

---

# **InterMol Documentation**

*Release 0.2a1*

**Christoph Klein, Christopher Lee, Ellen Zhong, and Michael Shirts**

January 25, 2015



|          |                                    |           |
|----------|------------------------------------|-----------|
| <b>1</b> | <b>Installation</b>                | <b>1</b>  |
| 1.1      | Installation . . . . .             | 1         |
| <b>2</b> | <b>Development</b>                 | <b>3</b>  |
| 2.1      | Contributing . . . . .             | 3         |
| <b>3</b> | <b>API Reference</b>               | <b>5</b>  |
| 3.1      | intermol package . . . . .         | 5         |
| 3.2      | intermol.gromacs package . . . . . | 40        |
| 3.3      | intermol.lammps package . . . . .  | 45        |
| 3.4      | intermol.forces package . . . . .  | 48        |
| <b>4</b> | <b>License</b>                     | <b>75</b> |
| <b>5</b> | <b>Indices and tables</b>          | <b>77</b> |
|          | <b>Python Module Index</b>         | <b>79</b> |



---

## Installation

---

### 1.1 Installation

#### 1.1.1 Install with pip (coming soon!)

InterMol will be added to PyPI as soon as we get our first stable release up and running.

#### 1.1.2 Install from source

```
$ git clone https://github.com/shirtsgroup/InterMol.git
$ cd InterMol
$ python setup.py install
```

Or if you plan on contributing something:

```
$ python setup.py develop
```

#### 1.1.3 Dependencies

To use InterMol, the following libraries and software will need to be installed.

**Linux, Mac OS X or Windows operating system** We develop mainly on 64-bit Mac and CentOS machines. TravisCI is currently only set up to perform testing on Debian.

**Python == 2.7** Once our unit tests flesh out a bit more, we intend to add support for  $\geq 2.6$ .

**NumPy  $\geq 1.6.0$**  Numpy is the base package for numerical computing in python.

**simtk.unit  $\geq 0.1$**  Python unit classes for dimensional analysis and unit conversion.



## 2.1 Contributing

Contributions are welcome and they are greatly appreciated! Every little bit helps and credit will always be given.

### 2.1.1 Bug Reports

If you find a bug, please file a detailed issue [here](#) and **provide input files** so that we can try to reproduce the problem.

### 2.1.2 Code Style

PEP8 describes the style guide for python code. Please try to follow it when reasonable, in particular the naming conventions (e.g. public methods should use `lower_case_with_underscores` instead of `camelCase`).

Two helpful tools for checking your code are `flake8` and `pylint`. The latter is a lot pickier but generally, your code should pass most `flake8` tests:

```
$ pip install flake8 pylint
```

Also some IDE's like `PyCharm` will notify you of PEP8 violations as you code.

---

**Note:** Some older chunks of the code do not yet adhere to PEP8. If you find something easily fixable, please do so!

---

### 2.1.3 Running our tests

InterMol uses `py.test` for unit testing. To run them simply type run the following while in the base directory:

```
$ py.test
```

We need a LOT more tests so any help here is especially welcome!

To debug failing tests, you typically get a clearer output by running a subset of the tests, e.g.:

```
$ python test_gromacs.py --type unit
```

which prints out log files to `intermol/tests/unit_test_outputs/from_gromacs/[system name]/debug.log`. Re-running a single test is best done directly via the `convert.py` script, e.g.:

```
$ python convert.py --gro_in tests/gromacs/unit_tests/[system name]/[system name].{top,gro} --gromacs
```

---

**Note:** If you have any ideas or suggestions for streamlining the testing process please let us know by filing an issue or opening a pull request!

---

## 2.1.4 Git Flow

Because we are supporting multiple molecular dynamics engines that should all work independently, we try to keep development of each engine in a separate branch until the basics are working.

To this end, we've started working with the [git flow branching model](#). The basic things to know are:

1. The `master` branch is strictly used for releases.
2. The `develop` branch is where (!!!) development happens.
3. When we start working on a new engine, we create a feature branch. E.g., at the time of this writing, there are branches called `feature/lammps` and `feature/desmond`. Once the overall structure in this branch is fairly stable and has a good amounts of tests, we merge it into `develop`.

So what do you, the interested developer, need to know?

1. Don't make pull requests against `master`.
2. Choose either `develop` or the appropriate `feature/*` branch to pull against.

For more reading and a neat tool to help with branching see:

<http://jeffkreeftmeijer.com/2010/why-arent-you-using-git-flow/>

<https://github.com/nvie/gitflow>

<http://danielkummer.github.io/git-flow-cheatsheet/>



## 3.1 intermol package

### 3.1.1 Subpackages

#### intermol.forces package

##### Submodules

##### intermol.forces.abstract\_2\_virtual\_type module

**class** `intermol.forces.abstract_2_virtual_type.Abstract2VirtualType` (*bondingtype1, bondingtype2, bondingtype3*)

Bases: `intermol.forces.abstract_type.AbstractType`

**bondingtype1**

**bondingtype2**

**placeholder**

##### intermol.forces.abstract\_3\_virtual\_type module

**class** `intermol.forces.abstract_3_virtual_type.Abstract3VirtualType` (*bondingtype1, bondingtype2, bondingtype3, bondingtype4*)

Bases: `intermol.forces.abstract_type.AbstractType`

**bondingtype1**

**bondingtype2**

**bondingtype3**

**bondingtype4**

**placeholder**

**intermol.forces.abstract\_4\_virtual\_type module**

```
class intermol.forces.abstract_4_virtual_type.Abstract4VirtualType (bondingtype1,  
bonding-  
type2, bond-  
ingtype3,  
bonding-  
type4, bond-  
ingtype5)
```

```
Bases: intermol.forces.abstract_type.AbstractType
```

```
bondingtype1
```

```
bondingtype2
```

```
bondingtype3
```

```
bondingtype4
```

**intermol.forces.abstract\_angle\_type module**

```
class intermol.forces.abstract_angle_type.AbstractAngleType (bondingtype1, bonding-  
type2, bondingtype3,  
c=False)
```

```
Bases: intermol.forces.abstract_type.AbstractType
```

```
bondingtype1
```

```
bondingtype2
```

```
bondingtype3
```

```
c
```

**intermol.forces.abstract\_atom\_type module**

```
class intermol.forces.abstract_atom_type.AbstractAtomType (atomtype, bondtype=None,  
atomic_number=None,  
mass=None, charge=None,  
ptype=None)
```

```
Bases: intermol.forces.abstract_type.AbstractType
```

```
atomic_number
```

```
atomtype
```

```
bondtype
```

```
charge
```

```
mass
```

```
ptype
```

**intermol.forces.abstract\_bond\_type module**

```
class intermol.forces.abstract_bond_type.AbstractBondType (bondingtype1, bonding-  
type2, order=1, c=False)
```

```
Bases: intermol.forces.abstract_type.AbstractType
```

```
bondingtype1
```

```
bondingtype2
```

```
c
```

**order**

**intermol.forces.abstract\_dihedral\_type** module

```
class intermol.forces.abstract_dihedral_type.AbstractDihedralType (bondingtype1,
bondingtype2,
bondingtype3,
bonding-
type4, improper=False)
```

Bases: `intermol.forces.abstract_type.AbstractType`

**bondingtype1**

**bondingtype2**

**bondingtype3**

**bondingtype4**

**improper**

**intermol.forces.abstract\_nonbonded\_type** module

```
class intermol.forces.abstract_nonbonded_type.AbstractNonbondedType (atom1,
atom2,
type)
```

Bases: `intermol.forces.abstract_type.AbstractType`

**atom1**

**atom2**

**type**

**intermol.forces.abstract\_pair\_type** module

```
class intermol.forces.abstract_pair_type.AbstractPairType (bondingtype1, bonding-
type2, scaleLJ=None,
scaleQQ=None,
long=False)
```

Bases: `intermol.forces.abstract_type.AbstractType`

**bondingtype1**

**bondingtype2**

**long**

**scaleLJ**

**scaleQQ**

**intermol.forces.abstract\_type** module

```
class intermol.forces.abstract_type.AbstractType
```

Bases: `object`

```
__repr__ ()
```

Print the object and all of its non-magic attributes.

**intermol.forces.atom\_c\_type module**

**class** `intermol.forces.atom_c_type.AtomCType` (*\*args, \*\*kws*)  
Bases: `intermol.forces.abstract_atom_type.AbstractAtomType`

**intermol.forces.atom\_sigeps\_type module**

**class** `intermol.forces.atom_sigeps_type.AtomSigepsType` (*\*args, \*\*kws*)  
Bases: `intermol.forces.abstract_atom_type.AbstractAtomType`

**intermol.forces.buckingham\_nonbonded\_type module**

**class** `intermol.forces.buckingham_nonbonded_type.BuckinghamNonbonded` (*atom1, atom2, bonding-type1=None, bonding-type2=None, a=Quantity(value=0.0, unit=kilojoule/mole), b=Quantity(value=0.0, unit=/nanometer), C6=Quantity(value=0.0, unit=nanometer\*\*6\*kilojoule/mole), type=False*)  
Bases: `intermol.forces.buckingham_nonbonded_type.BuckinghamNonbondedType`

stub documentation

**class** `intermol.forces.buckingham_nonbonded_type.BuckinghamNonbondedType` (*\*args, \*\*kws*)  
Bases: `intermol.forces.abstract_nonbonded_type.AbstractNonbondedType`

**C6**

**a**

**b**

**type**

**intermol.forces.connection\_bond\_type module**

**class** `intermol.forces.connection_bond_type.ConnectionBond` (*atom1, atom2, bonding-type1=None, bonding-type2=None, order=1, c=False*)  
Bases: `intermol.forces.connection_bond_type.ConnectionBondType`

stub documentation

**class** `intermol.forces.connection_bond_type.ConnectionBondType` (*\*args, \*\*kws*)  
Bases: `intermol.forces.abstract_bond_type.AbstractBondType`

**c**

**order**

**intermol.forces.convert\_dihedrals module**

`intermol.forces.convert_dihedrals.convert_dihedral_from_OPLS_to_RB` (*f*)

`intermol.forces.convert_dihedrals.convert_dihedral_from_RB_to_OPLS` (*c*)

`intermol.forces.convert_dihedrals.convert_dihedral_from_RB_to_trig` (*c*)

```
intermol.forces.convert_dihedrals.convert_dihedral_from_fourier_to_trig(f)
intermol.forces.convert_dihedrals.convert_dihedral_from_proper_to_trig(p)
intermol.forces.convert_dihedrals.convert_dihedral_from_trig_to_RB(fcs)
intermol.forces.convert_dihedrals.convert_dihedral_from_trig_to_fourier(fcs)
intermol.forces.convert_dihedrals.convert_dihedral_from_trig_to_proper(fcs,
                                                                    con-
                                                                    ven-
                                                                    tion='0')
```

intermol.forces.convert\_dihedrals.**convert\_nothing**(*x*)  
useful utility for not converting anything

**intermol.forces.cosine\_angle\_type module**

```
class intermol.forces.cosine_angle_type.CosineAngle(atom1, atom2, atom3, bond-
                                                    ingtype1=None, bonding-
                                                    type2=None, bondingtype3=None,
                                                    k=Quantity(value=0.0,
                                                    unit=kilojoule/mole), c=False)
```

Bases: intermol.forces.cosine\_angle\_type.CosineAngleType

[http://lammps.sandia.gov/doc/angle\\_cosine.html](http://lammps.sandia.gov/doc/angle_cosine.html)

```
class intermol.forces.cosine_angle_type.CosineAngleType(*args, **kwds)
Bases: intermol.forces.abstract_angle_type.AbstractAngleType
```

**c**

**k**

**intermol.forces.cosine\_squared\_angle\_type module**

```
class intermol.forces.cosine_squared_angle_type.CosineSquaredAngle(atom1,
                                                                    atom2,
                                                                    atom3,
                                                                    bonding-
                                                                    type1=None,
                                                                    bonding-
                                                                    type2=None,
                                                                    bonding-
                                                                    type3=None,
                                                                    theta=Quantity(value=0.0,
                                                                    unit=degree),
                                                                    k=Quantity(value=0.0,
                                                                    unit=kilojoule/mole),
                                                                    c=False)
```

Bases: intermol.forces.cosine\_squared\_angle\_type.CosineSquaredAngleType

stub documentation

```
class intermol.forces.cosine_squared_angle_type.CosineSquaredAngleType(*args,
                                                                    **kwds)
Bases: intermol.forces.abstract_angle_type.AbstractAngleType
```

**c**

**k**

**theta**

**intermol.forces.cross\_bond\_angle\_angle\_type module**

```
class intermol.forces.cross_bond_angle_angle_type.CrossBondAngleAngle (atom1,  
                                                                    atom2,  
                                                                    atom3,  
                                                                    bonding-  
                                                                    type1=None,  
                                                                    bonding-  
                                                                    type2=None,  
                                                                    bonding-  
                                                                    type3=None,  
                                                                    r1=Quantity(value=0.0,  
                                                                    unit=nanometer),  
                                                                    r2=Quantity(value=0.0,  
                                                                    unit=nanometer),  
                                                                    r3=Quantity(value=0.0,  
                                                                    unit=nanometer),  
                                                                    k=Quantity(value=0.0,  
                                                                    unit=kilojoule/(nanometer**2*mole),  
                                                                    c=False)
```

Bases: `intermol.forces.cross_bond_angle_angle_type.CrossBondAngleAngleType`

stub documentation

```
class intermol.forces.cross_bond_angle_angle_type.CrossBondAngleAngleType (*args,  
                                                                    **kwargs)
```

Bases: `intermol.forces.abstract_angle_type.AbstractAngleType`

**c**

**k**

**r1**

**r2**

**r3**

**intermol.forces.cross\_bond\_bond\_angle\_type module**

```
class intermol.forces.cross_bond_bond_angle_type.CrossBondBondAngle (atom1,  
                                                                    atom2,  
                                                                    atom3,  
                                                                    bonding-  
                                                                    type1=None,  
                                                                    bonding-  
                                                                    type2=None,  
                                                                    bonding-  
                                                                    type3=None,  
                                                                    r1=Quantity(value=0.0,  
                                                                    unit=nanometer),  
                                                                    r2=Quantity(value=0.0,  
                                                                    unit=nanometer),  
                                                                    k=Quantity(value=0.0,  
                                                                    unit=kilojoule/(nanometer**2*mole),  
                                                                    c=False)
```

Bases: `intermol.forces.cross_bond_bond_angle_type.CrossBondBondAngleType`

stub documentation

```

class intermol.forces.cross_bond_bond_angle_type.CrossBondBondAngleType (*args,
                                                                    **kwds)
    Bases: intermol.forces.abstract_angle_type.AbstractAngleType
    c
    k
    r1
    r2

```

#### intermol.forces.cubic\_bond\_type module

```

class intermol.forces.cubic_bond_type.CubicBond (atom1,          atom2,          bonding-
                                                    type1=None,          bondingtype2=None,
                                                    length=Quantity(value=0.0,
                                                            unit=nanometer),
                                                    C2=Quantity(value=0.0,
                                                       unit=kilojoule/(nanometer**2*mole)),
                                                    C3=Quantity(value=0.0,
                                                       unit=kilojoule/(nanometer**3*mole)),
                                                    order=1, c=False)
    Bases: intermol.forces.cubic_bond_type.CubicBondType

```

stub documentation

```

class intermol.forces.cubic_bond_type.CubicBondType (*args, **kwds)
    Bases: intermol.forces.abstract_bond_type.AbstractBondType
    C2
    C3
    c
    length
    order

```

#### intermol.forces.fene\_bond\_type module

```

class intermol.forces.fene_bond_type.FeneBond (atom1, atom2, bondingtype1=None, bond-
                                                    ingtype2=None, length=Quantity(value=0.0,
                                                            unit=nanometer), kb=Quantity(value=0.0,
                                                            unit=kilojoule/(nanometer**2*mole)),
                                                    order=1, c=False)
    Bases: intermol.forces.fene_bond_type.FeneBondType

```

stub documentation

```

class intermol.forces.fene_bond_type.FeneBondType (*args, **kwds)
    Bases: intermol.forces.abstract_bond_type.AbstractBondType
    c
    kb
    length
    order

```

**intermol.forces.fene\_expandable\_bond\_type module**

```
class intermol.forces.fene_expandable_bond_type.FeneExpandableBond (atom1,  
atom2,  
bonding-  
type1=None,  
bonding-  
type2=None,  
k=Quantity(value=0.0,  
unit=kilojoule/(nanometer**2*mole)),  
length=Quantity(value=0.0,  
unit=nanometer),  
ep-  
silon=Quantity(value=0.0,  
unit=kilojoule/mole),  
sigma=Quantity(value=0.0,  
unit=nanometer),  
delta=Quantity(value=0.0,  
unit=nanometer),  
order=1,  
c=False)
```

Bases: `intermol.forces.fene_expandable_bond_type.FeneExpandableBondType`

stub documentation

```
class intermol.forces.fene_expandable_bond_type.FeneExpandableBondType (*args,  
**kwds)
```

Bases: `intermol.forces.abstract_bond_type.AbstractBondType`

**c**

**delta**

**epsilon**

**k**

**length**

**order**

**sigma**

**intermol.forces.forcedata module****intermol.forces.forcefunctions module**

```
intermol.forces.forcefunctions.build_paramlist (program)
```

Create a paramlist specific for a given program.

```
intermol.forces.forcefunctions.build_unitvars (program, paramlist, dumself=None)
```

Takes a string program name (one of the supported programs), and a 'self' object it looks like the keyword is not being used, but it is used in the line eval(unit). The test name 'dumself' needs to match what is in the force data arrays. Currently only used for lammmps.

```
intermol.forces.forcefunctions.capifyname (forcename)
```

Return name of the class in camelCase.

```
intermol.forces.forcefunctions.create_kwd_dict (unitvars, paramlist, force_type_object,  
values, optvalues=None)
```



```
intermol.forces.forcefunctions.create_kwds_from_entries(unitvars, paramlist, entries, force_type, offset=0)
```

Create a keyword dictionary given an array of information from a file format

requires the master set of units, the master set of parameter lists, an object (either a force\_class or force\_type), the list of information to be converted into a keyword, and an offset.

**Parameters** *offset* (*int*) – how far over from the first entry we translate

```
intermol.forces.forcefunctions.get_parameter_kwds_from_force(force, forceparams, paramlist)
```

```
intermol.forces.forcefunctions.get_parameter_list_from_force(force, paramlist)
```

Create a function that returns the parameters of a function type.

First, we need make some additions to the parameter list dictionary, which we do once when the forcedata script is imported. Useful to put the forces here as well. We won't make this a function for now since it's needed in this module.

```
intermol.forces.forcefunctions.get_parameter_list_from_kwds(force, kwds, paramlist)
```

```
intermol.forces.forcefunctions.optforceparams(force_type, forcetype_object=None)
```

Return the dictionary of optional parameters of an abstract force type.

If no object is given, we fill with blanks.

```
intermol.forces.forcefunctions.optparamkeylookup(force_type)
```

Given a force\_type object, determine the key associated with the optional parameters.

```
intermol.forces.forcefunctions.optparamlookup(force_type_object, object_default=False)
```

A wrapper for optforceparams that takes a force\_type object and returns the optional parameter dictionary.

```
intermol.forces.forcefunctions.specify(program_units, unitset, dumself=None, shouldEval=True)
```

Takes the dict of units, and a set of dimensions and replaces the dimensions with the appropriate units.

### intermol.forces.four\_fdn\_virtual\_type module

```
class intermol.forces.four_fdn_virtual_type.FourFdnVirtual(atom1, atom2, atom3, atom4, atom5, bondingtype1=None, bondingtype2=None, bondingtype3=None, bondingtype4=None, bondingtype5=None, a=Quantity(value=0.0, unit=dimensionless), b=Quantity(value=0.0, unit=dimensionless), c=Quantity(value=0.0, unit=nanometer), placeholder=False)
```

Bases: `intermol.forces.four_fdn_virtual_type.FourFdnVirtualType`

stub documentation

```
class intermol.forces.four_fdn_virtual_type.FourFdnVirtualType(*args, **kwargs)
```

Bases: `intermol.forces.abstract_4_virtual_type.Abstract4VirtualType`

**a**

**b**

**c**

**placeholder**

### intermol.forces.fourier\_dihedral\_type module

```
class intermol.forces.fourier_dihedral_type.FourierDihedral (atom1, atom2, atom3,
                                                            atom4,          bonding-
                                                            type1=None,    bond-
                                                            ingtype2=None,
                                                            bondingtype3=None,
                                                            bondingtype4=None,
                                                            c1=Quantity(value=0.0,
                                                            unit=kilojoule/mole),
                                                            c2=Quantity(value=0.0,
                                                            unit=kilojoule/mole),
                                                            c3=Quantity(value=0.0,
                                                            unit=kilojoule/mole),
                                                            c4=Quantity(value=0.0,
                                                            unit=kilojoule/mole),
                                                            c5=Quantity(value=0.0,
                                                            unit=kilojoule/mole),
                                                            improper=False)
```

Bases: `intermol.forces.fourier_dihedral_type.FourierDihedralType`

stub documentation

```
class intermol.forces.fourier_dihedral_type.FourierDihedralType (*args, **kwds)
```

Bases: `intermol.forces.abstract_dihedral_type.AbstractDihedralType`

**c1**

**c2**

**c3**

**c4**

**c5**

**improper**

### intermol.forces.g96\_bond\_type module

```
class intermol.forces.g96_bond_type.G96Bond (atom1, atom2, bondingtype1=None, bond-
                                             ingtype2=None, length=Quantity(value=0.0,
                                             unit=nanometer), k=Quantity(value=0.0,
                                             unit=kilojoule/(nanometer**4*mole)), order=1,
                                             c=False)
```

Bases: `intermol.forces.g96_bond_type.G96BondType`

stub documentation

```
class intermol.forces.g96_bond_type.G96BondType (*args, **kwds)
```

Bases: `intermol.forces.abstract_bond_type.AbstractBondType`

**c**

**k**

**length**

**order**

**intermol.forces.harmonic\_angle\_type module**

```
class intermol.forces.harmonic_angle_type.HarmonicAngle (atom1, atom2, atom3,
                                                    bondingtype1=None,
                                                    bondingtype2=None,
                                                    bondingtype3=None,
                                                    theta=Quantity(value=0.0,
                                                    unit=degree),
                                                    k=Quantity(value=0.0,
                                                    unit=kilojoule/(mole*radian**2)),
                                                    c=False)
```

Bases: `intermol.forces.harmonic_angle_type.HarmonicAngleType`

stub documentation

```
class intermol.forces.harmonic_angle_type.HarmonicAngleType (*args, **kwds)
```

Bases: `intermol.forces.abstract_angle_type.AbstractAngleType`

**c**

**k**

**theta**

**intermol.forces.harmonic\_bond\_type module**

```
class intermol.forces.harmonic_bond_type.HarmonicBond (atom1, atom2, bond-
                                                    ingtype1=None, bond-
                                                    ingtype2=None,
                                                    length=Quantity(value=0.0,
                                                    unit=nanometer),
                                                    k=Quantity(value=0.0,
                                                    unit=kilojoule/(nanometer**2*mole)),
                                                    order=1, c=False)
```

Bases: `intermol.forces.harmonic_bond_type.HarmonicBondType`

stub documentation

```
class intermol.forces.harmonic_bond_type.HarmonicBondType (*args, **kwds)
```

Bases: `intermol.forces.abstract_bond_type.AbstractBondType`

**c**

**k**

**length**

**order**

**intermol.forces.harmonic\_potential\_bond\_type module**

```
class intermol.forces.harmonic_potential_bond_type.HarmonicPotentialBond (atom1,  
                                                                    atom2,  
                                                                    bond-  
                                                                    ing-  
                                                                    type1=None,  
                                                                    bond-  
                                                                    ing-  
                                                                    type2=None,  
                                                                    length=Quantity(value=0.0,  
                                                                    unit=nanometer),  
                                                                    k=Quantity(value=0.0,  
                                                                    unit=kilojoule/(nanometer**2,  
                                                                    or-  
                                                                    der=1,  
                                                                    c=False))
```

```
    Bases: intermol.forces.harmonic_potential_bond_type.HarmonicPotentialBondType
```

```
    stub documentation
```

```
class intermol.forces.harmonic_potential_bond_type.HarmonicPotentialBondType (*args,  
                                                                    **kwds)
```

```
    Bases: intermol.forces.abstract_bond_type.AbstractBondType
```

```
    c
```

```
    k
```

```
    length
```

```
    order
```

