> )	Register a to method if the target method name is not default.
to_bond_order_system( <i>data</i> , <i>rdkit_mol</i> , <i>*args</i> , ...)	Implement BondOrderSystem.to that converts from BondOrderSystem to this format.
to_labeled_system( <i>data</i> , <i>*args</i> , <b>**kwargs</b> )	Implement LabeledSystem.to that converts from LabeledSystem to this format.
to_multi_systems( <i>formulas</i> , <i>directory</i> , <b>**kwargs</b> )	Implement MultiSystems.to that converts from MultiSystems to this format.
to_system( <i>data</i> , <i>*args</i> , <b>**kwargs</b> )	Implement System.to that converts from System to this format.

**from\_labeled\_system**(*file\_name*, *begin=0*, *step=1*, *convergence\_check=True*, *\*\*kwargs*)

Implement LabeledSystem.from that converts from this format to LabeledSystem.

#### Parameters

**file\_name**

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

**\*\*kwargs**

[dict] keyword arguments that will be passed from the method

#### Returns

**data**

[dict] system data, whose keys are defined in LabeledSystem.DTYPES

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

Bases: *Format*

#### Methods

<b>MultiModes()</b>	File mode for MultiSystems.
<b>from_bond_order_system</b> ( <i>file_name</i> , <i>**kwargs</i> )	Implement BondOrderSystem.from that converts from this format to BondOrderSystem.
<b>from_labeled_system</b> ( <i>file_name</i> , <i>**kwargs</i> )	Implement LabeledSystem.from that converts from this format to LabeledSystem.
<b>from_multi_systems</b> ( <i>directory</i> , <i>**kwargs</i> )	Implement MultiSystems.from that converts from this format to MultiSystems.
<i>from_system</i> ( <i>file_name</i> , <i>**kwargs</i> )	Implement System.from that converts from this format to System.
<b>get_formats()</b>	Get all registered formats.
<b>get_from_methods()</b>	Get all registered from methods.
<b>get_to_methods()</b>	Get all registered to methods.
<b>mix_system</b> (* <i>system</i> , <i>type_map</i> , <i>**kwargs</i> )	Mix the systems into mixed_type ones according to the unified given type_map.
<b>post</b> ( <i>func_name</i> )	Register a post function for from method.
<b>register</b> ( <i>key</i> )	Register a format plugin.
<b>register_from</b> ( <i>key</i> )	Register a from method if the target method name is not default.
<b>register_to</b> ( <i>key</i> )	Register a to method if the target method name is not default.
<b>to_bond_order_system</b> ( <i>data</i> , <i>rdkit_mol</i> , <i>*args</i> , ...)	Implement BondOrderSystem.to that converts from BondOrderSystem to this format.
<b>to_labeled_system</b> ( <i>data</i> , <i>*args</i> , <i>**kwargs</i> )	Implement LabeledSystem.to that converts from LabeledSystem to this format.
<b>to_multi_systems</b> ( <i>formulas</i> , <i>directory</i> , <i>**kwargs</i> )	Implement MultiSystems.to that converts from MultiSystems to this format.
<i>to_system</i> ( <i>data</i> , <i>file_name</i> [, <i>frame_idx</i> ])	Dump the system in vasp POSCAR format.

**from\_system**(*file\_name*, *\*\*kwargs*)

Implement System.from that converts from this format to System.

#### Parameters

**file\_name**

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

**\*\*kwargs**

[dict] keyword arguments that will be passed from the method

### Returns

**data**

[dict] system data, whose keys are defined in System.DTYPES

**to\_system**(*data*, *file\_name*, *frame\_idx=0*, **\*\*kwargs**)

Dump the system in vasp POSCAR format.

### Parameters

**data**

[dict] The system data

**file\_name**

[str] The output file name

**frame\_idx**

[int] The index of the frame to dump

**\*\*kwargs**

[dict] other parameters

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

Bases: [\*Format\*](#)

## Methods

<code>MultiModes()</code>	File mode for MultiSystems.
<code>from_bond_order_system(file_name, **kwargs)</code>	Implement BondOrderSystem.from that converts from this format to BondOrderSystem.
<code>from_labeled_system(file_name, **kwargs)</code>	Implement LabeledSystem.from that converts from this format to LabeledSystem.
<code>from_multi_systems(directory, **kwargs)</code>	Implement MultiSystems.from that converts from this format to MultiSystems.
<code>from_system(file_name, **kwargs)</code>	Implement System.from that converts from this format to System.
<code>get_formats()</code>	Get all registered formats.
<code>get_from_methods()</code>	Get all registered from methods.
<code>get_to_methods()</code>	Get all registered to methods.
<code>mix_system(*system, type_map, **kwargs)</code>	Mix the systems into mixed_type ones according to the unified given type_map.
<code>post(func_name)</code>	Register a post function for from method.
<code>register(key)</code>	Register a format plugin.
<code>register_from(key)</code>	Register a from method if the target method name is not default.
<code>register_to(key)</code>	Register a to method if the target method name is not default.
<code>to_bond_order_system(data, rdkit_mol, *args, ...)</code>	Implement BondOrderSystem.to that converts from BondOrderSystem to this format.
<code>to_labeled_system(data, *args, **kwargs)</code>	Implement LabeledSystem.to that converts from LabeledSystem to this format.
<code>to_multi_systems(formulas, directory, **kwargs)</code>	Implement MultiSystems.to that converts from MultiSystems to this format.
<code>to_system(data[, frame_idx])</code>	Dump the system in vasp POSCAR format string.

**to\_system**(*data*, *frame\_idx*=0, *\*\*kwargs*)

Dump the system in vasp POSCAR format string.

### Parameters

#### **data**

[dict] The system data

#### **frame\_idx**

[int] The index of the frame to dump

#### **\*\*kwargs**

[dict] other parameters

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

Bases: *Format*



## Methods

<code>MultiModes()</code>	File mode for MultiSystems.
<code>from_bond_order_system(file_name, **kwargs)</code>	Implement BondOrderSystem.from that converts from this format to BondOrderSystem.
<code>from_labeled_system(file_name[, begin, step])</code>	Implement LabeledSystem.from that converts from this format to LabeledSystem.
<code>from_multi_systems(directory, **kwargs)</code>	Implement MultiSystems.from that converts from this format to MultiSystems.
<code>from_system(file_name, **kwargs)</code>	Implement System.from that converts from this format to System.
<code>get_formats()</code>	Get all registered formats.
<code>get_from_methods()</code>	Get all registered from methods.
<code>get_to_methods()</code>	Get all registered to methods.
<code>mix_system(*system, type_map, **kwargs)</code>	Mix the systems into mixed_type ones according to the unified given type_map.
<code>post(func_name)</code>	Register a post function for from method.
<code>register(key)</code>	Register a format plugin.
<code>register_from(key)</code>	Register a from method if the target method name is not default.
<code>register_to(key)</code>	Register a to method if the target method name is not default.
<code>to_bond_order_system(data, rdkit_mol, *args, ...)</code>	Implement BondOrderSystem.to that converts from BondOrderSystem to this format.
<code>to_labeled_system(data, *args, **kwargs)</code>	Implement LabeledSystem.to that converts from LabeledSystem to this format.
<code>to_multi_systems(formulas, directory, **kwargs)</code>	Implement MultiSystems.to that converts from MultiSystems to this format.
<code>to_system(data, *args, **kwargs)</code>	Implement System.to that converts from System to this format.

**from\_labeled\_system**(*file\_name*, *begin=0*, *step=1*, *\*\*kwargs*)

Implement LabeledSystem.from that converts from this format to LabeledSystem.

### Parameters

**file\_name**

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

**\*\*kwargs**

[dict] keyword arguments that will be passed from the method

### Returns

**data**

[dict] system data, whose keys are defined in LabeledSystem.DTYPES

**dpdata.plugins.xyz module****class** dpdata.plugins.xyz.QuipGapXYZFormatBases: *Format***Methods**

<code>MultiModes()</code>	File mode for MultiSystems.
<code>from_bond_order_system(file_name, **kwargs)</code>	Implement BondOrderSystem.from that converts from this format to BondOrderSystem.
<i><code>from_labeled_system(data, **kwargs)</code></i>	Implement LabeledSystem.from that converts from this format to LabeledSystem.
<i><code>from_multi_systems(file_name, **kwargs)</code></i>	Implement MultiSystems.from that converts from this format to MultiSystems.
<code>from_system(file_name, **kwargs)</code>	Implement System.from that converts from this format to System.
<code>get_formats()</code>	Get all registered formats.
<code>get_from_methods()</code>	Get all registered from methods.
<code>get_to_methods()</code>	Get all registered to methods.
<code>mix_system(*system, type_map, **kwargs)</code>	Mix the systems into mixed_type ones according to the unified given type_map.
<code>post(func_name)</code>	Register a post function for from method.
<code>register(key)</code>	Register a format plugin.
<code>register_from(key)</code>	Register a from method if the target method name is not default.
<code>register_to(key)</code>	Register a to method if the target method name is not default.
<code>to_bond_order_system(data, rdkit_mol, *args, ...)</code>	Implement BondOrderSystem.to that converts from BondOrderSystem to this format.
<code>to_labeled_system(data, *args, **kwargs)</code>	Implement LabeledSystem.to that converts from LabeledSystem to this format.
<code>to_multi_systems(formulas, directory, **kwargs)</code>	Implement MultiSystems.to that converts from MultiSystems to this format.
<code>to_system(data, *args, **kwargs)</code>	Implement System.to that converts from System to this format.

**from\_labeled\_system**(*data*, *\*\*kwargs*)

Implement LabeledSystem.from that converts from this format to LabeledSystem.

**Parameters****file\_name**

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

**\*\*kwargs**

[dict] keyword arguments that will be passed from the method

**Returns****data**

[dict] system data, whose keys are defined in LabeledSystem.DTYPES

**from\_multi\_systems**(*file\_name*, **\*\*kwargs**)

Implement MultiSystems.from that converts from this format to MultiSystems.

By default, this method follows MultiMode to implement the conversion.

#### Parameters

**directory**

[str] directory of system

**\*\*kwargs**

[dict] keyword arguments that will be passed from the method

#### Returns

**filenames:** list[str]

list of filenames

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

Bases: *Format*

XYZ format.

#### Examples

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

## Methods

<code>MultiModes()</code>	File mode for MultiSystems.
<code>from_bond_order_system(file_name, **kwargs)</code>	Implement BondOrderSystem.from that converts from this format to BondOrderSystem.
<code>from_labeled_system(file_name, **kwargs)</code>	Implement LabeledSystem.from that converts from this format to LabeledSystem.
<code>from_multi_systems(directory, **kwargs)</code>	Implement MultiSystems.from that converts from this format to MultiSystems.
<code>from_system(file_name, **kwargs)</code>	Implement System.from that converts from this format to System.
<code>get_formats()</code>	Get all registered formats.
<code>get_from_methods()</code>	Get all registered from methods.
<code>get_to_methods()</code>	Get all registered to methods.
<code>mix_system(*system, type_map, **kwargs)</code>	Mix the systems into mixed_type ones according to the unified given type_map.
<code>post(func_name)</code>	Register a post function for from method.
<code>register(key)</code>	Register a format plugin.
<code>register_from(key)</code>	Register a from method if the target method name is not default.
<code>register_to(key)</code>	Register a to method if the target method name is not default.
<code>to_bond_order_system(data, rdkit_mol, *args, ...)</code>	Implement BondOrderSystem.to that converts from BondOrderSystem to this format.
<code>to_labeled_system(data, *args, **kwargs)</code>	Implement LabeledSystem.to that converts from LabeledSystem to this format.
<code>to_multi_systems(formulas, directory, **kwargs)</code>	Implement MultiSystems.to that converts from MultiSystems to this format.
<code>to_system(data, file_name, **kwargs)</code>	Implement System.to that converts from System to this format.

**from\_system**(*file\_name*, *\*\*kwargs*)

Implement System.from that converts from this format to System.

### Parameters

**file\_name**

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

**\*\*kwargs**

[dict] keyword arguments that will be passed from the method

### Returns

**data**

[dict] system data, whose keys are defined in System.DTYPES

**to\_system**(*data*, *file\_name*, *\*\*kwargs*)

Implement System.to that converts from System to this format.

### Parameters

**data**

[dict] system data, whose keys are defined in System.DTYPES

**\*args**

[list] arguments that will be passed from the method

**\*\*kwargs**

[dict] keyword arguments that will be passed from the method

## dpdata.psi4 package

### Submodules

#### dpdata.psi4.input module

dpdata.psi4.input.**write\_psi4\_input**(*types: ndarray, coords: ndarray, method: str, basis: str, charge: int = 0, multiplicity: int = 1*) → str

Write Psi4 input file.

#### Parameters

##### types

[np.ndarray] atomic symbols

##### coords

[np.ndarray] atomic coordinates

##### method

[str] computational method

##### basis

[str] basis set; see [https://psicode.org/psi4manual/master/basissets\\_tables.html](https://psicode.org/psi4manual/master/basissets_tables.html)

##### charge

[int, default=0] charge of system

##### multiplicity

[int, default=1] multiplicity of system

#### Returns

##### str

content of Psi4 input file

#### dpdata.psi4.output module

dpdata.psi4.output.**read\_psi4\_output**(*fn: str*) → Tuple[str, ndarray, float, ndarray]

Read from Psi4 output.

Note that both the energy and the gradient should be printed.

#### Parameters

##### fn

[str] file name

#### Returns

##### str

atomic symbols

##### np.ndarray

atomic coordinates

**float**  
total potential energy

**np.ndarray**  
atomic forces

## dpdata.pwmat package

### Submodules

#### dpdata.pwmat.atomconfig module

dpdata.pwmat.atomconfig.**from\_system\_data**(system, f\_idx=0, skip\_zeros=True)

dpdata.pwmat.atomconfig.**to\_system\_data**(lines)

#### dpdata.pwmat.movement module

dpdata.pwmat.movement.**analyze\_block**(lines, ntot, nelm)

dpdata.pwmat.movement.**get\_frames**(fname, begin=0, step=1, convergence\_check=True)

dpdata.pwmat.movement.**get\_movement\_block**(fp)

dpdata.pwmat.movement.**system\_info**(lines, type\_idx\_zero=False)

## dpdata.pymatgen package

### Submodules

#### dpdata.pymatgen.molecule module

dpdata.pymatgen.molecule.**to\_system\_data**(file\_name, protect\_layer=9)

#### dpdata.pymatgen.structure module

dpdata.pymatgen.structure.**from\_system\_data**(structure) → dict

## dpdata.qe package

### Submodules

#### dpdata.qe.scf module

dpdata.qe.scf.**get\_block**(lines, keyword, skip=0)

dpdata.qe.scf.**get\_cell**(lines)

```

dpdata.qe.scf.get_coords(lines, cell)
dpdata.qe.scf.get_energy(lines)
dpdata.qe.scf.get_force(lines, natoms)
dpdata.qe.scf.get_frame(fname)
dpdata.qe.scf.get_stress(lines)

```

### dpdata.qe.traj module

```

dpdata.qe.traj.convert_celldm(ibrav, celldm)
dpdata.qe.traj.load_atom_names(lines, ntypes)
dpdata.qe.traj.load_atom_types(lines, natoms, atom_names)
dpdata.qe.traj.load_block(lines, key, nlines)
dpdata.qe.traj.load_cell_parameters(lines)
dpdata.qe.traj.load_celldm(lines)
dpdata.qe.traj.load_data(fname, natoms, begin=0, step=1, convert=1.0)
dpdata.qe.traj.load_energy(fname, begin=0, step=1)
dpdata.qe.traj.load_key(lines, key)
dpdata.qe.traj.load_param_file(fname)
dpdata.qe.traj.to_system_data(input_name, prefix, begin=0, step=1)
dpdata.qe.traj.to_system_label(input_name, prefix, begin=0, step=1)

```

### dpdata.rdkit package

#### Submodules

#### dpdata.rdkit.sanitize module

```

exception dpdata.rdkit.sanitize.SanitizeError(content='Sanitization Failed.')
    Bases: Exception

class dpdata.rdkit.sanitize.Sanitizer(level='medium', raise_errors=True, verbose=False)
    Bases: object

```

## Methods

<code>sanitize(mol)</code>	Sanitize mol according to <code>self.level</code> .
----------------------------	---

**sanitize**(*mol*)

Sanitize mol according to *self.level*. If failed, return None.

`dpdata.rdkit.sanitize.assign_formal_charge_for_atom(atom, verbose=False)`

Assign formal charge according to 8-electron rule for element B,C,N,O,S,P,As.

`dpdata.rdkit.sanitize.contain_hetero_aromatic(mol)`

`dpdata.rdkit.sanitize.convert_by_obabel(mol,`

*cache\_dir*='/home/docs/checkouts/readthedocs.org/user\_builds/dpdata/checkouts/le

*obabel\_path*='obabel')

`dpdata.rdkit.sanitize.get_explicit_valence(atom, verbose=False)`

`dpdata.rdkit.sanitize.get_terminal_NR2s(atom)`

`dpdata.rdkit.sanitize.get_terminal_oxygens(atom)`

`dpdata.rdkit.sanitize.is_terminal_NR2(N_atom)`

`dpdata.rdkit.sanitize.is_terminal_nitrogen(N_atom)`

`dpdata.rdkit.sanitize.is_terminal_oxygen(O_atom)`

`dpdata.rdkit.sanitize.kekulize_aromatic_heterocycles(mol_in, assign_formal_charge=True,`  
*sanitize=True)*

`dpdata.rdkit.sanitize.mol_edit_log(mol, i, j)`

`dpdata.rdkit.sanitize.print_atoms(mol)`

`dpdata.rdkit.sanitize.print_bonds(mol)`

`dpdata.rdkit.sanitize.regularize_carbon_bond_order(atom, verbose=True)`

`dpdata.rdkit.sanitize.regularize_formal_charges(mol, sanitize=True, verbose=False)`

Regularize formal charges of atoms.

`dpdata.rdkit.sanitize.regularize_nitrogen_bond_order(atom, verbose=True)`

`dpdata.rdkit.sanitize.sanitize_carboxyl(mol)`

`dpdata.rdkit.sanitize.sanitize_carboxyl_Catom(C_atom, verbose=True)`

`dpdata.rdkit.sanitize.sanitize_guanidine(mol)`

`dpdata.rdkit.sanitize.sanitize_guanidine_Catom(C_atom, verbose=True)`

`dpdata.rdkit.sanitize.sanitize_mol(mol, verbose=False)`

`dpdata.rdkit.sanitize.sanitize_nitrine_Natom(atom, verbose=True)`

`dpdata.rdkit.sanitize.sanitize_nitro(mol)`



```
dpdata.rdkit.sanitize.sanitize_nitro_Natom(N_atom, verbose=True)
dpdata.rdkit.sanitize.sanitize_phosphate(mol)
dpdata.rdkit.sanitize.sanitize_phosphate_Patom(P_atom, verbose=True)
dpdata.rdkit.sanitize.sanitize_sulfate(mol)
dpdata.rdkit.sanitize.sanitize_sulfate_Satom(S_atom, verbose=True)
dpdata.rdkit.sanitize.super_sanitize_mol(mol, name=None, verbose=True)
```

### dpdata.rdkit.utils module

```
dpdata.rdkit.utils.check_molecule_list(mols)
dpdata.rdkit.utils.check_same_atom(atom_1, atom_2)
dpdata.rdkit.utils.check_same_molecule(mol_1, mol_2)
dpdata.rdkit.utils.combine_molecules(mols)
dpdata.rdkit.utils.mol_to_system_data(mol)
dpdata.rdkit.utils.system_data_to_mol(data)
```

### dpdata.siesta package

#### Submodules

#### dpdata.siesta.aiMD\_output module

```
dpdata.siesta.aiMD_output.covert_dimension(arr, num)
dpdata.siesta.aiMD_output.extract_keyword(fout, keyword, down_line_num, begin_column,
                                           read_column_num, is_repeated_read, column_num)
dpdata.siesta.aiMD_output.get_aiMD_frame(fname)
dpdata.siesta.aiMD_output.get_atom_name(fout)
dpdata.siesta.aiMD_output.get_atom_numbs(atomtypes)
dpdata.siesta.aiMD_output.get_atom_types(fout, atomnums)
dpdata.siesta.aiMD_output.get_single_line_tail(fin, keyword, num=1)
dpdata.siesta.aiMD_output.get_virial(fout, cell)
dpdata.siesta.aiMD_output.obtain_nframe(fname)
```

## dpdata.siesta.output module

dpdata.siesta.output.**extract\_keyword**(*fout*, *keyword*, *down\_line\_num*, *begin\_column*, *column\_num*)

dpdata.siesta.output.**get\_atom\_name**(*fout*)

dpdata.siesta.output.**get\_atom\_numbs**(*atomtypes*)

dpdata.siesta.output.**get\_atom\_types**(*fout*, *atomnums*)

dpdata.siesta.output.**get\_single\_line\_tail**(*fin*, *keyword*, *num=1*)

dpdata.siesta.output.**get\_virial**(*fout*, *cells*)

dpdata.siesta.output.**obtain\_frame**(*fname*)

## dpdata.vasp package

### Submodules

#### dpdata.vasp.outcar module

dpdata.vasp.outcar.**analyze\_block**(*lines*, *ntot*, *nelm*, *ml=False*)

dpdata.vasp.outcar.**get\_frames**(*fname*, *begin=0*, *step=1*, *ml=False*, *convergence\_check=True*)

dpdata.vasp.outcar.**get\_outcar\_block**(*fp*, *ml=False*)

dpdata.vasp.outcar.**system\_info**(*lines*, *type\_idx\_zero=False*)

#### dpdata.vasp.poscar module

dpdata.vasp.poscar.**from\_system\_data**(*system*, *f\_idx=0*, *skip\_zeros=True*)

dpdata.vasp.poscar.**to\_system\_data**(*lines*)

#### dpdata.vasp.xml module

dpdata.vasp.xml.**analyze**(*fname*, *type\_idx\_zero=False*, *begin=0*, *step=1*)

Deal with broken xml file.

dpdata.vasp.xml.**analyze\_atominfo**(*atominfo\_xml*)

dpdata.vasp.xml.**analyze\_calculation**(*cc*)

dpdata.vasp.xml.**check\_name**(*item*, *name*)

dpdata.vasp.xml.**formulate\_config**(*eles*, *types*, *posi*, *cell*, *ener*, *forc*, *strs\_*)

dpdata.vasp.xml.**get\_varray**(*varray*)

## dpdata.xyz package

### Submodules

#### dpdata.xyz.quip\_gap\_xyz module

**class** dpdata.xyz.quip\_gap\_xyz.QuipGapxyzSystems(*file\_name*)

Bases: `object`

deal with QuipGapxyzFile.

#### Methods

<b>get_block_generator</b> <b>handle_single_xyz_frame</b>
--

**get\_block\_generator**()

**static handle\_single\_xyz\_frame**(*lines*)

#### dpdata.xyz.xyz module

dpdata.xyz.xyz.coord\_to\_xyz(*coord*: *ndarray*, *types*: *list*) → *str*

Convert coordinates and types to xyz format.

##### Parameters

**coord**

[*np.ndarray*] coordinates, Nx3 array

**types**

[*list*] list of types

##### Returns

**str**

xyz format string

#### Examples

```
>>> coord_to_xyz(np.ones((1,3)), ["C"])
1
```

C 1.000000 1.000000 1.000000

dpdata.xyz.xyz.xyz\_to\_coord(*xyz*: *str*) → *Tuple*[*ndarray*, *list*]

Convert xyz format to coordinates and types.

##### Parameters

**xyz**

[*str*] xyz format string

**Returns****coords**

[np.ndarray] coordinates, Nx3 array

**types**

[list] list of types

## 6.1.2 Submodules

### 6.1.3 dpdata.ase\_calculator module

### 6.1.4 dpdata.bond\_order\_system module

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

Bases: [System](#)

The system with chemical bond and formal charges information.

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

- *d\_example['bonds']*  
[a numpy array of size 4 x 3, and] the first column represents the index of begin atom, the second column represents the index of end atom, the third column represents the bond order:  
1 - single bond, 2 - double bond, 3 - triple bond, 1.5 - aromatic bond
- *d\_example['formal\_charges']*: a numpy array of size 5 x 1

**Attributes****formula**

Return the formula of this system, like C3H5O2.

**formula\_hash**

Return the hash of the formula of this system.

**nopbc****short\_formula**

Return the short formula of this system.

**short\_name**

Return the short name of this system (no more than 255 bytes), in the following order: -  
formula - short\_formula - formula\_hash.

**uniq\_formula**

Return the uniq\_formula of this system.

## Methods

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

continues on next page

Table 5 – continued from previous page

<code>from_cp2k_output(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.cp2k.CP2KOutputFormat</code> format.
<code>from_deepmd(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.deepmd.DeePMDRawFormat</code> format.
<code>from_deepmd_comp(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.deepmd.DeePMDCompFormat</code> format.
<code>from_deepmd_hdf5(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.deepmd.DeePMDHDF5Format</code> format.
<code>from_deepmd_npy(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.deepmd.DeePMDCompFormat</code> format.
<code>from_deepmd_npy_mixed(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.deepmd.DeePMDMixedFormat</code> format.
<code>from_deepmd_raw(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.deepmd.DeePMDRawFormat</code> format.
<code>from_dftbplus(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.dftbplus.DFTBplusFormat</code> format.
<code>from_dict(d)</code>	<p><b>param d</b> Dict representation.</p>
<code>from_dump(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.lammps.LAMMPSDumpFormat</code> format.
<code>from_fhi_aims_md(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.fhi_aims.FhiMDFormat</code> format.
<code>from_fhi_aims_output(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.fhi_aims.FhiMDFormat</code> format.
<code>from_fhi_aims_scf(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.fhi_aims.FhiSCFFormat</code> format.
<code>from_finalconfig(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.pwmat.PwmatAtomconfigFormat</code> format.
<code>from_gaussian_gjf(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.gaussian.GaussianGJFFormat</code> format.
<code>from_gaussian_log(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.gaussian.GaussianLogFormat</code> format.
<code>from_gaussian_md(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.gaussian.GaussianMDFormat</code> format.
<code>from_gro(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.gromacs.GromacsGroFormat</code> format.
<code>from_gromacs_gro(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.gromacs.GromacsGroFormat</code> format.
<code>from_lammps_dump(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.lammps.LAMMPSDumpFormat</code> format.
<code>from_lammps_lmp(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.lammps.LAMMPSLmpFormat</code> format.
<code>from_list(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.list.ListFormat</code> format.
<code>from_lmp(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.lammps.LAMMPSLmpFormat</code> format.
<code>from_mlmd(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.pwmat.PwmatOutputFormat</code> format.
<code>from_mol(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.rdkit.MolFormat</code> format.

continues on next page

Table 5 – continued from previous page

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

continues on next page

Table 5 – continued from previous page

<code>from_siesta_output(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.siesta.SiestaOutputFormat</code> format.
<code>from_sqm_in(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.amber.SQMInFormat</code> format.
<code>from_sqm_out(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.amber.SQMOutFormat</code> format.
<code>from_stru(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.abacus.AbacusSTRUFormat</code> format.
<code>from_vasp_contcar(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.vasp.VASPPoscarFormat</code> format.
<code>from_vasp_outcar(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.vasp.VASPOutcarFormat</code> format.
<code>from_vasp_poscar(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.vasp.VASPPoscarFormat</code> format.
<code>from_vasp_string(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.vasp.VASPStringFormat</code> format.
<code>from_vasp_xml(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.vasp.VASPXMLFormat</code> format.
<code>from_xml(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.vasp.VASPXMLFormat</code> format.
<code>from_xyz(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.xyz.XYZFormat</code> format.
<code>get_atom_names()</code>	Returns name of atoms.
<code>get_atom_nums()</code>	Returns number of atoms.
<code>get_atom_types()</code>	Returns type of atoms.
<code>get_bond_order(begin_atom_idx, end_atom_idx)</code>	Return the bond order between given atoms.
<code>get_charge()</code>	Return the total formal charge of the molecule.
<code>get_formal_charges()</code>	Return the formal charges on each atom.
<code>get_mol()</code>	Return the rdkit.Mol object.
<code>get_natoms()</code>	Returns total number of atoms in the system.
<code>get_nbonds()</code>	Return the number of bonds.
<code>get_nframes()</code>	Returns number of frames in the system.
<code>get_ntypes()</code>	Returns total number of atom types in the system.
<code>load(filename)</code>	Rebuild System obj.
<code>map_atom_types([type_map])</code>	Map the atom types of the system.
<code>minimize(*args, minimizer, **kwargs)</code>	Minimize the geometry.
<code>perturb(pert_num, cell_pert_fraction, ..., ...)</code>	Perturb each frame in the system randomly.
<code>pick_atom_idx(idx[, nopbc])</code>	Pick atom index.
<code>pick_by_amber_mask(param, maskstr[, ...])</code>	Pick atoms by amber mask.
<code>predict(*args[, driver])</code>	Predict energies and forces by a driver.
<code>register_data_type(*data_type)</code>	Register data type.
<code>remove_atom_names(atom_names)</code>	Remove atom names and all such atoms.
<code>remove_pbc([protect_layer])</code>	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 <i>protect_layer</i> (2) translates the system such that the center-of-geometry of the system locates at the center of the cell.
<code>replicate(ncopy)</code>	Replicate the each frame in the system in 3 dimensions.
<code>shuffle()</code>	Shuffle frames randomly.

continues on next page



Table 5 – continued from previous page

<code>sort_atom_names([type_map])</code>	Sort atom_names of the system and reorder atom_nums and atom_types according to atom_names.
<code>sort_atom_types()</code>	Sort atom types.
<code>sub_system(f_idx)</code>	Construct a subsystem from the system.
<code>to(fmt, *args, **kwargs)</code>	Dump systems to the specific format.
<code>to_3dmol(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.3dmol.Py3DMolFormat</code> format.
<code>to_abacus_lcao_md(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.abacus.AbacusMDFormat</code> format.
<code>to_abacus_lcao_relax(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.abacus.AbacusRelaxFormat</code> format.
<code>to_abacus_lcao_scf(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.abacus.AbacusSCFFormat</code> format.
<code>to_abacus_md(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.abacus.AbacusMDFormat</code> format.
<code>to_abacus_pw_md(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.abacus.AbacusMDFormat</code> format.
<code>to_abacus_pw_relax(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.abacus.AbacusRelaxFormat</code> format.
<code>to_abacus_pw_scf(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.abacus.AbacusSCFFormat</code> format.
<code>to_abacus_relax(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.abacus.AbacusRelaxFormat</code> format.
<code>to_abacus_scf(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.abacus.AbacusSCFFormat</code> format.
<code>to_abacus_stru(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.abacus.AbacusSTRUFormat</code> format.
<code>to_amber_md(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.amber.AmberMDFormat</code> format.
<code>to_ase_structure(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.ase.ASEStructureFormat</code> format.
<code>to_ase_traj(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.ase.ASETrajFormat</code> format.
<code>to_atomconfig(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.pwmat.PwmatAtomconfigFormat</code> format.
<code>to_contcar(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.vasp.VASPPoscarFormat</code> format.
<code>to_cp2k_aimd_output(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.cp2k.CP2KAIMDOutputFormat</code> format.
<code>to_cp2k_output(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.cp2k.CP2KOutputFormat</code> format.
<code>to_deepmd(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.deepmd.DeepMDRawFormat</code> format.
<code>to_deepmd_comp(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.deepmd.DeepMDCompFormat</code> format.
<code>to_deepmd_hdf5(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.deepmd.DeepMDHDF5Format</code> format.
<code>to_deepmd_npy(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.deepmd.DeepMDCompFormat</code> format.
<code>to_deepmd_npy_mixed(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.deepmd.DeepMDMixedFormat</code> format.

continues on next page

Table 5 – continued from previous page

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

continues on next page

Table 5 – continued from previous page

<code>to_psi4_inp(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.psi4.PSI4InputFormat</code> format.
<code>to_psi4_out(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.psi4.PSI4OutputFormat</code> format.
<code>to_pwmat_atomconfig(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.pwmat.PwmatAtomconfigFormat</code> format.
<code>to_pwmat_finalconfig(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.pwmat.PwmatAtomconfigFormat</code> format.
<code>to_pwmat_mlmd(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.pwmat.PwmatOutputFormat</code> format.
<code>to_pwmat_movement(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.pwmat.PwmatOutputFormat</code> format.
<code>to_pwmat_output(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.pwmat.PwmatOutputFormat</code> format.
<code>to_pymatgen_ComputedStructureEntry(*args, ...)</code>	Dump data to <code>dpdata.plugins.pymatgen.PyMatgenCSEFormat</code> format.
<code>to_pymatgen_computedstructureentry(*args, ...)</code>	Dump data to <code>dpdata.plugins.pymatgen.PyMatgenCSEFormat</code> format.
<code>to_pymatgen_molecule(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.pymatgen.PyMatgenMoleculeFormat</code> format.
<code>to_pymatgen_structure(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.pymatgen.PyMatgenStructureFormat</code> format.
<code>to_qe_cp_traj(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.qe.QECPTrajFormat</code> format.
<code>to_qe_pw_scf(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.qe.QECPPWSCFFormat</code> format.
<code>to_quip_gap_xyz(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.xyz.QuipGapXYZFormat</code> format.
<code>to_quip_gap_xyz_file(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.xyz.QuipGapXYZFormat</code> format.
<code>to_sdf(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.rdkit.SdfFormat</code> format.
<code>to_sdf_file(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.rdkit.SdfFormat</code> format.
<code>to_siesta_aimd_output(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.siesta.SiestaAIMDOutputFormat</code> format.
<code>to_siesta_output(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.siesta.SiestaOutputFormat</code> format.
<code>to_sqm_in(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.amber.SQMInFormat</code> format.
<code>to_sqm_out(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.amber.SQMOutFormat</code> format.
<code>to_stru(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.abacus.AbacusSTRUFormat</code> format.
<code>to_vasp_contcar(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.vasp.VASPPoscarFormat</code> format.
<code>to_vasp_outcar(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.vasp.VASPOutcarFormat</code> format.
<code>to_vasp_poscar(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.vasp.VASPPoscarFormat</code> format.
<code>to_vasp_string(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.vasp.VASPStringFormat</code> format.

continues on next page

Table 5 – continued from previous page

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

<code>affine_map</code>
<code>from_fmt</code>
<code>from_fmt_obj</code>
<code>replace</code>
<code>rot_frame_lower_triangular</code>
<code>rot_lower_triangular</code>
<code>to_fmt_obj</code>

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

`copy()`

Returns a copy of the system.

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

`from_rdkit_mol(rdkit_mol)`

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

`get_bond_order(begin_atom_idx, end_atom_idx)`

Return the bond order between given atoms.

`get_charge()`

Return the total formal charge of the molecule.

`get_formal_charges()`

Return the formal charges on each atom.

`get_mol()`

Return the rdkit.Mol object.

`get_nbonds()`

Return the number of bonds.

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

### 6.1.5 dpdata.cli module

Command line interface for dpdata.

`dpdata.cli.convert(*, from_file: str, from_format: str = 'auto', to_file: str | None = None, to_format: str | None = None, no_labeled: bool = False, multi: bool = False, type_map: list | None = None, **kwargs)`

Convert files from one format to another one.

#### Parameters

**from\_file**  
[str] read data from a file

**from\_format**  
[str] the format of from\_file

**to\_file**  
[str] dump data to a file

**to\_format**  
[str] the format of to\_file

**no\_labeled**  
[bool] labels aren't provided

**multi**  
[bool] the system contains multiple directories

**type\_map**  
[list] type map

**\*\*kwargs**  
[dict] Additional arguments for the format.

`dpdata.cli.dpdata_cli()`

Dpdata cli.

#### Examples

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

`dpdata.cli.dpdata_parser()` → [ArgumentParser](#)

Returns dpdata cli parser.

#### Returns

**argparse.ArgumentParser**  
dpdata cli parser

## 6.1.6 dpdata.data\_type module

**class** dpdata.data\_type.**AnyInt**

Bases: `int`

AnyInt equals to any other integer.

### Attributes

**denominator**

the denominator of a rational number in lowest terms

**imag**

the imaginary part of a complex number

**numerator**

the numerator of a rational number in lowest terms

**real**

the real part of a complex number

### Methods

<code>as_integer_ratio()</code>	Return a pair of integers, whose ratio is equal to the original int.
<code>bit_count()</code>	Number of ones in the binary representation of the absolute value of self.
<code>bit_length()</code>	Number of bits necessary to represent self in binary.
<code>conjugate</code>	Returns self, the complex conjugate of any int.
<code>from_bytes(/, bytes[, byteorder, signed])</code>	Return the integer represented by the given array of bytes.
<code>is_integer()</code>	Returns True.
<code>to_bytes(/[, length, byteorder, signed])</code>	Return an array of bytes representing an integer.

**class** dpdata.data\_type.**Axis**(*value, names=<not given>, \*values, module=None, qualname=None, type=None, start=1, boundary=None*)

Bases: `Enum`

Data axis.

**NATOMS** = 'natoms'

**NBONDS** = 'nbonds'

**NFRAMES** = 'nframes'

**NTYPES** = 'ntypes'

**exception** dpdata.data\_type.**DataError**

Bases: `Exception`

Data is not correct.

**class** dpdata.data\_type.**DataType**(*name: str, dtype: type, shape: Tuple[int, Axis] = None, required: bool = True*)

Bases: `object`

`DataType` represents a type of data, like coordinates, energies, etc.

#### Parameters

##### **name**

[str] name of data

##### **dtype**

[type or tuple[type]] data type, e.g. `np.ndarray`

##### **shape**

[tuple[int], optional] shape of data. Used when data is list or `np.ndarray`. Use `Axis` to represents numbers

##### **required**

[bool, default=True] whether this data is required

#### Methods

<code>check(system)</code>	Check if a system has correct data of this type.
<code>real_shape(system)</code>	Returns expected real shape of a system.

**check**(*system*: `System`)

Check if a system has correct data of this type.

#### Parameters

##### **system**

[`System`] checked system

#### Raises

##### **DataError**

type or shape of data is not correct

**real\_shape**(*system*: `System`) → `Tuple[int]`

Returns expected real shape of a system.

`dpdata.data_type.get_data_types(labeled: bool)`

Get all registered data types.

#### Parameters

##### **labeled**

[bool] whether this data type is for `LabeledSystem`

`dpdata.data_type.register_data_type(data_type: DataType, labeled: bool)`

Register a data type.

#### Parameters

##### **data\_type**

[`DataType`] data type to be registered

##### **labeled**

[bool] whether this data type is for `LabeledSystem`

## 6.1.7 dpdata.driver module

Driver plugin system.

**class** dpdata.driver.Driver(\*args, \*\*kwargs)

Bases: [ABC](#)

The base class for a driver plugin. A driver can label a pure System to generate the LabeledSystem.

**See also:**

[dpdata.plugins.deepmd.DPDriver](#)

an example of Driver

### Attributes

[ase\\_calculator](#)

Returns an ase calculator based on this driver.

### Methods

<a href="#">get_driver</a> (key)	Get a driver plugin.
<a href="#">get_drivers</a> ()	Get all driver plugins.
<a href="#">label</a> (data)	Label a system data.
<a href="#">register</a> (key)	Register a driver plugin.

**property** ase\_calculator: [ase.calculators.calculator.Calculator](#)

Returns an ase calculator based on this driver.

**static** get\_driver(key: *str*) → *Driver*

Get a driver plugin.

### Parameters

**key**

[str] key of the plugin.

### Returns

**Driver**

the specific driver class

### Raises

**RuntimeError**

if the requested driver is not implemented

**static** get\_drivers() → *dict*

Get all driver plugins.