### intermol.forces.improper\_harmonic\_dihedral\_type module

```
class intermol.forces.improper_harmonic_dihedral_type.ImproperHarmonicDihedral (atom1,  
                                                                    atom2,  
                                                                    atom3,  
                                                                    atom4,  
                                                                    bond-  
                                                                    ing-  
                                                                    type1=None,  
                                                                    bond-  
                                                                    ing-  
                                                                    type2=None,  
                                                                    bond-  
                                                                    ing-  
                                                                    type3=None,  
                                                                    bond-  
                                                                    ing-  
                                                                    type4=None,  
                                                                    xi=Quantity(value=0,  
                                                                    unit=degree),  
                                                                    k=Quantity(value=0,  
                                                                    unit=kilojoule/(mole*degree**2,  
                                                                    im-  
                                                                    proper=False))
```

```
    Bases: intermol.forces.improper_harmonic_dihedral_type.ImproperHarmonicDihedralType
```

```
    stub documentation
```

```

class intermol.forces.improper_harmonic_dihedral_type.ImproperHarmonicDihedralType (*args,
                                                                                   **kwds)
    Bases: intermol.forces.abstract_dihedral_type.AbstractDihedralType
    improper
    k
    xi

```

#### intermol.forces.lj\_c\_nonbonded\_type module

```

class intermol.forces.lj_c_nonbonded_type.LjCNonbonded (atom1,      atom2,      bond-
                                                         ingtype1=None,      bond-
                                                         ingtype2=None,
                                                         C6=Quantity(value=0.0,
                                                         unit=nanometer**6*kilojoule/mole),
                                                         C12=Quantity(value=0.0,
                                                         unit=nanometer**12*kilojoule/mole),
                                                         type=False)
    Bases: intermol.forces.lj_c_nonbonded_type.LjCNonbondedType

```

stub documentation

```

class intermol.forces.lj_c_nonbonded_type.LjCNonbondedType (*args, **kwds)
    Bases: intermol.forces.abstract_nonbonded_type.AbstractNonbondedType
    C12
    C6
    type

```

#### intermol.forces.lj\_c\_pair\_type module

```

class intermol.forces.lj_c_pair_type.LjCPair (atom1, atom2, bondingtype1=None, bond-
                                                         ingtype2=None,      C6=Quantity(value=0.0,
                                                         unit=nanometer**6*kilojoule/mole),
                                                         C12=Quantity(value=0.0,
                                                         unit=nanometer**12*kilojoule/mole),
                                                         scaleLJ=None, scaleQQ=None, long=False)
    Bases: intermol.forces.lj_c_pair_type.LjCPairType

```

stub documentation

```

class intermol.forces.lj_c_pair_type.LjCPairType (*args, **kwds)
    Bases: intermol.forces.abstract_pair_type.AbstractPairType
    C12
    C6
    long
    scaleLJ
    scaleQQ

```

#### intermol.forces.lj\_default\_pair\_type module

```
class intermol.forces.lj_default_pair_type.LjDefaultPair(atom1, atom2, bonding-  
type1=None, bonding-  
type2=None, scaleLJ=None,  
scaleQQ=None,  
long=False)
```

Bases: `intermol.forces.lj_default_pair_type.LjDefaultPairType`

stub documentation

```
class intermol.forces.lj_default_pair_type.LjDefaultPairType(*args,**kwds)
```

Bases: `intermol.forces.abstract_pair_type.AbstractPairType`

**long**

**scaleLJ**

**scaleQQ**

### **intermol.forces.lj\_sigeps\_nonbonded\_type** module

```
class intermol.forces.lj_sigeps_nonbonded_type.LjSigepsNonbonded(atom1, atom2,  
bonding-  
type1=None,  
bonding-  
type2=None,  
sigma=Quantity(value=0.0,  
unit=nanometer),  
ep-  
silon=Quantity(value=0.0,  
unit=kilojoule/mole),  
type=False)
```

Bases: `intermol.forces.lj_sigeps_nonbonded_type.LjSigepsNonbondedType`

stub documentation

```
class intermol.forces.lj_sigeps_nonbonded_type.LjSigepsNonbondedType(*args,  
**kwds)
```

Bases: `intermol.forces.abstract_nonbonded_type.AbstractNonbondedType`

**epsilon**

**sigma**

**type**

### **intermol.forces.lj\_sigeps\_pair\_type** module

```
class intermol.forces.lj_sigeps_pair_type.LjSigepsPair(atom1, atom2, bond-  
ingtype1=None, bond-  
ingtype2=None,  
sigma=Quantity(value=0.0,  
unit=nanometer), ep-  
silon=Quantity(value=0.0,  
unit=kilojoule/mole),  
scaleLJ=None,  
scaleQQ=None, long=False)
```

Bases: `intermol.forces.lj_sigeps_pair_type.LjSigepsPairType`

stub documentation

```
class intermol.forces.lj_sigeps_pair_type.LjSigepsPairType(*args,**kwds)
```

Bases: `intermol.forces.abstract_pair_type.AbstractPairType`

**epsilon**

**long**  
**scaleLJ**  
**scaleQQ**  
**sigma**

### intermol.forces.ljq\_c\_pair\_type module

```

class intermol.forces.ljq_c_pair_type.LjqCPair (atom1, atom2, bondingtype1=None, bondingtype2=None, qi=Quantity(value=0.0, unit=elementary charge), qj=Quantity(value=0.0, unit=elementary charge), C6=Quantity(value=0.0, unit=nanometer**6*kilojoule/mole), C12=Quantity(value=0.0, unit=nanometer**12*kilojoule/mole), scaleLJ=None, scaleQQ=None, long=False)

```

Bases: `intermol.forces.ljq_c_pair_type.LjqCPairType`

stub documentation

```

class intermol.forces.ljq_c_pair_type.LjqCPairType (*args, **kwds)
Bases: intermol.forces.abstract_pair_type.AbstractPairType

```

**C12**  
**C6**  
**long**  
**qi**  
**qj**  
**scaleLJ**  
**scaleQQ**

### intermol.forces.ljq\_default\_pair\_type module

```

class intermol.forces.ljq_default_pair_type.LjqDefaultPair (atom1, atom2, bondingtype1=None, bondingtype2=None, scaleLJ=None, scaleQQ=None, long=False)

```

Bases: `intermol.forces.ljq_default_pair_type.LjqDefaultPairType`

stub documentation

```

class intermol.forces.ljq_default_pair_type.LjqDefaultPairType (*args, **kwds)
Bases: intermol.forces.abstract_pair_type.AbstractPairType

```

**long**  
**scaleLJ**  
**scaleQQ**

**intermol.forces.ljq\_sigeps\_pair\_type module**

```
class intermol.forces.ljq_sigeps_pair_type.LjqSigepsPair (atom1, atom2, bondingtype1=None, bondingtype2=None, qi=Quantity(value=0.0, unit=elementary charge), qj=Quantity(value=0.0, unit=elementary charge), sigma=Quantity(value=0.0, unit=nanometer), epsilon=Quantity(value=0.0, unit=kilojoule/mole), scaleLJ=None, scaleQQ=None, long=False)
```

Bases: `intermol.forces.ljq_sigeps_pair_type.LjqSigepsPairType`

stub documentation

```
class intermol.forces.ljq_sigeps_pair_type.LjqSigepsPairType (*args, **kwds)
```

Bases: `intermol.forces.abstract_pair_type.AbstractPairType`

**epsilon**

**long**

**qi**

**qj**

**scaleLJ**

**scaleQQ**

**sigma**

**intermol.forces.make\_forces module****intermol.forces.morse\_bond\_type module**

```
class intermol.forces.morse_bond_type.MorseBond (atom1, atom2, bondingtype1=None, bondingtype2=None, length=Quantity(value=0.0, unit=nanometer), D=Quantity(value=0.0, unit=kilojoule/mole), beta=Quantity(value=0.0, unit=nanometer), order=1, c=False)
```

Bases: `intermol.forces.morse_bond_type.MorseBondType`

stub documentation

```
class intermol.forces.morse_bond_type.MorseBondType (*args, **kwds)
```

Bases: `intermol.forces.abstract_bond_type.AbstractBondType`

**D**

**beta**

**c**

**length**

**order**



**intermol.forces.nonlinear\_bond\_type module**

```
class intermol.forces.nonlinear_bond_type.NonlinearBond (atom1, atom2, bond-
ingtype1=None, bond-
ingtype2=None, ep-
silon=Quantity(value=0.0,
unit=kilojoule/mole),
r0=Quantity(value=0.0,
unit=nanometer),
lamda=Quantity(value=0.0,
unit=nanometer), order=1,
c=False)
```

Bases: `intermol.forces.nonlinear_bond_type.NonlinearBondType`

[http://lammps.sandia.gov/doc/bond\\_nonlinear.html](http://lammps.sandia.gov/doc/bond_nonlinear.html)

```
class intermol.forces.nonlinear_bond_type.NonlinearBondType (*args, **kwds)
```

Bases: `intermol.forces.abstract_bond_type.AbstractBondType`

**c**

**epsilon**

**lamda**

**order**

**r0**

**intermol.forces.proper\_periodic\_dihedral\_type module**

```
class intermol.forces.proper_periodic_dihedral_type.ProperPeriodicDihedral (atom1,
atom2,
atom3,
atom4,
bond-
ing-
type1=None,
bond-
ing-
type2=None,
bond-
ing-
type3=None,
bond-
ing-
type4=None,
phi=Quantity(value=0.0,
unit=degree),
k=Quantity(value=0.0,
unit=kilojoule/mole),
mul-
ti-
plic-
ity=Quantity(value=0.0,
unit=dimensionless),
weight=Quantity(value=0
unit=dimensionless),
im-
proper=False)
```

Bases: `intermol.forces.proper_periodic_dihedral_type.ProperPeriodicDihedralType`  
stub documentation

**class** `intermol.forces.proper_periodic_dihedral_type.ProperPeriodicDihedralType` (*\*args*,  
*\*\*kws*)

Bases: `intermol.forces.abstract_dihedral_type.AbstractDihedralType`

**improper**

**k**

**multiplicity**

**phi**

**weight**

#### **intermol.forces.quartic\_angle\_type module**

**class** `intermol.forces.quartic_angle_type.QuarticAngle` (*atom1*, *atom2*, *atom3*,  
*bondingtype1=None*,  
*bondingtype2=None*,  
*bondingtype3=None*,  
*theta=Quantity(value=0.0,*  
*unit=degree)*,  
*C0=Quantity(value=0.0,*  
*unit=kilojoule/mole)*,  
*C1=Quantity(value=0.0,*  
*unit=kilojoule/(mole\*radian))*,  
*C2=Quantity(value=0.0,*  
*unit=kilojoule/(mole\*radian\*\*2))*,  
*C3=Quantity(value=0.0,*  
*unit=kilojoule/(mole\*radian\*\*3))*,  
*C4=Quantity(value=0.0,*  
*unit=kilojoule/(mole\*radian\*\*4))*,  
*c=False*)

Bases: `intermol.forces.quartic_angle_type.QuarticAngleType`

stub documentation

**class** `intermol.forces.quartic_angle_type.QuarticAngleType` (*\*args*, *\*\*kws*)

Bases: `intermol.forces.abstract_angle_type.AbstractAngleType`

**C0**

**C1**

**C2**

**C3**

**C4**

**c**

**theta**

#### **intermol.forces.quartic\_bond\_type module**

```
class intermol.forces.quartic_bond_type.QuarticBond (atom1, atom2, bonding-
    type1=None, bondingtype2=None,
    length=Quantity(value=0.0,
    unit=nanometer),
    C2=Quantity(value=0.0,
    unit=kilojoule/(nanometer**2*mole)),
    C3=Quantity(value=0.0,
    unit=kilojoule/(nanometer**3*mole)),
    C4=Quantity(value=0.0,
    unit=kilojoule/(nanometer**4*mole)),
    order=1, c=False)
```

Bases: `intermol.forces.quartic_bond_type.QuarticBondType`

stub documentation

```
class intermol.forces.quartic_bond_type.QuarticBondType (*args, **kwds)
```

Bases: `intermol.forces.abstract_bond_type.AbstractBondType`

**C2**

**C3**

**C4**

**c**

**length**

**order**

### intermol.forces.quartic\_breakable\_bond\_type module

```
class intermol.forces.quartic_breakable_bond_type.QuarticBreakableBond (atom1,
    atom2,
    bond-
    ing-
    type1=None,
    bond-
    ing-
    type2=None,
    k=Quantity(value=0.0,
    unit=kilojoule/(nanometer**4*mole)),
    B1=Quantity(value=0.0,
    unit=nanometer),
    B2=Quantity(value=0.0,
    unit=nanometer),
    Rc=Quantity(value=0.0,
    unit=nanometer),
    U0=Quantity(value=0.0,
    unit=kilojoule/mole),
    or-
    der=1,
    c=False)
```

Bases: `intermol.forces.quartic_breakable_bond_type.QuarticBreakableBondType`

[http://lammmps.sandia.gov/doc/bond\\_quartic.html](http://lammmps.sandia.gov/doc/bond_quartic.html)

```
class intermol.forces.quartic_breakable_bond_type.QuarticBreakableBondType (*args,
    **kwds)
```

Bases: `intermol.forces.abstract_bond_type.AbstractBondType`

**B1**

**B2**  
**Rc**  
**U0**  
**c**  
**k**  
**order**

#### **intermol.forces.rb\_dihedral\_type module**

```
class intermol.forces.rb_dihedral_type.RbDihedral (atom1, atom2, atom3, atom4, bonding-  
type1=None, bondingtype2=None,  
bondingtype3=None, bonding-  
type4=None, C0=Quantity(value=0.0,  
unit=kilojoule/mole),  
C1=Quantity(value=0.0,  
unit=kilojoule/mole),  
C2=Quantity(value=0.0,  
unit=kilojoule/mole),  
C3=Quantity(value=0.0,  
unit=kilojoule/mole),  
C4=Quantity(value=0.0,  
unit=kilojoule/mole),  
C5=Quantity(value=0.0,  
unit=kilojoule/mole),  
C6=Quantity(value=0.0,  
unit=kilojoule/mole), im-  
proper=False)
```

Bases: `intermol.forces.rb_dihedral_type.RbDihedralType`

stub documentation

```
class intermol.forces.rb_dihedral_type.RbDihedralType (*args, **kwds)  
Bases: intermol.forces.abstract_dihedral_type.AbstractDihedralType
```

**C0**  
**C1**  
**C2**  
**C3**  
**C4**  
**C5**  
**C6**  
**improper**

#### **intermol.forces.settles module**

```
class intermol.forces.settles.Settles (*args, **kwds)  
Bases: intermol.forces.abstract_type.AbstractType
```

**intermol.forces.three\_fad\_virtual\_type module**

```
class intermol.forces.three_fad_virtual_type.ThreeFadVirtual (atom1, atom2, atom3,
                                                             atom4,      bonding-
                                                             type1=None,  bond-
                                                             ingtype2=None,
                                                             bondingtype3=None,
                                                             bondingtype4=None,
                                                             theta=Quantity(value=0.0,
                                                             unit=degree),
                                                             d=Quantity(value=0.0,
                                                             unit=nanometer),
                                                             placeholder=False)
```

Bases: `intermol.forces.three_fad_virtual_type.ThreeFadVirtualType`

stub documentation

```
class intermol.forces.three_fad_virtual_type.ThreeFadVirtualType (*args, **kws)
```

Bases: `intermol.forces.abstract_3_virtual_type.Abstract3VirtualType`

**d**

**placeholder**

**theta**

**intermol.forces.three\_fd\_virtual\_type module**

```
class intermol.forces.three_fd_virtual_type.ThreeFdVirtual (atom1, atom2, atom3,
                                                             atom4,      bonding-
                                                             type1=None,  bond-
                                                             ingtype2=None,
                                                             bondingtype3=None,
                                                             bondingtype4=None,
                                                             a=Quantity(value=0.0,
                                                             unit=dimensionless),
                                                             d=Quantity(value=0.0,
                                                             unit=nanometer), place-
                                                             holder=False)
```

Bases: `intermol.forces.three_fd_virtual_type.ThreeFdVirtualType`

stub documentation

```
class intermol.forces.three_fd_virtual_type.ThreeFdVirtualType (*args, **kws)
```

Bases: `intermol.forces.abstract_3_virtual_type.Abstract3VirtualType`

**a**

**d**

**placeholder**

**intermol.forces.three\_linear\_virtual\_type module**

```
class intermol.forces.three_linear_virtual_type.ThreeLinearVirtual (atom1,  
                                                                    atom2,  
                                                                    atom3,  
                                                                    atom4,  
                                                                    bonding-  
                                                                    type1=None,  
                                                                    bonding-  
                                                                    type2=None,  
                                                                    bonding-  
                                                                    type3=None,  
                                                                    bonding-  
                                                                    type4=None,  
                                                                    a=Quantity(value=0.0,  
                                                                    unit=dimensionless),  
                                                                    b=Quantity(value=0.0,  
                                                                    unit=dimensionless),  
                                                                    place-  
                                                                    holder=False)
```

```
    Bases: intermol.forces.three_linear_virtual_type.ThreeLinearVirtualType
```

```
    stub documentation
```

```
class intermol.forces.three_linear_virtual_type.ThreeLinearVirtualType (*args,  
                                                                    **kwds)
```

```
    Bases: intermol.forces.abstract_3_virtual_type.Abstract3VirtualType
```

```
    a
```

```
    b
```

```
    placeholder
```

#### intermol.forces.three\_out\_virtual\_type module

```
class intermol.forces.three_out_virtual_type.ThreeOutVirtual (atom1, atom2, atom3,  
                                                                    atom4,      bonding-  
                                                                    type1=None,  bond-  
                                                                    ingtype2=None,  
                                                                    bondingtype3=None,  
                                                                    bondingtype4=None,  
                                                                    a=Quantity(value=0.0,  
                                                                    unit=dimensionless),  
                                                                    b=Quantity(value=0.0,  
                                                                    unit=dimensionless),  
                                                                    c=Quantity(value=0.0,  
                                                                    unit=/nanometer),  
                                                                    placeholder=False)
```

```
    Bases: intermol.forces.three_out_virtual_type.ThreeOutVirtualType
```

```
    stub documentation
```

```
class intermol.forces.three_out_virtual_type.ThreeOutVirtualType (*args, **kwds)
```

```
    Bases: intermol.forces.abstract_3_virtual_type.Abstract3VirtualType
```

```
    a
```

```
    b
```

```
    c
```

```
    placeholder
```

**intermol.forces.trig\_dihedral\_type module**

```
class intermol.forces.trig_dihedral_type.TrigDihedral (atom1,      atom2,      atom3,
                                                    atom4,      bondingtype1=None,
                                                    bondingtype2=None,
                                                    bondingtype3=None,
                                                    bondingtype4=None,
                                                    phi=Quantity(value=0.0,
                                                         unit=degree),
                                                    fc0=Quantity(value=0.0,
                                                         unit=kilojoule/mole),
                                                    fc1=Quantity(value=0.0,
                                                         unit=kilojoule/mole),
                                                    fc2=Quantity(value=0.0,
                                                         unit=kilojoule/mole),
                                                    fc3=Quantity(value=0.0,
                                                         unit=kilojoule/mole),
                                                    fc4=Quantity(value=0.0,
                                                         unit=kilojoule/mole),
                                                    fc5=Quantity(value=0.0,
                                                         unit=kilojoule/mole),
                                                    fc6=Quantity(value=0.0,
                                                         unit=kilojoule/mole),      im-
                                                    improper=False)
```

Bases: `intermol.forces.trig_dihedral_type.TrigDihedralType`

stub documentation

```
class intermol.forces.trig_dihedral_type.TrigDihedralType (*args, **kwds)
```

Bases: `intermol.forces.abstract_dihedral_type.AbstractDihedralType`

**fc0**

**fc1**

**fc2**

**fc3**

**fc4**

**fc5**

**fc6**

**improper**

**phi**

**intermol.forces.two\_virtual\_type module**

```
class intermol.forces.two_virtual_type.TwoVirtual (atom1,      atom2,      atom3,
                                                    bondingtype1=None,      bond-
                                                    bondingtype2=None,      bonding-
                                                    type3=None,      a=Quantity(value=0.0,
                                                         unit=dimensionless),      place-
                                                    holder=False)
```

Bases: `intermol.forces.two_virtual_type.TwoVirtualType`

stub documentation

```
class intermol.forces.two_virtual_type.TwoVirtualType (*args, **kwds)
```

Bases: `intermol.forces.abstract_2_virtual_type.Abstract2VirtualType`

**a**

**placeholder**

### **intermol.forces.urey\_bradley\_angle\_type module**

```
class intermol.forces.urey_bradley_angle_type.UreyBradleyAngle (atom1, atom2, atom3, bonding-type1=None, bonding-type2=None, bonding-type3=None, theta=Quantity(value=0.0, unit=degree), k=Quantity(value=0.0, unit=kilojoule/(mole*radian**2)), r=Quantity(value=0.0, unit=nanometer), kUB=Quantity(value=0.0, unit=kilojoule/(nanometer**2*mole)), c=False)
```

Bases: `intermol.forces.urey_bradley_angle_type.UreyBradleyAngleType`

stub documentation

```
class intermol.forces.urey_bradley_angle_type.UreyBradleyAngleType (*args, **kwds)
```

Bases: `intermol.forces.abstract_angle_type.AbstractAngleType`

**c**

**k**

**kUB**

**r**

**theta**

### **Module contents**

#### **intermol.gromacs package**

##### **Submodules**

### **intermol.gromacs.grofile\_parser module**

```
class intermol.gromacs.grofile_parser.GromacsGroParser (gro_file)
```

Bases: `object`

GromacsGroParser reads and writes Gromacs .gro files

A .gro file also contains some topological information, such as elements and residue names, but not enough to construct a full Topology object. This information is recorded and stored in the object's public fields.

**read()**

**write** (*system*)

Write the system out in a Gromacs 4.6 format

**Parameters** **filename** (*str*) – the file to write out to



**intermol.gromacs.gromacs\_driver module**

`intermol.gromacs.gromacs_driver.gromacs_energies` (*top=None, gro=None, mdp=None, gropath=None, grosuff=None, grompp\_check=False*)  
     *gropath* = path to gromacs binaries *grosuff* = suffix of gromacs binaries, usually “” or ‘\_d’  
`intermol.gromacs.gromacs_driver.read_file` (*top\_in, gro\_in, gropath*)  
`intermol.gromacs.gromacs_driver.write_file` (*system, top\_out, gro\_out*)

**intermol.gromacs.gromacs\_parser module**

**class** `intermol.gromacs.gromacs_parser.GromacsParser` (*top\_file, gro\_file, system=None, include\_dir=None, defines=None*)

Bases: `object`

A class containing methods required to read in a Gromacs(4.5.4) Topology File

**class TopMoleculeType**

Bases: `object`

Inner class to store information about a molecule type.

`GromacsParser.canonical_angle` (*params, angle, direction='into'*)

**Parameters**

- **params** –
- **angle** –
- **direction** –

Returns:

`GromacsParser.canonical_bond` (*params, bond, direction='into'*)

**Parameters**

- **params** –
- **bond** –
- **direction** –

Returns:

`GromacsParser.canonical_dihedral` (*params, dihedral, direction='into'*)

We can fit everything into two types of dihedrals - `dihedral_trig`, and improper harmonic. Dihedral trig is of the form

$$fc0 + \sum_{i=1}^6 fci (\cos(nx-\phi))$$

Proper dihedrals can be stored easily in this form, since they have only 1 n. Improper dihedrals can as well (flag as improper). RB can be stored as well, assuming  $\phi = 0$  or 180. Fourier can also be stored. A full dihedral trig can be decomposed into multiple proper dihedrals.

Will need to handle multiple dihedrals little differently in that we will need to add multiple 9 dihedrals together into a single `dihedral_trig`, as long as they have the same  $\phi$  angle (seems to be always the case).

**Parameters**

- **params** –
- **dihedral** –
- **direction** –

Returns:

`GromacsParser.choose_parameter_kwds_from_forces` (*entries, n\_atoms, force\_type, gromacs\_force*)

Extract a force's parameters into a keyword dictionary.

#### Parameters

- **entries** (*str*) – The *split()* line being parsed.
- **n\_atoms** (*int*) – The number of atoms in the force.
- **force\_type** – The type of the force.
- **gromacs\_force** – The

#### Returns

**kwds** – The force's parameters, e.g.

```
{'length': Quantity(value=0.13, unit=nanometers), 'k': ...
}
```

**Return type** dict

`GromacsParser.create_angle` (*angle*)

`GromacsParser.create_atom` (*temp\_atom*)

`GromacsParser.create_bond` (*bond*)

`GromacsParser.create_dihedral` (*dihedral*)

Create a dihedral object based on a [ *dihedrals* ] entry.

`GromacsParser.create_exclusion` (*exclusion*)

`GromacsParser.create_kwds_from_entries` (*entries, force\_class, offset=0*)

`GromacsParser.create_molecule` (*top\_moltype, mol\_name*)

`GromacsParser.create_moleculetype` (*top\_moltype, mol\_name, mol\_count*)

`GromacsParser.create_pair` (*pair*)

Create a pair force object based on a [ *pairs* ] entry

`GromacsParser.create_settle` (*settle*)

`GromacsParser.directive_before_moleculetype` ()

`GromacsParser.find_dihedralttype` (*bondingtypes, improper*)

Determine the type of dihedral interaction between four atoms.

`GromacsParser.find_forcetype` (*bondingtypes, types\_of\_kind*)

`GromacsParser.get_parameter_kwds_from_force` (*force*)

`GromacsParser.get_parameter_list_from_force` (*force*)

`GromacsParser.gromacs_angle_types` = {'1': <class 'intermol.forces.harmonic\_angle\_type.HarmonicAngleType'>

`GromacsParser.gromacs_angles` = {'1': <class 'intermol.forces.harmonic\_angle\_type.HarmonicAngle'>, '3': <class 'intermol.forces.harmonic\_angle\_type.HarmonicAngle'>

`GromacsParser.gromacs_bond_types` = {'1': <class 'intermol.forces.harmonic\_bond\_type.HarmonicBondType'>, '3': <class 'intermol.forces.harmonic\_bond\_type.HarmonicBondType'>

`GromacsParser.gromacs_bonds` = {'1': <class 'intermol.forces.harmonic\_bond\_type.HarmonicBond'>, '3': <class 'intermol.forces.harmonic\_bond\_type.HarmonicBond'>

`GromacsParser.gromacs_combination_rules` = {'1': 'Multiply-C6C12', '3': 'Multiply-Sigeps', '2': 'Lorentz-Berendsen'}

`GromacsParser.gromacs_dihedral_types` = {'Trig': <class 'intermol.forces.trig\_dihedral\_type.TrigDihedralType'>

GromacsParser.**gromacs\_dihedrals** = {'Trig': <class 'intermol.forces.trig\_dihedral\_type.TrigDihedral'>, '1': <class 'intermol.forces.trig\_dihedral\_type.TrigDihedral'>}

GromacsParser.**gromacs\_pair\_types** = {'1A': <class 'intermol.forces.lj\_c\_pair\_type.LjCPairType'>, '1C': <class 'intermol.forces.lj\_c\_pair\_type.LjCPairType'>}

GromacsParser.**gromacs\_pairs** = {'1A': <class 'intermol.forces.lj\_c\_pair\_type.LjCPair'>, '1C': <class 'intermol.forces.lj\_c\_pair\_type.LjCPair'>}

GromacsParser.**invalid\_line** (*line*)

GromacsParser.**lookup\_atom\_atomtype** (*index, state=0*)

GromacsParser.**lookup\_atom\_bondingtype** (*index*)

GromacsParser.**lookup\_gromacs\_angles** = {<class 'intermol.forces.cross\_bond\_bond\_angle\_type.CrossBondBondAngle'>: '7', <class 'intermol.forces.fene\_bond\_type.FeneBond'>: '7'}

GromacsParser.**lookup\_gromacs\_bonds** = {<class 'intermol.forces.fene\_bond\_type.FeneBond'>: '7', <class 'intermol.forces.fene\_bond\_type.FeneBond'>: '7'}

GromacsParser.**lookup\_gromacs\_combination\_rules** = {'Multiply-Sigeps': '3', 'Lorentz-Berthelot': '2', 'Multiply-Sigeps': '3'}

GromacsParser.**lookup\_gromacs\_dihedrals** = {<class 'intermol.forces.fourier\_dihedral\_type.FourierDihedral'>: '7', <class 'intermol.forces.fourier\_dihedral\_type.FourierDihedral'>: '7'}

GromacsParser.**lookup\_gromacs\_pairs** = {<class 'intermol.forces.ljq\_sigeps\_pair\_type.LjqSigepsPair'>: '2B', <class 'intermol.forces.ljq\_sigeps\_pair\_type.LjqSigepsPair'>: '2B'}

GromacsParser.**paramlist** = {'QuarticBreakableBond': ['k', 'B1', 'B2', 'Rc', 'U0'], 'fene\_expandable\_bond': ['k', 'B1', 'B2', 'Rc', 'U0']}

GromacsParser.**process\_angle** (*line*)  
Process a line in the [ angles ] category.

GromacsParser.**process\_angletype** (*line*)  
Process a line in the [ angletypes ] category.

GromacsParser.**process\_atom** (*line*)  
Process a line in the [ atoms ] category.

GromacsParser.**process\_atomtype** (*line*)  
Process a line in the [ atomtypes ] category.

GromacsParser.**process\_bond** (*line*)  
Process a line in the [ bonds ] category.

GromacsParser.**process\_bondtype** (*line*)  
Process a line in the [ bondtypes ] category.

GromacsParser.**process\_cmap** (*line*)  
Process a line in the [ cmaps ] category.

GromacsParser.**process\_cmatype** (*line*)  
Process a line in the [ cmatypes ] category.

GromacsParser.**process\_defaults** (*line*)  
Process the [ defaults ] line.

GromacsParser.**process\_dihedral** (*line*)  
Process a line in the [ dihedrals ] category.

GromacsParser.**process\_dihedralttype** (*line*)  
Process a line in the [ dihedraltypes ] category.

GromacsParser.**process\_exclusion** (*line*)  
Process a line in the [ exclusions ] category.

GromacsParser.**process\_file** (*top\_file*)

GromacsParser.**process\_forcetype** (*bondingtypes, forcename, line, n\_atoms, gromacs\_force\_types, canonical\_force*)

GromacsParser.**process\_implicittype** (*line*)  
Process a line in the [ implicit\_genborn\_params ] category.

GromacsParser.**process\_line** (*top\_file*, *line*)

Process one line from a file.

GromacsParser.**process\_molecule** (*line*)

Process a line in the [ molecules ] category.

GromacsParser.**process\_moleculetype** (*line*)

Process a line in the [ moleculetypes ] category.

GromacsParser.**process\_nonbond\_params** (*line*)

Process a line in the [ nonbond\_param ] category.

GromacsParser.**process\_pair** (*line*)

Process a line in the [ pairs ] category.

GromacsParser.**process\_pairtype** (*line*)

Process a line in the [ pairtypes ] category.

GromacsParser.**process\_settle** (*line*)

Process a line in the [ settles ] category.

GromacsParser.**read** ()

**Returns** system

GromacsParser.**too\_few\_fields** (*line*)

**static** GromacsParser.**type\_parameters\_are\_unique** (*a*, *b*)

Check if two force types are unique.

Currently only tests TrigDihedralType and ImproperHarmonicDihedralType because these are the only two forcetypes that we currently allow to have multiple values for the same set of 4 atom bondingtypes.

GromacsParser.**unitvars** = {"QuarticBreakableBond": [Unit({BaseUnit(base\_dim=BaseDimension("length"), name=

GromacsParser.**write** ()

Write this topology in GROMACS file format.

**Parameters** filename – the name of the file to write out to

GromacsParser.**write\_angles** (*top*)

GromacsParser.**write\_atoms** (*top*)

GromacsParser.**write\_atomtypes** (*top*)

GromacsParser.**write\_bonds** (*top*)

GromacsParser.**write\_defaults** (*top*)

GromacsParser.**write\_dihedrals** (*top*)

GromacsParser.**write\_exclusions** (*top*)

GromacsParser.**write\_molecules** (*top*)

GromacsParser.**write\_moleculetypes** (*top*)

GromacsParser.**write\_nonbonded\_types** (*top*)

GromacsParser.**write\_pairs** (*top*)

GromacsParser.**write\_settles** (*top*)

GromacsParser.**write\_system** (*top*)

`intermol.gromacs.gromacs_parser.default_gromacs_include_dir()`

Find the location where gromacs #include files are referenced from, by searching for (1) gromacs environment variables, (2) just using the default gromacs install location, `/usr/local/gromacs/share/gromacs/top`.

`intermol.gromacs.gromacs_parser.load_gromacs(top_file, gro_file, include_dir=None, defines=None)`

Load a set of GROMACS input files into a *System*.

#### Parameters

- `top_file` –
- `gro_file` –
- `include_dir` –
- `defines` –

#### Returns

**Return type** `system`

`intermol.gromacs.gromacs_parser.write_gromacs(top_file, gro_file, system)`

Load a set of GROMACS input files into a *System*.

#### Parameters

- `top_file` –
- `gro_file` –
- `include_dir` –
- `defines` –

#### Returns

**Return type** `system`

## Module contents

### intermol.lammps package

#### Submodules

#### intermol.lammps.lammps\_driver module

`intermol.lammps.lammps_driver.lammps_energies(input_file, lmppath='lmp_openmpi')`

Evaluate energies of LAMMPS files

#### Parameters

- = path to input file (expects data file in same folder) (*input\_file*) –
- = path to LAMMPS binaries (*lmppath*) –

`intermol.lammps.lammps_driver.read_file(in_file)`

`intermol.lammps.lammps_driver.write_file(in_file, system, unit_set='real')`

#### intermol.lammps.lammps\_parser module

`class intermol.lammps.lammps_parser.LammpsParser(in_file, system=None, unit_set='real')`

Bases: `object`

A class containing methods to read and write LAMMPS files.

**SCALE\_FROM = 0.5**

**SCALE\_INT0 = 2.0**

**canonical\_angle** (*kwds, angle, direction*)

Convert from the canonical form of this interaction.

**canonical\_bond** (*kwds, bond, direction='into'*)

Convert to/from the canonical form of this interaction.

**canonical\_dihedral** (*params, dihedral, direction='into'*)

Convert from the canonical form of this interaction.

**create\_kwds\_from\_entries** (*entries, force\_class, offset=0*)

**get\_force\_atoms** (*force, forceclass*)

Return the atoms involved in a force.

**get\_force\_bondingtypes** (*force, forceclass*)

Return the atoms involved in a force.

**get\_parameter\_kwds\_from\_force** (*force*)

**get\_parameter\_list\_from\_force** (*force*)

**lammps\_angle\_types** = {'charmm': <class 'intermol.forces.urey\_bradley\_angle\_type.UreyBradleyAngleType'>, 'cosine': <class 'intermol.forces.cosine\_angle\_type.CosineAngleType'>, 'fourier': <class 'intermol.forces.fourier\_dihedral\_type.FourierDihedralType'>, 'harmonic': <class 'intermol.forces.harmonic\_bond\_type.HarmonicBondType'>, 'improper': <class 'intermol.forces.improper\_harmonic\_dihedral\_type.ImproperHarmonicDihedralType'>, 'nonlinear': <class 'intermol.forces.nonlinear\_bond\_type.NonlinearBondType'>, 'quartic': <class 'intermol.forces.quartic\_bond\_type.QuarticBondType'>, 'trig': <class 'intermol.forces.trig\_dihedral\_type.TrigDihedralType'>, 'vff': <class 'intermol.forces.vff\_type.VffType'>}

**lammps\_angles** = {'charmm': <class 'intermol.forces.urey\_bradley\_angle\_type.UreyBradleyAngle'>, 'cosine': <class 'intermol.forces.cosine\_angle\_type.CosineAngle'>, 'fourier': <class 'intermol.forces.fourier\_dihedral\_type.FourierDihedral'>, 'harmonic': <class 'intermol.forces.harmonic\_bond\_type.HarmonicBond'>, 'improper': <class 'intermol.forces.improper\_harmonic\_dihedral\_type.ImproperHarmonicDihedral'>, 'nonlinear': <class 'intermol.forces.nonlinear\_bond\_type.NonlinearBond'>, 'quartic': <class 'intermol.forces.quartic\_bond\_type.QuarticBond'>, 'trig': <class 'intermol.forces.trig\_dihedral\_type.TrigDihedral'>, 'vff': <class 'intermol.forces.vff\_type.Vff'>}

**lammps\_bond\_types** = {'nonlinear': <class 'intermol.forces.nonlinear\_bond\_type.NonlinearBondType'>, 'quartic': <class 'intermol.forces.quartic\_bond\_type.QuarticBondType'>, 'harmonic': <class 'intermol.forces.harmonic\_bond\_type.HarmonicBondType'>, 'improper': <class 'intermol.forces.improper\_harmonic\_dihedral\_type.ImproperHarmonicDihedralType'>, 'trig': <class 'intermol.forces.trig\_dihedral\_type.TrigDihedralType'>, 'vff': <class 'intermol.forces.vff\_type.VffType'>}

**lammps\_bonds** = {'nonlinear': <class 'intermol.forces.nonlinear\_bond\_type.NonlinearBond'>, 'quartic': <class 'intermol.forces.quartic\_bond\_type.QuarticBond'>, 'harmonic': <class 'intermol.forces.harmonic\_bond\_type.HarmonicBond'>, 'improper': <class 'intermol.forces.improper\_harmonic\_dihedral\_type.ImproperHarmonicDihedral'>, 'trig': <class 'intermol.forces.trig\_dihedral\_type.TrigDihedral'>, 'vff': <class 'intermol.forces.vff\_type.Vff'>}

**lammps\_dihedral\_types** = {'charmm': <class 'intermol.forces.proper\_periodic\_dihedral\_type.ProperPeriodicDihedralType'>, 'fourier': <class 'intermol.forces.fourier\_dihedral\_type.FourierDihedralType'>, 'trig': <class 'intermol.forces.trig\_dihedral\_type.TrigDihedralType'>, 'vff': <class 'intermol.forces.vff\_type.VffType'>}

**lammps\_dihedrals** = {'charmm': <class 'intermol.forces.proper\_periodic\_dihedral\_type.ProperPeriodicDihedral'>, 'fourier': <class 'intermol.forces.fourier\_dihedral\_type.FourierDihedral'>, 'trig': <class 'intermol.forces.trig\_dihedral\_type.TrigDihedral'>, 'vff': <class 'intermol.forces.vff\_type.Vff'>}

**lammps\_improper\_types** = {'cvff': <class 'intermol.forces.trig\_dihedral\_type.TrigDihedralType'>, 'harmonic': <class 'intermol.forces.improper\_harmonic\_dihedral\_type.ImproperHarmonicDihedralType'>, 'vff': <class 'intermol.forces.vff\_type.VffType'>}

**lammps\_impropers** = {'cvff': <class 'intermol.forces.trig\_dihedral\_type.TrigDihedral'>, 'harmonic': <class 'intermol.forces.improper\_harmonic\_dihedral\_type.ImproperHarmonicDihedral'>, 'vff': <class 'intermol.forces.vff\_type.Vff'>}

**lookup\_lammps\_angles** = {<class 'intermol.forces.cosine\_angle\_type.CosineAngle'>: 'cosine', <class 'intermol.forces.fourier\_dihedral\_type.FourierDihedral'>: 'fourier', <class 'intermol.forces.harmonic\_bond\_type.HarmonicBond'>: 'harmonic', <class 'intermol.forces.improper\_harmonic\_dihedral\_type.ImproperHarmonicDihedral'>: 'improper', <class 'intermol.forces.nonlinear\_bond\_type.NonlinearBond'>: 'nonlinear', <class 'intermol.forces.quartic\_bond\_type.QuarticBond'>: 'quartic', <class 'intermol.forces.trig\_dihedral\_type.TrigDihedral'>: 'trig', <class 'intermol.forces.vff\_type.Vff'>: 'vff'}

**lookup\_lammps\_bonds** = {<class 'intermol.forces.harmonic\_bond\_type.HarmonicBond'>: 'harmonic', <class 'intermol.forces.improper\_harmonic\_dihedral\_type.ImproperHarmonicDihedral'>: 'improper', <class 'intermol.forces.nonlinear\_bond\_type.NonlinearBond'>: 'nonlinear', <class 'intermol.forces.quartic\_bond\_type.QuarticBond'>: 'quartic', <class 'intermol.forces.trig\_dihedral\_type.TrigDihedral'>: 'trig', <class 'intermol.forces.vff\_type.Vff'>: 'vff'}

**lookup\_lammps\_dihedrals** = {<class 'intermol.forces.fourier\_dihedral\_type.FourierDihedral'>: 'opls', <class 'intermol.forces.trig\_dihedral\_type.TrigDihedral'>: 'trig', <class 'intermol.forces.vff\_type.Vff'>: 'vff'}

**lookup\_lammps\_impropers** = {<class 'intermol.forces.improper\_harmonic\_dihedral\_type.ImproperHarmonicDihedral'>: 'improper', <class 'intermol.forces.trig\_dihedral\_type.TrigDihedral'>: 'trig', <class 'intermol.forces.vff\_type.Vff'>: 'vff'}

**parse\_angle\_coeffs** (*data\_lines*)

**parse\_angle\_style** (*line*)

**parse\_angles** (*data\_lines*)

**parse\_atom\_style** (*line*)

---

**Note:** Assuming 'full' as default for everything else.

---

**parse\_atoms** (*data\_lines*)

Read atoms from data file.

**parse\_bond\_coeffs** (*data\_lines*)

**parse\_bond\_style** (*line*)

**parse\_bonded\_style** (*line*)

**parse\_bonds** (*data\_lines*)

**parse\_boundary** (*line*)

**parse\_box** (*line, dim*)

Read box information from data file.

**Parameters**

- **line** (*str*) – Current line in input file.
- **dim** (*int*) – Dimension specified in line.

**parse\_dihedral\_coeffs** (*data\_lines*)

**parse\_dihedral\_style** (*line*)

**parse\_dihedrals** (*data\_lines*)

**parse\_dimension** (*line*)

**parse\_force** (*data\_lines, force\_classes, forceSet, n=0*)

Read bonds, angles, dihedrals, impropers from data file.

**parse\_force\_coeffs** (*data\_lines, force\_name, force\_classes, force\_style, lammmps\_forces, canonical\_force*)

Read force coefficients from data file.

**parse\_improper\_coeffs** (*data\_lines*)

**parse\_improper\_style** (*line*)

**parse\_impropers** (*data\_lines*)

**parse\_kspace\_style** (*line*)

---

**Note:** Currently ignored.

---

**parse\_masses** (*data\_lines*)

Read masses from data file.

**parse\_pair\_coeffs** (*data\_lines*)

Read pair coefficients from data file.

**parse\_pair\_modify** (*line*)

**parse\_pair\_style** (*line*)

**parse\_read\_data** (*line*)

**parse\_special\_bonds** (*line*)

**parse\_units** (*line*)

**parse\_velocities** (*data\_lines*)

**read** ()

Reads a LAMMPS input file and a data file specified within.

**Parameters** **input\_file** (*str*) – Name of LAMMPS input file to read in.

**read\_data** (*data\_file*)

Reads a LAMMPS data file.

**Parameters** **data\_file** (*str*) – name of LAMMPS data file to read in.

**read\_input** ()

Reads a LAMMPS input file.

**Parameters** **input\_file** (*str*) – Name of LAMMPS input file to read in.

**set\_units** (*unit\_set*)

Set what unit set to use.

**write** (*unit\_set='real'*)

Writes a LAMMPS data and corresponding input file.

**Parameters**

- **data\_file** (*str*) – Name of LAMMPS data file to write to.
- **unit\_set** (*str*) – LAMMPS unit set for output file.

**write\_angles** (*mol\_type, offset*)

**write\_bonds** (*mol\_type, offset*)

**write\_dihedrals** (*mol\_type, offset*)

Separate dihedrals from impropers.

**write\_forces** (*forces, offset, force\_name, lookup\_lammps\_force, lammps\_force\_types, canonical\_force*)

The general force writing function.

Currently supports bonds, angles, dihedrals, impropers.

**write\_impropers** (*mol\_type, offset*)

Separate dihedrals from impropers.

**write\_virtuals** (*mol\_type, offset*)

`intermol.lammps.lammps_parser.load_lammps` (*in\_file*)

Load a LAMMPS input file into a *System*.

**Parameters**

- **in\_file** –
- **include\_dir** –
- **defines** –

**Returns**

**Return type** `system`

`intermol.lammps.lammps_parser.write_lammps` (*in\_file, system, unit\_set='real'*)

Load a LAMMPS input file into a *System*.

**Parameters**

- **in\_file** –
- **include\_dir** –
- **defines** –

**Returns**

**Return type** `system`



## Module contents

### 3.1.2 Submodules

#### intermol.atom module

**class** `intermol.atom.Atom` (*index, name=None, residue\_index=-1, residue\_name=None*)

Bases: `object`

**atomic\_number**

**atomtype**

**bondingtype**

**cgnr**

**charge**

**epsilon**

**force**

Return the force on the atom

**index**

**mass**

**name**

**position**

Return the cartesian coordinates of the atom

**ptype**

**residue\_index**

**residue\_name**

**sigma**

**velocity**

Return the velocity of the atom

#### intermol.convert module

`intermol.convert.find_match` (*key, dictionary, unit*)

Helper function for *summarize\_energy\_results*.

`intermol.convert.get_diff` (*e\_in, e\_out*)

Returns difference in potential energy.

##### Parameters

- - **dictionary of energy groups from input file** (*e\_in*) –
- - **dictionary of energy groups from output file** (*e\_out*) –

**Returns** potential energy difference in units of the input

`intermol.convert.main` (*args=None*)

`intermol.convert.parse_args` (*args*)

`intermol.convert.summarize_energy_results` (*energy\_input*, *energy\_outputs*, *input\_type*, *output\_types*)

Creates a table comparing input and output energy groups.

**Args:** *energy\_input* (dict): energy groups from input file *energy\_output*(list): containing dictionary of energy groups or -1 for each output file

*input\_type* (str): input engine *output\_types* (list): containing output formats

**Returns:** *out* (list of strings): which forms a summary table using “  
”.join(*out*)

## intermol.decorators module

**exception** `intermol.decorators.UnitsException` (*value*)

Bases: `exceptions.Exception`

Exception denoting that an argument has the incorrect units.

**exception** `intermol.decorators.ValueException` (*value*)

Bases: `exceptions.Exception`

Exception denoting that an argument has the incorrect value.

`intermol.decorators.accepts` (*\*types*)

Decorator for class methods that should accept only specified types.

EXAMPLE

```
@accepts(float, int) def function(a, b):  
    return b*a
```

`intermol.decorators.accepts_compatible_units` (*\*units*, *\*\*unitdict*)

Decorator for class methods that should accept only arguments compatible with specified units.

Each argument of the function will be matched with an argument of `@acceptunits`. Those arguments of the function that correspond `@acceptunits` which are not `None` will be checked to ensure they are compatible with the specified units.

EXAMPLE

```
@acceptunits(units.meter, None, units.kilocalories_per_mole) def function(a, b, c):  
    pass function(1.0 *  
    units.angstrom, 3, 1.0 * units.kilojoules_per_mole)
```

`intermol.decorators.returns` (*rtype*)

Decorator for functions that should only return specific types. EXAMPLE

```
@returns(int) def function(): return 7
```

## intermol.molecule module

**class** `intermol.molecule.Molecule` (*name=None*)

Bases: `object`

An abstract molecule object.