### Returns

**dict**

dict for all driver plugin



**abstract** `label(data: dict) → dict`

Label a system data. Returns new data with energy, forces, and virials.

**Parameters**

**data**  
[dict] data with coordinates and atom types

**Returns**

**dict**  
labeled data with energies and forces

**static** `register(key: str) → Callable`

Register a driver plugin. Used as decorators.

**Parameters**

**key**  
[str] key of the plugin.

**Returns**

**Callable**  
decorator of a class

**Examples**

```
>>> @Driver.register("some_driver")
... class SomeDriver(Driver):
...     pass
```

**class** `dpdata.driver.HybridDriver(drivers: List[dict | Driver])`

Bases: `Driver`

Hybrid driver, with mixed drivers.

**Parameters**

**drivers**  
[list[dict, Driver]] list of drivers or drivers dict. For a dict, it should contain *type* as the name of the driver, and others are arguments of the driver.

**Raises**

**TypeError**  
The value of *drivers* is not a dict or *Driver*.

**Examples**

```
>>> driver = HybridDriver([
...     {"type": "sqm", "qm_theory": "DFTB3"},
...     {"type": "dp", "dp": "frozen_model.pb"},
... ])
```

This driver is the hybrid of SQM and DP.

**Attributes**

**ase\_calculator**

Returns an ase calculator based on this driver.

**Methods**

<code>get_driver(key)</code>	Get a driver plugin.
<code>get_drivers()</code>	Get all driver plugins.
<code>label(data)</code>	Label a system data.
<code>register(key)</code>	Register a driver plugin.

**label**(*data*: *dict*) → *dict*

Label a system data.

Energies and forces are the sum of those of each driver.

**Parameters****data**

[dict] data with coordinates and atom types

**Returns****dict**

labeled data with energies and forces

**class** dpdata.driver.**Minimizer**(\*args, \*\*kwargs)

Bases: [ABC](#)

The base class for a minimizer plugin. A minimizer can minimize geometry.

**Methods**

<code>get_minimizer(key)</code>	Get a minimizer plugin.
<code>get_minimizers()</code>	Get all minimizer plugins.
<code>minimize(data)</code>	Minimize the geometry.
<code>register(key)</code>	Register a minimizer plugin.

**static** **get\_minimizer**(*key*: *str*) → *Minimizer*

Get a minimizer plugin.

**Parameters****key**

[str] key of the plugin.

**Returns****Minimizer**

the specific minimizer class

**Raises****RuntimeError**

if the requested minimizer is not implemented

**static** `get_minimizers()` → `dict`

Get all minimizer plugins.

**Returns**

**dict**

dict for all minimizer plugin

**abstract** `minimize(data: dict) → dict`

Minimize the geometry.

**Parameters**

**data**

[dict] data with coordinates and atom types

**Returns**

**dict**

labeled data with minimized coordinates, energies, and forces

**static** `register(key: str) → Callable`

Register a minimizer plugin. Used as decorators.

**Parameters**

**key**

[str] key of the plugin.

**Returns**

**Callable**

decorator of a class

### Examples

```
>>> @Minimizer.register("some_minimizer")
... class SomeMinimizer(Minimizer):
...     pass
```

## 6.1.8 dpdata.format module

Implement the format plugin system.

**class** `dpdata.format.Format`

Bases: `ABC`

The abstract base class for all formats.

To add a new format, one should create a new class inherited from this class, and then

- implement several methods, such as `from_system()`;
- register the format with a key;
- add documentation in the class docstring;

The new format can be either insider or outside the package.

## Methods

<i>MultiModes()</i>	File mode for MultiSystems.
<i>from_bond_order_system</i> (file_name, **kwargs)	Implement BondOrderSystem.from that converts from this format to BondOrderSystem.
<i>from_labeled_system</i> (file_name, **kwargs)	Implement LabeledSystem.from that converts from this format to LabeledSystem.
<i>from_multi_systems</i> (directory, **kwargs)	Implement MultiSystems.from that converts from this format to MultiSystems.
<i>from_system</i> (file_name, **kwargs)	Implement System.from that converts from this format to System.
<i>get_formats</i> ()	Get all registered formats.
<i>get_from_methods</i> ()	Get all registered from methods.
<i>get_to_methods</i> ()	Get all registered to methods.
<i>mix_system</i> (*system, type_map, **kwargs)	Mix the systems into mixed_type ones according to the unified given type_map.
<i>post</i> (func_name)	Register a post function for from method.
<i>register</i> (key)	Register a format plugin.
<i>register_from</i> (key)	Register a from method if the target method name is not default.
<i>register_to</i> (key)	Register a to method if the target method name is not default.
<i>to_bond_order_system</i> (data, rdkit_mol, *args, ...)	Implement BondOrderSystem.to that converts from BondOrderSystem to this format.
<i>to_labeled_system</i> (data, *args, **kwargs)	Implement LabeledSystem.to that converts from LabeledSystem to this format.
<i>to_multi_systems</i> (formulas, directory, **kwargs)	Implement MultiSystems.to that converts from MultiSystems to this format.
<i>to_system</i> (data, *args, **kwargs)	Implement System.to that converts from System to this format.

**MultiMode = 0**

**class MultiModes**

Bases: `object`

File mode for MultiSystems.

The current implemented modes are:

- 0 (default): not implemented
- 1: every directory under the top-level directory is a system.

**Directory = 1**

**NotImplemented = 0**

**from\_bond\_order\_system**(file\_name, \*\*kwargs)

Implement BondOrderSystem.from that converts from this format to BondOrderSystem.

**Parameters**

**file\_name**

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

**\*\*kwargs**

[dict] keyword arguments that will be passed from the method

#### Returns

**data**

[dict] system data

**from\_labeled\_system**(*file\_name*, **\*\*kwargs**)

Implement LabeledSystem.from that converts from this format to LabeledSystem.

#### Parameters

**file\_name**

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

**\*\*kwargs**

[dict] keyword arguments that will be passed from the method

#### Returns

**data**

[dict] system data, whose keys are defined in LabeledSystem.DTYPES

**from\_multi\_systems**(*directory*, **\*\*kwargs**)

Implement MultiSystems.from that converts from this format to MultiSystems.

By default, this method follows MultiMode to implement the conversion.

#### Parameters

**directory**

[str] directory of system

**\*\*kwargs**

[dict] keyword arguments that will be passed from the method

#### Returns

**filenames: list[str]**

list of filenames

**from\_system**(*file\_name*, **\*\*kwargs**)

Implement System.from that converts from this format to System.

#### Parameters

**file\_name**

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

**\*\*kwargs**

[dict] keyword arguments that will be passed from the method

#### Returns

**data**

[dict] system data, whose keys are defined in System.DTYPES

**static get\_formats()**

Get all registered formats.

**static get\_from\_methods()**

Get all registered from methods.

**static get\_to\_methods()**

Get all registered to methods.

**mix\_system**(\*system, type\_map, \*\*kwargs)

Mix the systems into mixed\_type ones according to the unified given type\_map.

#### Parameters

**\*system**

[System] The systems to mix

**type\_map**

[list of str] Maps atom type to name

**\*\*kwargs**

[dict] keyword arguments that will be passed from the method

#### Returns

**mixed\_systems: dict**

dict of mixed system with key 'atom\_numbs'

**static post**(func\_name)

Register a post function for from method.

Such function will be called after the “from” method is called.

#### Parameters

**func\_name**

[str or list of str] The name of the post function.

#### Returns

**function**

The decorator function.

## Examples

Register a post function:

```
>>> @Format.post('remove_pbc')
... @Format.register('test')
... class TestFormat(Format):
...     pass
```

**static register**(key)

Register a format plugin.

By default, after a format plugin is registered, the following methods will be registered as well for `System()`, `LabeledSystem()`, `MultiSystems()`, and `BondOrderSystem()`:

- `from_{key.replace('/', '_')}`
- `to_{key.replace('/', '_')}`
- `from({key}, ...)`
- `to({key}, ...)`

The decorator should be explicitly executed before `dpdata.system` is imported. A module will be imported automatically if it

- is a submodule of `dpdata.plugins`;
- is registered at the `dpdata.plugins` entry point

#### Parameters

**key**  
[str] The key to register the plugin.

#### Returns

**function**  
The decorator function.

### Examples

Register a format plugin:

```
>>> @Format.register('test')
... @Format.register('test2')
... class TestFormat(Format):
...     pass
```

#### `static register_from(key)`

Register a from method if the target method name is not default.

#### Parameters

**key**  
[str] The key to register the plugin.

#### Returns

**function**  
The decorator function.

### Examples

Register a from method:

```
>>> @Format.register_from('from_test_haha')
... @Format.register('test')
... class TestFormat(Format):
...     pass
```

This will register a from method named `from_test_haha`, although the format name is `test`.

#### `static register_to(key)`

Register a to method if the target method name is not default.

#### Parameters

**key**  
[str] The key to register the plugin.

#### Returns

**function**  
The decorator function.

## Examples

Register a to method:

```
>>> @Format.register_to('to_test_haha')
... @Format.register('test')
... class TestFormat(Format):
...     pass
```

This will register a to method named to\_test\_haha, although the format name is test.

**to\_bond\_order\_system**(*data*, *rdkit\_mol*, *\*args*, *\*\*kwargs*)

Implement BondOrderSystem.to that converts from BondOrderSystem to this format.

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

### Parameters

**data**

[dict] system data

**rdkit\_mol**

[rdkit.Chem.rdchem.Mol] rdkit mol object

**\*args**

[list] arguments that will be passed from the method

**\*\*kwargs**

[dict] keyword arguments that will be passed from the method

**to\_labeled\_system**(*data*, *\*args*, *\*\*kwargs*)

Implement LabeledSystem.to that converts from LabeledSystem to this format.

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

### Parameters

**data**

[dict] system data, whose keys are defined in LabeledSystem.DTYPES

**\*args**

[list] arguments that will be passed from the method

**\*\*kwargs**

[dict] keyword arguments that will be passed from the method

**to\_multi\_systems**(*formulas*, *directory*, *\*\*kwargs*)

Implement MultiSystems.to that converts from MultiSystems to this format.

By default, this method follows MultiMode to implement the conversion.

### Parameters

**formulas**

[list[str]] list of formulas

**directory**

[str] directory of system

**\*\*kwargs**

[dict] keyword arguments that will be passed from the method



**to\_system**(*data*, \**args*, \*\**kwargs*)

Implement System.to that converts from System to this format.

#### Parameters

**data**

[dict] system data, whose keys are defined in System.DTYPES

**\*args**

[list] arguments that will be passed from the method

**\*\*kwargs**

[dict] keyword arguments that will be passed from the method

### 6.1.9 dpdata.periodic\_table module

**class** dpdata.periodic\_table.**Element**(*symbol: str*)

Bases: `object`

#### Attributes

**X**

**Z**

**calculated\_radius**

**mass**

**name**

**radius**

#### Methods

<b>from_Z</b>
---------------

**property X**

**property Z**

**property calculated\_radius**

**classmethod from\_Z(Z)**

**property mass**

**property name**

**property radius**

### 6.1.10 dpdata.plugin module

Base of plugin systems.

**class** dpdata.plugin.Plugin

Bases: `object`

A class to register plugins.

#### Examples

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

#### Methods

<code>register(key)</code>	Register a plugin.
----------------------------	--------------------

<code>get_plugin</code>
-------------------------

**get\_plugin**(*key*)

**register**(*key*)

Register a plugin.

#### Parameters

**key**

[str] Key of the plugin.

### 6.1.11 dpdata.stat module

**class** dpdata.stat.Errors(*system\_1: object, system\_2: object*)

Bases: `ErrorsBase`

Compute errors (deviations) between two LabeledSystems.

#### Parameters

**system\_1**

[object] system 1

**system\_2**

[object] system 2

## Examples

Get errors between referenced system and predicted system:

```
>>> e = dpdata.stat.Errors(system_1, system_2)
>>> print("%.4f %.4f %.4f %.4f" % (e.e_mae, e.e_rmse, e.f_mae, e.f_rmse))
```

### Attributes

#### **e\_errors**

Energy errors.

#### **e\_mae**

Energy MAE.

#### **e\_rmse**

Energy RMSE.

#### **f\_errors**

Force errors.

#### **f\_mae**

Force MAE.

#### **f\_rmse**

Force RMSE.

### Methods

*SYSTEM\_TYPE*

alias of *LabeledSystem*

### SYSTEM\_TYPE

alias of *LabeledSystem*

**property e\_errors:** *ndarray*

Energy errors.

**property f\_errors:** *ndarray*

Force errors.

**class** dpdata.stat.**ErrorsBase**(*system\_1: object, system\_2: object*)

Bases: *object*

Compute errors (deviations) between two systems. The type of system is assigned by SYSTEM\_TYPE.

### Parameters

#### **system\_1**

[object] system 1

#### **system\_2**

[object] system 2

### Attributes

#### **e\_errors**

Energy errors.

**`e_mae`**  
Energy MAE.

**`e_rmse`**  
Energy RMSE.

**`f_errors`**  
Force errors.

**`f_mae`**  
Force MAE.

**`f_rmse`**  
Force RMSE.

## Methods

<b><code>SYSTEM_TYPE</code></b>	alias of <code>object</code>
---------------------------------	------------------------------

### **`SYSTEM_TYPE`**

alias of `object`

**abstract property `e_errors`:** `ndarray`

Energy errors.

**property `e_mae`:** `float64`

Energy MAE.

**property `e_rmse`:** `float64`

Energy RMSE.

**abstract property `f_errors`:** `ndarray`

Force errors.

**property `f_mae`:** `float64`

Force MAE.

**property `f_rmse`:** `float64`

Force RMSE.

**class** `dpdata.stat.MultiErrors`(*system\_1*: `object`, *system\_2*: `object`)

Bases: `ErrorsBase`

Compute errors (deviations) between two MultiSystems.

### **Parameters**

**`system_1`**  
[object] system 1

**`system_2`**  
[object] system 2

## Examples

Get errors between referenced system and predicted system:

```
>>> e = dpdata.stat.MultiErrors(system_1, system_2)
>>> print("%.4f %.4f %.4f %.4f" % (e.e_mae, e.e_rmse, e.f_mae, e.f_rmse))
```

### Attributes

#### **e\_errors**

Energy errors.

#### **e\_mae**

Energy MAE.

#### **e\_rmse**

Energy RMSE.

#### **f\_errors**

Force errors.

#### **f\_mae**

Force MAE.

#### **f\_rmse**

Force RMSE.

## Methods

*SYSTEM\_TYPE*

alias of *MultiSystems*

### SYSTEM\_TYPE

alias of *MultiSystems*

**property e\_errors:** *ndarray*

Energy errors.

**property f\_errors:** *ndarray*

Force errors.

**dpdata.stat.mae**(errors: *ndarray*) → float64

Compute the mean absolute error (MAE).

#### Parameters

##### **errors**

[np.ndarray] errors between two values

#### Returns

##### **np.float64**

mean absolute error (MAE)

**dpdata.stat.rmse**(errors: *ndarray*) → float64

Compute the root mean squared error (RMSE).

#### Parameters

**errors**

[np.ndarray] errors between two values

**Returns****np.float64**

root mean squared error (RMSE)

## 6.1.12 dpdata.system module

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

Bases: [System](#)

The labeled data System.

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

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

**It is noted that**

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

**Parameters****file\_name**

[str] The file to load the system

**fmt**

[str]

**Format of the file, supported formats are**

- auto: inferred from *file\_name*'s extension
- vasp/xml: vasp xml
- vasp/outcar: vasp OUTCAR
- deepmd/raw: deepmd-kit raw
- deepmd/npz: deepmd-kit compressed format (numpy binary)
- qe/cp/traj: Quantum Espresso CP trajectory files. should have: *file\_name*+'in', *file\_name*+'pos', *file\_name*+'evp' and *file\_name*+'for'
- qe/pw/scf: Quantum Espresso PW single point calculations. Both input and output files are required. If *file\_name* is a string, it denotes the output file name. Input file name is obtained by replacing 'out' by 'in' from *file\_name*. Or *file\_name* is a list, with the first element being the input file name and the second element being the output filename.
- siesta/output: siesta SCF output file
- siesta/aimd\_output: siesta aimd output file

- `gaussian/log`: gaussian logs
- `gaussian/md`: gaussian ab initio molecular dynamics
- `cp2k/output`: cp2k output file
- `cp2k/aimd_output`: cp2k aimd output dir(contains *pos.xyz* and \*.log file); optional *restart=True* if it is a cp2k restarted task.
- `pwmat/movement`: pwmat md output file
- `pwmat/out.mlmd`: pwmat scf output file

**type\_map**

[list of str] Maps atom type to name. The atom with type *ii* is mapped to *type\_map[ii]*. If not provided the atom names are assigned to 'Type\_1', 'Type\_2', 'Type\_3'...

**begin**

[int] The beginning frame when loading MD trajectory.

**step**

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

**Attributes****formula**

Return the formula of this system, like C3H5O2.

**formula\_hash**

Return the hash of the formula of this system.

**nopbc****short\_formula**

Return the short formula of this system.

**short\_name**

Return the short name of this system (no more than 255 bytes), in the following order: - formula - short\_formula - formula\_hash.

**uniq\_formula**

Return the uniq\_formula of this system.

**Methods**

<code>add_atom_names(atom_names)</code>	Add atom_names that do not exist.
<code>append(system)</code>	Append a system to this system.
<code>apply_pbc()</code>	Append periodic boundary condition.
<code>apply_type_map(type_map)</code>	Customize the element symbol order and it should maintain order consistency in dpnpy or deepmd-kit.
<code>as_dict()</code>	Returns data dict of System instance.
<code>check_data()</code>	Check if data is correct.
<code>check_type_map(type_map)</code>	Assign atom_names to type_map if type_map is given and different from atom_names.
<code>convert_to_mixed_type([type_map])</code>	Convert the data dict to mixed type format structure, in order to append systems with different formula but the same number of atoms.
<code>copy()</code>	Returns a copy of the system.

continues on next page

Table 6 – continued from previous page

<code>correction(hl_sys)</code>	Get energy and force correction between self and a high-level LabeledSystem.
<code>dump(filename[, indent])</code>	Dump .json or .yaml file.
<code>extend(systems)</code>	Extend a system list to this system.
<code>from_3dmol(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.3dmol.Py3DMolFormat</code> format.
<code>from_abacus_lcao_md(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.abacus.AbacusMDFormat</code> format.
<code>from_abacus_lcao_relax(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.abacus.AbacusRelaxFormat</code> format.
<code>from_abacus_lcao_scf(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.abacus.AbacusSCFFormat</code> format.
<code>from_abacus_md(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.abacus.AbacusMDFormat</code> format.
<code>from_abacus_pw_md(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.abacus.AbacusMDFormat</code> format.
<code>from_abacus_pw_relax(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.abacus.AbacusRelaxFormat</code> format.
<code>from_abacus_pw_scf(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.abacus.AbacusSCFFormat</code> format.
<code>from_abacus_relax(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.abacus.AbacusRelaxFormat</code> format.
<code>from_abacus_scf(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.abacus.AbacusSCFFormat</code> format.
<code>from_abacus_stru(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.abacus.AbacusSTRUFormat</code> format.
<code>from_amber_md(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.amber.AmberMDFormat</code> format.
<code>from_ase_structure(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.ase.ASEStructureFormat</code> format.
<code>from_ase_traj(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.ase.ASETrajFormat</code> format.
<code>from_atomconfig(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.pwmat.PwmatAtomconfigFormat</code> format.
<code>from_contcar(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.vasp.VASPPoscarFormat</code> format.
<code>from_cp2k_aimd_output(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.cp2k.CP2KAIMDOutputFormat</code> format.
<code>from_cp2k_output(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.cp2k.CP2KOutputFormat</code> format.
<code>from_deepmd(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.deepmd.DeepMDRawFormat</code> format.
<code>from_deepmd_comp(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.deepmd.DeepMDCompFormat</code> format.
<code>from_deepmd_hdf5(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.deepmd.DeepMDHDF5Format</code> format.
<code>from_deepmd_npy(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.deepmd.DeepMDCompFormat</code> format.
<code>from_deepmd_npy_mixed(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.deepmd.DeepMDMixedFormat</code> format.
<code>from_deepmd_raw(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.deepmd.DeepMDRawFormat</code> format.

continues on next page



Table 6 – continued from previous page

<code>from_dftbplus(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.dftbplus.DFTBplusFormat</code> format.
<code>from_dict(d)</code>	<p><b>param d</b> Dict representation.</p>
<code>from_dump(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.lammps.LAMMPSDumpFormat</code> format.
<code>from_fhi_aims_md(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.fhi_aims.FhiMDFormat</code> format.
<code>from_fhi_aims_output(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.fhi_aims.FhiMDFormat</code> format.
<code>from_fhi_aims_scf(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.fhi_aims.FhiSCFFormat</code> format.
<code>from_finalconfig(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.pwmat.PwmatAtomconfigFormat</code> format.
<code>from_gaussian_gjf(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.gaussian.GaussianGJFFormat</code> format.
<code>from_gaussian_log(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.gaussian.GaussianLogFormat</code> format.
<code>from_gaussian_md(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.gaussian.GaussianMDFormat</code> format.
<code>from_gro(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.gromacs.GromacsGroFormat</code> format.
<code>from_gromacs_gro(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.gromacs.GromacsGroFormat</code> format.
<code>from_lammps_dump(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.lammps.LAMMPSDumpFormat</code> format.
<code>from_lammps_lmp(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.lammps.LAMMPSLmpFormat</code> format.
<code>from_list(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.list.ListFormat</code> format.
<code>from_lmp(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.lammps.LAMMPSLmpFormat</code> format.
<code>from_mlmd(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.pwmat.PwmatOutputFormat</code> format.
<code>from_mol(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.rdkit.MolFormat</code> format.
<code>from_mol_file(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.rdkit.MolFormat</code> format.
<code>from_movement(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.pwmat.PwmatOutputFormat</code> format.
<code>from_n2p2(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.n2p2.N2P2Format</code> format.
<code>from_openmx_md(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.openmx.OPENMXFormat</code> format.
<code>from_orca_spout(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.orca.ORBASPOutFormat</code> format.
<code>from_outcar(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.vasp.VASPOutcarFormat</code> format.
<code>from_poscar(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.vasp.VASPPoscarFormat</code> format.

continues on next page

Table 6 – continued from previous page

<i>from_psi4_inp</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.psi4.PSI4InputFormat</i> format.
<i>from_psi4_out</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.psi4.PSI4OutputFormat</i> format.
<i>from_pwmat_atomconfig</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.pwmat.PwmatAtomconfigFormat</i> format.
<i>from_pwmat_finalconfig</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.pwmat.PwmatAtomconfigFormat</i> format.
<i>from_pwmat_mlmd</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.pwmat.PwmatOutputFormat</i> format.
<i>from_pwmat_movement</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.pwmat.PwmatOutputFormat</i> format.
<i>from_pwmat_output</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.pwmat.PwmatOutputFormat</i> format.
<i>from_pymatgen_computedstructureentry</i> (...)	Read data from <i>dpdata.plugins.pymatgen.PyMatgenCSEFormat</i> format.
<i>from_pymatgen_molecule</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.pymatgen.PyMatgenMoleculeFormat</i> format.
<i>from_pymatgen_structure</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.pymatgen.PyMatgenStructureFormat</i> format.
<i>from_qe_cp_traj</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.qe.QECPTrajFormat</i> format.
<i>from_qe_pw_scf</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.qe.QECPPWSCFFormat</i> format.
<i>from_quip_gap_xyz</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.xyz.QuipGapXYZFormat</i> format.
<i>from_quip_gap_xyz_file</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.xyz.QuipGapXYZFormat</i> format.
<i>from_sdf</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.rdkit.SdfFormat</i> format.
<i>from_sdf_file</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.rdkit.SdfFormat</i> format.
<i>from_siesta_aiMD_output</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.siesta.SiestaAIMDOutputFormat</i> format.
<i>from_siesta_aimd_output</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.siesta.SiestaAIMDOutputFormat</i> format.
<i>from_siesta_output</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.siesta.SiestaOutputFormat</i> format.
<i>from_sqm_in</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.amber.SQMInFormat</i> format.
<i>from_sqm_out</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.amber.SQMOutFormat</i> format.
<i>from_stru</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.abacus.AbacusSTRUFormat</i> format.
<i>from_vasp_contcar</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.vasp.VASPPoscarFormat</i> format.
<i>from_vasp_outcar</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.vasp.VASPOutcarFormat</i> format.
<i>from_vasp_poscar</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.vasp.VASPPoscarFormat</i> format.
<i>from_vasp_string</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.vasp.VASPStringFormat</i> format.

continues on next page

Table 6 – continued from previous page

<code>from_vasp_xml(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.vasp.VASPXMLFormat</code> format.
<code>from_xml(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.vasp.VASPXMLFormat</code> format.
<code>from_xyz(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.xyz.XYZFormat</code> format.
<code>get_atom_names()</code>	Returns name of atoms.
<code>get_atom_nums()</code>	Returns number of atoms.
<code>get_atom_types()</code>	Returns type of atoms.
<code>get_natoms()</code>	Returns total number of atoms in the system.
<code>get_nframes()</code>	Returns number of frames in the system.
<code>get_ntypes()</code>	Returns total number of atom types in the system.
<code>load(filename)</code>	Rebuild System obj.
<code>map_atom_types([type_map])</code>	Map the atom types of the system.
<code>minimize(*args, minimizer, **kwargs)</code>	Minimize the geometry.
<code>perturb(pert_num, cell_pert_fraction, ..., ...)</code>	Perturb each frame in the system randomly.
<code>pick_atom_idx(idx[, nopbc])</code>	Pick atom index.
<code>pick_by_amber_mask(param, maskstr[, ...])</code>	Pick atoms by amber mask.
<code>predict(*args[, driver])</code>	Predict energies and forces by a driver.
<code>register_data_type(*data_type)</code>	Register data type.
<code>remove_atom_names(atom_names)</code>	Remove atom names and all such atoms.
<code>remove_outlier([threshold])</code>	Remove outlier frames from the system.
<code>remove_pbc([protect_layer])</code>	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 <i>protect_layer</i> (2) translates the system such that the center-of-geometry of the system locates at the center of the cell.
<code>replicate(ncopy)</code>	Replicate the each frame in the system in 3 dimensions.
<code>shuffle()</code>	Shuffle frames randomly.
<code>sort_atom_names([type_map])</code>	Sort <code>atom_names</code> of the system and reorder <code>atom_nums</code> and <code>atom_types</code> according to <code>atom_names</code> .
<code>sort_atom_types()</code>	Sort atom types.
<code>sub_system(f_idx)</code>	Construct a subsystem from the system.
<code>to(fmt, *args, **kwargs)</code>	Dump systems to the specific format.
<code>to_3dmol(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.3dmol.Py3DMolFormat</code> format.
<code>to_abacus_lcao_md(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.abacus.AbacusMDFormat</code> format.
<code>to_abacus_lcao_relax(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.abacus.AbacusRelaxFormat</code> format.
<code>to_abacus_lcao_scf(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.abacus.AbacusSCFFormat</code> format.
<code>to_abacus_md(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.abacus.AbacusMDFormat</code> format.
<code>to_abacus_pw_md(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.abacus.AbacusMDFormat</code> format.
<code>to_abacus_pw_relax(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.abacus.AbacusRelaxFormat</code> format.

continues on next page

Table 6 – continued from previous page

<code>to_abacus_pw_scf(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.abacus.AbacusSCFFormat</code> format.
<code>to_abacus_relax(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.abacus.AbacusRelaxFormat</code> format.
<code>to_abacus_scf(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.abacus.AbacusSCFFormat</code> format.
<code>to_abacus_stru(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.abacus.AbacusSTRUFormat</code> format.
<code>to_amber_md(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.amber.AmberMDFormat</code> format.
<code>to_ase_structure(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.ase.ASEStructureFormat</code> format.
<code>to_ase_traj(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.ase.ASETrajFormat</code> format.
<code>to_atomconfig(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.pwmat.PwmatAtomconfigFormat</code> format.
<code>to_contcar(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.vasp.VASPPoscarFormat</code> format.
<code>to_cp2k_aimd_output(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.cp2k.CP2KAIMDOutputFormat</code> format.
<code>to_cp2k_output(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.cp2k.CP2KOutputFormat</code> format.
<code>to_deepmd(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.deepmd.DeePMDRawFormat</code> format.
<code>to_deepmd_comp(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.deepmd.DeePMDCompFormat</code> format.
<code>to_deepmd_hdf5(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.deepmd.DeePMDHDF5Format</code> format.
<code>to_deepmd_npy(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.deepmd.DeePMDCompFormat</code> format.
<code>to_deepmd_npy_mixed(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.deepmd.DeePMDMixedFormat</code> format.
<code>to_deepmd_raw(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.deepmd.DeePMDRawFormat</code> format.
<code>to_dftbplus(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.dftbplus.DFTBplusFormat</code> format.
<code>to_dump(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.lammps.LAMMPSDumpFormat</code> format.
<code>to_fhi_aims_md(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.fhi_aims.FhiMDFormat</code> format.
<code>to_fhi_aims_output(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.fhi_aims.FhiMDFormat</code> format.
<code>to_fhi_aims_scf(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.fhi_aims.FhiSCFFormat</code> format.
<code>to_finalconfig(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.pwmat.PwmatAtomconfigFormat</code> format.
<code>to_gaussian_gjf(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.gaussian.GaussianGJFFormat</code> format.
<code>to_gaussian_log(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.gaussian.GaussianLogFormat</code> format.
<code>to_gaussian_md(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.gaussian.GaussianMDFormat</code> format.

continues on next page

Table 6 – continued from previous page

<code>to_gro(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.gromacs.GromacsGroFormat</code> format.
<code>to_gromacs_gro(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.gromacs.GromacsGroFormat</code> format.
<code>to_json()</code>	Returns a json string representation of the MSONable object.
<code>to_lammps_dump(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.lammps.LAMMPSDumpFormat</code> format.
<code>to_lammps_lmp(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.lammps.LAMMPSLmpFormat</code> format.
<code>to_list(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.list.ListFormat</code> format.
<code>to_lmp(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.lammps.LAMMPSLmpFormat</code> format.
<code>to_mlmd(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.pwmat.PwmatOutputFormat</code> format.
<code>to_mol(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.rdkit.MolFormat</code> format.
<code>to_mol_file(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.rdkit.MolFormat</code> format.
<code>to_movement(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.pwmat.PwmatOutputFormat</code> format.
<code>to_n2p2(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.n2p2.N2P2Format</code> format.
<code>to_openmx_md(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.openmx.OPENMXFormat</code> format.
<code>to_orca_spout(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.orca.ORBASPOutFormat</code> format.
<code>to_outcar(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.vasp.VASPOutcarFormat</code> format.
<code>to_poscar(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.vasp.VASPPoscarFormat</code> format.
<code>to_psi4_inp(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.psi4.PSI4InputFormat</code> format.
<code>to_psi4_out(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.psi4.PSI4OutFormat</code> format.
<code>to_pwmat_atomconfig(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.pwmat.PwmatAtomconfigFormat</code> format.
<code>to_pwmat_finalconfig(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.pwmat.PwmatAtomconfigFormat</code> format.
<code>to_pwmat_mlmd(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.pwmat.PwmatOutputFormat</code> format.
<code>to_pwmat_movement(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.pwmat.PwmatOutputFormat</code> format.
<code>to_pwmat_output(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.pwmat.PwmatOutputFormat</code> format.
<code>to_pymatgen_ComputedStructureEntry(*args, ...)</code>	Dump data to <code>dpdata.plugins.pymatgen.PyMatgenCSEFormat</code> format.
<code>to_pymatgen_computedstructureentry(*args, ...)</code>	Dump data to <code>dpdata.plugins.pymatgen.PyMatgenCSEFormat</code> format.
<code>to_pymatgen_molecule(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.pymatgen.PyMatgenMoleculeFormat</code> format.

continues on next page

Table 6 – continued from previous page

<code>to_pymatgen_structure(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.pymatgen.PyMatgenStructureFormat</code> format.
<code>to_qe_cp_traj(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.qe.QECPTrajFormat</code> format.
<code>to_qe_pw_scf(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.qe.QECPWSCFFormat</code> format.
<code>to_quip_gap_xyz(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.xyz.QuipGapXYZFormat</code> format.
<code>to_quip_gap_xyz_file(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.xyz.QuipGapXYZFormat</code> format.
<code>to_sdf(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.rdkit.SdfFormat</code> format.
<code>to_sdf_file(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.rdkit.SdfFormat</code> format.
<code>to_siesta_aimd_output(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.siesta.SiestaAIMDOutputFormat</code> format.
<code>to_siesta_output(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.siesta.SiestaOutputFormat</code> format.
<code>to_sqm_in(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.amber.SQMInFormat</code> format.
<code>to_sqm_out(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.amber.SQMOutFormat</code> format.
<code>to_stru(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.abacus.AbacusSTRUFormat</code> format.
<code>to_vasp_contcar(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.vasp.VASPPoscarFormat</code> format.
<code>to_vasp_outcar(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.vasp.VASPOutcarFormat</code> format.
<code>to_vasp_poscar(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.vasp.VASPPoscarFormat</code> format.
<code>to_vasp_string(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.vasp.VASPStringFormat</code> format.
<code>to_vasp_xml(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.vasp.VASPXMLFormat</code> format.
<code>to_xml(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.vasp.VASPXMLFormat</code> format.
<code>to_xyz(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.xyz.XYZFormat</code> format.
<code>unsafe_hash()</code>	Returns an hash of the current object.
<code>validate_monty_v1(_MSONable__input_value)</code>	Pydantic validator with correct signature for pydantic v1.x
<code>validate_monty_v2(_MSONable__input_value, _)</code>	Pydantic validator with correct signature for pydantic v2.x



<code>affine_map</code>
<code>affine_map_fv</code>
<code>from_fmt</code>
<code>from_fmt_obj</code>
<code>has_virial</code>
<code>replace</code>
<code>rot_frame_lower_triangular</code>
<code>rot_lower_triangular</code>
<code>to_fmt_obj</code>

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

**`affine_map_fv`**(*trans*, *f\_idx*)

**`correction`**(*hl\_sys*)

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

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

#### Parameters

**`hl_sys`**  
[LabeledSystem] high-level LabeledSystem

#### Returns

**`corrected_sys`**: LabeledSystem  
Corrected LabeledSystem

**`from_3dmol`**(*file\_name*, *\*\*kwargs*)

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

**`from_abacus_lcao_md`**(*file\_name*, *\*\*kwargs*)

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

**`from_abacus_lcao_relax`**(*file\_name*, *\*\*kwargs*)

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

**`from_abacus_lcao_scf`**(*file\_name*, *\*\*kwargs*)

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

**`from_abacus_md`**(*file\_name*, *\*\*kwargs*)

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

**`from_abacus_pw_md`**(*file\_name*, *\*\*kwargs*)

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

**`from_abacus_pw_relax`**(*file\_name*, *\*\*kwargs*)

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

**from\_abacus\_pw\_scf**(*file\_name*, *\*\*kwargs*)  
Read data from `dpdata.plugins.abacus.AbacusSCFFormat` format.

**from\_abacus\_relax**(*file\_name*, *\*\*kwargs*)  
Read data from `dpdata.plugins.abacus.AbacusRelaxFormat` format.

**from\_abacus\_scf**(*file\_name*, *\*\*kwargs*)  
Read data from `dpdata.plugins.abacus.AbacusSCFFormat` format.

**from\_abacus\_stru**(*file\_name*, *\*\*kwargs*)  
Read data from `dpdata.plugins.abacus.AbacusSTRUFormat` format.

**from\_amber\_md**(*file\_name*, *\*\*kwargs*)  
Read data from `dpdata.plugins.amber.AmberMDFormat` format.

**from\_ase\_structure**(*file\_name*, *\*\*kwargs*)  
Read data from `dpdata.plugins.ase.ASEStructureFormat` format.

**from\_ase\_traj**(*file\_name*, *\*\*kwargs*)  
Read data from `dpdata.plugins.ase.ASETrajFormat` format.

**from\_atomconfig**(*file\_name*, *\*\*kwargs*)  
Read data from `dpdata.plugins.pwmat.PwmatAtomconfigFormat` format.

**from\_contcar**(*file\_name*, *\*\*kwargs*)  
Read data from `dpdata.plugins.vasp.VASPPoscarFormat` format.

**from\_cp2k\_aimd\_output**(*file\_name*, *\*\*kwargs*)  
Read data from `dpdata.plugins.cp2k.CP2KAIMDOutputFormat` format.

**from\_cp2k\_output**(*file\_name*, *\*\*kwargs*)  
Read data from `dpdata.plugins.cp2k.CP2KOutputFormat` format.

**from\_deepmd**(*file\_name*, *\*\*kwargs*)  
Read data from `dpdata.plugins.deepmd.DeepMDRawFormat` format.

**from\_deepmd\_comp**(*file\_name*, *\*\*kwargs*)  
Read data from `dpdata.plugins.deepmd.DeepMDCompFormat` format.

**from\_deepmd\_hdf5**(*file\_name*, *\*\*kwargs*)  
Read data from `dpdata.plugins.deepmd.DeepMDHDF5Format` format.

**from\_deepmd\_npy**(*file\_name*, *\*\*kwargs*)  
Read data from `dpdata.plugins.deepmd.DeepMDCompFormat` format.

**from\_deepmd\_npy\_mixed**(*file\_name*, *\*\*kwargs*)  
Read data from `dpdata.plugins.deepmd.DeepMDMixedFormat` format.