**add\_atom** (*atom*)

Add an atom

**Parameters** `atom` (*Atom*) – the atom to add into the molecule

**atoms**

Return an orderedset of atoms.

### intermol.moleculetype module

**class** `intermol.moleculetype.MoleculeType` (*name=None*)

Bases: `object`

An abstract container for molecules of one type.

**add\_molecule** (*molecule*)

Add a molecule into the moleculetype container

**Parameters** `molecule` (*Molecule*) – the molecule to append

### intermol.orderedset module

**class** `intermol.orderedset.OrderedSet` (*iterable=()*)

Bases: `_abcoll.Set`

**add** (*key*)

**difference\_update** (*\*args, \*\*kwargs*)

**discard** (*key*)

**intersection\_update** (*\*args, \*\*kwargs*)

### intermol.system module

**class** `intermol.system.System` (*name=None*)

Bases: `object`

**add\_atomtype** (*atomtype*)

**add\_molecule** (*molecule*)

Append a molecule into the System. :param molecule: The molecule object to be appended

**add\_molecule\_type** (*molecule\_type*)

Append a molecule\_type into the System. :param molecule\_type: The MoleculeType object to be appended

**atoms**

**atomtypes**

**box\_vector**

Return the box vector.

**molecule\_types**

**n\_atoms**

**nonbonded\_types**

### 3.1.3 Module contents

## 3.2 intermol.gromacs package

### 3.2.1 Submodules

#### intermol.gromacs.grofile\_parser module

**class** `intermol.gromacs.grofile_parser.GromacsGroParser` (*gro\_file*)

Bases: object

GromacsGroParser reads and writes Gromacs .gro files

A .gro file also contains some topological information, such as elements and residue names, but not enough to construct a full Topology object. This information is recorded and stored in the object's public fields.

**read** ()

**write** (*system*)

Write the system out in a Gromacs 4.6 format

**Parameters** `filename` (*str*) – the file to write out to

#### intermol.gromacs.gromacs\_driver module

`intermol.gromacs.gromacs_driver.gromacs_energies` (*top=None, gro=None, mdp=None, gropath=None, grosuff=None, grompp\_check=False*)

`gropath` = path to gromacs binaries `grosuff` = suffix of gromacs binaries, usually ‘\_’ or ‘\_d’

`intermol.gromacs.gromacs_driver.read_file` (*top\_in, gro\_in, gropath*)

`intermol.gromacs.gromacs_driver.write_file` (*system, top\_out, gro\_out*)

#### intermol.gromacs.gromacs\_parser module

**class** `intermol.gromacs.gromacs_parser.GromacsParser` (*top\_file, gro\_file, system=None, include\_dir=None, defines=None*)

Bases: object

A class containing methods required to read in a Gromacs(4.5.4) Topology File

**class** `TopMoleculeType`

Bases: object

Inner class to store information about a molecule type.

`GromacsParser.canonical_angle` (*params, angle, direction='into'*)

**Parameters**

- `params` –
- `angle` –
- `direction` –

Returns:

`GromacsParser.canonical_bond` (*params, bond, direction='into'*)

**Parameters**

- **params** –
- **bond** –
- **direction** –

Returns:

GromacsParser.**canonical\_dihedral** (*params, dihedral, direction='into'*)

We can fit everything into two types of dihedrals - dihedral\_trig, and improper harmonic. Dihedral trig is of the form

$$fc0 + \sum_{i=1}^6 fci (\cos(nx-\phi))$$

Proper dihedrals can be stored easily in this form, since they have only 1 n. Improper dihedrals can as well (flag as improper). RB can be stored as well, assuming  $\phi = 0$  or  $180$ . Fourier can also be stored. A full dihedral trig can be decomposed into multiple proper dihedrals.

Will need to handle multiple dihedrals little differently in that we will need to add multiple 9 dihedrals together into a single dihedral\_trig, as long as they have the same  $\phi$  angle (seems to be always the case).

**Parameters**

- **params** –
- **dihedral** –
- **direction** –

Returns:

GromacsParser.**choose\_parameter\_kwds\_from\_forces** (*entries, n\_atoms, force\_type, gromacs\_force*)

Extract a force's parameters into a keyword dictionary.

**Parameters**

- **entries** (*str*) – The *split()* line being parsed.
- **n\_atoms** (*int*) – The number of atoms in the force.
- **force\_type** – The type of the force.
- **gromacs\_force** – The

**Returns**

**kwds** – The force's parameters, e.g.

```
{'length': Quantity(value=0.13, unit=nanometers), 'k': ...
}
```

**Return type** dict

GromacsParser.**create\_angle** (*angle*)

GromacsParser.**create\_atom** (*temp\_atom*)

GromacsParser.**create\_bond** (*bond*)

GromacsParser.**create\_dihedral** (*dihedral*)

Create a dihedral object based on a [ dihedrals ] entry.

GromacsParser.**create\_exclusion** (*exclusion*)

GromacsParser.**create\_kwds\_from\_entries** (*entries, force\_class, offset=0*)

`GromacsParser.create_molecule` (*top\_moltype, mol\_name*)

`GromacsParser.create_moleculetype` (*top\_moltype, mol\_name, mol\_count*)

`GromacsParser.create_pair` (*pair*)  
Create a pair force object based on a [ pairs ] entry

`GromacsParser.create_settle` (*settle*)

`GromacsParser.directive_before_moleculetype` ()

`GromacsParser.find_dihedralttype` (*bondingtypes, improper*)  
Determine the type of dihedral interaction between four atoms.

`GromacsParser.find_forcetype` (*bondingtypes, types\_of\_kind*)

`GromacsParser.get_parameter_kwds_from_force` (*force*)

`GromacsParser.get_parameter_list_from_force` (*force*)

`GromacsParser.gromacs_angle_types` = {'1': <class 'intermol.forces.harmonic\_angle\_type.HarmonicAngleType'>, ...}

`GromacsParser.gromacs_angles` = {'1': <class 'intermol.forces.harmonic\_angle\_type.HarmonicAngle'>, '3': <class 'intermol.forces.harmonic\_angle\_type.HarmonicAngle'>, ...}

`GromacsParser.gromacs_bond_types` = {'1': <class 'intermol.forces.harmonic\_bond\_type.HarmonicBondType'>, ...}

`GromacsParser.gromacs_bonds` = {'1': <class 'intermol.forces.harmonic\_bond\_type.HarmonicBond'>, '3': <class 'intermol.forces.harmonic\_bond\_type.HarmonicBond'>, ...}

`GromacsParser.gromacs_combination_rules` = {'1': 'Multiply-C6C12', '3': 'Multiply-Sigeps', '2': 'Lorentz-Berthelot', ...}

`GromacsParser.gromacs_dihedral_types` = {'Trig': <class 'intermol.forces.trig\_dihedral\_type.TrigDihedralType'>, ...}

`GromacsParser.gromacs_dihedrals` = {'Trig': <class 'intermol.forces.trig\_dihedral\_type.TrigDihedral'>, '1': <class 'intermol.forces.trig\_dihedral\_type.TrigDihedral'>, ...}

`GromacsParser.gromacs_pair_types` = {'1A': <class 'intermol.forces.lj\_c\_pair\_type.LjCPairType'>, '1C': <class 'intermol.forces.lj\_c\_pair\_type.LjCPairType'>, ...}

`GromacsParser.gromacs_pairs` = {'1A': <class 'intermol.forces.lj\_c\_pair\_type.LjCPair'>, '1C': <class 'intermol.forces.lj\_c\_pair\_type.LjCPair'>, ...}

`GromacsParser.invalid_line` (*line*)

`GromacsParser.lookup_atom_atomtype` (*index, state=0*)

`GromacsParser.lookup_atom_bondingtype` (*index*)

`GromacsParser.lookup_gromacs_angles` = {<class 'intermol.forces.cross\_bond\_bond\_angle\_type.CrossBondBondAngleType'>, ...}

`GromacsParser.lookup_gromacs_bonds` = {<class 'intermol.forces.fene\_bond\_type.FeneBond'>: '7', <class 'intermol.forces.fene\_bond\_type.FeneBond'>: '7', ...}

`GromacsParser.lookup_gromacs_combination_rules` = {'Multiply-Sigeps': '3', 'Lorentz-Berthelot': '2', 'Multiply-C6C12': '1', ...}

`GromacsParser.lookup_gromacs_dihedrals` = {<class 'intermol.forces.fourier\_dihedral\_type.FourierDihedral'>: '4', <class 'intermol.forces.fourier\_dihedral\_type.FourierDihedral'>: '4', ...}

`GromacsParser.lookup_gromacs_pairs` = {<class 'intermol.forces.ljq\_sigeps\_pair\_type.LjqSigepsPair'>: '2B', <class 'intermol.forces.ljq\_sigeps\_pair\_type.LjqSigepsPair'>: '2B', ...}

`GromacsParser.paramlist` = {'QuarticBreakableBond': ['k', 'B1', 'B2', 'Rc', 'U0'], 'fene\_expandable\_bond': ['k', 'B1', 'B2', 'Rc', 'U0'], ...}

`GromacsParser.process_angle` (*line*)  
Process a line in the [ angles ] category.

`GromacsParser.process_angletype` (*line*)  
Process a line in the [ angletypes ] category.

`GromacsParser.process_atom` (*line*)  
Process a line in the [ atoms ] category.

`GromacsParser.process_atomtype` (*line*)  
Process a line in the [ atomtypes ] category.

GromacsParser.**process\_bond** (*line*)

Process a line in the [ bonds ] category.

GromacsParser.**process\_bondtype** (*line*)

Process a line in the [ bondtypes ] category.

GromacsParser.**process\_cmap** (*line*)

Process a line in the [ cmaps ] category.

GromacsParser.**process\_cmaptypes** (*line*)

Process a line in the [ cmaptypes ] category.

GromacsParser.**process\_defaults** (*line*)

Process the [ defaults ] line.

GromacsParser.**process\_dihedral** (*line*)

Process a line in the [ dihedrals ] category.

GromacsParser.**process\_dihedraltypes** (*line*)

Process a line in the [ dihedraltypes ] category.

GromacsParser.**process\_exclusion** (*line*)

Process a line in the [ exclusions ] category.

GromacsParser.**process\_file** (*top\_file*)

GromacsParser.**process\_forcetype** (*bondingtypes*, *forcename*, *line*, *n\_atoms*, *gromacs\_force\_types*, *canonical\_force*)

GromacsParser.**process\_implicittype** (*line*)

Process a line in the [ implicit\_genborn\_params ] category.

GromacsParser.**process\_line** (*top\_file*, *line*)

Process one line from a file.

GromacsParser.**process\_molecule** (*line*)

Process a line in the [ molecules ] category.

GromacsParser.**process\_moleculetypes** (*line*)

Process a line in the [ moleculetypes ] category.

GromacsParser.**process\_nonbond\_params** (*line*)

Process a line in the [ nonbond\_param ] category.

GromacsParser.**process\_pair** (*line*)

Process a line in the [ pairs ] category.

GromacsParser.**process\_pairtypes** (*line*)

Process a line in the [ pairtypes ] category.

GromacsParser.**process\_settle** (*line*)

Process a line in the [ settles ] category.

GromacsParser.**read** ()

**Returns** system

GromacsParser.**too\_few\_fields** (*line*)

**static** GromacsParser.**type\_parameters\_are\_unique** (*a*, *b*)

Check if two force types are unique.

Currently only tests TrigDihedralType and ImproperHarmonicDihedralType because these are the only two forcetypes that we currently allow to have multiple values for the same set of 4 atom bondingtypes.

GromacsParser.**unitvars** = {'QuarticBreakableBond': [Unit({BaseUnit(base\_dim=BaseDimension("length"), name=

GromacsParser.**write**()

Write this topology in GROMACS file format.

**Parameters** `filename` – the name of the file to write out to

GromacsParser.**write\_angles** (*top*)

GromacsParser.**write\_atoms** (*top*)

GromacsParser.**write\_atomtypes** (*top*)

GromacsParser.**write\_bonds** (*top*)

GromacsParser.**write\_defaults** (*top*)

GromacsParser.**write\_dihedrals** (*top*)

GromacsParser.**write\_exclusions** (*top*)

GromacsParser.**write\_molecules** (*top*)

GromacsParser.**write\_moleculetypes** (*top*)

GromacsParser.**write\_nonbonded\_types** (*top*)

GromacsParser.**write\_pairs** (*top*)

GromacsParser.**write\_settles** (*top*)

GromacsParser.**write\_system** (*top*)

intermol.gromacs.gromacs\_parser.**default\_gromacs\_include\_dir**()

Find the location where gromacs #include files are referenced from, by searching for (1) gromacs environment variables, (2) just using the default gromacs install location, /usr/local/gromacs/share/gromacs/top.

intermol.gromacs.gromacs\_parser.**load\_gromacs** (*top\_file*, *gro\_file*, *include\_dir=None*, *defines=None*)

Load a set of GROMACS input files into a *System*.

**Parameters**

- `top_file` –
- `gro_file` –
- `include_dir` –
- `defines` –

**Returns**

**Return type** `system`

intermol.gromacs.gromacs\_parser.**write\_gromacs** (*top\_file*, *gro\_file*, *system*)

Load a set of GROMACS input files into a *System*.

**Parameters**

- `top_file` –
- `gro_file` –
- `include_dir` –
- `defines` –

**Returns**

**Return type** `system`





```
lammms_dihedrals = {'charmm': <class 'intermol.forces.proper_periodic_dihedral_type.ProperPeriodicDihedral'>, 'r  
lammms_improper_types = {'cvff': <class 'intermol.forces.trig_dihedral_type.TrigDihedralType'>, 'harmonic': <clas  
lammms_impropers = {'cvff': <class 'intermol.forces.trig_dihedral_type.TrigDihedral'>, 'harmonic': <class 'intermol.  
lookup_lammms_angles = {<class 'intermol.forces.cosine_angle_type.CosineAngle'>: 'cosine', <class 'intermol.forces.  
lookup_lammms_bonds = {<class 'intermol.forces.harmonic_bond_type.HarmonicBond'>: 'harmonic', <class 'interme  
lookup_lammms_dihedrals = {<class 'intermol.forces.fourier_dihedral_type.FourierDihedral'>: 'opls', <class 'inter  
lookup_lammms_impropers = {<class 'intermol.forces.improper_harmonic_dihedral_type.ImproperHarmonicDihedr  
parse_angle_coeffs (data_lines)  
parse_angle_style (line)  
parse_angles (data_lines)  
parse_atom_style (line)
```

---

**Note:** Assuming 'full' as default for everything else.

---

```
parse_atoms (data_lines)  
    Read atoms from data file.  
parse_bond_coeffs (data_lines)  
parse_bond_style (line)  
parse_bonded_style (line)  
parse_bonds (data_lines)  
parse_boundary (line)  
parse_box (line, dim)  
    Read box information from data file.
```

#### Parameters

- **line** (*str*) – Current line in input file.
- **dim** (*int*) – Dimension specified in line.

```
parse_dihedral_coeffs (data_lines)  
parse_dihedral_style (line)  
parse_dihedrals (data_lines)  
parse_dimension (line)  
parse_force (data_lines, force_classes, forceSet, n=0)  
    Read bonds, angles, dihedrals, impropers from data file.  
parse_force_coeffs (data_lines, force_name, force_classes, force_style, lammms_forces, canoni-  
    cal_force)  
    Read force coefficients from data file.  
parse_improper_coeffs (data_lines)  
parse_improper_style (line)  
parse_impropers (data_lines)
```

**parse\_kspace\_style** (*line*)

---

**Note:** Currently ignored.

---

**parse\_masses** (*data\_lines*)

Read masses from data file.

**parse\_pair\_coeffs** (*data\_lines*)

Read pair coefficients from data file.

**parse\_pair\_modify** (*line*)

**parse\_pair\_style** (*line*)

**parse\_read\_data** (*line*)

**parse\_special\_bonds** (*line*)

**parse\_units** (*line*)

**parse\_velocities** (*data\_lines*)

**read** ()

Reads a LAMMPS input file and a data file specified within.

**Parameters** **input\_file** (*str*) – Name of LAMMPS input file to read in.

**read\_data** (*data\_file*)

Reads a LAMMPS data file.

**Parameters** **data\_file** (*str*) – name of LAMMPS data file to read in.

**read\_input** ()

Reads a LAMMPS input file.

**Parameters** **input\_file** (*str*) – Name of LAMMPS input file to read in.

**set\_units** (*unit\_set*)

Set what unit set to use.

**write** (*unit\_set='real'*)

Writes a LAMMPS data and corresponding input file.

**Parameters**

- **data\_file** (*str*) – Name of LAMMPS data file to write to.
- **unit\_set** (*str*) – LAMMPS unit set for output file.

**write\_angles** (*mol\_type, offset*)

**write\_bonds** (*mol\_type, offset*)

**write\_dihedrals** (*mol\_type, offset*)

Separate dihedrals from improper.

**write\_forces** (*forces, offset, force\_name, lookup\_lammps\_force, lammps\_force\_types, canonical\_force*)

The general force writing function.

Currently supports bonds, angles, dihedrals, improper.

**write\_impropers** (*mol\_type, offset*)

Separate dihedrals from improper.

**write\_virtuals** (*mol\_type*, *offset*)

`intermol.lammps.lammps_parser.load_lammps` (*in\_file*)

Load a LAMMPS input file into a *System*.

**Parameters**

- **in\_file** –
- **include\_dir** –
- **defines** –

**Returns**

**Return type** `system`

`intermol.lammps.lammps_parser.write_lammps` (*in\_file*, *system*, *unit\_set='real'*)

Load a LAMMPS input file into a *System*.

**Parameters**

- **in\_file** –
- **include\_dir** –
- **defines** –

**Returns**

**Return type** `system`

### 3.3.2 Module contents

## 3.4 intermol.forces package

### 3.4.1 Submodules

#### `intermol.forces.abstract_2_virtual_type` module

`class intermol.forces.abstract_2_virtual_type.Abstract2VirtualType` (*bondingtype1*,  
*bonding-*  
*type2*, *bond-*  
*ingtype3*)

Bases: `intermol.forces.abstract_type.AbstractType`

`bondingtype1`

`bondingtype2`

`placeholder`

**intermol.forces.abstract\_3\_virtual\_type module**

```
class intermol.forces.abstract_3_virtual_type.Abstract3VirtualType (bondingtype1,
                                                                    bonding-
                                                                    type2, bond-
                                                                    ingtype3,
                                                                    bonding-
                                                                    type4)
```

```
    Bases: intermol.forces.abstract_type.AbstractType
```

```
    bondingtype1
```

```
    bondingtype2
```

```
    bondingtype3
```

```
    bondingtype4
```

```
    placeholder
```

**intermol.forces.abstract\_4\_virtual\_type module**

```
class intermol.forces.abstract_4_virtual_type.Abstract4VirtualType (bondingtype1,
                                                                    bonding-
                                                                    type2, bond-
                                                                    ingtype3,
                                                                    bonding-
                                                                    type4, bond-
                                                                    ingtype5)
```

```
    Bases: intermol.forces.abstract_type.AbstractType
```

```
    bondingtype1
```

```
    bondingtype2
```

```
    bondingtype3
```

```
    bondingtype4
```

**intermol.forces.abstract\_angle\_type module**

```
class intermol.forces.abstract_angle_type.AbstractAngleType (bondingtype1, bonding-
                                                                    type2, bondingtype3,
                                                                    c=False)
```

```
    Bases: intermol.forces.abstract_type.AbstractType
```

```
    bondingtype1
```

```
    bondingtype2
```

```
    bondingtype3
```

```
    c
```

**intermol.forces.abstract\_atom\_type module**

```
class intermol.forces.abstract_atom_type.AbstractAtomType (atomtype, bondtype=None,
atomic_number=None,
mass=None, charge=None,
ptype=None)
```

```
    Bases: intermol.forces.abstract_type.AbstractType
```

```
    atomic_number
```

```
    atomtype
```

```
    bondtype
```

```
    charge
```

```
    mass
```

```
    ptype
```

**intermol.forces.abstract\_bond\_type module**

```
class intermol.forces.abstract_bond_type.AbstractBondType (bondingtype1, bonding-
type2, order=1, c=False)
```

```
    Bases: intermol.forces.abstract_type.AbstractType
```

```
    bondingtype1
```

```
    bondingtype2
```

```
    c
```

```
    order
```

**intermol.forces.abstract\_dihedral\_type module**

```
class intermol.forces.abstract_dihedral_type.AbstractDihedralType (bondingtype1,
bondingtype2,
bondingtype3,
bonding-
type4, im-
proper=False)
```

```
    Bases: intermol.forces.abstract_type.AbstractType
```

```
    bondingtype1
```

```
    bondingtype2
```

```
    bondingtype3
```

```
    bondingtype4
```

```
    improper
```

**intermol.forces.abstract\_nonbonded\_type module**

```
class intermol.forces.abstract_nonbonded_type.AbstractNonbondedType (atom1,
atom2,
type)
```

```
    Bases: intermol.forces.abstract_type.AbstractType
```

atom1  
atom2  
type

### intermol.forces.abstract\_pair\_type module

```
class intermol.forces.abstract_pair_type.AbstractPairType(bondingtype1, bonding-
type2, scaleLJ=None,
scaleQQ=None,
long=False)

Bases: intermol.forces.abstract_type.AbstractType

bondingtype1
bondingtype2
long
scaleLJ
scaleQQ
```

### intermol.forces.abstract\_type module

```
class intermol.forces.abstract_type.AbstractType
Bases: object

__repr__()
Print the object and all of its non-magic attributes.
```

### intermol.forces.atom\_c\_type module

```
class intermol.forces.atom_c_type.AtomCType(*args, **kws)
Bases: intermol.forces.abstract_atom_type.AbstractAtomType
```

### intermol.forces.atom\_sigeps\_type module

```
class intermol.forces.atom_sigeps_type.AtomSigepsType(*args, **kws)
Bases: intermol.forces.abstract_atom_type.AbstractAtomType
```

### intermol.forces.buckingham\_nonbonded\_type module

```
class intermol.forces.buckingham_nonbonded_type.BuckinghamNonbonded (atom1,  
atom2,  
bonding-  
type1=None,  
bonding-  
type2=None,  
a=Quantity(value=0.0,  
unit=kilojoule/mole),  
b=Quantity(value=0.0,  
unit=nanometer),  
C6=Quantity(value=0.0,  
unit=nanometer**6*kilojoule/mole),  
type=False)
```

Bases: `intermol.forces.buckingham_nonbonded_type.BuckinghamNonbondedType`

stub documentation

```
class intermol.forces.buckingham_nonbonded_type.BuckinghamNonbondedType (*args,  
**kwds)
```

Bases: `intermol.forces.abstract_nonbonded_type.AbstractNonbondedType`

**C6**

**a**

**b**

**type**

### intermol.forces.connection\_bond\_type module

```
class intermol.forces.connection_bond_type.ConnectionBond (atom1, atom2, bonding-  
type1=None, bonding-  
type2=None, order=1,  
c=False)
```

Bases: `intermol.forces.connection_bond_type.ConnectionBondType`

stub documentation

```
class intermol.forces.connection_bond_type.ConnectionBondType (*args, **kwds)
```

Bases: `intermol.forces.abstract_bond_type.AbstractBondType`

**c**

**order**

### intermol.forces.convert\_dihedrals module

`intermol.forces.convert_dihedrals.convert_dihedral_from_OPLS_to_RB` (*f*)

`intermol.forces.convert_dihedrals.convert_dihedral_from_RB_to_OPLS` (*c*)

`intermol.forces.convert_dihedrals.convert_dihedral_from_RB_to_trig` (*c*)

`intermol.forces.convert_dihedrals.convert_dihedral_from_fourier_to_trig` (*f*)

`intermol.forces.convert_dihedrals.convert_dihedral_from_proper_to_trig` (*p*)

`intermol.forces.convert_dihedrals.convert_dihedral_from_trig_to_RB` (*fcs*)



```
intermol.forces.convert_dihedrals.convert_dihedral_from_trig_to_fourier (fcs)
intermol.forces.convert_dihedrals.convert_dihedral_from_trig_to_proper (fcs,
                                                                    con-
                                                                    ven-
                                                                    tion='0')

intermol.forces.convert_dihedrals.convert_nothing (x)
    useful utility for not converting anything
```

### intermol.forces.cosine\_angle\_type module

```
class intermol.forces.cosine_angle_type.CosineAngle (atom1, atom2, atom3, bond-
                                                    ingtype1=None, bonding-
                                                    type2=None, bondingtype3=None,
                                                    k=Quantity(value=0.0,
                                                    unit=kilojoule/mole), c=False)

    Bases: intermol.forces.cosine_angle_type.CosineAngleType
```

[http://lammps.sandia.gov/doc/angle\\_cosine.html](http://lammps.sandia.gov/doc/angle_cosine.html)

```
class intermol.forces.cosine_angle_type.CosineAngleType (*args, **kws)
    Bases: intermol.forces.abstract_angle_type.AbstractAngleType

    c

    k
```

### intermol.forces.cosine\_squared\_angle\_type module

```
class intermol.forces.cosine_squared_angle_type.CosineSquaredAngle (atom1,
                                                                    atom2,
                                                                    atom3,
                                                                    bonding-
                                                                    type1=None,
                                                                    bonding-
                                                                    type2=None,
                                                                    bonding-
                                                                    type3=None,
                                                                    theta=Quantity(value=0.0,
                                                                    unit=degree),
                                                                    k=Quantity(value=0.0,
                                                                    unit=kilojoule/mole),
                                                                    c=False)

    Bases: intermol.forces.cosine_squared_angle_type.CosineSquaredAngleType

    stub documentation
```

```
class intermol.forces.cosine_squared_angle_type.CosineSquaredAngleType (*args,
                                                                    **kws)
    Bases: intermol.forces.abstract_angle_type.AbstractAngleType

    c

    k

    theta
```

**intermol.forces.cross\_bond\_angle\_angle\_type module**

```
class intermol.forces.cross_bond_angle_angle_type.CrossBondAngleAngle (atom1,
                                                                    atom2,
                                                                    atom3,
                                                                    bonding-
                                                                    type1=None,
                                                                    bonding-
                                                                    type2=None,
                                                                    bonding-
                                                                    type3=None,
                                                                    r1=Quantity(value=0.0,
                                                                    unit=nanometer),
                                                                    r2=Quantity(value=0.0,
                                                                    unit=nanometer),
                                                                    r3=Quantity(value=0.0,
                                                                    unit=nanometer),
                                                                    k=Quantity(value=0.0,
                                                                    unit=kilojoule/(nanometer**2*mole),
                                                                    c=False)

    Bases: intermol.forces.cross_bond_angle_angle_type.CrossBondAngleAngleType
    stub documentation
```

```
class intermol.forces.cross_bond_angle_angle_type.CrossBondAngleAngleType (*args,
                                                                              **kwargs)

    Bases: intermol.forces.abstract_angle_type.AbstractAngleType

    c
    k
    r1
    r2
    r3
```

**intermol.forces.cross\_bond\_bond\_angle\_type module**

```
class intermol.forces.cross_bond_bond_angle_type.CrossBondBondAngle (atom1,
                                                                    atom2,
                                                                    atom3,
                                                                    bonding-
                                                                    type1=None,
                                                                    bonding-
                                                                    type2=None,
                                                                    bonding-
                                                                    type3=None,
                                                                    r1=Quantity(value=0.0,
                                                                    unit=nanometer),
                                                                    r2=Quantity(value=0.0,
                                                                    unit=nanometer),
                                                                    k=Quantity(value=0.0,
                                                                    unit=kilojoule/(nanometer**2*mole),
                                                                    c=False)

    Bases: intermol.forces.cross_bond_bond_angle_type.CrossBondBondAngleType
```

stub documentation

```
class intermol.forces.cross_bond_bond_angle_type.CrossBondBondAngleType (*args,
                                                                    **kwds)
    Bases: intermol.forces.abstract_angle_type.AbstractAngleType
    c
    k
    r1
    r2
```

### intermol.forces.cubic\_bond\_type module

```
class intermol.forces.cubic_bond_type.CubicBond (atom1,          atom2,          bonding-
                                                type1=None,          bondingtype2=None,
                                                length=Quantity(value=0.0,
                                                            unit=nanometer),
                                                C2=Quantity(value=0.0,
                                                            unit=kilojoule/(nanometer**2*mole)),
                                                C3=Quantity(value=0.0,
                                                            unit=kilojoule/(nanometer**3*mole)),
                                                order=1, c=False)
    Bases: intermol.forces.cubic_bond_type.CubicBondType
```

stub documentation

```
class intermol.forces.cubic_bond_type.CubicBondType (*args, **kwds)
    Bases: intermol.forces.abstract_bond_type.AbstractBondType
    C2
    C3
    c
    length
    order
```

### intermol.forces.fene\_bond\_type module

```
class intermol.forces.fene_bond_type.FeneBond (atom1, atom2, bondingtype1=None, bond-
                                                ingtype2=None, length=Quantity(value=0.0,
                                                            unit=nanometer), kb=Quantity(value=0.0,
                                                            unit=kilojoule/(nanometer**2*mole)),
                                                order=1, c=False)
    Bases: intermol.forces.fene_bond_type.FeneBondType
```

stub documentation

```
class intermol.forces.fene_bond_type.FeneBondType (*args, **kwds)
    Bases: intermol.forces.abstract_bond_type.AbstractBondType
    c
    kb
    length
```

**order**

### intermol.forces.fene\_expandable\_bond\_type module

```
class intermol.forces.fene_expandable_bond_type.FeneExpandableBond (atom1,
                                                                    atom2,
                                                                    bonding-
                                                                    type1=None,
                                                                    bonding-
                                                                    type2=None,
                                                                    k=Quantity(value=0.0,
                                                                    unit=kilojoule/(nanometer**2*mole)),
                                                                    length=Quantity(value=0.0,
                                                                    unit=nanometer),
                                                                    ep-
                                                                    silon=Quantity(value=0.0,
                                                                    unit=kilojoule/mole),
                                                                    sigma=Quantity(value=0.0,
                                                                    unit=nanometer),
                                                                    delta=Quantity(value=0.0,
                                                                    unit=nanometer),
                                                                    order=1,
                                                                    c=False)
```

Bases: `intermol.forces.fene_expandable_bond_type.FeneExpandableBondType`

stub documentation

```
class intermol.forces.fene_expandable_bond_type.FeneExpandableBondType (*args,
                                                                    **kwds)
```

Bases: `intermol.forces.abstract_bond_type.AbstractBondType`

**c**

**delta**

**epsilon**

**k**

**length**

**order**

**sigma**

### intermol.forces.forcedata module

### intermol.forces.forcefunctions module

`intermol.forces.forcefunctions.build_paramlist (program)`

Create a paramlist specific for a given program.

`intermol.forces.forcefunctions.build_unitvars (program, paramlist, dumself=None)`

Takes a string program name (one of the supported programs), and a 'self' object it looks like the keyword is not being used, but it is used in the line `eval(unit)`. The test name 'dumself' needs to match what is in the force data arrays. Currently only used for lammmps.

`intermol.forces.forcefunctions.capiifyname (forcename)`

Return name of the class in camelCase.

`intermol.forces.forcefunctions.create_kwd_dict` (*unitvars*, *paramlist*, *force\_type\_object*, *values*, *optvalues=None*)

`intermol.forces.forcefunctions.create_kwds_from_entries` (*unitvars*, *paramlist*, *entries*, *force\_type*, *offset=0*)

Create a keyword dictionary given an array of information from a file format

requires the master set of units, the master set of parameter lists, an object (either a `force_class` or `force_type`), the list of information to be converted into a keyword, and an offset.

**Parameters** `offset` (*int*) – how far over from the first entry we translate

`intermol.forces.forcefunctions.get_parameter_kwds_from_force` (*force*, *forceparams*, *paramlist*)

`intermol.forces.forcefunctions.get_parameter_list_from_force` (*force*, *paramlist*)

Create a function that returns the paramters of a function type.

First, we need make some additions to the parameter list dictionary, which we do once when the `forcedata` script is imported. Useful to put the forces here as well. We won't make this a function for now since it's needed in this module.

`intermol.forces.forcefunctions.get_parameter_list_from_kwds` (*force*, *kwds*, *paramlist*)

`intermol.forces.forcefunctions.optforceparams` (*force\_type*, *forcetype\_object=None*)

Return the dictionary of optional paramters of an abstract force type.

If no object is given, we fill with blanks.

`intermol.forces.forcefunctions.optparamkeylookup` (*force\_type*)

Given a `force_type` object, determine the key associated with the optional parameters.

`intermol.forces.forcefunctions.optparamlookup` (*force\_type\_object*, *object\_default=False*)

A wrapper for `optforceparams` that takes a `force_type` object and returns the optional parameter dictionary.

`intermol.forces.forcefunctions.specify` (*program\_units*, *unitset*, *dumself=None*, *shouldEval=True*)

Takes the dict of units, and a set of dimensions and replaces the dimensions with the appropriate units.

### `intermol.forces.four_fdn_virtual_type` module

```
class intermol.forces.four_fdn_virtual_type.FourFdnVirtual (atom1, atom2, atom3,
atom4, atom5, bondingtype1=None,
bondingtype2=None,
bondingtype3=None,
bondingtype4=None,
bondingtype5=None,
a=Quantity(value=0.0, unit=dimensionless),
b=Quantity(value=0.0, unit=dimensionless),
c=Quantity(value=0.0, unit=nanometer), placeholder=False)
```

Bases: `intermol.forces.four_fdn_virtual_type.FourFdnVirtualType`

stub documentation

```
class intermol.forces.four_fdn_virtual_type.FourFdnVirtualType (*args, **kwds)
```

Bases: `intermol.forces.abstract_4_virtual_type.Abstract4VirtualType`

**a**  
**b**  
**c**  
**placeholder**

### intermol.forces.fourier\_dihedral\_type module

```
class intermol.forces.fourier_dihedral_type.FourierDihedral (atom1, atom2, atom3,  
atom4, bonding-  
type1=None, bond-  
ingtype2=None,  
bondingtype3=None,  
bondingtype4=None,  
c1=Quantity(value=0.0,  
unit=kilojoule/mole),  
c2=Quantity(value=0.0,  
unit=kilojoule/mole),  
c3=Quantity(value=0.0,  
unit=kilojoule/mole),  
c4=Quantity(value=0.0,  
unit=kilojoule/mole),  
c5=Quantity(value=0.0,  
unit=kilojoule/mole),  
improper=False)
```

Bases: `intermol.forces.fourier_dihedral_type.FourierDihedralType`

stub documentation

```
class intermol.forces.fourier_dihedral_type.FourierDihedralType (*args, **kws)  
Bases: intermol.forces.abstract_dihedral_type.AbstractDihedralType
```

**c1**

**c2**

**c3**

**c4**

**c5**

**improper**

### intermol.forces.g96\_bond\_type module

```
class intermol.forces.g96_bond_type.G96Bond (atom1, atom2, bondingtype1=None, bond-  
ingtype2=None, length=Quantity(value=0.0,  
unit=nanometer), k=Quantity(value=0.0,  
unit=kilojoule/(nanometer**4*mole)), order=1,  
c=False)
```

Bases: `intermol.forces.g96_bond_type.G96BondType`

stub documentation

```
class intermol.forces.g96_bond_type.G96BondType (*args, **kws)  
Bases: intermol.forces.abstract_bond_type.AbstractBondType
```

**c**  
**k**  
**length**  
**order**

### intermol.forces.harmonic\_angle\_type module

```
class intermol.forces.harmonic_angle_type.HarmonicAngle(atom1, atom2, atom3,
    bondingtype1=None,
    bondingtype2=None,
    bondingtype3=None,
    theta=Quantity(value=0.0,
    unit=degree),
    k=Quantity(value=0.0,
    unit=kilojoule/(mole*radian**2)),
    c=False)
```

Bases: `intermol.forces.harmonic_angle_type.HarmonicAngleType`

stub documentation

```
class intermol.forces.harmonic_angle_type.HarmonicAngleType(*args, **kws)
    Bases: intermol.forces.abstract_angle_type.AbstractAngleType
```

**c**  
**k**  
**theta**

### intermol.forces.harmonic\_bond\_type module

```
class intermol.forces.harmonic_bond_type.HarmonicBond(atom1, atom2, bond-
    ingtype1=None, bond-
    ingtype2=None,
    length=Quantity(value=0.0,
    unit=nanometer),
    k=Quantity(value=0.0,
    unit=kilojoule/(nanometer**2*mole)),
    order=1, c=False)
```

Bases: `intermol.forces.harmonic_bond_type.HarmonicBondType`

stub documentation

```
class intermol.forces.harmonic_bond_type.HarmonicBondType(*args, **kws)
    Bases: intermol.forces.abstract_bond_type.AbstractBondType
```

**c**  
**k**  
**length**  
**order**

**intermol.forces.harmonic\_potential\_bond\_type** module

```
class intermol.forces.harmonic_potential_bond_type.HarmonicPotentialBond (atom1,  
atom2,  
bond-  
ing-  
type1=None,  
bond-  
ing-  
type2=None,  
length=Quantity(value=0.0,  
unit=nanometer),  
k=Quantity(value=0.0,  
unit=kilojoule/(nanometer**2),  
or-  
der=1,  
c=False)
```

Bases: `intermol.forces.harmonic_potential_bond_type.HarmonicPotentialBondType`

stub documentation

```
class intermol.forces.harmonic_potential_bond_type.HarmonicPotentialBondType (*args,  
**kwds)
```

Bases: `intermol.forces.abstract_bond_type.AbstractBondType`

**c**

**k**

**length**

**order**



**intermol.forces.improper\_harmonic\_dihedral\_type** module

```
class intermol.forces.improper_harmonic_dihedral_type.ImproperHarmonicDihedral (atom1,
                                                                 atom2,
                                                                 atom3,
                                                                 atom4,
                                                                 bond-
                                                                 ing-
                                                                 type1=None,
                                                                 bond-
                                                                 ing-
                                                                 type2=None,
                                                                 bond-
                                                                 ing-
                                                                 type3=None,
                                                                 bond-
                                                                 ing-
                                                                 type4=None,
                                                                 xi=Quantity(value=0,
                                                                 unit=degree),
                                                                 k=Quantity(value=0,
                                                                 unit=kilojoule/mole),
                                                                 im-
                                                                 proper=False)
```

Bases: `intermol.forces.improper_harmonic_dihedral_type.ImproperHarmonicDihedralType`

stub documentation

```
class intermol.forces.improper_harmonic_dihedral_type.ImproperHarmonicDihedralType (*args,
                                                                                       **kwds)
```

Bases: `intermol.forces.abstract_dihedral_type.AbstractDihedralType`

**improper**

**k**

**xi**

**intermol.forces.lj\_c\_nonbonded\_type** module

```
class intermol.forces.lj_c_nonbonded_type.LjCNonbonded (atom1, atom2, bond-
                                                                 ingtype1=None, bond-
                                                                 ingtype2=None,
                                                                 C6=Quantity(value=0.0,
                                                                 unit=nanometer**6*kilojoule/mole),
                                                                 C12=Quantity(value=0.0,
                                                                 unit=nanometer**12*kilojoule/mole),
                                                                 type=False)
```

Bases: `intermol.forces.lj_c_nonbonded_type.LjCNonbondedType`

stub documentation

```
class intermol.forces.lj_c_nonbonded_type.LjCNonbondedType (*args, **kwds)
```

Bases: `intermol.forces.abstract_nonbonded_type.AbstractNonbondedType`

**C12**

**C6**

type

### intermol.forces.lj\_c\_pair\_type module

```
class intermol.forces.lj_c_pair_type.LjCPair(atom1, atom2, bondingtype1=None, bondingtype2=None, C6=Quantity(value=0.0, unit=nanometer**6*kilojoule/mole), C12=Quantity(value=0.0, unit=nanometer**12*kilojoule/mole), scaleLJ=None, scaleQQ=None, long=False)
```

Bases: `intermol.forces.lj_c_pair_type.LjCPairType`

stub documentation

```
class intermol.forces.lj_c_pair_type.LjCPairType(*args, **kwds)
Bases: intermol.forces.abstract_pair_type.AbstractPairType
```

**C12**

**C6**

**long**

**scaleLJ**

**scaleQQ**

### intermol.forces.lj\_default\_pair\_type module

```
class intermol.forces.lj_default_pair_type.LjDefaultPair(atom1, atom2, bondingtype1=None, bondingtype2=None, scaleLJ=None, scaleQQ=None, long=False)
```

Bases: `intermol.forces.lj_default_pair_type.LjDefaultPairType`

stub documentation

```
class intermol.forces.lj_default_pair_type.LjDefaultPairType(*args, **kwds)
Bases: intermol.forces.abstract_pair_type.AbstractPairType
```

**long**

**scaleLJ**

**scaleQQ**

**intermol.forces.lj\_sigeps\_nonbonded\_type module**

```
class intermol.forces.lj_sigeps_nonbonded_type.LjSigepsNonbonded (atom1, atom2,
                                                                bonding-
                                                                type1=None,
                                                                bonding-
                                                                type2=None,
                                                                sigma=Quantity(value=0.0,
                                                                unit=nanometer),
                                                                ep-
                                                                silon=Quantity(value=0.0,
                                                                unit=kilojoule/mole),
                                                                type=False)
```

Bases: `intermol.forces.lj_sigeps_nonbonded_type.LjSigepsNonbondedType`

stub documentation

```
class intermol.forces.lj_sigeps_nonbonded_type.LjSigepsNonbondedType (*args,
                                                                **kwds)
```

Bases: `intermol.forces.abstract_nonbonded_type.AbstractNonbondedType`

**epsilon**

**sigma**

**type**

**intermol.forces.lj\_sigeps\_pair\_type module**

```
class intermol.forces.lj_sigeps_pair_type.LjSigepsPair (atom1, atom2, bond-
                                                                ingtype1=None, bond-
                                                                ingtype2=None,
                                                                sigma=Quantity(value=0.0,
                                                                unit=nanometer), ep-
                                                                silon=Quantity(value=0.0,
                                                                unit=kilojoule/mole),
                                                                scaleLJ=None,
                                                                scaleQQ=None, long=False)
```

Bases: `intermol.forces.lj_sigeps_pair_type.LjSigepsPairType`

stub documentation

```
class intermol.forces.lj_sigeps_pair_type.LjSigepsPairType (*args, **kwds)
```

Bases: `intermol.forces.abstract_pair_type.AbstractPairType`

**epsilon**

**long**

**scaleLJ**

**scaleQQ**

**sigma**

**intermol.forces.ljq\_c\_pair\_type module**

```
class intermol.forces.ljq_c_pair_type.LjqCPair (atom1, atom2, bondingtype1=None, bondingtype2=None, qi=Quantity(value=0.0, unit=elementary charge), qj=Quantity(value=0.0, unit=elementary charge), C6=Quantity(value=0.0, unit=nanometer**6*kilojoule/mole), C12=Quantity(value=0.0, unit=nanometer**12*kilojoule/mole), scaleLJ=None, scaleQQ=None, long=False)
```

Bases: [intermol.forces.ljq\\_c\\_pair\\_type.LjqCPairType](#)

stub documentation

```
class intermol.forces.ljq_c_pair_type.LjqCPairType (*args, **kws)  
Bases: intermol.forces.abstract\_pair\_type.AbstractPairType
```

**C12**

**C6**

**long**

**qi**

**qj**

**scaleLJ**

**scaleQQ**

**intermol.forces.ljq\_default\_pair\_type module**

```
class intermol.forces.ljq_default_pair_type.LjqDefaultPair (atom1, atom2, bondingtype1=None, bondingtype2=None, scaleLJ=None, scaleQQ=None, long=False)
```

Bases: [intermol.forces.ljq\\_default\\_pair\\_type.LjqDefaultPairType](#)

stub documentation

```
class intermol.forces.ljq_default_pair_type.LjqDefaultPairType (*args, **kws)  
Bases: intermol.forces.abstract\_pair\_type.AbstractPairType
```

**long**

**scaleLJ**

**scaleQQ**

**intermol.forces.ljq\_sigeps\_pair\_type** module

```
class intermol.forces.ljq_sigeps_pair_type.LjqSigepsPair (atom1, atom2, bondingtype1=None,
bondingtype2=None,
qi=Quantity(value=0.0,
unit=elementary charge),
qj=Quantity(value=0.0,
unit=elementary charge),
sigma=Quantity(value=0.0,
unit=nanometer), epsilon=Quantity(value=0.0,
unit=kilojoule/mole),
scaleLJ=None,
scaleQQ=None,
long=False)
```

Bases: `intermol.forces.ljq_sigeps_pair_type.LjqSigepsPairType`

stub documentation

```
class intermol.forces.ljq_sigeps_pair_type.LjqSigepsPairType (*args, **kwds)
```

Bases: `intermol.forces.abstract_pair_type.AbstractPairType`

**epsilon**

**long**

**qi**

**qj**

**scaleLJ**

**scaleQQ**

**sigma**

**intermol.forces.make\_forces** module**intermol.forces.morse\_bond\_type** module

```
class intermol.forces.morse_bond_type.MorseBond (atom1, atom2, bondingtype1=None,
bondingtype2=None,
length=Quantity(value=0.0,
unit=nanometer), D=Quantity(value=0.0,
unit=kilojoule/mole),
beta=Quantity(value=0.0,
unit=nanometer), order=1, c=False)
```

Bases: `intermol.forces.morse_bond_type.MorseBondType`

stub documentation

```
class intermol.forces.morse_bond_type.MorseBondType (*args, **kwds)
```

Bases: `intermol.forces.abstract_bond_type.AbstractBondType`

**D**

**beta**

**c**

**length**

**order**

### **intermol.forces.nonlinear\_bond\_type module**

```
class intermol.forces.nonlinear_bond_type.NonlinearBond(atom1, atom2, bond-  
ingtype1=None, bond-  
ingtype2=None, ep-  
silon=Quantity(value=0.0,  
unit=kilojoule/mole),  
r0=Quantity(value=0.0,  
unit=nanometer),  
lamda=Quantity(value=0.0,  
unit=nanometer), order=1,  
c=False)
```

Bases: `intermol.forces.nonlinear_bond_type.NonlinearBondType`

[http://lammps.sandia.gov/doc/bond\\_nonlinear.html](http://lammps.sandia.gov/doc/bond_nonlinear.html)

```
class intermol.forces.nonlinear_bond_type.NonlinearBondType(*args, **kwds)
```

Bases: `intermol.forces.abstract_bond_type.AbstractBondType`

**c**

**epsilon**

**lamda**

**order**

**r0**

**intermol.forces.proper\_periodic\_dihedral\_type** module

```

class intermol.forces.proper_periodic_dihedral_type.ProperPeriodicDihedral (atom1,
                                                                    atom2,
                                                                    atom3,
                                                                    atom4,
                                                                    bond-
                                                                    ing-
                                                                    type1=None,
                                                                    bond-
                                                                    ing-
                                                                    type2=None,
                                                                    bond-
                                                                    ing-
                                                                    type3=None,
                                                                    bond-
                                                                    ing-
                                                                    type4=None,
                                                                    phi=Quantity(value=0.0,
                                                                    unit=degree),
                                                                    k=Quantity(value=0.0,
                                                                    unit=kilojoule/mole),
                                                                    mul-
                                                                    ti-
                                                                    plic-
                                                                    ity=Quantity(value=0.0,
                                                                    unit=dimensionless),
                                                                    weight=Quantity(value=0
                                                                    unit=dimensionless),
                                                                    im-
                                                                    proper=False)

```

Bases: `intermol.forces.proper_periodic_dihedral_type.ProperPeriodicDihedralType`  
 stub documentation

```

class intermol.forces.proper_periodic_dihedral_type.ProperPeriodicDihedralType (*args,
                                                                    **kwds)