**from\_deepmd\_raw**(*file\_name*, *\*\*kwargs*)  
Read data from `dpdata.plugins.deepmd.DeepMDRawFormat` format.

**from\_dftbplus**(*file\_name*, *\*\*kwargs*)  
Read data from `dpdata.plugins.dftbplus.DFTBplusFormat` format.

**from\_dump**(*file\_name*, *\*\*kwargs*)  
Read data from `dpdata.plugins.lammps.LAMMPSDumpFormat` format.



**from\_fhi\_aims\_md**(file\_name, \*\*kwargs)  
Read data from `dpdata.plugins.fhi_aims.FhiMDFormat` format.

**from\_fhi\_aims\_output**(file\_name, \*\*kwargs)  
Read data from `dpdata.plugins.fhi_aims.FhiMDFormat` format.

**from\_fhi\_aims\_scf**(file\_name, \*\*kwargs)  
Read data from `dpdata.plugins.fhi_aims.FhiSCFFormat` format.

**from\_finalconfig**(file\_name, \*\*kwargs)  
Read data from `dpdata.plugins.pwmat.PwmatAtomconfigFormat` format.

**from\_fmt\_obj**(fmtobj, file\_name, \*\*kwargs)

**from\_gaussian\_gjf**(file\_name, \*\*kwargs)  
Read data from `dpdata.plugins.gaussian.GaussianGJFFormat` format.

**from\_gaussian\_log**(file\_name, \*\*kwargs)  
Read data from `dpdata.plugins.gaussian.GaussianLogFormat` format.

**from\_gaussian\_md**(file\_name, \*\*kwargs)  
Read data from `dpdata.plugins.gaussian.GaussianMDFormat` format.

**from\_gro**(file\_name, \*\*kwargs)  
Read data from `dpdata.plugins.gromacs.GromacsGroFormat` format.

**from\_gromacs\_gro**(file\_name, \*\*kwargs)  
Read data from `dpdata.plugins.gromacs.GromacsGroFormat` format.

**from\_lammps\_dump**(file\_name, \*\*kwargs)  
Read data from `dpdata.plugins.lammps.LAMMPSDumpFormat` format.

**from\_lammps\_lmp**(file\_name, \*\*kwargs)  
Read data from `dpdata.plugins.lammps.LAMMPSLmpFormat` format.

**from\_list**(file\_name, \*\*kwargs)  
Read data from `dpdata.plugins.list.ListFormat` format.

**from\_lmp**(file\_name, \*\*kwargs)  
Read data from `dpdata.plugins.lammps.LAMMPSLmpFormat` format.

**from\_mlmd**(file\_name, \*\*kwargs)  
Read data from `dpdata.plugins.pwmat.PwmatOutputFormat` format.

**from\_mol**(file\_name, \*\*kwargs)  
Read data from `dpdata.plugins.rdkit.MolFormat` format.

**from\_mol\_file**(file\_name, \*\*kwargs)  
Read data from `dpdata.plugins.rdkit.MolFormat` format.

**from\_movement**(file\_name, \*\*kwargs)  
Read data from `dpdata.plugins.pwmat.PwmatOutputFormat` format.

**from\_n2p2**(file\_name, \*\*kwargs)  
Read data from `dpdata.plugins.n2p2.N2P2Format` format.

**from\_openmx\_md**(file\_name, \*\*kwargs)  
Read data from `dpdata.plugins.openmx.OPENMXFormat` format.

**from\_orca\_spout**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.orca.ORMCASPOutFormat` format.

**from\_outcar**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.vasp.VASPOutcarFormat` format.

**from\_poscar**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.vasp.VASPPoscarFormat` format.

**from\_psi4\_inp**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.psi4.PSI4InputFormat` format.

**from\_psi4\_out**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.psi4.PSI4OutFormat` format.

**from\_pwmat\_atomconfig**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.pwmat.PwmatAtomconfigFormat` format.

**from\_pwmat\_finalconfig**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.pwmat.PwmatAtomconfigFormat` format.

**from\_pwmat\_mlmd**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.pwmat.PwmatOutputFormat` format.

**from\_pwmat\_movement**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.pwmat.PwmatOutputFormat` format.

**from\_pwmat\_output**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.pwmat.PwmatOutputFormat` format.

**from\_pymatgen\_computedstructureentry**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.pymatgen.PyMatgenCSEFormat` format.

**from\_pymatgen\_molecule**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.pymatgen.PyMatgenMoleculeFormat` format.

**from\_pymatgen\_structure**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.pymatgen.PyMatgenStructureFormat` format.

**from\_qe\_cp\_traj**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.qe.QECPTrajFormat` format.

**from\_qe\_pw\_scf**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.qe.QECPWSCFFormat` format.

**from\_quip\_gap\_xyz**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.xyz.QuipGapXYZFormat` format.

**from\_quip\_gap\_xyz\_file**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.xyz.QuipGapXYZFormat` format.

**from\_sdf**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.rdkit.SdfFormat` format.

**from\_sdf\_file**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.rdkit.SdfFormat` format.

**from\_siesta\_aiMD\_output**(*file\_name*, **\*\*kwargs**)  
 Read data from `dpdata.plugins.siesta.SiestaAIMDOutputFormat` format.

**from\_siesta\_aimd\_output**(*file\_name*, **\*\*kwargs**)  
 Read data from `dpdata.plugins.siesta.SiestaAIMDOutputFormat` format.

**from\_siesta\_output**(*file\_name*, **\*\*kwargs**)  
 Read data from `dpdata.plugins.siesta.SiestaOutputFormat` format.

**from\_sqm\_in**(*file\_name*, **\*\*kwargs**)  
 Read data from `dpdata.plugins.amber.SQMInFormat` format.

**from\_sqm\_out**(*file\_name*, **\*\*kwargs**)  
 Read data from `dpdata.plugins.amber.SQMOutFormat` format.

**from\_stru**(*file\_name*, **\*\*kwargs**)  
 Read data from `dpdata.plugins.abacus.AbacusSTRUFormat` format.

**from\_vasp\_contcar**(*file\_name*, **\*\*kwargs**)  
 Read data from `dpdata.plugins.vasp.VASPPoscarFormat` format.

**from\_vasp\_outcar**(*file\_name*, **\*\*kwargs**)  
 Read data from `dpdata.plugins.vasp.VASPOutcarFormat` format.

**from\_vasp\_poscar**(*file\_name*, **\*\*kwargs**)  
 Read data from `dpdata.plugins.vasp.VASPPoscarFormat` format.

**from\_vasp\_string**(*file\_name*, **\*\*kwargs**)  
 Read data from `dpdata.plugins.vasp.VASPStringFormat` format.

**from\_vasp\_xml**(*file\_name*, **\*\*kwargs**)  
 Read data from `dpdata.plugins.vasp.VASPXMLFormat` format.

**from\_xml**(*file\_name*, **\*\*kwargs**)  
 Read data from `dpdata.plugins.vasp.VASPXMLFormat` format.

**from\_xyz**(*file\_name*, **\*\*kwargs**)  
 Read data from `dpdata.plugins.xyz.XYZFormat` format.

**has\_virial**()

**post\_funcs** = <dpdata.plugin.Plugin object>

**remove\_outlier**(*threshold*: *float* = 8.0) → *LabeledSystem*  
 Remove outlier frames from the system.  
 Remove the frames whose energies satisfy the condition

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

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

**Parameters**

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

**Returns**

**LabeledSystem**  
 The system without outlier frames.

## References

[1], [2]

**rot\_frame\_lower\_triangular**(*f\_idx=0*)

**to\_3dmol**(\*args, \*\*kwargs)

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

**to\_abacus\_lcao\_md**(\*args, \*\*kwargs)

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

**to\_abacus\_lcao\_relax**(\*args, \*\*kwargs)

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

**to\_abacus\_lcao\_scf**(\*args, \*\*kwargs)

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

**to\_abacus\_md**(\*args, \*\*kwargs)

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

**to\_abacus\_pw\_md**(\*args, \*\*kwargs)

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

**to\_abacus\_pw\_relax**(\*args, \*\*kwargs)

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

**to\_abacus\_pw\_scf**(\*args, \*\*kwargs)

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

**to\_abacus\_relax**(\*args, \*\*kwargs)

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

**to\_abacus\_scf**(\*args, \*\*kwargs)

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

**to\_abacus\_stru**(\*args, \*\*kwargs)

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

**to\_amber\_md**(\*args, \*\*kwargs)

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

**to\_ase\_structure**(\*args, \*\*kwargs)

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

**to\_ase\_traj**(\*args, \*\*kwargs)

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

**to\_atomconfig**(\*args, \*\*kwargs)

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

**to\_contcar**(\*args, \*\*kwargs)

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

**to\_cp2k\_aimd\_output**(\*args, \*\*kwargs)

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

**to\_cp2k\_output**(\*args, \*\*kwargs)

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

**to\_deepmd**(\*args, \*\*kwargs)  
 Dump data to `dpdata.plugins.deepmd.DeePMDRawFormat` format.

**to\_deepmd\_comp**(\*args, \*\*kwargs)  
 Dump data to `dpdata.plugins.deepmd.DeePMDCompFormat` format.

**to\_deepmd\_hdf5**(\*args, \*\*kwargs)  
 Dump data to `dpdata.plugins.deepmd.DeePMDHDF5Format` format.

**to\_deepmd\_npy**(\*args, \*\*kwargs)  
 Dump data to `dpdata.plugins.deepmd.DeePMDCompFormat` format.

**to\_deepmd\_npy\_mixed**(\*args, \*\*kwargs)  
 Dump data to `dpdata.plugins.deepmd.DeePMDMixedFormat` format.

**to\_deepmd\_raw**(\*args, \*\*kwargs)  
 Dump data to `dpdata.plugins.deepmd.DeePMDRawFormat` format.

**to\_dftbplus**(\*args, \*\*kwargs)  
 Dump data to `dpdata.plugins.dftbplus.DFTBplusFormat` format.

**to\_dump**(\*args, \*\*kwargs)  
 Dump data to `dpdata.plugins.lammps.LAMMPSDumpFormat` format.

**to\_fhi\_aims\_md**(\*args, \*\*kwargs)  
 Dump data to `dpdata.plugins.fhi_aims.FhiMDFormat` format.

**to\_fhi\_aims\_output**(\*args, \*\*kwargs)  
 Dump data to `dpdata.plugins.fhi_aims.FhiMDFormat` format.

**to\_fhi\_aims\_scf**(\*args, \*\*kwargs)  
 Dump data to `dpdata.plugins.fhi_aims.FhiSCFFormat` format.

**to\_finalconfig**(\*args, \*\*kwargs)  
 Dump data to `dpdata.plugins.pwmat.PwmatAtomconfigFormat` format.

**to\_fmt\_obj**(fmtobj, \*args, \*\*kwargs)  
 Dump data to `dpdata.plugins.fmt_obj.FmtObjFormat` format.

**to\_gaussian\_gjf**(\*args, \*\*kwargs)  
 Dump data to `dpdata.plugins.gaussian.GaussianGJFFormat` format.

**to\_gaussian\_log**(\*args, \*\*kwargs)  
 Dump data to `dpdata.plugins.gaussian.GaussianLogFormat` format.

**to\_gaussian\_md**(\*args, \*\*kwargs)  
 Dump data to `dpdata.plugins.gaussian.GaussianMDFormat` format.

**to\_gro**(\*args, \*\*kwargs)  
 Dump data to `dpdata.plugins.gromacs.GromacsGroFormat` format.

**to\_gromacs\_gro**(\*args, \*\*kwargs)  
 Dump data to `dpdata.plugins.gromacs.GromacsGroFormat` format.

**to\_lammps\_dump**(\*args, \*\*kwargs)  
 Dump data to `dpdata.plugins.lammps.LAMMPSDumpFormat` format.

**to\_lammps\_lmp**(\*args, \*\*kwargs)  
 Dump data to `dpdata.plugins.lammps.LAMMPSLmpFormat` format.

**to\_list**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.list.ListFormat` format.

**to\_lmp**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.lammps.LAMMPSLmpFormat` format.

**to\_mlmd**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.pwmat.PwmatOutputFormat` format.

**to\_mol**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.rdkit.MolFormat` format.

**to\_mol\_file**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.rdkit.MolFormat` format.

**to\_movement**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.pwmat.PwmatOutputFormat` format.

**to\_n2p2**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.n2p2.N2P2Format` format.

**to\_openmx\_md**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.openmx.OPENMXFormat` format.

**to\_orca\_spout**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.orca.OCASPOutFormat` format.

**to\_outcar**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.vasp.VASPOutcarFormat` format.

**to\_poscar**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.vasp.VASPPoscarFormat` format.

**to\_psi4\_inp**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.psi4.PSI4InputFormat` format.

**to\_psi4\_out**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.psi4.PSI4OutFormat` format.

**to\_pwmat\_atomconfig**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.pwmat.PwmatAtomconfigFormat` format.

**to\_pwmat\_finalconfig**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.pwmat.PwmatAtomconfigFormat` format.

**to\_pwmat\_mlmd**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.pwmat.PwmatOutputFormat` format.

**to\_pwmat\_movement**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.pwmat.PwmatOutputFormat` format.

**to\_pwmat\_output**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.pwmat.PwmatOutputFormat` format.

**to\_pymatgen\_ComputedStructureEntry**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.pymatgen.PyMatgenCSEFormat` format.

**to\_pymatgen\_computedstructureentry**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.pymatgen.PyMatgenCSEFormat` format.

**to\_pymatgen\_molecule**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.pymatgen.PyMatgenMoleculeFormat` format.

**to\_pymatgen\_structure**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.pymatgen.PyMatgenStructureFormat` format.

**to\_qe\_cp\_traj**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.qe.QECPTrajFormat` format.

**to\_qe\_pw\_scf**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.qe.QECPWSCFFormat` format.

**to\_quip\_gap\_xyz**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.xyz.QuipGapXYZFormat` format.

**to\_quip\_gap\_xyz\_file**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.xyz.QuipGapXYZFormat` format.

**to\_sdf**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.rdkit.SdfFormat` format.

**to\_sdf\_file**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.rdkit.SdfFormat` format.

**to\_siesta\_aimd\_output**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.siesta.SiestaAIMDOutputFormat` format.

**to\_siesta\_output**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.siesta.SiestaOutputFormat` format.

**to\_sqm\_in**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.amber.SQMInFormat` format.

**to\_sqm\_out**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.amber.SQMOutFormat` format.

**to\_stru**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.abacus.AbacusSTRUFormat` format.

**to\_vasp\_contcar**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.vasp.VASPPoscarFormat` format.

**to\_vasp\_outcar**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.vasp.VASPOutcarFormat` format.

**to\_vasp\_poscar**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.vasp.VASPPoscarFormat` format.

**to\_vasp\_string**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.vasp.VASPStringFormat` format.

**to\_vasp\_xml**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.vasp.VASPXMLFormat` format.



**to\_xml**(\*args, \*\*kwargs)

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

**to\_xyz**(\*args, \*\*kwargs)

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

**class** `dpdata.system.MultiSystems`(\*systems, type\_map=None)

Bases: `object`

A set containing several systems.

## Methods

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

continues on next page



Table 7 – continued from previous page

<i>from_cp2k_output</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.cp2k.CP2KOutputFormat</i> format.
<i>from_deepmd</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.deepmd.DeePMDRawFormat</i> format.
<i>from_deepmd_comp</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.deepmd.DeePMDCompFormat</i> format.
<i>from_deepmd_hdf5</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.deepmd.DeePMDHDF5Format</i> format.
<i>from_deepmd_npy</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.deepmd.DeePMDCompFormat</i> format.
<i>from_deepmd_npy_mixed</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.deepmd.DeePMDMixedFormat</i> format.
<i>from_deepmd_raw</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.deepmd.DeePMDRawFormat</i> format.
<i>from_dftbplus</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.dftbplus.DFTBplusFormat</i> format.
<i>from_dump</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.lammps.LAMMPSDumpFormat</i> format.
<i>from_fhi_aims_md</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.fhi_aims.FhiMDFormat</i> format.
<i>from_fhi_aims_output</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.fhi_aims.FhiMDFormat</i> format.
<i>from_fhi_aims_scf</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.fhi_aims.FhiSCFFormat</i> format.
<i>from_finalconfig</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.pwmat.PwmatAtomconfigFormat</i> format.
<i>from_gaussian_gjf</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.gaussian.GaussianGJFFormat</i> format.
<i>from_gaussian_log</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.gaussian.GaussianLogFormat</i> format.
<i>from_gaussian_md</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.gaussian.GaussianMDFormat</i> format.
<i>from_gro</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.gromacs.GromacsGroFormat</i> format.
<i>from_gromacs_gro</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.gromacs.GromacsGroFormat</i> format.
<i>from_lammps_dump</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.lammps.LAMMPSDumpFormat</i> format.
<i>from_lammps_lmp</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.lammps.LAMMPSLmpFormat</i> format.
<i>from_list</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.list.ListFormat</i> format.
<i>from_lmp</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.lammps.LAMMPSLmpFormat</i> format.
<i>from_mlmd</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.pwmat.PwmatOutputFormat</i> format.
<i>from_mol</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.rdkit.MolFormat</i> format.
<i>from_mol_file</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.rdkit.MolFormat</i> format.
<i>from_movement</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.pwmat.PwmatOutputFormat</i> format.

continues on next page

Table 7 – continued from previous page

<i>from_n2p2</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.n2p2.N2P2Format</i> format.
<i>from_openmx_md</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.openmx.OPENMXFormat</i> format.
<i>from_orca_spout</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.orca.ORBASPOutFormat</i> format.
<i>from_outcar</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.vasp.VASPOutcarFormat</i> format.
<i>from_poscar</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.vasp.VASPPoscarFormat</i> format.
<i>from_psi4_inp</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.psi4.PSI4InputFormat</i> format.
<i>from_psi4_out</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.psi4.PSI4OutFormat</i> format.
<i>from_pwmat_atomconfig</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.pwmat.PwmatAtomconfigFormat</i> format.
<i>from_pwmat_finalconfig</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.pwmat.PwmatAtomconfigFormat</i> format.
<i>from_pwmat_mlmd</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.pwmat.PwmatOutputFormat</i> format.
<i>from_pwmat_movement</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.pwmat.PwmatOutputFormat</i> format.
<i>from_pwmat_output</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.pwmat.PwmatOutputFormat</i> format.
<i>from_pymatgen_computedstructureentry</i> (...)	Read data from <i>dpdata.plugins.pymatgen.PyMatgenCSEFormat</i> format.
<i>from_pymatgen_molecule</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.pymatgen.PyMatgenMoleculeFormat</i> format.
<i>from_pymatgen_structure</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.pymatgen.PyMatgenStructureFormat</i> format.
<i>from_qe_cp_traj</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.qe.QECPTrajFormat</i> format.
<i>from_qe_pw_scf</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.qe.QECPWSCFFormat</i> format.
<i>from_quip_gap_xyz</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.xyz.QuipGapXYZFormat</i> format.
<i>from_quip_gap_xyz_file</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.xyz.QuipGapXYZFormat</i> format.
<i>from_sdf</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.rdkit.SdfFormat</i> format.
<i>from_sdf_file</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.rdkit.SdfFormat</i> format.
<i>from_siesta_aiMD_output</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.siesta.SiestaAIMDOutputFormat</i> format.
<i>from_siesta_aimd_output</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.siesta.SiestaAIMDOutputFormat</i> format.
<i>from_siesta_output</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.siesta.SiestaOutputFormat</i> format.
<i>from_sqm_in</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.amber.SQMInFormat</i> format.
<i>from_sqm_out</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.amber.SQMOutFormat</i> format.

continues on next page

Table 7 – continued from previous page

<i>from_stru</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.abacus.AbacusSTRUFormat</i> format.
<i>from_vasp_contcar</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.vasp.VASPPoscarFormat</i> format.
<i>from_vasp_outcar</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.vasp.VASPOutcarFormat</i> format.
<i>from_vasp_poscar</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.vasp.VASPPoscarFormat</i> format.
<i>from_vasp_string</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.vasp.VASPStringFormat</i> format.
<i>from_vasp_xml</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.vasp.VASPXMLFormat</i> format.
<i>from_xml</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.vasp.VASPXMLFormat</i> format.
<i>from_xyz</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.xyz.XYZFormat</i> format.
<i>get_nframes</i> ()	Returns number of frames in all systems.
<i>minimize</i> (*args, minimizer, **kwargs)	Minimize geometry by a minimizer.
<i>pick_atom_idx</i> (idx[, nopbc])	Pick atom index.
<i>predict</i> (*args[, driver])	Predict energies and forces by a driver.
<i>to</i> (fmt, *args, **kwargs)	Dump systems to the specific format.
<i>to_3dmol</i> (*args, **kwargs)	Dump data to <i>dpdata.plugins.3dmol.Py3DMolFormat</i> format.
<i>to_abacus_lcao_md</i> (*args, **kwargs)	Dump data to <i>dpdata.plugins.abacus.AbacusMDFormat</i> format.
<i>to_abacus_lcao_relax</i> (*args, **kwargs)	Dump data to <i>dpdata.plugins.abacus.AbacusRelaxFormat</i> format.
<i>to_abacus_lcao_scf</i> (*args, **kwargs)	Dump data to <i>dpdata.plugins.abacus.AbacusSCFFormat</i> format.
<i>to_abacus_md</i> (*args, **kwargs)	Dump data to <i>dpdata.plugins.abacus.AbacusMDFormat</i> format.
<i>to_abacus_pw_md</i> (*args, **kwargs)	Dump data to <i>dpdata.plugins.abacus.AbacusMDFormat</i> format.
<i>to_abacus_pw_relax</i> (*args, **kwargs)	Dump data to <i>dpdata.plugins.abacus.AbacusRelaxFormat</i> format.
<i>to_abacus_pw_scf</i> (*args, **kwargs)	Dump data to <i>dpdata.plugins.abacus.AbacusSCFFormat</i> format.
<i>to_abacus_relax</i> (*args, **kwargs)	Dump data to <i>dpdata.plugins.abacus.AbacusRelaxFormat</i> format.
<i>to_abacus_scf</i> (*args, **kwargs)	Dump data to <i>dpdata.plugins.abacus.AbacusSCFFormat</i> format.
<i>to_abacus_stru</i> (*args, **kwargs)	Dump data to <i>dpdata.plugins.abacus.AbacusSTRUFormat</i> format.
<i>to_amber_md</i> (*args, **kwargs)	Dump data to <i>dpdata.plugins.amber.AmberMDFormat</i> format.
<i>to_ase_structure</i> (*args, **kwargs)	Dump data to <i>dpdata.plugins.ase.ASEStructureFormat</i> format.
<i>to_ase_traj</i> (*args, **kwargs)	Dump data to <i>dpdata.plugins.ase.ASETrajFormat</i> format.
<i>to_atomconfig</i> (*args, **kwargs)	Dump data to <i>dpdata.plugins.pwmat.PwmatAtomconfigFormat</i> format.

continues on next page

Table 7 – continued from previous page

<code>to_contcar(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.vasp.VASPPoscarFormat</code> format.
<code>to_cp2k_aimd_output(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.cp2k.CP2KAIMDOutputFormat</code> format.
<code>to_cp2k_output(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.cp2k.CP2KOutputFormat</code> format.
<code>to_deepmd(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.deepmd.DeePMDRawFormat</code> format.
<code>to_deepmd_comp(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.deepmd.DeePMDCompFormat</code> format.
<code>to_deepmd_hdf5(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.deepmd.DeePMDHDF5Format</code> format.
<code>to_deepmd_npy(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.deepmd.DeePMDCompFormat</code> format.
<code>to_deepmd_npy_mixed(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.deepmd.DeePMDMixedFormat</code> format.
<code>to_deepmd_raw(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.deepmd.DeePMDRawFormat</code> format.
<code>to_dftbplus(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.dftbplus.DFTBplusFormat</code> format.
<code>to_dump(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.lammps.LAMMPSDumpFormat</code> format.
<code>to_fhi_aims_md(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.fhi_aims.FhiMDFormat</code> format.
<code>to_fhi_aims_output(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.fhi_aims.FhiMDFormat</code> format.
<code>to_fhi_aims_scf(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.fhi_aims.FhiSCFFormat</code> format.
<code>to_finalconfig(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.pwmat.PwmatAtomconfigFormat</code> format.
<code>to_gaussian_gjf(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.gaussian.GaussianGJFFormat</code> format.
<code>to_gaussian_log(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.gaussian.GaussianLogFormat</code> format.
<code>to_gaussian_md(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.gaussian.GaussianMDFormat</code> format.
<code>to_gro(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.gromacs.GromacsGroFormat</code> format.
<code>to_gromacs_gro(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.gromacs.GromacsGroFormat</code> format.
<code>to_lammps_dump(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.lammps.LAMMPSDumpFormat</code> format.
<code>to_lammps_lmp(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.lammps.LAMMPSLmpFormat</code> format.
<code>to_list(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.list.ListFormat</code> format.
<code>to_lmp(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.lammps.LAMMPSLmpFormat</code> format.
<code>to_mlmd(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.pwmat.PwmatOutputFormat</code> format.
<code>to_mol(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.rdkit.MolFormat</code> format.

continues on next page

Table 7 – continued from previous page

<code>to_mol_file(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.rdkit.MolFormat</code> format.
<code>to_movement(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.pwmat.PwmatOutputFormat</code> format.
<code>to_n2p2(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.n2p2.N2P2Format</code> format.
<code>to_openmx_md(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.openmx.OPENMXFormat</code> format.
<code>to_orca_spout(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.orca.ORBASPOutFormat</code> format.
<code>to_outcar(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.vasp.VASPOutcarFormat</code> format.
<code>to_poscar(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.vasp.VASPPoscarFormat</code> format.
<code>to_psi4_inp(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.psi4.PSI4InputFormat</code> format.
<code>to_psi4_out(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.psi4.PSI4OutFormat</code> format.
<code>to_pwmat_atomconfig(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.pwmat.PwmatAtomconfigFormat</code> format.
<code>to_pwmat_finalconfig(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.pwmat.PwmatAtomconfigFormat</code> format.
<code>to_pwmat_mlmd(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.pwmat.PwmatOutputFormat</code> format.
<code>to_pwmat_movement(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.pwmat.PwmatOutputFormat</code> format.
<code>to_pwmat_output(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.pwmat.PwmatOutputFormat</code> format.
<code>to_pymatgen_ComputedStructureEntry(*args, ...)</code>	Dump data to <code>dpdata.plugins.pymatgen.PyMatgenCSEFormat</code> format.
<code>to_pymatgen_computedstructureentry(*args, ...)</code>	Dump data to <code>dpdata.plugins.pymatgen.PyMatgenCSEFormat</code> format.
<code>to_pymatgen_molecule(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.pymatgen.PyMatgenMoleculeFormat</code> format.
<code>to_pymatgen_structure(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.pymatgen.PyMatgenStructureFormat</code> format.
<code>to_qe_cp_traj(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.qe.QECPTrajFormat</code> format.
<code>to_qe_pw_scf(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.qe.QECPWSCFFFormat</code> format.
<code>to_quip_gap_xyz(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.xyz.QuipGapXYZFormat</code> format.
<code>to_quip_gap_xyz_file(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.xyz.QuipGapXYZFormat</code> format.
<code>to_sdf(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.rdkit.SdfFormat</code> format.
<code>to_sdf_file(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.rdkit.SdfFormat</code> format.
<code>to_siesta_aimd_output(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.siesta.SiestaAIMDOutputFormat</code> format.
<code>to_siesta_output(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.siesta.SiestaOutputFormat</code> format.

continues on next page

Table 7 – continued from previous page

<code>to_sqm_in(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.amber.SQMInFormat</code> format.
<code>to_sqm_out(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.amber.SQMOutFormat</code> format.
<code>to_stru(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.abacus.AbacusSTRUFormat</code> format.
<code>to_vasp_contcar(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.vasp.VASPPoscarFormat</code> format.
<code>to_vasp_outcar(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.vasp.VASPOutcarFormat</code> format.
<code>to_vasp_poscar(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.vasp.VASPPoscarFormat</code> format.
<code>to_vasp_string(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.vasp.VASPStringFormat</code> format.
<code>to_vasp_xml(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.vasp.VASPXMLFormat</code> format.
<code>to_xml(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.vasp.VASPXMLFormat</code> format.
<code>to_xyz(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.xyz.XYZFormat</code> format.
<code>train_test_split(test_size[, seed])</code>	Split systems into random train and test subsets.

<code>from_dir</code>
<code>from_file</code>
<code>from_fmt_obj</code>
<code>load_systems_from_file</code>
<code>to_fmt_obj</code>

**append(\*systems)**

Append systems or MultiSystems to systems.

**Parameters**

**\*systems**

[System] The system to append

**check\_atom\_names(system)**

Make atom\_names in all systems equal, prevent inconsistent atom\_types.

**correction(hl\_sys: MultiSystems)**

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

**Parameters**

**hl\_sys**

[MultiSystems] high-level MultiSystems

**Returns**

**corrected\_sys**

[MultiSystems] Corrected MultiSystems



## Notes

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

## Examples

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

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

**from\_3dmol**(*file\_name*, *\*\*kwargs*)

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

**from\_abacus\_lcao\_md**(*file\_name*, *\*\*kwargs*)

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

**from\_abacus\_lcao\_relax**(*file\_name*, *\*\*kwargs*)

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

**from\_abacus\_lcao\_scf**(*file\_name*, *\*\*kwargs*)

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

**from\_abacus\_md**(*file\_name*, *\*\*kwargs*)

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

**from\_abacus\_pw\_md**(*file\_name*, *\*\*kwargs*)

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

**from\_abacus\_pw\_relax**(*file\_name*, *\*\*kwargs*)

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

**from\_abacus\_pw\_scf**(*file\_name*, *\*\*kwargs*)

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

**from\_abacus\_relax**(*file\_name*, *\*\*kwargs*)

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

**from\_abacus\_scf**(*file\_name*, *\*\*kwargs*)

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

**from\_abacus\_stru**(*file\_name*, *\*\*kwargs*)

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

**from\_amber\_md**(*file\_name*, *\*\*kwargs*)

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

**from\_ase\_structure**(*file\_name*, *\*\*kwargs*)

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

**from\_ase\_traj**(*file\_name*, *\*\*kwargs*)

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

**from\_atomconfig**(*file\_name*, *\*\*kwargs*)  
Read data from `dpdata.plugins.pwmat.PwmatAtomconfigFormat` format.

**from\_contcar**(*file\_name*, *\*\*kwargs*)  
Read data from `dpdata.plugins.vasp.VASPPoscarFormat` format.

**from\_cp2k\_aimd\_output**(*file\_name*, *\*\*kwargs*)  
Read data from `dpdata.plugins.cp2k.CP2KAIMDOutputFormat` format.

**from\_cp2k\_output**(*file\_name*, *\*\*kwargs*)  
Read data from `dpdata.plugins.cp2k.CP2KOutputFormat` format.

**from\_deepmd**(*file\_name*, *\*\*kwargs*)  
Read data from `dpdata.plugins.deepmd.DeepMDRawFormat` format.

**from\_deepmd\_comp**(*file\_name*, *\*\*kwargs*)  
Read data from `dpdata.plugins.deepmd.DeepMDCompFormat` format.

**from\_deepmd\_hdf5**(*file\_name*, *\*\*kwargs*)  
Read data from `dpdata.plugins.deepmd.DeepMDHDF5Format` format.

**from\_deepmd\_npy**(*file\_name*, *\*\*kwargs*)  
Read data from `dpdata.plugins.deepmd.DeepMDCompFormat` format.

**from\_deepmd\_npy\_mixed**(*file\_name*, *\*\*kwargs*)  
Read data from `dpdata.plugins.deepmd.DeepMDMixedFormat` format.

**from\_deepmd\_raw**(*file\_name*, *\*\*kwargs*)  
Read data from `dpdata.plugins.deepmd.DeepMDRawFormat` format.

**from\_dftbplus**(*file\_name*, *\*\*kwargs*)  
Read data from `dpdata.plugins.dftbplus.DFTBplusFormat` format.

**classmethod from\_dir**(*dir\_name*, *file\_name*, *fmt='auto'*, *type\_map=None*)

**from\_dump**(*file\_name*, *\*\*kwargs*)  
Read data from `dpdata.plugins.lammps.LAMMPSDumpFormat` format.

**from\_fhi\_aims\_md**(*file\_name*, *\*\*kwargs*)  
Read data from `dpdata.plugins.fhi_aims.FhiMDFormat` format.

**from\_fhi\_aims\_output**(*file\_name*, *\*\*kwargs*)  
Read data from `dpdata.plugins.fhi_aims.FhiMDFormat` format.

**from\_fhi\_aims\_scf**(*file\_name*, *\*\*kwargs*)  
Read data from `dpdata.plugins.fhi_aims.FhiSCFFormat` format.

**classmethod from\_file**(*file\_name*, *fmt*, *\*\*kwargs*)

**from\_finalconfig**(*file\_name*, *\*\*kwargs*)  
Read data from `dpdata.plugins.pwmat.PwmatAtomconfigFormat` format.

**from\_fmt\_obj**(*fmtobj*, *directory*, *labeled=True*, *\*\*kwargs*)

**from\_gaussian\_gjf**(*file\_name*, *\*\*kwargs*)  
Read data from `dpdata.plugins.gaussian.GaussianGJFFormat` 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.LAMPSDumpFormat` format.

**from\_lammps\_lmp**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.lammps.LAMPSLmpFormat` 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.LAMPSLmpFormat` format.

**from\_mlmd**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.pwmat.PwmatOutputFormat` format.

**from\_mol**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.rdkit.MolFormat` format.

**from\_mol\_file**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.rdkit.MolFormat` format.

**from\_movement**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.pwmat.PwmatOutputFormat` format.

**from\_n2p2**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.n2p2.N2P2Format` format.

**from\_openmx\_md**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.openmx.OPENMXFormat` format.

**from\_orca\_spout**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.orca.ORBASPOutFormat` format.

**from\_outcar**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.vasp.VASPOutcarFormat` format.

**from\_poscar**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.vasp.VASPPoscarFormat` format.

**from\_psi4\_inp**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.psi4.PSI4InputFormat` format.

**from\_psi4\_out**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.psi4.PSI4OutFormat` format.

**from\_pwmat\_atomconfig**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.pwmat.PwmatAtomconfigFormat` format.

**from\_pwmat\_finalconfig**(*file\_name*, *\*\*kwargs*)  
Read data from `dpdata.plugins.pwmat.PwmatAtomconfigFormat` format.

**from\_pwmat\_mlmd**(*file\_name*, *\*\*kwargs*)  
Read data from `dpdata.plugins.pwmat.PwmatOutputFormat` format.

**from\_pwmat\_movement**(*file\_name*, *\*\*kwargs*)  
Read data from `dpdata.plugins.pwmat.PwmatOutputFormat` format.

**from\_pwmat\_output**(*file\_name*, *\*\*kwargs*)  
Read data from `dpdata.plugins.pwmat.PwmatOutputFormat` format.

**from\_pymatgen\_computedstructureentry**(*file\_name*, *\*\*kwargs*)  
Read data from `dpdata.plugins.pymatgen.PyMatgenCSEFormat` format.

**from\_pymatgen\_molecule**(*file\_name*, *\*\*kwargs*)  
Read data from `dpdata.plugins.pymatgen.PyMatgenMoleculeFormat` format.

**from\_pymatgen\_structure**(*file\_name*, *\*\*kwargs*)  
Read data from `dpdata.plugins.pymatgen.PyMatgenStructureFormat` format.

**from\_qe\_cp\_traj**(*file\_name*, *\*\*kwargs*)  
Read data from `dpdata.plugins.qe.QECPTrajFormat` format.

**from\_qe\_pw\_scf**(*file\_name*, *\*\*kwargs*)  
Read data from `dpdata.plugins.qe.QECPPWSCFFormat` format.

**from\_quip\_gap\_xyz**(*file\_name*, *\*\*kwargs*)  
Read data from `dpdata.plugins.xyz.QuipGapXYZFormat` format.

**from\_quip\_gap\_xyz\_file**(*file\_name*, *\*\*kwargs*)  
Read data from `dpdata.plugins.xyz.QuipGapXYZFormat` format.

**from\_sdf**(*file\_name*, *\*\*kwargs*)  
Read data from `dpdata.plugins.rdkit.SdfFormat` format.

**from\_sdf\_file**(*file\_name*, *\*\*kwargs*)  
Read data from `dpdata.plugins.rdkit.SdfFormat` format.

**from\_siesta\_aiMD\_output**(*file\_name*, *\*\*kwargs*)  
Read data from `dpdata.plugins.siesta.SiestaAIMDOutputFormat` format.

**from\_siesta\_aimd\_output**(*file\_name*, *\*\*kwargs*)  
Read data from `dpdata.plugins.siesta.SiestaAIMDOutputFormat` format.

**from\_siesta\_output**(*file\_name*, *\*\*kwargs*)  
Read data from `dpdata.plugins.siesta.SiestaOutputFormat` format.

**from\_sqm\_in**(*file\_name*, *\*\*kwargs*)  
Read data from `dpdata.plugins.amber.SQMInFormat` format.

**from\_sqm\_out**(*file\_name*, *\*\*kwargs*)  
Read data from `dpdata.plugins.amber.SQMOutFormat` format.

**from\_stru**(*file\_name*, *\*\*kwargs*)  
Read data from `dpdata.plugins.abacus.AbacusSTRUFormat` format.

**from\_vasp\_contcar**(*file\_name*, *\*\*kwargs*)

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

**from\_vasp\_outcar**(*file\_name*, *\*\*kwargs*)

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

**from\_vasp\_poscar**(*file\_name*, *\*\*kwargs*)

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

**from\_vasp\_string**(*file\_name*, *\*\*kwargs*)

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

**from\_vasp\_xml**(*file\_name*, *\*\*kwargs*)

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

**from\_xml**(*file\_name*, *\*\*kwargs*)

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

**from\_xyz**(*file\_name*, *\*\*kwargs*)

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

**get\_nframes**()

Returns number of frames in all systems.

**load\_systems\_from\_file**(*file\_name=None*, *fmt=None*, *\*\*kwargs*)

**minimize**(\*args: *Any*, minimizer: *str* | *Minimizer*, *\*\*kwargs: Any*) → *MultiSystems*

Minimize geometry by a minimizer.

#### Parameters

**\*args**

[iterable] Arguments passing to the minimizer

**minimizer**

[str or *Minimizer*] The assigned minimizer

**\*\*kwargs**

[dict] Other arguments passing to the minimizer

#### Returns

**MultiSystems**

A new labeled MultiSystems.

### Examples

Minimize a system using ASE BFGS along with a DP driver:

```
>>> from dpdata.driver import Driver
>>> from ase.optimize import BFGS
>>> driver = Driver.get_driver("dp")("some_model.pb")
>>> some_system.minimize(minimizer="ase", driver=driver, optimizer=BFGS,
→ fmax=1e-5)
```

**pick\_atom\_idx**(*idx*, *nopbc=None*)

Pick atom index.

#### Parameters

**idx**

[int or list or slice] atom index

**nopbc**

[Boolean (default: None)] If nopbc is True or False, set nopbc

**Returns****new\_sys: MultiSystems**

new system

**predict**(\*args: *Any*, driver='dp', \*\*kwargs: *Any*) → *MultiSystems*

Predict energies and forces by a driver.

**Parameters****\*args**

[iterable] Arguments passing to the driver

**driver**

[str, default=dp] The assigned driver. For compatibility, default is dp

**\*\*kwargs**

[dict] Other arguments passing to the driver

**Returns****MultiSystems**

A new labeled MultiSystems.

**to**(fmt: *str*, \*args, \*\*kwargs) → *MultiSystems*

Dump systems to the specific format.

**Parameters****fmt**

[str] format

**\*args**

[list] arguments

**\*\*kwargs**

[dict] keyword arguments

**Returns****MultiSystems**

self

**to\_3dmol**(\*args, \*\*kwargs)Dump data to `dpdata.plugins.3dmol.Py3DMolFormat` format.**to\_abacus\_lcao\_md**(\*args, \*\*kwargs)Dump data to `dpdata.plugins.abacus.AbacusMDFormat` format.**to\_abacus\_lcao\_relax**(\*args, \*\*kwargs)Dump data to `dpdata.plugins.abacus.AbacusRelaxFormat` format.**to\_abacus\_lcao\_scf**(\*args, \*\*kwargs)Dump data to `dpdata.plugins.abacus.AbacusSCFFormat` format.**to\_abacus\_md**(\*args, \*\*kwargs)Dump data to `dpdata.plugins.abacus.AbacusMDFormat` format.