```

Bases: `intermol.forces.abstract_dihedral_type.AbstractDihedralType`

**improper**

**k**

**multiplicity**

**phi**

**weight**

**intermol.forces.quartic\_angle\_type module**

```
class intermol.forces.quartic_angle_type.QuarticAngle (atom1, atom2, atom3,  
bondingtype1=None,  
bondingtype2=None,  
bondingtype3=None,  
theta=Quantity(value=0.0,  
unit=degree),  
C0=Quantity(value=0.0,  
unit=kilojoule/mole),  
C1=Quantity(value=0.0,  
unit=kilojoule/(mole*radian)),  
C2=Quantity(value=0.0,  
unit=kilojoule/(mole*radian**2)),  
C3=Quantity(value=0.0,  
unit=kilojoule/(mole*radian**3)),  
C4=Quantity(value=0.0,  
unit=kilojoule/(mole*radian**4)),  
c=False)
```

Bases: `intermol.forces.quartic_angle_type.QuarticAngleType`

stub documentation

```
class intermol.forces.quartic_angle_type.QuarticAngleType (*args, **kwargs)
```

Bases: `intermol.forces.abstract_angle_type.AbstractAngleType`

**C0**

**C1**

**C2**

**C3**

**C4**

**c**

**theta**

**intermol.forces.quartic\_bond\_type module**

```
class intermol.forces.quartic_bond_type.QuarticBond (atom1, atom2, bonding-  
type1=None, bondingtype2=None,  
length=Quantity(value=0.0,  
unit=nanometer),  
C2=Quantity(value=0.0,  
unit=kilojoule/(nanometer**2*mole)),  
C3=Quantity(value=0.0,  
unit=kilojoule/(nanometer**3*mole)),  
C4=Quantity(value=0.0,  
unit=kilojoule/(nanometer**4*mole)),  
order=1, c=False)
```

Bases: `intermol.forces.quartic_bond_type.QuarticBondType`

stub documentation

```
class intermol.forces.quartic_bond_type.QuarticBondType (*args, **kwargs)
```

Bases: `intermol.forces.abstract_bond_type.AbstractBondType`



**C2**  
**C3**  
**C4**  
**c**  
**length**  
**order**

### intermol.forces.quartic\_breakable\_bond\_type module

```

class intermol.forces.quartic_breakable_bond_type.QuarticBreakableBond(atom1,
                                                                    atom2,
                                                                    bond-
                                                                    ing-
                                                                    type1=None,
                                                                    bond-
                                                                    ing-
                                                                    type2=None,
                                                                    k=Quantity(value=0.0,
                                                                    unit=kilojoule/(nanometer**4)*r
                                                                    B1=Quantity(value=0.0,
                                                                    unit=nanometer),
                                                                    B2=Quantity(value=0.0,
                                                                    unit=nanometer),
                                                                    Rc=Quantity(value=0.0,
                                                                    unit=nanometer),
                                                                    U0=Quantity(value=0.0,
                                                                    unit=kilojoule/mole),
                                                                    or-
                                                                    der=1,
                                                                    c=False)

```

Bases: `intermol.forces.quartic_breakable_bond_type.QuarticBreakableBondType`

[http://lammps.sandia.gov/doc/bond\\_quartic.html](http://lammps.sandia.gov/doc/bond_quartic.html)

```

class intermol.forces.quartic_breakable_bond_type.QuarticBreakableBondType(*args,
                                                                              **kwds)

```

Bases: `intermol.forces.abstract_bond_type.AbstractBondType`

**B1**  
**B2**  
**Rc**  
**U0**  
**c**  
**k**  
**order**

**intermol.forces.rb\_dihedral\_type module**

```
class intermol.forces.rb_dihedral_type.RbDihedral (atom1, atom2, atom3, atom4, bonding-  
type1=None, bondingtype2=None,  
bondingtype3=None, bonding-  
type4=None, C0=Quantity(value=0.0,  
unit=kilojoule/mole),  
C1=Quantity(value=0.0,  
unit=kilojoule/mole),  
C2=Quantity(value=0.0,  
unit=kilojoule/mole),  
C3=Quantity(value=0.0,  
unit=kilojoule/mole),  
C4=Quantity(value=0.0,  
unit=kilojoule/mole),  
C5=Quantity(value=0.0,  
unit=kilojoule/mole),  
C6=Quantity(value=0.0,  
unit=kilojoule/mole), im-  
proper=False)  
Bases: intermol.forces.rb_dihedral_type.RbDihedralType
```

stub documentation

```
class intermol.forces.rb_dihedral_type.RbDihedralType (*args, **kws)  
Bases: intermol.forces.abstract_dihedral_type.AbstractDihedralType  
C0  
C1  
C2  
C3  
C4  
C5  
C6  
improper
```

**intermol.forces.settles module**

```
class intermol.forces.settles.Settles (*args, **kws)  
Bases: intermol.forces.abstract_type.AbstractType
```

**intermol.forces.three\_fad\_virtual\_type module**

```
class intermol.forces.three_fad_virtual_type.ThreeFadVirtual (atom1, atom2, atom3,
                                                         atom4,      bonding-
                                                         type1=None,  bond-
                                                         ingtype2=None,
                                                         bondingtype3=None,
                                                         bondingtype4=None,
                                                         theta=Quantity(value=0.0,
                                                         unit=degree),
                                                         d=Quantity(value=0.0,
                                                         unit=nanometer),
                                                         placeholder=False)
```

Bases: `intermol.forces.three_fad_virtual_type.ThreeFadVirtualType`

stub documentation

```
class intermol.forces.three_fad_virtual_type.ThreeFadVirtualType (*args, **kwds)
```

Bases: `intermol.forces.abstract_3_virtual_type.Abstract3VirtualType`

**d**

**placeholder**

**theta**

**intermol.forces.three\_fd\_virtual\_type module**

```
class intermol.forces.three_fd_virtual_type.ThreeFdVirtual (atom1, atom2, atom3,
                                                         atom4,      bonding-
                                                         type1=None,  bond-
                                                         ingtype2=None,
                                                         bondingtype3=None,
                                                         bondingtype4=None,
                                                         a=Quantity(value=0.0,
                                                         unit=dimensionless),
                                                         d=Quantity(value=0.0,
                                                         unit=nanometer), place-
                                                         holder=False)
```

Bases: `intermol.forces.three_fd_virtual_type.ThreeFdVirtualType`

stub documentation

```
class intermol.forces.three_fd_virtual_type.ThreeFdVirtualType (*args, **kwds)
```

Bases: `intermol.forces.abstract_3_virtual_type.Abstract3VirtualType`

**a**

**d**

**placeholder**

**intermol.forces.three\_linear\_virtual\_type module**

```
class intermol.forces.three_linear_virtual_type.ThreeLinearVirtual (atom1,  
atom2,  
atom3,  
atom4,  
bonding-  
type1=None,  
bonding-  
type2=None,  
bonding-  
type3=None,  
bonding-  
type4=None,  
a=Quantity(value=0.0,  
unit=dimensionless),  
b=Quantity(value=0.0,  
unit=dimensionless),  
place-  
holder=False)  
  
Bases: intermol.forces.three_linear_virtual_type.ThreeLinearVirtualType  
stub documentation
```

```
class intermol.forces.three_linear_virtual_type.ThreeLinearVirtualType (*args,  
**kwds)  
Bases: intermol.forces.abstract_3_virtual_type.Abstract3VirtualType  
  
a  
  
b  
  
placeholder
```

**intermol.forces.three\_out\_virtual\_type module**

```
class intermol.forces.three_out_virtual_type.ThreeOutVirtual (atom1, atom2, atom3,  
atom4, bonding-  
type1=None, bond-  
ingtype2=None,  
bondingtype3=None,  
bondingtype4=None,  
a=Quantity(value=0.0,  
unit=dimensionless),  
b=Quantity(value=0.0,  
unit=dimensionless),  
c=Quantity(value=0.0,  
unit=/nanometer),  
placeholder=False)  
  
Bases: intermol.forces.three_out_virtual_type.ThreeOutVirtualType  
stub documentation
```

```
class intermol.forces.three_out_virtual_type.ThreeOutVirtualType (*args, **kwds)  
Bases: intermol.forces.abstract_3_virtual_type.Abstract3VirtualType  
  
a
```

**b**  
**c**  
**placeholder**

### intermol.forces.trig\_dihedral\_type module

```
class intermol.forces.trig_dihedral_type.TrigDihedral (atom1,      atom2,      atom3,
                                                    atom4,      bondingtype1=None,
                                                    bondingtype2=None,
                                                    bondingtype3=None,
                                                    bondingtype4=None,
                                                    phi=Quantity(value=0.0,
                                                         unit=degree),
                                                    fc0=Quantity(value=0.0,
                                                         unit=kilojoule/mole),
                                                    fc1=Quantity(value=0.0,
                                                         unit=kilojoule/mole),
                                                    fc2=Quantity(value=0.0,
                                                         unit=kilojoule/mole),
                                                    fc3=Quantity(value=0.0,
                                                         unit=kilojoule/mole),
                                                    fc4=Quantity(value=0.0,
                                                         unit=kilojoule/mole),
                                                    fc5=Quantity(value=0.0,
                                                         unit=kilojoule/mole),
                                                    fc6=Quantity(value=0.0,
                                                         unit=kilojoule/mole),
                                                    proper=False)           im-
```

Bases: `intermol.forces.trig_dihedral_type.TrigDihedralType`

stub documentation

```
class intermol.forces.trig_dihedral_type.TrigDihedralType (*args, **kwds)
Bases: intermol.forces.abstract_dihedral_type.AbstractDihedralType
fc0
fc1
fc2
fc3
fc4
fc5
fc6
improper
phi
```

### intermol.forces.two\_virtual\_type module

```
class intermol.forces.two_virtual_type.TwoVirtual (atom1, atom2, atom3,  
                                             bondingtype1=None, bond-  
                                             ingtype2=None, bonding-  
                                             type3=None, a=Quantity(value=0.0,  
                                             unit=dimensionless), place-  
                                             holder=False)
```

Bases: `intermol.forces.two_virtual_type.TwoVirtualType`

stub documentation

```
class intermol.forces.two_virtual_type.TwoVirtualType (*args, **kws)  
Bases: intermol.forces.abstract_2_virtual_type.Abstract2VirtualType
```

**a**

**placeholder**

### intermol.forces.urey\_bradley\_angle\_type module

```
class intermol.forces.urey_bradley_angle_type.UreyBradleyAngle (atom1, atom2,  
                                             atom3, bonding-  
                                             type1=None, bond-  
                                             ingtype2=None,  
                                             bonding-  
                                             type3=None,  
                                             theta=Quantity(value=0.0,  
                                             unit=degree),  
                                             k=Quantity(value=0.0,  
                                             unit=kilojoule/(mole*radian**2)),  
                                             r=Quantity(value=0.0,  
                                             unit=nanometer),  
                                             kUB=Quantity(value=0.0,  
                                             unit=kilojoule/(nanometer**2*mole)),  
                                             c=False)
```

Bases: `intermol.forces.urey_bradley_angle_type.UreyBradleyAngleType`

stub documentation

```
class intermol.forces.urey_bradley_angle_type.UreyBradleyAngleType (*args,  
                                                                     **kws)  
Bases: intermol.forces.abstract_angle_type.AbstractAngleType
```

**c**

**k**

**kUB**

**r**

**theta**

## 3.4.2 Module contents

---

**License**

---

InterMol is licensed under the MIT license.





---

## Indices and tables

---

- *genindex*
- *modindex*
- *search*



## i

intermol, 40  
intermol.atom, 37  
intermol.convert, 37  
intermol.decorators, 38  
intermol.forces, 74  
intermol.forces.abstract\_2\_virtual\_type, 48  
intermol.forces.abstract\_3\_virtual\_type, 49  
intermol.forces.abstract\_4\_virtual\_type, 49  
intermol.forces.abstract\_angle\_type, 49  
intermol.forces.abstract\_atom\_type, 50  
intermol.forces.abstract\_bond\_type, 50  
intermol.forces.abstract\_dihedral\_type, 50  
intermol.forces.abstract\_nonbonded\_type, 50  
intermol.forces.abstract\_pair\_type, 51  
intermol.forces.abstract\_type, 51  
intermol.forces.atom\_c\_type, 51  
intermol.forces.atom\_sigeps\_type, 51  
intermol.forces.buckingham\_nonbonded\_type, 52  
intermol.forces.connection\_bond\_type, 52  
intermol.forces.convert\_dihedrals, 52  
intermol.forces.cosine\_angle\_type, 53  
intermol.forces.cosine\_squared\_angle\_type, 53  
intermol.forces.cross\_bond\_angle\_angle\_type, 54  
intermol.forces.cross\_bond\_bond\_angle\_type, 54  
intermol.forces.cubic\_bond\_type, 55  
intermol.forces.fene\_bond\_type, 55  
intermol.forces.fene\_expandable\_bond\_type, 56  
intermol.forces.forcedata, 56  
intermol.forces.forcefunctions, 56  
intermol.forces.four\_fdn\_virtual\_type, 57  
intermol.forces.fourier\_dihedral\_type, 58  
intermol.forces.g96\_bond\_type, 58  
intermol.forces.harmonic\_angle\_type, 59  
intermol.forces.harmonic\_bond\_type, 59  
intermol.forces.harmonic\_potential\_bond\_type, 60  
intermol.forces.improper\_harmonic\_dihedral\_type, 61  
intermol.forces.lj\_c\_nonbonded\_type, 61  
intermol.forces.lj\_c\_pair\_type, 62  
intermol.forces.lj\_default\_pair\_type, 62  
intermol.forces.lj\_sigeps\_nonbonded\_type, 63  
intermol.forces.lj\_sigeps\_pair\_type, 63  
intermol.forces.ljq\_c\_pair\_type, 64  
intermol.forces.ljq\_default\_pair\_type, 64  
intermol.forces.ljq\_sigeps\_pair\_type, 65  
intermol.forces.make\_forces, 65  
intermol.forces.morse\_bond\_type, 65  
intermol.forces.nonlinear\_bond\_type, 66  
intermol.forces.proper\_periodic\_dihedral\_type, 67  
intermol.forces.quartic\_angle\_type, 68  
intermol.forces.quartic\_bond\_type, 68  
intermol.forces.quartic\_breakable\_bond\_type, 69  
intermol.forces.rb\_dihedral\_type, 70  
intermol.forces.settles, 70  
intermol.forces.three\_fad\_virtual\_type, 71  
intermol.forces.three\_fd\_virtual\_type, 71  
intermol.forces.three\_linear\_virtual\_type, 72

`intermol.forces.three_out_virtual_type`,  
72  
`intermol.forces.trig_dihedral_type`, 73  
`intermol.forces.two_virtual_type`, 74  
`intermol.forces.urey_bradley_angle_type`,  
74  
`intermol.gromacs`, 45  
`intermol.gromacs.grofile_parser`, 40  
`intermol.gromacs.gromacs_driver`, 40  
`intermol.gromacs.gromacs_parser`, 40  
`intermol.lammps`, 48  
`intermol.lammps.lammps_driver`, 45  
`intermol.lammps.lammps_parser`, 45  
`intermol.molecule`, 38  
`intermol.moleculetype`, 39  
`intermol.orderedset`, 39  
`intermol.system`, 39

## Symbols

`__repr__()` (intermol.forces.abstract\_type.AbstractType method), 7, 51

### A

`a` (intermol.forces.buckingham\_nonbonded\_type.BuckinghamNonbondedType attribute), 8, 52

`a` (intermol.forces.four\_fdn\_virtual\_type.FourFdnVirtualType attribute), 13, 57

`a` (intermol.forces.three\_fd\_virtual\_type.ThreeFdVirtualType attribute), 25, 71

`a` (intermol.forces.three\_linear\_virtual\_type.ThreeLinearVirtualType attribute), 26, 72

`a` (intermol.forces.three\_out\_virtual\_type.ThreeOutVirtualType attribute), 26, 72

`a` (intermol.forces.two\_virtual\_type.TwoVirtualType attribute), 27, 74

`Abstract2VirtualType` (class in intermol.forces.abstract\_2\_virtual\_type), 5, 48

`Abstract3VirtualType` (class in intermol.forces.abstract\_3\_virtual\_type), 5, 49

`Abstract4VirtualType` (class in intermol.forces.abstract\_4\_virtual\_type), 6, 49

`AbstractAngleType` (class in intermol.forces.abstract\_angle\_type), 6, 49

`AbstractAtomType` (class in intermol.forces.abstract\_atom\_type), 6, 50

`AbstractBondType` (class in intermol.forces.abstract\_bond\_type), 6, 50

`AbstractDihedralType` (class in intermol.forces.abstract\_dihedral\_type), 7, 50

`AbstractNonbondedType` (class in intermol.forces.abstract\_nonbonded\_type), 7, 50

`AbstractPairType` (class in intermol.forces.abstract\_pair\_type), 7, 51

`AbstractType` (class in intermol.forces.abstract\_type), 7, 51

`accepts()` (in module intermol.decorators), 38

`accepts_compatible_units()` (in module intermol.decorators), 38

`add()` (intermol.orderedsset.OrderedSet method), 39

`add_atom()` (intermol.molecule.Molecule method), 38

`add_atomtype()` (intermol.system.System method), 39

`add_molecule()` (intermol.moleculetype.MoleculeType method), 39

`add_molecule()` (intermol.system.System method), 39

`add_molecule_type()` (intermol.system.System method), 39

`Atom` (class in intermol.atom), 37

`atom1` (intermol.forces.abstract\_nonbonded\_type.AbstractNonbondedType attribute), 7, 50

`atom2` (intermol.forces.abstract\_nonbonded\_type.AbstractNonbondedType attribute), 7, 51

`AtomCTYPE` (class in intermol.forces.atom\_c\_type), 8, 51

`atomic_number` (intermol.atom.Atom attribute), 37

`atomic_number` (intermol.forces.abstract\_atom\_type.AbstractAtomType attribute), 6, 50

`atoms` (intermol.molecule.Molecule attribute), 39

`atoms` (intermol.system.System attribute), 39

`AtomSigepsType` (class in intermol.forces.atom\_sigeps\_type), 8, 51

`atomtype` (intermol.atom.Atom attribute), 37

`atomtype` (intermol.forces.abstract\_atom\_type.AbstractAtomType attribute), 6, 50

`atomtypes` (intermol.system.System attribute), 39

### B

`b` (intermol.forces.buckingham\_nonbonded\_type.BuckinghamNonbondedType attribute), 8, 52

`b` (intermol.forces.four\_fdn\_virtual\_type.FourFdnVirtualType attribute), 13, 58

`b` (intermol.forces.three\_linear\_virtual\_type.ThreeLinearVirtualType attribute), 26, 72

`b` (intermol.forces.three\_out\_virtual\_type.ThreeOutVirtualType attribute), 26, 72

`B1` (intermol.forces.quartic\_breakable\_bond\_type.QuarticBreakableBondType attribute), 23, 69

`B2` (intermol.forces.quartic\_breakable\_bond\_type.QuarticBreakableBondType attribute), 23, 69

beta (intermol.forces.morse\_bond\_type.MorseBondType attribute), 20, 65  
 build\_paramlist() (in module intermol.forces.forcefunctions), 12, 56  
 bondingtype (intermol.atom.Atom attribute), 37  
 build\_unitvars() (in module intermol.forces.forcefunctions), 12, 56  
 bondingtype1 (intermol.forces.abstract\_2\_virtual\_type.Abstract2VirtualType attribute), 5, 48  
 C0 (intermol.forces.abstract\_angle\_type.AbstractAngleType attribute), 6, 49  
 bondingtype1 (intermol.forces.abstract\_3\_virtual\_type.Abstract3VirtualType attribute), 5, 49  
 c (intermol.forces.abstract\_angle\_type.AbstractAngleType attribute), 6, 49  
 bondingtype1 (intermol.forces.abstract\_4\_virtual\_type.Abstract4VirtualType attribute), 6, 49  
 c (intermol.forces.abstract\_bond\_type.AbstractBondType attribute), 6, 50  
 bondingtype1 (intermol.forces.abstract\_angle\_type.AbstractAngleType attribute), 6, 49  
 c (intermol.forces.connection\_bond\_type.ConnectionBondType attribute), 8, 52  
 bondingtype1 (intermol.forces.abstract\_bond\_type.AbstractBondType attribute), 6, 50  
 c (intermol.forces.cosine\_angle\_type.CosineAngleType attribute), 9, 53  
 bondingtype1 (intermol.forces.abstract\_dihedral\_type.AbstractDihedralType attribute), 7, 50  
 c (intermol.forces.cosine\_squared\_angle\_type.CosineSquaredAngleType attribute), 9, 53  
 bondingtype1 (intermol.forces.abstract\_pair\_type.AbstractPairType attribute), 7, 51  
 c (intermol.forces.cross\_bond\_angle\_angle\_type.CrossBondAngleAngleType attribute), 10, 54  
 bondingtype2 (intermol.forces.abstract\_2\_virtual\_type.Abstract2VirtualType attribute), 5, 48  
 c (intermol.forces.cross\_bond\_bond\_angle\_type.CrossBondBondAngleType attribute), 11, 55  
 bondingtype2 (intermol.forces.abstract\_3\_virtual\_type.Abstract3VirtualType attribute), 5, 49  
 c (intermol.forces.cubic\_bond\_type.CubicBondType attribute), 11, 55  
 bondingtype2 (intermol.forces.abstract\_4\_virtual\_type.Abstract4VirtualType attribute), 6, 49  
 c (intermol.forces.fene\_bond\_type.FeneBondType attribute), 11, 55  
 bondingtype2 (intermol.forces.abstract\_angle\_type.AbstractAngleType attribute), 6, 49  
 c (intermol.forces.fene\_expandable\_bond\_type.FeneExpandableBondType attribute), 12, 56  
 bondingtype2 (intermol.forces.abstract\_bond\_type.AbstractBondType attribute), 6, 50  
 c (intermol.forces.four\_fdn\_virtual\_type.FourFdnVirtualType attribute), 13, 58  
 bondingtype2 (intermol.forces.abstract\_dihedral\_type.AbstractDihedralType attribute), 7, 50  
 c (intermol.forces.g96\_bond\_type.G96BondType attribute), 14, 58  
 bondingtype2 (intermol.forces.abstract\_pair\_type.AbstractPairType attribute), 7, 51  
 c (intermol.forces.harmonic\_angle\_type.HarmonicAngleType attribute), 15, 59  
 bondingtype3 (intermol.forces.abstract\_3\_virtual\_type.Abstract3VirtualType attribute), 5, 49  
 c (intermol.forces.harmonic\_bond\_type.HarmonicBondType attribute), 15, 59  
 bondingtype3 (intermol.forces.abstract\_4\_virtual\_type.Abstract4VirtualType attribute), 6, 49  
 c (intermol.forces.harmonic\_potential\_bond\_type.HarmonicPotentialBondType attribute), 16, 60  
 bondingtype3 (intermol.forces.abstract\_angle\_type.AbstractAngleType attribute), 6, 49  
 c (intermol.forces.morse\_bond\_type.MorseBondType attribute), 20, 65  
 bondingtype3 (intermol.forces.abstract\_dihedral\_type.AbstractDihedralType attribute), 7, 50  
 c (intermol.forces.nonlinear\_bond\_type.NonlinearBondType attribute), 21, 66  
 bondingtype4 (intermol.forces.abstract\_3\_virtual\_type.Abstract3VirtualType attribute), 5, 49  
 c (intermol.forces.quartic\_angle\_type.QuarticAngleType attribute), 22, 68  
 bondingtype4 (intermol.forces.abstract\_4\_virtual\_type.Abstract4VirtualType attribute), 6, 49  
 c (intermol.forces.quartic\_bond\_type.QuarticBondType attribute), 23, 69  
 bondingtype4 (intermol.forces.abstract\_dihedral\_type.AbstractDihedralType attribute), 7, 50  
 c (intermol.forces.quartic\_breakable\_bond\_type.QuarticBreakableBondType attribute), 24, 69  
 bondtype (intermol.forces.abstract\_atom\_type.AbstractAtomType attribute), 6, 50  
 c (intermol.forces.three\_out\_virtual\_type.ThreeOutVirtualType attribute), 26, 73  
 box\_vector (intermol.system.System attribute), 39  
 BuckinghamNonbonded (class in intermol.forces.buckingham\_nonbonded\_type), 8, 52  
 c (intermol.forces.urey\_bradley\_angle\_type.UreyBradleyAngleType attribute), 28, 74  
 BuckinghamNonbondedType (class in intermol.forces.buckingham\_nonbonded\_type), 8, 52  
 C0 (intermol.forces.quartic\_angle\_type.QuarticAngleType attribute), 22, 68  
 C0 (intermol.forces.rb\_dihedral\_type.RbDihedralType attribute), 24, 70

c1 (intermol.forces.fourier\_dihedral\_type.FourierDihedralType attribute), 14, 58  
 C1 (intermol.forces.quartic\_angle\_type.QuarticAngleType attribute), 22, 68  
 C1 (intermol.forces.rb\_dihedral\_type.RbDihedralType attribute), 24, 70  
 C12 (intermol.forces.lj\_c\_nonbonded\_type.LjCNonbondedType attribute), 17, 61  
 C12 (intermol.forces.lj\_c\_pair\_type.LjCPairType attribute), 17, 62  
 C12 (intermol.forces.ljq\_c\_pair\_type.LjqCPairType attribute), 19, 64  
 C2 (intermol.forces.cubic\_bond\_type.CubicBondType attribute), 11, 55  
 c2 (intermol.forces.fourier\_dihedral\_type.FourierDihedralType attribute), 14, 58  
 C2 (intermol.forces.quartic\_angle\_type.QuarticAngleType attribute), 22, 68  
 C2 (intermol.forces.quartic\_bond\_type.QuarticBondType attribute), 23, 68  
 C2 (intermol.forces.rb\_dihedral\_type.RbDihedralType attribute), 24, 70  
 C3 (intermol.forces.cubic\_bond\_type.CubicBondType attribute), 11, 55  
 c3 (intermol.forces.fourier\_dihedral\_type.FourierDihedralType attribute), 14, 58  
 C3 (intermol.forces.quartic\_angle\_type.QuarticAngleType attribute), 22, 68  
 C3 (intermol.forces.quartic\_bond\_type.QuarticBondType attribute), 23, 69  
 C3 (intermol.forces.rb\_dihedral\_type.RbDihedralType attribute), 24, 70  
 c4 (intermol.forces.fourier\_dihedral\_type.FourierDihedralType attribute), 14, 58  
 C4 (intermol.forces.quartic\_angle\_type.QuarticAngleType attribute), 22, 68  
 C4 (intermol.forces.quartic\_bond\_type.QuarticBondType attribute), 23, 69  
 C4 (intermol.forces.rb\_dihedral\_type.RbDihedralType attribute), 24, 70  
 c5 (intermol.forces.fourier\_dihedral\_type.FourierDihedralType attribute), 14, 58  
 C5 (intermol.forces.rb\_dihedral\_type.RbDihedralType attribute), 24, 70  
 C6 (intermol.forces.buckingham\_nonbonded\_type.BuckinghamNonbondedType attribute), 8, 52  
 C6 (intermol.forces.lj\_c\_nonbonded\_type.LjCNonbondedType attribute), 17, 61  
 C6 (intermol.forces.lj\_c\_pair\_type.LjCPairType attribute), 17, 62  
 C6 (intermol.forces.ljq\_c\_pair\_type.LjqCPairType attribute), 19, 64  
 C6 (intermol.forces.rb\_dihedral\_type.RbDihedralType attribute), 24, 70  
 canonical\_angle() (intermol.gromacs.gromacs\_parser.GromacsParser method), 29, 40  
 canonical\_angle() (intermol.lammps.lammps\_parser.LammpsParser method), 34, 45  
 canonical\_bond() (intermol.gromacs.gromacs\_parser.GromacsParser method), 29, 40  
 canonical\_bond() (intermol.lammps.lammps\_parser.LammpsParser method), 34, 45  
 canonical\_dihedral() (intermol.gromacs.gromacs\_parser.GromacsParser method), 29, 41  
 canonical\_dihedral() (intermol.lammps.lammps\_parser.LammpsParser method), 34, 45  
 capifyname() (in module intermol.forces.forcefunctions), 12, 56  
 cgnr (intermol.atom.Atom attribute), 37  
 charge (intermol.atom.Atom attribute), 37  
 charge (intermol.forces.abstract\_atom\_type.AbstractAtomType attribute), 6, 50  
 choose\_parameter\_kwds\_from\_forces() (intermol.gromacs.gromacs\_parser.GromacsParser method), 30, 41  
 ConnectionBond (class in intermol.forces.connection\_bond\_type), 8, 52  
 ConnectionBondType (class in intermol.forces.connection\_bond\_type), 8, 52  
 convert\_dihedral\_from\_fourier\_to\_trig() (in module intermol.forces.convert\_dihedrals), 9, 52  
 convert\_dihedral\_from\_OPLS\_to\_RB() (in module intermol.forces.convert\_dihedrals), 8, 52  
 convert\_dihedral\_from\_proper\_to\_trig() (in module intermol.forces.convert\_dihedrals), 9, 52  
 convert\_dihedral\_from\_RB\_to\_OPLS() (in module intermol.forces.convert\_dihedrals), 8, 52  
 convert\_dihedral\_from\_RB\_to\_trig() (in module intermol.forces.convert\_dihedrals), 8, 52  
 convert\_dihedral\_from\_trig\_to\_fourier() (in module intermol.forces.convert\_dihedrals), 9, 52  
 convert\_dihedral\_from\_trig\_to\_proper() (in module intermol.forces.convert\_dihedrals), 9, 53  
 convert\_dihedral\_from\_trig\_to\_RB() (in module intermol.forces.convert\_dihedrals), 9, 52  
 convert\_nothing() (in module intermol.forces.convert\_dihedrals), 9, 53  
 CosineAngle (class in intermol.forces.cosine\_angle\_type), 9, 53  
 CosineAngleType (class in intermol.forces.cosine\_angle\_type), 9, 53

- CosineSquaredAngle (class in intermol.forces.cosine\_squared\_angle\_type), 9, 53
- CosineSquaredAngleType (class in intermol.forces.cosine\_squared\_angle\_type), 9, 53
- create\_angle() (intermol.gromacs.gromacs\_parser.GromacsParser method), 30, 41
- create\_atom() (intermol.gromacs.gromacs\_parser.GromacsParser method), 30, 41
- create\_bond() (intermol.gromacs.gromacs\_parser.GromacsParser method), 30, 41
- create\_dihedral() (intermol.gromacs.gromacs\_parser.GromacsParser method), 30, 41
- create\_exclusion() (intermol.gromacs.gromacs\_parser.GromacsParser method), 30, 41
- create\_kwd\_dict() (in module intermol.forces.forcefunctions), 12, 56
- create\_kwds\_from\_entries() (in module intermol.forces.forcefunctions), 12, 57
- create\_kwds\_from\_entries() (intermol.gromacs.gromacs\_parser.GromacsParser method), 30, 41
- create\_kwds\_from\_entries() (intermol.lammps.lammps\_parser.LammpsParser method), 34, 45
- create\_molecule() (intermol.gromacs.gromacs\_parser.GromacsParser method), 30, 41
- create\_moleculetype() (intermol.gromacs.gromacs\_parser.GromacsParser method), 30, 42
- create\_pair() (intermol.gromacs.gromacs\_parser.GromacsParser method), 30, 42
- create\_settle() (intermol.gromacs.gromacs\_parser.GromacsParser method), 30, 42
- CrossBondAngleAngle (class in intermol.forces.cross\_bond\_angle\_angle\_type), 10, 54
- CrossBondAngleAngleType (class in intermol.forces.cross\_bond\_angle\_angle\_type), 10, 54
- CrossBondBondAngle (class in intermol.forces.cross\_bond\_bond\_angle\_type), 10, 54
- CrossBondBondAngleType (class in intermol.forces.cross\_bond\_bond\_angle\_type), 11, 55
- CubicBond (class in intermol.forces.cubic\_bond\_type), 11, 55
- CubicBondType (class in intermol.forces.cubic\_bond\_type), 11, 55
- D**
- D (intermol.forces.morse\_bond\_type.MorseBondType attribute), 20, 65
- d (intermol.forces.three\_fad\_virtual\_type.ThreeFadVirtualType attribute), 25, 71
- d (intermol.forces.three\_fd\_virtual\_type.ThreeFdVirtualType attribute), 25, 71
- default\_gromacs\_include\_dir() (in module intermol.gromacs.gromacs\_parser), 33, 44
- delta (intermol.forces.fene\_expandable\_bond\_type.FeneExpandableBondType attribute), 12, 56
- difference\_update() (intermol.orderedsset.OrderedSet method), 39
- directive\_before\_moleculetype() (intermol.gromacs.gromacs\_parser.GromacsParser method), 30, 42
- discard() (intermol.orderedsset.OrderedSet method), 39
- E**
- epsilon (intermol.atom.Atom attribute), 37
- epsilon (intermol.forces.fene\_expandable\_bond\_type.FeneExpandableBondType attribute), 12, 56
- epsilon (intermol.forces.lj\_sigeps\_nonbonded\_type.LjSigepsNonbondedType attribute), 18, 63
- epsilon (intermol.forces.lj\_sigeps\_pair\_type.LjSigepsPairType attribute), 18, 63
- epsilon (intermol.forces.ljq\_sigeps\_pair\_type.LjqSigepsPairType attribute), 20, 65
- epsilon (intermol.forces.nonlinear\_bond\_type.NonlinearBondType attribute), 21, 66
- F**
- fc0 (intermol.forces.trig\_dihedral\_type.TrigDihedralType attribute), 27, 73
- fc1 (intermol.forces.trig\_dihedral\_type.TrigDihedralType attribute), 27, 73
- fc2 (intermol.forces.trig\_dihedral\_type.TrigDihedralType attribute), 27, 73
- fc3 (intermol.forces.trig\_dihedral\_type.TrigDihedralType attribute), 27, 73
- fc4 (intermol.forces.trig\_dihedral\_type.TrigDihedralType attribute), 27, 73
- fc5 (intermol.forces.trig\_dihedral\_type.TrigDihedralType attribute), 27, 73
- fc6 (intermol.forces.trig\_dihedral\_type.TrigDihedralType attribute), 27, 73
- FeneBond (class in intermol.forces.fene\_bond\_type), 11, 55
- FeneBondType (class in intermol.forces.fene\_bond\_type), 11, 55
- FeneExpandableBond (class in intermol.forces.fene\_expandable\_bond\_type), 12, 56



- FeneExpandableBondType (class in intermol.forces.fene\_expandable\_bond\_type), 12, 56
- find\_dihedraltype() (intermol.gromacs.gromacs\_parser.GromacsParser method), 30, 42
- find\_forcetype() (intermol.gromacs.gromacs\_parser.GromacsParser method), 30, 42
- find\_match() (in module intermol.convert), 37
- force (intermol.atom.Atom attribute), 37
- FourFdnVirtual (class in intermol.forces.four\_fdn\_virtual\_type), 13, 57
- FourFdnVirtualType (class in intermol.forces.four\_fdn\_virtual\_type), 13, 57
- FourierDihedral (class in intermol.forces.fourier\_dihedral\_type), 14, 58
- FourierDihedralType (class in intermol.forces.fourier\_dihedral\_type), 14, 58
- ## G
- G96Bond (class in intermol.forces.g96\_bond\_type), 14, 58
- G96BondType (class in intermol.forces.g96\_bond\_type), 14, 58
- get\_diff() (in module intermol.convert), 37
- get\_force\_atoms() (intermol.lammps.lammps\_parser.LammpsParser method), 34, 45
- get\_force\_bondingtypes() (intermol.lammps.lammps\_parser.LammpsParser method), 34, 45
- get\_parameter\_kwds\_from\_force() (in module intermol.forces.forcefunctions), 13, 57
- get\_parameter\_kwds\_from\_force() (intermol.gromacs.gromacs\_parser.GromacsParser method), 30, 42
- get\_parameter\_kwds\_from\_force() (intermol.lammps.lammps\_parser.LammpsParser method), 34, 45
- get\_parameter\_list\_from\_force() (in module intermol.forces.forcefunctions), 13, 57
- get\_parameter\_list\_from\_force() (intermol.gromacs.gromacs\_parser.GromacsParser method), 30, 42
- get\_parameter\_list\_from\_force() (intermol.lammps.lammps\_parser.LammpsParser method), 34, 45
- get\_parameter\_list\_from\_kwds() (in module intermol.forces.forcefunctions), 13, 57
- gromacs\_angle\_types (intermol.gromacs.gromacs\_parser.GromacsParser attribute), 30, 42
- gromacs\_angles (intermol.gromacs.gromacs\_parser.GromacsParser attribute), 30, 42
- gromacs\_bond\_types (intermol.gromacs.gromacs\_parser.GromacsParser attribute), 30, 42
- gromacs\_bonds (intermol.gromacs.gromacs\_parser.GromacsParser attribute), 30, 42
- gromacs\_combination\_rules (intermol.gromacs.gromacs\_parser.GromacsParser attribute), 30, 42
- gromacs\_dihedral\_types (intermol.gromacs.gromacs\_parser.GromacsParser attribute), 30, 42
- gromacs\_dihedrals (intermol.gromacs.gromacs\_parser.GromacsParser attribute), 30, 42
- gromacs\_energies() (in module intermol.gromacs.gromacs\_driver), 29, 40
- gromacs\_pair\_types (intermol.gromacs.gromacs\_parser.GromacsParser attribute), 31, 42
- gromacs\_pairs (intermol.gromacs.gromacs\_parser.GromacsParser attribute), 31, 42
- GromacsGroParser (class in intermol.gromacs.grofile\_parser), 28, 40
- GromacsParser (class in intermol.gromacs.gromacs\_parser), 29, 40
- GromacsParser.TopMoleculeType (class in intermol.gromacs.gromacs\_parser), 29, 40
- ## H
- HarmonicAngle (class in intermol.forces.harmonic\_angle\_type), 15, 59
- HarmonicAngleType (class in intermol.forces.harmonic\_angle\_type), 15, 59
- HarmonicBond (class in intermol.forces.harmonic\_bond\_type), 15, 59
- HarmonicBondType (class in intermol.forces.harmonic\_bond\_type), 15, 59
- HarmonicPotentialBond (class in intermol.forces.harmonic\_potential\_bond\_type), 15, 60
- HarmonicPotentialBondType (class in intermol.forces.harmonic\_potential\_bond\_type), 16, 60
- ## I
- improper (intermol.forces.abstract\_dihedral\_type.AbstractDihedralType attribute), 7, 50
- improper (intermol.forces.fourier\_dihedral\_type.FourierDihedralType attribute), 14, 58
- improper (intermol.forces.improper\_harmonic\_dihedral\_type.ImproperHarmonicDihedralType attribute), 17, 61
- improper (intermol.forces.proper\_periodic\_dihedral\_type.ProperPeriodicDihedralType attribute), 22, 67

- improper (intermol.forces.rb\_dihedral\_type.RbDihedralType attribute), 24, 70
  - improper (intermol.forces.trig\_dihedral\_type.TrigDihedralType attribute), 27, 73
  - ImproperHarmonicDihedral (class in intermol.forces.improper\_harmonic\_dihedral\_type), 16, 61
  - ImproperHarmonicDihedralType (class in intermol.forces.improper\_harmonic\_dihedral\_type), 17, 61
  - index (intermol.atom.Atom attribute), 37
  - intermol (module), 40
  - intermol.atom (module), 37
  - intermol.convert (module), 37
  - intermol.decorators (module), 38
  - intermol.forces (module), 28, 74
  - intermol.forces.abstract\_2\_virtual\_type (module), 5, 48
  - intermol.forces.abstract\_3\_virtual\_type (module), 5, 49
  - intermol.forces.abstract\_4\_virtual\_type (module), 6, 49
  - intermol.forces.abstract\_angle\_type (module), 6, 49
  - intermol.forces.abstract\_atom\_type (module), 6, 50
  - intermol.forces.abstract\_bond\_type (module), 6, 50
  - intermol.forces.abstract\_dihedral\_type (module), 7, 50
  - intermol.forces.abstract\_nonbonded\_type (module), 7, 50
  - intermol.forces.abstract\_pair\_type (module), 7, 51
  - intermol.forces.abstract\_type (module), 7, 51
  - intermol.forces.atom\_c\_type (module), 8, 51
  - intermol.forces.atom\_sigeps\_type (module), 8, 51
  - intermol.forces.buckingham\_nonbonded\_type (module), 8, 52
  - intermol.forces.connection\_bond\_type (module), 8, 52
  - intermol.forces.convert\_dihedrals (module), 8, 52
  - intermol.forces.cosine\_angle\_type (module), 9, 53
  - intermol.forces.cosine\_squared\_angle\_type (module), 9, 53
  - intermol.forces.cross\_bond\_angle\_angle\_type (module), 10, 54
  - intermol.forces.cross\_bond\_bond\_angle\_type (module), 10, 54
  - intermol.forces.cubic\_bond\_type (module), 11, 55
  - intermol.forces.fene\_bond\_type (module), 11, 55
  - intermol.forces.fene\_expandable\_bond\_type (module), 12, 56
  - intermol.forces.forcedata (module), 12, 56
  - intermol.forces.forcefunctions (module), 12, 56
  - intermol.forces.four\_fdn\_virtual\_type (module), 13, 57
  - intermol.forces.fourier\_dihedral\_type (module), 14, 58
  - intermol.forces.g96\_bond\_type (module), 14, 58
  - intermol.forces.harmonic\_angle\_type (module), 15, 59
  - intermol.forces.harmonic\_bond\_type (module), 15, 59
  - intermol.forces.harmonic\_potential\_bond\_type (module), 15, 60
  - intermol.forces.improper\_harmonic\_dihedral\_type (module), 16, 61
  - intermol.forces.lj\_c\_nonbonded\_type (module), 17, 61
  - intermol.forces.lj\_c\_pair\_type (module), 17, 62
  - intermol.forces.lj\_default\_pair\_type (module), 17, 62
  - intermol.forces.lj\_sigeps\_nonbonded\_type (module), 18, 63
  - intermol.forces.lj\_sigeps\_pair\_type (module), 18, 63
  - intermol.forces.ljq\_c\_pair\_type (module), 19, 64
  - intermol.forces.ljq\_default\_pair\_type (module), 19, 64
  - intermol.forces.ljq\_sigeps\_pair\_type (module), 20, 65
  - intermol.forces.make\_forces (module), 20, 65
  - intermol.forces.morse\_bond\_type (module), 20, 65
  - intermol.forces.nonlinear\_bond\_type (module), 21, 66
  - intermol.forces.proper\_periodic\_dihedral\_type (module), 21, 67
  - intermol.forces.quartic\_angle\_type (module), 22, 68
  - intermol.forces.quartic\_bond\_type (module), 22, 68
  - intermol.forces.quartic\_breakable\_bond\_type (module), 23, 69
  - intermol.forces.rb\_dihedral\_type (module), 24, 70
  - intermol.forces.settles (module), 24, 70
  - intermol.forces.three\_fad\_virtual\_type (module), 25, 71
  - intermol.forces.three\_fd\_virtual\_type (module), 25, 71
  - intermol.forces.three\_linear\_virtual\_type (module), 25, 72
  - intermol.forces.three\_out\_virtual\_type (module), 26, 72
  - intermol.forces.trig\_dihedral\_type (module), 27, 73
  - intermol.forces.two\_virtual\_type (module), 27, 74
  - intermol.forces.urey\_bradley\_angle\_type (module), 28, 74
  - intermol.gromacs (module), 33, 45
  - intermol.gromacs.grofile\_parser (module), 28, 40
  - intermol.gromacs.gromacs\_driver (module), 29, 40
  - intermol.gromacs.gromacs\_parser (module), 29, 40
  - intermol.lammps (module), 37, 48
  - intermol.lammps.lammps\_driver (module), 33, 45
  - intermol.lammps.lammps\_parser (module), 33, 45
  - intermol.molecule (module), 38
  - intermol.moleculetype (module), 39
  - intermol.orderedset (module), 39
  - intermol.system (module), 39
  - intersection\_update() (intermol.orderedset.OrderedSet method), 39
  - invalid\_line() (intermol.gromacs.gromacs\_parser.GromacsParser method), 31, 42
- ## K
- k (intermol.forces.cosine\_angle\_type.CosineAngleType attribute), 9, 53
  - k (intermol.forces.cosine\_squared\_angle\_type.CosineSquaredAngleType attribute), 9, 53
  - k (intermol.forces.cross\_bond\_angle\_angle\_type.CrossBondAngleAngleType attribute), 10, 54
  - k (intermol.forces.cross\_bond\_bond\_angle\_type.CrossBondBondAngleType attribute), 11, 55

k (intermol.forces.fene\_expandable\_bond\_type.FeneExpandableBondType attribute), 12, 56  
 k (intermol.forces.g96\_bond\_type.G96BondType attribute), 14, 59  
 k (intermol.forces.harmonic\_angle\_type.HarmonicAngleType attribute), 15, 59  
 k (intermol.forces.harmonic\_bond\_type.HarmonicBondType attribute), 15, 59  
 k (intermol.forces.harmonic\_potential\_bond\_type.HarmonicPotentialBondType attribute), 16, 60  
 k (intermol.forces.improper\_harmonic\_dihedral\_type.ImproperHarmonicDihedralType attribute), 17, 61  
 k (intermol.forces.proper\_periodic\_dihedral\_type.ProperPeriodicDihedralType attribute), 22, 67  
 k (intermol.forces.quartic\_breakable\_bond\_type.QuarticBreakableBondType attribute), 24, 69  
 k (intermol.forces.urey\_bradley\_angle\_type.UreyBradleyAngleType attribute), 28, 74  
 kb (intermol.forces.fene\_bond\_type.FeneBondType attribute), 11, 55  
 kUB (intermol.forces.urey\_bradley\_angle\_type.UreyBradleyAngleType attribute), 28, 74

**L**

lamda (intermol.forces.nonlinear\_bond\_type.NonlinearBondType attribute), 21, 66  
 lammmps\_angle\_types (intermol.lammmps.lammmps\_parser.LammmpsParser attribute), 34, 45  
 lammmps\_angles (intermol.lammmps.lammmps\_parser.LammmpsParser attribute), 34, 45  
 lammmps\_bond\_types (intermol.lammmps.lammmps\_parser.LammmpsParser attribute), 34, 45  
 lammmps\_bonds (intermol.lammmps.lammmps\_parser.LammmpsParser attribute), 34, 45  
 lammmps\_dihedral\_types (intermol.lammmps.lammmps\_parser.LammmpsParser attribute), 34, 45  
 lammmps\_dihedrals (intermol.lammmps.lammmps\_parser.LammmpsParser attribute), 34, 45  
 lammmps\_energies() (in module intermol.lammmps.lammmps\_driver), 33, 45  
 lammmps\_improper\_types (intermol.lammmps.lammmps\_parser.LammmpsParser attribute), 34, 46  
 lammmps\_impropers (intermol.lammmps.lammmps\_parser.LammmpsParser attribute), 34, 46  
 LammmpsParser (class in intermol.lammmps.lammmps\_parser), 33, 45  
 length (intermol.forces.cubic\_bond\_type.CubicBondType attribute), 11, 55

length (intermol.forces.fene\_expandable\_bond\_type.FeneExpandableBondType attribute), 12, 56  
 length (intermol.forces.g96\_bond\_type.G96BondType attribute), 14, 59  
 length (intermol.forces.harmonic\_bond\_type.HarmonicBondType attribute), 15, 59  
 length (intermol.forces.harmonic\_potential\_bond\_type.HarmonicPotentialBondType attribute), 16, 60  
 length (intermol.forces.improper\_harmonic\_dihedral\_type.ImproperHarmonicDihedralType attribute), 17, 61  
 length (intermol.forces.proper\_periodic\_dihedral\_type.ProperPeriodicDihedralType attribute), 22, 67  
 length (intermol.forces.quartic\_bond\_type.QuarticBondType attribute), 23, 69  
 LjCNonbondedType (class in intermol.forces.lj\_c\_nonbonded\_type), 17, 61  
 LjCNonbondedType (class in intermol.forces.lj\_c\_nonbonded\_type), 17, 61  
 LjCPair (class in intermol.forces.lj\_c\_pair\_type), 17, 62  
 LjCPairType (class in intermol.forces.lj\_c\_pair\_type), 17, 62  
 LjDefaultPair (class in intermol.forces.lj\_default\_pair\_type), 17, 62  
 LjDefaultPairType (class in intermol.forces.lj\_default\_pair\_type), 18, 62  
 LjqCPair (class in intermol.forces.ljq\_c\_pair\_type), 19, 64  
 LjqCPairType (class in intermol.forces.ljq\_c\_pair\_type), 19, 64  
 LjqDefaultPair (class in intermol.forces.ljq\_default\_pair\_type), 19, 64  
 LjqDefaultPairType (class in intermol.forces.ljq\_default\_pair\_type), 19, 64  
 LjqSigepsPair (class in intermol.forces.ljq\_sigeps\_pair\_type), 20, 65  
 LjqSigepsPairType (class in intermol.forces.ljq\_sigeps\_pair\_type), 20, 65  
 LjSigepsNonbonded (class in intermol.forces.lj\_sigeps\_nonbonded\_type), 18, 63  
 LjSigepsNonbondedType (class in intermol.forces.lj\_sigeps\_nonbonded\_type), 18, 63  
 LjSigepsPair (class in intermol.forces.lj\_sigeps\_pair\_type), 18, 63  
 LjSigepsPairType (class in intermol.forces.lj\_sigeps\_pair\_type), 18, 63  
 load\_gromacs() (in module intermol.gromacs.gromacs\_parser), 33, 44  
 load\_lammmps() (in module intermol.lammmps.lammmps\_parser), 36, 48  
 long (intermol.forces.abstract\_pair\_type.AbstractPairType attribute), 7, 51