**to\_abacus\_pw\_md**(\*args, \*\*kwargs)  
 Dump data to `dpdata.plugins.abacus.AbacusMDFormat` format.

**to\_abacus\_pw\_relax**(\*args, \*\*kwargs)  
 Dump data to `dpdata.plugins.abacus.AbacusRelaxFormat` format.

**to\_abacus\_pw\_scf**(\*args, \*\*kwargs)  
 Dump data to `dpdata.plugins.abacus.AbacusSCFFormat` format.

**to\_abacus\_relax**(\*args, \*\*kwargs)  
 Dump data to `dpdata.plugins.abacus.AbacusRelaxFormat` format.

**to\_abacus\_scf**(\*args, \*\*kwargs)  
 Dump data to `dpdata.plugins.abacus.AbacusSCFFormat` format.

**to\_abacus\_stru**(\*args, \*\*kwargs)  
 Dump data to `dpdata.plugins.abacus.AbacusSTRUFormat` format.

**to\_amber\_md**(\*args, \*\*kwargs)  
 Dump data to `dpdata.plugins.amber.AmberMDFormat` format.

**to\_ase\_structure**(\*args, \*\*kwargs)  
 Dump data to `dpdata.plugins.ase.ASEStructureFormat` format.

**to\_ase\_traj**(\*args, \*\*kwargs)  
 Dump data to `dpdata.plugins.ase.ASETrajFormat` format.

**to\_atomconfig**(\*args, \*\*kwargs)  
 Dump data to `dpdata.plugins.pwmat.PwmatAtomconfigFormat` format.

**to\_contcar**(\*args, \*\*kwargs)  
 Dump data to `dpdata.plugins.vasp.VASPPoscarFormat` format.

**to\_cp2k\_aimd\_output**(\*args, \*\*kwargs)  
 Dump data to `dpdata.plugins.cp2k.CP2KAIMDOutputFormat` format.

**to\_cp2k\_output**(\*args, \*\*kwargs)  
 Dump data to `dpdata.plugins.cp2k.CP2KOutputFormat` format.

**to\_deepmd**(\*args, \*\*kwargs)  
 Dump data to `dpdata.plugins.deepmd.DeepMDRawFormat` format.

**to\_deepmd\_comp**(\*args, \*\*kwargs)  
 Dump data to `dpdata.plugins.deepmd.DeepMDCompFormat` format.

**to\_deepmd\_hdf5**(\*args, \*\*kwargs)  
 Dump data to `dpdata.plugins.deepmd.DeepMDHDF5Format` format.

**to\_deepmd\_npy**(\*args, \*\*kwargs)  
 Dump data to `dpdata.plugins.deepmd.DeepMDCompFormat` format.

**to\_deepmd\_npy\_mixed**(\*args, \*\*kwargs)  
 Dump data to `dpdata.plugins.deepmd.DeepMDMixedFormat` format.

**to\_deepmd\_raw**(\*args, \*\*kwargs)  
 Dump data to `dpdata.plugins.deepmd.DeepMDRawFormat` format.

**to\_dftbplus**(\*args, \*\*kwargs)

Dump data to `dpdata.plugins.dftbplus.DFTBplusFormat` format.

**to\_dump**(\*args, \*\*kwargs)

Dump data to `dpdata.plugins.lammps.LAMMPSDumpFormat` format.

**to\_fhi\_aims\_md**(\*args, \*\*kwargs)

Dump data to `dpdata.plugins.fhi_aims.FhiMDFormat` format.

**to\_fhi\_aims\_output**(\*args, \*\*kwargs)

Dump data to `dpdata.plugins.fhi_aims.FhiMDFormat` format.

**to\_fhi\_aims\_scf**(\*args, \*\*kwargs)

Dump data to `dpdata.plugins.fhi_aims.FhiSCFFormat` format.

**to\_finalconfig**(\*args, \*\*kwargs)

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

**to\_fmt\_obj**(fmtobj, directory, \*args, \*\*kwargs)

**to\_gaussian\_gjf**(\*args, \*\*kwargs)

Dump data to `dpdata.plugins.gaussian.GaussianGJFFormat` format.

**to\_gaussian\_log**(\*args, \*\*kwargs)

Dump data to `dpdata.plugins.gaussian.GaussianLogFormat` format.

**to\_gaussian\_md**(\*args, \*\*kwargs)

Dump data to `dpdata.plugins.gaussian.GaussianMDFormat` format.

**to\_gro**(\*args, \*\*kwargs)

Dump data to `dpdata.plugins.gromacs.GromacsGroFormat` format.

**to\_gromacs\_gro**(\*args, \*\*kwargs)

Dump data to `dpdata.plugins.gromacs.GromacsGroFormat` format.

**to\_lammps\_dump**(\*args, \*\*kwargs)

Dump data to `dpdata.plugins.lammps.LAMMPSDumpFormat` format.

**to\_lammps\_lmp**(\*args, \*\*kwargs)

Dump data to `dpdata.plugins.lammps.LAMMPSLmpFormat` format.

**to\_list**(\*args, \*\*kwargs)

Dump data to `dpdata.plugins.list.ListFormat` format.

**to\_lmp**(\*args, \*\*kwargs)

Dump data to `dpdata.plugins.lammps.LAMMPSLmpFormat` format.

**to\_mlmd**(\*args, \*\*kwargs)

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

**to\_mol**(\*args, \*\*kwargs)

Dump data to `dpdata.plugins.rdkit.MolFormat` format.

**to\_mol\_file**(\*args, \*\*kwargs)

Dump data to `dpdata.plugins.rdkit.MolFormat` format.

**to\_movement**(\*args, \*\*kwargs)

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

**to\_n2p2**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.n2p2.N2P2Format` format.

**to\_openmx\_md**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.openmx.OPENMXFormat` format.

**to\_orca\_spout**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.orca.ORCASPOutFormat` format.

**to\_outcar**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.vasp.VASPOutcarFormat` format.

**to\_poscar**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.vasp.VASPPoscarFormat` format.

**to\_psi4\_inp**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.psi4.PSI4InputFormat` format.

**to\_psi4\_out**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.psi4.PSI4OutFormat` format.

**to\_pwmat\_atomconfig**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.pwmat.PwmatAtomconfigFormat` format.

**to\_pwmat\_finalconfig**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.pwmat.PwmatAtomconfigFormat` format.

**to\_pwmat\_mlmd**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.pwmat.PwmatOutputFormat` format.

**to\_pwmat\_movement**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.pwmat.PwmatOutputFormat` format.

**to\_pwmat\_output**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.pwmat.PwmatOutputFormat` format.

**to\_pymatgen\_ComputedStructureEntry**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.pymatgen.PyMatgenCSEFormat` format.

**to\_pymatgen\_computedstructureentry**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.pymatgen.PyMatgenCSEFormat` format.

**to\_pymatgen\_molecule**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.pymatgen.PyMatgenMoleculeFormat` format.

**to\_pymatgen\_structure**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.pymatgen.PyMatgenStructureFormat` format.

**to\_qe\_cp\_traj**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.qe.QECPTrajFormat` format.

**to\_qe\_pw\_scf**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.qe.QECPPWSCFFormat` format.

**to\_quip\_gap\_xyz**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.xyz.QuipGapXYZFormat` format.

**to\_quip\_gap\_xyz\_file**(\*args, \*\*kwargs)

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

**to\_sdf**(\*args, \*\*kwargs)

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

**to\_sdf\_file**(\*args, \*\*kwargs)

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

**to\_siesta\_aimd\_output**(\*args, \*\*kwargs)

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

**to\_siesta\_output**(\*args, \*\*kwargs)

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

**to\_sqm\_in**(\*args, \*\*kwargs)

Dump data to `dpdata.plugins.amber.SQMInFormat` format.

**to\_sqm\_out**(\*args, \*\*kwargs)

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

**to\_stru**(\*args, \*\*kwargs)

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

**to\_vasp\_contcar**(\*args, \*\*kwargs)

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

**to\_vasp\_outcar**(\*args, \*\*kwargs)

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

**to\_vasp\_poscar**(\*args, \*\*kwargs)

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

**to\_vasp\_string**(\*args, \*\*kwargs)

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

**to\_vasp\_xml**(\*args, \*\*kwargs)

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

**to\_xml**(\*args, \*\*kwargs)

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

**to\_xyz**(\*args, \*\*kwargs)

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

**train\_test\_split**(test\_size: float | int, seed: int | None = None) → Tuple[MultiSystems, MultiSystems, Dict[str, ndarray]]

Split systems into random train and test subsets.

#### Parameters

##### test\_size

[float or int] If float, should be between 0.0 and 1.0 and represent the proportion of the dataset to include in the test split. If int, represents the absolute number of test samples.

##### seed

[int, default=None] Random seed

#### Returns



**MultiSystems**

The training set

**MultiSystems**

The testing set

**Dict[str, np.ndarray]**

The bool array of training and testing sets for each system. False for training set and True for testing set.

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

Bases: MSONable

The data System.

A data System (a concept used by [deepmd-kit](#)) contains frames (e.g. produced by an MD simulation) that has the same number of atoms of the same type. The order of the atoms should be consistent among the frames in one System.

For example, a water system named *d\_example* has two molecules. The properties can be accessed by

- *d\_example*['atom\_nums']: [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 (lammmps cell tensor convention)

**Attributes****DTYPES**

[tuple[DataType]] data types of this class

**Methods**

<code>add_atom_names(atom_names)</code>	Add atom_names that do not exist.
<code>append(system)</code>	Append a system to this system.
<code>apply_pbc()</code>	Append periodic boundary condition.
<code>apply_type_map(type_map)</code>	Customize the element symbol order and it should maintain order consistency in dpn or deepmd-kit.
<code>as_dict()</code>	Returns data dict of System instance.
<code>check_data()</code>	Check if data is correct.

continues on next page

Table 8 – continued from previous page

<i>check_type_map</i> (type_map)	Assign atom_names to type_map if type_map is given and different from atom_names.
<i>convert_to_mixed_type</i> ([type_map])	Convert the data dict to mixed type format structure, in order to append systems with different formula but the same number of atoms.
<i>copy</i> ()	Returns a copy of the system.
<i>dump</i> (filename[, indent])	Dump .json or .yaml file.
<i>extend</i> (systems)	Extend a system list to this system.
<i>from_3dmol</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.3dmol.Py3DMolFormat</i> format.
<i>from_abacus_lcao_md</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.abacus.AbacusMDFormat</i> format.
<i>from_abacus_lcao_relax</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.abacus.AbacusRelaxFormat</i> format.
<i>from_abacus_lcao_scf</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.abacus.AbacusSCFFormat</i> format.
<i>from_abacus_md</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.abacus.AbacusMDFormat</i> format.
<i>from_abacus_pw_md</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.abacus.AbacusMDFormat</i> format.
<i>from_abacus_pw_relax</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.abacus.AbacusRelaxFormat</i> format.
<i>from_abacus_pw_scf</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.abacus.AbacusSCFFormat</i> format.
<i>from_abacus_relax</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.abacus.AbacusRelaxFormat</i> format.
<i>from_abacus_scf</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.abacus.AbacusSCFFormat</i> format.
<i>from_abacus_stru</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.abacus.AbacusSTRUFormat</i> format.
<i>from_amber_md</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.amber.AmberMDFormat</i> format.
<i>from_ase_structure</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.ase.ASEStructureFormat</i> format.
<i>from_ase_traj</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.ase.ASETrajFormat</i> format.
<i>from_atomconfig</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.pwmat.PwmatAtomconfigFormat</i> format.
<i>from_contcar</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.vasp.VASPPoscarFormat</i> format.
<i>from_cp2k_aimd_output</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.cp2k.CP2KAIMDOutputFormat</i> format.
<i>from_cp2k_output</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.cp2k.CP2KOutputFormat</i> format.
<i>from_deepmd</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.deepmd.DeepMDRawFormat</i> format.
<i>from_deepmd_comp</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.deepmd.DeepMDCompFormat</i> format.
<i>from_deepmd_hdf5</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.deepmd.DeepMDHDF5Format</i> format.
<i>from_deepmd_npy</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.deepmd.DeepMDCompFormat</i> format.

continues on next page

Table 8 – continued from previous page

<code>from_deepmd_npy_mixed(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.deepmd.DeePMDMixedFormat</code> format.
<code>from_deepmd_raw(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.deepmd.DeePMDRawFormat</code> format.
<code>from_dftbplus(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.dftbplus.DFTBplusFormat</code> format.
<code>from_dict(d)</code>	<p><b>param d</b> Dict representation.</p>
<code>from_dump(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.lammps.LAMMPSDumpFormat</code> format.
<code>from_fhi_aims_md(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.fhi_aims.FhiMDFormat</code> format.
<code>from_fhi_aims_output(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.fhi_aims.FhiMDFormat</code> format.
<code>from_fhi_aims_scf(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.fhi_aims.FhiSCFFormat</code> format.
<code>from_finalconfig(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.pwmat.PwmatAtomconfigFormat</code> format.
<code>from_gaussian_gjf(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.gaussian.GaussianGJFFormat</code> format.
<code>from_gaussian_log(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.gaussian.GaussianLogFormat</code> format.
<code>from_gaussian_md(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.gaussian.GaussianMDFormat</code> format.
<code>from_gro(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.gromacs.GromacsGroFormat</code> format.
<code>from_gromacs_gro(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.gromacs.GromacsGroFormat</code> format.
<code>from_lammps_dump(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.lammps.LAMMPSDumpFormat</code> format.
<code>from_lammps_lmp(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.lammps.LAMMPSLmpFormat</code> format.
<code>from_list(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.list.ListFormat</code> format.
<code>from_lmp(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.lammps.LAMMPSLmpFormat</code> format.
<code>from_mlmd(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.pwmat.PwmatOutputFormat</code> format.
<code>from_mol(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.rdkit.MolFormat</code> format.
<code>from_mol_file(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.rdkit.MolFormat</code> format.
<code>from_movement(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.pwmat.PwmatOutputFormat</code> format.
<code>from_n2p2(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.n2p2.N2P2Format</code> format.
<code>from_openmx_md(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.openmx.OPENMXFormat</code> format.
<code>from_orca_spout(file_name, **kwargs)</code>	Read data from <code>dpdata.plugins.orca.OCASPOutFormat</code> format.

continues on next page

Table 8 – continued from previous page

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

continues on next page

Table 8 – continued from previous page

<i>from_vasp_poscar</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.vasp.VASPPoscarFormat</i> format.
<i>from_vasp_string</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.vasp.VASPStringFormat</i> format.
<i>from_vasp_xml</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.vasp.VASPXMLFormat</i> format.
<i>from_xml</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.vasp.VASPXMLFormat</i> format.
<i>from_xyz</i> (file_name, **kwargs)	Read data from <i>dpdata.plugins.xyz.XYZFormat</i> format.
<i>get_atom_names</i> ()	Returns name of atoms.
<i>get_atom_numbs</i> ()	Returns number of atoms.
<i>get_atom_types</i> ()	Returns type of atoms.
<i>get_natoms</i> ()	Returns total number of atoms in the system.
<i>get_nframes</i> ()	Returns number of frames in the system.
<i>get_ntypes</i> ()	Returns total number of atom types in the system.
<i>load</i> (filename)	Rebuild System obj.
<i>map_atom_types</i> ([type_map])	Map the atom types of the system.
<i>minimize</i> (*args, minimizer, **kwargs)	Minimize the geometry.
<i>perturb</i> (pert_num, cell_pert_fraction, ..., [...])	Perturb each frame in the system randomly.
<i>pick_atom_idx</i> (idx[, nopbc])	Pick atom index.
<i>pick_by_amber_mask</i> (param, maskstr[, ...])	Pick atoms by amber mask.
<i>predict</i> (*args[, driver])	Predict energies and forces by a driver.
<i>register_data_type</i> (*data_type)	Register data type.
<i>remove_atom_names</i> (atom_names)	Remove atom names and all such atoms.
<i>remove_pbc</i> ([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 <i>protect_layer</i> (2) translates the system such that the center-of-geometry of the system locates at the center of the cell.
<i>replicate</i> (ncopy)	Replicate the each frame in the system in 3 dimensions.
<i>shuffle</i> ()	Shuffle frames randomly.
<i>sort_atom_names</i> ([type_map])	Sort atom_names of the system and reorder atom_numbs and atom_types according to atom_names.
<i>sort_atom_types</i> ()	Sort atom types.
<i>sub_system</i> (f_idx)	Construct a subsystem from the system.
<i>to</i> (fmt, *args, **kwargs)	Dump systems to the specific format.
<i>to_3dmol</i> (*args, **kwargs)	Dump data to <i>dpdata.plugins.3dmol.Py3DMolFormat</i> format.
<i>to_abacus_lcao_md</i> (*args, **kwargs)	Dump data to <i>dpdata.plugins.abacus.AbacusMDFormat</i> format.
<i>to_abacus_lcao_relax</i> (*args, **kwargs)	Dump data to <i>dpdata.plugins.abacus.AbacusRelaxFormat</i> format.
<i>to_abacus_lcao_scf</i> (*args, **kwargs)	Dump data to <i>dpdata.plugins.abacus.AbacusSCFFormat</i> format.
<i>to_abacus_md</i> (*args, **kwargs)	Dump data to <i>dpdata.plugins.abacus.AbacusMDFormat</i> format.
<i>to_abacus_pw_md</i> (*args, **kwargs)	Dump data to <i>dpdata.plugins.abacus.AbacusMDFormat</i> format.

continues on next page

Table 8 – continued from previous page

<code>to_abacus_pw_relax(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.abacus.AbacusRelaxFormat</code> format.
<code>to_abacus_pw_scf(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.abacus.AbacusSCFFormat</code> format.
<code>to_abacus_relax(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.abacus.AbacusRelaxFormat</code> format.
<code>to_abacus_scf(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.abacus.AbacusSCFFormat</code> format.
<code>to_abacus_stru(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.abacus.AbacusSTRUFormat</code> format.
<code>to_amber_md(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.amber.AmberMDFormat</code> format.
<code>to_ase_structure(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.ase.ASEStructureFormat</code> format.
<code>to_ase_traj(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.ase.ASETrajFormat</code> format.
<code>to_atomconfig(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.pwmat.PwmatAtomconfigFormat</code> format.
<code>to_contcar(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.vasp.VASPPoscarFormat</code> format.
<code>to_cp2k_aimd_output(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.cp2k.CP2KAIMDOutputFormat</code> format.
<code>to_cp2k_output(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.cp2k.CP2KOutputFormat</code> format.
<code>to_deepmd(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.deepmd.DeePMDRawFormat</code> format.
<code>to_deepmd_comp(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.deepmd.DeePMDCompFormat</code> format.
<code>to_deepmd_hdf5(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.deepmd.DeePMDHDF5Format</code> format.
<code>to_deepmd_npy(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.deepmd.DeePMDCompFormat</code> format.
<code>to_deepmd_npy_mixed(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.deepmd.DeePMDMixedFormat</code> format.
<code>to_deepmd_raw(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.deepmd.DeePMDRawFormat</code> format.
<code>to_dftbplus(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.dftbplus.DFTBplusFormat</code> format.
<code>to_dump(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.lammps.LAMMPSDumpFormat</code> format.
<code>to_fhi_aims_md(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.fhi_aims.FhiMDFormat</code> format.
<code>to_fhi_aims_output(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.fhi_aims.FhiMDFormat</code> format.
<code>to_fhi_aims_scf(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.fhi_aims.FhiSCFFormat</code> format.
<code>to_finalconfig(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.pwmat.PwmatAtomconfigFormat</code> format.
<code>to_gaussian_gjf(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.gaussian.GaussianGJFFormat</code> format.
<code>to_gaussian_log(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.gaussian.GaussianLogFormat</code> format.

continues on next page



Table 8 – continued from previous page

<code>to_gaussian_md(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.gaussian.GaussianMDFormat</code> format.
<code>to_gro(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.gromacs.GromacsGroFormat</code> format.
<code>to_gromacs_gro(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.gromacs.GromacsGroFormat</code> format.
<code>to_json()</code>	Returns a json string representation of the MSONable object.
<code>to_lammps_dump(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.lammps.LAMMPSDumpFormat</code> format.
<code>to_lammps_lmp(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.lammps.LAMMPSLmpFormat</code> format.
<code>to_list(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.list.ListFormat</code> format.
<code>to_lmp(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.lammps.LAMMPSLmpFormat</code> format.
<code>to_mlmd(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.pwmat.PwmatOutputFormat</code> format.
<code>to_mol(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.rdkit.MolFormat</code> format.
<code>to_mol_file(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.rdkit.MolFormat</code> format.
<code>to_movement(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.pwmat.PwmatOutputFormat</code> format.
<code>to_n2p2(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.n2p2.N2P2Format</code> format.
<code>to_openmx_md(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.openmx.OPENMXFormat</code> format.
<code>to_orca_spout(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.orca.ORBASPOutFormat</code> format.
<code>to_outcar(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.vasp.VASPOutcarFormat</code> format.
<code>to_poscar(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.vasp.VASPPoscarFormat</code> format.
<code>to_psi4_inp(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.psi4.PSI4InputFormat</code> format.
<code>to_psi4_out(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.psi4.PSI4OutFormat</code> format.
<code>to_pwmat_atomconfig(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.pwmat.PwmatAtomconfigFormat</code> format.
<code>to_pwmat_finalconfig(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.pwmat.PwmatAtomconfigFormat</code> format.
<code>to_pwmat_mlmd(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.pwmat.PwmatOutputFormat</code> format.
<code>to_pwmat_movement(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.pwmat.PwmatOutputFormat</code> format.
<code>to_pwmat_output(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.pwmat.PwmatOutputFormat</code> format.
<code>to_pymatgen_ComputedStructureEntry(*args, ...)</code>	Dump data to <code>dpdata.plugins.pymatgen.PyMatgenCSEFormat</code> format.
<code>to_pymatgen_computedstructureentry(*args, ...)</code>	Dump data to <code>dpdata.plugins.pymatgen.PyMatgenCSEFormat</code> format.

continues on next page

Table 8 – continued from previous page

<code>to_pymatgen_molecule(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.pymatgen.PyMatgenMoleculeFormat</code> format.
<code>to_pymatgen_structure(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.pymatgen.PyMatgenStructureFormat</code> format.
<code>to_qe_cp_traj(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.qe.QECPTrajFormat</code> format.
<code>to_qe_pw_scf(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.qe.QECPWSCFFormat</code> format.
<code>to_quip_gap_xyz(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.xyz.QuipGapXYZFormat</code> format.
<code>to_quip_gap_xyz_file(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.xyz.QuipGapXYZFormat</code> format.
<code>to_sdf(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.rdkit.SdfFormat</code> format.
<code>to_sdf_file(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.rdkit.SdfFormat</code> format.
<code>to_siesta_aimd_output(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.siesta.SiestaAIMDOutputFormat</code> format.
<code>to_siesta_output(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.siesta.SiestaOutputFormat</code> format.
<code>to_sqm_in(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.amber.SQMInFormat</code> format.
<code>to_sqm_out(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.amber.SQMOutFormat</code> format.
<code>to_stru(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.abacus.AbacusSTRUFormat</code> format.
<code>to_vasp_contcar(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.vasp.VASPPoscarFormat</code> format.
<code>to_vasp_outcar(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.vasp.VASPOutcarFormat</code> format.
<code>to_vasp_poscar(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.vasp.VASPPoscarFormat</code> format.
<code>to_vasp_string(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.vasp.VASPStringFormat</code> format.
<code>to_vasp_xml(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.vasp.VASPXMLFormat</code> format.
<code>to_xml(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.vasp.VASPXMLFormat</code> format.
<code>to_xyz(*args, **kwargs)</code>	Dump data to <code>dpdata.plugins.xyz.XYZFormat</code> format.
<code>unsafe_hash()</code>	Returns an hash of the current object.
<code>validate_monty_v1(_MSONable__input_value)</code>	Pydantic validator with correct signature for pydantic v1.x
<code>validate_monty_v2(_MSONable__input_value, _)</code>	Pydantic validator with correct signature for pydantic v2.x



<b>affine_map</b>
<b>from_fmt</b>
<b>from_fmt_obj</b>
<b>replace</b>
<b>rot_frame_lower_triangular</b>
<b>rot_lower_triangular</b>
<b>to_fmt_obj</b>

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

**add\_atom\_names**(*atom\_names*)

Add atom\_names that do not exist.

**affine\_map**(*trans*, *f\_idx=0*)

**append**(*system*)

Append a system to this system.

#### Parameters

**system**

[System] The system to append

**apply\_pbc**()

Append periodic boundary condition.

**apply\_type\_map**(*type\_map*)

Customize the element symbol order and it should maintain order consistency in dpngen or deepmd-kit. It is especially recommended for multiple complex systems with multiple elements.

#### Parameters

**type\_map**

[list] type\_map

**as\_dict**()

Returns data dict of System instance.

**check\_data**()

Check if data is correct.

#### Raises

**DataError**

if data is not correct

**check\_type\_map**(*type\_map*)

Assign atom\_names to type\_map if type\_map is given and different from atom\_names.

#### Parameters

**type\_map**

[list] type\_map

**convert\_to\_mixed\_type**(*type\_map=None*)

Convert the data dict to mixed type format structure, in order to append systems with different formula but the same number of atoms. Change the 'atom\_names' to one placeholder type 'MIXED\_TOKEN' and add 'real\_atom\_types' to store the real type vectors according to the given type\_map.

**Parameters**

**type\_map**  
[list] type\_map

**copy**()

Returns a copy of the system.

**dump**(*filename, indent=4*)

Dump .json or .yaml file.

**extend**(*systems*)

Extend a system list to this system.

**Parameters**

**systems**  
[[System1, System2, System3 ]] The list to extend

**property formula**

Return the formula of this system, like C3H5O2.

**property formula\_hash:** **str**

Return the hash of the formula of this system.

**from\_3dmol**(*file\_name, \*\*kwargs*)

Read data from [dpdata.plugins.3dmol.Py3DMolFormat](#) format.

**from\_abacus\_lcao\_md**(*file\_name, \*\*kwargs*)

Read data from [dpdata.plugins.abacus.AbacusMDFormat](#) format.

**from\_abacus\_lcao\_relax**(*file\_name, \*\*kwargs*)

Read data from [dpdata.plugins.abacus.AbacusRelaxFormat](#) format.

**from\_abacus\_lcao\_scf**(*file\_name, \*\*kwargs*)

Read data from [dpdata.plugins.abacus.AbacusSCFFormat](#) format.

**from\_abacus\_md**(*file\_name, \*\*kwargs*)

Read data from [dpdata.plugins.abacus.AbacusMDFormat](#) format.

**from\_abacus\_pw\_md**(*file\_name, \*\*kwargs*)

Read data from [dpdata.plugins.abacus.AbacusMDFormat](#) format.

**from\_abacus\_pw\_relax**(*file\_name, \*\*kwargs*)

Read data from [dpdata.plugins.abacus.AbacusRelaxFormat](#) format.

**from\_abacus\_pw\_scf**(*file\_name, \*\*kwargs*)

Read data from [dpdata.plugins.abacus.AbacusSCFFormat](#) format.

**from\_abacus\_relax**(*file\_name, \*\*kwargs*)

Read data from [dpdata.plugins.abacus.AbacusRelaxFormat](#) format.

**from\_abacus\_scf**(*file\_name, \*\*kwargs*)

Read data from [dpdata.plugins.abacus.AbacusSCFFormat](#) format.

**from\_abacus\_stru**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.abacus.AbacusSTRUFormat` format.

**from\_amber\_md**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.amber.AmberMDFormat` format.

**from\_ase\_structure**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.ase.ASEStructureFormat` format.

**from\_ase\_traj**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.ase.ASETrajFormat` format.

**from\_atomconfig**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.pwmat.PwmatAtomconfigFormat` format.

**from\_contcar**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.vasp.VASPPoscarFormat` format.

**from\_cp2k\_aimd\_output**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.cp2k.CP2KAIMDOutputFormat` format.

**from\_cp2k\_output**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.cp2k.CP2KOutputFormat` format.

**from\_deepmd**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.deepmd.DeepMDRawFormat` format.

**from\_deepmd\_comp**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.deepmd.DeepMDCompFormat` format.

**from\_deepmd\_hdf5**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.deepmd.DeepMDHDF5Format` format.

**from\_deepmd\_numpy**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.deepmd.DeepMDCompFormat` format.

**from\_deepmd\_numpy\_mixed**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.deepmd.DeepMDMixedFormat` format.

**from\_deepmd\_raw**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.deepmd.DeepMDRawFormat` format.

**from\_dftbplus**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.dftbplus.DFTBplusFormat` format.

**from\_dump**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.lammps.LAMPSDumpFormat` format.

**from\_fhi\_aims\_md**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.fhi_aims.FhiMDFormat` format.

**from\_fhi\_aims\_output**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.fhi_aims.FhiMDFormat` format.

**from\_fhi\_aims\_scf**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.fhi_aims.FhiSCFFormat` format.

**from\_finalconfig**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.pwmat.PwmatAtomconfigFormat` format.

**from\_fmt**(*file\_name*, *fmt*='auto', **\*\*kwargs**)

**from\_fmt\_obj**(*fmtobj*, *file\_name*, **\*\*kwargs**)

**from\_gaussian\_gjf**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.gaussian.GaussianGJFFormat` format.

**from\_gaussian\_log**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.gaussian.GaussianLogFormat` format.

**from\_gaussian\_md**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.gaussian.GaussianMDFormat` format.

**from\_gro**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.gromacs.GromacsGroFormat` format.

**from\_gromacs\_gro**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.gromacs.GromacsGroFormat` format.

**from\_lammps\_dump**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.lammps.LAMMPSDumpFormat` format.

**from\_lammps\_lmp**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.lammps.LAMMPSLmpFormat` format.

**from\_list**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.list.ListFormat` format.

**from\_lmp**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.lammps.LAMMPSLmpFormat` format.

**from\_mlmd**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.pwmat.PwmatOutputFormat` format.

**from\_mol**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.rdkit.MolFormat` format.

**from\_mol\_file**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.rdkit.MolFormat` format.

**from\_movement**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.pwmat.PwmatOutputFormat` format.

**from\_n2p2**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.n2p2.N2P2Format` format.

**from\_openmx\_md**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.openmx.OPENMXFormat` format.

**from\_orca\_spout**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.orca.ORBASPOutFormat` format.

**from\_outcar**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.vasp.VASPOutcarFormat` format.

**from\_poscar**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.vasp.VASPPoscarFormat` format.

**from\_psi4\_inp**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.psi4.PSI4InputFormat` format.

**from\_psi4\_out**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.psi4.PSI4OutFormat` format.

**from\_pwmat\_atomconfig**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.pwmat.PwmatAtomconfigFormat` format.

**from\_pwmat\_finalconfig**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.pwmat.PwmatAtomconfigFormat` format.

**from\_pwmat\_mlmd**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.pwmat.PwmatOutputFormat` format.

**from\_pwmat\_movement**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.pwmat.PwmatOutputFormat` format.

**from\_pwmat\_output**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.pwmat.PwmatOutputFormat` format.

**from\_pymatgen\_computedstructureentry**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.pymatgen.PyMatgenCSEFormat` format.

**from\_pymatgen\_molecule**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.pymatgen.PyMatgenMoleculeFormat` format.

**from\_pymatgen\_structure**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.pymatgen.PyMatgenStructureFormat` format.

**from\_qe\_cp\_traj**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.qe.QECPTrajFormat` format.

**from\_qe\_pw\_scf**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.qe.QECPWSCFFormat` format.

**from\_quip\_gap\_xyz**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.xyz.QuipGapXYZFormat` format.

**from\_quip\_gap\_xyz\_file**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.xyz.QuipGapXYZFormat` format.

**from\_sdf**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.rdkit.SdfFormat` format.

**from\_sdf\_file**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.rdkit.SdfFormat` format.

**from\_siesta\_aiMD\_output**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.siesta.SiestaAIMDOutputFormat` format.

**from\_siesta\_aimd\_output**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.siesta.SiestaAIMDOutputFormat` format.

**from\_siesta\_output**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.siesta.SiestaOutputFormat` format.

**from\_sqm\_in**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.amber.SQMInFormat` format.

**from\_sqm\_out**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.amber.SQMOutFormat` format.

**from\_stru**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.abacus.AbacusSTRUFormat` format.

**from\_vasp\_contcar**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.vasp.VASPPoscarFormat` format.

**from\_vasp\_outcar**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.vasp.VASPOutcarFormat` format.

**from\_vasp\_poscar**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.vasp.VASPPoscarFormat` format.

**from\_vasp\_string**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.vasp.VASPStringFormat` format.

**from\_vasp\_xml**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.vasp.VASPXMLFormat` format.

**from\_xml**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.vasp.VASPXMLFormat` format.

**from\_xyz**(*file\_name*, **\*\*kwargs**)  
Read data from `dpdata.plugins.xyz.XYZFormat` format.

**get\_atom\_names**()  
Returns name of atoms.

**get\_atom\_numbs**()  
Returns number of atoms.

**get\_atom\_types**()  
Returns type of atoms.

**get\_natoms**()  
Returns total number of atoms in the system.

**get\_nframes**()  
Returns number of frames in the system.

**get\_ntypes**() → `int`  
Returns total number of atom types in the system.

**static load**(*filename*)  
Rebuild System obj. from .json or .yaml file.

**map\_atom\_types**(*type\_map=None*) → `ndarray`  
Map the atom types of the system.

#### Parameters

**type\_map**

dict : {"H":0,"O":1} or list ["H","C","O","N"] The map between elements and index if no map\_dict is given, index will be set according to atomic number

**Returns****new\_atom\_types**

[np.ndarray] The mapped atom types

**minimize**(\*args: *Any*, minimizer: *str* | *Minimizer*, \*\*kwargs: *Any*) → *LabeledSystem*

Minimize the geometry.

**Parameters****\*args**

[iterable] Arguments passing to the minimizer

**minimizer**

[str or *Minimizer*] The assigned minimizer

**\*\*kwargs**

[dict] Other arguments passing to the minimizer

**Returns****labeled\_sys**

[*LabeledSystem*] A new labeled system.

**property nopbc**

**perturb**(*pert\_num*, *cell\_pert\_fraction*, *atom\_pert\_distance*, *atom\_pert\_style*='normal')

Perturb each frame in the system randomly. The cell will be deformed randomly, and atoms will be displaced by a random distance in random direction.

**Parameters****pert\_num**

[int] Each frame in the system will make *pert\_num* copies, and all the copies will be perturbed. That means the system to be returned will contain *pert\_num* \* *frame\_num* of the input system.

**cell\_pert\_fraction**

[float] A fraction determines how much (relatively) will cell deform. The cell of each frame is deformed by a symmetric matrix perturbed from identity. The perturbation to the diagonal part is subject to a uniform distribution in [-*cell\_pert\_fraction*, *cell\_pert\_fraction*), and the perturbation to the off-diagonal part is subject to a uniform distribution in [-0.5\**cell\_pert\_fraction*, 0.5\**cell\_pert\_fraction*).

**atom\_pert\_distance**

[float] unit: Angstrom. A distance determines how far atoms will move. Atoms will move about *atom\_pert\_distance* in random direction. The distribution of the distance atoms move is determined by *atom\_pert\_style*

**atom\_pert\_style**

[str] Determines the distribution of the distance atoms move is subject to. Available options are

- **'normal': the distance will be object to chi-square distribution with 3 degrees of freedom after normalization.**

The mean value of the distance is *atom\_pert\_fraction*\**side\_length*

- **‘uniform’**: will generate uniformly random points in a 3D-balls with radius as *atom\_pert\_distance*.  
These points are treated as vector used by atoms to move. Obviously, the max length of the distance atoms move is *atom\_pert\_distance*.
- **‘const’**: The distance atoms move will be a constant *atom\_pert\_distance*.

**Returns****perturbed\_system**

[System] The perturbed\_system. It contains *pert\_num* \* *frame\_num* of the input system frames.

**pick\_atom\_idx**(*idx*, *nopbc*=None)

Pick atom index.

**Parameters****idx**

[int or list or slice] atom index

**nopbc**

[Boolean (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**

[Boolean (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**

[Boolean (default: None)] If *nopbc* is True or False, set *nopbc*

**post\_funcs** = <dpdata.plugin.Plugin object>

**predict**(\*args: Any, driver: str = 'dp', \*\*kwargs: Any) → LabeledSystem

Predict energies and forces by a driver.

**Parameters****\*args**

[iterable] Arguments passing to the driver

**driver**

[str, default=dp] The assigned driver. For compatibility, default is dp

**\*\*kwargs**

[dict] Other arguments passing to the driver



**Returns****labeled\_sys**

[LabeledSystem] A new labeled system.

**Examples**

The default driver is DP:

```
>>> labeled_sys = ori_sys.predict("frozen_model_compressed.pb")
```

**classmethod register\_data\_type**(\*data\_type: *Tuple*[DataType])

Register data type.

**Parameters****\*data\_type**

[tuple[DataType]] data type to be registered

**remove\_atom\_names**(atom\_names)

Remove atom names and all such atoms. For example, you may not remove EP atoms in TIP4P/Ew water, which is not a real atom.

**remove\_pbc**(protect\_layer=9)

This method does NOT delete the definition of the cells, it (1) revises the cell to a cubic cell and ensures that the cell boundary to any atom in the system is no less than *protect\_layer* (2) translates the system such that the center-of-geometry of the system locates at the center of the cell.

**Parameters****protect\_layer**

[the protect layer between the atoms and the cell] boundary

**replace**(initial\_atom\_type, end\_atom\_type, replace\_num)

**replicate**(ncopy)

Replicate the each frame in the system in 3 dimensions. Each frame in the system will become a supercell.

**Parameters****ncopy**

list: [4,2,3] or tuple: (4,2,3,) make *ncopy*[0] copys in x dimensions, make *ncopy*[1] copys in y dimensions, make *ncopy*[2] copys in z dimensions.

**Returns****tmp**

[System] The system after replication.

**rot\_frame\_lower\_triangular**(f\_idx=0)

**rot\_lower\_triangular**()

**property short\_formula:** *str*

Return the short formula of this system. Elements with zero number will be removed.

**property short\_name:** *str*

Return the short name of this system (no more than 255 bytes), in the following order:

- formula

- `short_formula`
- `formula_hash`.

**shuffle()**

Shuffle frames randomly.

**sort\_atom\_names**(*type\_map=None*)

Sort `atom_names` of the system and reorder `atom_nums` and `atom_types` according to `atom_names`. If `type_map` is not given, `atom_names` will be sorted by alphabetical order. If `type_map` is given, `atom_names` will be `type_map`.

**Parameters**

**type\_map**

[list] `type_map`

**sort\_atom\_types**() → `ndarray`

Sort atom types.

**Returns**

**idx**

[`np.ndarray`] new atom index in the `Axis.NATOMS`

**sub\_system**(*f\_idx*)

Construct a subsystem from the system.

**Parameters**

**f\_idx**

[int or index] Which frame to use in the subsystem

**Returns**

**sub\_system**

[`System`] The subsystem

**to**(*fmt: str, \*args, \*\*kwargs*) → `System`

Dump systems to the specific format.

**Parameters**

**fmt**

[str] format

**\*args**

arguments

**\*\*kwargs**

keyword arguments

**Returns**

**System**

self

**to\_3dmol**(*\*args, \*\*kwargs*)

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

**to\_abacus\_lcao\_md**(*\*args, \*\*kwargs*)

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

**to\_abacus\_lcao\_relax**(\*args, \*\*kwargs)

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

**to\_abacus\_lcao\_scf**(\*args, \*\*kwargs)

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

**to\_abacus\_md**(\*args, \*\*kwargs)

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

**to\_abacus\_pw\_md**(\*args, \*\*kwargs)

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

**to\_abacus\_pw\_relax**(\*args, \*\*kwargs)

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

**to\_abacus\_pw\_scf**(\*args, \*\*kwargs)

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

**to\_abacus\_relax**(\*args, \*\*kwargs)

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

**to\_abacus\_scf**(\*args, \*\*kwargs)

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

**to\_abacus\_stru**(\*args, \*\*kwargs)

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

**to\_amber\_md**(\*args, \*\*kwargs)

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

**to\_ase\_structure**(\*args, \*\*kwargs)

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

**to\_ase\_traj**(\*args, \*\*kwargs)

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

**to\_atomconfig**(\*args, \*\*kwargs)

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

**to\_contcar**(\*args, \*\*kwargs)

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

**to\_cp2k\_aimd\_output**(\*args, \*\*kwargs)

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

**to\_cp2k\_output**(\*args, \*\*kwargs)

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

**to\_deepmd**(\*args, \*\*kwargs)

Dump data to `dpdata.plugins.deepmd.DeepMDRawFormat` format.

**to\_deepmd\_comp**(\*args, \*\*kwargs)

Dump data to `dpdata.plugins.deepmd.DeepMDCompFormat` format.

**to\_deepmd\_hdf5**(\*args, \*\*kwargs)

Dump data to `dpdata.plugins.deepmd.DeepMDHDF5Format` format.

**to\_deepmd\_npy**(\*args, \*\*kwargs)

Dump data to `dpdata.plugins.deepmd.DeepMDCompFormat` format.

**to\_deepmd\_npy\_mixed**(\*args, \*\*kwargs)

Dump data to `dpdata.plugins.deepmd.DeepMDMixedFormat` format.

**to\_deepmd\_raw**(\*args, \*\*kwargs)

Dump data to `dpdata.plugins.deepmd.DeepMDRawFormat` format.

**to\_dftbplus**(\*args, \*\*kwargs)

Dump data to `dpdata.plugins.dftbplus.DFTBplusFormat` format.

**to\_dump**(\*args, \*\*kwargs)

Dump data to `dpdata.plugins.lammps.LAMPSDumpFormat` format.

**to\_fhi\_aims\_md**(\*args, \*\*kwargs)

Dump data to `dpdata.plugins.fhi_aims.FhiMDFormat` format.

**to\_fhi\_aims\_output**(\*args, \*\*kwargs)

Dump data to `dpdata.plugins.fhi_aims.FhiMDFormat` format.

**to\_fhi\_aims\_scf**(\*args, \*\*kwargs)

Dump data to `dpdata.plugins.fhi_aims.FhiSCFFormat` format.

**to\_finalconfig**(\*args, \*\*kwargs)

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

**to\_fmt\_obj**(fmtobj, \*args, \*\*kwargs)

**to\_gaussian\_gjf**(\*args, \*\*kwargs)

Dump data to `dpdata.plugins.gaussian.GaussianGJFFormat` 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.LAMPSDumpFormat` format.

**to\_lammps\_lmp**(\*args, \*\*kwargs)

Dump data to `dpdata.plugins.lammps.LAMPSLmpFormat` format.

**to\_list**(\*args, \*\*kwargs)

Dump data to `dpdata.plugins.list.ListFormat` format.

**to\_lmp**(\*args, \*\*kwargs)

Dump data to `dpdata.plugins.lammps.LAMPSLmpFormat` format.

**to\_mlmd**(\*args, \*\*kwargs)

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

**to\_mol**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.rdkit.MolFormat` format.

**to\_mol\_file**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.rdkit.MolFormat` format.

**to\_movement**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.pwmat.PwmatOutputFormat` format.

**to\_n2p2**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.n2p2.N2P2Format` format.

**to\_openmx\_md**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.openmx.OPENMXFormat` format.

**to\_orca\_spout**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.orca.ORBASPOutFormat` format.

**to\_outcar**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.vasp.VASPOutcarFormat` format.

**to\_poscar**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.vasp.VASPPoscarFormat` format.

**to\_psi4\_inp**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.psi4.PSI4InputFormat` format.

**to\_psi4\_out**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.psi4.PSI4OutFormat` format.

**to\_pwmat\_atomconfig**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.pwmat.PwmatAtomconfigFormat` format.

**to\_pwmat\_finalconfig**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.pwmat.PwmatAtomconfigFormat` format.

**to\_pwmat\_mlmd**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.pwmat.PwmatOutputFormat` format.

**to\_pwmat\_movement**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.pwmat.PwmatOutputFormat` format.

**to\_pwmat\_output**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.pwmat.PwmatOutputFormat` format.

**to\_pymatgen\_ComputedStructureEntry**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.pymatgen.PyMatgenCSEFormat` format.

**to\_pymatgen\_computedstructureentry**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.pymatgen.PyMatgenCSEFormat` format.

**to\_pymatgen\_molecule**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.pymatgen.PyMatgenMoleculeFormat` format.

**to\_pymatgen\_structure**(\*args, \*\*kwargs)  
Dump data to `dpdata.plugins.pymatgen.PyMatgenStructureFormat` format.

**to\_qe\_cp\_traj**(\*args, \*\*kwargs)

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

**to\_qe\_pw\_scf**(\*args, \*\*kwargs)

Dump data to `dpdata.plugins.qe.QECPWSCFFormat` format.

**to\_quip\_gap\_xyz**(\*args, \*\*kwargs)

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

**to\_quip\_gap\_xyz\_file**(\*args, \*\*kwargs)

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

**to\_sdf**(\*args, \*\*kwargs)

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

**to\_sdf\_file**(\*args, \*\*kwargs)

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

**to\_siesta\_aimd\_output**(\*args, \*\*kwargs)

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

**to\_siesta\_output**(\*args, \*\*kwargs)

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

**to\_sqm\_in**(\*args, \*\*kwargs)

Dump data to `dpdata.plugins.amber.SQMInFormat` format.

**to\_sqm\_out**(\*args, \*\*kwargs)

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

**to\_stru**(\*args, \*\*kwargs)

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

**to\_vasp\_contcar**(\*args, \*\*kwargs)

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

**to\_vasp\_outcar**(\*args, \*\*kwargs)

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

**to\_vasp\_poscar**(\*args, \*\*kwargs)

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

**to\_vasp\_string**(\*args, \*\*kwargs)

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

**to\_vasp\_xml**(\*args, \*\*kwargs)

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

**to\_xml**(\*args, \*\*kwargs)

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

**to\_xyz**(\*args, \*\*kwargs)

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

**property uniq\_formula**

Return the `uniq_formula` of this system. The `uniq_formula` sort the elements in formula by names. Systems with the same `uniq_formula` can be append together.

**dpdata.system.add\_format\_methods()**

Add format methods to System, LabeledSystem, and MultiSystems; add data types to System and LabeledSystem.

## Notes

Ensure all plugins have been loaded before executing this function!

`dpdata.system.get_atom_perturb_vector(atom_pert_distance, atom_pert_style='normal')`

`dpdata.system.get_cell_perturb_matrix(cell_pert_fraction)`

`dpdata.system.get_cls_name(cls: object) → str`

Returns the fully qualified name of a class, such as `np.ndarray`.

### Parameters

**cls**  
[object] the class

### Returns

**str**  
the fully qualified name of a class

`dpdata.system.load_format(fmt)`

## 6.1.13 dpdata.unit module

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

Bases: `ABC`

### Methods

<b>set_value</b>
<b>value</b>

**set\_value**(value)

**value**()

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

Bases: `Conversion`

### Methods

<b>set_value</b>
<b>value</b>

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

Bases: `Conversion`

## Methods

set_value
value

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

Bases: [Conversion](#)

## Methods

set_value
value

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

Bases: [Conversion](#)

## Methods

set_value
value

```
dpdata.unit.check_unit(unit)
```

## 6.1.14 dpdata.utils module

```
dpdata.utils.add_atom_names(data, atom_names)
```

Add atom\_names that do not exist.

```
dpdata.utils.elements_index_map(elements, standard=False, inverse=False)
```

```
dpdata.utils.remove_pbc(system, protect_layer=9)
```

```
dpdata.utils.sort_atom_names(data, type_map=None)
```

Sort atom\_names of the system and reorder atom\_nums and atom\_types according to atom\_names. If type\_map is not given, atom\_names will be sorted by alphabetical order. If type\_map is given, atom\_names will be type\_map.

### Parameters

**data**

[dict] system data

**type\_map**

[list] type\_map



`dpdata.utils.uniq_atom_names(data)`

Make the atom names uniq. For example ['O', 'H', 'O', 'H', 'O'] -> ['O', 'H'].

**Parameters**

**data**

[dict] data dict of *System*, *LabeledSystem*

`dpdata.utils.utf8len(s: str) → int`

Return the byte length of a string.



## AUTHORS

- A bot of @njzjz
- AngelJia
- AnguseZhang
- C. Thang Nguyen
- Chen Tao
- Chentao168
- Chenxing Luo
- DULinhan
- Duo
- Ericwang6
- Feifei Tian
- Han Wang
- Haruka
- Hedley
- Hedley Dong
- HuangJiameng
- Jia-Xin Zhu
- Jingchao Zhang
- Jinzhe Zeng
- Junhan Chang
- L-RuiHao
- Levi Zhou
- LiangWenshuo1118
- Linfeng Zhang
- Liu Renxi
- Liu-RX
- LiuGroupHNU

- LiuHanyu
- LiuLiping
- Logan Ward
- Marián Rynik
- PKUfjh
- Pan Xiang
- Peng Xingliang
- Shigetomo Yanase
- Silvia-liu
- Wanrun Jiang
- Wenxin Zhang
- Yifan Li
- Yingze Wang
- Yixiao Chen
- Yongbin Zhuang
- Yu Liu
- Yuan Fengbo
- Zezhong Zhang
- deepmodeling
- dependabot[bot]
- ericwang6
- felix5572
- haidi
- hl2500
- hongriTianqi
- jameswind
- kiwi
- link89
- liuliping
- m0sey
- pee8379
- pre-commit-ci[bot]
- pxlxingliang
- repo-ranger[bot]
- robinzhuang
- robinzyb

- tianhongzhen
- tuoping
- wang laosi
- xfanak
- yuzhi
- zezhong-zhang



**dpdata** is a python package for manipulating data formats of software in computational science, including DeePMD-kit, VASP, LAMMPS, GROMACS, Gaussian. dpdata only works with python 3.7 or above.

## 8.1 Installation

One can download the source code of dpdata by

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

then use pip to install the module from source

```
cd dpdata
pip install .
```

dpdata can also be installed via pip without source

```
pip install dpdata
```

## 8.2 Quick start

This section gives some examples on how dpdata works. Firstly one needs to import the module in a python 3.x compatible code.

```
import dpdata
```

The typical workflow of dpdata is

1. Load data from vasp or lammmps or deepmd-kit data files.
2. Manipulate data
3. Dump data to in a desired format

## 8.2.1 Load data

```
d_poscar = dpdata.System("POSCAR", fmt="vasp/poscar")
```

or let dpdata infer the format (vasp/poscar) of the file from the file name extension

```
d_poscar = dpdata.System("my.POSCAR")
```

The number of atoms, atom types, coordinates are loaded from the POSCAR and stored to a data System called `d_poscar`. A data System (a concept used by `deepmd-kit`) contains frames that has the same number of atoms of the same type. The order of the atoms should be consistent among the frames in one System. It is noted that POSCAR only contains one frame. If the multiple frames stored in, for example, a OUTCAR is wanted,

```
d_outcar = dpdata.LabeledSystem("OUTCAR")
```

The labels provided in the OUTCAR, i.e. energies, forces and virials (if any), are loaded by `LabeledSystem`. It is noted that the forces of atoms are always assumed to exist. `LabeledSystem` is a derived class of `System`.

The `System` or `LabeledSystem` can be constructed from the following file formats with the `format` key in the table passed to argument `fmt`:

Software	format	multi frames	labeled	class	format key
vasp	poscar	False	False	System	'vasp/poscar'
vasp	outcar	True	True	LabeledSystem	'vasp/outcar'
vasp	xml	True	True	LabeledSystem	'vasp/xml'
lammps	lmp	False	False	System	'lammps/lmp'
lammps	dump	True	False	System	'lammps/dump'
deepmd	raw	True	False	System	'deepmd/raw'
deepmd	npz	True	False	System	'deepmd/npz'
deepmd	raw	True	True	LabeledSystem	'deepmd/raw'
deepmd	npz	True	True	LabeledSystem	'deepmd/npz'
deepmd	npz	True	True	MultiSystems	'deepmd/npz/mixed'
deepmd	npz	True	False	MultiSystems	'deepmd/npz/mixed'
gaussian	log	False	True	LabeledSystem	'gaussian/log'
gaussian	log	True	True	LabeledSystem	'gaussian/md'
siesta	output	False	True	LabeledSystem	'siesta/output'
siesta	aimd_output	True	True	LabeledSystem	'siesta/aimd_output'
cp2k(deprecated in future)	output	False	True	LabeledSystem	'cp2k/output'
cp2k(deprecated in future)	aimd_output	True	True	LabeledSystem	'cp2k/aimd_output'
cp2k(plug-in)	stdout	False	True	LabeledSystem	'cp2kdata/e_f'
cp2k(plug-in)	stdout	True	True	LabeledSystem	'cp2kdata/md'
QE	log	False	True	LabeledSystem	'qe/pw/scf'
QE	log	True	False	System	'qe/cp/traj'
QE	log	True	True	LabeledSystem	'qe/cp/traj'
Fhi-aims	output	True	True	LabeledSystem	'fhi_aims/md'
Fhi-aims	output	False	True	LabeledSystem	'fhi_aims/scf'
quip/gap	xyz	True	True	MultiSystems	'quip/gap/xyz'
PWmat	atom.config	False	False	System	'pwmat/atom.config'
PWmat	movement	True	True	LabeledSystem	'pwmat/movement'
PWmat	OUT.MLMD	True	True	LabeledSystem	'pwmat/out.mlmd'
Amber	multi	True	True	LabeledSystem	'amber/md'
Amber/sqm	sqm.out	False	False	System	'sqm/out'
Gromacs	gro	True	False	System	'gromacs/gro'

continues on next page



Table 1 – continued from previous page

Software	format	multi frames	labeled	class	format key
ABACUS	STRU	False	False	System	'abacus/stru'
ABACUS	STRU	False	True	LabeledSystem	'abacus/scf'
ABACUS	cif	True	True	LabeledSystem	'abacus/md'
ABACUS	STRU	True	True	LabeledSystem	'abacus/relax'
ase	structure	True	True	MultiSystems	'ase/structure'
DFTB+	dftbplus	False	True	LabeledSystem	'dftbplus'
n2p2	n2p2	True	True	LabeledSystem	'n2p2'

The Class `dpdata.MultiSystems` can read data from a dir which may contains many files of different systems, or from single xyz file which contains different systems.

Use `dpdata.MultiSystems.from_dir` to read from a directory, `dpdata.MultiSystems` will walk in the directory Recursively and find all file with specific file\_name. Supports all the file formats that `dpdata.LabeledSystem` supports.

Use `dpdata.MultiSystems.from_file` to read from single file. Single-file support is available for the `quip/gap/xyz` and `ase/structure` formats.

For example, for `quip/gap xyz` files, single .xyz file may contain many different configurations with different atom numbers and atom type.

The following commands relating to Class `dpdata.MultiSystems` may be useful.

```
# load data

xyz_multi_systems = dpdata.MultiSystems.from_file(
    file_name="tests/xyz/xyz_unittest.xyz", fmt="quip/gap/xyz"
)
vasp_multi_systems = dpdata.MultiSystems.from_dir(
    dir_name="./mgal_outcar", file_name="OUTCAR", fmt="vasp/outcar"
)

# use wildcard
vasp_multi_systems = dpdata.MultiSystems.from_dir(
    dir_name="./mgal_outcar", file_name="*OUTCAR", fmt="vasp/outcar"
)

# print the multi_system infomation
print(xyz_multi_systems)
print(xyz_multi_systems.systems) # return a dictionaries

# print the system infomation
print(xyz_multi_systems.systems["B1C9"].data)

# dump a system's data to ./my_work_dir/B1C9_raw folder
xyz_multi_systems.systems["B1C9"].to_deepmd_raw("./my_work_dir/B1C9_raw")

# dump all systems
xyz_multi_systems.to_deepmd_raw("./my_deepmd_data/")
```

You may also use the following code to parse muti-system:

```

from dpdata import LabeledSystem, MultiSystems
from glob import glob

"""
process multi systems
"""
fs = glob("./*/OUTCAR") # remeber to change here !!!
ms = MultiSystems()
for f in fs:
    try:
        ls = LabeledSystem(f)
    except:
        print(f)
    if len(ls) > 0:
        ms.append(ls)

ms.to_deepmd_raw("deepmd")
ms.to_deepmd_npy("deepmd")

```

## 8.2.2 Access data

These properties stored in `System` and `LabeledSystem` can be accessed by operator `[]` with the key of the property supplied, for example

```
coords = d_outcar["coords"]
```

Available properties are (nframe: number of frames in the system, natoms: total number of atoms in the system)

key	type	dimension	are labels	description
'atom_names'	list of str	ntypes	False	The name of each atom type
'atom_numbs'	list of int	ntypes	False	The number of atoms of each atom type
'atom_types'	np.ndarray	natoms	False	Array assigning type to each atom
'cells'	np.ndarray	nframes x 3 x 3	False	The cell tensor of each frame
'coords'	np.ndarray	nframes x natoms x 3	False	The atom coordinates
'energies'	np.ndarray	nframes	True	The frame energies
'forces'	np.ndarray	nframes x natoms x 3	True	The atom forces
'virials'	np.ndarray	nframes x 3 x 3	True	The virial tensor of each frame

## 8.2.3 Dump data

The data stored in `System` or `LabeledSystem` can be dumped in 'lammps/lmp' or 'vasp/poscar' format, for example:

```
d_outcar.to("lammps/lmp", "conf.lmp", frame_idx=0)
```

The first frames of `d_outcar` will be dumped to 'conf.lmp'

```
d_outcar.to("vasp/poscar", "POSCAR", frame_idx=-1)
```

The last frames of `d_outcar` will be dumped to 'POSCAR'.

The data stored in `LabeledSystem` can be dumped to deepmd-kit raw format, for example

```
d_outcar.to("deepmd/raw", "dpmd_raw")
```

Or a simpler command:

```
dpdata.LabeledSystem("OUTCAR").to("deepmd/raw", "dpmd_raw")
```

Frame selection can be implemented by

```
dpdata.LabeledSystem("OUTCAR").sub_system([0, -1]).to("deepmd/raw", "dpmd_raw")
```

by which only the first and last frames are dumped to dpmd\_raw.

## 8.2.4 replicate

dpdata will create a super cell of the current atom configuration.

```
dpdata.System("./POSCAR").replicate(
    (
        1,
        2,
        3,
    )
)
```

tuple(1,2,3) means don't copy atom configuration in x direction, make 2 copys in y direction, make 3 copys in z direction.

## 8.2.5 perturb

By the following example, each frame of the original system (`dpdata.System('./POSCAR')`) is perturbed to generate three new frames. For each frame, the cell is perturbed by 5% and the atom positions are perturbed by 0.6 Angstrom. `atom_pert_style` indicates that the perturbation to the atom positions is subject to normal distribution. Other available options to `atom_pert_style` are `uniform` (uniform in a ball), and `const` (uniform on a sphere).

```
perturbed_system = dpdata.System("./POSCAR").perturb(
    pert_num=3,
    cell_pert_fraction=0.05,
    atom_pert_distance=0.6,
    atom_pert_style="normal",
)
print(perturbed_system.data)
```

## 8.2.6 replace

By the following example, Random 8 Hf atoms in the system will be replaced by Zr atoms with the atom position unchanged.

```
s = dpdata.System("tests/poscars/POSCAR.P42nmc", fmt="vasp/poscar")
s.replace("Hf", "Zr", 8)
s.to_vasp_poscar("POSCAR.P42nmc.replace")
```

## 8.3 BondOrderSystem

A new class `BondOrderSystem` which inherits from class `System` is introduced in `dpdata`. This new class contains information of chemical bonds and formal charges (stored in `BondOrderSystem.data['bonds']`, `BondOrderSystem.data['formal_charges']`). Now `BondOrderSystem` can only read from `.mol/.sdf` formats, because of its dependency on `rdkit` (which means `rdkit` must be installed if you want to use this function). Other formats, such as `pdb`, must be converted to `.mol/.sdf` format (maybe with software like `open babel`).

```
import dpdata

system_1 = dpdata.BondOrderSystem(
    "tests/bond_order/CH3OH.mol", fmt="mol"
) # read from .mol file
system_2 = dpdata.BondOrderSystem(
    "tests/bond_order/methane.sdf", fmt="sdf"
) # read from .sdf file
```

In `sdf` file, all molecules must be of the same topology (i.e. conformers of the same molecular configuration). `BondOrderSystem` also supports initialize from a `rdkit.Chem.rdchem.Mol` object directly.

```
from rdkit import Chem
from rdkit.Chem import AllChem
import dpdata

mol = Chem.MolFromSmiles("CC")
mol = Chem.AddHs(mol)
AllChem.EmbedMultipleConfs(mol, 10)
system = dpdata.BondOrderSystem(rdkit_mol=mol)
```

### 8.3.1 Bond Order Assignment

The `BondOrderSystem` implements a more robust sanitize procedure for `rdkit Mol`, as defined in `dpdata.rdkit.santizie.Sanitizer`. This class defines 3 level of sanitization process by: low, medium and high. (default is medium).

- low: use `rdkit.Chem.SanitizeMol()` function to sanitize molecule.
- medium: before using `rdkit`, the program 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 hetrocycles, 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 want to use this level of sanitization, please ensure ``obabel`` is installed in the environment.** According to our test, our sanitization procedure can successfully read 4852 small molecules in the `PDBBind-refined-set`. It is necessary to point out that the in the molecule file (`mol/sdf`), the number of explicit hydrogens has to be correct. Thus, we recommend to use `obabel xxx -O xxx -h` to pre-process the file. The reason why we do not implement this hydrogen-adding procedure in `dpdata` is that we can not ensure its correctness.

```
import dpdata

for sdf_file in glob.glob("bond_order/refined-set-ligands/obabel/*.sdf"):
    syst = dpdata.BondOrderSystem(sdf_file, sanitize_level="high", verbose=False)
```

### 8.3.2 Formal Charge Assignment

BondOrderSystem implement a method to assign formal charge for each atom based on the 8-electron rule (see below). Note that it only supports common elements in bio-system: B,C,N,O,P,S,As

```
import dpdata

syst = dpdata.BondOrderSystem("tests/bond_order/CH3NH3+.mol", fmt="mol")
print(syst.get_formal_charges()) # return the formal charge on each atom
print(syst.get_charge()) # return the total charge of the system
```

If a valence of 3 is detected on carbon, the formal charge will be assigned to -1. Because for most cases (in alkynyl anion, isonitrile, cyclopentadienyl anion), the formal charge on 3-valence carbon is -1, and this is also consistent with the 8-electron rule.

## 8.4 Mixed Type Format

The format deepmd/npz/mixed is the mixed type numpy format for DeePMD-kit, and can be loaded or dumped through class dpdata.MultiSystems.

Under this format, systems with the same number of atoms but different formula can be put together for a larger system, especially when the frame numbers in systems are sparse.

This also helps to mixture the type information together for model training with type embedding network.

Here are examples using deepmd/npz/mixed format:

- Dump a MultiSystems into a mixed type numpy directory: `python import dpdata`  
dpdata.MultiSystems(\*systems).to\_deepmd\_npy\_mixed("mixed\_dir")

```
- Load a mixed type data into a MultiSystems:
python
import dpdata

dpdata.MultiSystems().load_systems_from_file("mixed_dir", fmt="deepmd/npz/mixed")
```

## 8.5 Plugins

One can follow a [simple example](#) to add their own format by creating and installing plugins. It's critical to add the Format class to entry\_points['dpdata.plugins'] in `pyproject.toml` <plugin\_example/pyproject.toml>\_:

```
[project.entry-points.'dpdata.plugins']
random = "dpdata_random:RandomFormat"
```



## INDICES AND TABLES

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





## BIBLIOGRAPHY

- [1] Gao, X.; Ramezanghorbani, F.; Isayev, O.; Smith, J. S.; Roitberg, A. E. TorchANI: A Free and Open Source PyTorch-Based Deep Learning Implementation of the ANI Neural Network Potentials. *J. Chem. Inf. Model.* 2020, 60, 3408-3415.
- [2] Zeng, J.; Tao, Y.; Giese, T. J.; York, D. M.. QD: A Quantum Deep Potential Interaction Model for Drug Discovery. *J. Comput. Chem.* 2023, 19, 1261-1275.
- [1] Gao, X.; Ramezanghorbani, F.; Isayev, O.; Smith, J. S.; Roitberg, A. E. TorchANI: A Free and Open Source PyTorch-Based Deep Learning Implementation of the ANI Neural Network Potentials. *J. Chem. Inf. Model.* 2020, 60, 3408-3415.
- [2] Zeng, J.; Tao, Y.; Giese, T. J.; York, D. M.. QD: A Quantum Deep Potential Interaction Model for Drug Discovery. *J. Comput. Chem.* 2023, 19, 1261-1275.



## PYTHON MODULE INDEX

### d

dpdata, 57  
dpdata.abacus, 123  
dpdata.abacus.md, 123  
dpdata.abacus.relax, 123  
dpdata.abacus.scf, 123  
dpdata.amber, 124  
dpdata.amber.mask, 124  
dpdata.amber.md, 124  
dpdata.amber.sqm, 125  
dpdata.bond\_order\_system, 206  
dpdata.cli, 215  
dpdata.cp2k, 125  
dpdata.cp2k.cell, 125  
dpdata.cp2k.output, 125  
dpdata.data\_type, 216  
dpdata.deepmd, 126  
dpdata.deepmd.comp, 126  
dpdata.deepmd.hdf5, 126  
dpdata.deepmd.mixed, 127  
dpdata.deepmd.raw, 128  
dpdata.dftbplus, 128  
dpdata.dftbplus.output, 128  
dpdata.driver, 218  
dpdata.fhi\_aims, 128  
dpdata.fhi\_aims.output, 128  
dpdata.format, 221  
dpdata.gaussian, 129  
dpdata.gaussian.gjf, 129  
dpdata.gaussian.log, 130  
dpdata.gromacs, 130  
dpdata.gromacs.gro, 130  
dpdata.lammps, 131  
dpdata.lammps.dump, 131  
dpdata.lammps.lmp, 131  
dpdata.openmx, 132  
dpdata.openmx.omx, 132  
dpdata.orca, 132  
dpdata.orca.output, 132  
dpdata.periodic\_table, 227  
dpdata.plugin, 228  
dpdata.plugins, 133  
dpdata.plugins.3dmol, 133  
dpdata.plugins.abacus, 134  
dpdata.plugins.amber, 138  
dpdata.plugins.ase, 143  
dpdata.plugins.cp2k, 148  
dpdata.plugins.deepmd, 151  
dpdata.plugins.dftbplus, 160  
dpdata.plugins.fhi\_aims, 160  
dpdata.plugins.gaussian, 163  
dpdata.plugins.gromacs, 167  
dpdata.plugins.lammps, 168  
dpdata.plugins.list, 170  
dpdata.plugins.n2p2, 171  
dpdata.plugins.openmx, 173  
dpdata.plugins.orca, 174  
dpdata.plugins.psi4, 176  
dpdata.plugins.pwmat, 178  
dpdata.plugins.pymatgen, 181  
dpdata.plugins.qe, 184  
dpdata.plugins.rdkit, 186  
dpdata.plugins.siesta, 188  
dpdata.plugins.vasp, 191  
dpdata.plugins.xyz, 196  
dpdata.psi4, 199  
dpdata.psi4.input, 199  
dpdata.psi4.output, 199  
dpdata.pwmat, 200  
dpdata.pwmat.atomconfig, 200  
dpdata.pwmat.movement, 200  
dpdata.pymatgen, 200  
dpdata.pymatgen.molecule, 200  
dpdata.pymatgen.structure, 200  
dpdata.qe, 200  
dpdata.qe.scf, 200  
dpdata.qe.traj, 201  
dpdata.rdkit, 201  
dpdata.rdkit.sanitize, 201  
dpdata.rdkit.utils, 203  
dpdata.siesta, 203  
dpdata.siesta.aiMD\_output, 203  
dpdata.siesta.output, 204  
dpdata.stat, 228

`dpdata.system`, [232](#)  
`dpdata.unit`, [289](#)  
`dpdata.utils`, [290](#)  
`dpdata.vasp`, [204](#)  
`dpdata.vasp.outcar`, [204](#)  
`dpdata.vasp.poscar`, [204](#)  
`dpdata.vasp.xml`, [204](#)  
`dpdata.xyz`, [205](#)  
`dpdata.xyz.quiv_gap_xyz`, [205](#)  
`dpdata.xyz.xyz`, [205](#)

## A

AbacusMDFormat (class in *dpdata.plugins.abacus*), 134  
 AbacusRelaxFormat (class in *dpdata.plugins.abacus*), 135  
 AbacusSCFFormat (class in *dpdata.plugins.abacus*), 136  
 AbacusSTRUFormat (class in *dpdata.plugins.abacus*), 137  
 add\_atom\_names() (*dpdata.System* method), 109  
 add\_atom\_names() (*dpdata.system.System* method), 275  
 add\_atom\_names() (in module *dpdata.utils*), 290  
 add\_format\_methods() (in module *dpdata.system*), 288  
 affine\_map() (*dpdata.System* method), 109  
 affine\_map() (*dpdata.system.System* method), 275  
 affine\_map\_fv() (*dpdata.LabeledSystem* method), 75  
 affine\_map\_fv() (*dpdata.system.LabeledSystem* method), 241  
 AmberMDFormat (class in *dpdata.plugins.amber*), 138  
 analyze() (in module *dpdata.vasp.xml*), 204  
 analyze\_atominfo() (in module *dpdata.vasp.xml*), 204  
 analyze\_block() (in module *dpdata.fhi\_aims.output*), 128  
 analyze\_block() (in module *dpdata.pwmat.movement*), 200  
 analyze\_block() (in module *dpdata.vasp.outcar*), 204  
 analyze\_calculation() (in module *dpdata.vasp.xml*), 204  
 AnyInt (class in *dpdata.data\_type*), 216  
 append() (*dpdata.MultiSystems* method), 90  
 append() (*dpdata.System* method), 109  
 append() (*dpdata.system.MultiSystems* method), 256  
 append() (*dpdata.system.System* method), 275  
 apply\_pbc() (*dpdata.System* method), 109  
 apply\_pbc() (*dpdata.system.System* method), 275  
 apply\_type\_map() (*dpdata.System* method), 109  
 apply\_type\_map() (*dpdata.system.System* method), 275  
 as\_dict() (*dpdata.System* method), 109  
 as\_dict() (*dpdata.system.System* method), 275  
 ase\_calculator (*dpdata.driver.Driver* property), 218  
 ASEDriver (class in *dpdata.plugins.ase*), 143  
 ASEMinimizer (class in *dpdata.plugins.ase*), 144  
 ASEStructureFormat (class in *dpdata.plugins.ase*), 145  
 ASETrajFormat (class in *dpdata.plugins.ase*), 146

assign\_formal\_charge\_for\_atom() (in module *dpdata.rdkit.sanitize*), 202  
 Axis (class in *dpdata.data\_type*), 216

## B

BondOrderSystem (class in *dpdata*), 57  
 BondOrderSystem (class in *dpdata.bond\_order\_system*), 206  
 box2dumpbox() (in module *dpdata.lammps.dump*), 131  
 box2lmpbox() (in module *dpdata.lammps.lmp*), 131

## C

calculated\_radius (*dpdata.periodic\_table.Element* property), 227  
 cell\_to\_low\_triangle() (in module *dpdata.cp2k.cell*), 125  
 check() (*dpdata.data\_type.DataType* method), 217  
 check\_atom\_names() (*dpdata.MultiSystems* method), 90  
 check\_atom\_names() (*dpdata.system.MultiSystems* method), 256  
 check\_data() (*dpdata.System* method), 109  
 check\_data() (*dpdata.system.System* method), 275  
 check\_molecule\_list() (in module *dpdata.rdkit.utils*), 203  
 check\_name() (in module *dpdata.vasp.xml*), 204  
 check\_same\_atom() (in module *dpdata.rdkit.utils*), 203  
 check\_same\_molecule() (in module *dpdata.rdkit.utils*), 203  
 check\_type\_map() (*dpdata.System* method), 109  
 check\_type\_map() (*dpdata.system.System* method), 275  
 check\_unit() (in module *dpdata.unit*), 290  
 CheckFile() (in module *dpdata.abacus.scf*), 123  
 collect\_force() (in module *dpdata.abacus.scf*), 123  
 collect\_stress() (in module *dpdata.abacus.scf*), 123  
 combine\_molecules() (in module *dpdata.rdkit.utils*), 203  
 contain\_hetero\_aromatic() (in module *dpdata.rdkit.sanitize*), 202  
 Conversion (class in *dpdata.unit*), 289  
 convert() (in module *dpdata.cli*), 215

`convert_by_obabel()` (in module `dpdata.rdkit.sanitize`), 202  
`convert_celldm()` (in module `dpdata.qe.traj`), 201  
`convert_to_mixed_type()` (`dpdata.System` method), 109  
`convert_to_mixed_type()` (`dpdata.system.System` method), 275  
`coord_to_xyz()` (in module `dpdata.xyz.xyz`), 205  
`copy()` (`dpdata.bond_order_system.BondOrderSystem` method), 214  
`copy()` (`dpdata.BondOrderSystem` method), 65  
`copy()` (`dpdata.System` method), 110  
`copy()` (`dpdata.system.System` method), 276  
`correction()` (`dpdata.LabeledSystem` method), 75  
`correction()` (`dpdata.MultiSystems` method), 90  
`correction()` (`dpdata.system.LabeledSystem` method), 241  
`correction()` (`dpdata.system.MultiSystems` method), 256  
`covert_dimension()` (in module `dpdata.siesta.aiMD_output`), 203  
`CP2KAIMDOutputFormat` (class in `dpdata.plugins.cp2k`), 148  
`CP2KOutputFormat` (class in `dpdata.plugins.cp2k`), 149  
`Cp2kSystems` (class in `dpdata.cp2k.output`), 125

## D

`DataError`, 216  
`DataType` (class in `dpdata.data_type`), 216  
`DeePMDCompFormat` (class in `dpdata.plugins.deepmd`), 151  
`DeePMDHDF5Format` (class in `dpdata.plugins.deepmd`), 153  
`DeePMDMixedFormat` (class in `dpdata.plugins.deepmd`), 156  
`DeePMDRawFormat` (class in `dpdata.plugins.deepmd`), 158  
`detect_multiplicity()` (in module `dpdata.gaussian.gjf`), 129  
`DFTBplusFormat` (class in `dpdata.plugins.dftbplus`), 160  
`Directory` (`dpdata.format.Format.MultiModes` attribute), 222  
`dpdata`  
  module, 57  
`dpdata.abacus`  
  module, 123  
`dpdata.abacus.md`  
  module, 123  
`dpdata.abacus.relax`  
  module, 123  
`dpdata.abacus.scf`  
  module, 123  
`dpdata.amber`  
  module, 124  
`dpdata.amber.mask`  
  module, 124  
`dpdata.amber.md`  
  module, 124  
`dpdata.amber.sqm`  
  module, 125  
`dpdata.bond_order_system`  
  module, 206  
`dpdata.cli`  
  module, 215  
`dpdata.cp2k`  
  module, 125  
`dpdata.cp2k.cell`  
  module, 125  
`dpdata.cp2k.output`  
  module, 125  
`dpdata.data_type`  
  module, 216  
`dpdata.deepmd`  
  module, 126  
`dpdata.deepmd.comp`  
  module, 126  
`dpdata.deepmd.hdf5`  
  module, 126  
`dpdata.deepmd.mixed`  
  module, 127  
`dpdata.deepmd.raw`  
  module, 128  
`dpdata.dftbplus`  
  module, 128  
`dpdata.dftbplus.output`  
  module, 128  
`dpdata.driver`  
  module, 218  
`dpdata.fhi_aims`  
  module, 128  
`dpdata.fhi_aims.output`  
  module, 128  
`dpdata.format`  
  module, 221  
`dpdata.gaussian`  
  module, 129  
`dpdata.gaussian.gjf`  
  module, 129  
`dpdata.gaussian.log`  
  module, 130  
`dpdata.gromacs`  
  module, 130  
`dpdata.gromacs.gro`  
  module, 130  
`dpdata.lammps`  
  module, 131  
`dpdata.lammps.dump`  
  module, 131

dpdata.lammps.lmp  
  module, 131

dpdata.openmx  
  module, 132

dpdata.openmx.omx  
  module, 132

dpdata.orca  
  module, 132

dpdata.orca.output  
  module, 132

dpdata.periodic\_table  
  module, 227

dpdata.plugin  
  module, 228

dpdata.plugins  
  module, 133

dpdata.plugins.3dmol  
  module, 133

dpdata.plugins.abacus  
  module, 134

dpdata.plugins.amber  
  module, 138

dpdata.plugins.ase  
  module, 143

dpdata.plugins.cp2k  
  module, 148

dpdata.plugins.deepmd  
  module, 151

dpdata.plugins.dftbplus  
  module, 160

dpdata.plugins.fhi\_aims  
  module, 160

dpdata.plugins.gaussian  
  module, 163

dpdata.plugins.gromacs  
  module, 167

dpdata.plugins.lammps  
  module, 168

dpdata.plugins.list  
  module, 170

dpdata.plugins.np2  
  module, 171

dpdata.plugins.openmx  
  module, 173

dpdata.plugins.orca  
  module, 174

dpdata.plugins.psi4  
  module, 176

dpdata.plugins.pwmat  
  module, 178

dpdata.plugins.pymatgen  
  module, 181

dpdata.plugins.qe  
  module, 184

dpdata.plugins.rdkit  
  module, 186

dpdata.plugins.siesta  
  module, 188

dpdata.plugins.vasp  
  module, 191

dpdata.plugins.xyz  
  module, 196

dpdata.psi4  
  module, 199

dpdata.psi4.input  
  module, 199

dpdata.psi4.output  
  module, 199

dpdata.pwmat  
  module, 200

dpdata.pwmat.atomconfig  
  module, 200

dpdata.pwmat.movement  
  module, 200

dpdata.pymatgen  
  module, 200

dpdata.pymatgen.molecule  
  module, 200

dpdata.pymatgen.structure  
  module, 200

dpdata.qe  
  module, 200

dpdata.qe.scf  
  module, 200

dpdata.qe.traj  
  module, 201

dpdata.rdkit  
  module, 201

dpdata.rdkit.sanitize  
  module, 201

dpdata.rdkit.utils  
  module, 203

dpdata.siesta  
  module, 203

dpdata.siesta.aiMD\_output  
  module, 203

dpdata.siesta.output  
  module, 204

dpdata.stat  
  module, 228

dpdata.system  
  module, 232

dpdata.unit  
  module, 289

dpdata.utils  
  module, 290

dpdata.vasp  
  module, 204

dpdata.vasp.outcar  
     module, 204  
 dpdata.vasp.poscar  
     module, 204  
 dpdata.vasp.xml  
     module, 204  
 dpdata.xyz  
     module, 205  
 dpdata.xyz.quip\_gap\_xyz  
     module, 205  
 dpdata.xyz.xyz  
     module, 205  
 dpdata\_cli() (in module dpdata.cli), 215  
 dpdata\_parser() (in module dpdata.cli), 215  
 DPDriver (class in dpdata.plugins.deepmd), 151  
 Driver (class in dpdata.driver), 218  
 DTYPES (dpdata.bond\_order\_system.BondOrderSystem  
     attribute), 214  
 DTYPES (dpdata.BondOrderSystem attribute), 65  
 DTYPES (dpdata.LabeledSystem attribute), 74  
 DTYPES (dpdata.System attribute), 109  
 DTYPES (dpdata.system.LabeledSystem attribute), 241  
 DTYPES (dpdata.system.System attribute), 275  
 dump() (dpdata.System method), 110  
 dump() (dpdata.system.System method), 276  
 dump() (in module dpdata.deepmd.comp), 126  
 dump() (in module dpdata.deepmd.hdf5), 126  
 dump() (in module dpdata.deepmd.mixed), 127  
 dump() (in module dpdata.deepmd.raw), 128  
 dumpbox2box() (in module dpdata.lammps.dump), 131