- long (intermol.forces.lj\_c\_pair\_type.LjCPairType attribute), 17, 62
- long (intermol.forces.lj\_default\_pair\_type.LjDefaultPairType attribute), 18, 62
- long (intermol.forces.lj\_sigeps\_pair\_type.LjSigepsPairType attribute), 19, 63
- long (intermol.forces.ljq\_c\_pair\_type.LjqCPairType attribute), 19, 64
- long (intermol.forces.ljq\_default\_pair\_type.LjqDefaultPairType attribute), 19, 64
- long (intermol.forces.ljq\_sigeps\_pair\_type.LjqSigepsPairType attribute), 20, 65
- lookup\_atom\_atomtype() (intermol.gromacs.gromacs\_parser.GromacsParser method), 31, 42
- lookup\_atom\_bondingtype() (intermol.gromacs.gromacs\_parser.GromacsParser method), 31, 42
- lookup\_gromacs\_angles (intermol.gromacs.gromacs\_parser.GromacsParser attribute), 31, 42
- lookup\_gromacs\_bonds (intermol.gromacs.gromacs\_parser.GromacsParser attribute), 31, 42
- lookup\_gromacs\_combination\_rules (intermol.gromacs.gromacs\_parser.GromacsParser attribute), 31, 42
- lookup\_gromacs\_dihedrals (intermol.gromacs.gromacs\_parser.GromacsParser attribute), 31, 42
- lookup\_gromacs\_pairs (intermol.gromacs.gromacs\_parser.GromacsParser attribute), 31, 42
- lookup\_lammps\_angles (intermol.lammps.lammps\_parser.LammpsParser attribute), 34, 46
- lookup\_lammps\_bonds (intermol.lammps.lammps\_parser.LammpsParser attribute), 34, 46
- lookup\_lammps\_dihedrals (intermol.lammps.lammps\_parser.LammpsParser attribute), 34, 46
- lookup\_lammps\_impropers (intermol.lammps.lammps\_parser.LammpsParser attribute), 34, 46
- M**
- main() (in module intermol.convert), 37
- mass (intermol.atom.Atom attribute), 37
- mass (intermol.forces.abstract\_atom\_type.AbstractAtomType attribute), 6, 50
- Molecule (class in intermol.molecule), 38
- molecule\_types (intermol.system.System attribute), 39
- MoleculeType (class in intermol.molecule), 39
- MorseBond (class in intermol.forces.morse\_bond\_type), 20, 65
- MorseBondType (class in intermol.forces.morse\_bond\_type), 20, 65
- multiplicity (intermol.forces.proper\_periodic\_dihedral\_type.ProperPeriodic attribute), 22, 67
- N**
- name (intermol.system.System attribute), 39
- name (intermol.atom.Atom attribute), 37
- nonbonded\_types (intermol.system.System attribute), 39
- NonlinearBond (class in intermol.forces.nonlinear\_bond\_type), 21, 66
- NonlinearBondType (class in intermol.forces.nonlinear\_bond\_type), 21, 66
- O**
- optforceparams() (in module intermol.forces.forcefunctions), 13, 57
- optparamkeylookup() (in module intermol.forces.forcefunctions), 13, 57
- optparamlookup() (in module intermol.forces.forcefunctions), 13, 57
- order (intermol.forces.abstract\_bond\_type.AbstractBondType attribute), 6, 50
- order (intermol.forces.connection\_bond\_type.ConnectionBondType attribute), 8, 52
- order (intermol.forces.cubic\_bond\_type.CubicBondType attribute), 11, 55
- order (intermol.forces.fene\_bond\_type.FeneBondType attribute), 11, 55
- order (intermol.forces.fene\_expandable\_bond\_type.FeneExpandableBondType attribute), 12, 56
- order (intermol.forces.g96\_bond\_type.G96BondType attribute), 14, 59
- order (intermol.forces.harmonic\_bond\_type.HarmonicBondType attribute), 15, 59
- order (intermol.forces.harmonic\_potential\_bond\_type.HarmonicPotentialBondType attribute), 16, 60
- order (intermol.forces.morse\_bond\_type.MorseBondType attribute), 20, 66
- order (intermol.forces.nonlinear\_bond\_type.NonlinearBondType attribute), 21, 66
- order (intermol.forces.quartic\_bond\_type.QuarticBondType attribute), 23, 69
- order (intermol.forces.quartic\_breakable\_bond\_type.QuarticBreakableBondType attribute), 24, 69
- OrderedSet (class in intermol.orderedset), 39
- P**
- paramlist (intermol.gromacs.gromacs\_parser.GromacsParser attribute), 31, 42

parse\_angle\_coeffs() (intermol.lammps.lammps\_parser.LammpsParser method), 34, 46  
 parse\_angle\_style() (intermol.lammps.lammps\_parser.LammpsParser method), 34, 46  
 parse\_angles() (intermol.lammps.lammps\_parser.LammpsParser method), 34, 46  
 parse\_args() (in module intermol.convert), 37  
 parse\_atom\_style() (intermol.lammps.lammps\_parser.LammpsParser method), 34, 46  
 parse\_atoms() (intermol.lammps.lammps\_parser.LammpsParser method), 34, 46  
 parse\_bond\_coeffs() (intermol.lammps.lammps\_parser.LammpsParser method), 34, 46  
 parse\_bond\_style() (intermol.lammps.lammps\_parser.LammpsParser method), 34, 46  
 parse\_bonded\_style() (intermol.lammps.lammps\_parser.LammpsParser method), 34, 46  
 parse\_bonds() (intermol.lammps.lammps\_parser.LammpsParser method), 34, 46  
 parse\_boundary() (intermol.lammps.lammps\_parser.LammpsParser method), 35, 46  
 parse\_box() (intermol.lammps.lammps\_parser.LammpsParser method), 35, 46  
 parse\_dihedral\_coeffs() (intermol.lammps.lammps\_parser.LammpsParser method), 35, 46  
 parse\_dihedral\_style() (intermol.lammps.lammps\_parser.LammpsParser method), 35, 46  
 parse\_dihedrals() (intermol.lammps.lammps\_parser.LammpsParser method), 35, 46  
 parse\_dimension() (intermol.lammps.lammps\_parser.LammpsParser method), 35, 46  
 parse\_force() (intermol.lammps.lammps\_parser.LammpsParser method), 35, 46  
 parse\_force\_coeffs() (intermol.lammps.lammps\_parser.LammpsParser method), 35, 46  
 parse\_improper\_coeffs() (intermol.lammps.lammps\_parser.LammpsParser method), 35, 46  
 parse\_improper\_style() (intermol.lammps.lammps\_parser.LammpsParser method), 35, 46  
 parse\_impropers() (intermol.lammps.lammps\_parser.LammpsParser method), 35, 46  
 parse\_kspace\_style() (intermol.lammps.lammps\_parser.LammpsParser method), 35, 46  
 parse\_masses() (intermol.lammps.lammps\_parser.LammpsParser method), 35, 47  
 parse\_pair\_coeffs() (intermol.lammps.lammps\_parser.LammpsParser method), 35, 47  
 parse\_pair\_modify() (intermol.lammps.lammps\_parser.LammpsParser method), 35, 47  
 parse\_pair\_style() (intermol.lammps.lammps\_parser.LammpsParser method), 35, 47  
 parse\_read\_data() (intermol.lammps.lammps\_parser.LammpsParser method), 35, 47  
 parse\_special\_bonds() (intermol.lammps.lammps\_parser.LammpsParser method), 35, 47  
 parse\_units() (intermol.lammps.lammps\_parser.LammpsParser method), 35, 47  
 parse\_velocities() (intermol.lammps.lammps\_parser.LammpsParser method), 35, 47  
 phi (intermol.forces.proper\_periodic\_dihedral\_type.ProperPeriodicDihedralType attribute), 22, 67  
 phi (intermol.forces.trig\_dihedral\_type.TrigDihedralType attribute), 27, 73  
 placeholder (intermol.forces.abstract\_2\_virtual\_type.Abstract2VirtualType attribute), 5, 48  
 placeholder (intermol.forces.abstract\_3\_virtual\_type.Abstract3VirtualType attribute), 5, 49  
 placeholder (intermol.forces.four\_fdn\_virtual\_type.FourFdnVirtualType attribute), 14, 58  
 placeholder (intermol.forces.three\_fad\_virtual\_type.ThreeFadVirtualType attribute), 25, 71  
 placeholder (intermol.forces.three\_fd\_virtual\_type.ThreeFdVirtualType attribute), 25, 71  
 placeholder (intermol.forces.three\_linear\_virtual\_type.ThreeLinearVirtualType attribute), 26, 72  
 placeholder (intermol.forces.three\_out\_virtual\_type.ThreeOutVirtualType attribute), 26, 73  
 placeholder (intermol.forces.two\_virtual\_type.TwoVirtualType attribute), 28, 74  
 position (intermol.atom.Atom attribute), 37  
 process\_angle() (intermol.gromacs.gromacs\_parser.GromacsParser method), 31, 42  
 process\_angletype() (intermol.gromacs.gromacs\_parser.GromacsParser method), 31, 42

process\_atom() (intermol.gromacs.gromacs\_parser.GromacsParser method), 31, 42  
 process\_atomtype() (intermol.gromacs.gromacs\_parser.GromacsParser method), 31, 42  
 process\_bond() (intermol.gromacs.gromacs\_parser.GromacsParser method), 31, 42  
 process\_bondtype() (intermol.gromacs.gromacs\_parser.GromacsParser method), 31, 43  
 process\_cmap() (intermol.gromacs.gromacs\_parser.GromacsParser method), 31, 43  
 process\_cmaptype() (intermol.gromacs.gromacs\_parser.GromacsParser method), 31, 43  
 process\_defaults() (intermol.gromacs.gromacs\_parser.GromacsParser method), 31, 43  
 process\_dihedral() (intermol.gromacs.gromacs\_parser.GromacsParser method), 31, 43  
 process\_dihedraltype() (intermol.gromacs.gromacs\_parser.GromacsParser method), 31, 43  
 process\_exclusion() (intermol.gromacs.gromacs\_parser.GromacsParser method), 31, 43  
 process\_file() (intermol.gromacs.gromacs\_parser.GromacsParser method), 31, 43  
 process\_forcetype() (intermol.gromacs.gromacs\_parser.GromacsParser method), 31, 43  
 process\_implicittype() (intermol.gromacs.gromacs\_parser.GromacsParser method), 31, 43  
 process\_line() (intermol.gromacs.gromacs\_parser.GromacsParser method), 31, 43  
 process\_molecule() (intermol.gromacs.gromacs\_parser.GromacsParser method), 32, 43  
 process\_moleculetype() (intermol.gromacs.gromacs\_parser.GromacsParser method), 32, 43  
 process\_nonbond\_params() (intermol.gromacs.gromacs\_parser.GromacsParser method), 32, 43  
 process\_pair() (intermol.gromacs.gromacs\_parser.GromacsParser method), 32, 43  
 process\_pairtype() (intermol.gromacs.gromacs\_parser.GromacsParser method), 32, 43  
 process\_settle() (intermol.gromacs.gromacs\_parser.GromacsParser method), 32, 43

**P**  
 ProperPeriodicDihedral (class in intermol.forces.proper\_periodic\_dihedral\_type), 21, 67  
 ProperPeriodicDihedralType (class in intermol.forces.proper\_periodic\_dihedral\_type), 22, 67  
 ptype (intermol.atom.Atom attribute), 37  
 ptype (intermol.forces.abstract\_atom\_type.AbstractAtomType attribute), 6, 50

**Q**  
 qi (intermol.forces.ljq\_c\_pair\_type.LjqCPairType attribute), 19, 64  
 qi (intermol.forces.ljq\_sigeps\_pair\_type.LjqSigepsPairType attribute), 20, 65  
 qj (intermol.forces.ljq\_c\_pair\_type.LjqCPairType attribute), 19, 64  
 qj (intermol.forces.ljq\_sigeps\_pair\_type.LjqSigepsPairType attribute), 20, 65  
 QuarticAngle (class in intermol.forces.quartic\_angle\_type), 22, 68  
 QuarticAngleType (class in intermol.forces.quartic\_angle\_type), 22, 68  
 QuarticBond (class in intermol.forces.quartic\_bond\_type), 22, 68  
 QuarticBondType (class in intermol.forces.quartic\_bond\_type), 23, 68  
 QuarticBreakableBond (class in intermol.forces.quartic\_breakable\_bond\_type), 23, 69  
 QuarticBreakableBondType (class in intermol.forces.quartic\_breakable\_bond\_type), 23, 69

**R**  
 r (intermol.forces.urey\_bradley\_angle\_type.UreyBradleyAngleType attribute), 28, 74  
 r0 (intermol.forces.nonlinear\_bond\_type.NonlinearBondType attribute), 21, 66  
 r1 (intermol.forces.cross\_bond\_angle\_angle\_type.CrossBondAngleAngleType attribute), 10, 54  
 r1 (intermol.forces.cross\_bond\_bond\_angle\_type.CrossBondBondAngleType attribute), 11, 55  
 r2 (intermol.forces.cross\_bond\_angle\_angle\_type.CrossBondAngleAngleType attribute), 10, 54  
 r2 (intermol.forces.cross\_bond\_bond\_angle\_type.CrossBondBondAngleType attribute), 11, 55  
 r3 (intermol.forces.cross\_bond\_angle\_angle\_type.CrossBondAngleAngleType attribute), 10, 54  
 RbDihedral (class in intermol.forces.rb\_dihedral\_type), 24, 70  
 RbDihedralType (class in intermol.forces.rb\_dihedral\_type), 24, 70

Rc (intermol.forces.quartic\_breakable\_bond\_type.QuarticBreakableBondType attribute), 24, 69

read() (intermol.gromacs.grofile\_parser.GromacsGroParser method), 28, 40

read() (intermol.gromacs.gromacs\_parser.GromacsParser method), 32, 43

read() (intermol.lammps.lammps\_parser.LammpsParser method), 35, 47

read\_data() (intermol.lammps.lammps\_parser.LammpsParser method), 35, 47

read\_file() (in module intermol.gromacs.gromacs\_driver), 29, 40

read\_file() (in module intermol.lammps.lammps\_driver), 33, 45

read\_input() (intermol.lammps.lammps\_parser.LammpsParser method), 35, 47

residue\_index (intermol.atom.Atom attribute), 37

residue\_name (intermol.atom.Atom attribute), 37

returns() (in module intermol.decorators), 38

## S

SCALE\_FROM (intermol.lammps.lammps\_parser.LammpsParser attribute), 33, 45

SCALE\_INT0 (intermol.lammps.lammps\_parser.LammpsParser attribute), 34, 45

scaleLJ (intermol.forces.abstract\_pair\_type.AbstractPairType attribute), 7, 51

scaleLJ (intermol.forces.lj\_c\_pair\_type.LjCPairType attribute), 17, 62

scaleLJ (intermol.forces.lj\_default\_pair\_type.LjDefaultPairType attribute), 18, 62

scaleLJ (intermol.forces.lj\_sigeps\_pair\_type.LjSigepsPairType attribute), 19, 63

scaleLJ (intermol.forces.ljq\_c\_pair\_type.LjqCPairType attribute), 19, 64

scaleLJ (intermol.forces.ljq\_default\_pair\_type.LjqDefaultPairType attribute), 19, 64

scaleLJ (intermol.forces.ljq\_sigeps\_pair\_type.LjqSigepsPairType attribute), 20, 65

scaleQQ (intermol.forces.abstract\_pair\_type.AbstractPairType attribute), 7, 51

scaleQQ (intermol.forces.lj\_c\_pair\_type.LjCPairType attribute), 17, 62

scaleQQ (intermol.forces.lj\_default\_pair\_type.LjDefaultPairType attribute), 18, 62

scaleQQ (intermol.forces.lj\_sigeps\_pair\_type.LjSigepsPairType attribute), 19, 63

scaleQQ (intermol.forces.ljq\_c\_pair\_type.LjqCPairType attribute), 19, 64

scaleQQ (intermol.forces.ljq\_default\_pair\_type.LjqDefaultPairType attribute), 19, 64

scaleQQ (intermol.forces.ljq\_sigeps\_pair\_type.LjqSigepsPairType attribute), 20, 65

Settles (class in intermol.forces.settles), 24, 70

sigma (intermol.atom.Atom attribute), 37

sigma (intermol.forces.fene\_expandable\_bond\_type.FeneExpandableBondType attribute), 12, 56

sigma (intermol.forces.lj\_sigeps\_nonbonded\_type.LjSigepsNonbondedType attribute), 18, 63

sigma (intermol.forces.lj\_sigeps\_pair\_type.LjSigepsPairType attribute), 19, 63

sigma (intermol.forces.ljq\_sigeps\_pair\_type.LjqSigepsPairType attribute), 20, 65

specify() (in module intermol.forces.forcefunctions), 13, 57

summarize\_energy\_results() (in module intermol.convert), 37

System (class in intermol.system), 39

## T

theta (intermol.forces.cosine\_squared\_angle\_type.CosineSquaredAngleType attribute), 9, 53

theta (intermol.forces.harmonic\_angle\_type.HarmonicAngleType attribute), 15, 59

theta (intermol.forces.quartic\_angle\_type.QuarticAngleType attribute), 22, 68

theta (intermol.forces.three\_fad\_virtual\_type.ThreeFadVirtualType attribute), 25, 71

theta (intermol.forces.urey\_bradley\_angle\_type.UreyBradleyAngleType attribute), 28, 74

ThreeFadVirtual (class in intermol.forces.three\_fad\_virtual\_type), 25, 71

ThreeFadVirtualType (class in intermol.forces.three\_fad\_virtual\_type), 25, 71

ThreeFdVirtual (class in intermol.forces.three\_fd\_virtual\_type), 25, 71

ThreeFdVirtualType (class in intermol.forces.three\_fd\_virtual\_type), 25, 71

ThreeLinearVirtual (class in intermol.forces.three\_linear\_virtual\_type), 25, 72

ThreeLinearVirtualType (class in intermol.forces.three\_linear\_virtual\_type), 26, 72

ThreeOutVirtual (class in intermol.forces.three\_out\_virtual\_type), 26, 72

ThreeOutVirtualType (class in intermol.forces.three\_out\_virtual\_type), 26, 72

too\_few\_fields() (intermol.gromacs.gromacs\_parser.GromacsParser method), 32, 43

TrigDihedral (class in intermol.forces.trig\_dihedral\_type), 27, 73

TrigDihedralType (class in intermol.forces.trig\_dihedral\_type), 27, 73

- TwoVirtual (class in intermol.forces.two\_virtual\_type), 27, 74
- TwoVirtualType (class in intermol.forces.two\_virtual\_type), 27, 74
- type (intermol.forces.abstract\_nonbonded\_type.AbstractNonbondedType attribute), 7, 51
- type (intermol.forces.buckingham\_nonbonded\_type.BuckinghamNonbondedType attribute), 8, 52
- type (intermol.forces.lj\_c\_nonbonded\_type.LjCNonbondedType attribute), 17, 61
- type (intermol.forces.lj\_sigeps\_nonbonded\_type.LjSigepsNonbondedType attribute), 18, 63
- type\_parameters\_are\_unique() (intermol.gromacs.gromacs\_parser.GromacsParser static method), 32, 43
- U**
- U0 (intermol.forces.quartic\_breakable\_bond\_type.QuarticBreakableBondType attribute), 24, 69
- UnitsException, 38
- unitvars (intermol.gromacs.gromacs\_parser.GromacsParser attribute), 32, 43
- UreyBradleyAngle (class in intermol.forces.urey\_bradley\_angle\_type), 28, 74
- UreyBradleyAngleType (class in intermol.forces.urey\_bradley\_angle\_type), 28, 74
- V**
- ValueException, 38
- velocity (intermol.atom.Atom attribute), 37
- W**
- weight (intermol.forces.proper\_periodic\_dihedral\_type.ProperPeriodicDihedralType attribute), 22, 67
- write() (intermol.gromacs.grofile\_parser.GromacsGroParser method), 28, 40
- write() (intermol.gromacs.gromacs\_parser.GromacsParser method), 32, 44
- write() (intermol.lammps.lammps\_parser.LammpsParser method), 36, 47
- write\_angles() (intermol.gromacs.gromacs\_parser.GromacsParser method), 32, 44
- write\_angles() (intermol.lammps.lammps\_parser.LammpsParser method), 36, 47
- write\_atoms() (intermol.gromacs.gromacs\_parser.GromacsParser method), 32, 44
- write\_atomtypes() (intermol.gromacs.gromacs\_parser.GromacsParser method), 32, 44
- write\_bonds() (intermol.gromacs.gromacs\_parser.GromacsParser method), 32, 44
- write\_bonds() (intermol.lammps.lammps\_parser.LammpsParser method), 36, 47
- write\_defaults() (intermol.gromacs.gromacs\_parser.GromacsParser method), 32, 44
- write\_defaults() (intermol.gromacs.gromacs\_parser.GromacsParser method), 32, 44
- write\_dihedrals() (intermol.lammps.lammps\_parser.LammpsParser method), 36, 47
- write\_dihedrals() (intermol.lammps.lammps\_parser.LammpsParser method), 36, 47
- write\_impropers() (intermol.lammps.lammps\_parser.LammpsParser method), 36, 47
- write\_lammps() (in module intermol.lammps.lammps\_parser), 36, 48
- write\_molecules() (intermol.gromacs.gromacs\_parser.GromacsParser method), 32, 44
- write\_moleculetypes() (intermol.gromacs.gromacs\_parser.GromacsParser method), 32, 44
- write\_nonbonded\_types() (intermol.gromacs.gromacs\_parser.GromacsParser method), 32, 44
- write\_pairs() (intermol.gromacs.gromacs\_parser.GromacsParser method), 32, 44
- write\_settles() (intermol.gromacs.gromacs\_parser.GromacsParser method), 32, 44
- write\_system() (intermol.gromacs.gromacs\_parser.GromacsParser method), 32, 44
- write\_virtuals() (intermol.lammps.lammps\_parser.LammpsParser method), 36, 47
- X**
- xi (intermol.forces.improper\_harmonic\_dihedral\_type.ImproperHarmonicDihedralType attribute), 17, 61