## E

e\_errors (dpdata.stat.Errors property), 229  
 e\_errors (dpdata.stat.ErrorsBase property), 230  
 e\_errors (dpdata.stat.MultiErrors property), 231  
 e\_mae (dpdata.stat.ErrorsBase property), 230  
 e\_rmse (dpdata.stat.ErrorsBase property), 230  
 Element (class in dpdata.periodic\_table), 227  
 elements\_index\_map() (in module dpdata.utils), 290  
 EnergyConversion (class in dpdata.unit), 289  
 Errors (class in dpdata.stat), 228  
 ErrorsBase (class in dpdata.stat), 229  
 extend() (dpdata.System method), 110  
 extend() (dpdata.system.System method), 276  
 extract\_keyword() (in module dp-  
     data.siesta.aiMD\_output), 203  
 extract\_keyword() (in module dpdata.siesta.output),  
     204

## F

f\_errors (dpdata.stat.Errors property), 229  
 f\_errors (dpdata.stat.ErrorsBase property), 230  
 f\_errors (dpdata.stat.MultiErrors property), 231  
 f\_mae (dpdata.stat.ErrorsBase property), 230

f\_rmse (dpdata.stat.ErrorsBase property), 230  
 FhiMDFormat (class in dpdata.plugins.fhi\_aims), 160  
 FhiSCFFormat (class in dpdata.plugins.fhi\_aims), 161  
 file\_to\_system\_data() (in module dp-  
     data.gromacs.gro), 130  
 ForceConversion (class in dpdata.unit), 289  
 Format (class in dpdata.format), 221  
 Format.MultiModes (class in dpdata.format), 222  
 formula (dpdata.System property), 110  
 formula (dpdata.system.System property), 276  
 formula() (in module dpdata.deepmd.mixed), 127  
 formula\_hash (dpdata.System property), 110  
 formula\_hash (dpdata.system.System property), 276  
 formulate\_config() (in module dpdata.vasp.xml), 204  
 from\_3dmol() (dpdata.LabeledSystem method), 75  
 from\_3dmol() (dpdata.MultiSystems method), 91  
 from\_3dmol() (dpdata.System method), 110  
 from\_3dmol() (dpdata.system.LabeledSystem method),  
     241  
 from\_3dmol() (dpdata.system.MultiSystems method),  
     257  
 from\_3dmol() (dpdata.system.System method), 276  
 from\_abacus\_lcao\_md() (dpdata.LabeledSystem  
     method), 75  
 from\_abacus\_lcao\_md() (dpdata.MultiSystems  
     method), 91  
 from\_abacus\_lcao\_md() (dpdata.System method), 110  
 from\_abacus\_lcao\_md() (dp-  
     data.system.LabeledSystem method), 241  
 from\_abacus\_lcao\_md() (dpdata.system.MultiSystems  
     method), 257  
 from\_abacus\_lcao\_md() (dpdata.system.System  
     method), 276  
 from\_abacus\_lcao\_relax() (dpdata.LabeledSystem  
     method), 75  
 from\_abacus\_lcao\_relax() (dpdata.MultiSystems  
     method), 91  
 from\_abacus\_lcao\_relax() (dpdata.System method),  
     110  
 from\_abacus\_lcao\_relax() (dp-  
     data.system.LabeledSystem method), 241  
 from\_abacus\_lcao\_relax() (dp-  
     data.system.MultiSystems method), 257  
 from\_abacus\_lcao\_relax() (dpdata.system.System  
     method), 276  
 from\_abacus\_lcao\_scf() (dpdata.LabeledSystem  
     method), 75  
 from\_abacus\_lcao\_scf() (dpdata.MultiSystems  
     method), 91  
 from\_abacus\_lcao\_scf() (dpdata.System method),  
     110  
 from\_abacus\_lcao\_scf() (dp-  
     data.system.LabeledSystem method), 241  
 from\_abacus\_lcao\_scf() (dp-



- data.system.MultiSystems method*), 257
- `from_abacus_lcao_scf()` (*dpdata.system.System method*), 276
- `from_abacus_md()` (*dpdata.LabeledSystem method*), 75
- `from_abacus_md()` (*dpdata.MultiSystems method*), 91
- `from_abacus_md()` (*dpdata.System method*), 110
- `from_abacus_md()` (*dpdata.system.LabeledSystem method*), 241
- `from_abacus_md()` (*dpdata.system.MultiSystems method*), 257
- `from_abacus_md()` (*dpdata.system.System method*), 276
- `from_abacus_pw_md()` (*dpdata.LabeledSystem method*), 75
- `from_abacus_pw_md()` (*dpdata.MultiSystems method*), 91
- `from_abacus_pw_md()` (*dpdata.System method*), 110
- `from_abacus_pw_md()` (*dpdata.system.LabeledSystem method*), 241
- `from_abacus_pw_md()` (*dpdata.system.MultiSystems method*), 257
- `from_abacus_pw_md()` (*dpdata.system.System method*), 276
- `from_abacus_pw_relax()` (*dpdata.LabeledSystem method*), 75
- `from_abacus_pw_relax()` (*dpdata.MultiSystems method*), 91
- `from_abacus_pw_relax()` (*dpdata.System method*), 110
- `from_abacus_pw_relax()` (*dpdata.system.LabeledSystem method*), 241
- `from_abacus_pw_relax()` (*dpdata.system.MultiSystems method*), 257
- `from_abacus_pw_relax()` (*dpdata.system.System method*), 276
- `from_abacus_pw_scf()` (*dpdata.LabeledSystem method*), 75
- `from_abacus_pw_scf()` (*dpdata.MultiSystems method*), 91
- `from_abacus_pw_scf()` (*dpdata.System method*), 110
- `from_abacus_pw_scf()` (*dpdata.system.LabeledSystem method*), 241
- `from_abacus_pw_scf()` (*dpdata.system.MultiSystems method*), 257
- `from_abacus_pw_scf()` (*dpdata.system.System method*), 276
- `from_abacus_relax()` (*dpdata.LabeledSystem method*), 75
- `from_abacus_relax()` (*dpdata.MultiSystems method*), 91
- `from_abacus_relax()` (*dpdata.System method*), 110
- `from_abacus_relax()` (*dpdata.system.LabeledSystem method*), 242
- `from_abacus_relax()` (*dpdata.system.MultiSystems method*), 257
- `from_abacus_relax()` (*dpdata.system.System method*), 276
- `from_abacus_scf()` (*dpdata.LabeledSystem method*), 75
- `from_abacus_scf()` (*dpdata.MultiSystems method*), 91
- `from_abacus_scf()` (*dpdata.System method*), 110
- `from_abacus_scf()` (*dpdata.system.LabeledSystem method*), 242
- `from_abacus_scf()` (*dpdata.system.MultiSystems method*), 257
- `from_abacus_scf()` (*dpdata.system.System method*), 276
- `from_abacus_stru()` (*dpdata.LabeledSystem method*), 75
- `from_abacus_stru()` (*dpdata.MultiSystems method*), 91
- `from_abacus_stru()` (*dpdata.System method*), 110
- `from_abacus_stru()` (*dpdata.system.LabeledSystem method*), 242
- `from_abacus_stru()` (*dpdata.system.MultiSystems method*), 257
- `from_abacus_stru()` (*dpdata.system.System method*), 276
- `from_amber_md()` (*dpdata.LabeledSystem method*), 75
- `from_amber_md()` (*dpdata.MultiSystems method*), 91
- `from_amber_md()` (*dpdata.System method*), 111
- `from_amber_md()` (*dpdata.system.LabeledSystem method*), 242
- `from_amber_md()` (*dpdata.system.MultiSystems method*), 257
- `from_amber_md()` (*dpdata.system.System method*), 277
- `from_ase_structure()` (*dpdata.LabeledSystem method*), 75
- `from_ase_structure()` (*dpdata.MultiSystems method*), 91
- `from_ase_structure()` (*dpdata.System method*), 111
- `from_ase_structure()` (*dpdata.system.LabeledSystem method*), 242
- `from_ase_structure()` (*dpdata.system.MultiSystems method*), 257
- `from_ase_structure()` (*dpdata.system.System method*), 277
- `from_ase_traj()` (*dpdata.LabeledSystem method*), 75
- `from_ase_traj()` (*dpdata.MultiSystems method*), 91
- `from_ase_traj()` (*dpdata.System method*), 111
- `from_ase_traj()` (*dpdata.system.LabeledSystem method*), 242
- `from_ase_traj()` (*dpdata.system.MultiSystems method*), 257
- `from_ase_traj()` (*dpdata.system.System method*), 277
- `from_atomconfig()` (*dpdata.LabeledSystem method*), 75
- `from_atomconfig()` (*dpdata.MultiSystems method*), 91
- `from_atomconfig()` (*dpdata.System method*), 111

`from_atomconfig()` (*dpdata.system.LabeledSystem method*), 242  
`from_atomconfig()` (*dpdata.system.MultiSystems method*), 257  
`from_atomconfig()` (*dpdata.system.System method*), 277  
`from_bond_order_system()` (*dpdata.format.Format method*), 222  
`from_bond_order_system()` (*dpdata.plugins.rdkit.MolFormat method*), 186  
`from_bond_order_system()` (*dpdata.plugins.rdkit.SdfFormat method*), 187  
`from_contcar()` (*dpdata.LabeledSystem method*), 76  
`from_contcar()` (*dpdata.MultiSystems method*), 92  
`from_contcar()` (*dpdata.System method*), 111  
`from_contcar()` (*dpdata.system.LabeledSystem method*), 242  
`from_contcar()` (*dpdata.system.MultiSystems method*), 258  
`from_contcar()` (*dpdata.system.System method*), 277  
`from_cp2k_aimd_output()` (*dpdata.LabeledSystem method*), 76  
`from_cp2k_aimd_output()` (*dpdata.MultiSystems method*), 92  
`from_cp2k_aimd_output()` (*dpdata.System method*), 111  
`from_cp2k_aimd_output()` (*dpdata.system.LabeledSystem method*), 242  
`from_cp2k_aimd_output()` (*dpdata.system.MultiSystems method*), 258  
`from_cp2k_aimd_output()` (*dpdata.system.System method*), 277  
`from_cp2k_output()` (*dpdata.LabeledSystem method*), 76  
`from_cp2k_output()` (*dpdata.MultiSystems method*), 92  
`from_cp2k_output()` (*dpdata.System method*), 111  
`from_cp2k_output()` (*dpdata.system.LabeledSystem method*), 242  
`from_cp2k_output()` (*dpdata.system.MultiSystems method*), 258  
`from_cp2k_output()` (*dpdata.system.System method*), 277  
`from_deepmd()` (*dpdata.LabeledSystem method*), 76  
`from_deepmd()` (*dpdata.MultiSystems method*), 92  
`from_deepmd()` (*dpdata.System method*), 111  
`from_deepmd()` (*dpdata.system.LabeledSystem method*), 242  
`from_deepmd()` (*dpdata.system.MultiSystems method*), 258  
`from_deepmd()` (*dpdata.system.System method*), 277  
`from_deepmd_comp()` (*dpdata.LabeledSystem method*), 76  
`from_deepmd_comp()` (*dpdata.MultiSystems method*), 92  
`from_deepmd_comp()` (*dpdata.System method*), 111  
`from_deepmd_comp()` (*dpdata.system.LabeledSystem method*), 242  
`from_deepmd_comp()` (*dpdata.system.MultiSystems method*), 258  
`from_deepmd_comp()` (*dpdata.system.System method*), 277  
`from_deepmd_hdf5()` (*dpdata.LabeledSystem method*), 76  
`from_deepmd_hdf5()` (*dpdata.MultiSystems method*), 92  
`from_deepmd_hdf5()` (*dpdata.System method*), 111  
`from_deepmd_hdf5()` (*dpdata.system.LabeledSystem method*), 242  
`from_deepmd_hdf5()` (*dpdata.system.MultiSystems method*), 258  
`from_deepmd_hdf5()` (*dpdata.system.System method*), 277  
`from_deepmd_npy()` (*dpdata.LabeledSystem method*), 76  
`from_deepmd_npy()` (*dpdata.MultiSystems method*), 92  
`from_deepmd_npy()` (*dpdata.System method*), 111  
`from_deepmd_npy()` (*dpdata.system.LabeledSystem method*), 242  
`from_deepmd_npy()` (*dpdata.system.MultiSystems method*), 258  
`from_deepmd_npy()` (*dpdata.system.System method*), 277  
`from_deepmd_npy_mixed()` (*dpdata.LabeledSystem method*), 76  
`from_deepmd_npy_mixed()` (*dpdata.MultiSystems method*), 92  
`from_deepmd_npy_mixed()` (*dpdata.System method*), 111  
`from_deepmd_npy_mixed()` (*dpdata.system.LabeledSystem method*), 242  
`from_deepmd_npy_mixed()` (*dpdata.system.MultiSystems method*), 258  
`from_deepmd_npy_mixed()` (*dpdata.system.System method*), 277  
`from_deepmd_raw()` (*dpdata.LabeledSystem method*), 76  
`from_deepmd_raw()` (*dpdata.MultiSystems method*), 92  
`from_deepmd_raw()` (*dpdata.System method*), 111  
`from_deepmd_raw()` (*dpdata.system.LabeledSystem method*), 242  
`from_deepmd_raw()` (*dpdata.system.MultiSystems method*), 258  
`from_deepmd_raw()` (*dpdata.system.System method*), 277  
`from_dftbplus()` (*dpdata.LabeledSystem method*), 76  
`from_dftbplus()` (*dpdata.MultiSystems method*), 92  
`from_dftbplus()` (*dpdata.System method*), 111

`from_dftbplus()` (*dpdata.system.LabeledSystem method*), 242  
`from_dftbplus()` (*dpdata.system.MultiSystems method*), 258  
`from_dftbplus()` (*dpdata.system.System method*), 277  
`from_dir()` (*dpdata.MultiSystems class method*), 92  
`from_dir()` (*dpdata.system.MultiSystems class method*), 258  
`from_dump()` (*dpdata.LabeledSystem method*), 76  
`from_dump()` (*dpdata.MultiSystems method*), 92  
`from_dump()` (*dpdata.System method*), 111  
`from_dump()` (*dpdata.system.LabeledSystem method*), 242  
`from_dump()` (*dpdata.system.MultiSystems method*), 258  
`from_dump()` (*dpdata.system.System method*), 277  
`from_fhi_aims_md()` (*dpdata.LabeledSystem method*), 76  
`from_fhi_aims_md()` (*dpdata.MultiSystems method*), 92  
`from_fhi_aims_md()` (*dpdata.System method*), 111  
`from_fhi_aims_md()` (*dpdata.system.LabeledSystem method*), 242  
`from_fhi_aims_md()` (*dpdata.system.MultiSystems method*), 258  
`from_fhi_aims_md()` (*dpdata.system.System method*), 277  
`from_fhi_aims_output()` (*dpdata.LabeledSystem method*), 76  
`from_fhi_aims_output()` (*dpdata.MultiSystems method*), 92  
`from_fhi_aims_output()` (*dpdata.System method*), 111  
`from_fhi_aims_output()` (*dpdata.system.LabeledSystem method*), 243  
`from_fhi_aims_output()` (*dpdata.system.MultiSystems method*), 258  
`from_fhi_aims_output()` (*dpdata.system.System method*), 277  
`from_fhi_aims_scf()` (*dpdata.LabeledSystem method*), 76  
`from_fhi_aims_scf()` (*dpdata.MultiSystems method*), 92  
`from_fhi_aims_scf()` (*dpdata.System method*), 111  
`from_fhi_aims_scf()` (*dpdata.system.LabeledSystem method*), 243  
`from_fhi_aims_scf()` (*dpdata.system.MultiSystems method*), 258  
`from_fhi_aims_scf()` (*dpdata.system.System method*), 277  
`from_file()` (*dpdata.MultiSystems class method*), 92  
`from_file()` (*dpdata.system.MultiSystems class method*), 258  
`from_finalconfig()` (*dpdata.LabeledSystem method*), 76  
`from_finalconfig()` (*dpdata.MultiSystems method*), 92  
`from_finalconfig()` (*dpdata.System method*), 111  
`from_finalconfig()` (*dpdata.system.LabeledSystem method*), 243  
`from_finalconfig()` (*dpdata.system.MultiSystems method*), 258  
`from_finalconfig()` (*dpdata.system.System method*), 277  
`from_fmt()` (*dpdata.System method*), 112  
`from_fmt()` (*dpdata.system.System method*), 278  
`from_fmt_obj()` (*dpdata.bond\_order\_system.BondOrderSystem method*), 214  
`from_fmt_obj()` (*dpdata.BondOrderSystem method*), 65  
`from_fmt_obj()` (*dpdata.LabeledSystem method*), 76  
`from_fmt_obj()` (*dpdata.MultiSystems method*), 92  
`from_fmt_obj()` (*dpdata.System method*), 112  
`from_fmt_obj()` (*dpdata.system.LabeledSystem method*), 243  
`from_fmt_obj()` (*dpdata.system.MultiSystems method*), 258  
`from_fmt_obj()` (*dpdata.system.System method*), 278  
`from_gaussian_gjf()` (*dpdata.LabeledSystem method*), 76  
`from_gaussian_gjf()` (*dpdata.MultiSystems method*), 92  
`from_gaussian_gjf()` (*dpdata.System method*), 112  
`from_gaussian_gjf()` (*dpdata.system.LabeledSystem method*), 243  
`from_gaussian_gjf()` (*dpdata.system.MultiSystems method*), 258  
`from_gaussian_gjf()` (*dpdata.system.System method*), 278  
`from_gaussian_log()` (*dpdata.LabeledSystem method*), 76  
`from_gaussian_log()` (*dpdata.MultiSystems method*), 92  
`from_gaussian_log()` (*dpdata.System method*), 112  
`from_gaussian_log()` (*dpdata.system.LabeledSystem method*), 243  
`from_gaussian_log()` (*dpdata.system.MultiSystems method*), 258  
`from_gaussian_log()` (*dpdata.system.System method*), 278  
`from_gaussian_md()` (*dpdata.LabeledSystem method*), 76  
`from_gaussian_md()` (*dpdata.MultiSystems method*), 92  
`from_gaussian_md()` (*dpdata.System method*), 112  
`from_gaussian_md()` (*dpdata.system.LabeledSystem method*), 243  
`from_gaussian_md()` (*dpdata.system.MultiSystems method*), 258

[from\\_gaussian\\_md\(\)](#) (*dpdata.system.System* method), 278  
[from\\_gro\(\)](#) (*dpdata.LabeledSystem* method), 76  
[from\\_gro\(\)](#) (*dpdata.MultiSystems* method), 93  
[from\\_gro\(\)](#) (*dpdata.System* method), 112  
[from\\_gro\(\)](#) (*dpdata.system.LabeledSystem* method), 243  
[from\\_gro\(\)](#) (*dpdata.system.MultiSystems* method), 259  
[from\\_gro\(\)](#) (*dpdata.system.System* method), 278  
[from\\_gromacs\\_gro\(\)](#) (*dpdata.LabeledSystem* method), 77  
[from\\_gromacs\\_gro\(\)](#) (*dpdata.MultiSystems* method), 93  
[from\\_gromacs\\_gro\(\)](#) (*dpdata.System* method), 112  
[from\\_gromacs\\_gro\(\)](#) (*dpdata.system.LabeledSystem* method), 243  
[from\\_gromacs\\_gro\(\)](#) (*dpdata.system.MultiSystems* method), 259  
[from\\_gromacs\\_gro\(\)](#) (*dpdata.system.System* method), 278  
[from\\_labeled\\_system\(\)](#) (*dpdata.format.Format* method), 223  
[from\\_labeled\\_system\(\)](#) (*dpdata.plugins.abacus.AbacusMDFormat* method), 135  
[from\\_labeled\\_system\(\)](#) (*dpdata.plugins.abacus.AbacusRelaxFormat* method), 135  
[from\\_labeled\\_system\(\)](#) (*dpdata.plugins.abacus.AbacusSCFFormat* method), 136  
[from\\_labeled\\_system\(\)](#) (*dpdata.plugins.amber.AmberMDFormat* method), 139  
[from\\_labeled\\_system\(\)](#) (*dpdata.plugins.amber.SQMOutFormat* method), 143  
[from\\_labeled\\_system\(\)](#) (*dpdata.plugins.ase.ASEStructureFormat* method), 145  
[from\\_labeled\\_system\(\)](#) (*dpdata.plugins.ase.ASETrajFormat* method), 147  
[from\\_labeled\\_system\(\)](#) (*dpdata.plugins.cp2k.CP2KAIMDOutputFormat* method), 149  
[from\\_labeled\\_system\(\)](#) (*dpdata.plugins.cp2k.CP2KOutputFormat* method), 150  
[from\\_labeled\\_system\(\)](#) (*dpdata.plugins.deepmd.DeePMDCompFormat* method), 152  
[from\\_labeled\\_system\(\)](#) (*dpdata.plugins.deepmd.DeePMDHDF5Format* method), 154  
[from\\_labeled\\_system\(\)](#) (*dpdata.plugins.deepmd.DeePMDRawFormat* method), 159  
[from\\_labeled\\_system\(\)](#) (*dpdata.plugins.dftbplus.DFTBplusFormat* method), 160  
[from\\_labeled\\_system\(\)](#) (*dpdata.plugins.fhi\_aims.FhiMDFormat* method), 161  
[from\\_labeled\\_system\(\)](#) (*dpdata.plugins.fhi\_aims.FhiSCFFormat* method), 162  
[from\\_labeled\\_system\(\)](#) (*dpdata.plugins.gaussian.GaussianLogFormat* method), 165  
[from\\_labeled\\_system\(\)](#) (*dpdata.plugins.gaussian.GaussianMDFormat* method), 166  
[from\\_labeled\\_system\(\)](#) (*dpdata.plugins.n2p2.N2P2Format* method), 172  
[from\\_labeled\\_system\(\)](#) (*dpdata.plugins.openmx.OPENMXFormat* method), 173  
[from\\_labeled\\_system\(\)](#) (*dpdata.plugins.orca.ORCASPOutFormat* method), 175  
[from\\_labeled\\_system\(\)](#) (*dpdata.plugins.psi4.PSI4OutFormat* method), 177  
[from\\_labeled\\_system\(\)](#) (*dpdata.plugins.pwmat.PwmatOutputFormat* method), 180  
[from\\_labeled\\_system\(\)](#) (*dpdata.plugins.qe.QECPWSCFFormat* method), 184  
[from\\_labeled\\_system\(\)](#) (*dpdata.plugins.qe.QECPTrajFormat* method), 185  
[from\\_labeled\\_system\(\)](#) (*dpdata.plugins.siesta.SiestaAIMDOutputFormat* method), 189  
[from\\_labeled\\_system\(\)](#) (*dpdata.plugins.siesta.SiestaOutputFormat* method), 190  
[from\\_labeled\\_system\(\)](#) (*dpdata.plugins.vasp.VASPOutcarFormat* method), 191  
[from\\_labeled\\_system\(\)](#) (*dpdata.plugins.vasp.VASPXMLFormat* method), 195  
[from\\_labeled\\_system\(\)](#) (*dpdata.plugins.xyz.QuipGapXYZFormat* method),



- 196
- `from_labeled_system_mix()` (*dpdata.plugins.deepmd.DeePDMixedFormat method*), 157
- `from_lammps_dump()` (*dpdata.LabeledSystem method*), 77
- `from_lammps_dump()` (*dpdata.MultiSystems method*), 93
- `from_lammps_dump()` (*dpdata.System method*), 112
- `from_lammps_dump()` (*dpdata.system.LabeledSystem method*), 243
- `from_lammps_dump()` (*dpdata.system.MultiSystems method*), 259
- `from_lammps_dump()` (*dpdata.system.System method*), 278
- `from_lammps_lmp()` (*dpdata.LabeledSystem method*), 77
- `from_lammps_lmp()` (*dpdata.MultiSystems method*), 93
- `from_lammps_lmp()` (*dpdata.System method*), 112
- `from_lammps_lmp()` (*dpdata.system.LabeledSystem method*), 243
- `from_lammps_lmp()` (*dpdata.system.MultiSystems method*), 259
- `from_lammps_lmp()` (*dpdata.system.System method*), 278
- `from_list()` (*dpdata.LabeledSystem method*), 77
- `from_list()` (*dpdata.MultiSystems method*), 93
- `from_list()` (*dpdata.System method*), 112
- `from_list()` (*dpdata.system.LabeledSystem method*), 243
- `from_list()` (*dpdata.system.MultiSystems method*), 259
- `from_list()` (*dpdata.system.System method*), 278
- `from_lmp()` (*dpdata.LabeledSystem method*), 77
- `from_lmp()` (*dpdata.MultiSystems method*), 93
- `from_lmp()` (*dpdata.System method*), 112
- `from_lmp()` (*dpdata.system.LabeledSystem method*), 243
- `from_lmp()` (*dpdata.system.MultiSystems method*), 259
- `from_lmp()` (*dpdata.system.System method*), 278
- `from_mlmd()` (*dpdata.LabeledSystem method*), 77
- `from_mlmd()` (*dpdata.MultiSystems method*), 93
- `from_mlmd()` (*dpdata.System method*), 112
- `from_mlmd()` (*dpdata.system.LabeledSystem method*), 243
- `from_mlmd()` (*dpdata.system.MultiSystems method*), 259
- `from_mlmd()` (*dpdata.system.System method*), 278
- `from_mol()` (*dpdata.LabeledSystem method*), 77
- `from_mol()` (*dpdata.MultiSystems method*), 93
- `from_mol()` (*dpdata.System method*), 112
- `from_mol()` (*dpdata.system.LabeledSystem method*), 243
- `from_mol()` (*dpdata.system.MultiSystems method*), 259
- `from_mol()` (*dpdata.system.System method*), 278
- `from_mol_file()` (*dpdata.LabeledSystem method*), 77
- `from_mol_file()` (*dpdata.MultiSystems method*), 93
- `from_mol_file()` (*dpdata.System method*), 112
- `from_mol_file()` (*dpdata.system.LabeledSystem method*), 243
- `from_mol_file()` (*dpdata.system.MultiSystems method*), 259
- `from_mol_file()` (*dpdata.system.System method*), 278
- `from_movement()` (*dpdata.LabeledSystem method*), 77
- `from_movement()` (*dpdata.MultiSystems method*), 93
- `from_movement()` (*dpdata.System method*), 112
- `from_movement()` (*dpdata.system.LabeledSystem method*), 243
- `from_movement()` (*dpdata.system.MultiSystems method*), 259
- `from_movement()` (*dpdata.system.System method*), 278
- `from_multi_systems()` (*dpdata.format.Format method*), 223
- `from_multi_systems()` (*dpdata.plugins.ase.ASEStructureFormat method*), 146
- `from_multi_systems()` (*dpdata.plugins.deepmd.DeePMDHDF5Format method*), 154
- `from_multi_systems()` (*dpdata.plugins.deepmd.DeePDMixedFormat method*), 157
- `from_multi_systems()` (*dpdata.plugins.xyz.QuipGapXYZFormat method*), 196
- `from_n2p2()` (*dpdata.LabeledSystem method*), 77
- `from_n2p2()` (*dpdata.MultiSystems method*), 93
- `from_n2p2()` (*dpdata.System method*), 112
- `from_n2p2()` (*dpdata.system.LabeledSystem method*), 243
- `from_n2p2()` (*dpdata.system.MultiSystems method*), 259
- `from_n2p2()` (*dpdata.system.System method*), 278
- `from_openmx_md()` (*dpdata.LabeledSystem method*), 77
- `from_openmx_md()` (*dpdata.MultiSystems method*), 93
- `from_openmx_md()` (*dpdata.System method*), 112
- `from_openmx_md()` (*dpdata.system.LabeledSystem method*), 243
- `from_openmx_md()` (*dpdata.system.MultiSystems method*), 259
- `from_openmx_md()` (*dpdata.system.System method*), 278
- `from_orca_spout()` (*dpdata.LabeledSystem method*), 77
- `from_orca_spout()` (*dpdata.MultiSystems method*), 93
- `from_orca_spout()` (*dpdata.System method*), 112
- `from_orca_spout()` (*dpdata.system.LabeledSystem method*), 243
- `from_orca_spout()` (*dpdata.system.MultiSystems method*), 259
- `from_orca_spout()` (*dpdata.system.System method*), 278

from\_outcar() (*dpdata.LabeledSystem* method), 77  
 from\_outcar() (*dpdata.MultiSystems* method), 93  
 from\_outcar() (*dpdata.System* method), 112  
 from\_outcar() (*dpdata.system.LabeledSystem* method), 244  
 from\_outcar() (*dpdata.system.MultiSystems* method), 259  
 from\_outcar() (*dpdata.system.System* method), 278  
 from\_poscar() (*dpdata.LabeledSystem* method), 77  
 from\_poscar() (*dpdata.MultiSystems* method), 93  
 from\_poscar() (*dpdata.System* method), 112  
 from\_poscar() (*dpdata.system.LabeledSystem* method), 244  
 from\_poscar() (*dpdata.system.MultiSystems* method), 259  
 from\_poscar() (*dpdata.system.System* method), 278  
 from\_psi4\_inp() (*dpdata.LabeledSystem* method), 77  
 from\_psi4\_inp() (*dpdata.MultiSystems* method), 93  
 from\_psi4\_inp() (*dpdata.System* method), 113  
 from\_psi4\_inp() (*dpdata.system.LabeledSystem* method), 244  
 from\_psi4\_inp() (*dpdata.system.MultiSystems* method), 259  
 from\_psi4\_inp() (*dpdata.system.System* method), 279  
 from\_psi4\_out() (*dpdata.LabeledSystem* method), 77  
 from\_psi4\_out() (*dpdata.MultiSystems* method), 93  
 from\_psi4\_out() (*dpdata.System* method), 113  
 from\_psi4\_out() (*dpdata.system.LabeledSystem* method), 244  
 from\_psi4\_out() (*dpdata.system.MultiSystems* method), 259  
 from\_psi4\_out() (*dpdata.system.System* method), 279  
 from\_pwmat\_atomconfig() (*dpdata.LabeledSystem* method), 77  
 from\_pwmat\_atomconfig() (*dpdata.MultiSystems* method), 93  
 from\_pwmat\_atomconfig() (*dpdata.System* method), 113  
 from\_pwmat\_atomconfig() (*dpdata.system.LabeledSystem* method), 244  
 from\_pwmat\_atomconfig() (*dpdata.system.MultiSystems* method), 259  
 from\_pwmat\_atomconfig() (*dpdata.system.System* method), 279  
 from\_pwmat\_finalconfig() (*dpdata.LabeledSystem* method), 77  
 from\_pwmat\_finalconfig() (*dpdata.MultiSystems* method), 93  
 from\_pwmat\_finalconfig() (*dpdata.System* method), 113  
 from\_pwmat\_finalconfig() (*dpdata.system.LabeledSystem* method), 244  
 from\_pwmat\_finalconfig() (*dpdata.system.MultiSystems* method), 259  
 from\_pwmat\_finalconfig() (*dpdata.system.System* method), 279  
 from\_pwmat\_mlmd() (*dpdata.LabeledSystem* method), 77  
 from\_pwmat\_mlmd() (*dpdata.MultiSystems* method), 94  
 from\_pwmat\_mlmd() (*dpdata.System* method), 113  
 from\_pwmat\_mlmd() (*dpdata.system.LabeledSystem* method), 244  
 from\_pwmat\_mlmd() (*dpdata.system.MultiSystems* method), 260  
 from\_pwmat\_mlmd() (*dpdata.system.System* method), 279  
 from\_pwmat\_movement() (*dpdata.LabeledSystem* method), 78  
 from\_pwmat\_movement() (*dpdata.MultiSystems* method), 94  
 from\_pwmat\_movement() (*dpdata.System* method), 113  
 from\_pwmat\_movement() (*dpdata.system.LabeledSystem* method), 244  
 from\_pwmat\_movement() (*dpdata.system.MultiSystems* method), 260  
 from\_pwmat\_movement() (*dpdata.system.System* method), 279  
 from\_pwmat\_output() (*dpdata.LabeledSystem* method), 78  
 from\_pwmat\_output() (*dpdata.MultiSystems* method), 94  
 from\_pwmat\_output() (*dpdata.System* method), 113  
 from\_pwmat\_output() (*dpdata.system.LabeledSystem* method), 244  
 from\_pwmat\_output() (*dpdata.system.MultiSystems* method), 260  
 from\_pwmat\_output() (*dpdata.system.System* method), 279  
 from\_pymatgen\_computedstructureentry() (*dpdata.LabeledSystem* method), 78  
 from\_pymatgen\_computedstructureentry() (*dpdata.MultiSystems* method), 94  
 from\_pymatgen\_computedstructureentry() (*dpdata.System* method), 113  
 from\_pymatgen\_computedstructureentry() (*dpdata.system.LabeledSystem* method), 244  
 from\_pymatgen\_computedstructureentry() (*dpdata.system.MultiSystems* method), 260  
 from\_pymatgen\_computedstructureentry() (*dpdata.system.System* method), 279  
 from\_pymatgen\_molecule() (*dpdata.LabeledSystem* method), 78  
 from\_pymatgen\_molecule() (*dpdata.MultiSystems* method), 94  
 from\_pymatgen\_molecule() (*dpdata.System* method), 113  
 from\_pymatgen\_molecule() (*dpdata.system.LabeledSystem* method), 244

[from\\_pymatgen\\_molecule\(\)](#) (*dpdata.system.MultiSystems method*), 260  
[from\\_pymatgen\\_molecule\(\)](#) (*dpdata.system.System method*), 279  
[from\\_pymatgen\\_structure\(\)](#) (*dpdata.LabeledSystem method*), 78  
[from\\_pymatgen\\_structure\(\)](#) (*dpdata.MultiSystems method*), 94  
[from\\_pymatgen\\_structure\(\)](#) (*dpdata.System method*), 113  
[from\\_pymatgen\\_structure\(\)](#) (*dpdata.system.LabeledSystem method*), 244  
[from\\_pymatgen\\_structure\(\)](#) (*dpdata.system.MultiSystems method*), 260  
[from\\_pymatgen\\_structure\(\)](#) (*dpdata.system.System method*), 279  
[from\\_qe\\_cp\\_traj\(\)](#) (*dpdata.LabeledSystem method*), 78  
[from\\_qe\\_cp\\_traj\(\)](#) (*dpdata.MultiSystems method*), 94  
[from\\_qe\\_cp\\_traj\(\)](#) (*dpdata.System method*), 113  
[from\\_qe\\_cp\\_traj\(\)](#) (*dpdata.system.LabeledSystem method*), 244  
[from\\_qe\\_cp\\_traj\(\)](#) (*dpdata.system.MultiSystems method*), 260  
[from\\_qe\\_cp\\_traj\(\)](#) (*dpdata.system.System method*), 279  
[from\\_qe\\_pw\\_scf\(\)](#) (*dpdata.LabeledSystem method*), 78  
[from\\_qe\\_pw\\_scf\(\)](#) (*dpdata.MultiSystems method*), 94  
[from\\_qe\\_pw\\_scf\(\)](#) (*dpdata.System method*), 113  
[from\\_qe\\_pw\\_scf\(\)](#) (*dpdata.system.LabeledSystem method*), 244  
[from\\_qe\\_pw\\_scf\(\)](#) (*dpdata.system.MultiSystems method*), 260  
[from\\_qe\\_pw\\_scf\(\)](#) (*dpdata.system.System method*), 279  
[from\\_quip\\_gap\\_xyz\(\)](#) (*dpdata.LabeledSystem method*), 78  
[from\\_quip\\_gap\\_xyz\(\)](#) (*dpdata.MultiSystems method*), 94  
[from\\_quip\\_gap\\_xyz\(\)](#) (*dpdata.System method*), 113  
[from\\_quip\\_gap\\_xyz\(\)](#) (*dpdata.system.LabeledSystem method*), 244  
[from\\_quip\\_gap\\_xyz\(\)](#) (*dpdata.system.MultiSystems method*), 260  
[from\\_quip\\_gap\\_xyz\(\)](#) (*dpdata.system.System method*), 279  
[from\\_quip\\_gap\\_xyz\\_file\(\)](#) (*dpdata.LabeledSystem method*), 78  
[from\\_quip\\_gap\\_xyz\\_file\(\)](#) (*dpdata.MultiSystems method*), 94  
[from\\_quip\\_gap\\_xyz\\_file\(\)](#) (*dpdata.System method*), 113  
[from\\_quip\\_gap\\_xyz\\_file\(\)](#) (*dpdata.system.LabeledSystem method*), 244  
[from\\_quip\\_gap\\_xyz\\_file\(\)](#) (*dpdata.system.MultiSystems method*), 260  
[from\\_quip\\_gap\\_xyz\\_file\(\)](#) (*dpdata.system.System method*), 279  
[from\\_rdkit\\_mol\(\)](#) (*dpdata.BondOrderSystem method*), 214  
[from\\_rdkit\\_mol\(\)](#) (*dpdata.MultiSystems method*), 260  
[from\\_rdkit\\_mol\(\)](#) (*dpdata.system.LabeledSystem method*), 244  
[from\\_rdkit\\_mol\(\)](#) (*dpdata.system.MultiSystems method*), 260  
[from\\_rdkit\\_mol\(\)](#) (*dpdata.system.System method*), 279  
[from\\_sdf\(\)](#) (*dpdata.LabeledSystem method*), 78  
[from\\_sdf\(\)](#) (*dpdata.MultiSystems method*), 94  
[from\\_sdf\(\)](#) (*dpdata.System method*), 113  
[from\\_sdf\(\)](#) (*dpdata.system.LabeledSystem method*), 244  
[from\\_sdf\(\)](#) (*dpdata.system.MultiSystems method*), 260  
[from\\_sdf\(\)](#) (*dpdata.system.System method*), 279  
[from\\_sdf\\_file\(\)](#) (*dpdata.LabeledSystem method*), 78  
[from\\_sdf\\_file\(\)](#) (*dpdata.MultiSystems method*), 94  
[from\\_sdf\\_file\(\)](#) (*dpdata.System method*), 113  
[from\\_sdf\\_file\(\)](#) (*dpdata.system.LabeledSystem method*), 244  
[from\\_sdf\\_file\(\)](#) (*dpdata.system.MultiSystems method*), 260  
[from\\_sdf\\_file\(\)](#) (*dpdata.system.System method*), 279  
[from\\_siesta\\_aimd\\_output\(\)](#) (*dpdata.LabeledSystem method*), 78  
[from\\_siesta\\_aimd\\_output\(\)](#) (*dpdata.LabeledSystem method*), 78  
[from\\_siesta\\_aimd\\_output\(\)](#) (*dpdata.MultiSystems method*), 94  
[from\\_siesta\\_aimd\\_output\(\)](#) (*dpdata.MultiSystems method*), 94  
[from\\_siesta\\_aimd\\_output\(\)](#) (*dpdata.System method*), 113  
[from\\_siesta\\_aimd\\_output\(\)](#) (*dpdata.System method*), 113  
[from\\_siesta\\_aimd\\_output\(\)](#) (*dpdata.system.LabeledSystem method*), 244  
[from\\_siesta\\_aimd\\_output\(\)](#) (*dpdata.system.LabeledSystem method*), 245  
[from\\_siesta\\_aimd\\_output\(\)](#) (*dpdata.system.MultiSystems method*), 260  
[from\\_siesta\\_aimd\\_output\(\)](#) (*dpdata.system.MultiSystems method*), 260  
[from\\_siesta\\_aimd\\_output\(\)](#) (*dpdata.system.System method*), 279  
[from\\_siesta\\_aimd\\_output\(\)](#) (*dpdata.system.System method*), 279  
[from\\_siesta\\_output\(\)](#) (*dpdata.LabeledSystem method*), 78  
[from\\_siesta\\_output\(\)](#) (*dpdata.MultiSystems method*), 94  
[from\\_siesta\\_output\(\)](#) (*dpdata.System method*), 113  
[from\\_siesta\\_output\(\)](#) (*dpdata.system.LabeledSystem method*), 245

`from_siesta_output()` (*dpdata.system.MultiSystems* method), 260  
`from_siesta_output()` (*dpdata.system.System* method), 279  
`from_sqm_in()` (*dpdata.LabeledSystem* method), 78  
`from_sqm_in()` (*dpdata.MultiSystems* method), 94  
`from_sqm_in()` (*dpdata.System* method), 114  
`from_sqm_in()` (*dpdata.system.LabeledSystem* method), 245  
`from_sqm_in()` (*dpdata.system.MultiSystems* method), 260  
`from_sqm_in()` (*dpdata.system.System* method), 280  
`from_sqm_out()` (*dpdata.LabeledSystem* method), 78  
`from_sqm_out()` (*dpdata.MultiSystems* method), 94  
`from_sqm_out()` (*dpdata.System* method), 114  
`from_sqm_out()` (*dpdata.system.LabeledSystem* method), 245  
`from_sqm_out()` (*dpdata.system.MultiSystems* method), 260  
`from_sqm_out()` (*dpdata.system.System* method), 280  
`from_stru()` (*dpdata.LabeledSystem* method), 78  
`from_stru()` (*dpdata.MultiSystems* method), 94  
`from_stru()` (*dpdata.System* method), 114  
`from_stru()` (*dpdata.system.LabeledSystem* method), 245  
`from_stru()` (*dpdata.system.MultiSystems* method), 260  
`from_stru()` (*dpdata.system.System* method), 280  
`from_system()` (*dpdata.format.Format* method), 223  
`from_system()` (*dpdata.plugins.abacus.AbacusSTRUFormat* method), 137  
`from_system()` (*dpdata.plugins.amber.AmberMDFormat* method), 139  
`from_system()` (*dpdata.plugins.amber.SQMOutFormat* method), 143  
`from_system()` (*dpdata.plugins.ase.ASEStructureFormat* method), 146  
`from_system()` (*dpdata.plugins.ase.ASETrajFormat* method), 148  
`from_system()` (*dpdata.plugins.deepmd.DeePMDCompFormat* method), 152  
`from_system()` (*dpdata.plugins.deepmd.DeePMDHDF5Format* method), 155  
`from_system()` (*dpdata.plugins.deepmd.DeePMDRawFormat* method), 159  
`from_system()` (*dpdata.plugins.gaussian.GaussianGJFFormat* method), 163  
`from_system()` (*dpdata.plugins.gromacs.GromacsGroFormat* method), 167  
`from_system()` (*dpdata.plugins.lammps.LAMMPSDumpFormat* method), 168  
`from_system()` (*dpdata.plugins.lammps.LAMMPSLmpFormat* method), 169  
`from_system()` (*dpdata.plugins.openmx.OPENMXFormat* method), 174  
`from_system()` (*dpdata.plugins.pwmat.PwmatAtomconfigFormat* method), 178  
`from_system()` (*dpdata.plugins.pymatgen.PyMatgenMoleculeFormat* method), 182  
`from_system()` (*dpdata.plugins.pymatgen.PyMatgenStructureFormat* method), 183  
`from_system()` (*dpdata.plugins.qe.QECPTrajFormat* method), 185  
`from_system()` (*dpdata.plugins.siesta.SiestaAIMDOutputFormat* method), 189  
`from_system()` (*dpdata.plugins.siesta.SiestaOutputFormat* method), 190  
`from_system()` (*dpdata.plugins.vasp.VASPPoscarFormat* method), 192  
`from_system()` (*dpdata.plugins.xyz.XYZFormat* method), 198  
`from_system_data()` (in module *dpdata.gromacs.gro*), 130  
`from_system_data()` (in module *dpdata.lammps.lmp*), 131  
`from_system_data()` (in module *dpdata.pwmat.atomconfig*), 200  
`from_system_data()` (in module *dpdata.pymatgen.structure*), 200  
`from_system_data()` (in module *dpdata.vasp.poscar*), 204  
`from_system_mix()` (*dpdata.plugins.deepmd.DeePMDMixedFormat* method), 157  
`from_vasp_contcar()` (*dpdata.LabeledSystem* method), 78  
`from_vasp_contcar()` (*dpdata.MultiSystems* method), 94  
`from_vasp_contcar()` (*dpdata.System* method), 114  
`from_vasp_contcar()` (*dpdata.system.LabeledSystem* method), 245  
`from_vasp_contcar()` (*dpdata.system.MultiSystems* method), 260  
`from_vasp_contcar()` (*dpdata.system.System* method), 280  
`from_vasp_outcar()` (*dpdata.LabeledSystem* method), 78  
`from_vasp_outcar()` (*dpdata.MultiSystems* method), 95  
`from_vasp_outcar()` (*dpdata.System* method), 114  
`from_vasp_outcar()` (*dpdata.system.LabeledSystem* method), 245  
`from_vasp_outcar()` (*dpdata.system.MultiSystems* method), 261  
`from_vasp_outcar()` (*dpdata.system.System* method), 280  
`from_vasp_poscar()` (*dpdata.LabeledSystem* method), 79  
`from_vasp_poscar()` (*dpdata.MultiSystems* method),



- 95
- `from_vasp_poscar()` (*dpdata.System* method), 114
- `from_vasp_poscar()` (*dpdata.system.LabeledSystem* method), 245
- `from_vasp_poscar()` (*dpdata.system.MultiSystems* method), 261
- `from_vasp_poscar()` (*dpdata.system.System* method), 280
- `from_vasp_string()` (*dpdata.LabeledSystem* method), 79
- `from_vasp_string()` (*dpdata.MultiSystems* method), 95
- `from_vasp_string()` (*dpdata.System* method), 114
- `from_vasp_string()` (*dpdata.system.LabeledSystem* method), 245
- `from_vasp_string()` (*dpdata.system.MultiSystems* method), 261
- `from_vasp_string()` (*dpdata.system.System* method), 280
- `from_vasp_xml()` (*dpdata.LabeledSystem* method), 79
- `from_vasp_xml()` (*dpdata.MultiSystems* method), 95
- `from_vasp_xml()` (*dpdata.System* method), 114
- `from_vasp_xml()` (*dpdata.system.LabeledSystem* method), 245
- `from_vasp_xml()` (*dpdata.system.MultiSystems* method), 261
- `from_vasp_xml()` (*dpdata.system.System* method), 280
- `from_xml()` (*dpdata.LabeledSystem* method), 79
- `from_xml()` (*dpdata.MultiSystems* method), 95
- `from_xml()` (*dpdata.System* method), 114
- `from_xml()` (*dpdata.system.LabeledSystem* method), 245
- `from_xml()` (*dpdata.system.MultiSystems* method), 261
- `from_xml()` (*dpdata.system.System* method), 280
- `from_xyz()` (*dpdata.LabeledSystem* method), 79
- `from_xyz()` (*dpdata.MultiSystems* method), 95
- `from_xyz()` (*dpdata.System* method), 114
- `from_xyz()` (*dpdata.system.LabeledSystem* method), 245
- `from_xyz()` (*dpdata.system.MultiSystems* method), 261
- `from_xyz()` (*dpdata.system.System* method), 280
- `from_Z()` (*dpdata.periodic\_table.Element* class method), 227
- ## G
- `GaussianGJFFormat` (class in *dpdata.plugins.gaussian*), 163
- `GaussianDriver` (class in *dpdata.plugins.gaussian*), 164
- `GaussianLogFormat` (class in *dpdata.plugins.gaussian*), 165
- `GaussianMDFormat` (class in *dpdata.plugins.gaussian*), 165
- `get_aiMD_frame()` (in module *dpdata.siesta.aiMD\_output*), 203
- `get_atom_name()` (in module *dpdata.siesta.aiMD\_output*), 203
- `get_atom_name()` (in module *dpdata.siesta.output*), 204
- `get_atom_names()` (*dpdata.System* method), 114
- `get_atom_names()` (*dpdata.system.System* method), 280
- `get_atom_nums()` (*dpdata.System* method), 114
- `get_atom_nums()` (*dpdata.system.System* method), 280
- `get_atom_nums()` (in module *dpdata.siesta.aiMD\_output*), 203
- `get_atom_nums()` (in module *dpdata.siesta.output*), 204
- `get_atom_perturb_vector()` (in module *dpdata.system*), 289
- `get_atom_types()` (*dpdata.System* method), 114
- `get_atom_types()` (*dpdata.system.System* method), 280
- `get_atom_types()` (in module *dpdata.siesta.aiMD\_output*), 203
- `get_atom_types()` (in module *dpdata.siesta.output*), 204
- `get_atoms()` (in module *dpdata.lammps.lmp*), 131
- `get_atype()` (in module *dpdata.lammps.dump*), 131
- `get_atype()` (in module *dpdata.lammps.lmp*), 131
- `get_block()` (in module *dpdata.abacus.scf*), 123
- `get_block()` (in module *dpdata.qe.scf*), 200
- `get_block_generator()` (*dpdata.xyz.quip\_gap\_xyz.QuipGapxyzSystems* method), 205
- `get_bond_order()` (*dpdata.bond\_order\_system.BondOrderSystem* method), 214
- `get_bond_order()` (*dpdata.BondOrderSystem* method), 65
- `get_cell()` (in module *dpdata.abacus.scf*), 123
- `get_cell()` (in module *dpdata.qe.scf*), 200
- `get_cell_perturb_matrix()` (in module *dpdata.system*), 289
- `get_charge()` (*dpdata.bond\_order\_system.BondOrderSystem* method), 214
- `get_charge()` (*dpdata.BondOrderSystem* method), 65
- `get_cls_name()` (in module *dpdata.system*), 289
- `get_coord_dump_freq()` (in module *dpdata.abacus.md*), 123
- `get_coords()` (in module *dpdata.abacus.scf*), 123
- `get_coords()` (in module *dpdata.qe.scf*), 200
- `get_coords_from_dump()` (in module *dpdata.abacus.md*), 123
- `get_coords_from_log()` (in module *dpdata.abacus.relax*), 123
- `get_coordtype_and_scalefactor()` (in module *dpdata.lammps.dump*), 131
- `get_data_types()` (in module *dpdata.data\_type*), 217
- `get_driver()` (*dpdata.driver.Driver* static method), 218
- `get_drivers()` (*dpdata.driver.Driver* static method), 218

`get_dumpbox()` (in module `dpdata.lammps.dump`), 131  
`get_energy()` (in module `dpdata.abacus.md`), 123  
`get_energy()` (in module `dpdata.abacus.scf`), 123  
`get_energy()` (in module `dpdata.qe.scf`), 201  
`get_explicit_valence()` (in module `dpdata.rdkit.sanitize`), 202  
`get_fhi_aims_block()` (in module `dpdata.fhi_aims.output`), 128  
`get_force()` (in module `dpdata.abacus.scf`), 123  
`get_force()` (in module `dpdata.qe.scf`), 201  
`get_formal_charges()` (`dpdata.bond_order_system.BondOrderSystem` method), 214  
`get_formal_charges()` (`dpdata.BondOrderSystem` method), 65  
`get_formats()` (`dpdata.format.Format` static method), 223  
`get_frame()` (in module `dpdata.abacus.md`), 123  
`get_frame()` (in module `dpdata.abacus.relax`), 123  
`get_frame()` (in module `dpdata.abacus.scf`), 123  
`get_frame()` (in module `dpdata.qe.scf`), 201  
`get_frame_from_stru()` (in module `dpdata.abacus.scf`), 123  
`get_frames()` (in module `dpdata.cp2k.output`), 126  
`get_frames()` (in module `dpdata.fhi_aims.output`), 128  
`get_frames()` (in module `dpdata.pwmat.movement`), 200  
`get_frames()` (in module `dpdata.vasp.outcar`), 204  
`get_from_methods()` (`dpdata.format.Format` static method), 223  
`get_geometry_in()` (in module `dpdata.abacus.scf`), 123  
`get_info()` (in module `dpdata.fhi_aims.output`), 128  
`get_lmpbox()` (in module `dpdata.lammps.lmp`), 131  
`get_log_block_generator()` (`dpdata.cp2k.output.Cp2kSystems` method), 126  
`get_log_file()` (in module `dpdata.abacus.relax`), 123  
`get_minimizer()` (`dpdata.driver.Minimizer` static method), 220  
`get_minimizers()` (`dpdata.driver.Minimizer` static method), 220  
`get_mol()` (`dpdata.bond_order_system.BondOrderSystem` method), 214  
`get_mol()` (`dpdata.BondOrderSystem` method), 65  
`get_movement_block()` (in module `dpdata.pwmat.movement`), 200  
`get_natoms()` (`dpdata.System` method), 114  
`get_natoms()` (`dpdata.system.System` method), 280  
`get_natoms()` (in module `dpdata.lammps.dump`), 131  
`get_natoms()` (in module `dpdata.lammps.lmp`), 131  
`get_natoms_vec()` (in module `dpdata.lammps.dump`), 131  
`get_natoms_vec()` (in module `dpdata.lammps.lmp`), 131  
`get_natomtypes()` (in module `dpdata.lammps.dump`), 131  
`get_natomtypes()` (in module `dpdata.lammps.lmp`), 131  
`get_nbonds()` (`dpdata.bond_order_system.BondOrderSystem` method), 214  
`get_nbonds()` (`dpdata.BondOrderSystem` method), 65  
`get_nele_from_stru()` (in module `dpdata.abacus.scf`), 123  
`get_nframes()` (`dpdata.MultiSystems` method), 95  
`get_nframes()` (`dpdata.System` method), 114  
`get_nframes()` (`dpdata.system.MultiSystems` method), 261  
`get_nframes()` (`dpdata.system.System` method), 280  
`get_ntypes()` (`dpdata.System` method), 114  
`get_ntypes()` (`dpdata.system.System` method), 280  
`get_outcar_block()` (in module `dpdata.vasp.outcar`), 204  
`get_path_out()` (in module `dpdata.abacus.md`), 123  
`get_path_out()` (in module `dpdata.abacus.scf`), 124  
`get_plugin()` (`dpdata.plugin.Plugin` method), 228  
`get_posi()` (in module `dpdata.lammps.lmp`), 131  
`get_single_line_tail()` (in module `dpdata.siesta.aiMD_output`), 203  
`get_single_line_tail()` (in module `dpdata.siesta.output`), 204  
`get_stress()` (in module `dpdata.abacus.scf`), 124  
`get_stress()` (in module `dpdata.qe.scf`), 201  
`get_stru_block()` (in module `dpdata.abacus.scf`), 124  
`get_terminal_NR2s()` (in module `dpdata.rdkit.sanitize`), 202  
`get_terminal_oxygens()` (in module `dpdata.rdkit.sanitize`), 202  
`get_to_methods()` (`dpdata.format.Format` static method), 223  
`get_varray()` (in module `dpdata.vasp.xml`), 204  
`get_virial()` (in module `dpdata.siesta.aiMD_output`), 203  
`get_virial()` (in module `dpdata.siesta.output`), 204  
`get_xyz_block_generator()` (`dpdata.cp2k.output.Cp2kSystems` method), 126  
`GromacsGroFormat` (class in `dpdata.plugins.gromacs`), 167

## H

`handle_single_log_frame()` (`dpdata.cp2k.output.Cp2kSystems` method), 126  
`handle_single_xyz_frame()` (`dpdata.cp2k.output.Cp2kSystems` method), 126

`handle_single_xyz_frame()` (*dpdata.xyz.quip\_gap\_xyz.QuipGapxyzSystems static method*), 205

`has_virial()` (*dpdata.LabeledSystem method*), 79

`has_virial()` (*dpdata.system.LabeledSystem method*), 245

`HybridDriver` (*class in dpdata.driver*), 219

## I

`is_terminal_nitrogen()` (*in module dpdata.rdkit.sanitize*), 202

`is_terminal_NR2()` (*in module dpdata.rdkit.sanitize*), 202

`is_terminal_oxygen()` (*in module dpdata.rdkit.sanitize*), 202

## K

`kekulize_aromatic_heterocycles()` (*in module dpdata.rdkit.sanitize*), 202

## L

`label()` (*dpdata.driver.Driver method*), 218

`label()` (*dpdata.driver.HybridDriver method*), 220

`label()` (*dpdata.plugins.amber.SQMDriver method*), 140

`label()` (*dpdata.plugins.ase.ASEDriver method*), 144

`label()` (*dpdata.plugins.deepmd.DPDriver method*), 151

`label()` (*dpdata.plugins.gaussian.GaussianDriver method*), 164

`LabeledSystem` (*class in dpdata*), 65

`LabeledSystem` (*class in dpdata.system*), 232

`LAMMPSDumpFormat` (*class in dpdata.plugins.lammps*), 168

`LAMMPSLmpFormat` (*class in dpdata.plugins.lammps*), 169

`LengthConversion` (*class in dpdata.unit*), 290

`ListFormat` (*class in dpdata.plugins.list*), 170

`lmpbox2box()` (*in module dpdata.lammps.lmp*), 131

`load()` (*dpdata.System static method*), 114

`load()` (*dpdata.system.System static method*), 280

`load_atom()` (*in module dpdata.openmx.omx*), 132

`load_atom_names()` (*in module dpdata.qe.traj*), 201

`load_atom_types()` (*in module dpdata.qe.traj*), 201

`load_block()` (*in module dpdata.qe.traj*), 201

`load_cell_parameters()` (*in module dpdata.qe.traj*), 201

`load_cellldm()` (*in module dpdata.qe.traj*), 201

`load_cells()` (*in module dpdata.openmx.omx*), 132

`load_coords()` (*in module dpdata.openmx.omx*), 132

`load_data()` (*in module dpdata.openmx.omx*), 132

`load_data()` (*in module dpdata.qe.traj*), 201

`load_energy()` (*in module dpdata.openmx.omx*), 132

`load_energy()` (*in module dpdata.qe.traj*), 201

`load_file()` (*in module dpdata.lammps.dump*), 131

`load_force()` (*in module dpdata.openmx.omx*), 132

`load_format()` (*in module dpdata.system*), 289

`load_key()` (*in module dpdata.qe.traj*), 201

`load_param_file()` (*in module dpdata.amber.mask*), 124

`load_param_file()` (*in module dpdata.openmx.omx*), 132

`load_param_file()` (*in module dpdata.qe.traj*), 201

`load_systems_from_file()` (*dpdata.MultiSystems method*), 95

`load_systems_from_file()` (*dpdata.system.MultiSystems method*), 261

`load_type()` (*in module dpdata.deepmd.mixed*), 127

`load_type()` (*in module dpdata.deepmd.raw*), 128

## M

`mae()` (*in module dpdata.stat*), 231

`make_gaussian_input()` (*in module dpdata.gaussian.gif*), 129

`make_sqm_in()` (*in module dpdata.amber.sqm*), 125

`make_unlabeled_stru()` (*in module dpdata.abacus.scf*), 124

`map_atom_types()` (*dpdata.System method*), 114

`map_atom_types()` (*dpdata.system.System method*), 280

`mass` (*dpdata.periodic\_table.Element property*), 227

`match_indices()` (*in module dpdata.plugins.n2p2*), 173

`minimize()` (*dpdata.driver.Minimizer method*), 221

`minimize()` (*dpdata.MultiSystems method*), 95

`minimize()` (*dpdata.plugins.amber.SQMMinimizer method*), 142

`minimize()` (*dpdata.plugins.ase.ASEMinimizer method*), 144

`minimize()` (*dpdata.System method*), 115

`minimize()` (*dpdata.system.MultiSystems method*), 261

`minimize()` (*dpdata.system.System method*), 281

`Minimizer` (*class in dpdata.driver*), 220

`mix_system()` (*dpdata.format.Format method*), 224

`mix_system()` (*dpdata.plugins.deepmd.DeePMDMixedFormat method*), 158

`mix_system()` (*in module dpdata.deepmd.mixed*), 127

`module`

`dpdata`, 57

`dpdata.abacus`, 123

`dpdata.abacus.md`, 123

`dpdata.abacus.relax`, 123

`dpdata.abacus.scf`, 123

`dpdata.amber`, 124

`dpdata.amber.mask`, 124

`dpdata.amber.md`, 124

`dpdata.amber.sqm`, 125

`dpdata.bond_order_system`, 206

`dpdata.cli`, 215

`dpdata.cp2k`, 125

[dpdata.cp2k.cell](#), 125  
[dpdata.cp2k.output](#), 125  
[dpdata.data\\_type](#), 216  
[dpdata.deepmd](#), 126  
[dpdata.deepmd.comp](#), 126  
[dpdata.deepmd.hdf5](#), 126  
[dpdata.deepmd.mixed](#), 127  
[dpdata.deepmd.raw](#), 128  
[dpdata.dftbplus](#), 128  
[dpdata.dftbplus.output](#), 128  
[dpdata.driver](#), 218  
[dpdata.fhi\\_aims](#), 128  
[dpdata.fhi\\_aims.output](#), 128  
[dpdata.format](#), 221  
[dpdata.gaussian](#), 129  
[dpdata.gaussian.gjf](#), 129  
[dpdata.gaussian.log](#), 130  
[dpdata.gromacs](#), 130  
[dpdata.gromacs.gro](#), 130  
[dpdata.lammps](#), 131  
[dpdata.lammps.dump](#), 131  
[dpdata.lammps.lmp](#), 131  
[dpdata.openmx](#), 132  
[dpdata.openmx.omx](#), 132  
[dpdata.orca](#), 132  
[dpdata.orca.output](#), 132  
[dpdata.periodic\\_table](#), 227  
[dpdata.plugin](#), 228  
[dpdata.plugins](#), 133  
[dpdata.plugins.3dmol](#), 133  
[dpdata.plugins.abacus](#), 134  
[dpdata.plugins.amber](#), 138  
[dpdata.plugins.ase](#), 143  
[dpdata.plugins.cp2k](#), 148  
[dpdata.plugins.deepmd](#), 151  
[dpdata.plugins.dftbplus](#), 160  
[dpdata.plugins.fhi\\_aims](#), 160  
[dpdata.plugins.gaussian](#), 163  
[dpdata.plugins.gromacs](#), 167  
[dpdata.plugins.lammps](#), 168  
[dpdata.plugins.list](#), 170  
[dpdata.plugins.n2p2](#), 171  
[dpdata.plugins.openmx](#), 173  
[dpdata.plugins.orca](#), 174  
[dpdata.plugins.psi4](#), 176  
[dpdata.plugins.pwmat](#), 178  
[dpdata.plugins.pymatgen](#), 181  
[dpdata.plugins.qe](#), 184  
[dpdata.plugins.rdkit](#), 186  
[dpdata.plugins.siesta](#), 188  
[dpdata.plugins.vasp](#), 191  
[dpdata.plugins.xyz](#), 196  
[dpdata.psi4](#), 199  
[dpdata.psi4.input](#), 199

[dpdata.psi4.output](#), 199  
[dpdata.pwmat](#), 200  
[dpdata.pwmat.atomconfig](#), 200  
[dpdata.pwmat.movement](#), 200  
[dpdata.pymatgen](#), 200  
[dpdata.pymatgen.molecule](#), 200  
[dpdata.pymatgen.structure](#), 200  
[dpdata.qe](#), 200  
[dpdata.qe.scf](#), 200  
[dpdata.qe.traj](#), 201  
[dpdata.rdkit](#), 201  
[dpdata.rdkit.sanitize](#), 201  
[dpdata.rdkit.utils](#), 203  
[dpdata.siesta](#), 203  
[dpdata.siesta.aiMD\\_output](#), 203  
[dpdata.siesta.output](#), 204  
[dpdata.stat](#), 228  
[dpdata.system](#), 232  
[dpdata.unit](#), 289  
[dpdata.utils](#), 290  
[dpdata.vasp](#), 204  
[dpdata.vasp.outcar](#), 204  
[dpdata.vasp.poscar](#), 204  
[dpdata.vasp.xml](#), 204  
[dpdata.xyz](#), 205  
[dpdata.xyz.quip\\_gap\\_xyz](#), 205  
[dpdata.xyz.xyz](#), 205  
[mol\\_edit\\_log\(\)](#) (in module [dpdata.rdkit.sanitize](#)), 202  
[mol\\_to\\_system\\_data\(\)](#) (in module [dpdata.rdkit.utils](#)), 203  
[MolFormat](#) (class in [dpdata.plugins.rdkit](#)), 186  
[MultiErrors](#) (class in [dpdata.stat](#)), 230  
[MultiMode](#) ([dpdata.format.Format](#) attribute), 222  
[MultiMode](#) ([dpdata.plugins.deepmd.DeePMDCompFormat](#) attribute), 152  
[MultiMode](#) ([dpdata.plugins.deepmd.DeePMDMixedFormat](#) attribute), 157  
[MultiMode](#) ([dpdata.plugins.deepmd.DeePMDRawFormat](#) attribute), 159  
[MultiSystems](#) (class in [dpdata](#)), 83  
[MultiSystems](#) (class in [dpdata.system](#)), 250

## N

[N2P2Format](#) (class in [dpdata.plugins.n2p2](#)), 171  
[name](#) ([dpdata.periodic\\_table.Element](#) property), 227  
[NATOMS](#) ([dpdata.data\\_type.Axis](#) attribute), 216  
[NBONDS](#) ([dpdata.data\\_type.Axis](#) attribute), 216  
[NFRAMES](#) ([dpdata.data\\_type.Axis](#) attribute), 216  
[nopbc](#) ([dpdata.System](#) property), 115  
[nopbc](#) ([dpdata.system.System](#) property), 281  
[NotImplemented](#) ([dpdata.format.Format.MultiModes](#) attribute), 222  
[NTYPES](#) ([dpdata.data\\_type.Axis](#) attribute), 216



## O

obtain\_frame() (in module *dpdata.siesta.output*), 204  
 obtain\_nframe() (in module *dpdata.siesta.aiMD.output*), 203

OPENMXFormat (class in *dpdata.plugins.openmx*), 173

ORCASPOutFormat (class in *dpdata.plugins.orca*), 174

## P

parse\_sqm\_out() (in module *dpdata.amber.sqm*), 125

perturb() (*dpdata.System* method), 115

perturb() (*dpdata.system.System* method), 281

pick\_atom\_idx() (*dpdata.MultiSystems* method), 95

pick\_atom\_idx() (*dpdata.System* method), 116

pick\_atom\_idx() (*dpdata.system.MultiSystems* method), 261

pick\_atom\_idx() (*dpdata.system.System* method), 282

pick\_by\_amber\_mask() (*dpdata.System* method), 116

pick\_by\_amber\_mask() (*dpdata.system.System* method), 282

pick\_by\_amber\_mask() (in module *dpdata.amber.mask*), 124

Plugin (class in *dpdata.plugin*), 228

post() (*dpdata.format.Format* static method), 224

post\_funcs (*dpdata.LabeledSystem* attribute), 79

post\_funcs (*dpdata.System* attribute), 116

post\_funcs (*dpdata.system.LabeledSystem* attribute), 245

post\_funcs (*dpdata.system.System* attribute), 282

predict() (*dpdata.MultiSystems* method), 96

predict() (*dpdata.System* method), 116

predict() (*dpdata.system.MultiSystems* method), 262

predict() (*dpdata.system.System* method), 282

PressureConversion (class in *dpdata.unit*), 290

print\_atoms() (in module *dpdata.rdkit.sanitize*), 202

print\_bonds() (in module *dpdata.rdkit.sanitize*), 202

PSI4InputFormat (class in *dpdata.plugins.psi4*), 176

PSI4OutFormat (class in *dpdata.plugins.psi4*), 177

PwmatAtomconfigFormat (class in *dpdata.plugins.pwmat*), 178

PwmatOutputFormat (class in *dpdata.plugins.pwmat*), 179

Py3DMolFormat (class in *dpdata.plugins.3dmol*), 133

PyMatgenCSEFormat (class in *dpdata.plugins.pymatgen*), 181

PyMatgenMoleculeFormat (class in *dpdata.plugins.pymatgen*), 181

PyMatgenStructureFormat (class in *dpdata.plugins.pymatgen*), 182

## Q

QECPWSCFFFormat (class in *dpdata.plugins.qe*), 184

QECPTrajFormat (class in *dpdata.plugins.qe*), 184

QuipGapXYZFormat (class in *dpdata.plugins.xyz*), 196

QuipGapxyzSystems (class in *dpdata.xyz.quip\_gap\_xyz*), 205

## R

radius (*dpdata.periodic\_table.Element* property), 227

read\_amber\_traj() (in module *dpdata.amber.md*), 124

read\_dftb\_plus() (in module *dpdata.dftbplus.output*), 128

read\_gaussian\_input() (in module *dpdata.gaussian.gif*), 130

read\_orca\_sp\_output() (in module *dpdata.orca.output*), 132

read\_psi4\_output() (in module *dpdata.psi4.output*), 199

real\_shape() (*dpdata.data\_type.DataType* method), 217

register() (*dpdata.driver.Driver* static method), 219

register() (*dpdata.driver.Minimizer* static method), 221

register() (*dpdata.format.Format* static method), 224

register() (*dpdata.plugin.Plugin* method), 228

register\_data\_type() (*dpdata.System* class method), 117

register\_data\_type() (*dpdata.system.System* class method), 283

register\_data\_type() (in module *dpdata.data\_type*), 217

register\_from() (*dpdata.format.Format* static method), 225

register\_to() (*dpdata.format.Format* static method), 225

regularize\_carbon\_bond\_order() (in module *dpdata.rdkit.sanitize*), 202

regularize\_formal\_charges() (in module *dpdata.rdkit.sanitize*), 202

regularize\_nitrogen\_bond\_order() (in module *dpdata.rdkit.sanitize*), 202

remove\_atom\_names() (*dpdata.System* method), 117

remove\_atom\_names() (*dpdata.system.System* method), 283

remove\_outlier() (*dpdata.LabeledSystem* method), 79

remove\_outlier() (*dpdata.system.LabeledSystem* method), 245

remove\_pbc() (*dpdata.System* method), 117

remove\_pbc() (*dpdata.system.System* method), 283

remove\_pbc() (in module *dpdata.utils*), 290

replace() (*dpdata.System* method), 117

replace() (*dpdata.system.System* method), 283

replicate() (*dpdata.System* method), 117

replicate() (*dpdata.system.System* method), 283

rmse() (in module *dpdata.stat*), 231

rot\_frame\_lower\_triangular() (*dpdata.LabeledSystem* method), 79

- rot\_frame\_lower\_triangular() (*dpdata.System method*), 117  
 rot\_frame\_lower\_triangular() (*dpdata.system.LabeledSystem method*), 246  
 rot\_frame\_lower\_triangular() (*dpdata.system.System method*), 283  
 rot\_lower\_triangular() (*dpdata.System method*), 117  
 rot\_lower\_triangular() (*dpdata.system.System method*), 283
- ## S
- safe\_get\_posi() (*in module dpdata.lammps.dump*), 131  
 sanitize() (*dpdata.rdkit.sanitize.Sanitizer method*), 202  
 sanitize\_carboxyl() (*in module dpdata.rdkit.sanitize*), 202  
 sanitize\_carboxyl\_Catom() (*in module dpdata.rdkit.sanitize*), 202  
 sanitize\_guanidine() (*in module dpdata.rdkit.sanitize*), 202  
 sanitize\_guanidine\_Catom() (*in module dpdata.rdkit.sanitize*), 202  
 sanitize\_mol() (*in module dpdata.rdkit.sanitize*), 202  
 sanitize\_nitrine\_Natom() (*in module dpdata.rdkit.sanitize*), 202  
 sanitize\_nitro() (*in module dpdata.rdkit.sanitize*), 202  
 sanitize\_nitro\_Natom() (*in module dpdata.rdkit.sanitize*), 202  
 sanitize\_phosphate() (*in module dpdata.rdkit.sanitize*), 203  
 sanitize\_phosphate\_Patom() (*in module dpdata.rdkit.sanitize*), 203  
 sanitize\_sulfate() (*in module dpdata.rdkit.sanitize*), 203  
 sanitize\_sulfate\_Satom() (*in module dpdata.rdkit.sanitize*), 203  
 SanitizeError, 201  
 Sanitizer (*class in dpdata.rdkit.sanitize*), 201  
 SdfFormat (*class in dpdata.plugins.rdkit*), 187  
 set\_value() (*dpdata.unit.Conversion method*), 289  
 short\_formula (*dpdata.System property*), 117  
 short\_formula (*dpdata.system.System property*), 283  
 short\_name (*dpdata.System property*), 117  
 short\_name (*dpdata.system.System property*), 283  
 shuffle() (*dpdata.System method*), 118  
 shuffle() (*dpdata.system.System method*), 284  
 SiestaAIMDOutputFormat (*class in dpdata.plugins.siesta*), 188  
 SiestaOutputFormat (*class in dpdata.plugins.siesta*), 190  
 sort\_atom\_names() (*dpdata.System method*), 118  
 sort\_atom\_names() (*dpdata.system.System method*), 284  
 sort\_atom\_names() (*in module dpdata.utils*), 290  
 sort\_atom\_types() (*dpdata.System method*), 118  
 sort\_atom\_types() (*dpdata.system.System method*), 284  
 split\_system() (*in module dpdata.deepmd.mixed*), 127  
 split\_traj() (*in module dpdata.lammps.dump*), 131  
 SQMDriver (*class in dpdata.plugins.amber*), 140  
 SQMINFormat (*class in dpdata.plugins.amber*), 140  
 SQMMinimizer (*class in dpdata.plugins.amber*), 142  
 SQMOutFormat (*class in dpdata.plugins.amber*), 142  
 sub\_system() (*dpdata.System method*), 118  
 sub\_system() (*dpdata.system.System method*), 284  
 super\_sanitize\_mol() (*in module dpdata.rdkit.sanitize*), 203  
 System (*class in dpdata*), 101  
 System (*class in dpdata.system*), 267  
 system\_data() (*in module dpdata.lammps.dump*), 131  
 system\_data() (*in module dpdata.lammps.lmp*), 131  
 system\_data\_to\_mol() (*in module dpdata.rdkit.utils*), 203  
 system\_info() (*in module dpdata.pwmat.movement*), 200  
 system\_info() (*in module dpdata.vasp.outcar*), 204  
 SYSTEM\_TYPE (*dpdata.stat.Errors attribute*), 229  
 SYSTEM\_TYPE (*dpdata.stat.ErrorsBase attribute*), 230  
 SYSTEM\_TYPE (*dpdata.stat.MultiErrors attribute*), 231
- ## T
- to() (*dpdata.MultiSystems method*), 96  
 to() (*dpdata.System method*), 118  
 to() (*dpdata.system.MultiSystems method*), 262  
 to() (*dpdata.system.System method*), 284  
 to\_3dmol() (*dpdata.LabeledSystem method*), 79  
 to\_3dmol() (*dpdata.MultiSystems method*), 96  
 to\_3dmol() (*dpdata.System method*), 118  
 to\_3dmol() (*dpdata.system.LabeledSystem method*), 246  
 to\_3dmol() (*dpdata.system.MultiSystems method*), 262  
 to\_3dmol() (*dpdata.system.System method*), 284  
 to\_abacus\_lcao\_md() (*dpdata.LabeledSystem method*), 79  
 to\_abacus\_lcao\_md() (*dpdata.MultiSystems method*), 96  
 to\_abacus\_lcao\_md() (*dpdata.System method*), 118  
 to\_abacus\_lcao\_md() (*dpdata.system.LabeledSystem method*), 246  
 to\_abacus\_lcao\_md() (*dpdata.system.MultiSystems method*), 262  
 to\_abacus\_lcao\_md() (*dpdata.system.System method*), 284

`to_abacus_lcao_relax()` (*dpdata.LabeledSystem method*), 79  
`to_abacus_lcao_relax()` (*dpdata.MultiSystems method*), 96  
`to_abacus_lcao_relax()` (*dpdata.System method*), 118  
`to_abacus_lcao_relax()` (*dpdata.system.LabeledSystem method*), 246  
`to_abacus_lcao_relax()` (*dpdata.system.MultiSystems method*), 262  
`to_abacus_lcao_relax()` (*dpdata.system.System method*), 284  
`to_abacus_lcao_scf()` (*dpdata.LabeledSystem method*), 79  
`to_abacus_lcao_scf()` (*dpdata.MultiSystems method*), 96  
`to_abacus_lcao_scf()` (*dpdata.System method*), 119  
`to_abacus_lcao_scf()` (*dpdata.system.LabeledSystem method*), 246  
`to_abacus_lcao_scf()` (*dpdata.system.MultiSystems method*), 262  
`to_abacus_lcao_scf()` (*dpdata.system.System method*), 285  
`to_abacus_md()` (*dpdata.LabeledSystem method*), 79  
`to_abacus_md()` (*dpdata.MultiSystems method*), 96  
`to_abacus_md()` (*dpdata.System method*), 119  
`to_abacus_md()` (*dpdata.system.LabeledSystem method*), 246  
`to_abacus_md()` (*dpdata.system.MultiSystems method*), 262  
`to_abacus_md()` (*dpdata.system.System method*), 285  
`to_abacus_pw_md()` (*dpdata.LabeledSystem method*), 80  
`to_abacus_pw_md()` (*dpdata.MultiSystems method*), 96  
`to_abacus_pw_md()` (*dpdata.System method*), 119  
`to_abacus_pw_md()` (*dpdata.system.LabeledSystem method*), 246  
`to_abacus_pw_md()` (*dpdata.system.MultiSystems method*), 262  
`to_abacus_pw_md()` (*dpdata.system.System method*), 285  
`to_abacus_pw_relax()` (*dpdata.LabeledSystem method*), 80  
`to_abacus_pw_relax()` (*dpdata.MultiSystems method*), 97  
`to_abacus_pw_relax()` (*dpdata.System method*), 119  
`to_abacus_pw_relax()` (*dpdata.system.LabeledSystem method*), 246  
`to_abacus_pw_relax()` (*dpdata.system.MultiSystems method*), 263  
`to_abacus_pw_relax()` (*dpdata.system.System method*), 285  
`to_abacus_pw_scf()` (*dpdata.LabeledSystem method*), 80  
`to_abacus_pw_scf()` (*dpdata.MultiSystems method*), 97  
`to_abacus_pw_scf()` (*dpdata.System method*), 119  
`to_abacus_pw_scf()` (*dpdata.system.LabeledSystem method*), 246  
`to_abacus_pw_scf()` (*dpdata.system.MultiSystems method*), 263  
`to_abacus_pw_scf()` (*dpdata.system.System method*), 285  
`to_abacus_relax()` (*dpdata.LabeledSystem method*), 80  
`to_abacus_relax()` (*dpdata.MultiSystems method*), 97  
`to_abacus_relax()` (*dpdata.System method*), 119  
`to_abacus_relax()` (*dpdata.system.LabeledSystem method*), 246  
`to_abacus_relax()` (*dpdata.system.MultiSystems method*), 263  
`to_abacus_relax()` (*dpdata.system.System method*), 285  
`to_abacus_scf()` (*dpdata.LabeledSystem method*), 80  
`to_abacus_scf()` (*dpdata.MultiSystems method*), 97  
`to_abacus_scf()` (*dpdata.System method*), 119  
`to_abacus_scf()` (*dpdata.system.LabeledSystem method*), 246  
`to_abacus_scf()` (*dpdata.system.MultiSystems method*), 263  
`to_abacus_scf()` (*dpdata.system.System method*), 285  
`to_abacus_stru()` (*dpdata.LabeledSystem method*), 80  
`to_abacus_stru()` (*dpdata.MultiSystems method*), 97  
`to_abacus_stru()` (*dpdata.System method*), 119  
`to_abacus_stru()` (*dpdata.system.LabeledSystem method*), 246  
`to_abacus_stru()` (*dpdata.system.MultiSystems method*), 263  
`to_abacus_stru()` (*dpdata.system.System method*), 285  
`to_amber_md()` (*dpdata.LabeledSystem method*), 80  
`to_amber_md()` (*dpdata.MultiSystems method*), 97  
`to_amber_md()` (*dpdata.System method*), 119  
`to_amber_md()` (*dpdata.system.LabeledSystem method*), 246  
`to_amber_md()` (*dpdata.system.MultiSystems method*), 263  
`to_amber_md()` (*dpdata.system.System method*), 285  
`to_ase_structure()` (*dpdata.LabeledSystem method*), 80  
`to_ase_structure()` (*dpdata.MultiSystems method*), 97  
`to_ase_structure()` (*dpdata.System method*), 119  
`to_ase_structure()` (*dpdata.system.LabeledSystem method*), 246  
`to_ase_structure()` (*dpdata.system.MultiSystems method*), 263  
`to_ase_structure()` (*dpdata.system.System method*), 285

`to_ase_traj()` (*dpdata.LabeledSystem* method), 80  
`to_ase_traj()` (*dpdata.MultiSystems* method), 97  
`to_ase_traj()` (*dpdata.System* method), 119  
`to_ase_traj()` (*dpdata.system.LabeledSystem* method), 246  
`to_ase_traj()` (*dpdata.system.MultiSystems* method), 263  
`to_ase_traj()` (*dpdata.system.System* method), 285  
`to_atomconfig()` (*dpdata.LabeledSystem* method), 80  
`to_atomconfig()` (*dpdata.MultiSystems* method), 97  
`to_atomconfig()` (*dpdata.System* method), 119  
`to_atomconfig()` (*dpdata.system.LabeledSystem* method), 246  
`to_atomconfig()` (*dpdata.system.MultiSystems* method), 263  
`to_atomconfig()` (*dpdata.system.System* method), 285  
`to_bond_order_system()` (*dpdata.format.Format* method), 226  
`to_bond_order_system()` (*dpdata.plugins.rdkit.MolFormat* method), 187  
`to_bond_order_system()` (*dpdata.plugins.rdkit.SdfFormat* method), 188  
`to_contcar()` (*dpdata.LabeledSystem* method), 80  
`to_contcar()` (*dpdata.MultiSystems* method), 97  
`to_contcar()` (*dpdata.System* method), 119  
`to_contcar()` (*dpdata.system.LabeledSystem* method), 246  
`to_contcar()` (*dpdata.system.MultiSystems* method), 263  
`to_contcar()` (*dpdata.system.System* method), 285  
`to_cp2k_aimd_output()` (*dpdata.LabeledSystem* method), 80  
`to_cp2k_aimd_output()` (*dpdata.MultiSystems* method), 97  
`to_cp2k_aimd_output()` (*dpdata.System* method), 119  
`to_cp2k_aimd_output()` (*dpdata.system.LabeledSystem* method), 246  
`to_cp2k_aimd_output()` (*dpdata.system.MultiSystems* method), 263  
`to_cp2k_aimd_output()` (*dpdata.system.System* method), 285  
`to_cp2k_output()` (*dpdata.LabeledSystem* method), 80  
`to_cp2k_output()` (*dpdata.MultiSystems* method), 97  
`to_cp2k_output()` (*dpdata.System* method), 119  
`to_cp2k_output()` (*dpdata.system.LabeledSystem* method), 246  
`to_cp2k_output()` (*dpdata.system.MultiSystems* method), 263  
`to_cp2k_output()` (*dpdata.system.System* method), 285  
`to_deepmd()` (*dpdata.LabeledSystem* method), 80  
`to_deepmd()` (*dpdata.MultiSystems* method), 97  
`to_deepmd()` (*dpdata.System* method), 119  
`to_deepmd()` (*dpdata.system.LabeledSystem* method), 246  
`to_deepmd()` (*dpdata.system.MultiSystems* method), 263  
`to_deepmd()` (*dpdata.system.System* method), 285  
`to_deepmd_comp()` (*dpdata.LabeledSystem* method), 80  
`to_deepmd_comp()` (*dpdata.MultiSystems* method), 97  
`to_deepmd_comp()` (*dpdata.System* method), 119  
`to_deepmd_comp()` (*dpdata.system.LabeledSystem* method), 247  
`to_deepmd_comp()` (*dpdata.system.MultiSystems* method), 263  
`to_deepmd_comp()` (*dpdata.system.System* method), 285  
`to_deepmd_hdf5()` (*dpdata.LabeledSystem* method), 80  
`to_deepmd_hdf5()` (*dpdata.MultiSystems* method), 97  
`to_deepmd_hdf5()` (*dpdata.System* method), 119  
`to_deepmd_hdf5()` (*dpdata.system.LabeledSystem* method), 247  
`to_deepmd_hdf5()` (*dpdata.system.MultiSystems* method), 263  
`to_deepmd_hdf5()` (*dpdata.system.System* method), 285  
`to_deepmd_npy()` (*dpdata.LabeledSystem* method), 80  
`to_deepmd_npy()` (*dpdata.MultiSystems* method), 97  
`to_deepmd_npy()` (*dpdata.System* method), 119  
`to_deepmd_npy()` (*dpdata.system.LabeledSystem* method), 247  
`to_deepmd_npy()` (*dpdata.system.MultiSystems* method), 263  
`to_deepmd_npy()` (*dpdata.system.System* method), 285  
`to_deepmd_npy_mixed()` (*dpdata.LabeledSystem* method), 80  
`to_deepmd_npy_mixed()` (*dpdata.MultiSystems* method), 97  
`to_deepmd_npy_mixed()` (*dpdata.System* method), 120  
`to_deepmd_npy_mixed()` (*dpdata.system.LabeledSystem* method), 247  
`to_deepmd_npy_mixed()` (*dpdata.system.MultiSystems* method), 263  
`to_deepmd_npy_mixed()` (*dpdata.system.System* method), 286  
`to_deepmd_raw()` (*dpdata.LabeledSystem* method), 80  
`to_deepmd_raw()` (*dpdata.MultiSystems* method), 97  
`to_deepmd_raw()` (*dpdata.System* method), 120  
`to_deepmd_raw()` (*dpdata.system.LabeledSystem* method), 247  
`to_deepmd_raw()` (*dpdata.system.MultiSystems* method), 263  
`to_deepmd_raw()` (*dpdata.system.System* method), 286  
`to_dftbplus()` (*dpdata.LabeledSystem* method), 81  
`to_dftbplus()` (*dpdata.MultiSystems* method), 97  
`to_dftbplus()` (*dpdata.System* method), 120  
`to_dftbplus()` (*dpdata.system.LabeledSystem* method), 247  
`to_dftbplus()` (*dpdata.system.MultiSystems* method), 263  
`to_dftbplus()` (*dpdata.system.System* method), 286  
`to_dump()` (*dpdata.LabeledSystem* method), 81



- `to_dump()` (*dpdata.MultiSystems* method), 98
- `to_dump()` (*dpdata.System* method), 120
- `to_dump()` (*dpdata.system.LabeledSystem* method), 247
- `to_dump()` (*dpdata.system.MultiSystems* method), 264
- `to_dump()` (*dpdata.system.System* method), 286
- `to_fhi_aims_md()` (*dpdata.LabeledSystem* method), 81
- `to_fhi_aims_md()` (*dpdata.MultiSystems* method), 98
- `to_fhi_aims_md()` (*dpdata.System* method), 120
- `to_fhi_aims_md()` (*dpdata.system.LabeledSystem* method), 247
- `to_fhi_aims_md()` (*dpdata.system.MultiSystems* method), 264
- `to_fhi_aims_md()` (*dpdata.system.System* method), 286
- `to_fhi_aims_output()` (*dpdata.LabeledSystem* method), 81
- `to_fhi_aims_output()` (*dpdata.MultiSystems* method), 98
- `to_fhi_aims_output()` (*dpdata.System* method), 120
- `to_fhi_aims_output()` (*dpdata.system.LabeledSystem* method), 247
- `to_fhi_aims_output()` (*dpdata.system.MultiSystems* method), 264
- `to_fhi_aims_output()` (*dpdata.system.System* method), 286
- `to_fhi_aims_scf()` (*dpdata.LabeledSystem* method), 81
- `to_fhi_aims_scf()` (*dpdata.MultiSystems* method), 98
- `to_fhi_aims_scf()` (*dpdata.System* method), 120
- `to_fhi_aims_scf()` (*dpdata.system.LabeledSystem* method), 247
- `to_fhi_aims_scf()` (*dpdata.system.MultiSystems* method), 264
- `to_fhi_aims_scf()` (*dpdata.system.System* method), 286
- `to_finalconfig()` (*dpdata.LabeledSystem* method), 81
- `to_finalconfig()` (*dpdata.MultiSystems* method), 98
- `to_finalconfig()` (*dpdata.System* method), 120
- `to_finalconfig()` (*dpdata.system.LabeledSystem* method), 247
- `to_finalconfig()` (*dpdata.system.MultiSystems* method), 264
- `to_finalconfig()` (*dpdata.system.System* method), 286
- `to_fmt_obj()` (*dpdata.bond\_order\_system.BondOrderSystem* method), 214
- `to_fmt_obj()` (*dpdata.BondOrderSystem* method), 65
- `to_fmt_obj()` (*dpdata.LabeledSystem* method), 81
- `to_fmt_obj()` (*dpdata.MultiSystems* method), 98
- `to_fmt_obj()` (*dpdata.System* method), 120
- `to_fmt_obj()` (*dpdata.system.LabeledSystem* method), 247
- `to_fmt_obj()` (*dpdata.system.MultiSystems* method), 264
- `to_fmt_obj()` (*dpdata.system.System* method), 286
- `to_gaussian_gjf()` (*dpdata.LabeledSystem* method), 81
- `to_gaussian_gjf()` (*dpdata.MultiSystems* method), 98
- `to_gaussian_gjf()` (*dpdata.System* method), 120
- `to_gaussian_gjf()` (*dpdata.system.LabeledSystem* method), 247
- `to_gaussian_gjf()` (*dpdata.system.MultiSystems* method), 264
- `to_gaussian_gjf()` (*dpdata.system.System* method), 286
- `to_gaussian_log()` (*dpdata.LabeledSystem* method), 81
- `to_gaussian_log()` (*dpdata.MultiSystems* method), 98
- `to_gaussian_log()` (*dpdata.System* method), 120
- `to_gaussian_log()` (*dpdata.system.LabeledSystem* method), 247
- `to_gaussian_log()` (*dpdata.system.MultiSystems* method), 264
- `to_gaussian_log()` (*dpdata.system.System* method), 286
- `to_gaussian_md()` (*dpdata.LabeledSystem* method), 81
- `to_gaussian_md()` (*dpdata.MultiSystems* method), 98
- `to_gaussian_md()` (*dpdata.System* method), 120
- `to_gaussian_md()` (*dpdata.system.LabeledSystem* method), 247
- `to_gaussian_md()` (*dpdata.system.MultiSystems* method), 264
- `to_gaussian_md()` (*dpdata.system.System* method), 286
- `to_gro()` (*dpdata.LabeledSystem* method), 81
- `to_gro()` (*dpdata.MultiSystems* method), 98
- `to_gro()` (*dpdata.System* method), 120
- `to_gro()` (*dpdata.system.LabeledSystem* method), 247
- `to_gro()` (*dpdata.system.MultiSystems* method), 264
- `to_gro()` (*dpdata.system.System* method), 286
- `to_gromacs_gro()` (*dpdata.LabeledSystem* method), 81
- `to_gromacs_gro()` (*dpdata.MultiSystems* method), 98
- `to_gromacs_gro()` (*dpdata.System* method), 120
- `to_gromacs_gro()` (*dpdata.system.LabeledSystem* method), 247
- `to_gromacs_gro()` (*dpdata.system.MultiSystems* method), 264
- `to_gromacs_gro()` (*dpdata.system.System* method), 286
- `to_labeled_system()` (*dpdata.format.Format* method), 226
- `to_labeled_system()` (*dpdata.plugins.ase.ASEStructureFormat* method), 146
- `to_labeled_system()` (*dpdata.plugins.n2p2.N2P2Format* method), 172
- `to_labeled_system()` (*dpdata.plugins.pymatgen.PyMatgenCSEFormat* method), 181
- `to_lammps_dump()` (*dpdata.LabeledSystem* method), 81
- `to_lammps_dump()` (*dpdata.MultiSystems* method), 98

[to\\_lammps\\_dump\(\)](#) (*dpdata.System method*), 120  
[to\\_lammps\\_dump\(\)](#) (*dpdata.system.LabeledSystem method*), 247  
[to\\_lammps\\_dump\(\)](#) (*dpdata.system.MultiSystems method*), 264  
[to\\_lammps\\_dump\(\)](#) (*dpdata.system.System method*), 286  
[to\\_lammps\\_lmp\(\)](#) (*dpdata.LabeledSystem method*), 81  
[to\\_lammps\\_lmp\(\)](#) (*dpdata.MultiSystems method*), 98  
[to\\_lammps\\_lmp\(\)](#) (*dpdata.System method*), 120  
[to\\_lammps\\_lmp\(\)](#) (*dpdata.system.LabeledSystem method*), 247  
[to\\_lammps\\_lmp\(\)](#) (*dpdata.system.MultiSystems method*), 264  
[to\\_lammps\\_lmp\(\)](#) (*dpdata.system.System method*), 286  
[to\\_list\(\)](#) (*dpdata.LabeledSystem method*), 81  
[to\\_list\(\)](#) (*dpdata.MultiSystems method*), 98  
[to\\_list\(\)](#) (*dpdata.System method*), 120  
[to\\_list\(\)](#) (*dpdata.system.LabeledSystem method*), 247  
[to\\_list\(\)](#) (*dpdata.system.MultiSystems method*), 264  
[to\\_list\(\)](#) (*dpdata.system.System method*), 286  
[to\\_lmp\(\)](#) (*dpdata.LabeledSystem method*), 81  
[to\\_lmp\(\)](#) (*dpdata.MultiSystems method*), 98  
[to\\_lmp\(\)](#) (*dpdata.System method*), 120  
[to\\_lmp\(\)](#) (*dpdata.system.LabeledSystem method*), 248  
[to\\_lmp\(\)](#) (*dpdata.system.MultiSystems method*), 264  
[to\\_lmp\(\)](#) (*dpdata.system.System method*), 286  
[to\\_mlmd\(\)](#) (*dpdata.LabeledSystem method*), 81  
[to\\_mlmd\(\)](#) (*dpdata.MultiSystems method*), 98  
[to\\_mlmd\(\)](#) (*dpdata.System method*), 120  
[to\\_mlmd\(\)](#) (*dpdata.system.LabeledSystem method*), 248  
[to\\_mlmd\(\)](#) (*dpdata.system.MultiSystems method*), 264  
[to\\_mlmd\(\)](#) (*dpdata.system.System method*), 286  
[to\\_mol\(\)](#) (*dpdata.LabeledSystem method*), 81  
[to\\_mol\(\)](#) (*dpdata.MultiSystems method*), 98  
[to\\_mol\(\)](#) (*dpdata.System method*), 120  
[to\\_mol\(\)](#) (*dpdata.system.LabeledSystem method*), 248  
[to\\_mol\(\)](#) (*dpdata.system.MultiSystems method*), 264  
[to\\_mol\(\)](#) (*dpdata.system.System method*), 286  
[to\\_mol\\_file\(\)](#) (*dpdata.LabeledSystem method*), 81  
[to\\_mol\\_file\(\)](#) (*dpdata.MultiSystems method*), 98  
[to\\_mol\\_file\(\)](#) (*dpdata.System method*), 121  
[to\\_mol\\_file\(\)](#) (*dpdata.system.LabeledSystem method*), 248  
[to\\_mol\\_file\(\)](#) (*dpdata.system.MultiSystems method*), 264  
[to\\_mol\\_file\(\)](#) (*dpdata.system.System method*), 287  
[to\\_movement\(\)](#) (*dpdata.LabeledSystem method*), 81  
[to\\_movement\(\)](#) (*dpdata.MultiSystems method*), 98  
[to\\_movement\(\)](#) (*dpdata.System method*), 121  
[to\\_movement\(\)](#) (*dpdata.system.LabeledSystem method*), 248  
[to\\_movement\(\)](#) (*dpdata.system.MultiSystems method*), 264  
[to\\_movement\(\)](#) (*dpdata.system.System method*), 287  
[to\\_multi\\_systems\(\)](#) (*dpdata.format.Format method*), 226  
[to\\_multi\\_systems\(\)](#) (*dpdata.plugins.deepmd.DeePMDHDF5Format method*), 155  
[to\\_n2p2\(\)](#) (*dpdata.LabeledSystem method*), 82  
[to\\_n2p2\(\)](#) (*dpdata.MultiSystems method*), 98  
[to\\_n2p2\(\)](#) (*dpdata.System method*), 121  
[to\\_n2p2\(\)](#) (*dpdata.system.LabeledSystem method*), 248  
[to\\_n2p2\(\)](#) (*dpdata.system.MultiSystems method*), 264  
[to\\_n2p2\(\)](#) (*dpdata.system.System method*), 287  
[to\\_openmx\\_md\(\)](#) (*dpdata.LabeledSystem method*), 82  
[to\\_openmx\\_md\(\)](#) (*dpdata.MultiSystems method*), 99  
[to\\_openmx\\_md\(\)](#) (*dpdata.System method*), 121  
[to\\_openmx\\_md\(\)](#) (*dpdata.system.LabeledSystem method*), 248  
[to\\_openmx\\_md\(\)](#) (*dpdata.system.MultiSystems method*), 265  
[to\\_openmx\\_md\(\)](#) (*dpdata.system.System method*), 287  
[to\\_orca\\_spout\(\)](#) (*dpdata.LabeledSystem method*), 82  
[to\\_orca\\_spout\(\)](#) (*dpdata.MultiSystems method*), 99  
[to\\_orca\\_spout\(\)](#) (*dpdata.System method*), 121  
[to\\_orca\\_spout\(\)](#) (*dpdata.system.LabeledSystem method*), 248  
[to\\_orca\\_spout\(\)](#) (*dpdata.system.MultiSystems method*), 265  
[to\\_orca\\_spout\(\)](#) (*dpdata.system.System method*), 287  
[to\\_outcar\(\)](#) (*dpdata.LabeledSystem method*), 82  
[to\\_outcar\(\)](#) (*dpdata.MultiSystems method*), 99  
[to\\_outcar\(\)](#) (*dpdata.System method*), 121  
[to\\_outcar\(\)](#) (*dpdata.system.LabeledSystem method*), 248  
[to\\_outcar\(\)](#) (*dpdata.system.MultiSystems method*), 265  
[to\\_outcar\(\)](#) (*dpdata.system.System method*), 287  
[to\\_poscar\(\)](#) (*dpdata.LabeledSystem method*), 82  
[to\\_poscar\(\)](#) (*dpdata.MultiSystems method*), 99  
[to\\_poscar\(\)](#) (*dpdata.System method*), 121  
[to\\_poscar\(\)](#) (*dpdata.system.LabeledSystem method*), 248  
[to\\_poscar\(\)](#) (*dpdata.system.MultiSystems method*), 265  
[to\\_poscar\(\)](#) (*dpdata.system.System method*), 287  
[to\\_psi4\\_inp\(\)](#) (*dpdata.LabeledSystem method*), 82  
[to\\_psi4\\_inp\(\)](#) (*dpdata.MultiSystems method*), 99  
[to\\_psi4\\_inp\(\)](#) (*dpdata.System method*), 121  
[to\\_psi4\\_inp\(\)](#) (*dpdata.system.LabeledSystem method*), 248  
[to\\_psi4\\_inp\(\)](#) (*dpdata.system.MultiSystems method*), 265  
[to\\_psi4\\_inp\(\)](#) (*dpdata.system.System method*), 287  
[to\\_psi4\\_out\(\)](#) (*dpdata.LabeledSystem method*), 82  
[to\\_psi4\\_out\(\)](#) (*dpdata.MultiSystems method*), 99  
[to\\_psi4\\_out\(\)](#) (*dpdata.System method*), 121  
[to\\_psi4\\_out\(\)](#) (*dpdata.system.LabeledSystem method*), 248

- to\_psi4\_out() (*dpdata.system.MultiSystems* method), 265
- to\_psi4\_out() (*dpdata.system.System* method), 287
- to\_pwmat\_atomconfig() (*dpdata.LabeledSystem* method), 82
- to\_pwmat\_atomconfig() (*dpdata.MultiSystems* method), 99
- to\_pwmat\_atomconfig() (*dpdata.System* method), 121
- to\_pwmat\_atomconfig() (*dpdata.system.LabeledSystem* method), 248
- to\_pwmat\_atomconfig() (*dpdata.system.MultiSystems* method), 265
- to\_pwmat\_atomconfig() (*dpdata.system.System* method), 287
- to\_pwmat\_finalconfig() (*dpdata.LabeledSystem* method), 82
- to\_pwmat\_finalconfig() (*dpdata.MultiSystems* method), 99
- to\_pwmat\_finalconfig() (*dpdata.System* method), 121
- to\_pwmat\_finalconfig() (*dpdata.system.LabeledSystem* method), 248
- to\_pwmat\_finalconfig() (*dpdata.system.MultiSystems* method), 265
- to\_pwmat\_finalconfig() (*dpdata.system.System* method), 287
- to\_pwmat\_mlmd() (*dpdata.LabeledSystem* method), 82
- to\_pwmat\_mlmd() (*dpdata.MultiSystems* method), 99
- to\_pwmat\_mlmd() (*dpdata.System* method), 121
- to\_pwmat\_mlmd() (*dpdata.system.LabeledSystem* method), 248
- to\_pwmat\_mlmd() (*dpdata.system.MultiSystems* method), 265
- to\_pwmat\_mlmd() (*dpdata.system.System* method), 287
- to\_pwmat\_movement() (*dpdata.LabeledSystem* method), 82
- to\_pwmat\_movement() (*dpdata.MultiSystems* method), 99
- to\_pwmat\_movement() (*dpdata.System* method), 121
- to\_pwmat\_movement() (*dpdata.system.LabeledSystem* method), 248
- to\_pwmat\_movement() (*dpdata.system.MultiSystems* method), 265
- to\_pwmat\_movement() (*dpdata.system.System* method), 287
- to\_pwmat\_output() (*dpdata.LabeledSystem* method), 82
- to\_pwmat\_output() (*dpdata.MultiSystems* method), 99
- to\_pwmat\_output() (*dpdata.System* method), 121
- to\_pwmat\_output() (*dpdata.system.LabeledSystem* method), 248
- to\_pwmat\_output() (*dpdata.system.MultiSystems* method), 265
- to\_pwmat\_output() (*dpdata.system.System* method), 287
- to\_pymatgen\_ComputedStructureEntry() (*dpdata.LabeledSystem* method), 82
- to\_pymatgen\_computedstructureentry() (*dpdata.LabeledSystem* method), 82
- to\_pymatgen\_ComputedStructureEntry() (*dpdata.MultiSystems* method), 99
- to\_pymatgen\_computedstructureentry() (*dpdata.MultiSystems* method), 99
- to\_pymatgen\_ComputedStructureEntry() (*dpdata.System* method), 121
- to\_pymatgen\_computedstructureentry() (*dpdata.System* method), 121
- to\_pymatgen\_ComputedStructureEntry() (*dpdata.system.LabeledSystem* method), 248
- to\_pymatgen\_computedstructureentry() (*dpdata.system.LabeledSystem* method), 248
- to\_pymatgen\_ComputedStructureEntry() (*dpdata.system.MultiSystems* method), 265
- to\_pymatgen\_computedstructureentry() (*dpdata.system.MultiSystems* method), 265
- to\_pymatgen\_ComputedStructureEntry() (*dpdata.system.System* method), 287
- to\_pymatgen\_computedstructureentry() (*dpdata.system.System* method), 287
- to\_pymatgen\_molecule() (*dpdata.LabeledSystem* method), 82
- to\_pymatgen\_molecule() (*dpdata.MultiSystems* method), 99
- to\_pymatgen\_molecule() (*dpdata.System* method), 121
- to\_pymatgen\_molecule() (*dpdata.system.LabeledSystem* method), 249
- to\_pymatgen\_molecule() (*dpdata.system.MultiSystems* method), 265
- to\_pymatgen\_molecule() (*dpdata.system.System* method), 287
- to\_pymatgen\_structure() (*dpdata.LabeledSystem* method), 82
- to\_pymatgen\_structure() (*dpdata.MultiSystems* method), 99
- to\_pymatgen\_structure() (*dpdata.System* method), 121
- to\_pymatgen\_structure() (*dpdata.system.LabeledSystem* method), 249
- to\_pymatgen\_structure() (*dpdata.system.MultiSystems* method), 265
- to\_pymatgen\_structure() (*dpdata.system.System* method), 287
- to\_qe\_cp\_traj() (*dpdata.LabeledSystem* method), 82
- to\_qe\_cp\_traj() (*dpdata.MultiSystems* method), 99
- to\_qe\_cp\_traj() (*dpdata.System* method), 121
- to\_qe\_cp\_traj() (*dpdata.system.LabeledSystem* method), 249

[to\\_qe\\_cp\\_traj\(\)](#) (*dpdata.system.MultiSystems method*), 265  
[to\\_qe\\_cp\\_traj\(\)](#) (*dpdata.system.System method*), 287  
[to\\_qe\\_pw\\_scf\(\)](#) (*dpdata.LabeledSystem method*), 82  
[to\\_qe\\_pw\\_scf\(\)](#) (*dpdata.MultiSystems method*), 99  
[to\\_qe\\_pw\\_scf\(\)](#) (*dpdata.System method*), 122  
[to\\_qe\\_pw\\_scf\(\)](#) (*dpdata.system.LabeledSystem method*), 249  
[to\\_qe\\_pw\\_scf\(\)](#) (*dpdata.system.MultiSystems method*), 265  
[to\\_qe\\_pw\\_scf\(\)](#) (*dpdata.system.System method*), 288  
[to\\_quip\\_gap\\_xyz\(\)](#) (*dpdata.LabeledSystem method*), 82  
[to\\_quip\\_gap\\_xyz\(\)](#) (*dpdata.MultiSystems method*), 99  
[to\\_quip\\_gap\\_xyz\(\)](#) (*dpdata.System method*), 122  
[to\\_quip\\_gap\\_xyz\(\)](#) (*dpdata.system.LabeledSystem method*), 249  
[to\\_quip\\_gap\\_xyz\(\)](#) (*dpdata.system.MultiSystems method*), 265  
[to\\_quip\\_gap\\_xyz\(\)](#) (*dpdata.system.System method*), 288  
[to\\_quip\\_gap\\_xyz\\_file\(\)](#) (*dpdata.LabeledSystem method*), 83  
[to\\_quip\\_gap\\_xyz\\_file\(\)](#) (*dpdata.MultiSystems method*), 99  
[to\\_quip\\_gap\\_xyz\\_file\(\)](#) (*dpdata.System method*), 122  
[to\\_quip\\_gap\\_xyz\\_file\(\)](#) (*dpdata.system.LabeledSystem method*), 249  
[to\\_quip\\_gap\\_xyz\\_file\(\)](#) (*dpdata.system.MultiSystems method*), 265  
[to\\_quip\\_gap\\_xyz\\_file\(\)](#) (*dpdata.system.System method*), 288  
[to\\_sdf\(\)](#) (*dpdata.LabeledSystem method*), 83  
[to\\_sdf\(\)](#) (*dpdata.MultiSystems method*), 100  
[to\\_sdf\(\)](#) (*dpdata.System method*), 122  
[to\\_sdf\(\)](#) (*dpdata.system.LabeledSystem method*), 249  
[to\\_sdf\(\)](#) (*dpdata.system.MultiSystems method*), 266  
[to\\_sdf\(\)](#) (*dpdata.system.System method*), 288  
[to\\_sdf\\_file\(\)](#) (*dpdata.LabeledSystem method*), 83  
[to\\_sdf\\_file\(\)](#) (*dpdata.MultiSystems method*), 100  
[to\\_sdf\\_file\(\)](#) (*dpdata.System method*), 122  
[to\\_sdf\\_file\(\)](#) (*dpdata.system.LabeledSystem method*), 249  
[to\\_sdf\\_file\(\)](#) (*dpdata.system.MultiSystems method*), 266  
[to\\_sdf\\_file\(\)](#) (*dpdata.system.System method*), 288  
[to\\_siesta\\_aimd\\_output\(\)](#) (*dpdata.LabeledSystem method*), 83  
[to\\_siesta\\_aimd\\_output\(\)](#) (*dpdata.MultiSystems method*), 100  
[to\\_siesta\\_aimd\\_output\(\)](#) (*dpdata.System method*), 122  
[to\\_siesta\\_aimd\\_output\(\)](#) (*dpdata.system.LabeledSystem method*), 249  
[to\\_siesta\\_aimd\\_output\(\)](#) (*dpdata.system.MultiSystems method*), 266  
[to\\_siesta\\_aimd\\_output\(\)](#) (*dpdata.system.System method*), 288  
[to\\_siesta\\_output\(\)](#) (*dpdata.LabeledSystem method*), 83  
[to\\_siesta\\_output\(\)](#) (*dpdata.MultiSystems method*), 100  
[to\\_siesta\\_output\(\)](#) (*dpdata.System method*), 122  
[to\\_siesta\\_output\(\)](#) (*dpdata.system.LabeledSystem method*), 249  
[to\\_siesta\\_output\(\)](#) (*dpdata.system.MultiSystems method*), 266  
[to\\_siesta\\_output\(\)](#) (*dpdata.system.System method*), 288  
[to\\_sqm\\_in\(\)](#) (*dpdata.LabeledSystem method*), 83  
[to\\_sqm\\_in\(\)](#) (*dpdata.MultiSystems method*), 100  
[to\\_sqm\\_in\(\)](#) (*dpdata.System method*), 122  
[to\\_sqm\\_in\(\)](#) (*dpdata.system.LabeledSystem method*), 249  
[to\\_sqm\\_in\(\)](#) (*dpdata.system.MultiSystems method*), 266  
[to\\_sqm\\_in\(\)](#) (*dpdata.system.System method*), 288  
[to\\_sqm\\_out\(\)](#) (*dpdata.LabeledSystem method*), 83  
[to\\_sqm\\_out\(\)](#) (*dpdata.MultiSystems method*), 100  
[to\\_sqm\\_out\(\)](#) (*dpdata.System method*), 122  
[to\\_sqm\\_out\(\)](#) (*dpdata.system.LabeledSystem method*), 249  
[to\\_sqm\\_out\(\)](#) (*dpdata.system.MultiSystems method*), 266  
[to\\_sqm\\_out\(\)](#) (*dpdata.system.System method*), 288  
[to\\_stru\(\)](#) (*dpdata.LabeledSystem method*), 83  
[to\\_stru\(\)](#) (*dpdata.MultiSystems method*), 100  
[to\\_stru\(\)](#) (*dpdata.System method*), 122  
[to\\_stru\(\)](#) (*dpdata.system.LabeledSystem method*), 249  
[to\\_stru\(\)](#) (*dpdata.system.MultiSystems method*), 266  
[to\\_stru\(\)](#) (*dpdata.system.System method*), 288  
[to\\_system\(\)](#) (*dpdata.format.Format method*), 226  
[to\\_system\(\)](#) (*dpdata.plugins.3dmol.Py3DMolFormat method*), 133  
[to\\_system\(\)](#) (*dpdata.plugins.abacus.AbacusSTRUFormat method*), 137  
[to\\_system\(\)](#) (*dpdata.plugins.amber.SQMINFormat method*), 141  
[to\\_system\(\)](#) (*dpdata.plugins.ase.ASEStructureFormat method*), 146  
[to\\_system\(\)](#) (*dpdata.plugins.deepmd.DeePMDCompFormat method*), 153  
[to\\_system\(\)](#) (*dpdata.plugins.deepmd.DeePMDHDF5Format method*), 155  
[to\\_system\(\)](#) (*dpdata.plugins.deepmd.DeePMDMixedFormat method*), 158  
[to\\_system\(\)](#) (*dpdata.plugins.deepmd.DeePMDRawFormat method*), 160



[to\\_system\(\)](#) (*dpdata.plugins.gaussian.GaussianGJFFormat method*), 266  
[to\\_system\(\)](#) (*dpdata.plugins.gromacs.GromacsGroFormat method*), 163  
[to\\_system\(\)](#) (*dpdata.plugins.lammps.LAMMPSLmpFormat method*), 170  
[to\\_system\(\)](#) (*dpdata.plugins.list.ListFormat method*), 171  
[to\\_system\(\)](#) (*dpdata.plugins.psi4.PSI4InputFormat method*), 176  
[to\\_system\(\)](#) (*dpdata.plugins.pwmat.PwmatAtomconfigFormat method*), 179  
[to\\_system\(\)](#) (*dpdata.plugins.pymatgen.PyMatgenMoleculeFormat method*), 182  
[to\\_system\(\)](#) (*dpdata.plugins.pymatgen.PyMatgenStructureFormat method*), 183  
[to\\_system\(\)](#) (*dpdata.plugins.vasp.VASPPoscarFormat method*), 193  
[to\\_system\(\)](#) (*dpdata.plugins.vasp.VASPStringFormat method*), 194  
[to\\_system\(\)](#) (*dpdata.plugins.xyz.XYZFormat method*), 198  
[to\\_system\\_data\(\)](#) (in module *dpdata.deepmd.comp*), 126  
[to\\_system\\_data\(\)](#) (in module *dpdata.deepmd.hdf5*), 126  
[to\\_system\\_data\(\)](#) (in module *dpdata.deepmd.mixed*), 127  
[to\\_system\\_data\(\)](#) (in module *dpdata.deepmd.raw*), 128  
[to\\_system\\_data\(\)](#) (in module *dpdata.gaussian.log*), 130  
[to\\_system\\_data\(\)](#) (in module *dpdata.lammps.lmp*), 131  
[to\\_system\\_data\(\)](#) (in module *dpdata.openmx.omx*), 132  
[to\\_system\\_data\(\)](#) (in module *dpdata.pwmat.atomconfig*), 200  
[to\\_system\\_data\(\)](#) (in module *dpdata.pymatgen.molecule*), 200  
[to\\_system\\_data\(\)](#) (in module *dpdata.qe.traj*), 201  
[to\\_system\\_data\(\)](#) (in module *dpdata.vasp.poscar*), 204  
[to\\_system\\_label\(\)](#) (in module *dpdata.openmx.omx*), 132  
[to\\_system\\_label\(\)](#) (in module *dpdata.qe.traj*), 201  
[to\\_vasp\\_contcar\(\)](#) (*dpdata.LabeledSystem method*), 83  
[to\\_vasp\\_contcar\(\)](#) (*dpdata.MultiSystems method*), 100  
[to\\_vasp\\_contcar\(\)](#) (*dpdata.System method*), 122  
[to\\_vasp\\_contcar\(\)](#) (*dpdata.system.LabeledSystem method*), 249  
[to\\_vasp\\_contcar\(\)](#) (*dpdata.system.MultiSystems method*), 266  
[to\\_vasp\\_outcar\(\)](#) (*dpdata.LabeledSystem method*), 83  
[to\\_vasp\\_outcar\(\)](#) (*dpdata.MultiSystems method*), 100  
[to\\_vasp\\_outcar\(\)](#) (*dpdata.System method*), 122  
[to\\_vasp\\_outcar\(\)](#) (*dpdata.system.LabeledSystem method*), 249  
[to\\_vasp\\_outcar\(\)](#) (*dpdata.system.MultiSystems method*), 266  
[to\\_vasp\\_poscar\(\)](#) (*dpdata.LabeledSystem method*), 83  
[to\\_vasp\\_poscar\(\)](#) (*dpdata.MultiSystems method*), 100  
[to\\_vasp\\_poscar\(\)](#) (*dpdata.System method*), 122  
[to\\_vasp\\_poscar\(\)](#) (*dpdata.system.LabeledSystem method*), 249  
[to\\_vasp\\_poscar\(\)](#) (*dpdata.system.MultiSystems method*), 266  
[to\\_vasp\\_string\(\)](#) (*dpdata.LabeledSystem method*), 83  
[to\\_vasp\\_string\(\)](#) (*dpdata.MultiSystems method*), 100  
[to\\_vasp\\_string\(\)](#) (*dpdata.System method*), 122  
[to\\_vasp\\_string\(\)](#) (*dpdata.system.LabeledSystem method*), 249  
[to\\_vasp\\_string\(\)](#) (*dpdata.system.MultiSystems method*), 266  
[to\\_vasp\\_string\(\)](#) (*dpdata.system.System method*), 288  
[to\\_vasp\\_xml\(\)](#) (*dpdata.LabeledSystem method*), 83  
[to\\_vasp\\_xml\(\)](#) (*dpdata.MultiSystems method*), 100  
[to\\_vasp\\_xml\(\)](#) (*dpdata.System method*), 122  
[to\\_vasp\\_xml\(\)](#) (*dpdata.system.LabeledSystem method*), 249  
[to\\_vasp\\_xml\(\)](#) (*dpdata.system.MultiSystems method*), 266  
[to\\_vasp\\_xml\(\)](#) (*dpdata.system.System method*), 288  
[to\\_xml\(\)](#) (*dpdata.LabeledSystem method*), 83  
[to\\_xml\(\)](#) (*dpdata.MultiSystems method*), 100  
[to\\_xml\(\)](#) (*dpdata.System method*), 122  
[to\\_xml\(\)](#) (*dpdata.system.LabeledSystem method*), 249  
[to\\_xml\(\)](#) (*dpdata.system.MultiSystems method*), 266  
[to\\_xml\(\)](#) (*dpdata.system.System method*), 288  
[to\\_xyz\(\)](#) (*dpdata.LabeledSystem method*), 83  
[to\\_xyz\(\)](#) (*dpdata.MultiSystems method*), 100  
[to\\_xyz\(\)](#) (*dpdata.System method*), 122  
[to\\_xyz\(\)](#) (*dpdata.system.LabeledSystem method*), 250  
[to\\_xyz\(\)](#) (*dpdata.system.MultiSystems method*), 266  
[to\\_xyz\(\)](#) (*dpdata.system.System method*), 288  
[train\\_test\\_split\(\)](#) (*dpdata.MultiSystems method*), 100  
[train\\_test\\_split\(\)](#) (*dpdata.system.MultiSystems method*), 266

## U

[uniq\\_atom\\_names\(\)](#) (in module *dpdata.utils*), 290

`uniq_formula` (*dpdata.System* property), [122](#)  
`uniq_formula` (*dpdata.system.System* property), [288](#)  
`UnwrapWarning`, [131](#)  
`utf8len()` (*in module dpdata.utils*), [291](#)

## V

`value()` (*dpdata.unit.Conversion* method), [289](#)  
`VASPOutcarFormat` (*class in dpdata.plugins.vasp*), [191](#)  
`VASPPoscarFormat` (*class in dpdata.plugins.vasp*), [192](#)  
`VASPStringFormat` (*class in dpdata.plugins.vasp*), [193](#)  
`VASXMLFormat` (*class in dpdata.plugins.vasp*), [194](#)

## W

`write_psi4_input()` (*in module dpdata.psi4.input*),  
[199](#)

## X

`X` (*dpdata.periodic\_table.Element* property), [227](#)  
`xyz_to_coord()` (*in module dpdata.xyz.xyz*), [205](#)  
`XYZFormat` (*class in dpdata.plugins.xyz*), [197](#)

## Z

`Z` (*dpdata.periodic\_table.Element* property), [227](#)